Simulation software for safe, sustainable and future deltas

Delft3D FM Suite 1D2D

Deltares systems

D-Flow Flexible Mesh

User Manual

Deltares

D-Flow Flexible Mesh

Computational Cores and User Interface

User Manual

Released for:
Delft3D FM Suite 1D2D 2023

Version: 2023
Revision: 79016

17 April 2024

D-Flow Flexible Mesh, User Manual

Published and printed by:
Deltares
Boussinesqweg 1
2629 HV Delft
P.O. 177
2600 MH Delft
The Netherlands

telephone: +31 88 335 82 73
e-mail: Information
www: Deltares

For sales contact:
telephone: +31 88 335 81 88
e-mail: Sales
www: Sales & Support

For support contact:
telephone: +31 88 335 81 00
e-mail: Support
www: Sales & Support

Contents

List of Tables

List of Figures

List of To Do’s

1 A guide to this manual

1.1 Introduction

This User Manual describes the hydrodynamic module D-Flow Flexible Mesh (D-Flow FM) which is part of the Delft3D Flexible Mesh Suite or D-HYDRO Suite.

This module is part of several Modelling suites, released by Deltares as Deltares Systems or Dutch Delta Systems. These modelling suites are build with use of the Delta Shell framework. The framework enables to develop a range of modeling suites, each distinguished by the components and — most significantly — the (numerical) modules, which are plugged in. The modules which are compliant with the Delta Shell framework are released as D-*Name of the module*, for example: D-Flow Flexible Mesh, D-Waves, D-Water Quality, D-Real Time Control and D-Rainfall Runoff.

Therefore, this User Manual is shipped with several modelling suites. On the *Start Page* links are provided to all relevant User Manuals (and Technical Reference Manuals) for that modelling suite. Other user manuals can be referenced. In that case, you need to open the specific user manual from the *Start Page* in the central window. Some texts are shared in different user manuals, in order to improve the readability.

1.2 Overview

To make this manual more accessible we will briefly describe the contents of each chapter.

If this is your first time to start working with D-Flow FM we suggest you to read Chapter 3, Getting started and practice the tutorial of Chapter 4. These chapters explain the user interface and guide you through the modelling process resulting in your first simulation.

Chapter 2: Introduction to D-Flow Flexible Mesh, provides specifications of D-Flow FM, such as the areas of application, the standard and specific features provided, coupling to other modules and utilities.

Chapter 3: Getting started, gives an overview of the basic features of the D-Flow FM GUI and will guide you through the main steps to set up a basic D-Flow FM model.

Chapter 4: Tutorial, gives you hands-on experience in using the D-Flow FM GUI to define the input of a simple problem, in validating this input, in executing the simulation and in inspecting the results.

Chapter 5: All about the modelling process, provides practical information on the GUI, setting up a model with all its parameters and tuning the model.

Chapter 6: Running a model, discusses how to validate and execute a model run. Either in the GUI, or in batch mode and/or in parallel using MPI. It also provides some information on run times and file sizes.

Chapter 7: Visualize results, explains in short the visualization of results within the GUI. It introduces the programs Quickplot and Muppet to visualize or animate the simulation results, and Matlab for general post-processing.

Chapter 8: Hydrodynamics, gives some background information on the conceptual model of the D-Flow FM module.

Chapter 9: Transport of matter, discusses the modeled transport processes, their governing equations, boundary and initial conditions and user-relevant numerical and physical settings.

??: **??** provides a detailed insight into the modelling of stratified conditions for salinity, temperature, sediments, tracers or water quality substances.

Chapter 10: Heat transport, provides a detailed insight into (the modelling of) heat transport.

Chapter 11: Wind, gives background information of how wind fields should be imposed, the relevant definitions and the supported file formats.

Chapter 14: Hydraulic structures, gives background information of the available hydraulic structures in D-Flow FM, the relevant definitions and the supported file formats.

??: **??**, provides guidance on the integrated modelling of hydrodynamics (D-Flow FM) and waves (D-Waves).

Chapter 17: Coupling with D-RTC (FBC-Tools), provides guidance on the integrated modelling of hydrodynamics (D-Flow FM) and real time control of hydraulic structures (D-RTC).

Chapter 19: Coupling with D-Water Quality (Delwaq), provides guidance on the integrated modelling of hydrodynamics (D-Flow FM) and water quality (D-Water Quality).

Chapter 22: Calibration and data assimilation, describes how the OpenDA toolbox could be deployed to apply calibration and data assimilation.

1.3 Typographical conventions

Throughout this manual, the following conventions help you to distinguish between different elements of text.

Example	**Description**
Module **Project**	Title of a window or a sub-window are in given in **bold**. Sub-windows are displayed in the **Module** window and cannot be moved. Windows can be moved independently from the **Module** window, such as the **Visualisation Area** window.
Save	Item from a menu, title of a push button or the name of a user interface input field. Upon selecting this item (click or in some cases double click with the left mouse button on it) a related action will be executed; in most cases it will result in displaying some other (sub-)window. In case of an input field you are supposed to enter input data of the required format and in the required domain.
<\tutorial\wave\swan-curvi> <siu.mdw>	Directory names, filenames, and path names are expressed between angle brackets, <>. For Linux environments a forward slash (/) is used instead of the backward slash (\) for Windows environments.

Example	Description
“27 08 1999”	Data to be typed by you into the input fields are displayed between double quotes. Selections of menu items, option boxes etc. are described as such: for instance ‘select *Save* and go to the next window’.
`delft3d-menu`	Commands to be typed by you are given in the font Courier New, 10 points.
➢	In this User manual, user actions are indicated with this arrow.
[m s^{-1}] [–]	Units are given between square brackets when used next to the formulae. Leaving them out might result in misinterpretation. Most units will be in SI notation. [m AD] stands for ’meter Above Datum’, which denotes a level relative to the vertical reference system in the model.

Command prompts and terminal output are shown in framed boxes with typewriter font:

```
> ./dflowfm --version
Deltares, D-Flow FM Version 1.1.149.41663, Sep 02 2015, 10:40:42
Compiled with support for:
IntGUI: no
OpenGL: no
OpenMP: yes
MPI   : yes
PETSc : yes
METIS : yes
```

1.4 Changes with respect to previous versions

Several descriptions of β-functionality are added or marked as β-functionality. Screencasts are updated. Chapter 4: Tutorial is restructured and a tutorial for Morphology is added.

2 Introduction to D-Flow Flexible Mesh

D-Flow Flexible Mesh (D-Flow FM) is a hydrodynamic simulation program developed by Deltares. It is part of Deltares' unique, fully integrated computer software suite for a multi-disciplinary approach and 1D, 2D and 3D computations for coastal, river and estuarine areas, named Delft3D Flexible Mesh Suite or D-HYDRO Suite. It can carry out simulations of hydrodynamic flow, waves, water quality and ecology.

It has been designed for experts and non-experts alike. The Delft3D Flexible Mesh Suite is composed of several modules, grouped around a mutual interface, while being capable to interact with one another. D-Flow FM, which this manual is about, is one of these modules. D-Flow FM is a multi-dimensional (1D, 2D and 3D) hydrodynamic (and transport) simulation program which calculates non-steady flow and transport phenomena that result from tidal and meteorological forcing on structured and unstructured, boundary fitted grids. The term Flexible Mesh in the name refers to the flexible combination of unstructured grids consisting of triangles, quadrangles, pentagons and hexagons. In 3D simulations the vertical grid is using the σ co-ordinate approach. As an alternative a fixed z layers approach is also possible. The 2D functionality in D-Flow FM has been released, while the functionality for 3D and 1D is in development.

2.1 Areas of application

- Tide and wind-driven flows (i.e., storm surges).
- Stratified and density driven flows.
- River flow simulations.
- Rural channel networks.
- Rainfall runoff in urban environments.
- Simulation of tsunamis, hydraulic jumps, bores and flood waves.
- Fresh-water river discharges in bays.
- Salt intrusion.
- Cooling water intakes and waste water outlets.
- Transport of dissolved material and pollutants.

2.2 Standard features

- Tidal forcing.
- The effect of the Earth's rotation (Coriolis force).
- Density driven flows (pressure gradients terms in the momentum equations).
- Advection-diffusion solver included to compute density gradients.
- Space and time varying wind and atmospheric pressure.
- Advanced turbulence models to account for the vertical turbulent viscosity and diffusivity based on the eddy viscosity concept. Four options are provided:

  1. constant,
  2. algebraic,
  3. k-ε and
  4. k-τ model

  .
- Time varying sources and sinks (e.g., river discharges).
- Simulation of the thermal discharge, effluent discharge and the intake of cooling water at any location and any depth.
- Robust simulation of drying and flooding of inter-tidal flats and river winter beds.

2.3 Special features

- Built-in automatic switch converting 2D bottom-stress coefficient to 3D coefficient.
- Built-in anti-creep correction to suppress artificial vertical diffusion and artificial flow due to σ-grids.
- Heat exchange through the free water surface.
- Wave induced stresses and mass fluxes.
- Influence of waves on the bed shear stress.
- Optional facility to calculate the intensity of the spiral motion phenomenon in the flow (e.g., in river bends) which is especially important in sedimentation and erosion studies (for depth averaged — 2DH — computations only).
- Non-linear iterations in the solver can be enabled for accurate flooding results.
- Optional facility for tidal analysis of output parameters.
- Optional facility for special structures such as pumping stations, bridges, fixed weirs and controllable barriers (1D, 2D and 3D)
- Default advection scheme suitable for various flow regimes, from bore propagation to eddy shedding.
- Domain partitioning for parallelized runs on MPI-based High Performance Computing clusters.

2.4 Important differences compared to Delft3D-FLOW

The most noticeable difference between Delft3D-FLOW and D-Flow FM is the use of unstructured grids. Large regions with quadrangles can be coupled with much greater freedom than before, using triangles, pentagons and hexagons. Grid refinement (and coarsening) without DD-coupling is now possible in one and the same model grid. In future, 1D networks will be coupled to 2D grids, either adjacent to each other or the 1D network overlying the 2D grid. Finally, many of Delft3D-FLOW's grid restrictions are now gone: since there are no true grid 'rows' and 'columns' anymore, rows of grid cells may be coupled to columns, in any direction and at any position.

In addition to the unstructured grid files, all geometric model input is now specified in geographical coordinates, either in Cartesian or spherical coordinates (x, y or longitude, latitude). This is different from Delft3D-FLOW which required model input in grid indices. This so-called model-independent coordinates input allows for easy change of a model grid, after which the remaining model input can remain the same.

The D-Flow FM Graphical User Interface provides a much more powerful and integrated environment for setting up D-Flow FM models and inspecting model input such as time-dependent forcings (boundary conditions and barrier control). Another improvement within the User Interface is the use of scripting for running and live interaction with a model.

Coupled running of D-Flow FM with other modules has been extended with real time control of hydraulic structures, as listed in the following section.

Like Delft3D-FLOW, D-Flow FM implements a finite volume solver on a staggered grid. However, since there is no concept of grid 'rows' and 'columns', there is also no ADI-solver possible. The continuity equation is solved implicitly for all points in a single combined system. Time integration is done explicitly for part of the advection term, and the resulting dynamic time-step limitation is automatically set based on the Courant criterium. The possible performance penalty that may result from this approach can often be remedied by refining and coarsening the computational grid at the right locations.

In D-Flow FM, the advection scheme is suitable for both sub-critical and critical flows. The scheme is 'shock proof', is capable of reproducing correct bore propagation velocities.

2.5 Coupling to other modules

The hydrodynamic conditions (velocities, water elevations, density, salinity, vertical eddy viscosity and vertical eddy diffusivity) calculated in the D-Flow FM module are used as input to the other modules of the Delft3D Flexible Mesh Suite, which are:

Module	**Description**
D-Morphology	cohesive and non-cohesive sediment transport
D-Real Time Control	flow-triggered control of hydrodynamic structures
D-Water Quality (Delwaq)	far-field water quality, see also chapter 19
D-Waves (SWAN)	short wave propagation, see also **??**

For using D-Flow FM the following utilities are important:

Module	**Description**
User Interface	for complete model set-up and model runs, see chapter 3 and chapter 5
RGFGRID	for generating curvilinear and unstructured grids
Delft3D-QUICKPLOT	for visualisation and animation of simulation results
OpenEarthTools	set of MATLAB scripts for post-processing of output files, see http://www.openearth.eu
DFMOUTPUT	◇ for merging partitioned map files into one, see section 6.4.4. ◇ output of derived properties, such as the maximum value of a time series, see section F.5.

For details on using these utility programs you are referred to the respective User Manuals.

2.6 Installation of D-Flow Flexible Mesh

D-Flow FM consists of a set of Computational Cores and a User Interface. The User Interface is only available for Windows operating systems. The Computational Cores are available both on Windows and Linux operating systems.

On Windows, D-Flow FM is available as msi-file. Double-click it to start the installation and follow the instructions. Both the Computational Cores and the User Interface will be installed. Start the application from *Start* →*All Programs* →*Deltares* or by double-clicking the short-cut on your desktop.

On Linux, the Computational Cores of D-Flow FM are available via an rpm-file (2021 versions and newer) or a tar.gz-file (2020 versions and older). See the installation manual for more details.

A Docker version is available upon request. Please contact Deltares if needed.

3 Getting started

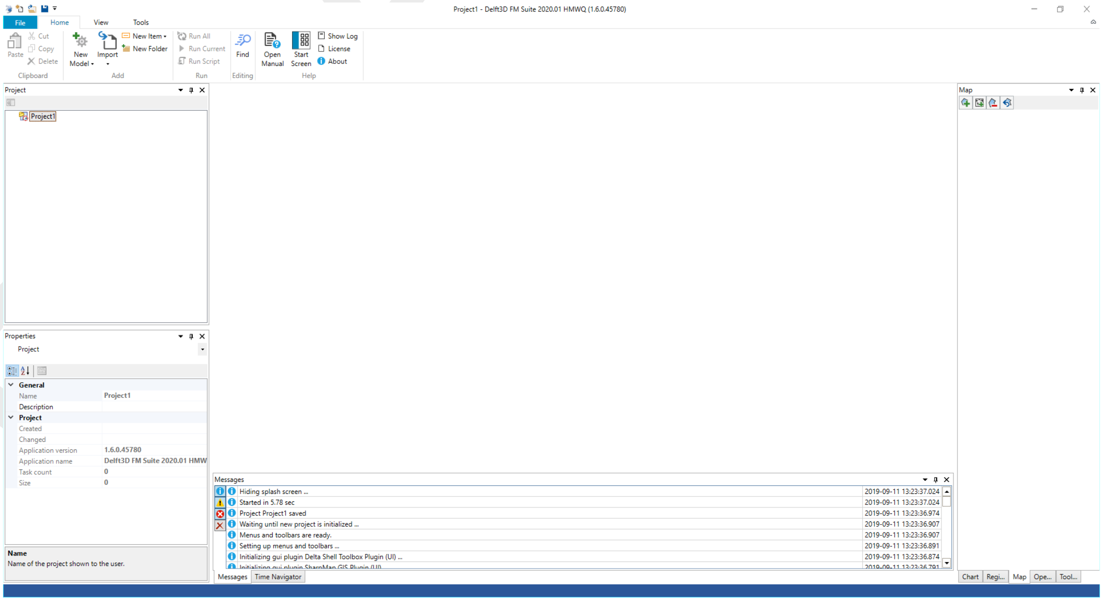
3.1 Introduction

This chapter gives an overview of the basic features of the D-Flow FM User Interface and will guide you through the main steps to set up a D-Flow FM model. For a more detailed description of the GUI features you are referred to chapter 5. For technical documentation you are referred to Deltares (2024a).

3.2 Overview of D-Flow FM GUI

When you start the application for the first time the lay-out might look like Figure 3.1. The basic lay-out consists of the following items:

- *Ribbon* - top
- **Project** window - up left
- The central window, containing the *Start Page*
- **Messages** and **Time Navigator** window - down centre
- **Toolbox**, **Chart**, **Region**, **Map** and **Operations** window - to the right
- **Properties** window - down left



Properties

Project

General	
Name	Project1
Description	
Project	
Created	
Changed	
Application version	1.6.0.45780
Application name	Delft3D FM Suite 2020.01 HMW
Task count	0
Size	0

Name
Name of the project shown to the user.

Messages

```
Hiding splash screen ...                                    2019-09-11 13:23:37.024
Started in 5.78 sec                                         2019-09-11 13:23:37.024
Project Project1 saved                                      2019-09-11 13:23:36.974
Waiting until new project is initialized ...                2019-09-11 13:23:36.907
Menus and toolbars are ready.                               2019-09-11 13:23:36.907
Setting up menus and toolbars ...                           2019-09-11 13:23:36.891
Initializing gui plugin Delta Shell Toolbox Plugin (UI) ... 2019-09-11 13:23:36.874
Initializing gui plugin SharpMap GIS Plugin (UI)            2019-09-11 13:23:36.791
```

Messages | Time Navigator

Figure 3.1: *Start-up lay-out D-Flow FM User Interface*

All the windows can be customized/hidden according to your own preferences.

These settings will be automatically saved for the next time you start the application.

The most important windows for the D-Flow FM plugin are the **Project**, **Map**, **Messages**, **Time Navigator** and **Properties** windows. The central window displays the **Start Page**. Here the *Map view* or specific editors will be displayed.

The contents of these windows are briefly discussed in the subsections below.

3.2.1 Project window

After adding or importing a D-Flow FM model (see section 3.5.1 and section 3.5.2), the *Project* window will be extended with D-Flow FM specific features (see Figure 3.2). The *Project* window provides you with the basic steps to set up a D-Flow FM model.

The *Project* window consists of the following features:

General general model information such as depth layer specification, model coordinate system and angle of latitude and longitude

Area geographical (GIS based) features, such as observation points, structures, dry points and land boundaries

Grid computational grid

Bed Level model bed level

Time Frame model time frame and time step

Processes active physical processes in the model such as salinity, temperature, wind and tide generating forces

Initial Conditions initial conditions for water levels and other physical processes

Boundary Conditions model boundaries and boundary condition specification

Physical Parameters physical settings for processes such as roughness, viscosity, wind and temperature

Sources and Sinks location and time series specification for point sources and sinks

Numerical Parameters numerical simulation settings

Output Parameters output specification

Output output after running the simulation

Upon clicking the items in the *Project* window the corresponding tab (in case of non-geographic model settings), attribute table (in case of geographic model settings) or editor view (in case of advanced editing options) will open in the central window. Using the right mouse button gives options such as importing/exporting model data.

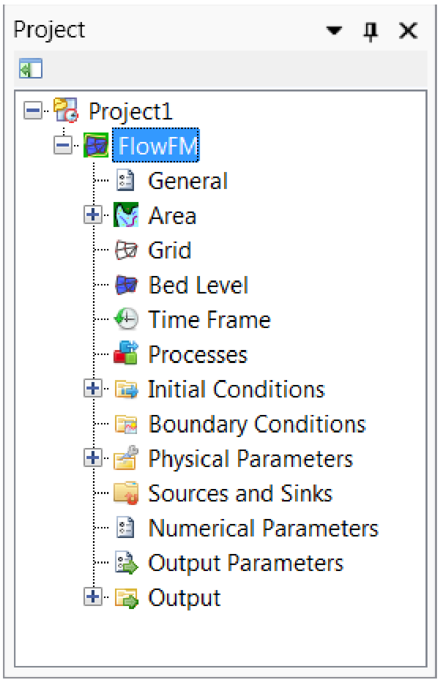


***Figure 3.2: Project** window of D-Flow FM plugin*

3.2.2 The central window

The central window shows the contents of the map or the editor you are working with. Multiple maps and editors may be docked at the central window location; each accessible as a tabs along the top (see Figure 3.3). The map is used to edit geographic model data, the editor for the overall model settings. Moreover, the contents of the central window can also be a specific editor such as the time point editor or the boundary condition editor. Each of these editors will open as a separate tab.

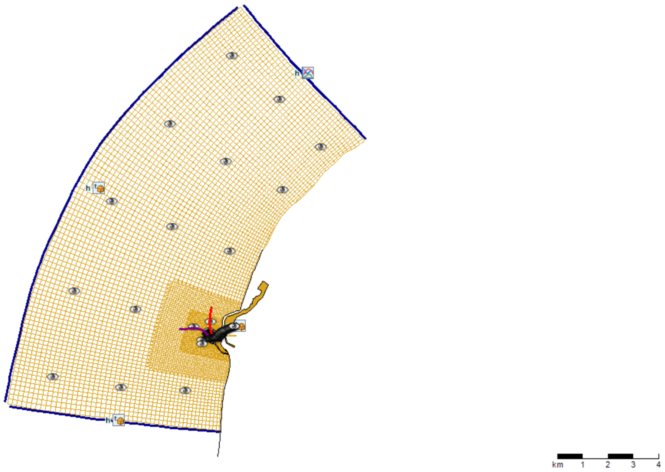
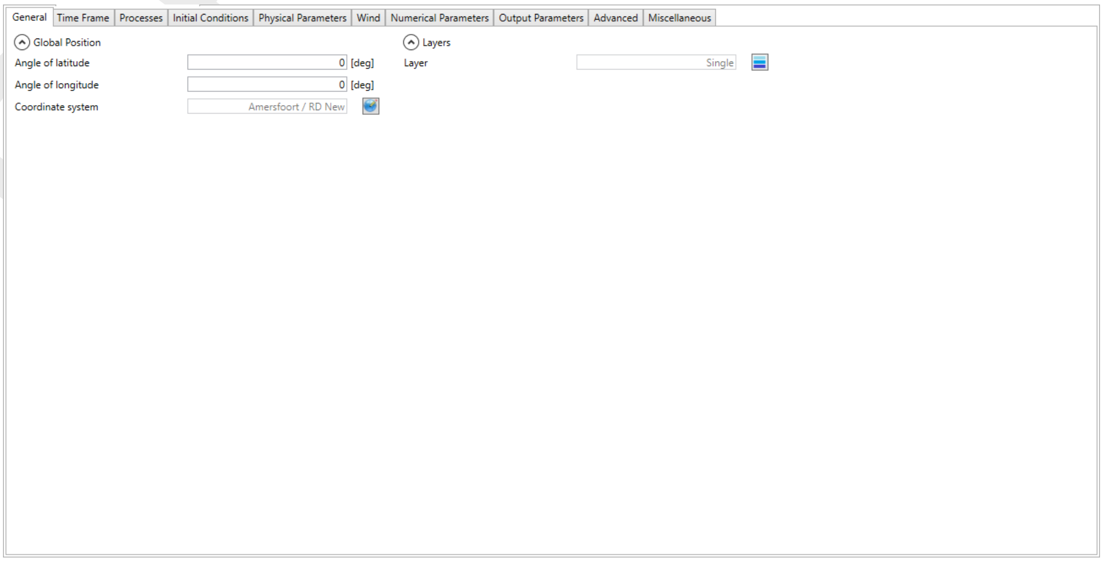


Figure 3.3: *The central window with a map and contents of the D-Flow FM model*

3.2.3 Settings window



Global Position

Angle of latitude 0 [deg]

Angle of longitude 0 [deg]

Coordinate system Amersfoort / RD New

Layers

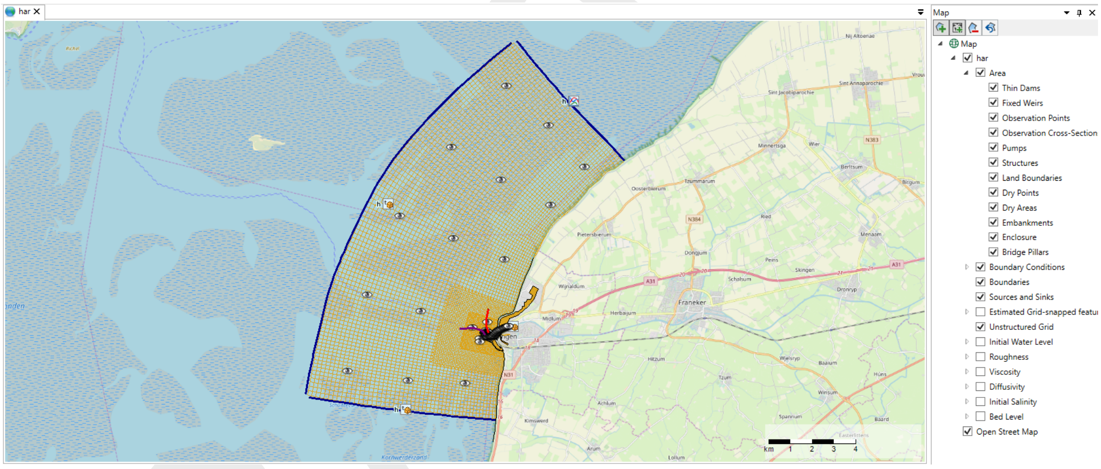
Layer Single

Figure 3.4: *The model **Settings** window with General settings tab enabled*

3.2.4 Map window

The **Map** window allows the user to control the visibility of the contents of the central map using checkboxes. Furthermore, the user can add (wms) layers, such as satellite imagery or open street maps (see Figure 3.5).

Note: Please note that the map usually has a different coordinate system than the model. In rendering the model attributes they are transformed to the map coordinate system (for visual inspection on the map), but the model will be saved in the model coordinate system.

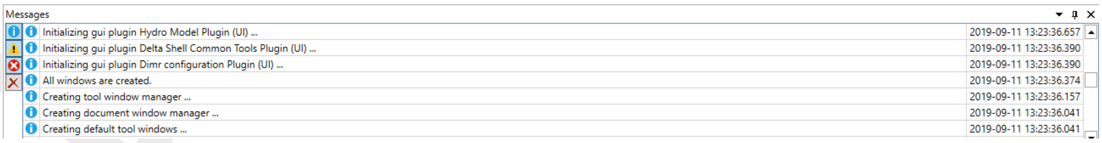


- Map
  - ☑ har
    - ☑ Area
      - ☑ Thin Dams
      - ☑ Fixed Weirs
      - ☑ Observation Points
      - ☑ Observation Cross-Section
      - ☑ Pumps
      - ☑ Structures
      - ☑ Land Boundaries
      - ☑ Dry Points
      - ☑ Dry Areas
      - ☑ Embankments
      - ☑ Enclosure
      - ☑ Bridge Pillars
    - ☑ Boundary Conditions
    - ☑ Boundaries
    - ☑ Sources and Sinks
    - □ Estimated Grid-snapped featu
    - ☑ Unstructured Grid
    - □ Initial Water Level
    - □ Roughness
    - □ Viscosity
    - □ Diffusivity
    - □ Initial Salinity
    - □ Bed Level
  - ☑ Open Street Map

***Figure 3.5:* Map** *window controlling map contents*

3.2.5 Messages window

The **Message** window (Figure 3.6) provides a log of information on the recent activities in the User Interface. It also provides warning and error messages.

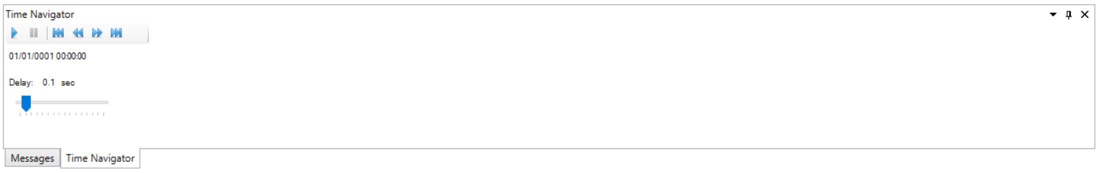


```
Messages
Initializing gui plugin Hydro Model Plugin (UI) ...             2019-09-11 13:23:36.657
Initializing gui plugin Delta Shell Common Tools Plugin (UI) ... 2019-09-11 13:23:36.390
Initializing gui plugin Dimr configuration Plugin (UI) ...       2019-09-11 13:23:36.390
All windows are created.                                         2019-09-11 13:23:36.374
Creating tool window manager ...                                 2019-09-11 13:23:36.157
Creating document window manager ...                             2019-09-11 13:23:36.041
Creating default tool windows ...                                2019-09-11 13:23:36.041
```

Figure 3.6: *Log of messages, warnings and errors in message window*

3.2.6 Time Navigator window

The **Time Navigator** (Figure 3.7) can be used to step through time dependent model output and other time dependent geographic features on the map.



01/01/0001 00:00:00

Delay: 0.1 sec

Figure 3.7: Time Navigator *window in D-Flow FM User Interface*

3.3 Dockable views

The User Interface offers lots of freedom to customize dockable views, which are discussed in this section.

3.3.1 Docking tabs separately

Within the User Interface the user can dock the separate windows according to personal preferences. These preferences are then saved for future use of the framework. An example of such preferences is presented in Figure 3.8, where windows have been docked on two screens.

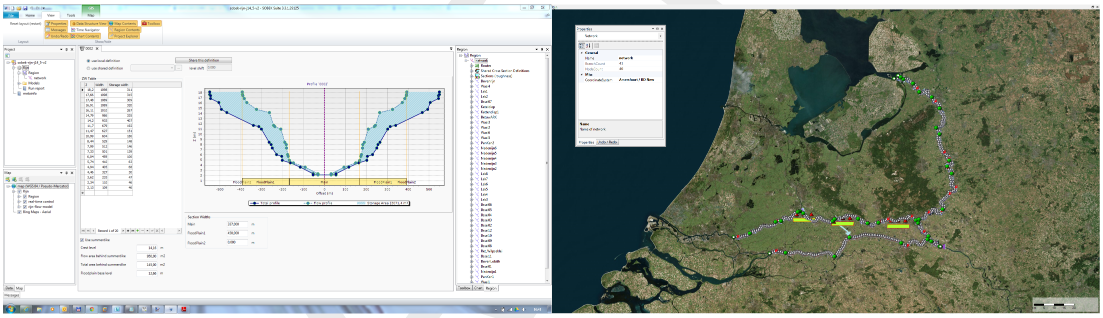
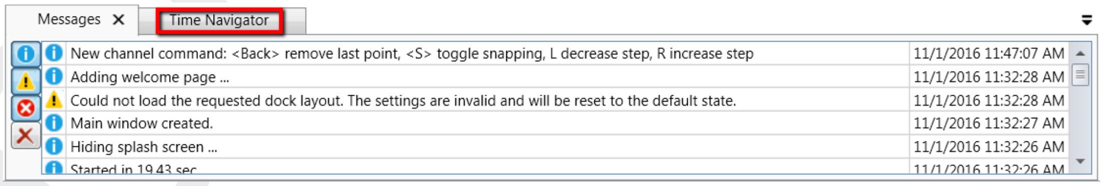


Figure 3.8: *Docking windows on two screens within the User Interface.*

3.3.2 Multiple tabs

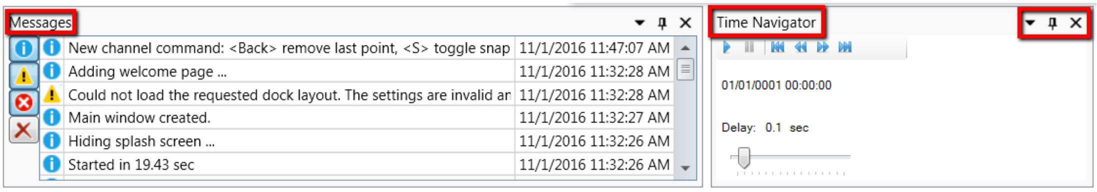
In case two windows are docked in one view, the underlying window (tab) can be brought to the front by simply selecting the tab, as is shown here.



New channel command: <Back> remove last point, <S> toggle snapping, L decrease step, R increase step	11/1/2016 11:47:07 AM
Adding welcome page ...	11/1/2016 11:32:28 AM
Could not load the requested dock layout. The settings are invalid and will be reset to the default state.	11/1/2016 11:32:28 AM
Main window created.	11/1/2016 11:32:27 AM
Hiding splash screen ...	11/1/2016 11:32:26 AM
Started in 19.43 sec	11/1/2016 11:32:26 AM

Figure 3.9: *Bringing the* ***Time Navigator*** *window to the front*

By dragging dockable windows with the left mouse button and dropping the window left, right, above or below another one the graphical user interface can be customized according to personal preferences. Here an example of the **Time Navigator** window being docked to the right of the **Messages** window.



New channel command: <Back> remove last point, <S> toggle snap	11/1/2016 11:47:07 AM
Adding welcome page ...	11/1/2016 11:32:28 AM
Could not load the requested dock layout. The settings are invalid ar	11/1/2016 11:32:28 AM
Main window created.	11/1/2016 11:32:27 AM
Hiding splash screen ...	11/1/2016 11:32:26 AM
Started in 19.43 sec	11/1/2016 11:32:26 AM

01/01/0001 00:00:00

Delay: 0.1 sec

Figure 3.10: *Docking the* ***Time Navigator*** *window.*

Additional features are the possibility to remove or (auto) hide the window (top right in Figure 3.10). In case of removal, the window can be retrieved by two left mouse-clicks on *Time Navigator* in the *View* ribbon. Hiding the **Properties** window results in:

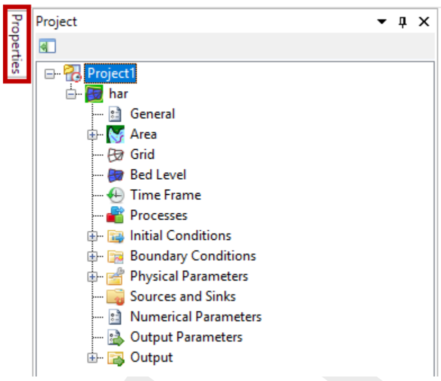


Figure 3.11: *Auto hide the* ***Properties*** *window*

3.4 Ribbons and toolbars

The user can access the toolbars arranged in *ribbons*. Model plug-ins can have their own model specific *ribbon*. The *ribbon* may be auto collapsed by activating the *Collapse the Ribbon* button when right-mouse-clicking on the *ribbon*.

3.4.1 Ribbons (shortcut keys)

The User Interface makes use of ribbons, just like Microsoft Office. You can use these ribbons for most of the operations. With the ribbons comes shortcut key functionality, providing shortcuts to perform operations. If you press `Alt`, you will see the letters and numbers to access the ribbons and the ribbon contents (i.e. operations). For example, `Alt + H` will lead you to the *Home*-ribbon (Figure 3.12).

Note: Implementation of the shortcut key functionality is still work in progress.

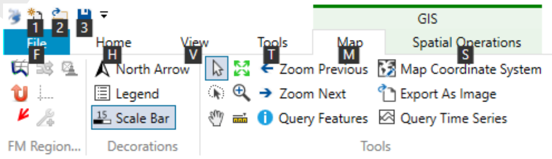
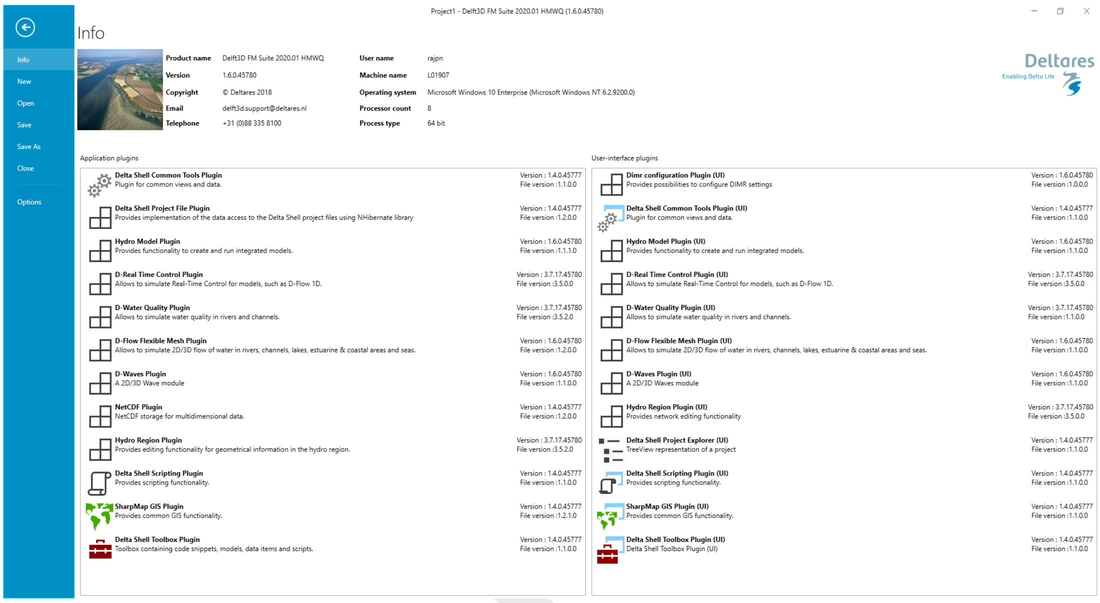


Figure 3.12: *Perform operations using the shortcut keys*

3.4.2 File

The left-most *ribbon* is the *File* ribbon. It has menu-items comparable to most Microsoft applications. Furthermore, it offers users save and open functionality, as well as the *Info* and *Options* dialogs, as shown in Figure 3.13 and Figure 3.14.



Project1 - Delft3D FM Suite 2020.01 HMWQ (1.6.0.45780)

Info

Product name	Delft3D FM Suite 2020.01 HMWQ	**User name**	rajpn
Version	1.6.0.45780	**Machine name**	L01907
Copyright	© Deltares 2018	**Operating system**	Microsoft Windows 10 Enterprise (Microsoft Windows NT 6.2.9200.0)
Email	delft3d.support@deltares.nl	**Processor count**	8
Telephone	+31 (0)88 335 8100	**Process type**	64 bit

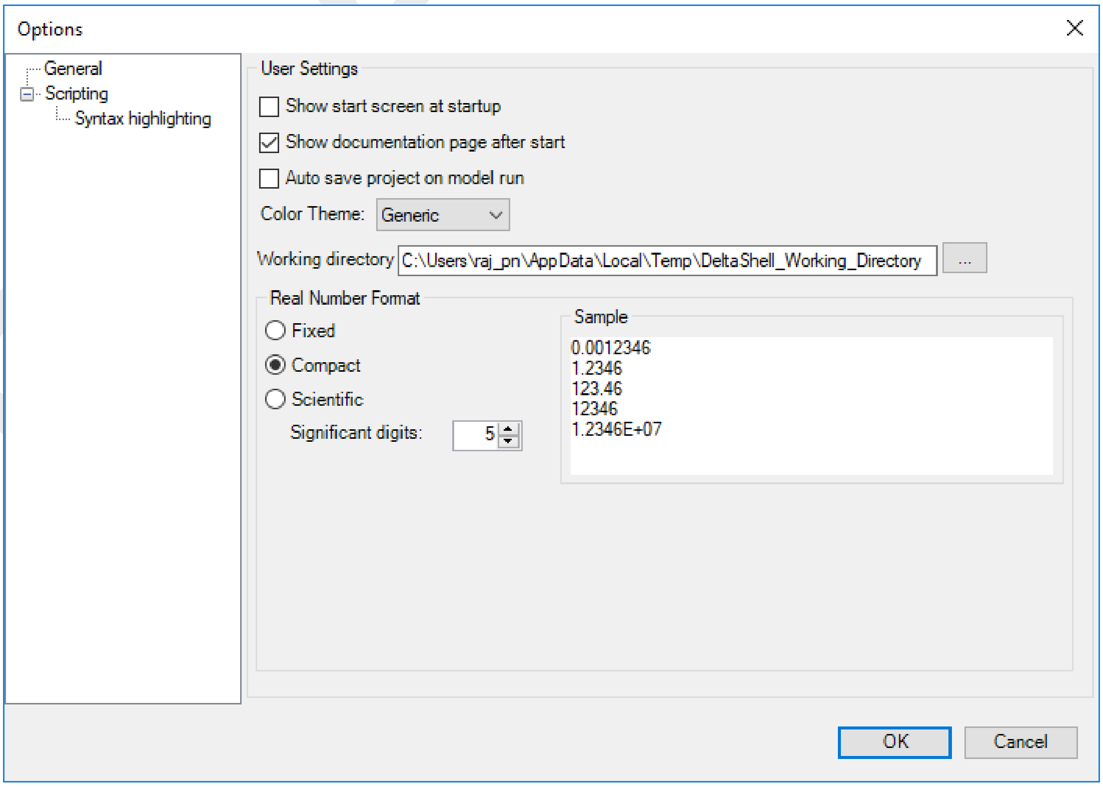
Application plugins

Delta Shell Common Tools Plugin Plugin for common views and data.	Version : 1.4.0.45777 File version :1.1.0.0
Delta Shell Project File Plugin Provides implementation of the data access to the Delta Shell project files using NHibernate library	Version : 1.4.0.45777 File version :1.2.0.0
Hydro Model Plugin Provides functionality to create and run integrated models.	Version : 1.6.0.45780 File version :1.1.1.0
D-Real Time Control Plugin Allows to simulate Real-Time Control for models, such as D-Flow 1D.	Version : 3.7.17.45780 File version :3.5.0.0
D-Water Quality Plugin Allows to simulate water quality in rivers and channels.	Version : 3.7.17.45780 File version :3.5.2.0
D-Flow Flexible Mesh Plugin Allows to simulate 2D/3D flow of water in rivers, channels, lakes, estuarine & coastal areas and seas.	Version : 1.6.0.45780 File version :1.2.0.0
D-Waves Plugin A 2D/3D Wave module	Version : 1.6.0.45780 File version :1.1.0.0
NetCDF Plugin NetCDF storage for multidimensional data.	Version : 1.4.0.45777 File version :1.2.0.0
Hydro Region Plugin Provides editing functionality for geometrical information in the hydro region.	Version : 3.7.17.45780 File version :3.5.2.0
Delta Shell Scripting Plugin Provides scripting functionality.	Version : 1.4.0.45777 File version :1.1.0.0
SharpMap GIS Plugin Provides common GIS functionality.	Version : 1.4.0.45777 File version :1.2.1.0
Delta Shell Toolbox Plugin Toolbox containing code snippets, models, data items and scripts.	Version : 1.4.0.45777 File version :1.1.0.0

User-interface plugins

Dimr configuration Plugin (UI) Provides possibilities to configure DIMR settings	Version : 1.6.0.45780 File version :1.0.0.0
Delta Shell Common Tools Plugin (UI) Plugin for common views and data.	Version : 1.4.0.45777 File version :1.1.0.0
Hydro Model Plugin (UI) Provides functionality to create and run integrated models.	Version : 1.6.0.45780 File version :1.1.0.0
D-Real Time Control Plugin (UI) Allows to simulate Real-Time Control for models, such as D-Flow 1D.	Version : 3.7.17.45780 File version :3.5.0.0
D-Water Quality Plugin (UI) Allows to simulate water quality in rivers and channels.	Version : 3.7.17.45780 File version :1.1.0.0
D-Flow Flexible Mesh Plugin (UI) Allows to simulate 2D/3D flow of water in rivers, channels, lakes, estuarine & coastal areas and seas.	Version : 1.6.0.45780 File version :1.1.0.0
D-Waves Plugin (UI) A 2D/3D Waves module	Version : 1.6.0.45780 File version :1.1.0.0
Hydro Region Plugin (UI) Provides network editing functionality	Version : 3.7.17.45780 File version :3.5.0.0
Delta Shell Project Explorer (UI) TreeView representation of a project	Version : 1.4.0.45777 File version :1.1.0.0
Delta Shell Scripting Plugin (UI) Provides scripting functionality.	Version : 1.4.0.45777 File version :1.1.0.0
SharpMap GIS Plugin (UI) Provides common GIS functionality.	Version : 1.4.0.45777 File version :1.1.0.0
Delta Shell Toolbox Plugin (UI) Delta Shell Toolbox Plugin (UI)	Version : 1.4.0.45777 File version :1.1.0.0

Figure 3.13: *The* File *ribbon.*



Options

User Settings

☐ Show start screen at startup

☑ Show documentation page after start

☐ Auto save project on model run

Color Theme: Generic

Working directory C:\Users\raj_pn\AppData\Local\Temp\DeltaShell_Working_Directory

Real Number Format

○ Fixed

● Compact

○ Scientific

Significant digits: 5

Sample

0.0012346
1.2346
123.46
12346
1.2346E+07

Figure 3.14: *The User Interface options dialog.*

3.4.3 Home

The second *ribbon* is the *Home* ribbon (Figure 3.15). It harbours some general features for clipboard actions, addition of items, running models, finding items within projects or views, and help functionality.

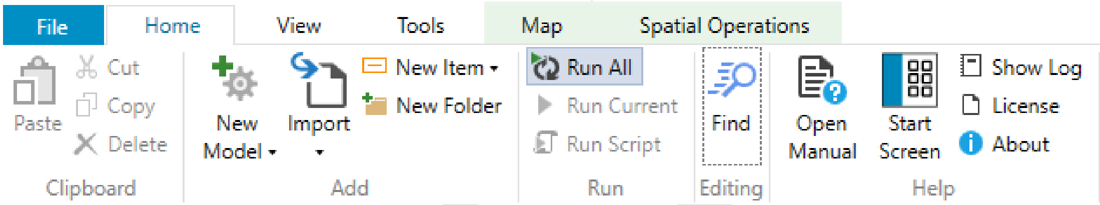


Figure 3.15: *The* Home *ribbon.*

3.4.4 View

The third *ribbon* is the *View* ribbon (Figure 3.16). Here, the user can show or hide windows.

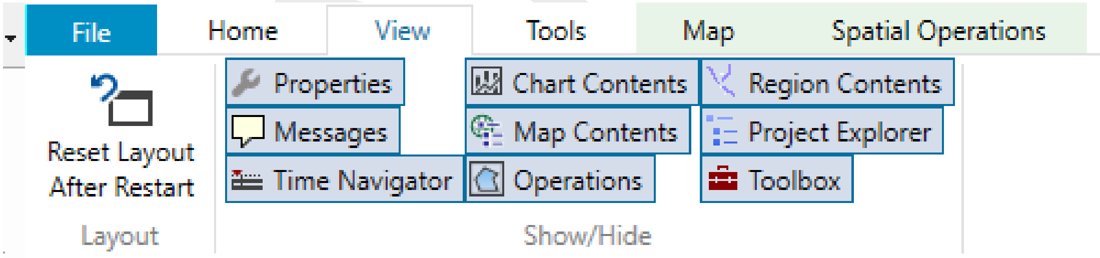


Figure 3.16: *The* View *ribbon.*

3.4.5 Tools

The fourth *ribbon* is the *Tools* ribbon (Figure 3.17). By default, it contains only the *Open Case Analysis View* tool. Some model plug-ins offer the installation of extra tools that may be installed. These are documented within the user documentation of those model plug-ins.

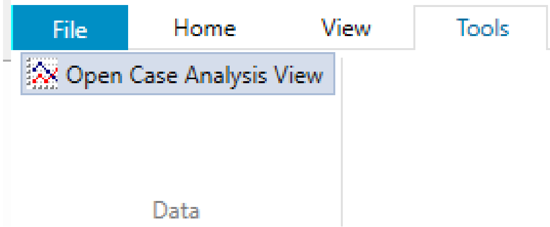


Figure 3.17: *The* Tools *ribbon contains just the* Data *item.*

3.4.6 Map

The last *ribbon* is the *Map* ribbon (Figure 3.18).

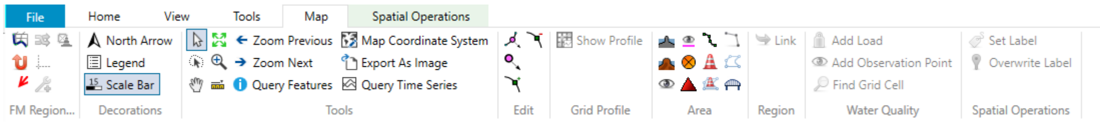


Figure 3.18: *The* Map *ribbon.*

This will be used heavily, while it harbours all *Geospatial* functions, like:

- *Decorations* for the map
  - North arrow
  - Scale bar
  - Legend
  - ...
- *Tools* to customize the map view
  - Select a single item
  - Select multiple items by drawing a curve
  - Pan
  - Zoom to Extents
  - Zoom by drawing a rectangle
  - Zoom to Measure distance
  - ...
- *Edit* polygons, for example within a network, basin, or waterbody
  - Move geometry point(s)
  - Add geometry point(s)
  - Remove geometry point(s)
- Addition of *Area* elements, such as
  - Add 2D Cross-section
  - Add 2D Structure
  - Add 2D Pump
  - ...

Still, all functions of the category can be activated as they will appear in the drop-down panel.

3.4.7 Scripting

When you open the scripting editor in the User Interface, a Scripting *ribbon* category will appear. This ribbon has the following additional options (see also Figure 3.19), which are described in Table 3.1:

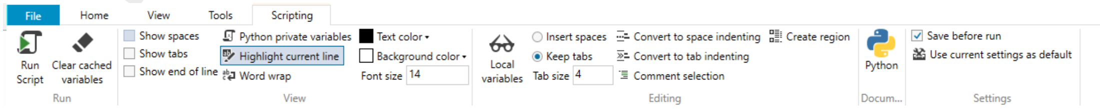


Figure 3.19: *The scripting* ribbon *within the User Interface.*

Table 3.1: *Functions and their descriptions within the scripting* ribbon *of the User Interface.*

Function	**Description**
Run script	Executes the selected text. If no text is selected then it will execute the entire script
Clear cached variables	Clears all variables and loaded libraries from memory

Table 3.1: *Functions and their descriptions within the scripting* ribbon *of the User Interface.*

Function	**Description**
Debugging	Enables/Disables the debug option. When enabled you can add breakpoint to the code (using `F9` or clicking in the margin) and the code will stop at this point before executing the statement (use F10 (step over) or F11 (step into) for a more step by step approach)
Python variables	Show or hide python variables (like _var_) in code completion
Insert spaces/tabs	Determines if spaces or tab characters are added when pressing tab
Tab size	Sets the number of spaces that are considered equal to a tab character
Save before run	Saves the changes to the file before running
Create region	Creates a new region surrounding the selected text
Comment selection	Comments out the selected text
Convert to space indenting	Converts all tab characters in the script to spaces. The number of spaces is determined by Tab size
Convert to tab indenting	Converts all x number of space characters (determined by Tab size) in the script to tabs
Python (documentation)	Opens a link to the python website, showing you the python syntax and standard libraries

3.4.8 Shortcuts

The shortcut keys of the scripting editor within the User Interface are documented in Table 3.2.

Table 3.2: *Shortcut keys within the scripting editor of the User Interface.*

Shortcut	**Function**
`Ctrl + Enter`	Run selection (or entire script with no selection)
`Ctrl + Shift + Enter`	Run current region (region where the cursor is in)
`Ctrl + X`	Cut selection
`Ctrl + C`	Copy selection
`Ctrl + V`	Paste selection
`Ctrl + S`	Save script
`Ctrl + -`	Collapse all regions
`Ctrl + +`	Expand all regions
`Ctrl + "`	Comment or Uncomment current selection
`Ctrl + W`	Add selection as watch variable
`Ctrl + H`	Highlight current selection in script (press esc to cancel)
`F9`	Add/remove breakpoint (In debug mode only)

Table 3.2: Shortcut keys within the scripting editor of the User Interface.

Shortcut	Function
F5	Continue running (In debug mode only — when on breakpoint)
Shift + F5	Stop running (In debug mode only — when on breakpoint)
F10	Step over current line and break on next line (In debug mode only - when on breakpoint)
F11	Step into current line if possible, otherwise go to next line (In debug mode only — when on breakpoint). This is used to debug functions declared in the same script (that have already runned)

3.4.9 Quick access toolbar

Note: The user can make frequently used functions available by a single mouse-click in the *Quick Access Toolbar*, the top-most part of the application-window. Do this by right-mouse-clicking a ribbon item and selecting *Add to Quick Access Toolbar*.

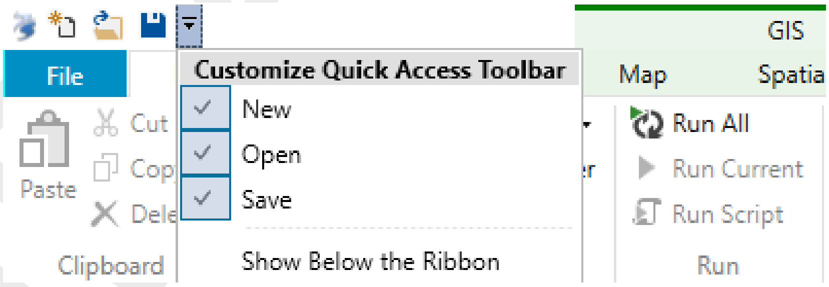


Figure 3.20: The quick access toolbar.

3.5 Important differences compared to Delft3D-FLOW GUI

The differences between the former Delft3D-FLOW GUI and the D-Flow FM GUI in lay-out and functionality are numerous. Here, we address only the most important differences in the workflow.

3.5.1 Project vs model

The entity “project” is new in the D-Flow FM GUI. In the hierarchy the entity “project” is on a higher level than the entity “model”. A project can contain multiple models, which can either run standalone or coupled. The user can run all models in the project at once (on project level) or each model separately (on model level). When the user saves the project, the project settings will be saved in a <∗.dsproj> configuration file and the project data in a <*.dsproj_data folder>. The <*.dsproj_data> folder contains folders with all input and output files for the models within the project. There is no model intelligence in the <∗.dsproj> configuration file, meaning that the models can also be run outside the GUI from the <*.dsproj_data> folder.

3.5.2 Load/save vs import/export

The user can load an existing D-Flow FM project, make changes in the GUI and, consequently, save all the project data. Loading and saving means working on the original project data, i.e. the changes made by the user overwrite the original project data. Alternatively, use *Save As* to keep the original project data and save the changes project data at another location (or with another name).

Import/export functionality can be used to copy data from another location into the project (import) or, vice versa, to copy data from the project to another location (export). Import/export is literally copying, e.g.:

- import: changes on the imported data will only affect the data in the project and not the source data (upon saving the project)
- export: the model data is copied to another location "as is", changes made afterwards will only affect the data in the project not the exported data (upon saving the project)

3.5.3 Working from the map

One of the most important differences with the former GUI is the central map. The central map is comparable with the former "visualization area", but with much more functionality and flexibility. The map helps you to see what you are doing and inspect the model at all times. You can use the *Region* and *Map* ribbons to add/edit model features in the map.

3.5.4 Coordinate conversion

With the map as a central feature, functionality to convert model and map coordinates is an indispensable feature. In the *General* tab you can set the model coordinate system. In the map tree you can set the map coordinate system using the right mouse button (Figure 3.21). The coordinate systems are subdivided in geographic and projected systems. Use the quick search bar to find the coordinate system you need either by name or EPSG code (Figure 3.22).

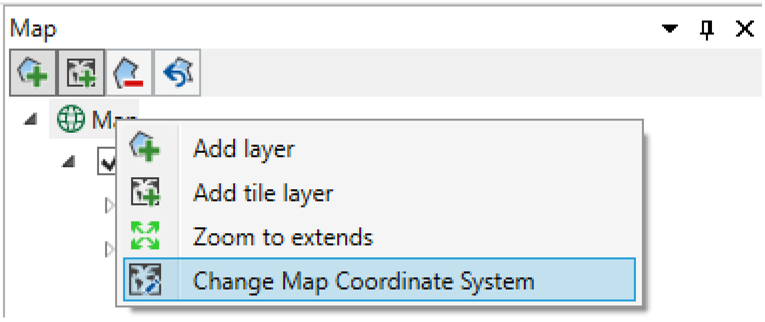
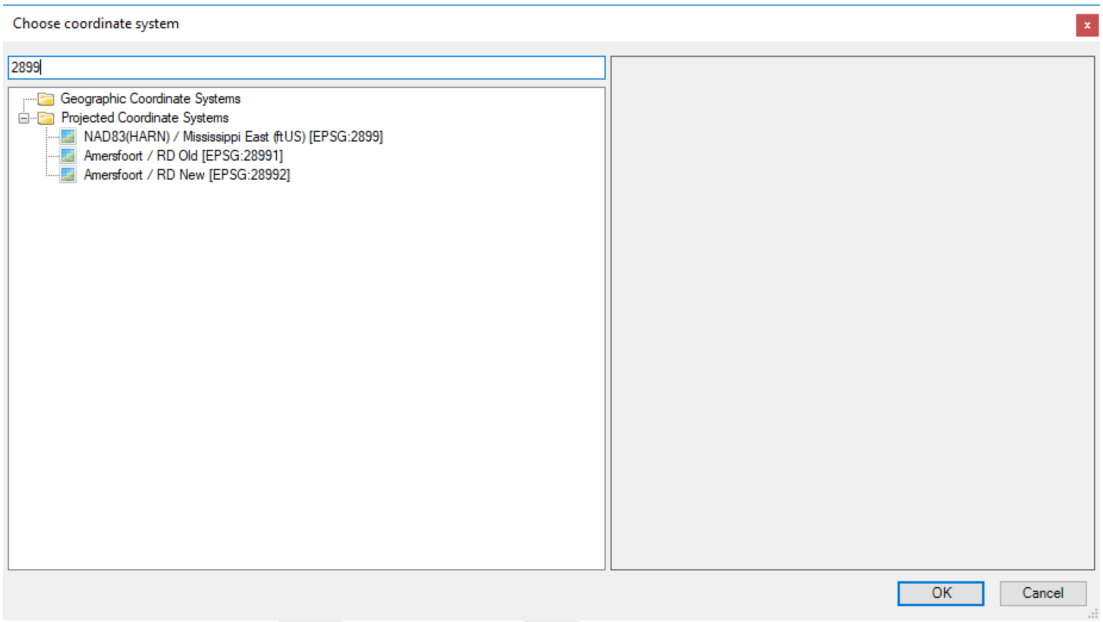


Figure 3.21: *Set map coordinate system using right mouse button*



2899

- Geographic Coordinate Systems
- Projected Coordinate Systems
  - NAD83(HARN) / Mississippi East (ftUS) [EPSG:2899]
  - Amersfoort / RD Old [EPSG:28991]
  - Amersfoort / RD New [EPSG:28992]

Figure 3.22: *Select a coordinate system using the quick search bar*

3.5.5 Model area

The model area contains geographical features, such as observation points & curves and obstacles. In contrast to the former GUI, these features can even exist without a grid or outside the grid and they are not based on grid coordinates, implying that their location remains the same when the grid is changed (for example by (de-)refining).

Finally, for the computations, the SWAN computational core interpolates the features to the grid. In the future we would like to show to which grid points the features are snapped before running the computation. However, this requires some updates in the SWAN computational core.

3.5.6 Integrated models (model couplings)

The D-Flow FM User Interface implements the concept of an *Integrated model* in order to couple different models, such as: hydrodynamics coupled with the controlling of structures, waves, morphology and/or water quality.

Two types of coupling are distinguished:

1. *offline* coupling and
2. *online* coupling.

Note: *Offline* is also referred to as *sequential* coupling and *online* as *parallel* coupling.

Offline

In case of an *Integrated model* with *offline* coupling, the entire hydrodynamic simulation is done first, i.e., *separately* from the second simulation. The file-based hydrodynamic output serves as input for the second simulation. As such, the hydrodynamic results drives the controlling of structures or the simulation of waves or water quality. In this offline case, there is no feedback from the waves or water quality to the hydrodynamic simulation. For many applications, this is good practice.

Online

An *online* coupling, on the other hand, exchanges data *every time* after computing a specified time interval. This tight coupling allows for direct feedback of the various processes on one another. This is crucial for controlling structures.

3.5.7 Ribbons (shortcut keys)

The User Interface makes use of ribbons, just like Microsoft Office. You can use these ribbons for most of the operations. With the ribbons comes shortcut key functionality, providing shortcuts to perform operations. If you press `Alt`, you will see the letters and numbers to access the ribbons and the ribbon contents (i.e. operations). For example, `Alt + H` will lead you to the *Home*-ribbon (Figure 3.23).

Note: Implementation of the shortcut key functionality is still work in progress.

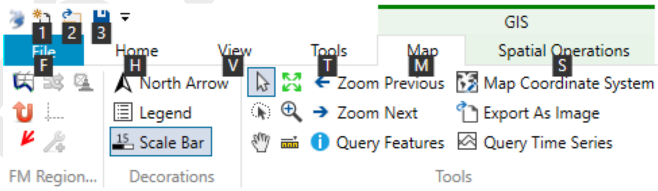


***Figure 3.23:** Perform operations using the shortcut keys*

3.5.8 Context menus

Context menus are the menus that pop up using the right mouse button. These context menus provide you with some handy functionality and shortcuts specific for the selected item. The functionality is available in all User Interface windows and context dependent. You can best try it yourself to explore the possibilities.

3.5.9 Scripting

The User Interface has a direct link with scripting in Iron Python (NB: this is not the same as C-Python). This means that you can get and set data, views and model files by means of scripting instead of having to do it all manually. Scripting can be a very powerful tool to automate certain steps of your model setup or to add new functionality to the GUI. You can add a new script by adding a new item, either in the *Home*-ribbon or through the right mouse button.

4 Tutorial

This chapter includes a number of basic tutorials on setting up and running a hydrodynamic simulation. However, before you start with the tutorials, it's important to learn some **basic grid concepts**.

Note: in this chapter, read for Delft3D Flexible Mesh Suite either "D-HYDRO Suite" or "Delft3D Flexible Mesh Suite".

4.1 Introduction: Basic grid concepts

In D-Flow FM, grids (sometimes denoted as 'networks') consist of net cells and are described by **net nodes** (corners of a cell), **net links** (edges of a cell, connecting net nodes), **flow nodes** (the cell circumcentre) and **flow links** (a line segment connecting two flow nodes).

This grid topology is illustrated in Figure 4.1. Important properties of the mesh are the *orthogonality* and *smoothness*. The *orthogonality* is defined as the cosine of the angle φ between a flowlink and a netlink. Ideally 0, angle $\varphi = 90°$. The *smoothness* of a mesh is defined as the ratio of the areas of two adjacent cells. Ideally 1, the areas of the cells are equal to each other. A nearly ideal setup is shown in Figure 4.2.

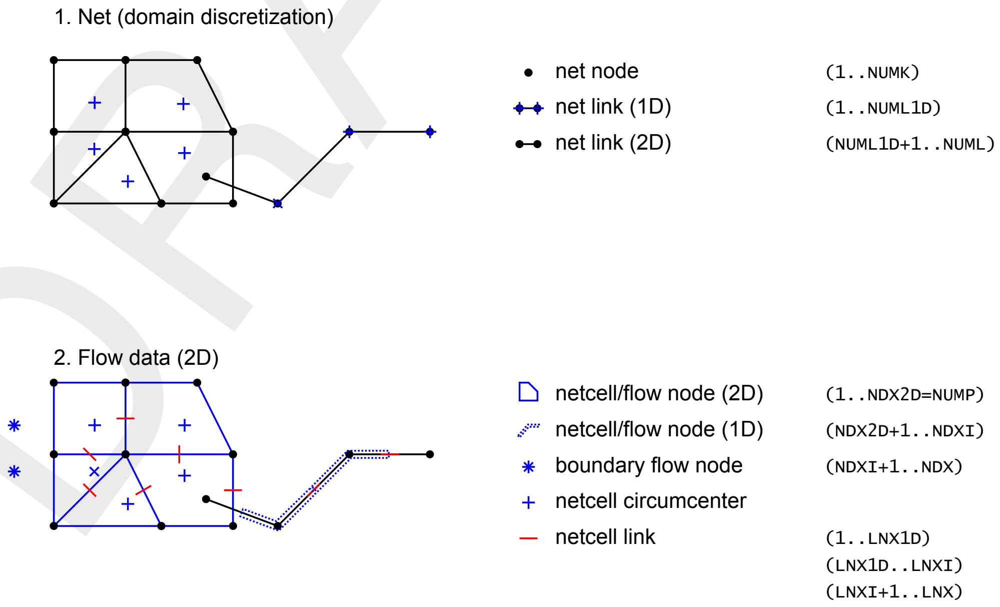

1. Net (domain discretization)
● net node (1..NUMK)
net link (1D) (1..NUML1D)
net link (2D) (NUML1D+1..NUML)
2. Flow data (2D)
netcell/flow node (2D) (1..NDX2D=NUMP)
netcell/flow node (1D) (NDX2D+1..NDXI)
boundary flow node (NDXI+1..NDX)
+ netcell circumcenter
netcell link (1..LNX1D)
(LNX1D..LNXI)
(LNXI+1..LNX)

***Figure 4.1:** Topology and definitions for a grid as used in D-Flow FM.*

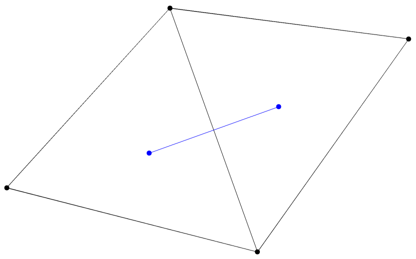
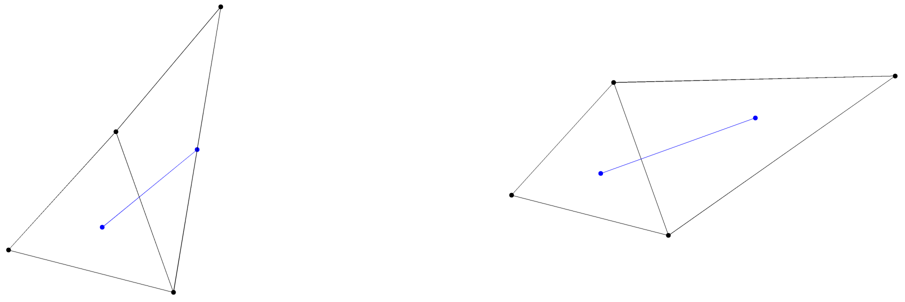


Figure 4.2: *Perfect orthogonality and nearly perfect smoothness along the edge connecting two triangles. Black lines/dots are network links/nodes, blue lines/dots are flow links/nodes.*

It is rather easy to generate grids that violate the orthogonality and smoothness requirements. In Figure 4.3, two different setups of two grid cells are shown with different grid properties. Figure 4.3a shows how orthogonality can be detoriated by skewing the right triangle with respect to the left triangle. While having the same area (perfect smoothness), the mutually oblique orientation results in poor orthogonality. In this particular case the circumcentre of the right triangle is outside the area of the triangle because it is an obtuse triangle, which is bad for computations with D-Flow FM. The opposite is shown in Figure 4.3b in which the right triangle has strongly been elongated, disturbing the smoothness property. However, the orthogonality is perfect (both triangles are acute). Nonetheless, both grids need to be improved to assure accurate model results.



(a) *Perfect smoothness, but poor orthogonality.*

(b) *Perfect orthogonality, but poor smoothness*

Figure 4.3: *Poor grid properties due to the violation of either smoothness or orthogonality at the edge connecting two triangles.*
Black lines/dots are network links/nodes,
blue lines/dots are flow links/nodes.

4.2 Tutorial Hydrodynamics for depth-averaged (2D) modelling

4.2.1 Outline of the tutorials

In this three-hours tutorial you will experience the basic functionality of D-Flow FM. On the one hand you will learn how to capture a certain area with a computational grid. On the other hand you will set up and run a computation yourself. This involves for example inserting a bed level, imposing boundary conditions, running a D-Flow FM model and postprocessing with a D-Flow FM output file. This manual contains nine tutorials (with indication of the estimated time):

- Tutorial 1 (section 4.2.2): Creating a curvilinear grid (25 minutes).
- Tutorial 2 (section 4.2.3): Creating a triangular grid (20 minutes).
- Tutorial 3 (section 4.2.4): Coupling multiple distinct grids (15 minutes).
- Tutorial 4 (section 4.2.5): Inserting a bed level (15 minutes).
- Tutorial 5 (section 4.2.6): Imposing boundary conditions (25 minutes).
- Tutorial 6 (section 4.2.7): Defining output locations (15 minutes).
- Tutorial 7 (section 4.2.8): Defining computational parameters (10 minutes).
- Tutorial 8 (section 4.2.9): Running a model simulation (20 minutes).
- Tutorial 9 (section 4.2.10): Viewing the output of a model simulation (15 minutes).

The necessary input files for a tutorial can be found in the folders *tutorial01* through *tutorial09*. When saving your model data while working through these tutorials, be sure not to overwrite the tutorial data. Ideally, you should create a separate working folder somewhere else on your computer to save your work. The folder *finished* in each tutorial folder contains the output generated by Deltares as a reference.

We will use the Western Scheldt and the harbour of Antwerp as an example case for generating a grid and for setting up and running a computation.

4.2.2 Tutorial 1: Creating a curvilinear grid

Generating grids is done with the RGFGRID tool. Because this is an iterative process and the Undo functionality in RGFGRID is still rather limited, it is important to frequently save intermediate RGFGRID results. By clicking on `Esc` it is sometimes possible to undo the last action. RGFGRID is a separate tool that communicates with the Delft3D Flexible Mesh Suite GUI through a temporary folder whose path is shown in the top menu bar. In order to transfer the generated grid to the Delft3D Flexible Mesh Suite GUI, be sure to save your data to this path when leaving RGFGRID.

To start RGFGRID in the Delft3D Flexible Mesh Suite Graphical User Interface the following steps have to be carried out:

- Double click on the Delft3D Flexible Mesh Suite icon on the desktop of your computer.
- Click on *New Model* in the main toolbar at the top of the Delft3D Flexible Mesh Suite GUI.
- A new window pops up in which you have to select *Flow Flexible Mesh Model* → *OK*.
- Double click on *Grid*, see Figure 4.4. RGFGRID will now start.

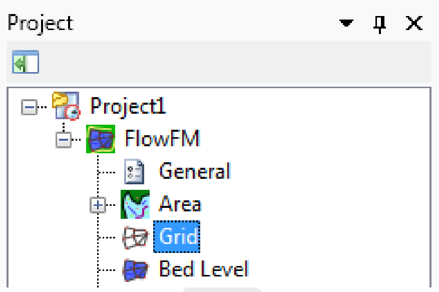


Figure 4.4: *Start RGFGRID by a double-click on* Grid.

➢ Choose from the main menubar *Coordinate System* → *Cartesian Coordinates*. This is the default setting, which means that you do not have to change anything.

A curvilinear grid is iteratively generated by first drawing splines. The approach is as follows:deltashe

➢ Import a land boundary via *File* → *Attribute Files* → *Open Land Boundaries* and select the file <tutorial01\scheldtriver.ldb>

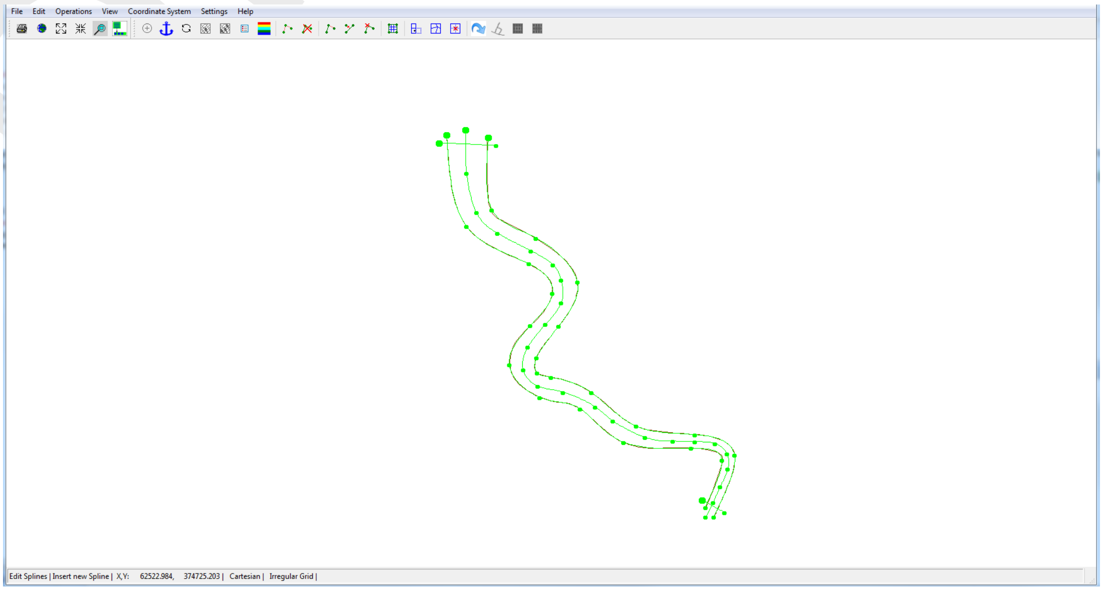
➢ Select *Edit* → *Splines* → *New* and draw a spline roughly following the left boundary of the river (use the leftmousebutton). Finalise the spline by one rightmouseclick.
Now we want to bring (attach) the spline close to the boundary.

➢ Choose the option *Edit* → *Splines* → *Attach to Land Boundary*. Select the the two outer points delimiting the spline to be snapped. The vertices which will be snapped are now marked by a square. Press the rightmousebutton to perform the snapping.

➢ A window appears for extra snapping: Press *Yes* several times. You will see the spline evolve towards the land boundary. Press *No* when you are satisfied with the result.

➢ Repeat the previous three steps for the righthandside spline.

➢ Draw a third spline along the river axis; see Figure 4.5.

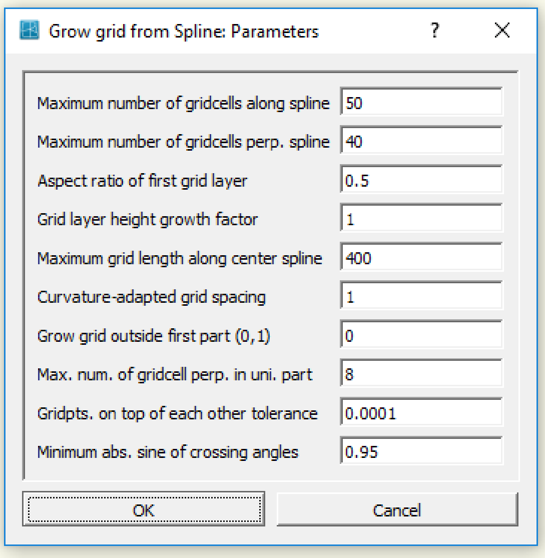


Edit Splines | Insert new Spline | X,Y: 62522.984, 374725.203 | Cartesian | Irregular Grid |

Figure 4.5: *Splines in Tutorial01*

➢ Draw two cross-splines, each containing exact two vertices: one spline at the North side and one spline at the South side of the river (see Figure 4.5). The channel is now enveloped by four splines and there is an extra spline along the river axis.
Now we can grow a grid from these splines.

➢ Choose *Settings* → *Grow Grid from Splines. . .* . You will be able to set several settings for this process. Take over the values shown in Figure 4.6.



Grow grid from Spline: Parameters

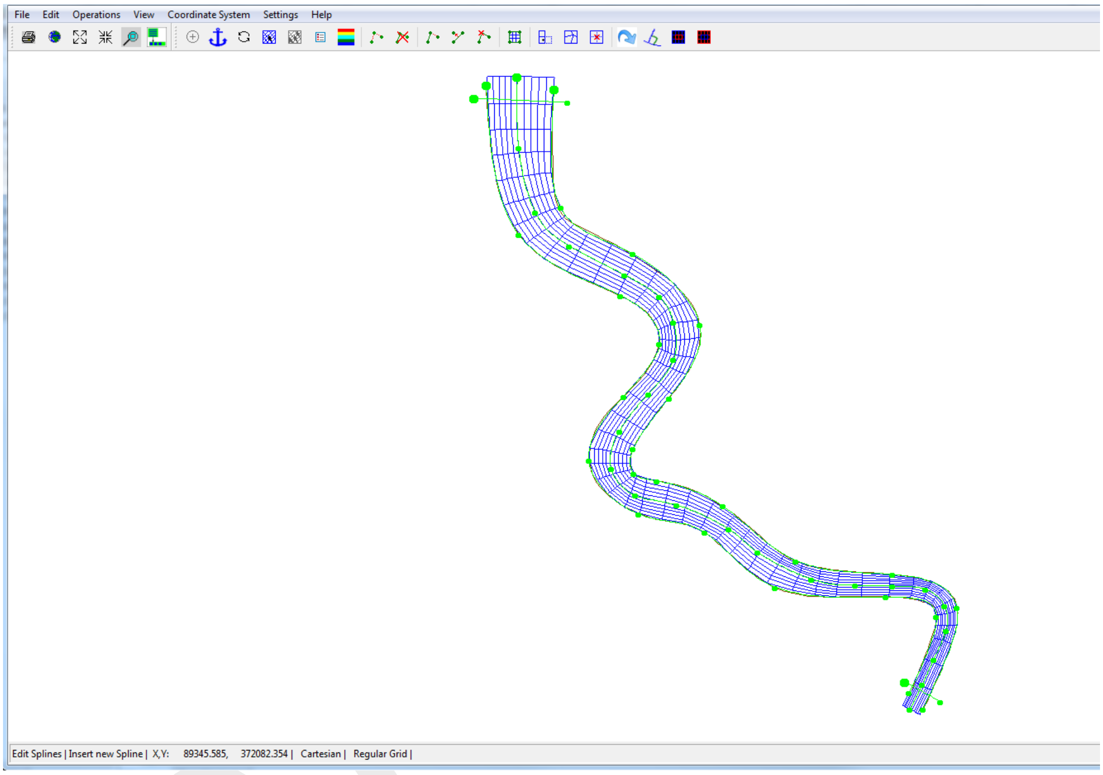
Maximum number of gridcells along spline	50
Maximum number of gridcells perp. spline	40
Aspect ratio of first grid layer	0.5
Grid layer height growth factor	1
Maximum grid length along center spline	400
Curvature-adapted grid spacing	1
Grow grid outside first part (0,1)	0
Max. num. of gridcell perp. in uni. part	8
Gridpts. on top of each other tolerance	0.0001
Minimum abs. sine of crossing angles	0.95

Figure 4.6: *Settings for the* Grow Grid from Splines *procedure.*

A brief explanation:

- ◇ Using the parameter *Max. num. of gridcells perp. in uni. part*, the user can give an indication of the number of cells across the width between the longitudinal splines.
- ◇ By using the parameters *Maximum grid length along center spline*, the user can give an indication of the length of the cells in longitudinal direction. Based on the value of the parameter *Aspect ratio of first grid layer*, the algorithm establishes a suitable grid, under the restrictions of the prevailing maximum numbers of gridcells (first two entries).
- ◇ The option *Grid layer height growth factor* enables the user to demand for a non-equidistant grid in cross-sectional direction. The value represents the width-ratio of two adjacent cells. Using the option *Grow grid outside first part (0/1)*, one can extend a grid outside the longitudinal splines, for instance to capture winterbed regions.

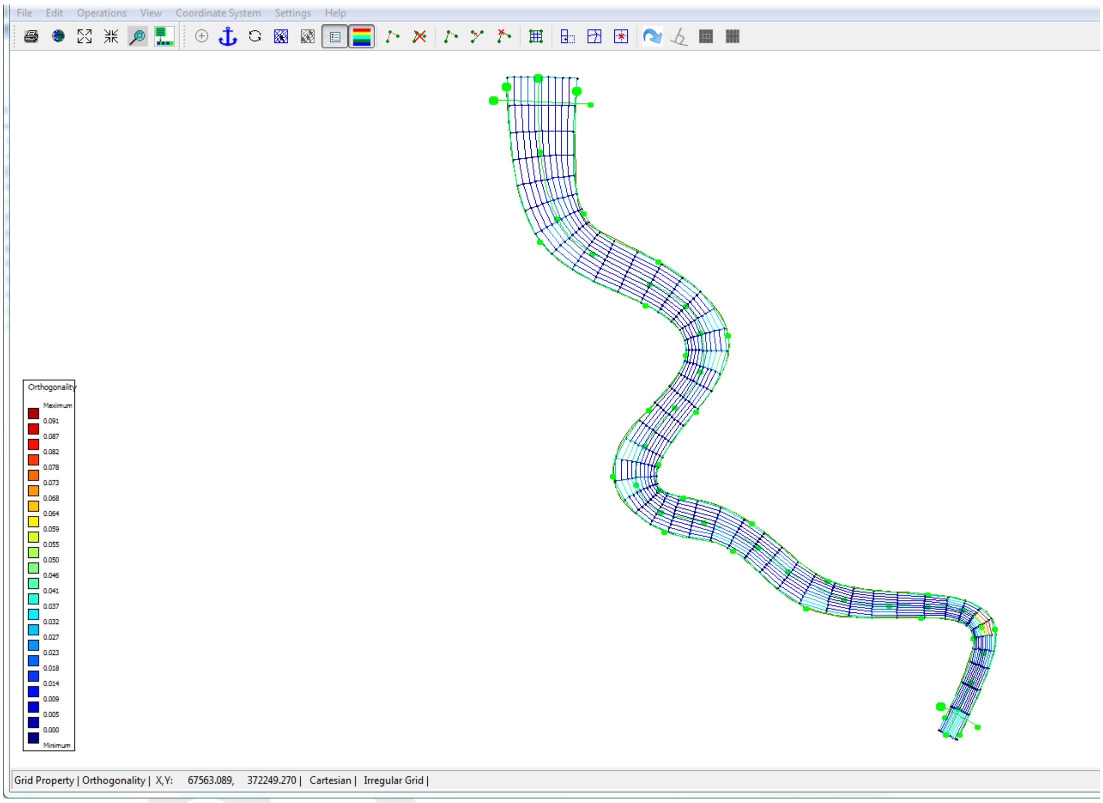
➢ After entering the values of Figure 4.6, choose *Operations* → *Grow Grid from Splines*. This will deliver the grid as shown in Figure 4.7.



Edit Splines | Insert new Spline | X,Y: 89345.585, 372082.354 | Cartesian | Regular Grid |

Figure 4.7: *Generated curvilinear grid after the new* Grow Grid from Splines *procedure.*

- ➢ To be able to further improve the grid, choose *Operations* → *Convert grid* → *Regular to Irregular*. Strictly, the grid is now not curvilinear anymore, but unstructured.
- ➢ Choose *View* → *Grid Property Style* → *Coloured Edge* and then *View* → *Grid Property* → *Orthogonality*. The result is shown in Figure 4.8. We remark that no orthogonalisation iterations have been performed yet. Try to reach the level of orthogonality as shown in Figure 4.8. By clicking on Menu option *Operations* →*Orthogonalize* the orthogonality can be improved.



Grid Property | Orthogonality | X,Y: 67563.089, 372249.270 | Cartesian | Irregular Grid |

Figure 4.8: *Orthogonality of the generated curvilinear grid after the* Grow Grid from Splines *procedure.*

To save the grid and close RGFGRID the following steps have to be carried out:

- Choose from the menubar *File* → *Export* → *UGRID (D-Flow FM)...* and save the grid **in your working folder** as <tutorial01\scheldt01_net.nc>. The grid has now been saved.
- Choose *File* → *Attribute Files* → *Save splines with Intermediate Points* and save the splines **in your working folder** as <tutorial01\scheldt01.spl>. The splines have now also been saved.
- Choose *File* → *Save Project*. Press Cancel when a file selection box asks to save the land boundary.
- Close RGFGRID by choosing *File* → *Exit*

Folder <tutorial01\finished> contains splines and a grid for this tutorial exercise that has been prepared by Deltares.

Now you are back in the Delft3D Flexible Mesh Suite GUI. As you can see only the grid has been imported from RGFGRID. If you want to see the land boundary here too, you need to import it:

- Open *Area* under *Project1* →*FlowFM* →*Area*.
- Import the land boundary from <tutorial01\scheldtriver.ldb> by choosing *Import* via the right mouse button on *Project1* →*FlowFM* →*Area* →*Land Boundaries* (see Figure 4.9).
- Do not apply a coordinate transformation by pressing *OK*.

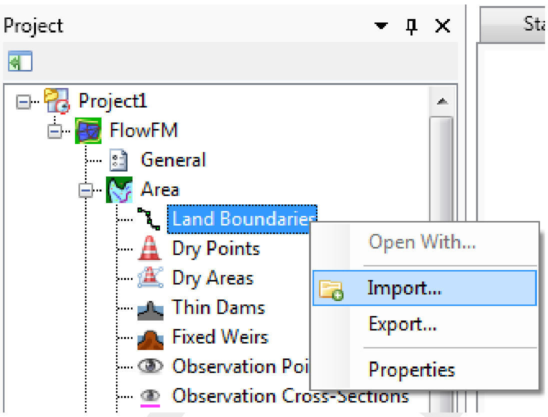


Figure 4.9: *Importing a land boundary*

Now the grid and land boundary are both visible in the central map of the Delft3D Flexible Mesh Suite GUI, see Figure 4.10.

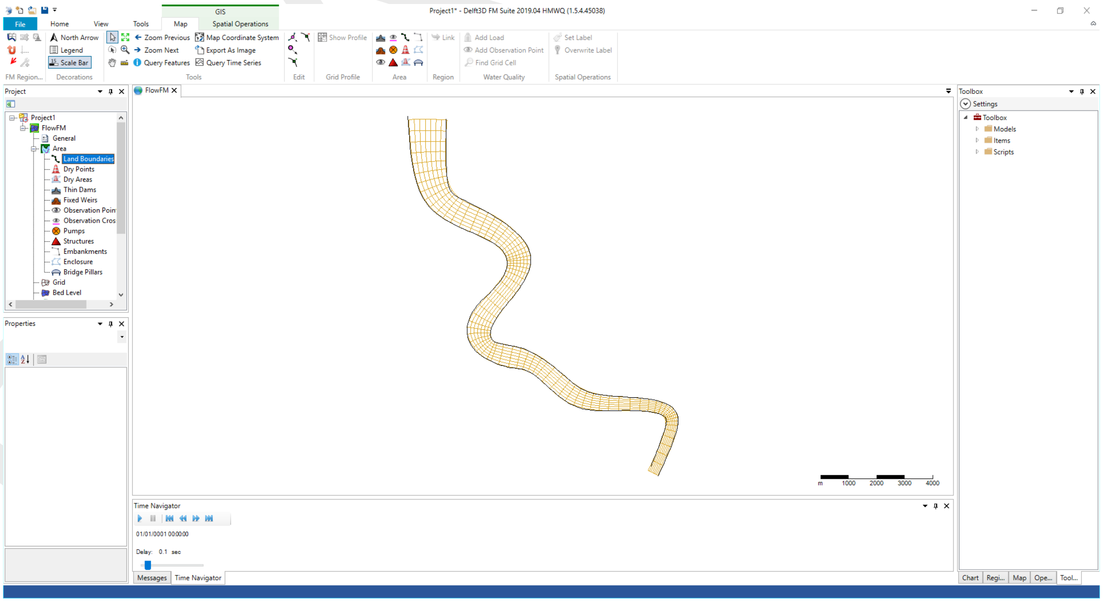


Figure 4.10: *After closing RGFGRID the grid is visible in the Delft3D Flexible Mesh Suite GUI.*

This is the end of Tutorial 1.

➢ To close the Delft3D Flexible Mesh Suite, click on *File* → *Close*. Then, the following question arises: "Save changes to the project: Project?" Click on *No*. The current D-Flow FM model has now been closed.

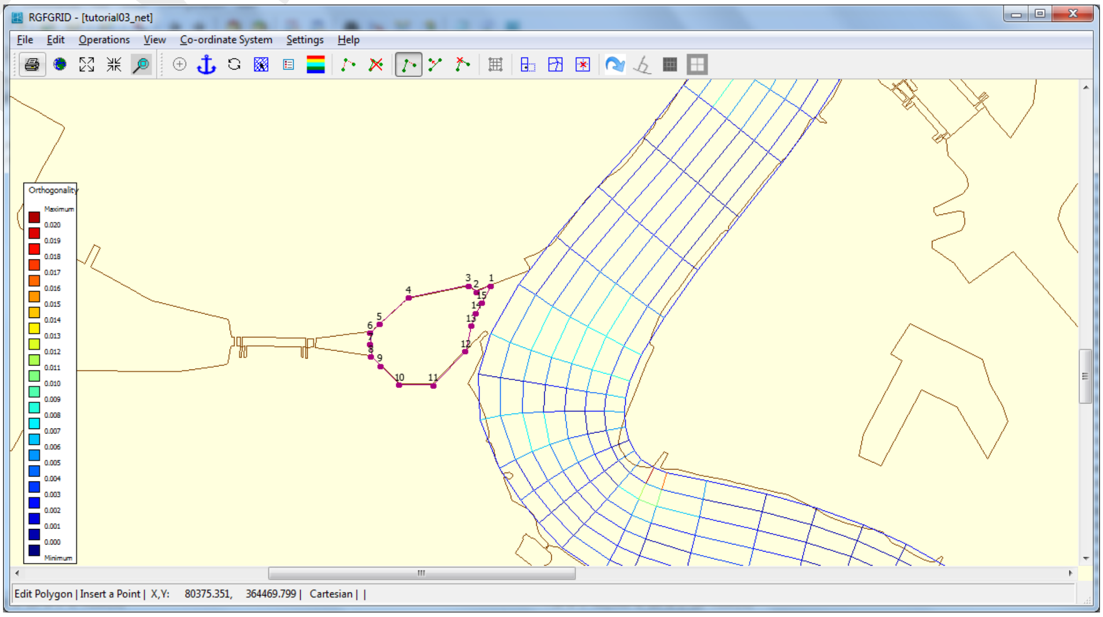
➢ Exit the Delft3D Flexible Mesh Suite by choosing the close icon (✕) in the windows title bar.

4.2.3 Tutorial 2: Creating a triangular grid

We will continue with the grid created in the previous tutorial for the Scheldt river. The river is separated from the harbour, west of the river, by a sluice. The small area between the sluice and the Scheldt will benefit from an unstructured grid because of its irregular geometry. The unstructured grid for this irregular geometry is created first in this section.

The following steps have to be carried out:

- ➢ Start the the Delft3D Flexible Mesh Suite and create a new Flow Flexible Mesh Model as described in section 4.2.2.
- ➢ Import the land boundary from <tutorial02\scheldtharbour.ldb> by choosing *Import* via the right mouse button on *Project1* →*FlowFM* →*Area* →*Land Boundaries* (see Figure 4.9). Select item *Land boundaries from .ldb file* and press *OK*. Select the file <scheldtharbour.ldb> with the file selection window. Do not apply a coordinate transformation, press *OK*.
- ➢ Import the grid file by choosing *Import* via the right mouse button on *Project1* →*FlowFM* →*Grid*. Choose the grid file <scheldt02_net.nc> in the folder <tutorial02>.
- ➢ Start RGFGRID by double clicking on *Grid*, see Figure 4.4. Set the Coordinate System (if necessary), as has been explained in section 4.2.2 (Figure 4.4).
- ➢ Choose from the menubar *Edit* → *Polygons* → *New*. The intention is to mark the area of interest (i.e. the area for which we want to generate the grid) by means of a polygon, see Figure 4.11.
- ➢ Start drawing the polygon at a distance from the curvilinear grid that is similar to the width of the curvilinear grid cells (since we want to have a smooth transition in grid resolution). Specify the second point at a relatively small distance from the first one. This distance is later used as the first indication of the size of the triangular gridcells to be placed.
- ➢ Mark the characteristic locations of the area (follow the land boundary) and place the final points at the same distance away from the river grid as the first point. The distance between the last two points should be similar to the distance between the first two points. The result is shown in Figure 4.11.
- ➢ Choose *Edit* → *Polygons* → *Refine (linear)* and click on the first and last points of the polyline (in Figure 4.11 we would select point 1 first and then point 15). Now, the polygon is divided into a finer set of line elements (based on the length of the first and last segment of the polyline).



Edit Polygon | Insert a Point | X,Y: 80375.351, 364469.799 | Cartesian | |

Figure 4.11: *Polygon that envelopes the area in which an unstructured grid is aimed to be established.*

Remarks:

◇ The distance between the points of the polygon is derived from the distance of the two polyline segments at both sides of the *selected* segment. The length of the polyline segments varies linearly from the segment length at the one side of the selected segment towards the segment length at the other side of the selected segment.

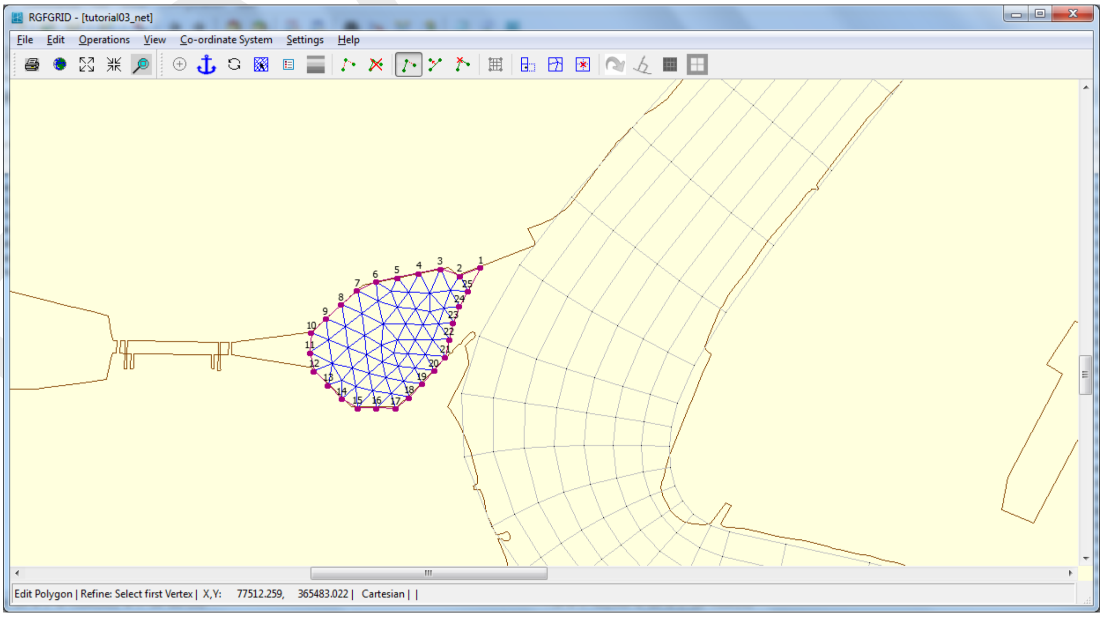
- ◇ You can play around to see how this works. If needed, you can add extra polyline points by choosing *Edit* → *Polygon* → *Insert point*. Choose *Edit* → *Polygon* → *Move point* if a point move would make sense.
- ◇ You can snap the refined polygon to the land boundary through *Edit* → *Polygons* → *Attach to Land Boundary*. Select two points for this.

Now we continue with this tutorial by carrying out the following steps:

- ➢ Choose *Operations* →*Domain* →*New* to indicate that the new to be generated grid should be created next to the already existing one (and not replace it).
- ➢ Choose *Operations* → *Grow Grid from Polygons*. The result is shown in Figure 4.12.
- ➢ Improve the orthogonality through *Operations* → *Orthogonalise grid*. The default settings are set to also improve the smoothness.

Since the solver only supports one grid, we need to merge the newly created grid with the original grid of the river. The newly created grid is currently selected. Next:

- ➢ Choose *Edit* →*Multi Select* and click on the river Scheldt grid to select it as well. After selection both grids should be coloured blue with black dots on the nodes.
- ➢ Choose *Operations* →*Grid* →*Merge Grids* to merge the nodes of the (two) highlighted grids. Now, the grids have been merged. However, nothing will visually change.



Edit Polygon | Refine: Select first Vertex | X,Y: 77512.259, 365483.022 | Cartesian | |

Figure 4.12: *Unstructured grid, after having refined the polygon.*

To save the grid and go back to the Delft3D Flexible Mesh Suite GUI:

- ➢ Choose from the menubar *File* → *Export* → *UGRID (D-Flow FM)...* and save the grid **in your working folder** as <tutorial02\scheldt02_net.nc>. Both grids grids have now been merged into one <.nc> file.
- ➢ Choose *File* → *Save Project As* and save the project in your working folder. Press Cancel when a file selection box asks to save the land boundary. Press Cancel twice when a file selection box asks to save the grid.
- ➢ Close RGFGRID by choosing *File* → *Exit*

Now you are back in the Delft3D Flexible Mesh Suite GUI but the grid is not yet updated in the D-Flow FM model. Therefore finalyse this tutorial as follows:

- ➢ Choose *Import* by a right mouse click on *Project1* → *FlowFM* → *Grid* and select the grid file <scheldt02_net.nc> **in your working folder**.
- ➢ Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar. You don't have to save the project.

This is the end of Tutorial 2.

Folder <tutorial02\finished> contains splines and a grid for this tutorial exercise that has been prepared by Deltares.

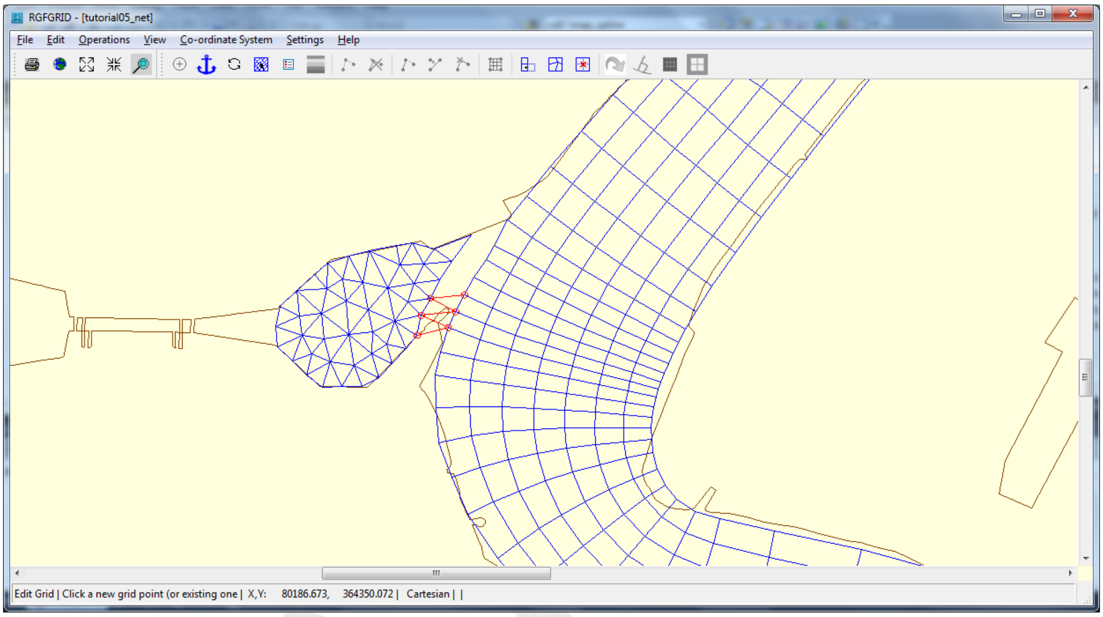
4.2.4 Tutorial 3: Coupling multiple separate grids

From the previous tutorial we have ended up with two separate grids that are not connected yet. Obviously, these two grids should properly be integrated into one single grid. Before we can couple the two grids, we should first make sure that the typical grid size is of the same order of magnitude for both grids at the location where the connection is to be laid. Hence, basically two operations are to be done:

- ◇ Split the grid cells in the Scheldt river along this length so that the cell sizes of both grids match.
- ◇ Merge the two grids and put connections in between.

The splitting can be established as follows:

- ➢ Start the the Delft3D Flexible Mesh Suite and create a new Flow Flexible Mesh Model as described in section 4.2.2.
- ➢ Import the land boundary file and the grid file from the folder <tutorial03> as described in section 4.2.3.
- ➢ Start RGFGRID and set the Coordinate System (if necessary), as has been explained in section 4.2.2 (Figure 4.4).
- ➢ Choose *Edit* → *Irregular Grid* → *Split Row or Column*.
- ➢ Select the locations where the grid lines should be split by clicking on the left boundary of the Scheldt river.
- ➢ Try to achieve the picture shown in Figure 4.13, in particular the typical grid size in the curved area. *N.B. Ignore the red lines in this picture for now!*



Edit Grid | Click a new grid point (or existing one | X,Y: 80186.673, 364350.072 | Cartesian | |

Figure 4.13: *Connection of the river grid and the unstructured grid. The red lines show the inserted grid lines used to couple the two grids manually.*

The edges of the two the grid segments can be connected as follows:

- ➢ Since the two visually separated grids have already been merged into one logical grid in the previous tutorial, we can now physically connect them by laying new gridlines between them. Therefore, select *Edit* → *Irregular Grid* → *Insert Node*. Insert new gridlines in a zigzag-like style: see the red grid lines in Figure 4.13. Now, you will benefit from the (more or less) equal resolutions in the river and unstructured regions.
- ➢ Finally, you will end up with a picture like shown in Figure 4.14. You can visualize the orthogonality through *View* → *Grid Property Style* → *Coloured Edge* and *View* → *Grid property* → *Orthogonality*.

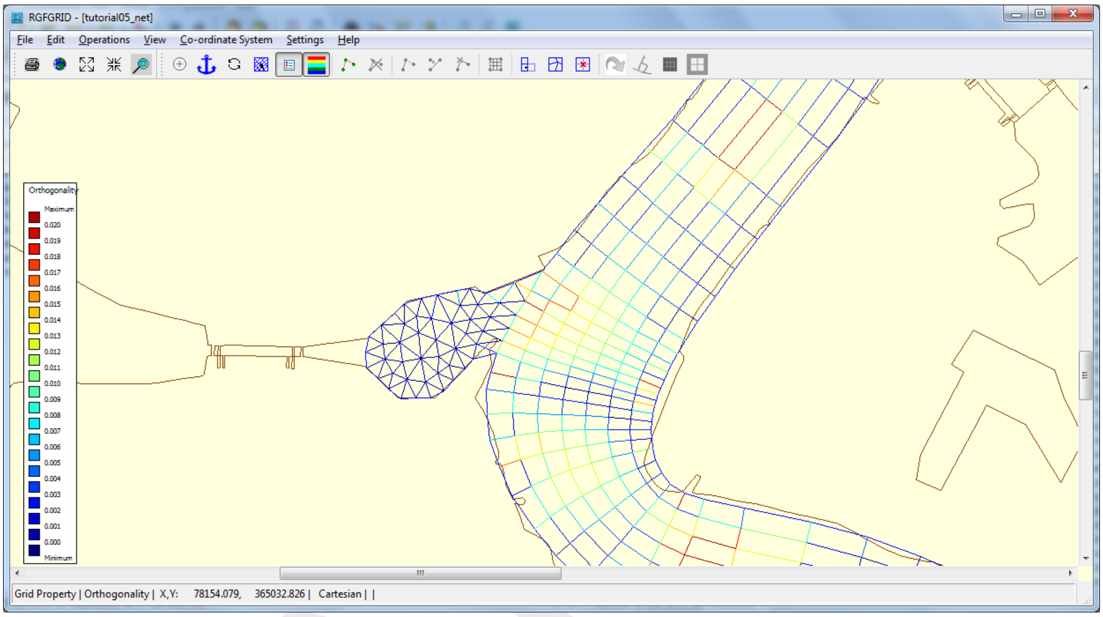


Figure 4.14: *Orthogonality of the integrated grid, containing the curvilinear part, the triangular part and the coupling between the two grids.*

To save the grid and close RGFGRID is done as follows:

- *File* → *Export* → *UGRID (D-Flow FM)...* and save the grid **to your working folder** as <tutorial03\scheldt03_net.nc>.
  The grid has now been saved.
- Choose *File* → *Save Project*.
- Close RGFGRID by clicking on *Exit*.

This is the end of Tutorial 3.

- Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar. You don't have to save the project.

4.2.5 Tutorial 4: Inserting a bed level

In previous tutorial exercises we focused on the grid generation. The next step is to couple the bed level data to the grid. The grid for this tutorial exercise is available in file <westernscheldt04_net.nc> in the directory <tutorial04>. A harbour has been added to the grid, as well as the Westernscheldt part at the North side. Bed levels are available in the file <westernscheldt_bed_level.xyz>.

The file with bed levels should first be projected onto the unstructured grid by means of interpolation. To this purpose, the following actions should be carried out:

- Start Delft3D Flexible Mesh Suite, create a new Flow Flexible Mesh model, import the land boundary <sealand.ldb> and the grid <westernscheldt04_net.nc> from directory <tutorial04> as explained in the previous tutorial exercises.
- Import a sample file containing the measured bed levels: in the drop down menu at the top of the screen (See Figure 4.15), select the *Spatial Operators* menu, choose layer *Bed Level*, click on Import *Import from file* and select the file <westernscheldt_bed_level.xyz> from directory <tutorial04>, *Import without transformation (as-is)* → *OK*. This file contains measured bed level data for the Westernscheldt, from the North Sea up to Antwerp

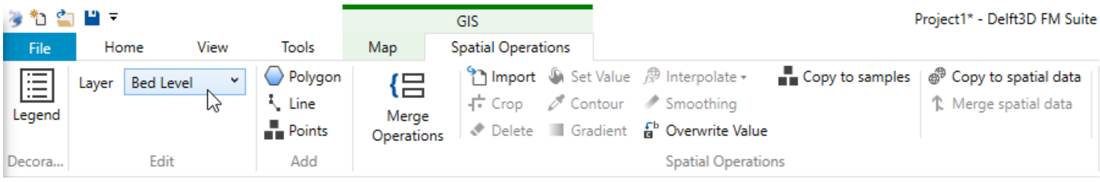


Figure 4.15: *Map-ribbon with the* Spatial Operations *menu.*

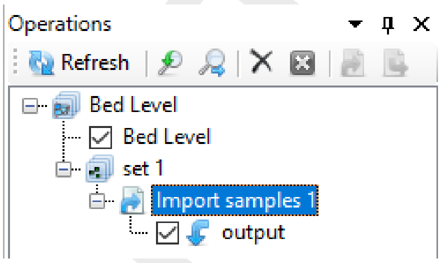


Figure 4.16: *Activate import samples 1.*

Now we are going to project the bed level data onto the nodes:

- Click on *Operations* and click on *"Import samples 1"*; see Figure 4.16.
- Click Interpolate and select *Interpolate selected set* (see Figure 4.15).
- The window **Interpolation operation** appears. Select the *Overwrite* option for *Pointwise operation*, then press *OK*. The result will look like Figure 4.17. If you do not see a color bar, click Legend *Legend* in the *Decorations* menu (see Figure 4.15) and switch off items in the *Map* pane to bring the color bar into view.

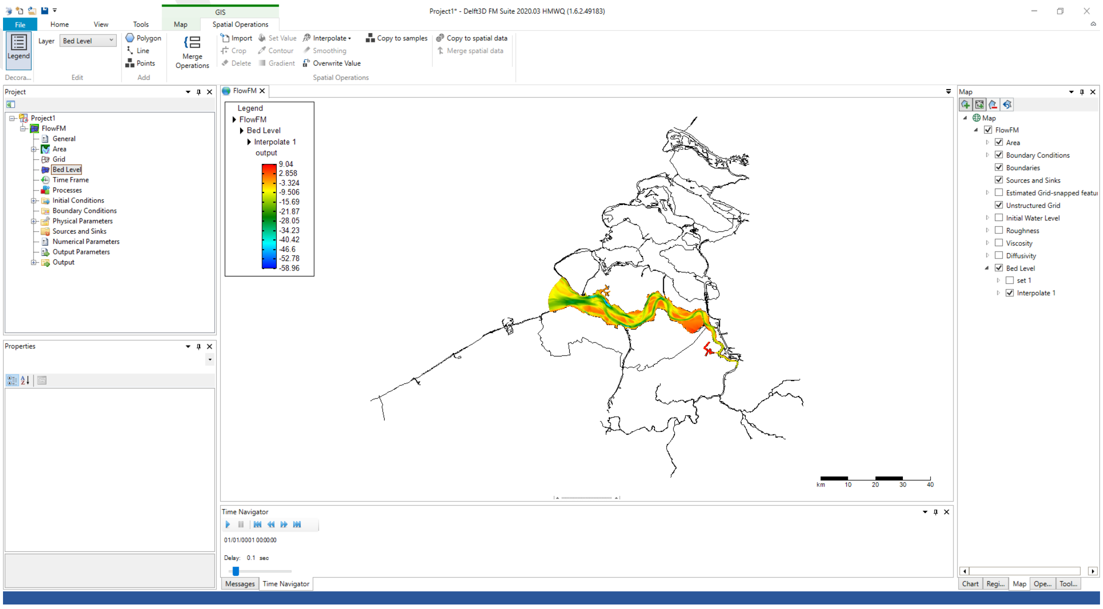


Figure 4.17: *Interpolated bed levels values at the grid (estuary).*

- Have a look at the sluice and the docks of the harbour. You can see that the bed level in this area is unrealistically high. This unrealistic value has been assigned because of undesired extrapolation, since no bed level data has been available in this area. To repair this, first draw a polygon that envelops the sluice and the docks. Do this by clicking Polygon *Draw polygon* (see Figure 4.15). Double-click to finish the polygon. See Figure 4.18.

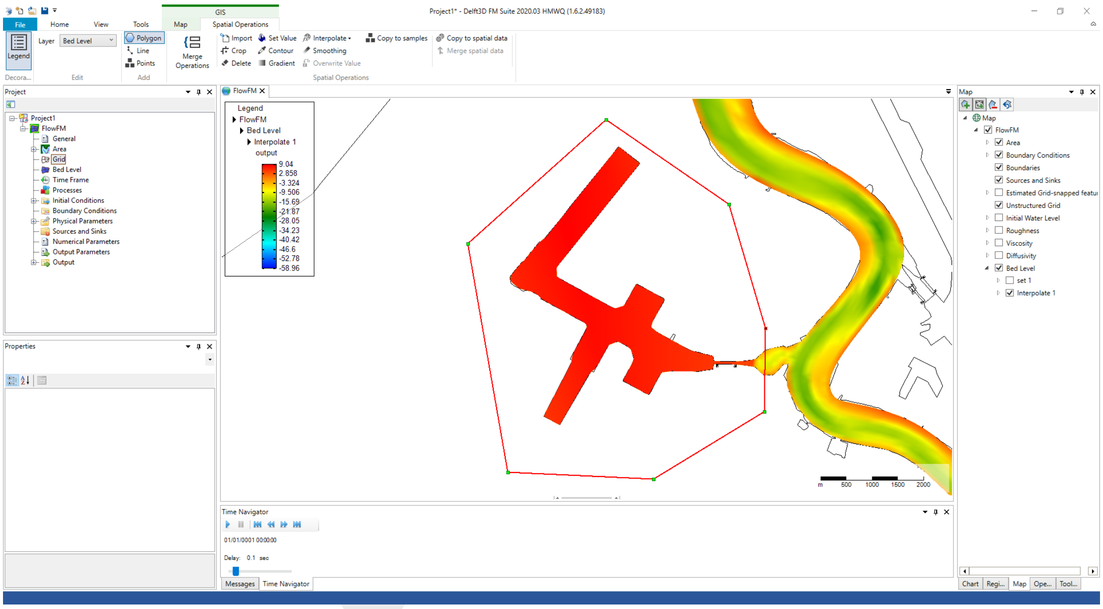


Figure 4.18: *Polygon drawn to around the missing depths in the harbour.*

➢ Choose Set Value *Set value* (see Figure 4.15): specify an appropriate value, for instance "-10" m. The result will look like as shown in Figure 4.19.

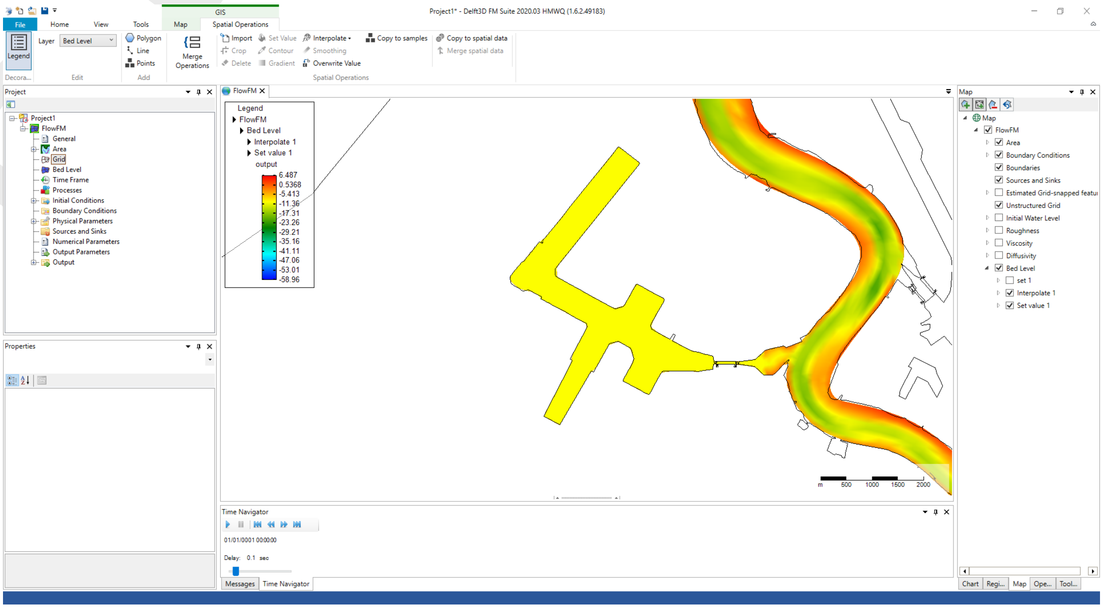


Figure 4.19: *Interpolated bed levels values at the grid (harbour).*

Note: The mesh file (<∗_net.nc>) will contain the grid together with the bed level data at the nodes.

To save the mesh and bed level, and save the model within the Delft3D Flexible Mesh Suite GUI the following should be done:

- To save the mesh file you have to rightmouseclick on *Grid* → *Export* in the project tree. Choose *Grid exporter* and save the file in the folder <tutorial04> of **your working environment** as <westernscheldt04_net.nc>. The grid including the bed level have now been saved.
- To close the D-Flow FM User Interface click on *File* → *Close*. Then, the following question arises: "Save changes to the project: Project?" Click on *No*.
- Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar.

Folder <tutorial04\finished> contains a bed level for this tutorial exercise that has been prepared by Deltares.

4.2.6 Tutorial 5: Imposing boundary conditions

Along both the sea boundary and the Scheldt river boundary, appropriate boundary conditions have to be imposed. The boundary conditions can be prescribed in many ways, for instance as water levels, velocities (tangential and normal direction), discharge and Riemann. In this tutorial exercise we will impose boundary conditions for the Western Scheldt in the following way:

- At the sea boundary: a periodic water level boundary, described as a harmonic signal with a mean 0.5 m+NAP, an amplitude 2.5 m and a frequency of 28.984 °h^{-1},
- At the river boundary: a constant discharge boundary, described by a constant 2500 m^3 s^{-1}.

Let us start with a new D-Flow FM model and import the network and land boundaries:

- Start Delft3D Flexible Mesh Suite, create a new Flow Flexible Mesh model, import the boundary <sealand.ldb> and the grid <westernscheldt05_net.nc> from directory <tutorial05> as explained in the previous tutorial exercises.

In order to define boundary conditions one has to follow this strategy:

1. draw a polyline along the boundary;
2. fill a file with essential information;
3. link the boundary files with the model setup.

The boundary conditions can be imposed as follows:

- In the *FM Region...* (in the top ribbon, see Figure 4.20) choose *Add a (2D) flow boundary* and insert a polyline along the sea boundary. Doubleclick to finish the polyline. In the Project bar at the left, under *Boundary Conditions*, this boundary has now been added. Figure 4.20 shows how this will look like.
  Tip: to remove a wrongly placed point in the polyline, use *Remove point from geometry* (top of screen, above *Edit*, see Figure 4.20). It is also possible to replace or add a point.

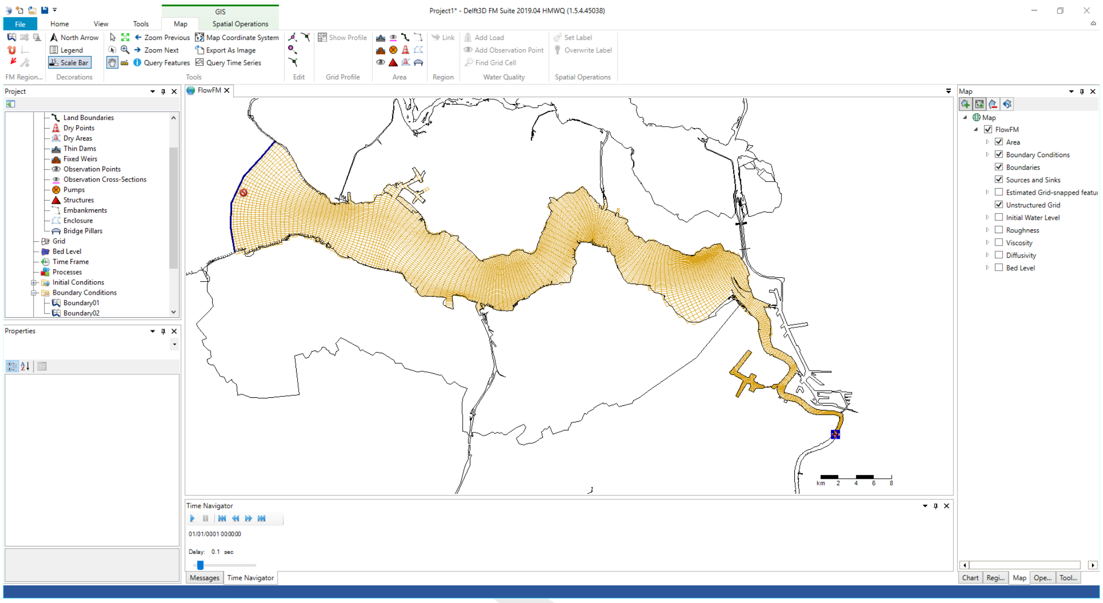


Figure 4.20: *Location of the two open boundaries at the sea and river side.*

- ➢ Draw another polyline at the river entrance south of Antwerp, see the right bottom corner of Figure 4.20).
- ➢ To impose boundary conditions at the seaboundary, double click at *Boundary01*, see Figure 4.21.

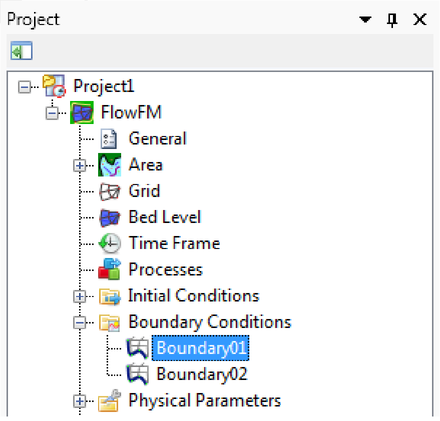
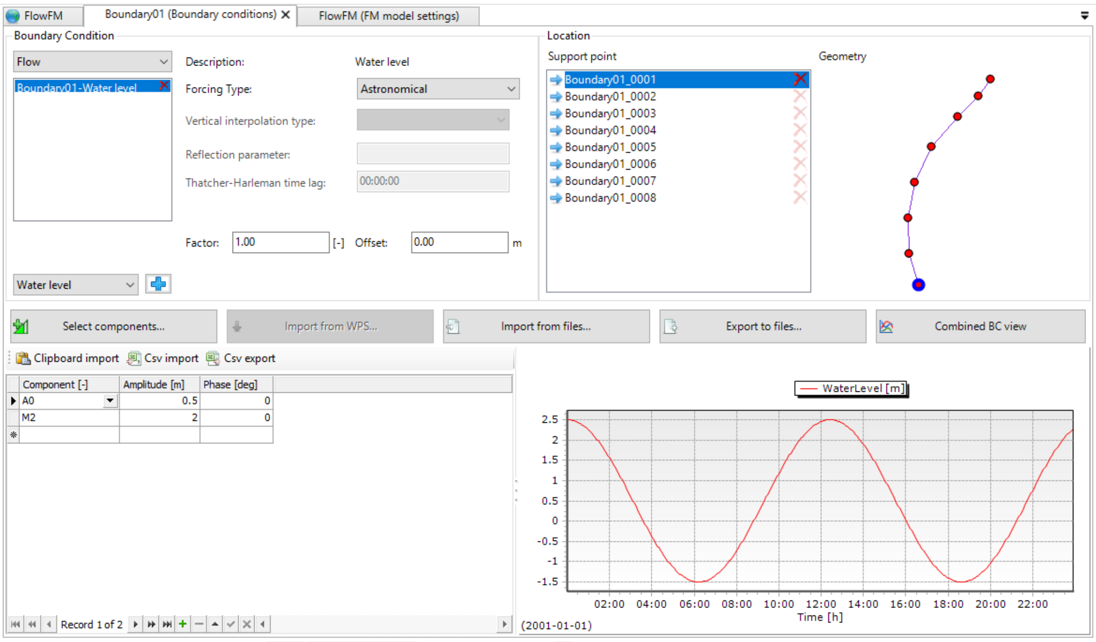


Figure 4.21: *Selection of Boundary01.*

- ➢ Choose *Water level*, press + and choose *Forcing type: Astronomical*.
- ➢ Selected the first boundary support point *Boundary01_0001* by clicking on the plus-sign behind it, repeat this for the last boundary support point.
- ➢ Choose *Select components. . .* (at the lefthandside of the screen).
- ➢ Choose the components *A0* and *M2*, and press *OK*,
- ➢ Select *All support points* and press *OK*. All support points will now have components *A0* and *M2* defined.
- ➢ Fill in the amplitudes of the components in the table, see Figure 4.22.



FlowFM | Boundary01 (Boundary conditions) | FlowFM (FM model settings)

Boundary Condition

Flow

Boundary01-Water level

Description: Water level

Forcing Type: Astronomical

Vertical interpolation type:

Reflection parameter:

Thatcher-Harleman time lag: 00:00:00

Factor: 1.00 [-] Offset: 0.00 m

Water level

Location

Support point

- Boundary01_0001
- Boundary01_0002
- Boundary01_0003
- Boundary01_0004
- Boundary01_0005
- Boundary01_0006
- Boundary01_0007
- Boundary01_0008

Component [-]	Amplitude [m]	Phase [deg]
A0	0.5	0
M2	2	0

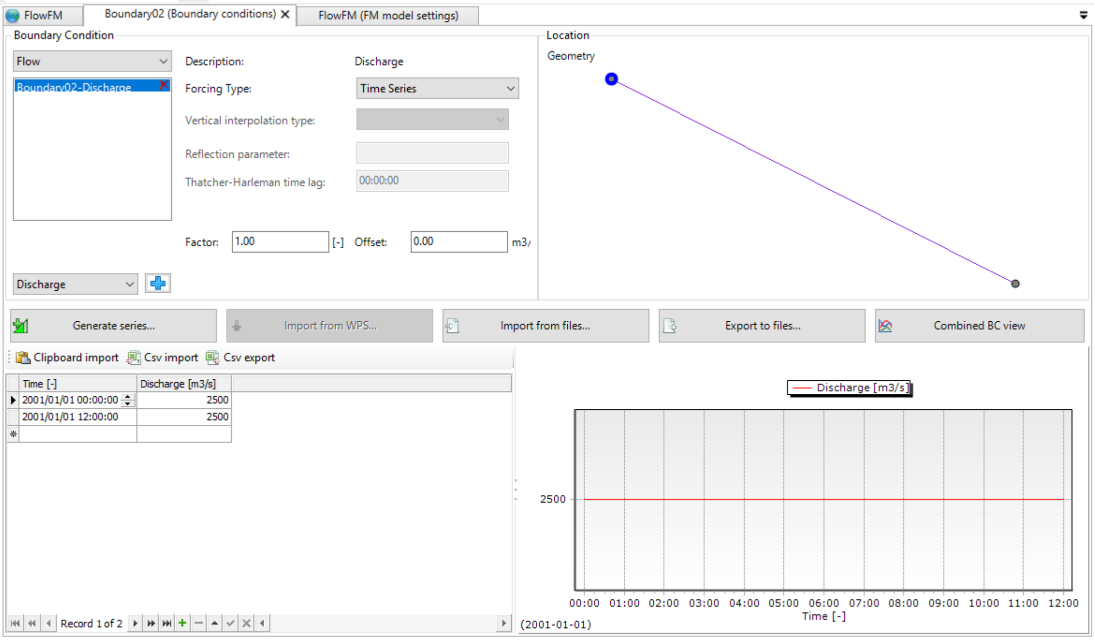
Record 1 of 2

Figure 4.22: *Boundary conditions seaside.*

We have now prescribed an astronomical water level signal, consisting of two components. The first component has an amplitude equal to 0.5 m with a frequency of 0 degrees/hour, i.e. a constant value of 0.5 m+NAP. The second component has an amplitude of 2.5 m with a frequency of 28.98 degrees/hour. Basically, the signal $h(t) = 0.5 + 2.5\ \cos(28.98\ t)$ has been prescribed now, with h in meters w.r.t. the reference level (in this case: NAP) and t in hours.

➢ To impose boundary conditions at the riverboundary, double click at *Boundary02*, Choose *Discharge* and press +. Choose *Forcing type: Time Series*.

➢ To impose a contant discharge of 2500 $\mathrm{m}^3\,\mathrm{s}^{-1}$, fill in the table as in Figure 4.23. Make sure you insert the correct times!



FlowFM | Boundary02 (Boundary conditions) | FlowFM (FM model settings)

Boundary Condition

Flow

Boundary02-Discharge

Description: Discharge

Forcing Type: Time Series

Vertical interpolation type:

Reflection parameter:

Thatcher-Harleman time lag: 00:00:00

Factor: 1.00 [-] Offset: 0.00 m3/

Discharge

Location

Time [-]	Discharge [m3/s]
2001/01/01 00:00:00	2500
2001/01/01 12:00:00	2500

Record 1 of 2

Figure 4.23: *Boundary condition riverside.*

To be able to load this D-Flow FM model in the future, it is necessary to save the model:

- In the project tree, rightclick on *FlowFM* and select *Rename*. Rename the model to “westernscheldt05”.
- Save the model in folder <tutorial05> **of your working environment**: in the project tree rightclick on *westernscheldt05*. Choose *Export* select *Flow Flexible Mesh model*, press *OK* and save the model as <westernscheldt05.mdu>.

The model can now be loaded in the next tutorial exercise without the need of inserting all network files or land and flow boundaries individually again.

- Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar. You don’t have to save the project.

4.2.7 Tutorial 6: Defining output locations

Often one would like to monitor computational results at certain specific cross sections or locations (i.e. ’observation points’). In this tutorial exercise we will add cross sections and observation points in a D-Flow FM model.

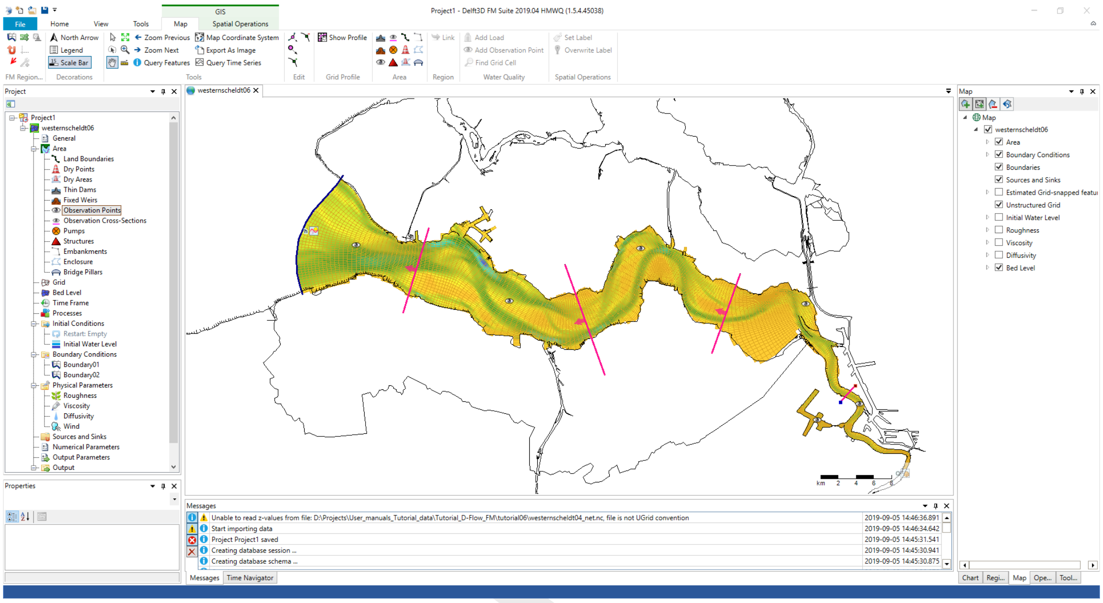
We will start with importing an existing D-Flow FM model without cross sections and observation points:

- Start Delft3D Flexible Mesh Suite and import a D-Flow FM Model by choosing (in the top ribbon) *Import* → *Flow Flexible Mesh Model*.
- Choose <westernscheldt06.mdu> from the folder <tutorial06> and click on *Open*.

Inserting cross sections and observation points is rather straightforward. The following actions are necessary:

Inserting cross sections:

- Choose *Add new observation cross section (2D)* (see the *Area* menu in the ribbon).
- Draw as many cross sections as you like. Finalize each cross section by doubleclicking the left mouse button.
- To see how many cross sections have been made, double click on *Area* → *Observation Cross-Sections* in the project tree, see Figure 4.24.



```
Unable to read z-values from file: D:\Projects\User_manuals_Tutorial_data\Tutorial_D-Flow_FM\tutorial06\westernscheldt04_net.nc, file is not UGrid convention   2019-09-05 14:46:36.891
Start importing data                                                                                                                                              2019-09-05 14:46:34.642
Project Project1 saved                                                                                                                                            2019-09-05 14:45:31.541
Creating database session ...                                                                                                                                     2019-09-05 14:45:30.941
Creating database schema ...                                                                                                                                      2019-09-05 14:45:30.875
```

__Figure 4.24:__ Overview cross sections and observation points.

Now we will add observation points:

- ➢ Choose 👁 *Add new observation point*. Add as many observation points as you like.
- ➢ To see how many observation points have been made, double click on *Area* → *Observation Points* in the project tree.
- ➢ Save the current FM model by clicking the right mouse in the project tree on *westernscheldt06*.
- ➢ Choose *Export*, choose *Flow Flexible Mesh Model* and press *OK*. Then save the model with name "westernscheldt06" in the folder <tutorial06> **of your working environment**. The observation points and cross sections have now been saved in the new <mdu>-file.

☞ **Tip:** It is also possible to save only the observations points. For instance to a different folder. To try this, click the right mouse on *Observation points* → *Export* → *Observation points to* <*xyn*>*-file*. Then, choose a filename and a folder.

- ➢ Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar. You don't have to save the project..

4.2.8 Tutorial 7: Defining computational parameters

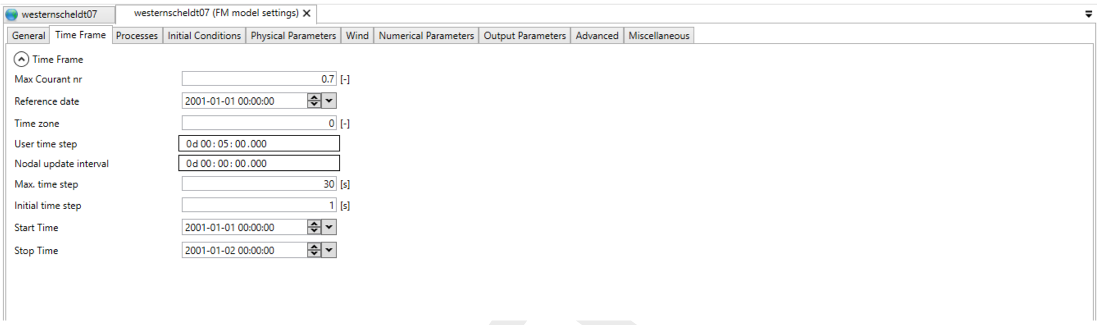
Before we can run a model, all computational parameters need to be set. This involves for example the *Time Frame*, *Initial Conditions* and the *Output Parameters*. Most parameters can be set in parameter screens accessible through the project tree. All other parameters are set by default, and will not be considered in this tutorial exercise.

We will start by importing an existing D-Flow FM model:

- ➢ Start Delft3D Flexible Mesh Suite and import the D-Flow FM Model from the folder <tutorial07>, by choosing *Home* → *Import* → *Flow Flexible Mesh Model* → *OK*. Choose <westernscheldt07.mdu> and press *Open*.

Several parameters will be set. We will start with the *Time Frame*, see Figure 4.25:

- ➢ At the Time Frame section: fill in the parameters shown in Figure 4.25.



Time Frame

Max Courant nr 0.7 [-]

Reference date 2001-01-01 00:00:00

Time zone 0 [-]

User time step 0d 00 : 05 : 00.000

Nodal update interval 0d 00 : 00 : 00.000

Max. time step 30 [s]

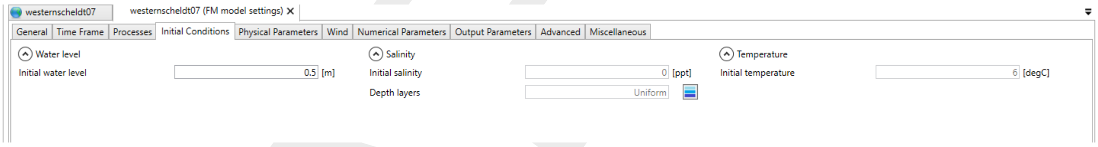
Initial time step 1 [s]

Start Time 2001-01-01 00:00:00

Stop Time 2001-01-02 00:00:00

Figure 4.25: *The time frame of the simulation.*

➢ At the Initial Conditions section: fill in “0.5” m for the Initial water level, see Figure 4.26.



Water level

Initial water level 0.5 [m]

Salinity

Initial salinity 0 [ppt]

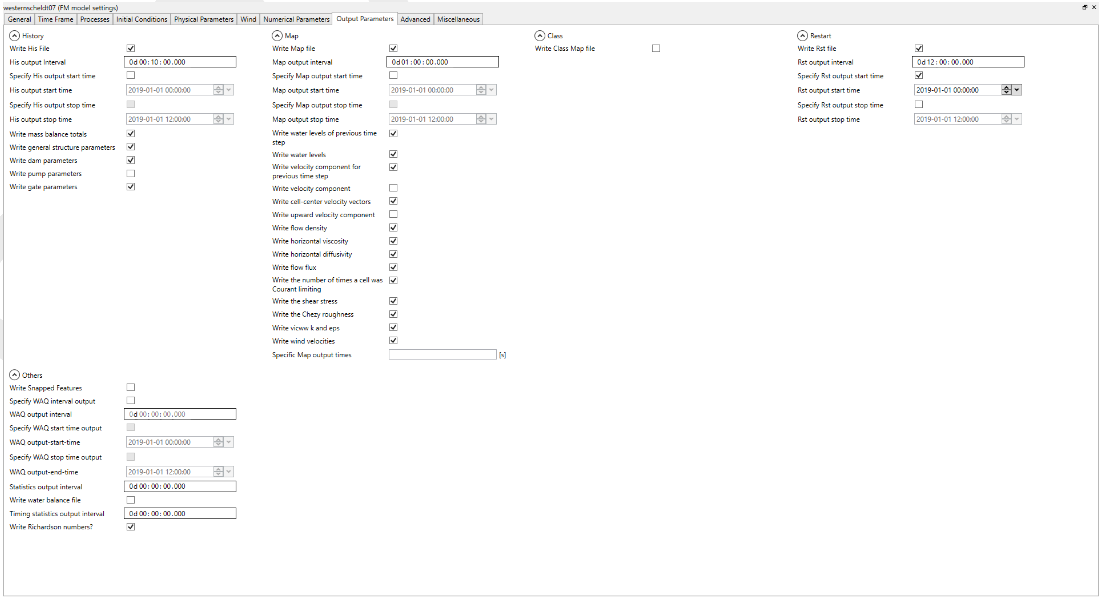
Depth layers Uniform

Temperature

Initial temperature 6 [degC]

Figure 4.26: *Imposed initial conditions for the simulation.*

➢ At the Output Parameters section: fill in the parameters shown in Figure 4.27.



History

Write His File ☑

His output Interval 0d 00 : 10 : 00.000

Specify His output start time ☐

His output start time 2019-01-01 00:00:00

Specify His output stop time ☐

His output stop time 2019-01-01 12:00:00

Write mass balance totals ☑

Write general structure parameters ☑

Write dam parameters ☑

Write pump parameters ☐

Write gate parameters ☑

Map

Write Map file ☑

Map output interval 0d 01 : 00 : 00.000

Specify Map output start time ☐

Map output start time 2019-01-01 00:00:00

Specify Map output stop time ☐

Map output stop time 2019-01-01 12:00:00

Write water levels of previous time step ☑

Write water levels ☑

Write velocity component for previous time step ☑

Write velocity component ☐

Write cell-center velocity vectors ☑

Write upward velocity component ☐

Write flow density ☑

Write horizontal viscosity ☑

Write horizontal diffusivity ☑

Write flow flux ☑

Write the number of times a cell was Courant limiting ☑

Write the shear stress ☑

Write the Chezy roughness ☑

Write vicww k and eps ☑

Write wind velocities ☑

Specific Map output times [s]

Class

Write Class Map file ☐

Restart

Write Rst file ☑

Rst output interval 0d 12 : 00 : 00.000

Specify Rst output start time ☑

Rst output start time 2019-01-01 00:00:00

Specify Rst output stop time ☐

Rst output stop time 2019-01-01 12:00:00

Others

Write Snapped Features ☐

Specify WAQ interval output ☐

WAQ output interval 0d 00 : 00 : 00.000

Specify WAQ start time output ☐

WAQ output-start-time 2019-01-01 00:00:00

Specify WAQ stop time output ☐

WAQ output-end-time 2019-01-01 12:00:00

Statistics output interval 0d 00 : 00 : 00.000

Write water balance file ☐

Timing statistics output interval 0d 00 : 00 : 00.000

Write Richardson numbers? ☑

Figure 4.27: *Optional output parameters for the computation.*

As an explanation, the computation will generate history-files and map-files:

- ◇ In history-files, timeseries are stored at the cross-sections and observation locations, at a frequency specified via the parameter *His Output Interval*.
- ◇ The map-files collect data over the entire domain, at a frequency specified via the parameter *Map Output Interval*.

  Note: Be aware that these periods are clipped by the parameter *User time step*, which has been specified under *Time Frame*. That means, if *User time step* > *His Output Interval*, then the period with which his-files are written is given by *User time step*.
- ◇ Restart files will be written when *Write Rst File* is selected.

The period with which these files are written can then be specified at *Rst Output Interval*. This period is not clipped by *User time step*. To start a computation with a restart file is not part of this tutorial.

➢ Save the current D-Flow FM model in the folder <tutorial07> of **your working environment** by rightmouseclicking in the project tree on *westernscheldt07*, choose *Export* and save the model as *Flow Flexible Mesh model* <westernscheldt07.mdu>.

The computational parameters have now been saved in the <mdu>-file.

➢ Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar.

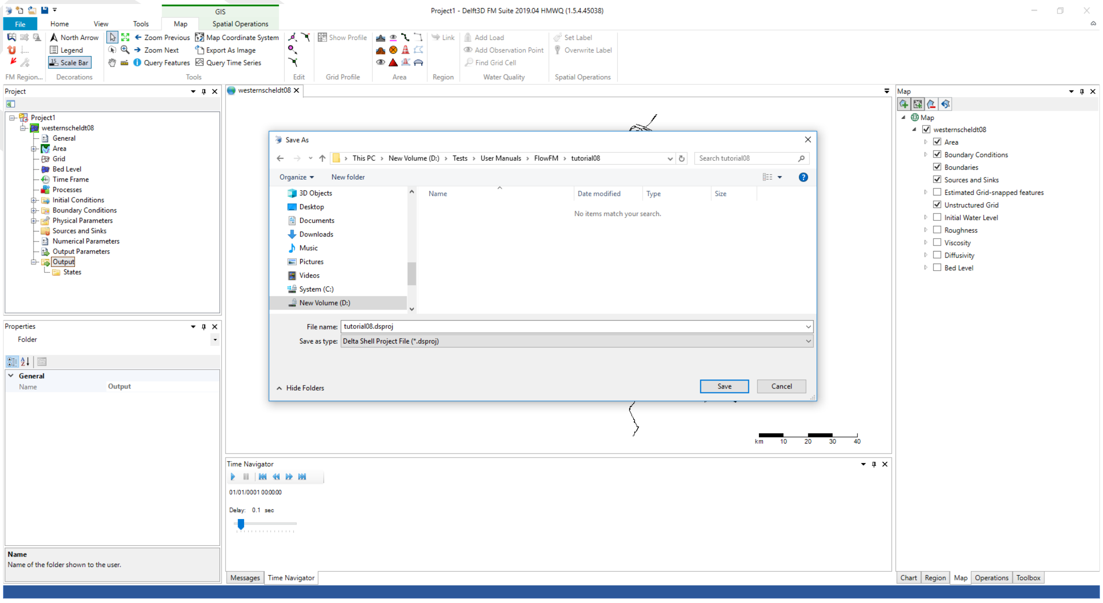
4.2.9 Tutorial 8: Running a model simulation

Up till now we have saved the D-Flow FM model by means of a <mdu>-file. For running computations, it is necessary to save the entire project, which will be done in this tutorial exercise:

➢ Import the D-Flow FM Model from the folder <tutorial08>, by choosing *Import* → *Flow Flexible Mesh Model* → *OK*. Choose <westernscheldt08.mdu> and *Open*.

For saving the entire project, the following should be done:

➢ Choose *File* → *Save As* to save <tutorial08.dsproj> in the folder <tutorial08> of **your working environment** and click *Save*.
The project has now been saved in a folder called <tutorial08\tutorial08.dsproj_data> and a project file <tutorial08\tutorial08.dsproj> has been created (See Figure 4.28). Within this folder you will find all input files of the model (some are ASCII), output files of the model (when it has been run) and, if applicable, zip-folders containing the restart files. Note that after saving, the project has been renamed to *tutorial08*



No items match your search.

File name: tutorial08.dsproj
Save as type: Delta Shell Project File (*.dsproj)

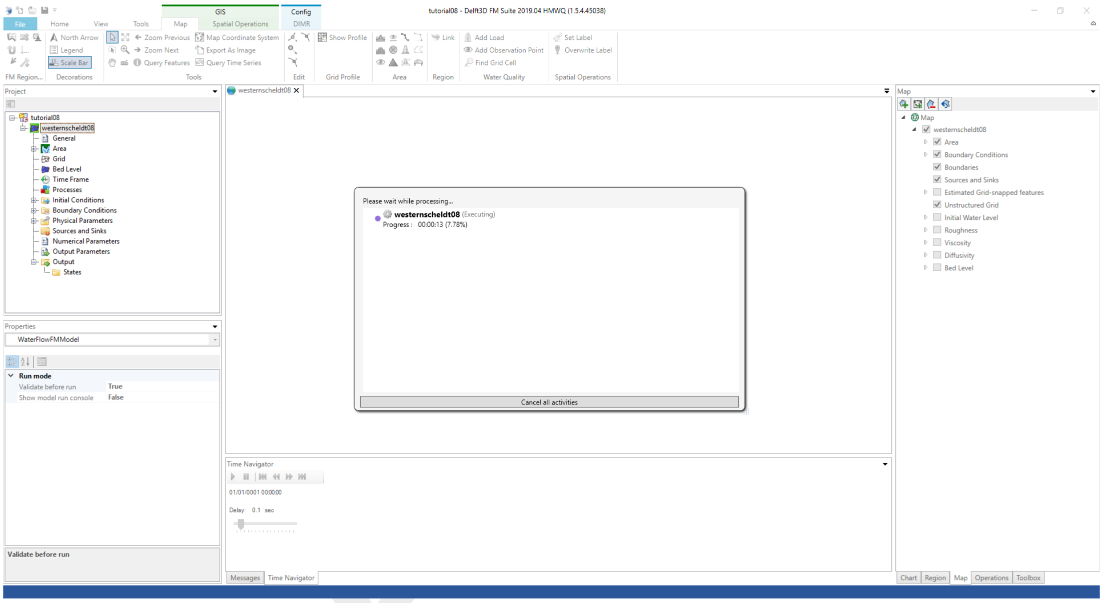
Figure 4.28: *Menu for saving a project.*

To start a computation, the following needs to be done:

➢ In the project tree select the model you want to run (which has been named *westernscheldt08*, see Figure 4.29).

- ➢ Press the right mouse and choose *Run Model*

The simulation will now start running.



westernscheldt08

Please wait while processing...

westernscheldt08 (Executing)
Progress : 00:00:13 (7.78%)

Run mode
Validate before run True
Show model run console False

Validate before run

Figure 4.29: *View of Delft3D Flexible Mesh Suite when running a model.*

Now:

- ➢ Save the project with generated output in the folder <tutorial08> **of your working environment** (*File* → *Save As* and save the project with name “tutorial08.dsproj”). Press *Yes* to overwrite the project.
- ➢ Now close the current D-Flow FM model (In the project bar, rightmouseclick on *westernscheldt08* → *Delete* → *OK*).
- ➢ Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar.

Folder <tutorial08\finished> contains the output for this tutorial exercise that has been prepared by Deltares.

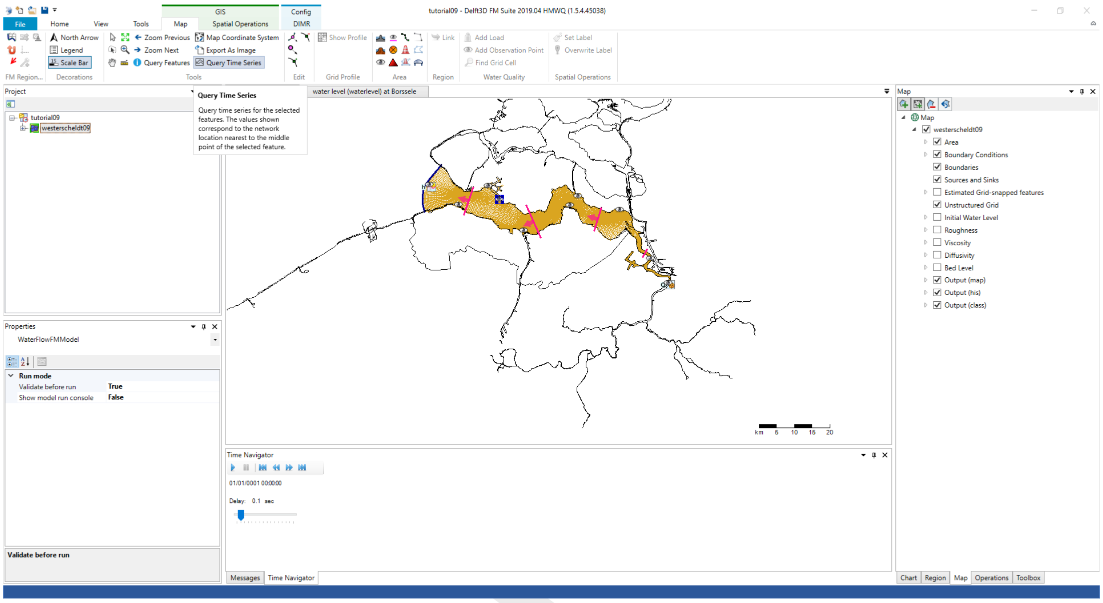
4.2.10 Tutorial 9: Viewing the output of a model simulation

The output of the model can be observed within Delft3D Flexible Mesh Suite. At first a project has to be imported:

- ➢ Choose *File* → *Open*. Select the file <tutorial9.dsproj> from the folder <tutorial09> and choose *Open*. The project is now activated.
- ➢ Doubleclick on *westernscheldt09*.

To view the output of the history files (observation points and cross-sections):

- ➢ Click with leftmousebutton on an observation point (while being in *select* mode. Your mouse looks like an arrow in this mode).
- ➢ Choose at the top part of the screen *Query Time Series*, see Figure 4.30 (option available within the Map-ribbon).



Query Time Series

Query time series for the selected features. The values shown correspond to the network location nearest to the middle point of the selected feature.

Properties

WaterFlowFMModel

Run mode

Validate before run	True
Show model run console	False

Validate before run

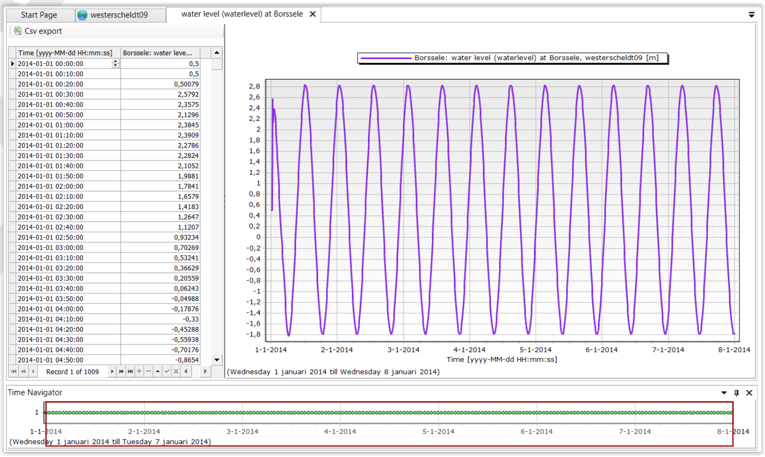
Time Navigator

01/01/0001 00:00:00

Delay: 0.1 sec

Figure 4.30: *View of Delft3D Flexible Mesh Suite, available to select a location for time-series in.*

- ➢ Choose *Water level (waterlevel)* → *OK*. You can now observe the water levels in the observation point you selected, see Figure 4.31. You can choose other parameters as well to observe. Also cross-sections can be observed this way.
- ➢ Select the Time Navigator in the *View* section of the ribbon. The Time Navigator will appear at the bottom of Delft3D Flexible Mesh Suite.


Start Page
westerscheldt09
water level (waterlevel) at Borssele
Csv export
Borssele: water level (waterlevel) at Borssele, westerscheldt09 [m]
2,8
2,6
2,4
2,2
2
1,8
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
-0,2
-0,4
-0,6
-0,8
-1
-1,2
-1,4
-1,6
-1,8
1-1-2014
2-1-2014
3-1-2014
4-1-2014
5-1-2014
6-1-2014
7-1-2014
8-1-2014
Time [yyyy-MM-dd HH:mm:ss]
(Wednesday 1 januari 2014 till Wednesday 8 januari 2014)
Time Navigator
1
(Wednesday 1 januari 2014 till Tuesday 7 januari 2014)

Time [yyyy-MM-dd HH:mm:ss]	Borssele: water leve...
2014-01-01 00:00:00	0,5
2014-01-01 00:10:00	0,5
2014-01-01 00:20:00	0,50079
2014-01-01 00:30:00	2,5792
2014-01-01 00:40:00	2,3575
2014-01-01 00:50:00	2,1296
2014-01-01 01:00:00	2,3845
2014-01-01 01:10:00	2,3909
2014-01-01 01:20:00	2,2786
2014-01-01 01:30:00	2,2824
2014-01-01 01:40:00	2,1052
2014-01-01 01:50:00	1,9881
2014-01-01 02:00:00	1,7841
2014-01-01 02:10:00	1,6579
2014-01-01 02:20:00	1,4183
2014-01-01 02:30:00	1,2647
2014-01-01 02:40:00	1,1207
2014-01-01 02:50:00	0,93234
2014-01-01 03:00:00	0,70269
2014-01-01 03:10:00	0,53241
2014-01-01 03:20:00	0,36629
2014-01-01 03:30:00	0,20559
2014-01-01 03:40:00	0,06243
2014-01-01 03:50:00	-0,04988
2014-01-01 04:00:00	-0,17876
2014-01-01 04:10:00	-0,33
2014-01-01 04:20:00	-0,45288
2014-01-01 04:30:00	-0,55938
2014-01-01 04:40:00	-0,70176
2014-01-01 04:50:00	-0,8654

Record 1 of 1009

Figure 4.31: *View of Delft3D Flexible Mesh Suite, time-series for observation point "water level at Borsele".*

To view the output of the map files:

- ➢ Select the pane 'westernscheldt09'
- ➢ Choose in the Map-tree *Add New Wms Layer* (see Figure 4.32) and choose <Open Street Map>.

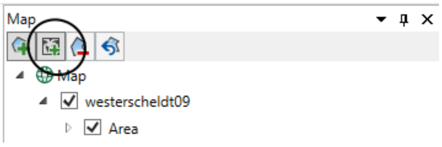


Figure 4.32: *WMS layer icon at the top of the map-tree viewer.*

Now, the model can be observed with OpenStreetMap in the background. See Figure 4.33.

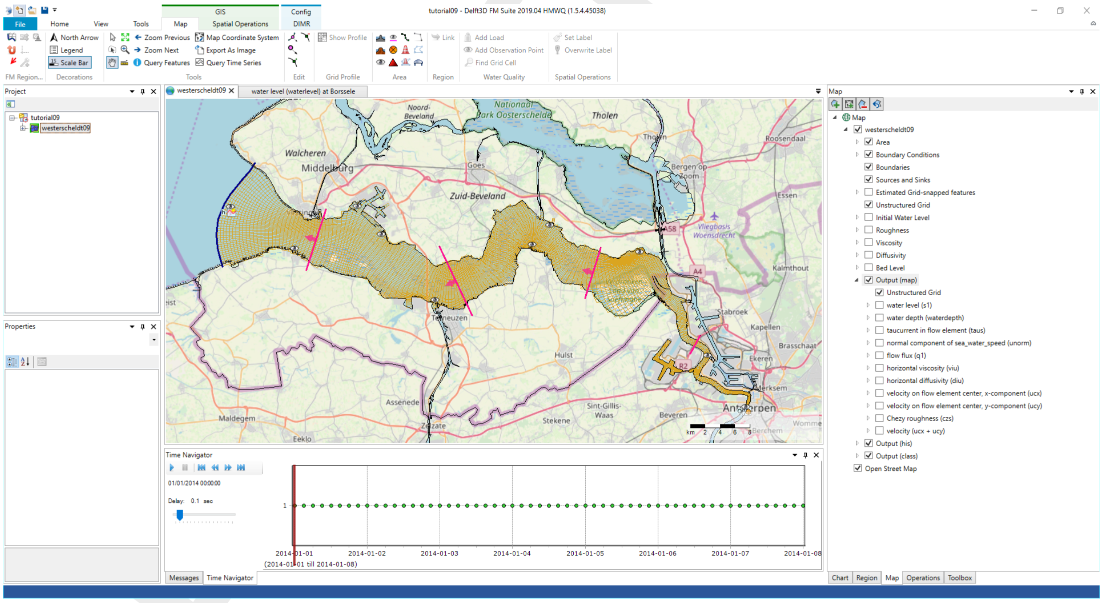


Figure 4.33: *View of Delft3D Flexible Mesh Suite in combination with OpenStreetMap.*

➢ Open *Output* in the map tree. Check *waterlevel(s1)* to observe water levels, see Figure 4.34.

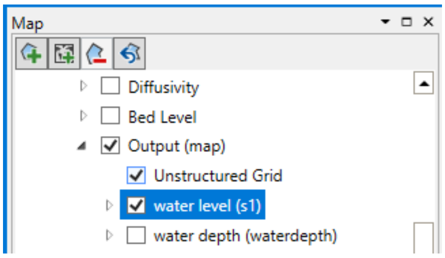


Figure 4.34: *Select* waterlevel(s1) *from the map tree*

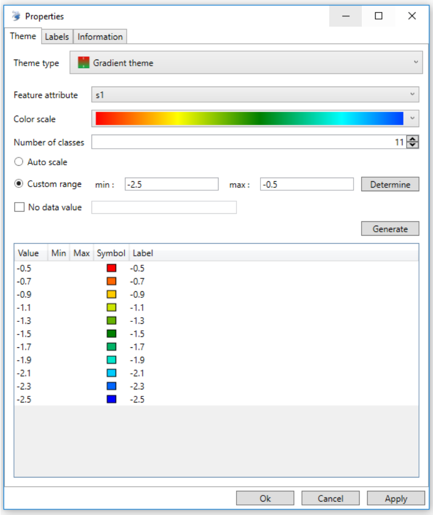
➢ To adjust the legend, rightmouseclick on *Output(map)* →*waterlevel (s1)* and select *Properties*. Now the window **Layer Properties Editor** appears in which the legend can be adjusted.

➢ Set classes to “11”

➢ Select *Custom range* and set the legend values for minimum and maximum value to “-2.5” to “-0.5”.

➢ Press *Generate*

➢ Press *OK*, see Figure 4.35.



Theme type: Gradient theme

Feature attribute: s1

Color scale

Number of classes: 11

○ Auto scale

● Custom range min : -2.5 max : -0.5

☐ No data value

Value	Min	Max	Symbol	Label
-0.5			🟥	-0.5
-0.7			🟧	-0.7
-0.9			🟨	-0.9
-1.1			🟨	-1.1
-1.3			🟩	-1.3
-1.5			🟩	-1.5
-1.7			🟩	-1.7
-1.9			🟦	-1.9
-2.1			🟦	-2.1
-2.3			🟦	-2.3
-2.5			🟦	-2.5

Figure 4.35: *Layer properties editor*

- ➢ Click the play button in the *Time Navigator* to see the water levels change in time. To see the water depths, uncheck the water level in the map tree and check the water depth.
- ➢ Now close the current D-Flow FM model (In the project bar, rightmouseclick on *westernscheldt09* → *Delete* → *OK*).
- ➢ Exit Delft3D Flexible Mesh Suite by clicking the close icon in the menu bar.

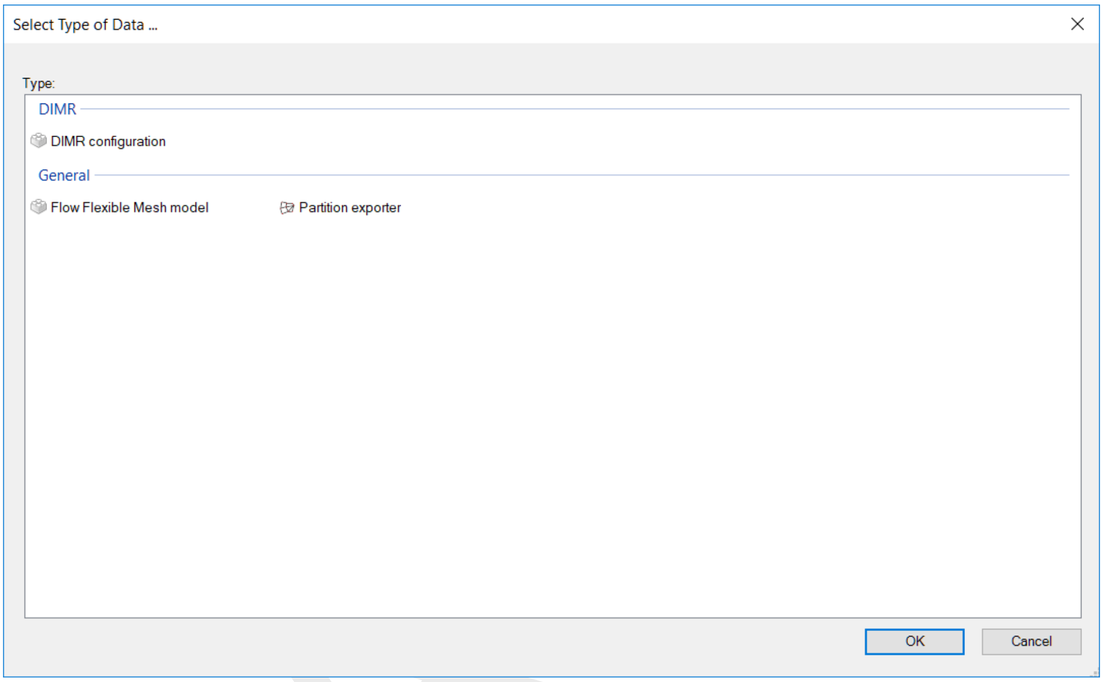
4.2.11 Tutorial 10: Exporting a model

A model must be exported to be able to run it outside D-Flow FM GUI, see section 6.3.

- ➢ Import the D-Flow FM Model from the folder <tutorial10>, by choosing *Import* → *Flow Flexible Mesh Model* → *OK*. Choose <westernscheldt10.mdu> and *Open*.

To export:

- ➢ Rightmouseclick on *Westernscheldt10* in the project view on the left. Choose *Export* → *DIMR configuration* → *OK*, see Figure 4.36. Select an empty directory to store the resulting files, <tutorial10\workdir>, and choose a name for the DIMR configfile, default <dimr_config.xml>. Directory <tutorial10\workdir> will now contain file <dimr_config.xml> and subdirectory <dflowfm>. Besides some cleaning, the files inside subdirectory <dflowfm> should be the same as in <tutorial10\input>.


Select Type of Data ...

Type:

DIMR

DIMR configuration

General

Flow Flexible Mesh model

Partition exporter

Figure 4.36: *Menu for exporting a project.*

Folder <tutorial10\output> contains the output for this tutorial exercise that has been prepared by Deltares. Differences may appear related to changes in execution date and time, and using another version of the GUI.

4.2.12 Tutorial 11: Partitioning a model

A model must be partitioned before it can be run in parallel by using MPI, see section 6.4.2. You can skip this tutorial if you don't want to run in parallel. This tutorial does not use the GUI.

To partition:

- Copy folder <tutorial11\input> to <tutorial11\workdir>.
- Create a new file using a text editor, containing the following single line (Windows):

  ```
  call "<installDir>\x64\dflowfm\scripts\run_dflowfm.bat"
           "--partition:ndomains=3:icgsolver=6" westernscheldt11.mdu
  ```

  or (Linux):

  ```
  <installDir>/lnx64/bin/run_dflowfm.sh
            --partition:ndomains=3:icgsolver=6 westernscheldt11.mdu
  ```

  where `<installDir>` refers to your Delft3D installation directory.
- Save the file in folder <tutorial11\workdir\dflowfm>
  Windows:
  Extension *.bat* is common, e.g. <run_partitioner.bat>. Double-quotes are necessary around the executable name (if the path contains white spaces) and each argument (if it contains the equal sign).
  Linux:
  Extension *.sh* is common, e.g. <run_partitioner.sh>. It's a good habit to start the file with an interpreter reference, e.g. `#!/bin/bash`. Be sure that the line endings of your script file are conform Linux ("line feed" only). You can use the Linux command `dos2unix <scriptfile>` to convert it.
- Execute your `run_partitioner` script.

Remarks:

- ◇ Example models, with run scripts, are included in the (open source) code. After registering yourself at https://oss.deltares.nl/, you can have a look at:
  https://svn.oss.deltares.nl/repos/delft3d/trunk/examples
  /12_dflowfm/test_data/e02_f14_c040_westerscheldt
- ◇ The following text block is a subset of the text echoed to the command box via stdout (version numbers and dates may differ). The full stdout text is saved in file <tutorial11\output\dflowfm\run_partitioner_stdout.log>:

```
call "c:\Deltares\x64\dflowfm\scripts\run_dflowfm.bat" "--partition:ndomains=3:icgsolver=6" westernscheldt11.mdu
OMP_NUM_THREADS is 2
Working directory: c:\checkouts\ds\doc\user_manuals_tutorial_data\Tutorial_D-Flow_FM\tutorial11\workdir\dflowfm
D3D_HOME         : c:\Deltares\x64\dflowfm\scripts\..\..\..
ARCH             : x64
executing: "c:\Deltares\x64\dflowfm\bin\dflowfm-cli.exe" --nodisplay --autostartstop "--partition:ndomains=3:icgsolver=6" westernscheldt11.mdu
* Deltares, D-Flow FM Version 1.2.94.66020M, Feb 20 2020, 09:26:11
** WARNING: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
** WARNING: readMDUFile: [physics] effectspiral=0 was in file, but not used. Check possible typo.
** WARNING: readMDUFile: [output] wrishp_enc=0 was in file, but not used. Check possible typo.
** INFO   : Network UGRID-File: No 1D-Mesh Present, Skipped 1D
ug_get_meshgeom, #12, ierr=0
** WARNING: Could not read net cells from netCDF file 'westernscheldt04_net.nc' (is not critical). Details follow:

gk_mcore statistics
coresize:       100432         nmops:          2048  cmop:       0
num_callocs:          249   num_hallocs:            0
size_callocs:       831160  size_hallocs:            0
cur_callocs:            0   cur_hallocs:            0
max_callocs:       100280   max_hallocs:            0
nbrpool statistics
nbrpoolsize:            0   nbrpoolcpos:            0
nbrpoolreallocs:            0

** INFO   : added node-based ghostcells:             25
** WARNING: 1 hanging 2D links disregarded.
** INFO   : added node-based ghostcells:             18
** INFO   : added node-based ghostcells:             43
```

Folder <tutorial11\output\dflowfm> contains the resulting `run_partitioner` scripts (containing paths that fit to the installation directories used at Deltares), Windows and Linux, and the output for this tutorial exercise:

- ◇ Three additional mdu-files:
  <westernscheldt11_0000.mdu>
  <westernscheldt11_0001.mdu>
  <westernscheldt11_0002.mdu>
- ◇ Three additional net.nc-files:
  <westernscheldt04_0000_net.nc>
  <westernscheldt04_0001_net.nc>
  <westernscheldt04_0002_net.nc>
- ◇ File <DFM_interpreted_idomain_westernscheldt04_net.nc> containing partition domain information and cell information
- ◇ Diagnosis file <DFM_OUTPUT_westernscheldt11\westernscheldt11.dia>
- ◇ File <tutorial11\output\dflowfm\run_partitioner_stdout.log> containing the full text echoed to the command box via stdout (version numbers and dates may differ).

4.2.13 Tutorial 12: Running a model using a batch script

This tutorial explains how to run D-Flow FM computations outside the GUI, using a batch script. The resulting files of section 4.2.12 are used as input. At first, the model is run sequentially (without parallelMPI), both on Windows and Linux. Next, the model is run with parallelMPI (Windows and Linux). See section 6.3 for more information.

Remark:

- ◇ Performing a computation in parallel is intended to speed-up the computation. But when using limited resources for a small testcase (e.g. this tutorial), a parallel computation might need *more* time. See also section 6.5.

Preparation:

➢ Copy folder <tutorial12\input> to <tutorial12\workdir>.

Folder <tutorial12\output> contains the resulting files of all four runs:

- ◇ Sequential on Windows
  - <dimr_config.xml>, as copied from the input folder
  - <run_model.bat>, as created in section 4.2.13.1
  - <run_model_bat_stdout.log>, containing the messages written to the command box via stdout during the simulation
  - <dflowfm\DFM_OUTPUT_westernscheldt12_sequential> containing all the output files.
- ◇ Sequential on Linux
  - <dimr_config.xml>, as copied from the input folder
  - <run_model.sh>, as created in section 4.2.13.2
  - <run_model_sh_stdout.log>, containing the messages written to the command box via stdout during the simulation
  - <dflowfm\DFM_OUTPUT_westernscheldt12_sequential> containing all the output files (produced on Windows).
- ◇ Parallel on Windows
  - <dimr_config_parallel.xml>, as created in section 4.2.13.3
  - <run_model_parallel.bat>, as created in section 4.2.13.3
  - <run_model_parallel_bat_stdout.log>, containing the messages written to the command box via stdout during the simulation
  - <dflowfm\DFM_OUTPUT_westernscheldt12_parallel> containing all the output files.
- ◇ Parallel on Linux
  - <dimr_config_parallel.xml>, as created in section 4.2.13.4
  - <run_model_parallel.sh>, as created in section 4.2.13.4
  - <run_model_parallel_sh_stdout.log>, containing the messages written to the command box via stdout during the simulation
  - <dflowfm\DFM_OUTPUT_westernscheldt12_parallel> containing all the output files (produced on Windows).

4.2.13.1 Sequential on Windows

➢ Create a new file using a text editor, containing the following single line:
`call "<installDir>\x64\dimr\scripts\run_dimr.bat" dimr_config.xml`
where `<installDir>` refers to your Delft3D installation directory.

➢ Save the file in folder <tutorial12\workdir>
Extension *.bat* is common, e.g. <run_model.bat>. Double-quotes are necessary around the executable name (if the path contains white spaces).

➢ Execute your `run_model` script.

The following text block is a subset of the text echoed to the command box via stdout (version numbers and dates may differ). The full stdout text is saved in file <tutorial12\output\run_model_bat_stdout.log>:

```
call "c:\Deltares\x64\dimr\scripts\run_dimr.bat" dimr_config.xml
Configfile:dimr_config.xml
OMP_NUM_THREADS is 2
Working directory: c:\Tutorial_D-Flow_FM\tutorial12\workdir
executing: "c:\Deltares\x64\dimr\scripts\..\..\..\x64\dimr\bin\dimr.exe"  dimr_config.xml
Dimr [2020-04-28 17:46:14.198] #0 >> Deltares, DIMR_EXE Version 2.00.00.66020M, Feb 20 2020, 09:06:14
Dimr [2020-04-28 17:46:14.225] #0 >>
Dimr [2020-04-28 17:46:14.225] #0 >> Version Information of Components
Dimr [2020-04-28 17:46:14.227] #0 >> ================================
Dimr [2020-04-28 17:46:14.228] #0 >> dflowfm                           : 1.2.94.66020M
Dimr [2020-04-28 17:46:14.229] #0 >> --------------------------------
Dimr [2020-04-28 17:46:14.229] #0 >>
Dimr [2020-04-28 17:46:14.233] #0 >> westernscheldt10.Initialize(westernscheldt12.mdu)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
loading model
Dimr [2020-04-28 17:46:14.370] #0 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:46:14.372] #0 >> kernel: readMDUFile: [physics] effectspiral=0 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:46:14.373] #0 >> kernel: readMDUFile: [output] wrishp_enc=0 was in file, but not used. Check possible typo.
ug_get_meshgeom, #12, ierr=0
** INFO   : Opened file : westernscheldt05.ldb
** INFO   : Closed file : westernscheldt05.ldb
** INFO   : Opened file : westernscheldt06_obs.xyn
** INFO   : Closed file : westernscheldt06_obs.xyn
** INFO   : Opened file : westernscheldt06crs_crs.pli
** INFO   : Closed file : westernscheldt06crs_crs.pli
model loaded
Initializing flow: flow_modelinit
** INFO   : Initializing flow model geometry...
** INFO   : Done writing initial output to file(s).
Dimr [2020-04-28 17:46:14.913] #0 >> westernscheldt10.Update(43200.0)
Dimr [2020-04-28 17:46:23.950] #0 >> westernscheldt10.Finalize()
** INFO   :
** INFO   : Computation started  at: 17:46:14, 28-04-2020
** INFO   : Computation finished at: 17:46:24, 28-04-2020
** INFO   :
** INFO   : simulation period      (h)  :            12.0000000000
** INFO   : total time in timeloop (h)  :             0.0025600000
** INFO   : MPI    : no.
** INFO   : OpenMP : yes.         #threads max : 2
** INFO   :
** INFO   :
** INFO   :
** INFO   : Opened file : DFM_OUTPUT_westernscheldt12\westernscheldt12_numlimdt.xyz
** INFO   : Closed file : DFM_OUTPUT_westernscheldt12\westernscheldt12_numlimdt.xyz
** INFO   :
Dimr [2020-04-28 17:46:24.048] #0 >> TIMER INFO:

Dimr [2020-04-28 17:46:24.048] #0 >> westernscheldt10   : 9.999813 sec
```

4.2.13.2 Sequential on Linux

➢ Create a new file using a text editor, containing the following single line:
`<installDir>/lnx64/bin/run_dimr.sh dimr_config.xml`
where `<installDir>` refers to your Delft3D installation directory.

➢ Save the file in folder <tutorial12\workdir>
Extension *.sh* is common, e.g. <run_model.sh>. It's a good habit to start the file with an interpreter reference, e.g. `#!/bin/bash`. Be sure that the line endings of your script file are conform Linux ("line feed" only). You can use the Linux command `dos2unix <scriptfile>` to convert it.

➢ Execute your `run_model` script.

The following text block is a subset of the text echoed to the command box via stdout (version numbers and dates may differ). The full stdout text is saved in file

<tutorial12\output\run_model_sh_stdout.log>:

```
OMP_NUM_THREADS is 2
Configfile       : dimr_config.xml
D3D_HOME         : /opt/delft3d/lnx64
Working directory: /p/tutorial12/workdir
Number of slots  : 1

executing:
/opt/delft3d/lnx64/bin/dimr dimr_config.xml
Dimr [2020-04-28 17:53:43.80130] #0 >> Deltares, DIMR_EXE Version 2.00.00.66020, Feb 19 2020, 18:57:44
Dimr [2020-04-28 17:53:43.161476] #0 >>
Dimr [2020-04-28 17:53:43.161494] #0 >> Version Information of Components
Dimr [2020-04-28 17:53:43.161497] #0 >> ================================
Dimr [2020-04-28 17:53:43.165592] #0 >> dflowfm                          : 1.2.94.66020
Dimr [2020-04-28 17:53:43.165605] #0 >> ---------------------------------
Dimr [2020-04-28 17:53:43.165609] #0 >>
Dimr [2020-04-28 17:53:43.165633] #0 >> westernscheldt10.Initialize(westernscheldt12.mdu)
** DEBUG  :             0  0.1000010      NOD(KMAX)
** DEBUG  :             6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
loading model
Dimr [2020-04-28 17:53:43.212208] #0 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:53:43.212220] #0 >> kernel: readMDUFile: [physics] effectspiral=0 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:53:43.212233] #0 >> kernel: readMDUFile: [output] wrishp_enc=0 was in file, but not used. Check possible typo.
** INFO   : Done writing initial output to file(s).
Dimr [2020-04-28 17:53:43.846829] #0 >> westernscheldt10.Update(43200.0)
Dimr [2020-04-28 17:53:55.290457] #0 >> westernscheldt10.Finalize()
** INFO   :
** INFO   : Computation started  at: 17:53:43, 28-04-2020
** INFO   : Computation finished at: 17:53:55, 28-04-2020
** INFO   :
** INFO   : simulation period      (h)  :            12.0000000000
** INFO   : total time in timeloop (h)  :             0.0032725000
** INFO   : MPI    : no.
** INFO   : OpenMP : unavailable.
** INFO   :
** INFO   :
** INFO   :
** INFO   : Opened file : DFM_OUTPUT_westernscheldt12/westernscheldt12_numlimdt.xyz
** INFO   : Closed file : DFM_OUTPUT_westernscheldt12/westernscheldt12_numlimdt.xyz
** INFO   :
Dimr [2020-04-28 17:53:55.313464] #0 >> TIMER INFO:

Dimr [2020-04-28 17:53:55.313478] #0 >> westernscheldt10       : 12.147715 sec
```

4.2.13.3 Parallel on Windows

- Check that *hydra_service* is running as a service on your machine, see section 6.4.3.
- Check that your model is partitioned correctly, see section 4.2.12. In this example the model is partitioned in three parts.
- Open file <dimr_config.xml> in a text editor, add the following two lines to the *<component>*-group:
  `<mpiCommunicator>DFM_COMM_DFMWORLD</mpiCommunicator>`
  `<process>0 1 2</process>`
  where the process numbers must match exactly with the number of partitions. Save the file as <dimr_config_parallel.xml>.
  This step is identical for Linux and Windows.
- Create a new file using a text editor, containing the following single line:
  `call "<installDir>\x64\dimr\scripts\run_dimr_parallel.bat" 3 dimr_config_parallel.xml`
  where `<installDir>` refers to your Delft3D installation directory.
- Save the file in folder <tutorial12\workdir>
  Extension *.bat* is common, e.g. <run_model_parallel.bat>. Double-quotes are necessary around the executable name (if the path contains white spaces).
- Execute your `run_model_parallel` script.

The following text block is a subset of the text echoed to the command box via stdout (version numbers and dates may differ). The full stdout text is saved in file <tutorial12\output\run_model_parallel_bat_stdout.log>.
The "Access is denied" error messages are generated by the operating system and can be neglected:

```
c:\Deltares\x64\dimr\scripts\run_dimr_parallel.bat" 3 dimr_config_parallel.xml
Configfile:dimr_config_parallel.xml
```

```
OMP_NUM_THREADS is 2
number of partitions: 3
Working directory: c:\tutorial12\workdir
executing: "c:\Deltares\x64\share\bin\mpiexec.exe" -n 3 -localonly "c:\Deltares\x64\dimr\bin\dimr.exe"  dimr_config_parallel.xml
#1: Running parallel with 3 partitions
#0: Running parallel with 3 partitions
#2: Running parallel with 3 partitions
Dimr [2020-04-28 17:48:31.383] #1 >> Deltares, DIMR_EXE Version 2.00.00.66020M, Feb 20 2020, 09:06:14
Dimr [2020-04-28 17:48:31.383] #2 >> Deltares, DIMR_EXE Version 2.00.00.66020M, Feb 20 2020, 09:06:14
Dimr [2020-04-28 17:48:31.383] #0 >> Deltares, DIMR_EXE Version 2.00.00.66020M, Feb 20 2020, 09:06:14
Dimr [2020-04-28 17:48:31.428] #0 >>
Dimr [2020-04-28 17:48:31.428] #0 >> Version Information of Components
Dimr [2020-04-28 17:48:31.428] #0 >> =================================
Dimr [2020-04-28 17:48:31.428] #0 >> dflowfm                           : 1.2.94.66020M
Dimr [2020-04-28 17:48:31.428] #0 >> ---------------------------------
Dimr [2020-04-28 17:48:31.428] #0 >>
Dimr [2020-04-28 17:48:31.429] #2 >> westernscheldt10.Initialize(westernscheldt12.mdu)
Dimr [2020-04-28 17:48:31.429] #1 >> westernscheldt10.Initialize(westernscheldt12.mdu)
Dimr [2020-04-28 17:48:31.429] #0 >> westernscheldt10.Initialize(westernscheldt12.mdu)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
loading model
loading model
loading model
** INFO   : no partitioning polygons file: read subdomain numbering from netfiles
** INFO   : no partitioning polygons file: read subdomain numbering from netfiles
** INFO   : no partitioning polygons file: read subdomain numbering from netfiles
Dimr [2020-04-28 17:48:31.585] #1 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:48:31.585] #0 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:48:31.586] #2 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
model loaded
model loaded
model loaded
Initializing flow: flow_modelinit
Initializing flow: flow_modelinit
Initializing flow: flow_modelinit
** INFO   : Initializing flow model geometry...
** INFO   : Initializing flow model geometry...
** INFO   : Initializing flow model geometry...
** INFO   : Modelinit finished   at: 17:48:31, 28-04-2020
** INFO   : Modelinit finished   at: 17:48:31, 28-04-2020
** INFO   : Modelinit finished   at: 17:48:31, 28-04-2020
Dimr [2020-04-28 17:48:31.918] #1 >> westernscheldt10.Update(43200.0)
Dimr [2020-04-28 17:48:31.927] #2 >> westernscheldt10.Update(43200.0)
Dimr [2020-04-28 17:48:31.943] #0 >> westernscheldt10.Update(43200.0)
** INFO   : Solver converged in 1 iterations, res= 0.2520E-14 dt =   1.0909
** INFO   : Solver converged in 1 iterations, res= 0.1215E-13 dt =   1.1891
** INFO   : Solver converged in 1 iterations, res= 0.4507E-13 dt =   1.2936
** INFO   : Solver converged in 1 iterations, res= 0.1650E-12 dt =   1.4198
** INFO   : Solver converged in 2 iterations, res= 0.1549E-19 dt =   1.5541
** INFO   : Solver converged in 2 iterations, res= 0.2529E-19 dt =   1.6933
** INFO   : Solver converged in 22 iterations, res= 0.6721E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6637E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6550E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6461E-12 dt =  30.0000
Dimr [2020-04-28 17:48:39.823] #0 >> westernscheldt10.Finalize()
Dimr [2020-04-28 17:48:39.823] #2 >> westernscheldt10.Finalize()
Dimr [2020-04-28 17:48:39.825] #1 >> westernscheldt10.Finalize()
Dimr [2020-04-28 17:48:39.829] #0 >> TIMER INFO:

Dimr [2020-04-28 17:48:39.829] #0 >> westernscheldt10   : 8.400 sec
Dimr [2020-04-28 17:48:39.830] #2 >> TIMER INFO:

Dimr [2020-04-28 17:48:39.830] #2 >> westernscheldt10   : 8.400 sec
Dimr [2020-04-28 17:48:39.831] #1 >> TIMER INFO:

Dimr [2020-04-28 17:48:39.831] #1 >> westernscheldt10   : 8.402 sec
```

4.2.13.4 Parallel on Linux

- ➢ Check that the Intel MPI library is installed on your system. If not, then open the D-Flow FM Installation Manual how to install the Intel MPI library either on Windows or on CentOS.
- ➢ Check that your model is partitioned correctly, see section 4.2.12. In this example the model is partitioned in three parts.
- ➢ Open file <dimr_config.xml> in a text editor, add the following two lines to the *<component>*-group:

  `<mpiCommunicator>DFM_COMM_DFMWORLD</mpiCommunicator>`

  `<process>0 1 2</process>`

  where the process numbers must match exactly with the number of partitions. Save the file as <dimr_config_parallel.xml>.
  This step is identical for Linux and Windows.
- ➢ Create a new file using a text editor, containing the following single line:

  `<installDir>/lnx64/bin/run_dimr.sh -c 3 -m dimr_config_parallel.xml` where `<installDir>` refers to your Delft3D installation directory.

➢ Save the file in folder <tutorial12\workdir>
Extension *.sh* is common, e.g. <run_model_parallel.sh>. It's a good habit to start the file with an interpreter reference, e.g. `#!/bin/bash`. Be sure that the line endings of your script file are conform Linux ("line feed" only). You can use the Linux command `dos2unix <scriptfile>` to convert it.

➢ Execute your `run_model_parallel` script.

The following text block is a subset of the text echoed to the command box via stdout (version numbers and dates may differ). The full stdout text is saved in file <tutorial12\output\run_model_parallel_sh_stdout.log>:

```
OMP_NUM_THREADS is 2
Configfile       : dimr_config_parallel.xml
D3D_HOME         : /opt/delft3/lnx64
Working directory: /p/tutorial12/workdir
Number of slots  : 3

Contents of machinefile:

----------------------------------------------------------------------
executing:
mpiexec -np 3 /opt/delft3d/lnx64/bin/dimr dimr_config_parallel.xml
#2: Running parallel with 3 partitions
Dimr [2020-04-28 17:56:13.428935] #2 >> Deltares, DIMR_EXE Version 2.00.00.66020, Feb 19 2020, 18:57:44
#0: Running parallel with 3 partitions
Dimr [2020-04-28 17:56:13.429264] #0 >> Deltares, DIMR_EXE Version 2.00.00.66020, Feb 19 2020, 18:57:44
#1: Running parallel with 3 partitions
Dimr [2020-04-28 17:56:13.429398] #1 >> Deltares, DIMR_EXE Version 2.00.00.66020, Feb 19 2020, 18:57:44
Dimr [2020-04-28 17:56:13.457510] #2 >> westernscheldt10.Initialize(westernscheldt12.mdu)
Dimr [2020-04-28 17:56:13.457522] #1 >> westernscheldt10.Initialize(westernscheldt12.mdu)
Dimr [2020-04-28 17:56:13.457722] #0 >>
Dimr [2020-04-28 17:56:13.457742] #0 >> Version Information of Components
Dimr [2020-04-28 17:56:13.457746] #0 >> =================================
Dimr [2020-04-28 17:56:13.457802] #0 >> dflowfm                            : 1.2.94.66020
Dimr [2020-04-28 17:56:13.457807] #0 >> ---------------------------------
Dimr [2020-04-28 17:56:13.457810] #0 >>
Dimr [2020-04-28 17:56:13.457827] #0 >> westernscheldt10.Initialize(westernscheldt12.mdu)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
** DEBUG  :            0  0.1000010      NOD(KMAX)
** DEBUG  :            6  0.7000070      XK (KMAX), YK (KMAX), ZK (KMAX), KC (KMAX), NMK (KMAX), RNOD(KMAX)
loading model
loading model
loading model
Dimr [2020-04-28 17:56:13.498662] #0 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:56:13.499212] #2 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
Dimr [2020-04-28 17:56:13.501699] #1 >> kernel: readMDUFile: [numerics] jaorgsethu=1 was in file, but not used. Check possible typo.
model loaded
model loaded
model loaded
Initializing flow: flow_modelinit
Initializing flow: flow_modelinit
Initializing flow: flow_modelinit
** INFO   : Initializing flow model geometry...
** INFO   : Initializing flow model geometry...
** INFO   : Initializing flow model geometry...
Dimr [2020-04-28 17:56:14.3507] #2 >> westernscheldt10.Update(43200.0)
Dimr [2020-04-28 17:56:14.28873] #1 >> westernscheldt10.Update(43200.0)
Dimr [2020-04-28 17:56:14.218837] #0 >> westernscheldt10.Update(43200.0)
** INFO   : Solver converged in 1 iterations, res= 0.2520E-14 dt =   1.0909
** INFO   : Solver converged in 1 iterations, res= 0.1215E-13 dt =   1.1891
** INFO   : Solver converged in 1 iterations, res= 0.4507E-13 dt =   1.2936
** INFO   : Solver converged in 1 iterations, res= 0.1650E-12 dt =   1.4198
** INFO   : Solver converged in 2 iterations, res= 0.1562E-19 dt =   1.5541
** INFO   : Solver converged in 2 iterations, res= 0.2280E-19 dt =   1.6933
** INFO   : Solver converged in 22 iterations, res= 0.6871E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6800E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6721E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6637E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6550E-12 dt =  30.0000
** INFO   : Solver converged in 22 iterations, res= 0.6461E-12 dt =  30.0000
Dimr [2020-04-28 17:56:21.41192] #2 >> westernscheldt10.Finalize()
Dimr [2020-04-28 17:56:21.42046] #1 >> westernscheldt10.Finalize()
Dimr [2020-04-28 17:56:21.48210] #0 >> westernscheldt10.Finalize()
Dimr [2020-04-28 17:56:21.51045] #1 >> TIMER INFO:

Dimr [2020-04-28 17:56:21.51055] #1 >> westernscheldt10 : 7.593429 sec
Dimr [2020-04-28 17:56:21.51064] #2 >> TIMER INFO:

Dimr [2020-04-28 17:56:21.51078] #2 >> westernscheldt10 : 7.593434 sec
Dimr [2020-04-28 17:56:21.53908] #0 >> TIMER INFO:

Dimr [2020-04-28 17:56:21.53917] #0 >> westernscheldt10 : 7.595980 sec
```

4.3 Tutorial Hydrodynamics for 3D modelling

4.3.1 Outline of the tutorials for 3D modelling

In this one-hour tutorial you will experience the basic functionality for 3D modelling with D-Flow FM. This manual contains two tutorials on 3D modelling (with indication of the estimated time):

- ◇ Tutorial 13 (section 4.3.2): Set-up of a 3D model and running this model (30 minutes).
- ◇ Tutorial 14 (section 4.3.3): Viewing the output of a 3D model simulation (30 minutes).

The necessary input files for these two tutorials can be found in the folders *tutorial13* and *tutorial14*. When saving your model data while working through these tutorials, be sure not to overwrite the tutorial data. Ideally, you should create a seperate working folder somewhere else on your computer to save your work. The folder *finished* in each tutorial folder contains the output generated by Deltares as a reference.

It is strongly recommend to at first read **??** on 3D modelling. In this chapter the keywords for 3D modelling are explained, which is not the case in the two 3D tutorial exercises below.

4.3.2 Tutorial 13: Set-up of a 3D model, running this model and postprocessing

In this tutorial exercise a simplified deep water 3D model will be set-up. Also a simulation will be carried out and the model results will be visualized for this 3D model. We start with a simplified 2D model for a shelf break and will step by step extend this model towards a 3D model with z-σ-co-ordinates in the vertical.

- ➢ Import the D-Flow FM model from the folder <tutorial13>, by choosing *Import* → *Flow Flexible Mesh Model* → *OK*. Choose <shelf_break_2D.mdu> and press *Open*.
- ➢ In the Map window select <Bed Level>. Switch on the legend by selecting via *Legend* in the main menu bar. Next, select contour classes as shown in Figure 4.37 by right mouse clicking on *Bed Level* in the Map window, selecting *Legend Properties* and choosing 11 classes in the range of 500 to -4500. Also switch on the <Open Street Map>. Then, your window should look like in Figure 4.37.

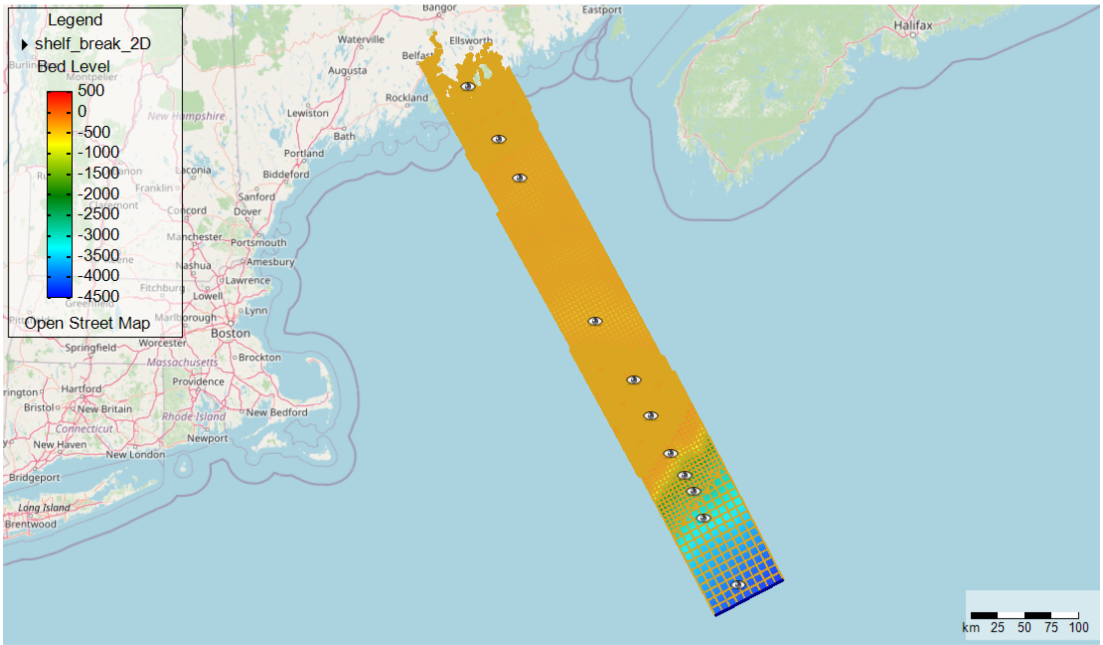
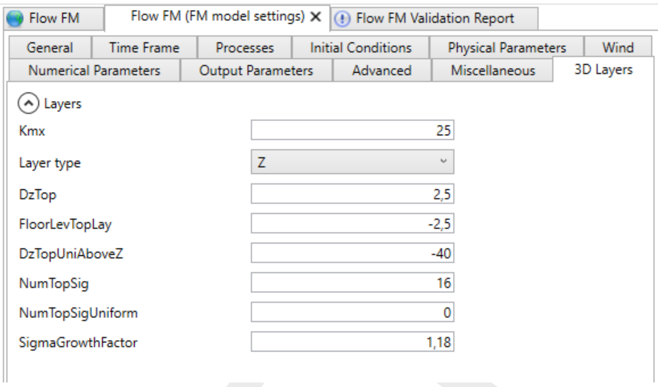


Figure 4.37: *Model domain with a shelf break.*

It can be seen that the bed levels varies from 0 m to a few hundred meters in the coastal zone. Then, after the shelf break the bed level rapidly increases to a depth of about 4300 m.

➢ DoubleClick on `General` in the `Project` window. Apart form the `General` settings other tabs will appear.

➢ Choose the tab `3D Layers` and specify the following:
`Kmx = 25`, the number of vertical layers;
`Layer type = 2`, indicating that the layer type is based on z-layers;
`DzTop = 2.5`, being the z-layer thickness of layers above level `Dztopuniabovez`;
`FloorLevTopLay = - 2.5`, indicating the floor level of top layer;
`DzTopUniAboveZ = -40.0`, indicating that above level `Dztopuniabovez` layers will have uniform a thickness of Dztop, while the `sigmaGrowthFactor` is applied below this level;
`NumTopSig = 16`, representing the number of sigma-layers on top of the z-layers;
`NumTopSigUniform = 0`, indicating whether the number of sigma-layers in a z-sigma-model is constant (=1) or decreasing (=0) (depending on local depth);
`SigmaGrowthFactor = 1.18`, representing the increase in layer thickness below the level specified for `Dztopuniabovez`.



Layers

Kmx	25
Layer type	Z
DzTop	2,5
FloorLevTopLay	-2,5
DzTopUniAboveZ	-40
NumTopSig	16
NumTopSigUniform	0
SigmaGrowthFactor	1,18

Figure 4.38: *Specification of Z-sigma layers*

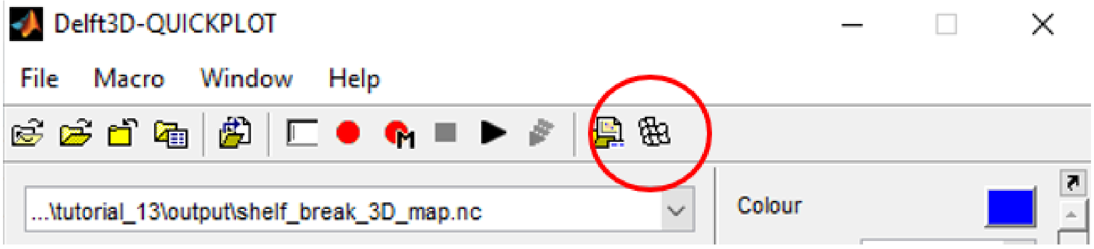
- ➢ Finally: Rename the `Flow FM` model in the `Project` window to `shelf_break_3D`; and
- ➢ Export the model to (DIMR Configuration) files.

Next, we will carry out a simulation with this 3D model. This can be done via a simulation inside the GUI or outside the GUI. We prefer to do the simulation outside the GUI, because this will go slightly faster. Moreover, then simulations can be done in parallel mode on multiple cores. The latter is not possible via the GUI.

- ➢ Open file <run_dimr.bat> and check whether the path specified in this file for the D-Flow FM executable is valid on your computer.
- ➢ If the path is not valid, because for example another Delft3D Flexible Mesh release has been installed on your computer, then change this path so that it refer to a D-Flow FM executable on your computer. Noted that there is no progress available. If the keyword `StatInterval` has been activated, then the progress can be checked in the diagnostic file of the simulation.
- ➢ Double click on <run_dimr.bat> so that the simulation will start. The simulation will roughly take five minutes, depending on the computing power of your computer.
- ➢ When the simulation has finished, go to folder <output> in folder <tutorial13> and open file <shelf_break_3D.dia>. Check whether the simulation has come to a normal end, which can be checked at the end of this file. For example, the average time step should be in the order of 45 s.

Now we will visualize the results of this 3D model.

- ➢ To that purpose start the Delft3D-QUICKPLOT program by double clicking on its icon on your desktop.
- ➢ Select the MAP output via *File* → *Open File* and then go the folder <output> and select file <shelf_break_3D_map.nc>.
- ➢ Click on the <Grid icon> as shown in Figure 4.39.



...\tutorial_13\output\shelf_break_3D_map.nc

Figure 4.39: *Icon for grid selection.*

- ➢ Click on the sixth icon and draw an arbitrary line as shown in Figure 4.40.

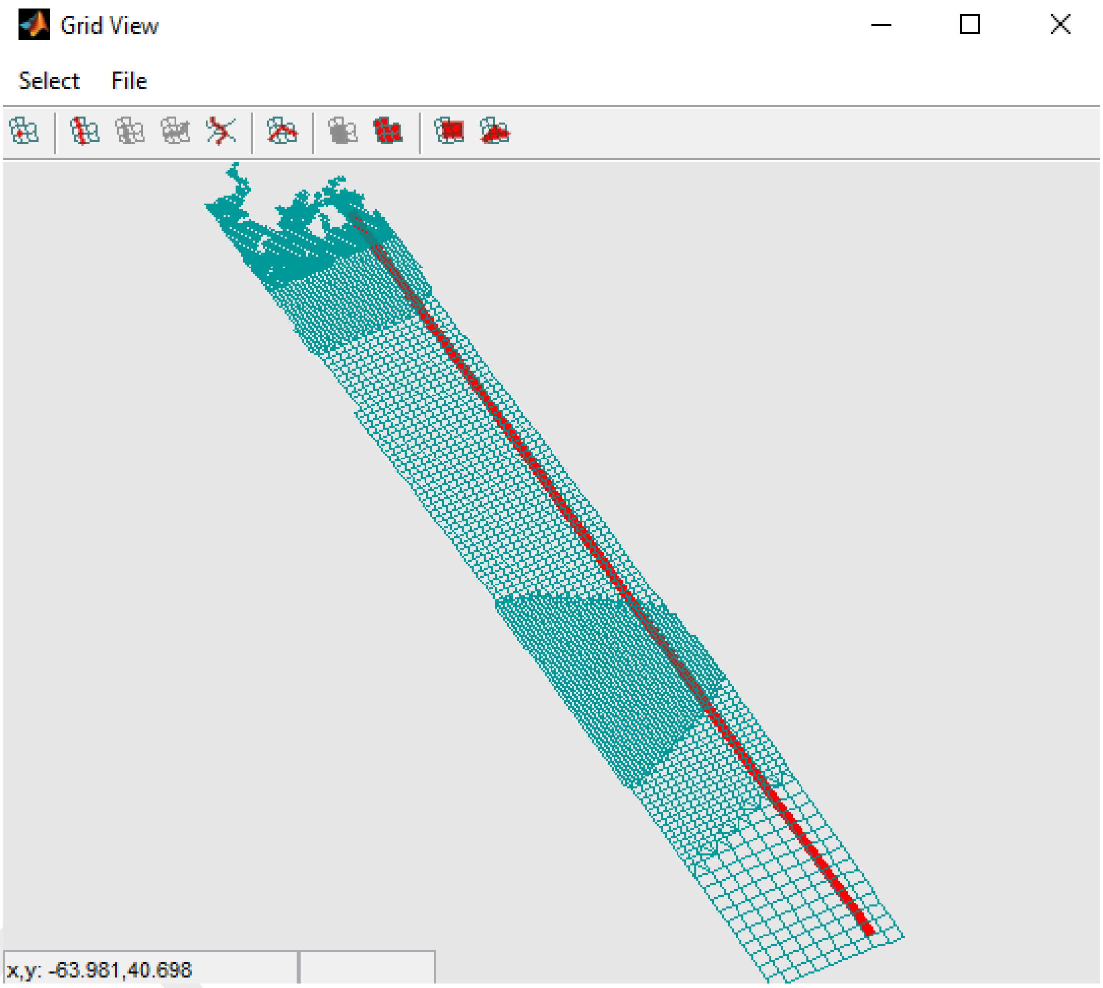


Figure 4.40: *Location of 2DV cross section.*

- ➢ Select quantity *Salinity in flow element –nmesh2d_face: mean*.
- ➢ For the *Presentation Type* click on *patches with lines*.
- ➢ For the *Colour* select the white colour instead of the blue colour, which is the default.
- ➢ Click on *Quick View* and the figure will show up as in Figure 4.41.

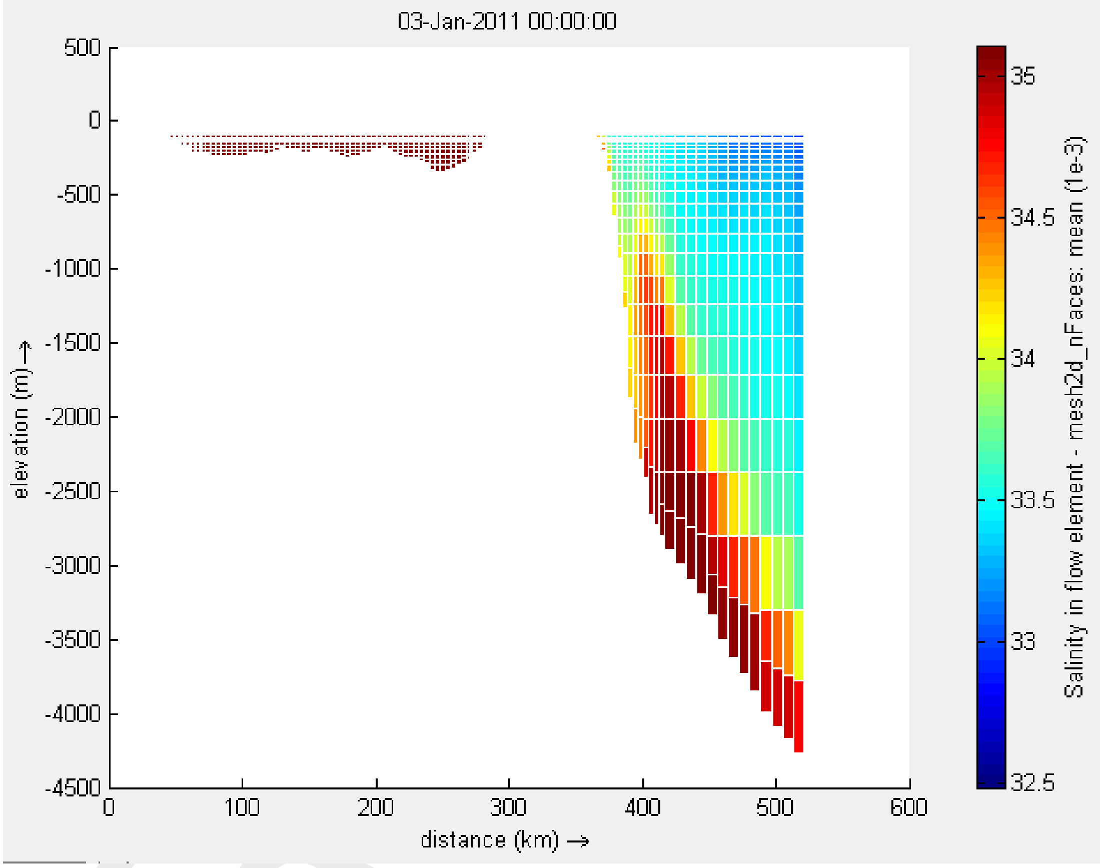


Figure 4.41: *Overall 2DV cross section of salinity including layer interfaces.*

➢ Now zoom into the shelf break so that a figure will appear as in Figure 4.42.

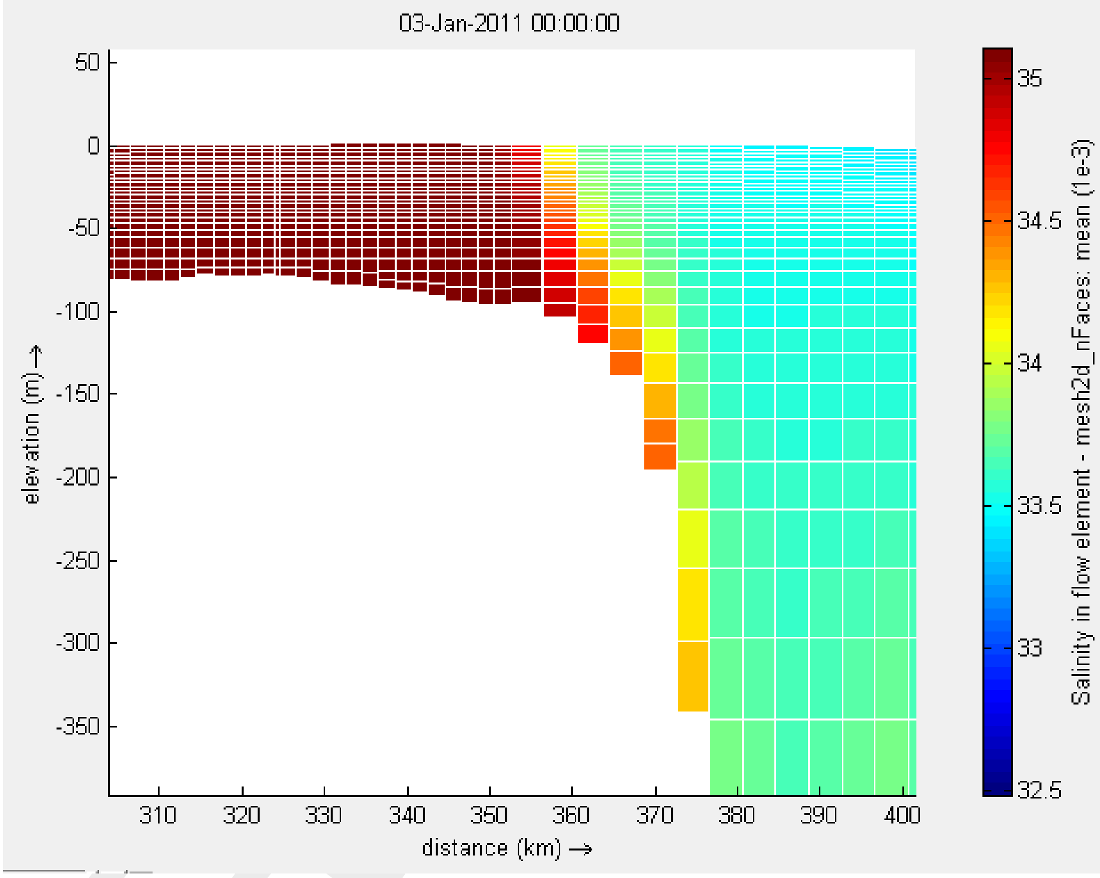


Figure 4.42: *2DV cross section of salinity including layer interfaces at shelf break.*

It can be seen that in the top 40 meters of the water column 16 layers of a thickness of about 2.5 m appear, which represent the σ layers in this model. Now, we will zoom into the deeper part of the model domain.

➢ Therefore, at first zoom out again to the whole cross section and then zoom in to the deep part as shown in Figure 4.43.

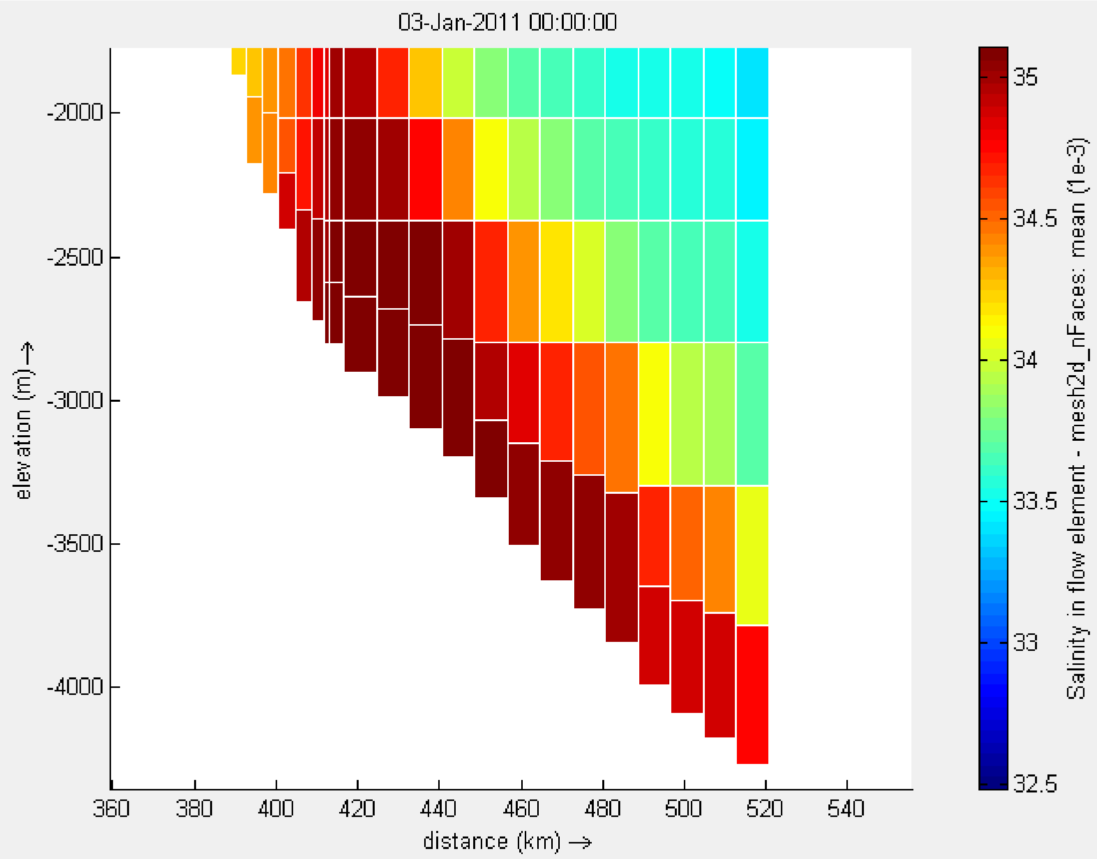


Figure 4.43: *2DV cross section of salinity including layer interfaces at deep part.*

It can be seen that the layer thicknesses rapidly increase for the larger depths. All layer interfaces are strictly horizontal except for the two deepest layers. In order to prevent very thin layers near the bed, which is unwanted from a numerical point of view, the two deepest layers have the same thickness. Consequently, the interfaces in the two bed layers aren't strictly horizontal.
In this model there are σ layers near the surface and z-layers in the deeper part. This is a much preferred combination, because σ layers can lead to artificial mixing in areas of steep bed topography, due to the hydrostatic inconsistency (Haney, 1991). On the other hand, z layers can lead to stair cases near the surface. Moreover, a top layer can become very thin, so that the ratio in layer thickness with the next layer can be large, which is unwanted from a numerical point of view. This might lead to a small time step and therefore huge computation time. By applying σ layers near the surface and z layers in the deeper parts, the disadvantages of both approaches are largely circumvented.

In summary, in this tutorial exercise a simplified 3D model was set-up that consists of a relatively shallow part with depths up to about 100 meter, a shelf break and a deep part of more than four kilometers deep. D-Flow FM has been developed for coastal applications and therefore focusses on processes in this relatively shallow areas of roughly a few hundred meters deep. For such applications D-Flow FM performs adequately. However, D-Flow FM is also used in oceanic applications with areas of kilometers deep. For these oceanic applications software systems like HYCOM, NEMO and ROMS have been developed. These software systems are based on different numerical schemes and different turbulence models compared to D-Flow FM, so that model results of D-Flow FM can be different for deep water applications.

4.3.3 Tutorial 14: Extra options for postprocessing of 3D model simulations

In this tutorial extra postprocessing options will be described for 3D modelling. This will be done for a flume model with a length of 130 m. At the left open boundary there is a tidal boundary with either inflow or outflow of water. At the right open boundary there is a small inflow discharge of water.

In Figure 4.44 this flume model is visualized in the GUI of D-Flow FM. A meandering grid is applied consisting of quadrangles with locally a grid refinement. In this figure an arbitrary line has been drawn (in red). This section will be used for postprocessing, as we did in the previous tutorial.

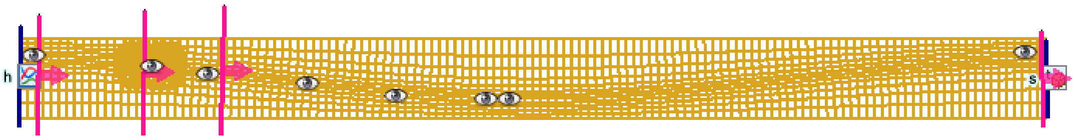


Figure 4.44: *Location of 2DV cross section.*

- ➢ Start the Delft3D-QUICKPLOT program by double clicking on its icon on your desktop.
- ➢ Select the MAP output via *File* → *Open File* and then go the folder <tutorial14_output> and select file <tidalflume_map.nc>.
- ➢ Click on the *Grid icon* as shown in Figure 4.39.
- ➢ Click on the sixth icon and draw an arbitrary line as shown in Figure 4.45.

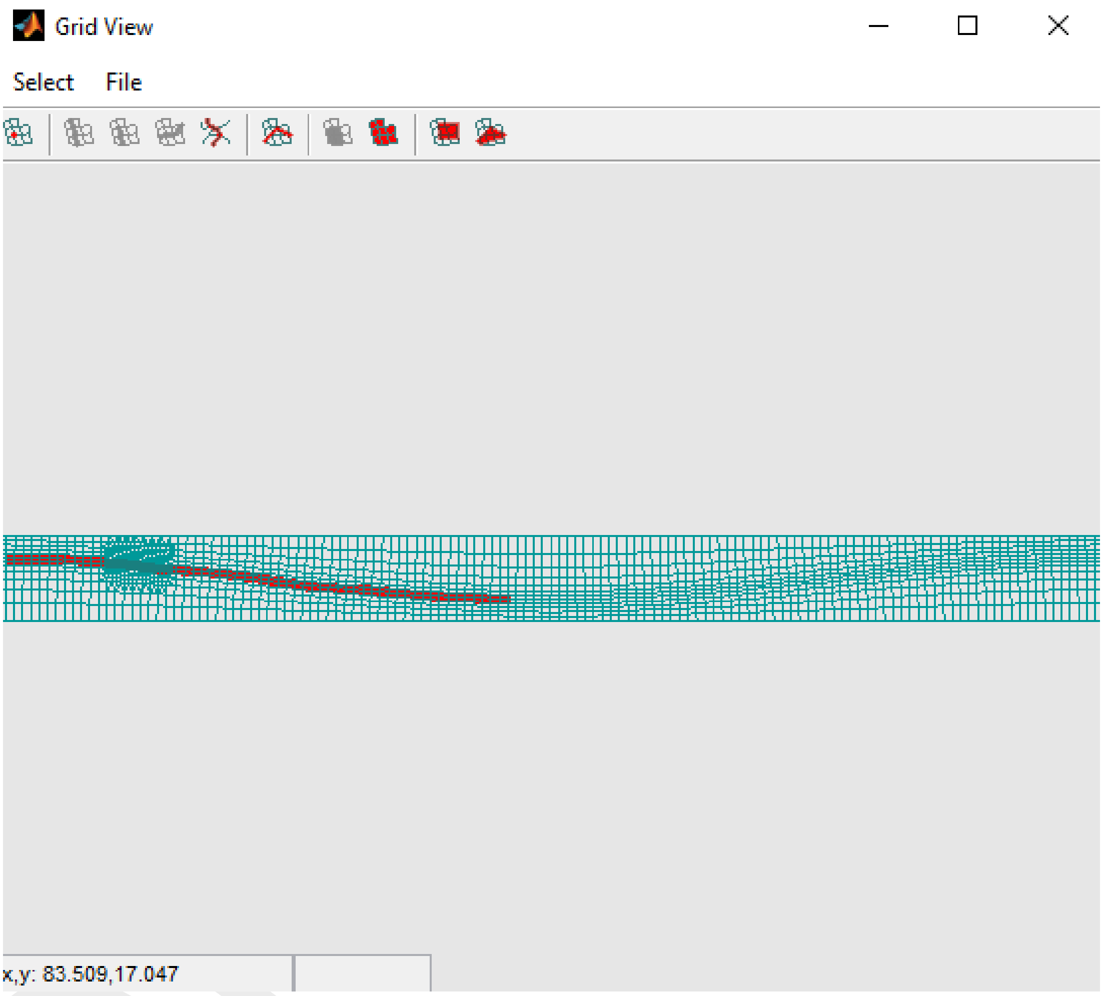


Figure 4.45: *Location of 2DV cross section.*

- ➢ Select quantity *Salinity in flow element –nmesh2d_face: mean*.
- ➢ For the *Presentation Type* click on *continuous shades*.
- ➢ Click on *Show Times* and select `01-Sep-1992 00:54:00`.
- ➢ Click on *Quick View* and a figure will show up as in Figure 4.46.

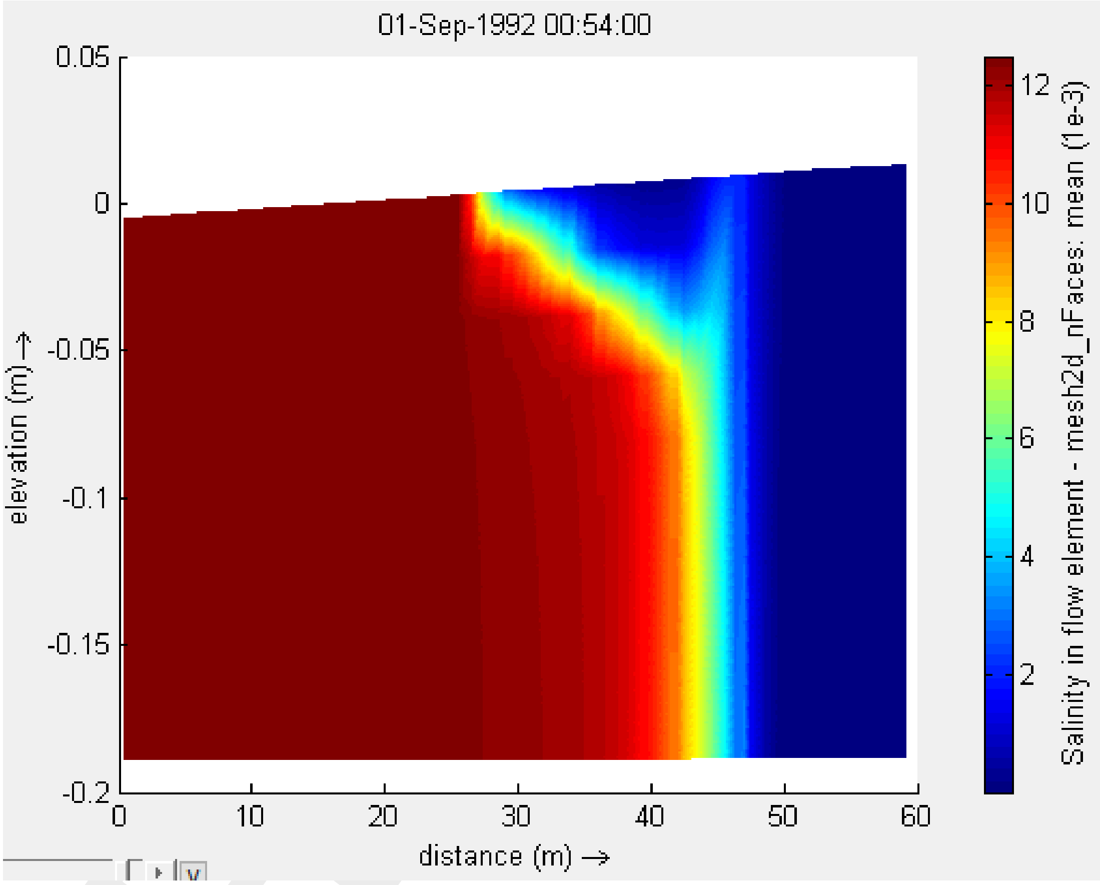


Figure 4.46: 2DV cross sectional view of salinity.

Next, we will make a 2DV(ertical) cross sectional plot for the vertical eddy viscosity.

- ➢ Therefore, Select quantity *turbulent vertical eddy viscosity –mesh2d_nEdges: mean*.
- ➢ For the *Presentation Type* click on *continuous shades*.
- ➢ Click on *Show Times* and select `01-Sep-1992 00:54:00`.
- ➢ Click on *Quick View* and a figure will show up as in Figure 4.47. Neglect the white vertical bars, which is due to the postprocessing when interpolating along the chosen grid line.

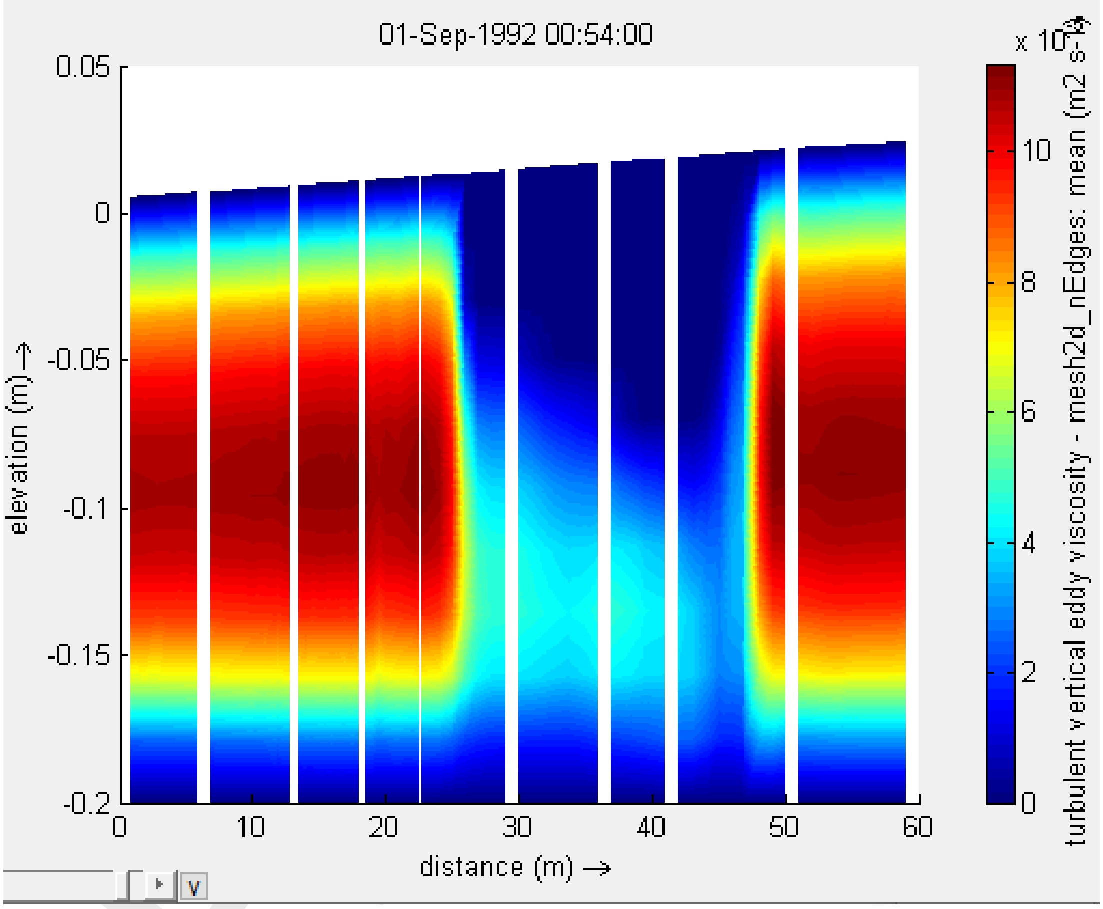


Figure 4.47: *2DV cross sectional view of vertical eddy viscosity.*

In Figure 4.46 it can be seen that in the area between 25 m en 45 m from the inflow boundary salinity stratification occurs. In other parts of the model domain no vertical stratification of salinity occurs. When looking at the vertical profiles for the vertical eddy viscosity, it can be seen that parabolic profiles occur in the vertical in the areas without any stratification. This is in line with the theory for well-mixed areas. However, in the area of vertical stratification, the vertical profiles for the vertical eddy viscosity are different, which is in line with the theory as well. In the validation document of D-Flow FM an almost similar flume model is validated on salinity measurements. In this validation document it is shown that D-Flow FM accurately computes the salinity intrusion in this tidal flume.

The previous figures were based on the map output file. We will now switch to proprocessing for the history output file.

- ➢ Therefore, select the history output via *File* → *Open File* and then go the folder <tutorial14_output> and select file <tidalflume_his.nc>.
- ➢ Select quantity *sea_water_salinity*.
- ➢ Select station *at 24 m*.
- ➢ Select layer *k=10*.
- ➢ Select all time steps.
- ➢ Click on *Quick View* and a figure will show up with only the blue line as in Figure 4.48.
- ➢ Select layer *k=1*.
- ➢ For the *Colour* select the red colour instead of the blue colour.
- ➢ Click on *Add to Plot* and a figure will show up as in Figure 4.48.

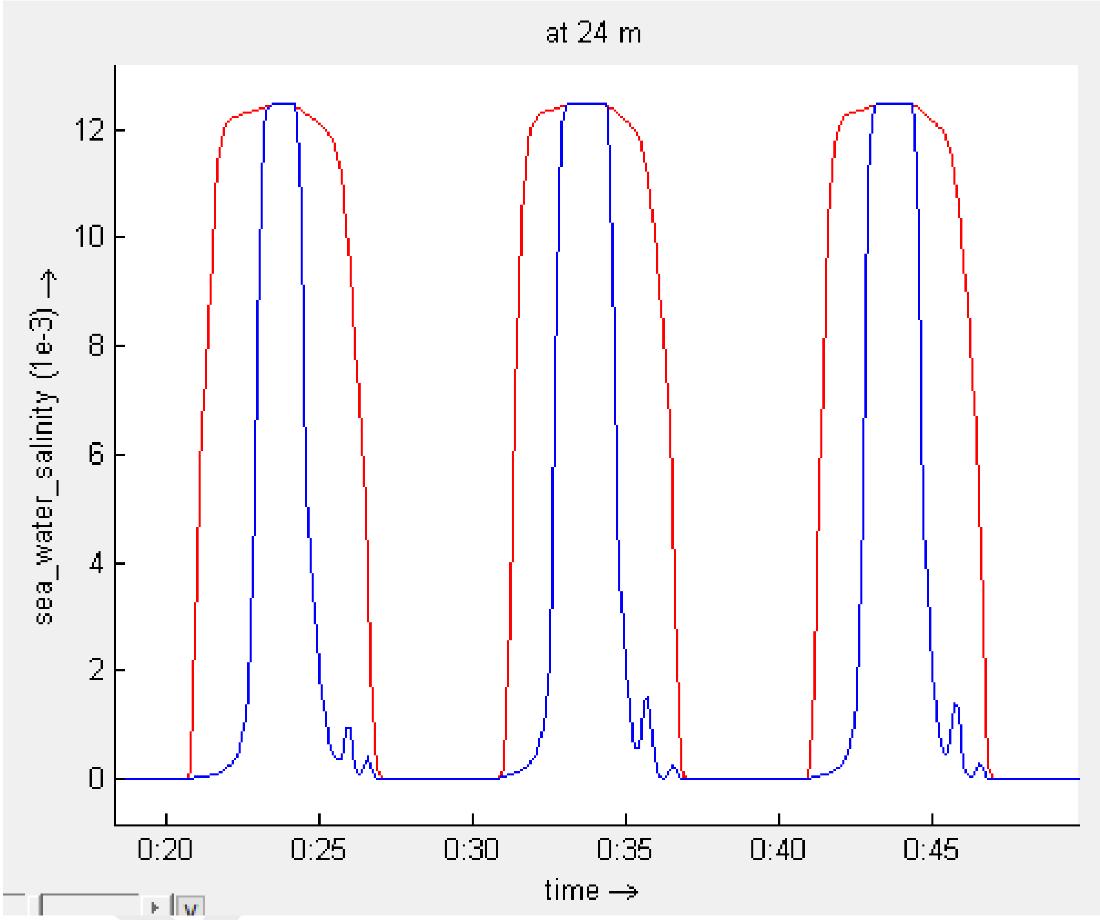


Figure 4.48: *Time history of salinity near surface and near bed.*

Finally, we will generate a ZT(ime)-plot in which for one observation point the salinity concentrations will be shown during the whole simulation.

- To that purpose select station *at 24 m*.
- Select all layers.
- Select all time steps.
- For the *Axes Type* select *Time-Z*
- For the *Presentation Type* click on *continuous shades*.
- Click on *Quick View* and a figure will show up with salinity concentrations at 24 m form the inflow boundary for the whole simulation period.
- Now, zoom in the period from 20 to 50 minutes after the start. Then a figure will show up as in Figure 4.49.

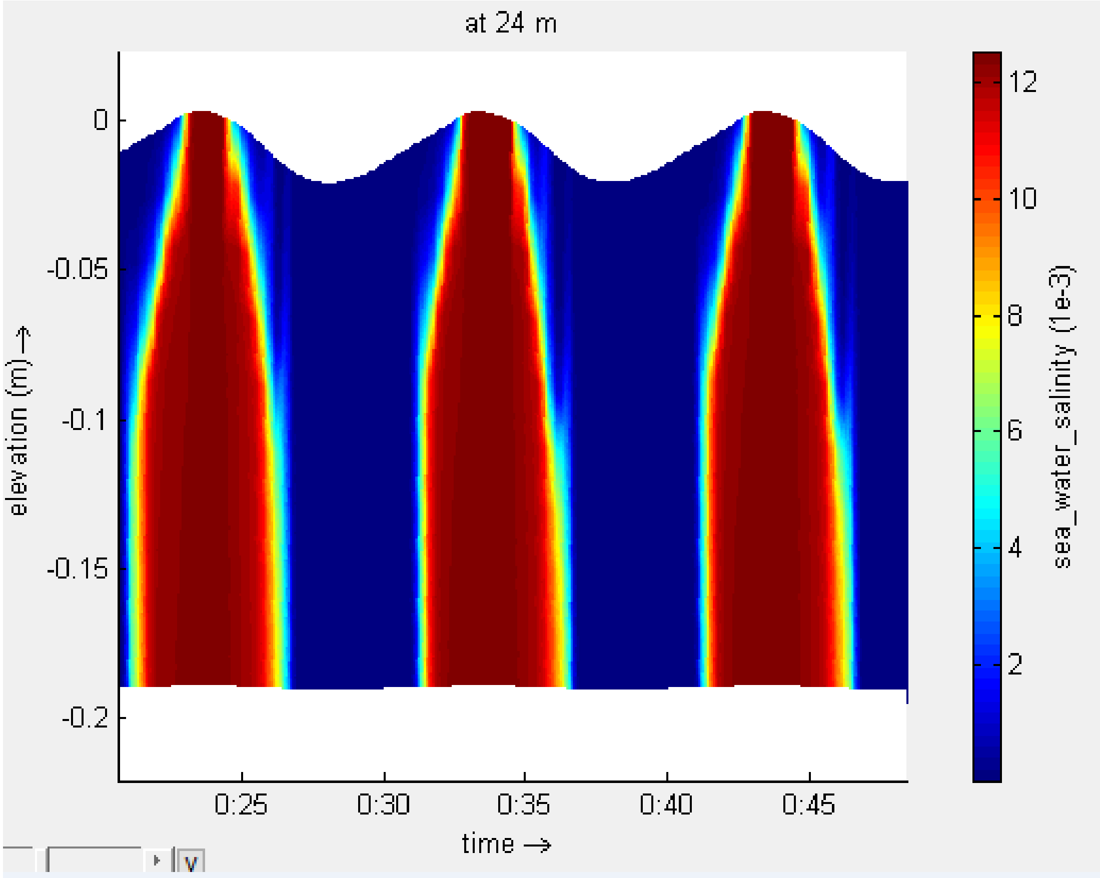


Figure 4.49: *ZT plot of salinity at 24 m from inflow boundary.*

In summary, in this tutorial several extra postprocessing options have been explained for a tidal flume model. In this model only 10 σ layers are used. Despite its relatively low number of layers an accurate vertical salinity stratification was computed. The computation of salinity stratification and temperature stratification in the vertical is one of the nice features of 3D modelling with D-Flow FM.

5 All about the modelling process

5.1 Introduction

In order to set up a hydrodynamic model you must prepare an input file. All parameters to be used originate from the physical phenomena being modelled. Also from the numerical techniques being used to solve the equations that describe these phenomena, and finally, from decisions being made to control the simulation and to store its results. Within the range of realistic values, it is likely that the solution is sensitive to the selected parameter values, so a concise description of all parameters is required. This input data is collected into the Master Definition Unstructured file, called a mdu-file.

In section 5.2 we discuss some general aspects of the mdu-file and its attribute files. The sections thereafter describe how the actual modelling process can be done in the GUI.

5.2 mdu-file and attribute files

The Master Definition Unstructured file (mdu-file) is the input file for the hydrodynamic simulation program. It contains all the necessary data required for defining a model and running the simulation program. In the mdu-file you can define attribute files in which relevant data (for some parameters) is stored. This will be particularly the case when parameters contain a large number of data (e.g., time-dependent or space varying data). The mdu-file and all possible user-definable attribute files are listed and described in Appendix A.

Although you are not supposed to work directly on the mdu-file it is useful to have some ideas on it as it reflects the idea of the designer on how to handle large amounts of input data and it might help you to gain a better idea on how to work with this file.

The basic characteristics of an mdu-file are:

- It is an ASCII file.
- Each line contains a maximum of 256 characters.
- Each (set of) input parameter(s) is preceded by a (set of) *keyword*(s).

The results of all modules are written to platform independent binary (netCDF-)files, so also these result files you can transfer across hardware platforms without any conversion.

The mdu-file contains several sections, denoted by square brackets, below are the most relevant ones:

`[general]` this section contains the program name and its version.

`[geometry]` in this section, the main entry comprises the specification of the grid (i.e. the netCDF network file). In addition, thin dams and thin dykes can be specified.

`[numerics]` this section contains the settings of specific parts of the flow solver, such as limiters and the iterative solver type.

`[physics]` in this field, physical model parameters can be inserted, for instance related to friction modelling and turbulence modelling.

`[wind]` the wind section prescribed the dependency of the wind drag coefficient to the wind velocity through 2 or 3 breakpoints. This field also contains pressure information.

`[time]` in this section, the start time and the stop time of the simulation are specified (both as a combination of a date and time).

`[restart]` in this section, the restart file can be specified, either as a <∗_map.nc>-file or as an <∗_rst.nc> file. Because a <∗_map.nc>-file can contain multiple stages, in this case also a `RestartDateTime` must be specified.

`[external forcing]` this section only contains the name of the external forcings file.

`[output]` in this section, the writing frequency of output data can be prescribed.

Appendix A contains the full list of MDU sections and keywords.

5.3 Filenames and conventions

Filenames and file extensions hardly have any strict requirements in D-Flow FM, but we do advise to use the suggested file naming patterns below:

file pattern	description
mdu_name.mdu	mdu-file
∗_net.nc	Unstructured grid (network) file
∗.xyz	Sample file (for spatial fields)
∗.ldb	Land boundary file (polyline file format)
∗_thd.pli	Thin dam file (polyline file format)
∗_fxw.pliz	Fixed weir file (polyline file format with z values)
∗_crsdef.ini	1D cross section definition file (see section C.16.1)
∗_crsloc.ini	1D cross section location file (see section C.16.2)
roughness-∗ini	1D roughness variable file (see section C.15)
∗_part.pol	Partitioning polygon file (polyline file format)
∗.ext	External forcings file
pli_name.pli	Boundary condition location file (polyline file format)
*pli_name*_000X.tim	Timeseries boundary data file at point #X
*pli_name*_000X.cmp	Astronomic/harmonical component boundary data file at point #X
∗.bc	BC-format boundary data file with polyline and point labels in file
∗.xyn or ∗_obs.ini	Observation station file (see section F.2.2)
∗_crs.pli or ∗_crs.ini	Observation cross-sections file (see section F.2.4)
*mdu_name*_map.nc	Output map file
*mdu_name*_his.nc	Output his file
*mdu_name*_clm.nc	Output class map file (see section F.3.3)
mdu_name.dia	Output diagnostics (log) file

5.4 Setting up a D-Flow FM model

This chapter describes how to set up a D-Flow FM model in an empty User Interface project. When you open the GUI, an empty project is automatically created. Starting from scratch, you have to create an empty D-Flow FM model in a User Interface project:

◇ In the ribbon menu items, go to *Home* and click on *New Model*. In the drop-down menu Figure 5.1 that appears, select "Flow Flexible Mesh Model" to add an empty D-Flow FM model.

◇ Or use the right mouse button on the name of your project (project1 by default). In the context menu that appears Figure 5.2, select *Add* and click *New Model*. The *Select model ...* window appears allowing you to add an empty D-Flow FM model.

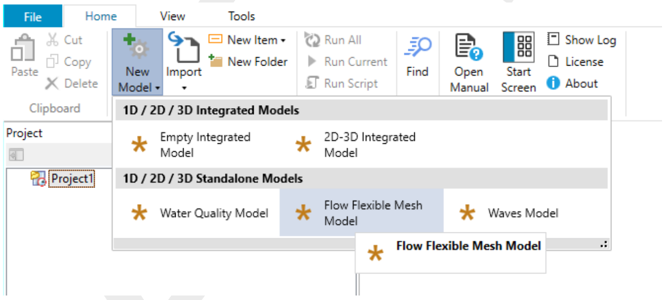


Figure 5.1: *The Add Model drop-down menu*

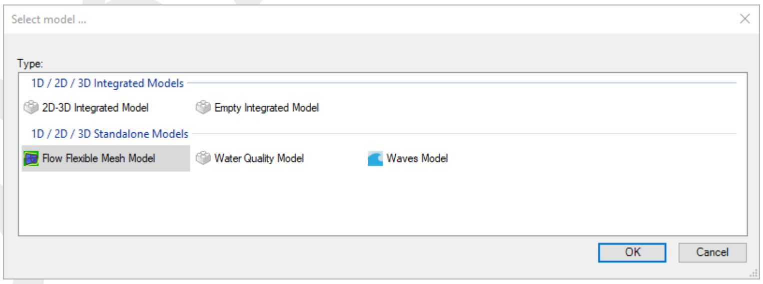
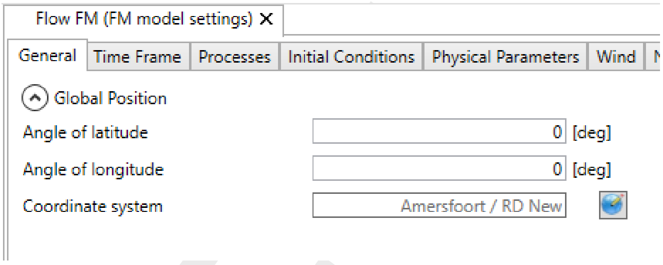


Figure 5.2: *The* Select model ... *window*

In the **Project** window, an empty model has appeared (FlowFM by default). Click the plus sign (+) before the name of the model to expand all model attributes in the **Project** window: General, Area, Grid, Bed Level, Time Frame, Processes, Initial Conditions, Boundary Conditions, Physical Parameters, Sources and Sinks, Numerical Parameters, Output Parameters and Output. In the following paragraphs, all model attributes are treated separately.

5.4.1 General

When you double click on *General* in the **Project** window, the tabbed editor for the D-Flow FM appears (Figure 5.3). The general tab contains general model information such as the model coordinate system and the angle of latitude and of longitude.



Angle of latitude 0 [deg]

Angle of longitude 0 [deg]

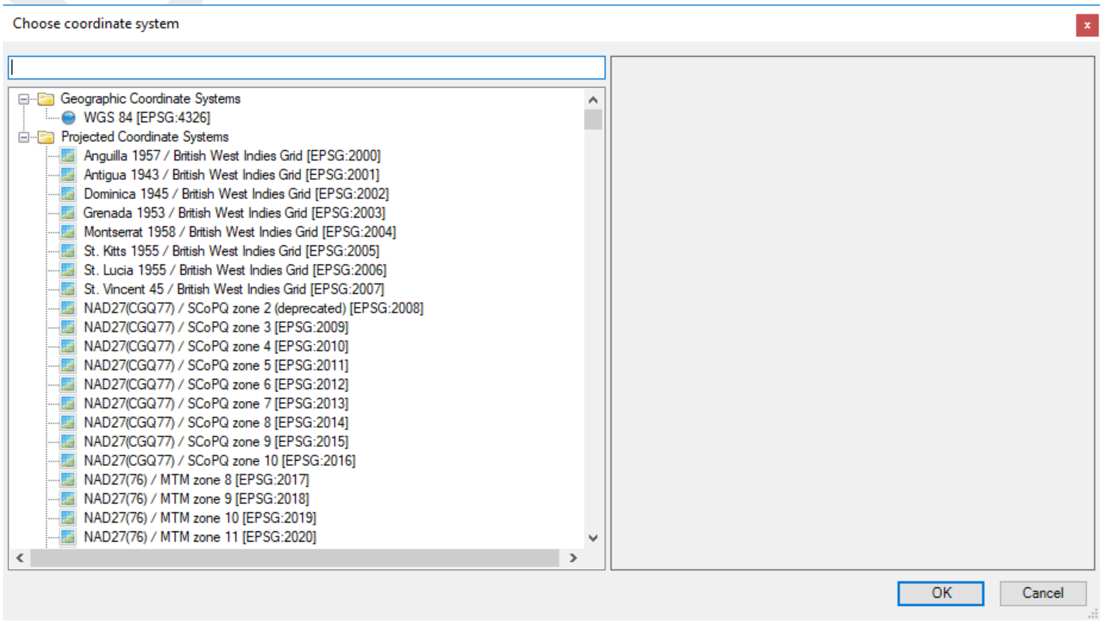
Coordinate system Amersfoort / RD New

Figure 5.3: *Overview of general tab*

5.4.1.1 Model coordinate system

A very important property of your model is the coordinate system in which it is specified. Within the interface, there is a clear distinction between the coordinate system of your model and all of its attributes and the coordinate system of the central map and all of its items. Both coordinate systems can be set independent from each other. Keep in mind, that the coordinate system of your model is saved and used when you run your model. Only coordinate systems supported by the computational core are supported.

The model coordinate system can be set using the globe icon next to the Coordinate system text box. After clicking this button, the coordinate system wizard is opened (Figure 5.4). This wizard allows you to choose one of many possible coordinate systems and apply it to your model. You can use the search bar to browse the various coordinate systems (searching possible by name and EPSG code). If your model is specified in a certain coordinate system already, it is possible to convert the model coordinate system using the same button. After clicking *OK*, all model attributes are converted to the system of choice. Note that you have to close and re-open all map views for the changes to take effect in these views.



- Geographic Coordinate Systems
  - WGS 84 [EPSG:4326]
- Projected Coordinate Systems
  - Anguilla 1957 / British West Indies Grid [EPSG:2000]
  - Antigua 1943 / British West Indies Grid [EPSG:2001]
  - Dominica 1945 / British West Indies Grid [EPSG:2002]
  - Grenada 1953 / British West Indies Grid [EPSG:2003]
  - Montserrat 1958 / British West Indies Grid [EPSG:2004]
  - St. Kitts 1955 / British West Indies Grid [EPSG:2005]
  - St. Lucia 1955 / British West Indies Grid [EPSG:2006]
  - St. Vincent 45 / British West Indies Grid [EPSG:2007]
  - NAD27(CGQ77) / SCoPQ zone 2 (deprecated) [EPSG:2008]
  - NAD27(CGQ77) / SCoPQ zone 3 [EPSG:2009]
  - NAD27(CGQ77) / SCoPQ zone 4 [EPSG:2010]
  - NAD27(CGQ77) / SCoPQ zone 5 [EPSG:2011]
  - NAD27(CGQ77) / SCoPQ zone 6 [EPSG:2012]
  - NAD27(CGQ77) / SCoPQ zone 7 [EPSG:2013]
  - NAD27(CGQ77) / SCoPQ zone 8 [EPSG:2014]
  - NAD27(CGQ77) / SCoPQ zone 9 [EPSG:2015]
  - NAD27(CGQ77) / SCoPQ zone 10 [EPSG:2016]
  - NAD27(76) / MTM zone 8 [EPSG:2017]
  - NAD27(76) / MTM zone 9 [EPSG:2018]
  - NAD27(76) / MTM zone 10 [EPSG:2019]
  - NAD27(76) / MTM zone 11 [EPSG:2020]

Figure 5.4: *Coordinate system wizard*

The map coordinate system applies to all items in the map **Project** window. To change the map coordinate system, navigate the menu ribbons to *Map* and click on *Map coordinate system*. Alternatively, right mouse click on *Map* in the map *Project* window and select *Change Map Coordinate System*. The coordinate system wizard appears, allowing you to set the map coordinate system of your choice. When the original model coordinate system differs from the selected map coordinate system, all map items are automatically converted to the specified map coordinate system.

5.4.1.2 Angle of latitude

For a Cartesian grid you have to specify the latitude of the model area; this is used to calculate a fixed Coriolis force for the entire area. For a spherical grid the Coriolis force is calculated from the latitude coordinates in the grid file and thus varies in the latitude direction. Typically, you use spherical co-ordinates for large areas, such as a regional model. When a value of 0 is entered, the Coriolis force is not taken into account.

5.4.2 Area

The model area contains all geographical features, such as the observation points, structures and land boundaries. These features can exist without a grid or outside the grid as they are not based on grid coordinates but xy-coordinates. This means that the location of the model area features remains the same when the grid is changed (for example by (de-)refining). When you expand the model attribute *Area* in the *Project* window, a list of supported geographical features is displayed (Figure 5.5).

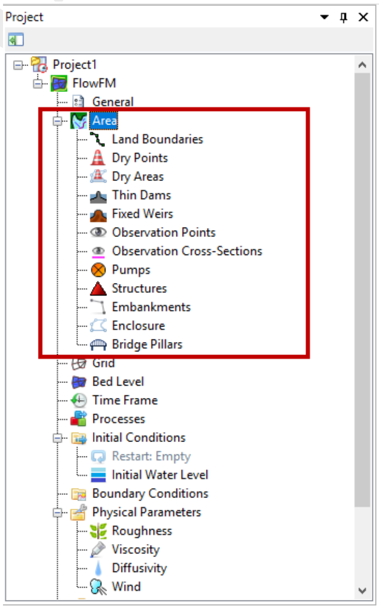


Figure 5.5: *Overview of geographical features*

Below is Figure 5.6. It displays an overview of the (*Map*) ribbon.

The red boxes to the left (*FM Region 2D / 3D*) and right (*Area*) can be used to *Add ...* the various geographical features to the model: click the corresponding item in the box and use your mouse to indicate the location of the selected feature on the central map. The red box in the middle highlights the buttons available to edit the location of geographical features. The specifics for each feature are discussed separately in the following sections.

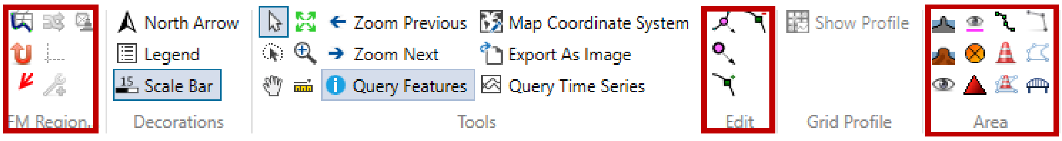


Figure 5.6: *Overview of* Map *ribbon*

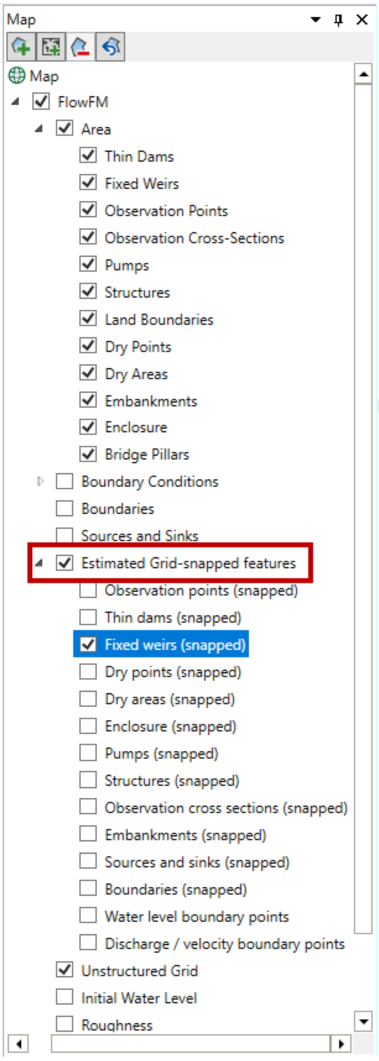
Importing and exporting of model features is done with the context menu by using your right mouse button on the different features in the *Project* window.

5.4.2.1 Grid-snapped features

All geographical features of your model that are described by x-, and y-coordinates have to be interpolated to your computational grid when you run your model. The computational core of D-Flow FM automatically assigns these features to the corresponding parts of your grid. The so-called Grid-snapped features are part of the output of a simulation: *shape* files. In section 5.4.12 is described how - in short: select *Write Snapped Features* in the *Output Parameters* tab below the central map.

However, the graphical user interface allows you to inspect the grid-snapped features before a simulation run.

Note: Mark that this way, the grid-snapping is an estimation because it would take too much processing time. In the neighbourhood of *Dry Areas* the grid-snapping will be different, as the *Dry Areas* are not taken into account in this interactive *Estimated Grid-snapped features* mode.



Map

- Map
  - ☑ FlowFM
    - ☑ Area
      - ☑ Thin Dams
      - ☑ Fixed Weirs
      - ☑ Observation Points
      - ☑ Observation Cross-Sections
      - ☑ Pumps
      - ☑ Structures
      - ☑ Land Boundaries
      - ☑ Dry Points
      - ☑ Dry Areas
      - ☑ Embankments
      - ☑ Enclosure
      - ☑ Bridge Pillars
    - ☐ Boundary Conditions
    - ☐ Boundaries
    - ☐ Sources and Sinks
    - ☑ Estimated Grid-snapped features
      - ☐ Observation points (snapped)
      - ☐ Thin dams (snapped)
      - ☑ Fixed weirs (snapped)
      - ☐ Dry points (snapped)
      - ☐ Dry areas (snapped)
      - ☐ Enclosure (snapped)
      - ☐ Pumps (snapped)
      - ☐ Structures (snapped)
      - ☐ Observation cross sections (snapped)
      - ☐ Embankments (snapped)
      - ☐ Sources and sinks (snapped)
      - ☐ Boundaries (snapped)
      - ☐ Water level boundary points
      - ☐ Discharge / velocity boundary points
    - ☑ Unstructured Grid
    - ☐ Initial Water Level
    - ☐ Roughness

Figure 5.7: *Example of expanded* Estimated Grid-snapped features *attribute in the* Map *window*

Figure 5.7 shows a part of the *Map* window, showing the *Area* and *Estimated Grid-snapped features* attributes. The x- and y-locations of all spatial model features are shown within the *Area* attribute. You can hide or show any of these attributes by means of clicking the check boxes in front of the attributes. When you enable the Grid-snapped features, all items within the *Area* attribute, as well as all boundaries are interpolated to their corresponding locations on the computational grid. The interpolation is performed instantaneously by the computational core of D-Flow FM, which enables you to directly inspect the numerical interpretation of all features on the computational grid. Figure 5.8 shows an example of four observation points and one thin dam in the central map, showing both the x- and y-locations of these features as well as their representation on the computational grid.

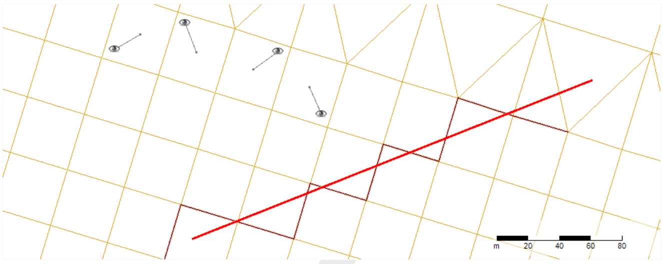


Figure 5.8: Example of grid snapped features displayed on the central map

5.4.2.2 Observation points

Observation points are used to monitor the time-dependent behaviour of one or all computed quantities as a function of time at a specific location, i.e., water level, velocity, salinity, temperature and concentration of the constituents. Observation points represent an Eulerian viewpoint at the results. (**Note: Sediment transport is a β-functionality**)

To add an observation point, click the corresponding icon from the *FM Region 2D / 3D* menu in the map ribbon (Figure 5.6). By clicking in the central map, observation points are placed in your model. Selected observation points (first click the *Select* icon from the “Tools” menu in the map ribbon) can be deleted using backspace or directly from the attribute table (explained below). The grid snapped representation (Figure 5.9) is indicated by a line linking the observation points to the closest cell center, indicating that output will be stored of this cell. Importing and exporting of observation points is possible via the context menu of “Observation points” in the *Project* window (right mouse button).

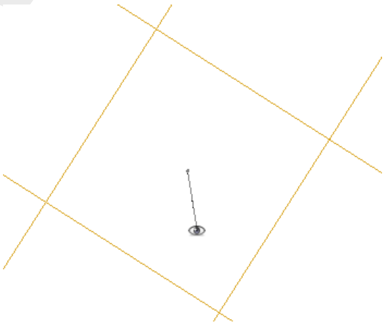
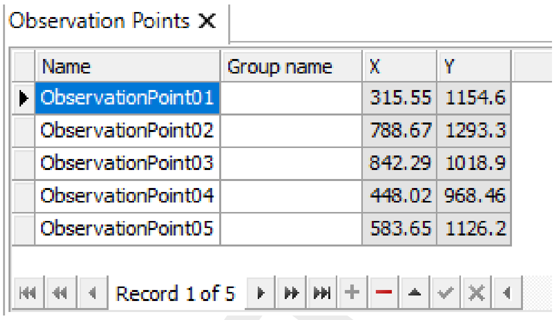


Figure 5.9: Geographical and grid snapped representation of an observation point

When you double click the *Observation points* attribute in the *Project* window, the observation points tab is displayed underneath the central map (Figure 5.10). This tab shows an attribute table with the names, x- and y-locations (in the model coordinate system) of the various observation points within the model. When one of the entries is selected, the corresponding observation point is highlighted in the central map.



Name	Group name	X	Y
ObservationPoint01		315.55	1154.6
ObservationPoint02		788.67	1293.3
ObservationPoint03		842.29	1018.9
ObservationPoint04		448.02	968.46
ObservationPoint05		583.65	1126.2

Figure 5.10: *Attribute table with observation points*

5.4.2.3 Observation cross-sections

Cross-sections (Figure 5.11) are used to store the sum of computed fluxes (hydrodynamic), flux rates (hydrodynamic), fluxes of matter (if existing) and transport rates of matter (if existing) sequentially in time at a prescribed interval.

To add a cross section, click the corresponding icon from the *FM Region 2D / 3D* menu in the map ribbon (Figure 5.6). By clicking in the central map, cross section points are added. Note that a cross section consists of a minimum of two points, but an arbitrary amount of intermediate points can be added. The last point is indicated by double clicking the left mouse button. The distance in meters (independent of the local coordinate system) in between the last point and the mouse pointer is indicated in pink; in case more than two points are used, the cumulative length of the entire cross section is shown in black. Once highlighted in the central map, a cross section is deleted with backspace or directly from the attribute table described below. The positive direction through the cross section is indicated by a pink arrow. The direction of this arrow is dependent on the order in which the cross section points are drawn (to change the direction, flip the start and end points). Importing and exporting of cross sections is possible via the context menu of “Observation cross-sections” in the *Project* window (right mouse button).

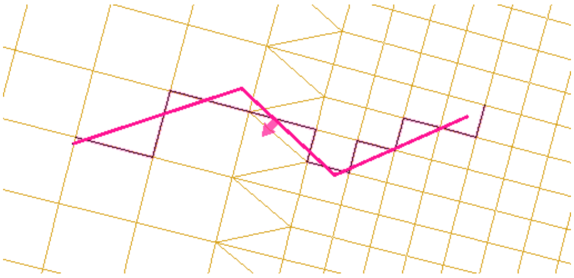
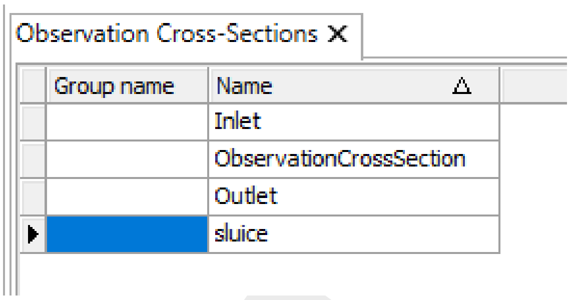


Figure 5.11: *Geographical and grid snapped representation of a cross section*

Double clicking the *Observation cross-sections* attribute in the *Project* window enables the Observation cross-sections tab underneath the central map view (Figure 5.12). Alternatively, you can double click on any cross section in the map. The attribute table displayed in the tab contains the names of the various cross sections of your model. When one of the entries is selected, the corresponding cross section is highlighted in the central map. A cross section entry can be deleted from the table via the context menu (right mouse button).



Observation Cross-Sections ✕

Group name	Name △
	Inlet
	ObservationCrossSection
	Outlet
	sluice

Figure 5.12: *Attribute table with observation cross sections*

5.4.2.4 Thin dams

Thin dams (Figure 5.13) are infinitely thin objects defined at the velocity points which prohibit flow exchange between the two adjacent computational cells without reducing the total wet surface and the volume of the model. The purpose of a thin dam is to represent small obstacles (e.g. breakwaters, dams) in the model which have sub-grid dimensions, but large enough to influence the local flow pattern. A thin dam is assumed to have an infinite level in the model; no water will ever overflow a thin dam.

To add a thin dam, click the corresponding icon from the *FM Region 2D / 3D* menu in the map ribbon (Figure 5.6). Adding, deleting, importing and exporting of a line feature such as a thin dam is discussed in more detail in section 5.4.2.3 on cross sections.

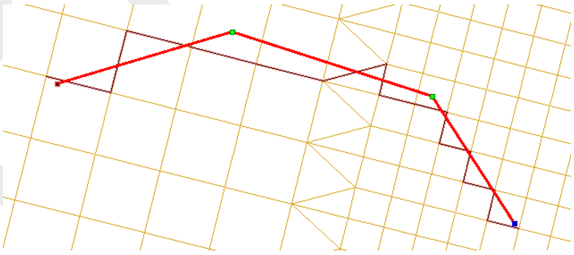
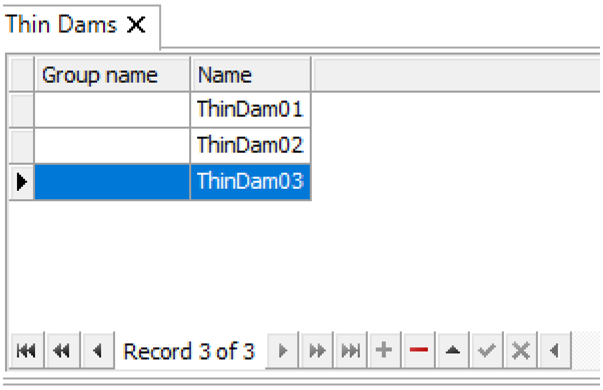


Figure 5.13: *Geographical and grid snapped representation of a thin dam*

When you double click the *Thin dams* attribute in the *Project* window, the corresponding Thin dams tab appears underneath the central map (Figure 5.14). Alternatively, you can also double click on any thin dam in the central map. Within this tab, an attribute table is shown which displays the names of all thin dams within your model. When one of the entries is selected, the corresponding item is highlighted in the central map. A thin dam entry can be deleted from the table via the context menu (right mouse button).



Thin Dams

	Group name	Name
		ThinDam01
		ThinDam02
▶		ThinDam03

Record 3 of 3

Figure 5.14: Attribute table with thin dams

5.4.2.5 Fixed weirs

A fixed weir (Figure 5.15) has the same function as a thin dam (section 5.4.2.4). However, unlike a thin dam, a fixed weir can be assigned both xy- and z-values. Furthermore, a fixed weir can be assigned a crest length (a thin dam is infinitely thin). The z-values correspond to the crest level of the fixed weir at the corresponding x- and y-locations; the level can vary in space, but is constant in time (Figure 5.16). Consequently, a fixed weir can overflow if the water level exceeds the crest level of the fixed weir. The level is specified with regard to the same vertical reference level as all other model items with level specifications (e.g. bed level values and initial water levels).

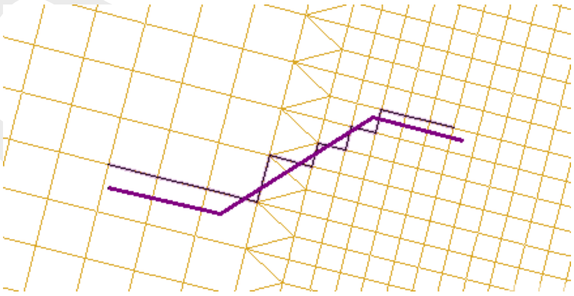


Figure 5.15: Geographical and grid snapped representation of a fixed weir

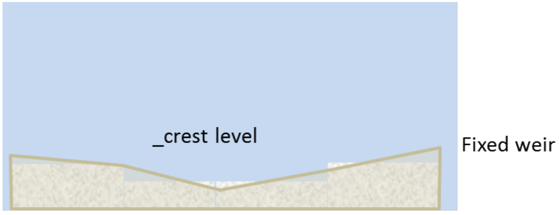
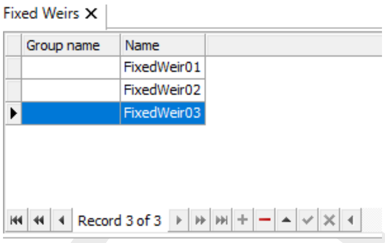


Figure 5.16: Schematic representation of a fixed weir

To add a fixed weir, click the corresponding icon from the *FM Region 2D / 3D* menu in the map ribbon (Figure 5.6). Adding, deleting, importing and exporting of a line feature such as a fixed weir is discussed in more detail in section 5.4.2.3 on cross sections.

When you double click the *Fixed weirs* attribute in the *Project* window, the corresponding Fixed weirs tab appears underneath the central map (Figure 5.17). Alternatively, you can also double click on any fixed weir in the central map. Within this tab, an attribute table is shown which displays the names of all fixed weirs within your model. When one of the entries is selected, the corresponding item is highlighted in the central map. A fixed weir entry can be deleted from the table via the context menu (right mouse button).

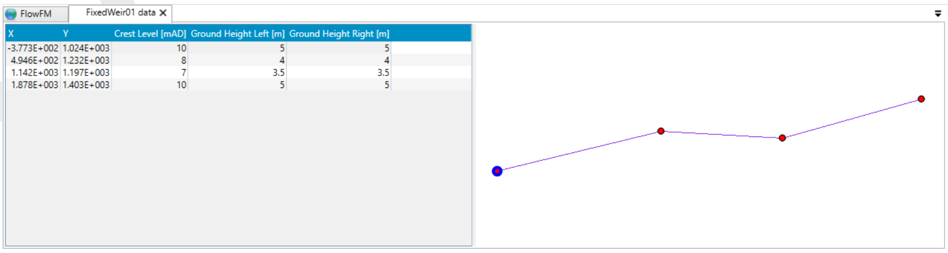


Group name	Name
	FixedWeir01
	FixedWeir02
	FixedWeir03

Figure 5.17: *Attribute table with fixed weirs*

When you double click on a fixed weir in the central map, the fixed weir editor opens in a separate view (Figure 5.18). On the right, a graphic representation (top view) of the fixed weir is displayed. The support point that is selected in the table is highlighted by means of a blue circle. On the left, a table is displayed showing the following properties of each support point of the fixed weir under consideration:

- X: x-coordinate of support point
- Y: y-coordinate of support point
- Crest level: crest level of the support point of support point of a fixed weir
- Ground height left: Difference between crest level and toe level at the left side
- Ground height right: Difference between crest level and toe level at the right side
- Crest width: Width of the crest of a fixed weir in perpendicular direction
- Slope left: Slope at left side of support point of a fixed weir
- Slope Right: Slope at right side at support point of support point of a fixed weir
- Roughness code: Roughness due to vegetation on a support point of a fixed weir



X	Y	Crest Level [mAD]	Ground Height Left [m]	Ground Height Right [m]
-3.773E+002	1.024E+003	10	5	5
4.946E+002	1.232E+003	8	4	4
1.142E+003	1.197E+003	7	3.5	3.5
1.878E+003	1.403E+003	10	5	5

Figure 5.18: *Fixed weir editor*

5.4.2.6 Land boundaries

A land boundary (Figure 5.19) encloses the main geographic features surrounding your model and indicates the intersection of the water and land masses. When you set up your computational grid in RGFGRID, a land boundary determines the onshore extent of your model. When you open RGFGRID to edit your grid, the land boundary is automatically transferred and displayed. For more details on grid generation, you are referred to the User Manual of RGFGRID (RGFGRID UM, 2016).

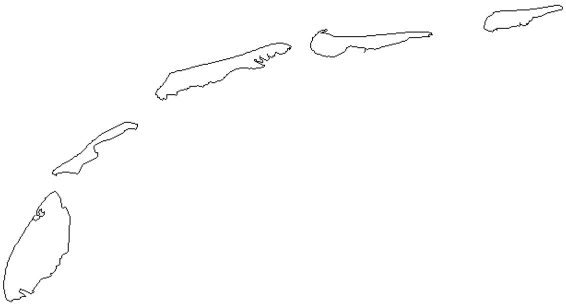
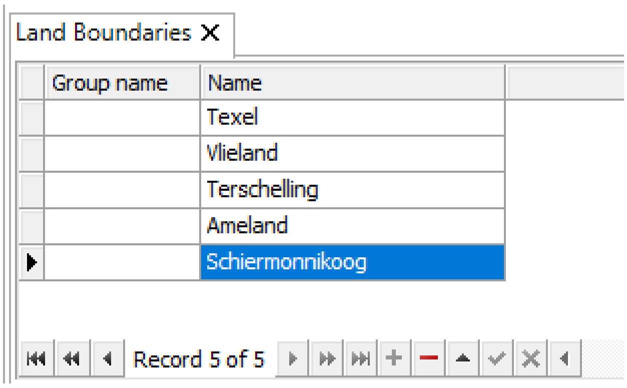


Figure 5.19: *Geographical representation of a land boundary*

To add a land boundary, click the corresponding icon from the *FM Region 2D / 3D* menu in the map ribbon (Figure 5.6). Adding, deleting, importing and exporting of a line feature such as a land boundary is discussed in more detail in section 5.4.2.3 on cross sections.

When you double click the *Land boundaries* attribute in the *Project* window, the corresponding land boundaries tab appears underneath the central map (Figure 5.20). Alternatively, you can also double click on any land boundary in the central map. Within this tab, an attribute table is shown which displays the names of all land boundaries within your model. When one of the entries is selected, the corresponding item is highlighted in the central map. A land boundary entry can be deleted from the table via the context menu (right mouse button).



Land Boundaries ✕

	Group name	Name
		Texel
		Vlieland
		Terschelling
		Ameland
▶		Schiermonnikoog

⏮ ⏪ ◀ Record 5 of 5 ▶ ⏩ ⏭ + − ▲ ✓ ✕ ◀

Figure 5.20: *Attribute table with land boundaries*

5.4.2.7 Dry points and Dry areas

Dry points are grid cells centred around a water level point that are permanently dry during a computation, irrespective of the local water depth and without changing the water depth as seen from the wet points. Dry areas are the same, but provide an easy way of defining many grid points as a single dry area at once.

Note that the flexibility of unstructured grids makes the use of dry points less necessary than with structured grid models, such as Delft3D-FLOW. In the interior of unstructured grids, some or more grid cells can easily be deleted during grid manipulation, e.g., in RGFGRID. Still, a dry points file can be used to explicitly mark locations or regions inside the grid as dry cells.

Dry points in the GUI

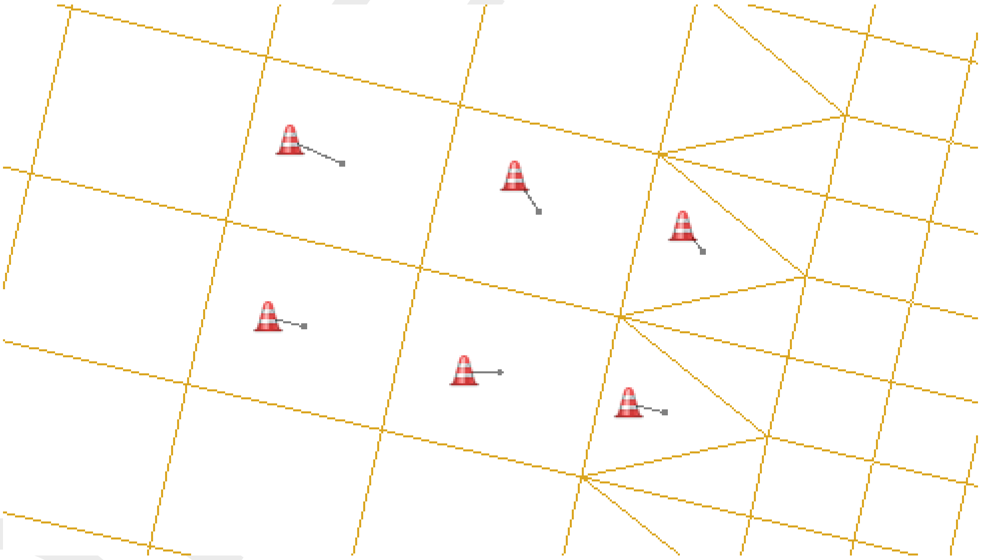
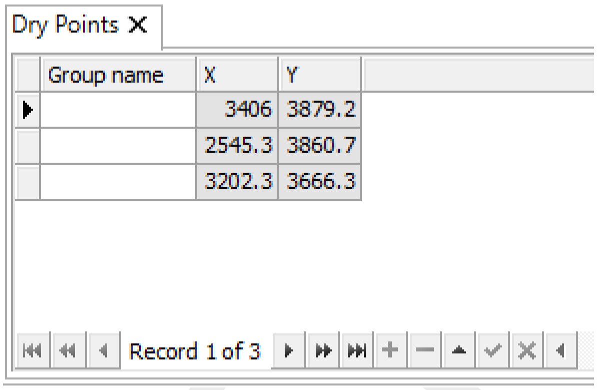


Figure 5.21: *Geographical and grid snapped representation of several dry points*

To add a dry point, click the corresponding icon from the *FM Region 2D / 3D* menu in the map ribbon (Figure 5.6). Adding, deleting, importing and exporting of a point feature such as a dry point is discussed in more detail in section 5.4.2.2 on observation points. The grid snapped representation of a dry point (Figure 5.21) is indicated by a line linking the dry point to the closest cell center.

When you double click the “Dry points” attribute in the *Project* window, the corresponding Dry points tab appears underneath the central map (Figure 5.22). Alternatively, you can also double click on any dry point in the central map. Within this tab, an attribute table is shown which displays the names of all dry points within your model. When one of the entries is selected, the corresponding item is highlighted in the central map. A dry point entry can be deleted from the table via the context menu (right mouse button).



Dry Points

Group name	X	Y
	3406	3879.2
	2545.3	3860.7
	3202.3	3666.3

Figure 5.22: *Attribute table with dry points*

Dry areas in the GUI

Dry areas (Figure 5.23), like dry points, indicate areas that permanently dry during a computation. Instead of adding many separate dry points, you can draw a polygon that encloses all required computational cells. Only cells which centers are strictly inside the polygon are taken into account. The grid snapped representation of the dry area clearly indicates which cells are considered within the dry area.

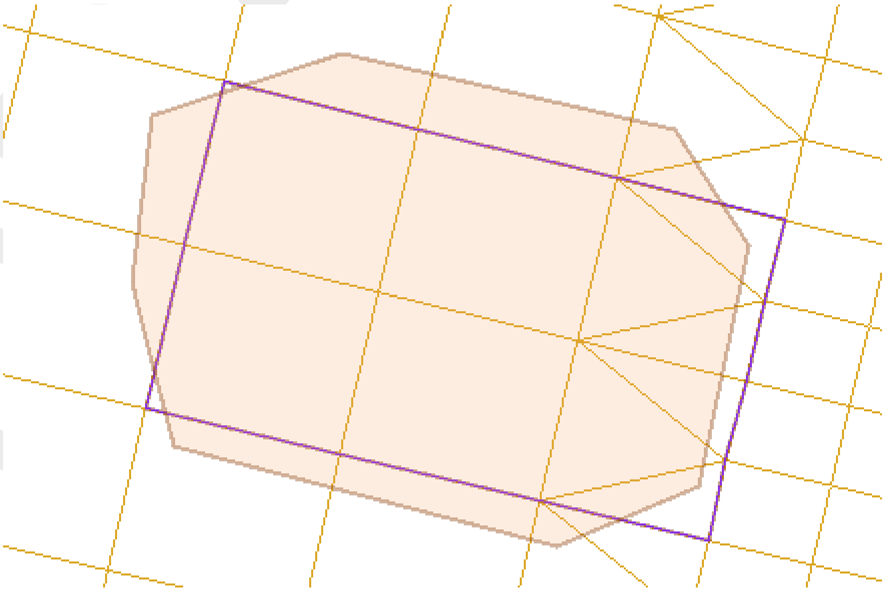
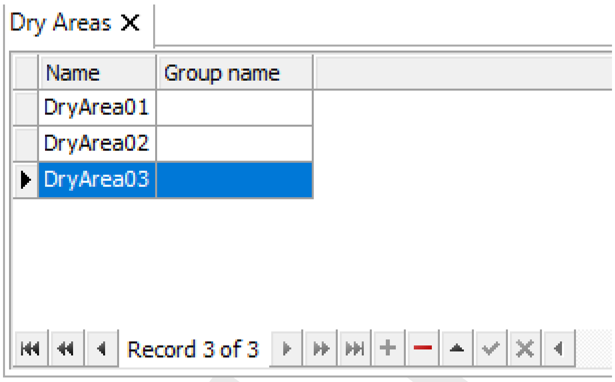


Figure 5.23: *Geographical and grid snapped representation of a dry area*

When you double click the *Dry areas* attribute in the *Project* window, the corresponding Dry areas tab appears underneath the central map (Figure 5.24). Alternatively, you can also double click on any dry area in the central map. Within this tab, an attribute table is shown which displays the names of all dry areas within your model. When one of the entries is selected, the corresponding item is highlighted in the central map. A dry area entry can be deleted from the table via the context menu (right mouse button).



Dry Areas ✕

Name	Group name
DryArea01	
DryArea02	
DryArea03	

Record 3 of 3

Figure 5.24: *Attribute table with dry areas*

Dry points file input

Dry points are defined by a sample file <∗.xyz>, dry areas are defined by a polygon file <∗.pol>. Add the filename to the MDU as below:

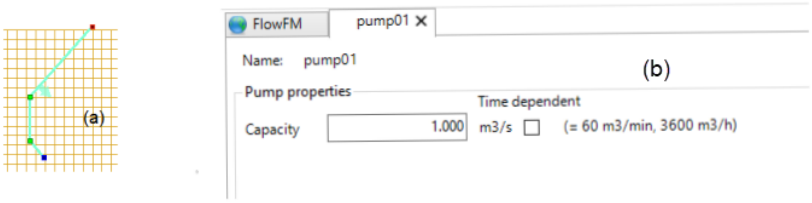
```
[geometry]
# ...
DryPointsFile = <filename.xyz> # Dry points file *.xyz, third column dummy z values,
                               # or polygon file *.pol.
```

The format of the sample file is defined in section C.3. The format of the polygon file is defined in section C.2. All grid cells that contain a sample point are removed from the model grid, and as a result do not appear in any of the output files. Alternatively, for a polygon file, all grid cells whose mass center lies within the polygon will be removed from the model grid.

5.4.2.8 Pumps

Pumps are a type of structures in D-Flow FM. Unlike the other structures, pumps can force the flow only on one direction. This direction is determined by arrow in D-Flow FM. The direction of pump can be reverted by mouse right-click and selecting "Reverse line(s)".

Like all other structures in D-Flow FM, the pump can be defined by a polygon. The input data of the pumps can be given by selecting and editing the pump polygon (see Figure 5.25). Right click on the pump polygon and selecting "Delete Selection" leads to deletion of the selected pump. Double clicking the pump polygons (or right click the pump in the list and select "Open view"), it opens a tab for editing the pump properties. The tab includes pump capacity. If the pump capacity is time dependent, it can be given by time series data (Figure 5.25).



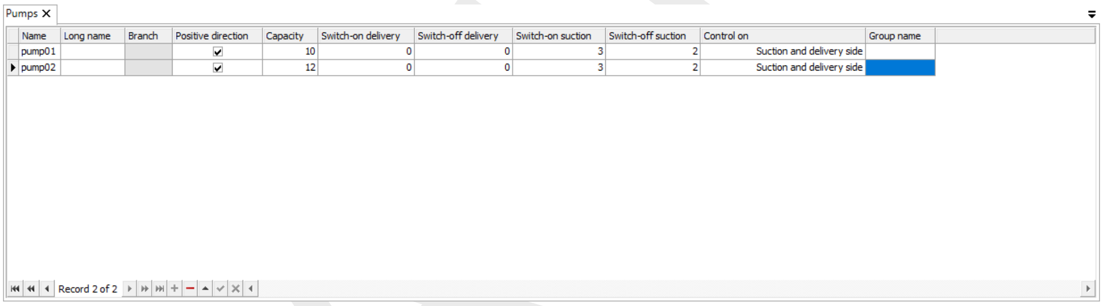
Name: pump01

Capacity 1.000 m3/s □ (= 60 m3/min, 3600 m3/h)

Figure 5.25: *Polygon for pump (a) and adjustment of physical properties (b).*

Right clicking the pumps attribute in the *Project* window opens a pop-down window on which you can select to import or export pumps. The pumps can be imported as polyline by a <∗.pli> file or by a structure file, and they can be exported as a <∗.pli> file, structure file, or <*shapefile*>.

Double clicking the pumps attribute in the *Project* window opens the pumps tab underneath the central map. The attribute table in this tab shows all pumps with their corresponding properties. When one of the pumps is selected, the corresponding item is highlighted in the central map. Double clicking any of the pumps in the central map opens the Structure Editor underneath the central map in which all parameters related to the pump can be set (Figure 5.26).



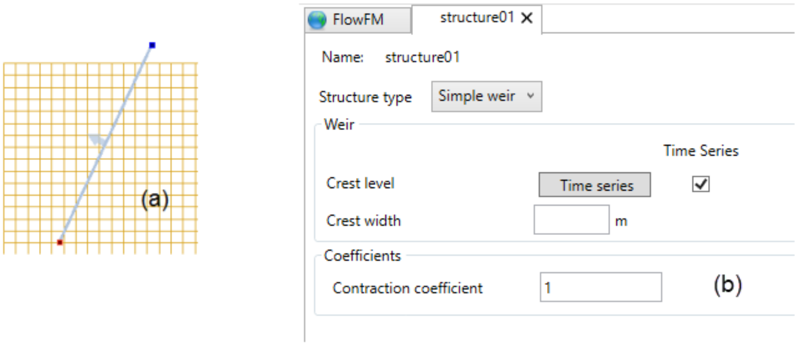
Pumps ✕

	Name	Long name	Branch	Positive direction	Capacity	Switch-on delivery	Switch-off delivery	Switch-on suction	Switch-off suction	Control on	Group name
	pump01			☑	10	0	0	3	2	Suction and delivery side	
▶	pump02			☑	12	0	0	3	2	Suction and delivery side	

Figure 5.26: *Selection of the pumps*

5.4.2.9 Weirs

Unlike the fixed weir, weir (or adjustable weir) can be adjusted based on the user requirements. To set an adjustable weir in the computational domain, you can select the icon *Add new structure (2D)* from the toolbar, and draw a line by mouse. This line includes direction, which defines the sign of total flux passing above the structure (positive flux in the direction of weir, otherwise negative). This direction can be inverted by mouse right-click and selecting *Reverse line(s)*. By double-click on the line, you can define this structure as *Simple weir* and add the geometrical and time-dependent parameters such as *Crest level*, *Crest width*, *Crest level time series* and *Lateral concentration coefficient* (See Figure 5.27).



Name: structure01

Structure type: Simple weir

Weir

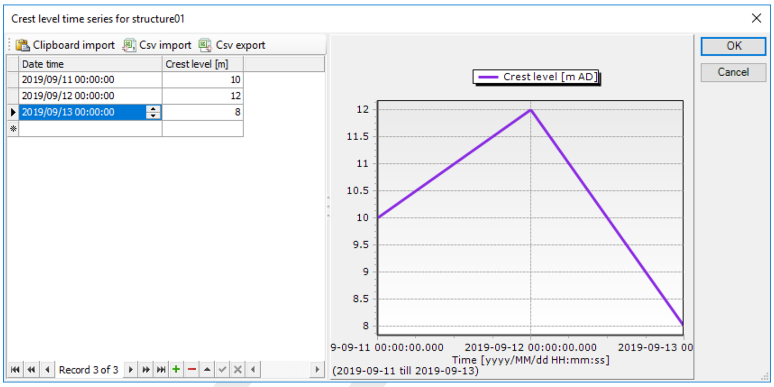
		Time Series
Crest level	Time series	☑
Crest width		m

Coefficients

Contraction coefficient: 1

Figure 5.27: *Polygon for adjustable weir (a) and adjustment of geometrical and temporal conditions (b).*

Moreover, the time series of the crest level can be set in the case the crest level is time dependent. The time dependency diagram can be defined by the help of time series diagram as shown in Figure 5.28. The time series can also be imported (and exported) from external <csv>-file.

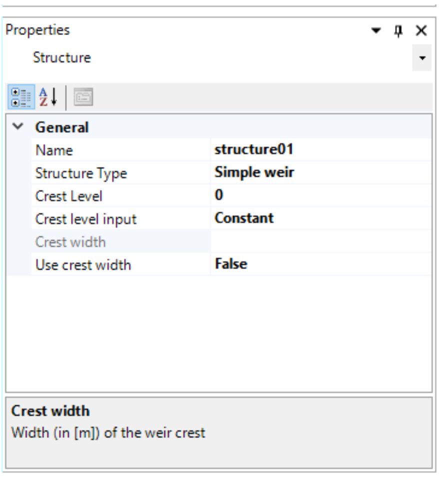


Date time	Crest level [m]
2019/09/11 00:00:00	10
2019/09/12 00:00:00	12
2019/09/13 00:00:00	8

Figure 5.28: *Time series for crest level.*

The weirs can be deleted, imported and exported. By right clicking on the weir polygon, and selecting "Delete Selection" from the pop-down window, you can delete the selected weir. Right clicking the "Weirs" attribute in the *Project* window opens the "Weirs" tab opens the options for import and export. You can import weirs as polygon (<∗.pli> file) or as a structure by structure file. The weirs can also be exported to a polygon file, to a structure data file, or by the help of a shape file.

Double clicking the "Weirs" attribute in the *Project* window opens the "Weirs" tab underneath the central map. The attribute table in this tab shows all weirs with their corresponding properties. When one of the weirs is selected, the corresponding item is highlighted in the central map. Double clicking any of the weirs in the central map opens the Structure Editor as a new map view in which all parameters related to the weir can be set (Figure 5.29).



General

Name	structure01
Structure Type	Simple weir
Crest Level	0
Crest level input	Constant
Crest width	
Use crest width	False

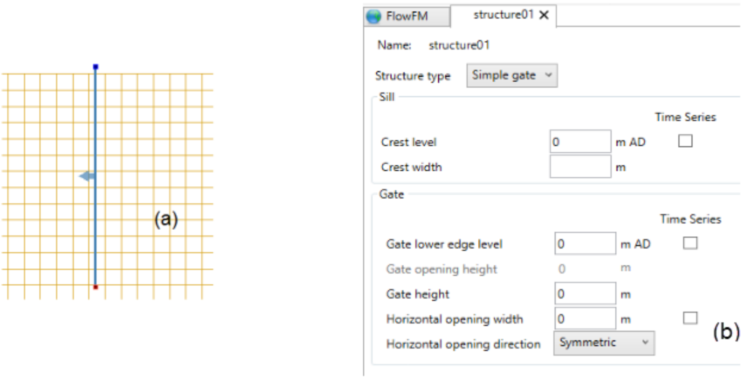
Crest width
Width (in [m]) of the weir crest

Figure 5.29: *Time series for crest level.*

5.4.2.10 Gates

In D-Flow FM the gates can be imposed by polygon, and can be edited in a similar way as the other structures (see Figure 5.30). Like the other structures, mentioned above, the gates can be imported and exported by means of structure file or <∗.pli> file.

Figure 5.30 shows the edit tab of the gate properties. The gate can be opened horizontally, as well as vertically.



Name: structure01

Structure type: Simple gate

Sill

			Time Series
Crest level	0	m AD	□
Crest width		m	

Gate

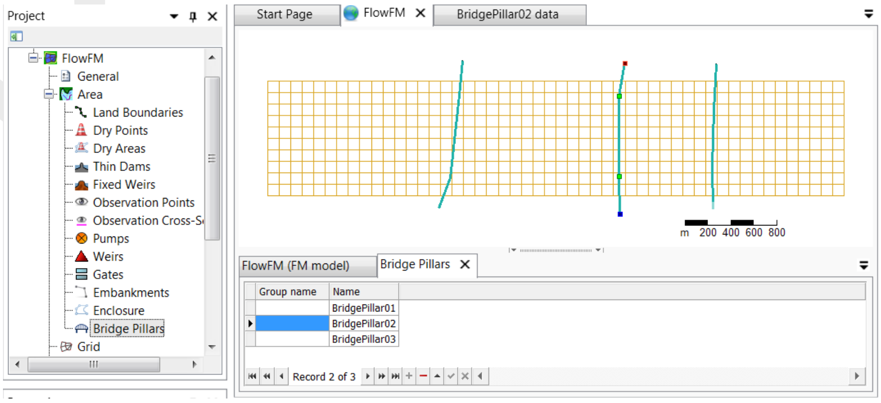
			Time Series
Gate lower edge level	0	m AD	□
Gate opening height	0	m	
Gate height	0	m	
Horizontal opening width	0	m	□
Horizontal opening direction	Symmetric		

Figure 5.30: *Polygon for gate (a) and adjustment of geometrical and temporal conditions (b).*

5.4.2.11 Bridge Pillars

Bridge pillars are a type of structures in D-Flow FM that are relevant for the hydrodynamics. In D-Flow FM, a bridge is defined by a polygon on the central map. The user can add a bridge to the model by

- ◇ selecting the *Add new bridge pillar* icon in the *Map* ribbon; and
- ◇ defining the location of the bridge pillars by clicking on the map. The polyline for a bridge is finalized by double-clicking on the last bridge pillar.)
- ◇ repeat the above for another bridge and end by pressing the *<ESC>* key



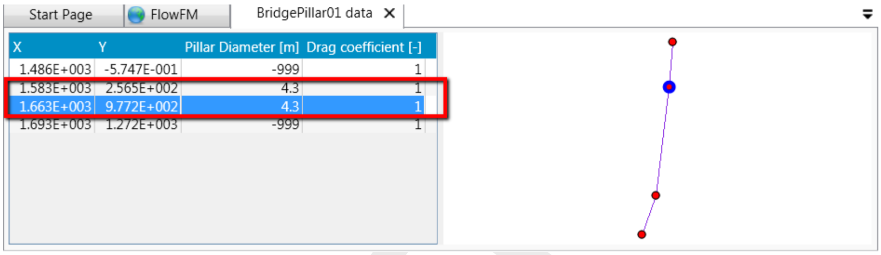
Group name	Name
	BridgePillar01
	BridgePillar02
	BridgePillar03

Figure 5.31: *Selection of a bridge*

Right clicking on the bridge polygon and selecting *Delete Selection* leads to deletion of the selected bridge. After double clicking a bridge polygon or right clicking the bridge in the list and selecting *Open view*, a tab for editing the bridge properties opens (Figure 5.32), with the following parameters:

- ◇ Pillar diameter [m] - (use -999.0 for an auxiliary location)

◇ Drag coefficient [-]: default is 1.0



X	Y	Pillar Diameter [m]	Drag coefficient [-]
1.486E+003	-5.747E-001	-999	1
1.583E+003	2.565E+002	4.3	1
1.663E+003	9.772E+002	4.3	1
1.693E+003	1.272E+003	-999	1

***Figure 5.32:** Adjustment of the properties of the pillars*

Right clicking the *Bridge Pillars* attribute in the *Project* window (Figure 5.31) opens a pop-down window on which you can select to import or export bridges. The bridges can be imported and exported as polyline by a <∗.pliz> file or <*shapefile*>.

As can be seen in Figure 5.32, some of the points of the polygon are relevant pillars with real dimensions and some are not. For example, if a bridge is of curved shape, then additional points can be used to visualize such a bridge, while some of the points of a polyline do not coincide with a pillar. Such points are modelled with a pillar diameter of -999.0.

5.4.3 Computational grid

To set up a grid, click on the *Edit grid* button which opens the program RGFGRID. All features of grid setup in RGFGRID are treated separately in RGFGRID UM (2016). If a land boundary is present in your project, this is exported to RGFGRID automatically. Once you have setup your grid in RGFGRID, click *File* →*Save Project* and close the program. The grid will now be visible within the central map. Editing of the grid remains possible at any point in time during the setup of your model by means of clicking the *edit grid* button. Any changes you make are always saved after clicking *File* →*Save Project* and loaded back into the central map.

5.4.4 Bed level

When you double click on *Bed level* in the *Project* window or select *Bed level* from the drop-down box in the spatial editor section of the *Map* ribbon, the spatial editor is activated (Figure 5.33). This editor can be used to generate a bathymetry for your computational grid. How to work with the spatial editor is described in Appendix H. Be aware that the bathymetry in D-Flow FM is defined as the bed level (e.g. positive upward), implying that all bed levels below the reference plane are negative. By default the bed levels are defined on the net nodes.

Note: Please note that, currently, other bed level definition types (e.g. `BedlevType`s) are not visually supported by the GUI. If you would like to switch the bed level defintions to another type, you have to set the `BedlevType` in the *Physical Parameters* tab. However, the bed level locations will not be updated accordingly in the central map.

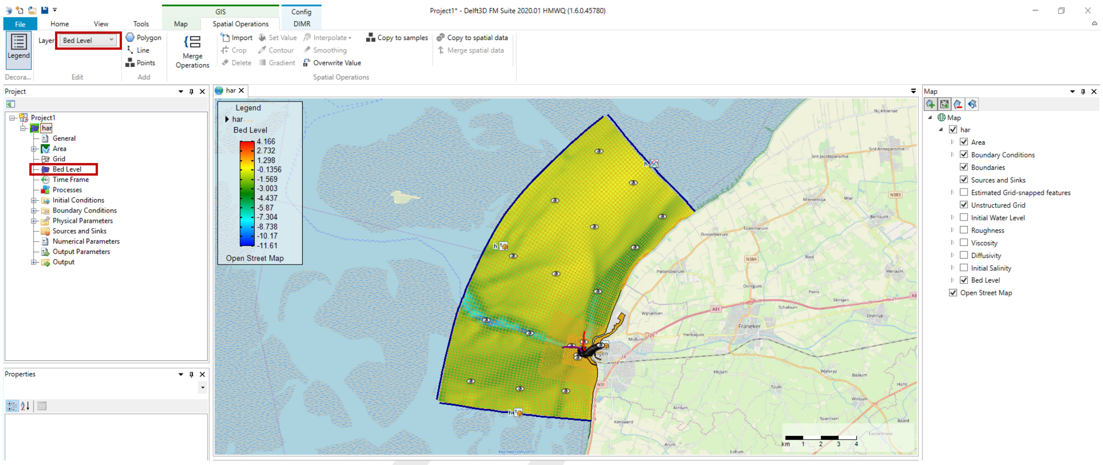
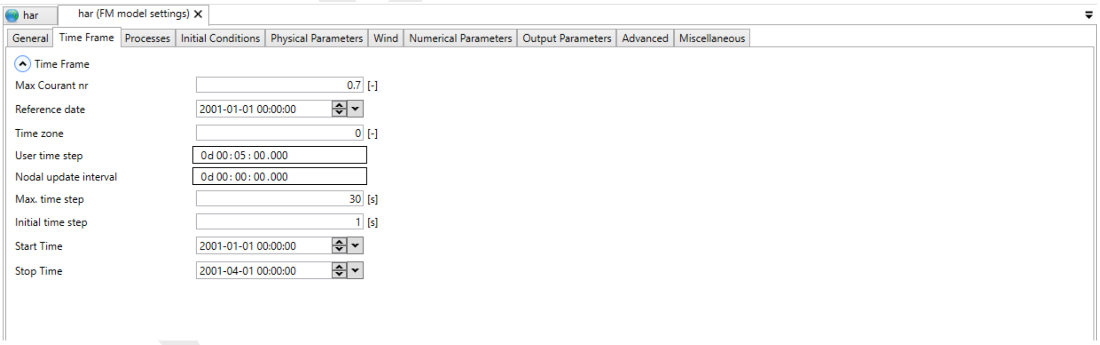


Figure 5.33: *Bed level activated in the spatial editor*

5.4.5 Time frame

In the settings tab, in the sub-tab time frame (Figure 5.34), you can specify everything related to the time frame in which your model will run.



Time Frame

Max Courant nr	0.7	[-]
Reference date	2001-01-01 00:00:00	
Time zone	0	[-]
User time step	0d 00 : 05 : 00 .000	
Nodal update interval	0d 00 : 00 : 00 .000	
Max. time step	30	[s]
Initial time step	1	[s]
Start Time	2001-01-01 00:00:00	
Stop Time	2001-04-01 00:00:00	

Figure 5.34: *Overview time frame tab*

In general, the time frame is defined by a reference date and a start and stop time. The time step size of your model is automatically limited (every time step) based on a Courant condition. In more detail, you must define the following input data:

Max Courant nr — The maximum allowed Courant number, which is used to compute the time step size from the CFL criterium. D-Flow FM uses an explicit advection scheme, therefore a value of 0.7 or lower is advised.

Remark:

◇ You should check the influence of the time step on your results at least once.

Reference date — The reference date and time of the simulation. It defines the (arbitrary) $t = 0$ point for all time-series as used in the simulation. In the GUI, time-specifications are always absolute, but in the underlying model input files, these are stored as time values relative to the reference date. Typically, input time-series files are specified in minutes after this $t = 0$ point. See for an illustration Figure 5.35.

Time zone — The time difference between local time and UTC.
The time zone is defined as the time difference (in hours) between the local time (normally used as the time frame for D-Flow FM) and Coordinated Universal Time (UTC). The time difference can consist

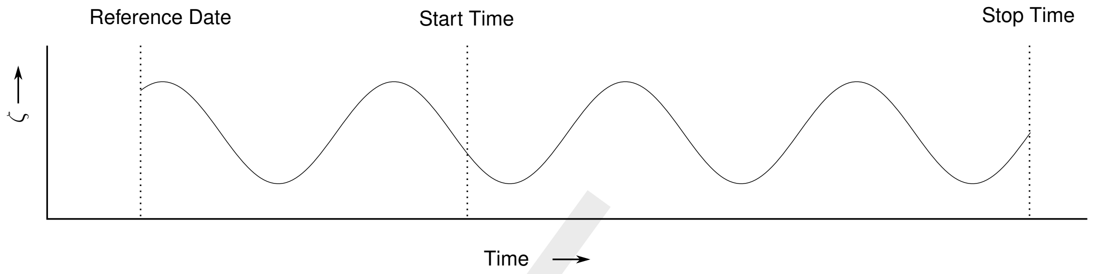


Figure 5.35: *Relation between Reference Date and the simulation start and stop time for astronomic- and harmonic-series as used in the simulation. Time-series should cover the simulation time.*

of a part of an hour, for example 1 hour and 30 minutes (specify as a decimal value: 1.5 (hours)). The local Time Zone is used for two processes:

- ◇ To determine the phases in local time of the tidal components when tide generating forces are included in the simulation, see section 8.12.
- ◇ To compare the local time of the simulation with the times at which meteo input is specified, e.g., wind velocities and atmospheric pressure. These can be specified in a different time zone.

If the *Time Zone* = 0 then the simulation time frame will refer to UTC.

User time step The interval that is highest in the hierarchy. It specifies the interval with which the meteorological forcings are updated. The *Max. time step* cannot be larger than the *User time step*, and it will automatically be set back if it is. Also, the output intervals should be a multiple of this *User time step*, see Appendix F. Finally the computational time steps will be fitted to end up exactly at each *User time step*, such that proper equidistant output time series are produced.

Nodal update interval When using astronomic boundary conditions, the nodal factors can be updated with certain intervals, see section 8.12.

Max. time step The *Max. time step* is the upper limit for the computational time step. The automatic time step can not be switched off explicitly. (If you want to enforce a fixed time step anyway, set the parameter *Max. time step (s)* to the desired step size, and the parameter *Max. Courant nr.* to an arbitrary high value.)

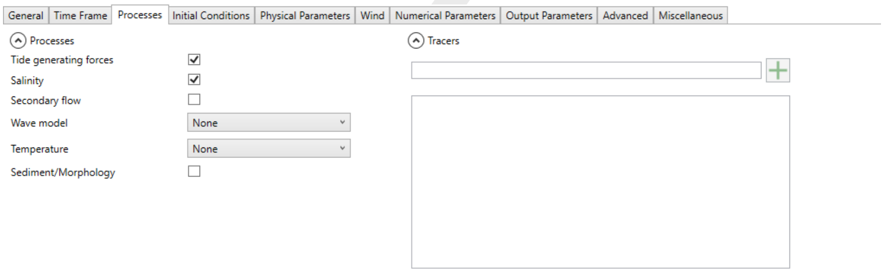
Initial time step the initial time step of the model; there is no data available yet during the first time step to compute the time step automatically based on a Courant condition. The computational time step then gradually increases from *Initial time step* to the CFL-number limited time step (assuming that *Initial time step* is relatively small).

Start Time The start date and time of the simulation.

Stop Time The stop date and time of the simulation. Always make sure that the model *Stop Time* is larger than the model *Start Time* to avoid errors during your calculation.

5.4.6 Processes

In the processes tab (Figure 5.36) you can specify which processes you want to incorporate into your model. You can choose whether or not to include tidal forcing, salinity, temperature and sediment/morphology by means of check boxes. In addition, you can specify which *Wave model* you want to use. Note that when ticking the sediment/morphology check box, two tabs for setting sediment and morphology parameters appear.



Tide generating forces ☑
Salinity ☑
Secondary flow □
Wave model: None
Temperature: None
Sediment/Morphology □

Figure 5.36: *Overview processes tab*

5.4.7 Initial conditions

When expanding the initial conditions in the *Project* window, all quantities requiring an initial state are shown (Figure 5.37). The number of quantities depends on the activated physical processes in the 'Processes' tab (see section 5.4.6). The initial conditions for each quantity can be specified as a uniform value or as a coverage (e.g. a spatially varying field).

The uniform values can edited in the 'Initial Conditions' tab, which opens upon double clicking 'Initial Conditions' in the *Project* window (Figure 5.38). In this tab you can also specify the layer distribution for the initial condition specification in case of a 3 dimensional quantity (i.e. salinity). **Note:** Please note that for 3 dimensional initial conditions currently only the option 'top-bottom' is supported.

In case of spatially varying initial conditions you can double click the quantity in the *Project* window or select it from the drop-down box in the spatial editor section of the Special Operations ribbon (Figure 5.39). Then the spatial editor is activated, which you can use to edit spatially varying fields. For more information on how to use the spatial editor you are referred to Appendix H. In case of 3 dimensional initial conditions, you can select the layer from the quantity dropdown box in the 'Map' ribbon (Figure 5.40).

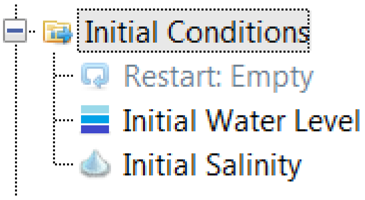
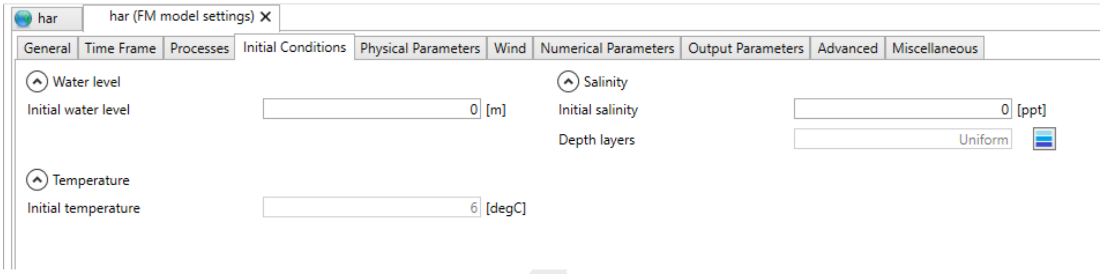


Figure 5.37: *Initial conditions in the* Project *window*



Water level

Initial water level 0 [m]

Temperature

Initial temperature 6 [degC]

Salinity

Initial salinity 0 [ppt]

Depth layers Uniform

Figure 5.38: *The 'Initial Conditions' tab where you can specify the uniform values and the layer distributions of the active physical quantities.*

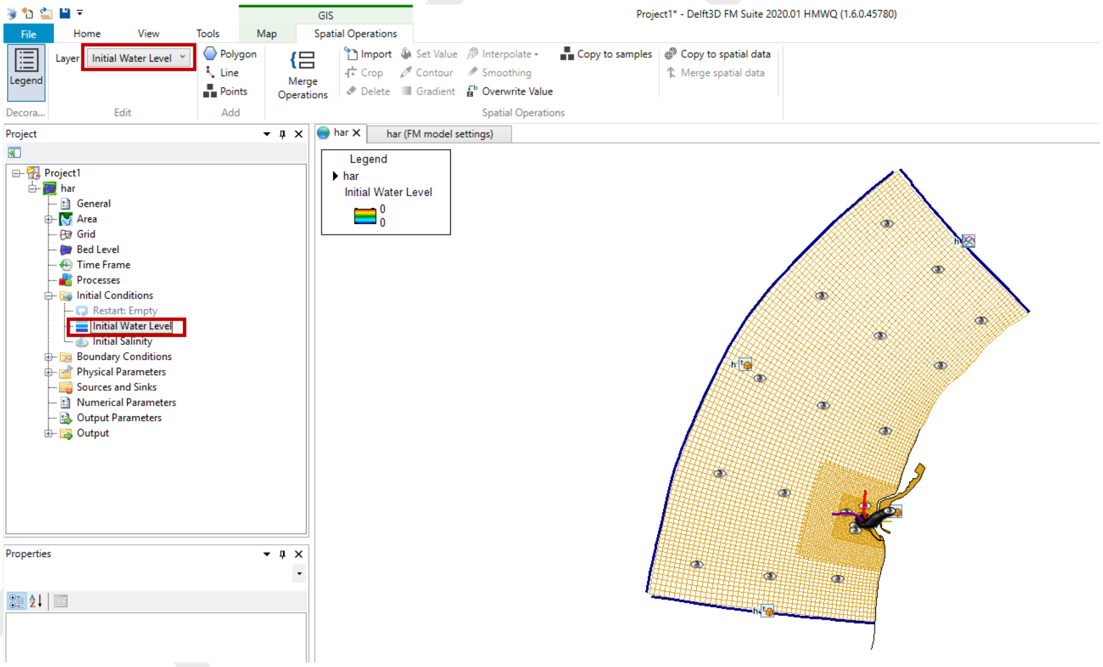


Figure 5.39: *Initial water levels activated in the spatial editor*

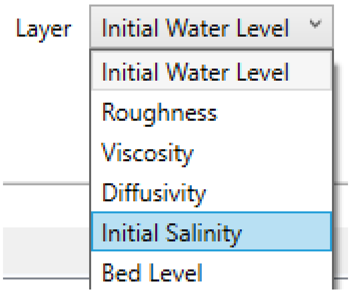
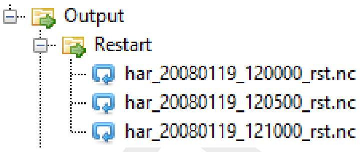


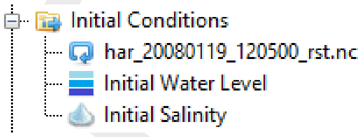
Figure 5.40: *Selecting 3 dimensional initial fields from the dropdown box in the 'Map' ribbon to edit them in the spatial editor*

Instead of defining initial conditions from scratch you can also import fields from a previous computation (using restart files). When you run a model using the D-Flow FM GUI which is writing restart files, the restart states will appear in the "Output" folder in the *Project* window (Figure 5.41). For a description of the specification of restart files, please refer to paragraph section 5.4.12. To use a restart file as initial conditions, apply the right mouse button and select "Use as initial state". The file will now appear under "Initial Conditions" in the *Project* window (Figure 5.42). To activate the restart file utilize the right mouse button and select “Use restart”. The file will no longer be grey, but is now highlighted in black. The model will now restart from this file. Notice that the simulation still starts at the original *User Start Time*, rather than the time of the restart file. (The restart file only provides initial conditions.)



Output
Restart
har_20080119_120000_rst.nc
har_20080119_120500_rst.nc
har_20080119_121000_rst.nc

Figure 5.41: *Restart files in output states folder*



Initial Conditions
har_20080119_120500_rst.nc
Initial Water Level
Initial Salinity

Figure 5.42: *Restart file in initial conditions attribute*

5.4.8 Boundary conditions

Boundary conditions consist of a location specification (‘support points’) and a forcing for that location.

Section 5.4.8.1 describes how support points can be specified in D-Flow FM User Interface. The boundary forcing can be specified in the boundary data editor.

Section 5.4.8.2 describes the functionality of the boundary data editor. Finally, section 5.4.8.4 describes how the user can get an overview of the boundary locations and forcing in the attribute table.

5.4.8.1 Specification of boundary locations (support points)

In D-Flow FM the boundary locations are defined as ‘support points’ on a polyline (<∗.pli>). The user can add a boundary polyline (<∗.pli>) in the central map by selecting the ‘Add Boundary’ icon in ’FM Region 2D / 3D’ of the ‘Map’ ribbon (see Figure 5.43). The number of individual mouse clicks determines the number of support points on the polyline. The polyline is closed by a double click. Once the polyline is added it becomes visible in the *Project* window under ‘Boundary Conditions’ (see Figure 5.44). The polyline can be edited by the general edit operations in the ‘Map’ ribbon (i.e. add/delete/move individual geometry points or the complete geometry, see Figure 5.45). The name of the polyline (or ‘boundary’) can be edited in the Boundaries tab, which can be opened by double clicking ‘Boundaries’ in the *Map* window (see Figure 5.46).

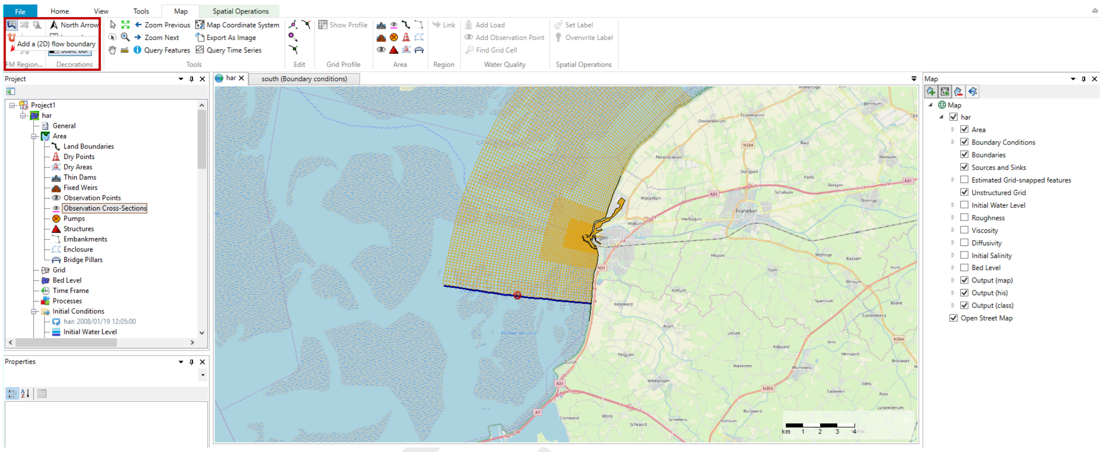


Figure 5.43: *Adding a boundary support point on a polyline in the central map. By double clicking on the polyline in the map, the boundary condition editor will open to edit the forcing data on the polyline.*

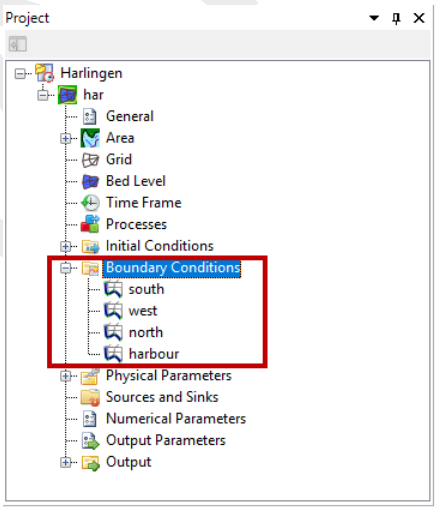


Figure 5.44: *Polyline added in* Project *window under 'Boundary Conditions'. By double clicking on the name of the polyline, the boundary condition editor will open to edit the forcing data on the polyline.*

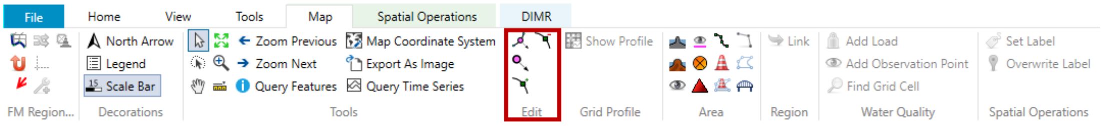
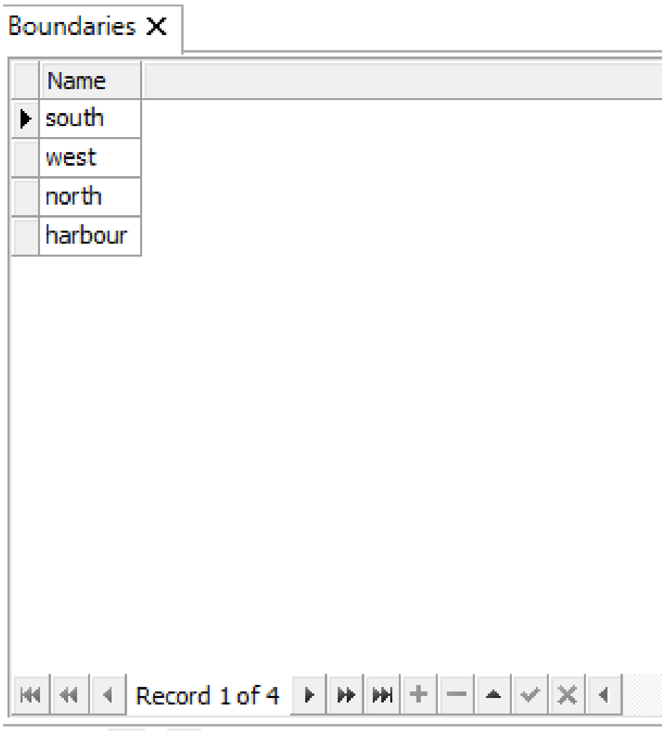


Figure 5.45: *Geometry edit options in* Map *ribbon*

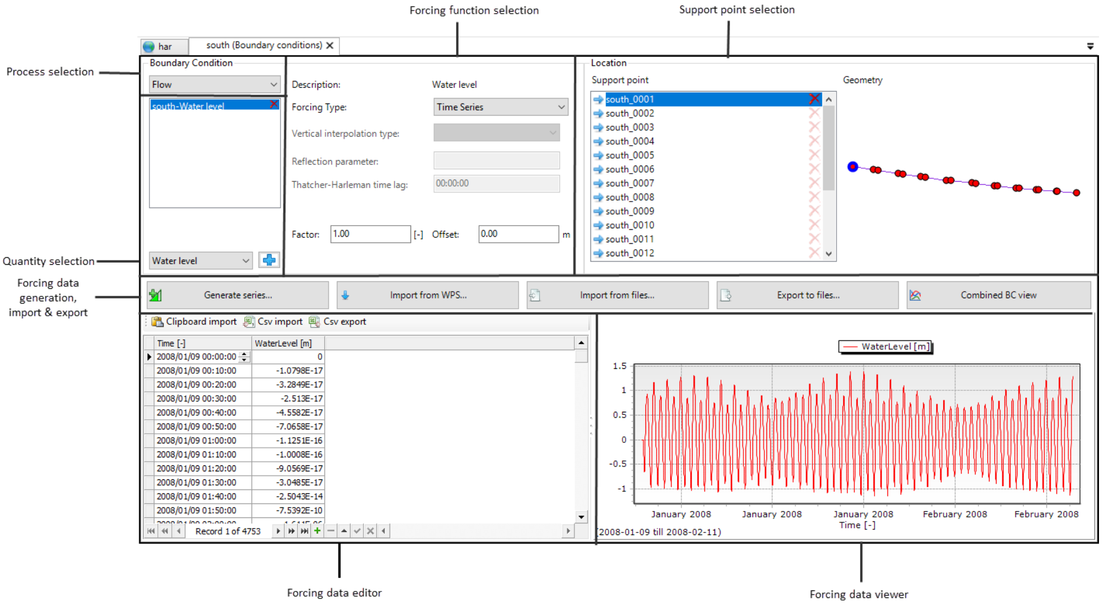


Name
south
west
north
harbour

__Figure 5.46:__ Edit name of polyline/boundary in Boundaries tab

5.4.8.2 Boundary data editor (forcing)

The boundary condition editor can be opened either by double clicking on the boundary name in the *Project* window (Figure 5.44) or by double clicking the boundary polyline in the map window (Figure 5.43). An overview of the boundary data editor is given in Figure 5.47. This editor can be used to specify the boundary forcing for different quantities (i.e. water level, velocity, discharge, salinity, etc.) corresponding to different processes (i.e. flow, salinity, temperature, tracers). The user can select the processes and quantities in the upper left corner. In the upper centre panel the user can select the forcing function for the selected quantity (i.e. time series, harmonic components, astronomical components or Q-h relation). The upper right corner contains a list of the support points on the polyline. The geometry view shows the location of the selected support point on the polyline (<∗.pli>). In the middle panel are some handy buttons to generate, import and export forcing data. In the lower left panel the user can specify the boundary data for the selected support point. The lower right corner shows the signal of the boundary data. The following sections describe the features of the boundary data editor in more detail.



Boundary Condition: Flow

south-Water level

Water level

Description: Water level

Forcing Type: Time Series

Vertical interpolation type:

Reflection parameter:

Thatcher-Harleman time lag: 00:00:00

Factor: 1.00 [-] Offset: 0.00 m

Support point:

- south_0001
- south_0002
- south_0003
- south_0004
- south_0005
- south_0006
- south_0007
- south_0008
- south_0009
- south_0010
- south_0011
- south_0012

Time [-]	WaterLevel [m]
2008/01/09 00:00:00	0
2008/01/09 00:10:00	-1.0798E-17
2008/01/09 00:20:00	-3.2849E-17
2008/01/09 00:30:00	-2.513E-17
2008/01/09 00:40:00	-4.5582E-17
2008/01/09 00:50:00	-7.0658E-17
2008/01/09 01:00:00	-1.1251E-16
2008/01/09 01:10:00	-1.0008E-16
2008/01/09 01:20:00	-9.0569E-17
2008/01/09 01:30:00	-3.0485E-17
2008/01/09 01:40:00	-2.5043E-14
2008/01/09 01:50:00	-7.5392E-10

Record 1 of 4753

Figure 5.47: *Overview of the boundary data editor*

Process and quantity selection:

Currently, D-Flow FM supports the processes flow, salinity, temperature and tracers (**Note: tracers are not yet fully editable**). The processes available in the boundary condition editor depend on the selected processes in the processes tab (see section 5.4.6). After selecting the process from the dropdown box the user can select one of the corresponding quantities, as illustrated in Figure 5.48. For the process flow the user can choose from five principal quantities: water level, velocity, Riemann invariant, Neuman gradient and discharge. All quantities are specified per support point, except for discharges which are specified per polyline (<∗.pli>). A support point can have multiple forcing quantities of the same type (i.e. water level, velocity, Riemann, Neumann or discharge). These quantities are added up. This can be relevant to add a surge level to an astronomical water level for example. Furthermore, the user can apply normal and tangential velocities as ‘add on’ quantities. The following combinations of quantities are allowed:

- Water level + normal velocity
- Water level + tangential velocity
- Water level + normal velocity + tangential velocity
- Velocity (= normal) + tangential velocity
- Riemann + tangential velocity

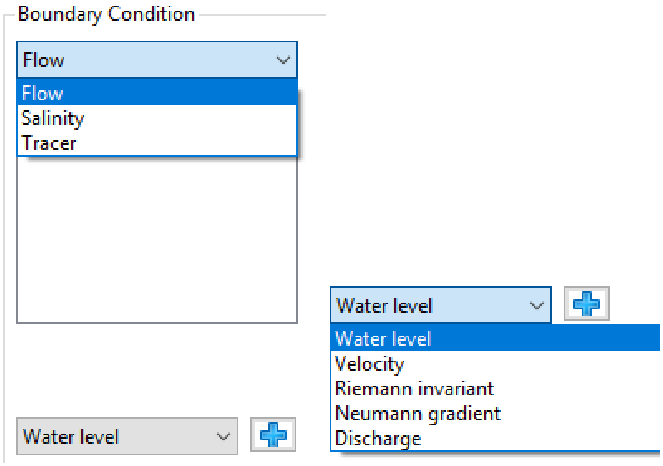


Figure 5.48: *Process and quantity selection in the boundary data editor*

Forcing function selection:

The user can select one of the following forcing functions:

- Time series
- Harmonic components
- Harmonic components + correction
- Astronomic components
- Astronomic components + correction
- Q-h relation (only for water levels)

The next section describes how data can be added for these types of forcing functions.

Add forcing data to a support point or polyline:

By default all support points on the polyline are deactivated. To add forcing data to a support point the user first needs to activate it by pressing the green ‘add’-symbol in the list of support points (see Figure 5.49). Consequently, the user can specify the forcing data in the lower left panel based on the selected forcing function. Of which a preview is shown in the lower right panel. **Note: Please note that once a support point containing forcing data is made inactive, all data on the support point is lost!**

The user can choose from the forcing functions time series, harmonic components (+ correction), astronomic components (+ correction) and Q-h relation. Examples of the boundary data specifications for these different forcing functions are given below. **Note: Please note that once the user changes the forcing function of a polyline, all data on the polyline is lost!**

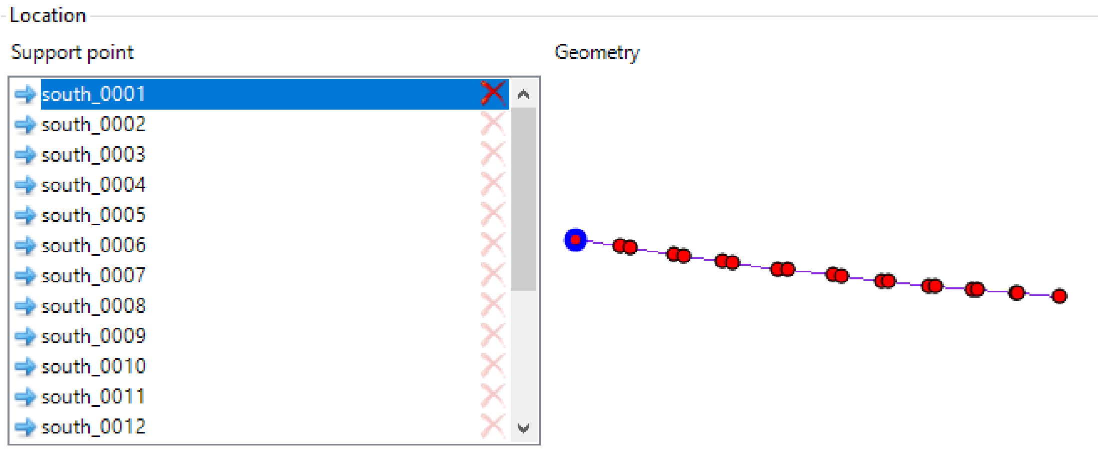
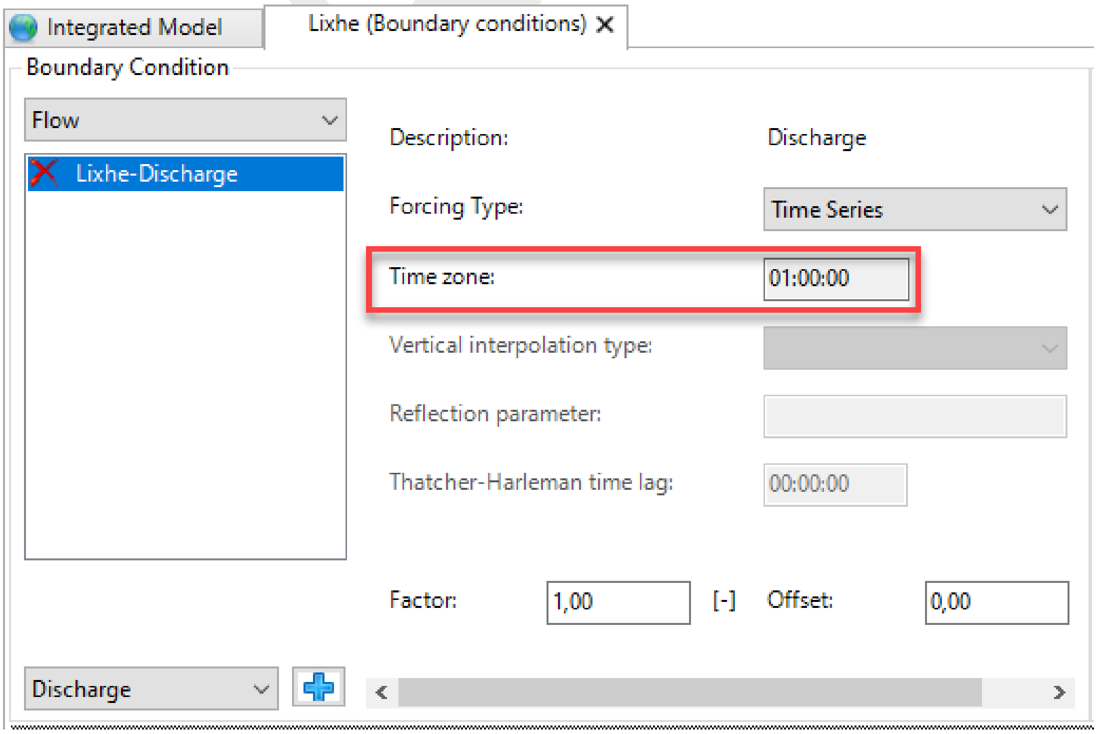


Figure 5.49: Activate a support point

Time series

Note: Time series have an attribute `Time zone`, as can be seen below.



Description: Discharge

Forcing Type: Time Series

Time zone: 01:00:00

Vertical interpolation type:

Reflection parameter:

Thatcher-Harleman time lag: 00:00:00

Factor: 1,00 [-] Offset: 0,00

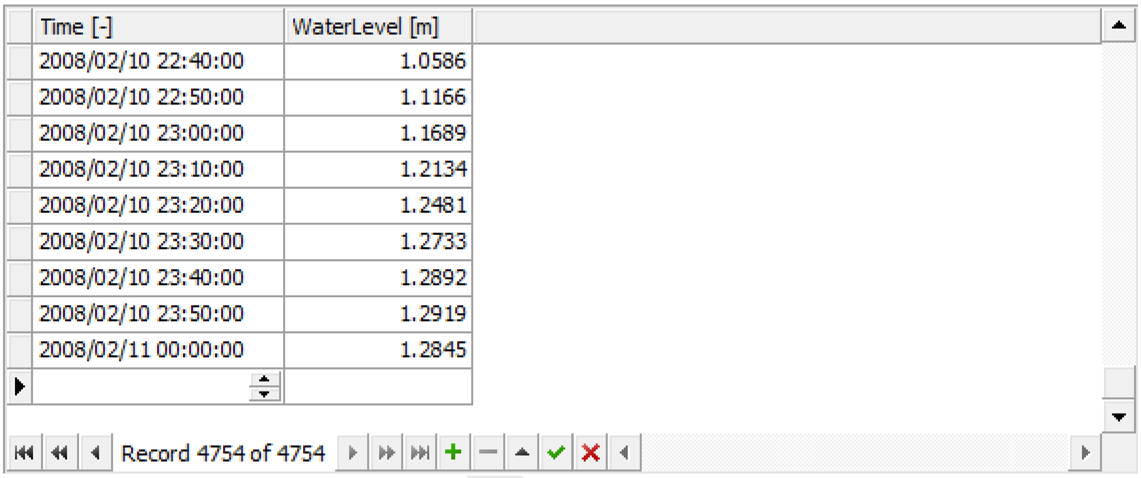
Figure 5.50: Time zone in the Boundary conditions editor

This attribute is un-editable, as it belongs to the data. The `Time zone` is defined as the time difference between local time and UTC. By default the `Time zone` is `0`. In case external data is used, the `Time zone` can be different - as in Figure 5.50. The `Time zone` will be used to transpose the (external) data to the `Time zone` of the time frame of the simulation (see also section 5.4.5).

The time format for time series is yyyy-mm-dd HH:MM:SS.

There are multiple ways to specify time series for the selected quantity:

- ◇ Specify the time series step by step in the table (Figure 5.51): the user can add or delete rows with the “plus”- and “minus”-signs below the table.

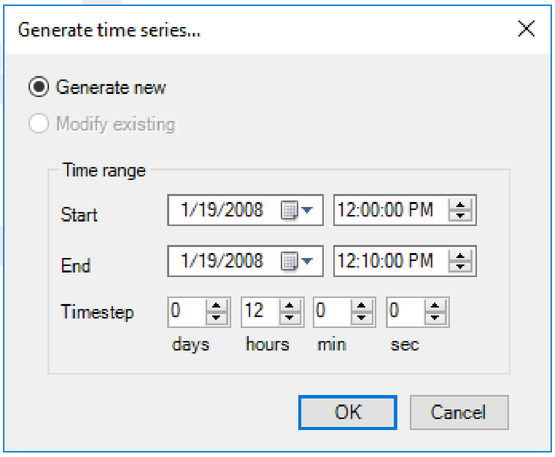


Time [-]	WaterLevel [m]
2008/02/10 22:40:00	1.0586
2008/02/10 22:50:00	1.1166
2008/02/10 23:00:00	1.1689
2008/02/10 23:10:00	1.2134
2008/02/10 23:20:00	1.2481
2008/02/10 23:30:00	1.2733
2008/02/10 23:40:00	1.2892
2008/02/10 23:50:00	1.2919
2008/02/11 00:00:00	1.2845

Record 4754 of 4754

Figure 5.51: *Specification of time series in the boundary data editor (left panel)*

- ◇ Generate time series using the ‘Generate time series’ button (Figure 5.52): the user can specify start time, stop time and time step.



Generate time series...

● Generate new

○ Modify existing

Time range

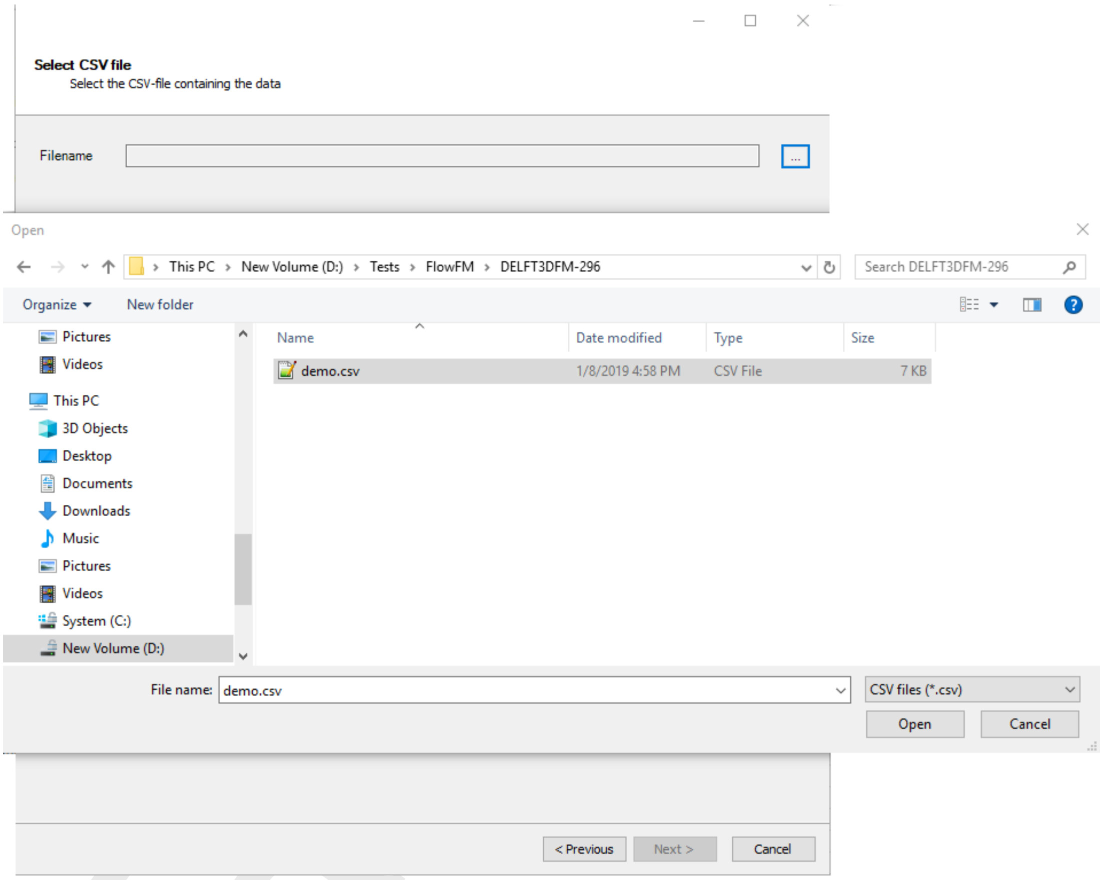
Start 1/19/2008 12:00:00 PM

End 1/19/2008 12:10:00 PM

Timestep 0 days 12 hours 0 min 0 sec

Figure 5.52: *Window for generating series of time points*

- ◇ Import from csv using the ‘Csv import’ button: a wizard will open in which the user can (1) select a csv-file (Figure 5.53), (2) specify how data should be parsed into columns (Figure 5.54) and (3) how the values should be parsed and mapped into columns (Figure 5.55).



Select CSV file

Select the CSV-file containing the data

Filename

Open

This PC › New Volume (D:) › Tests › FlowFM › DELFT3DFM-296

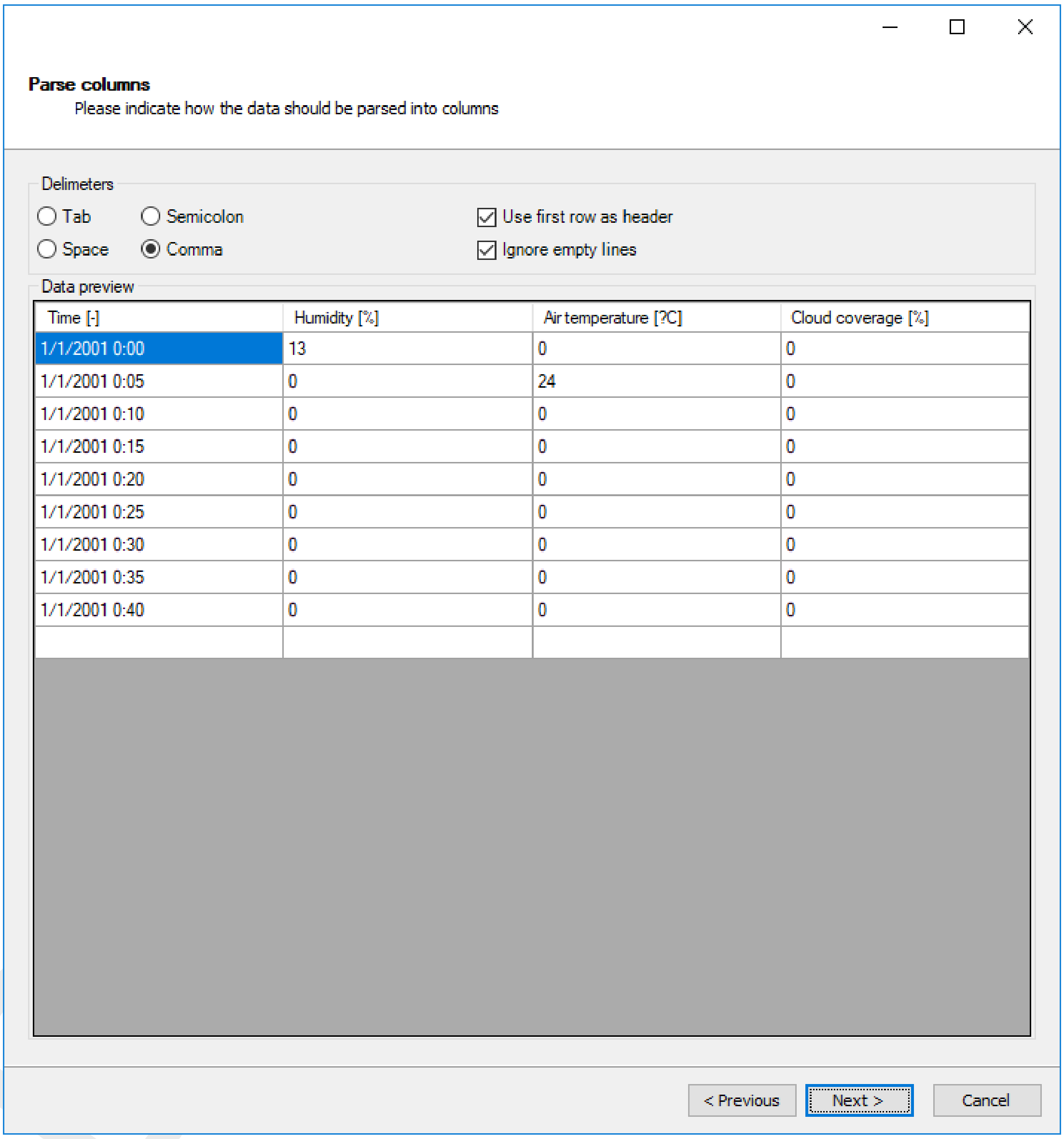
Search DELFT3DFM-296

Name	Date modified	Type	Size
demo.csv	1/8/2019 4:58 PM	CSV File	7 KB

File name: demo.csv

CSV files (*.csv)

Figure 5.53: *Csv import wizard: csv file selection*



Parse columns

Please indicate how the data should be parsed into columns

Delimeters

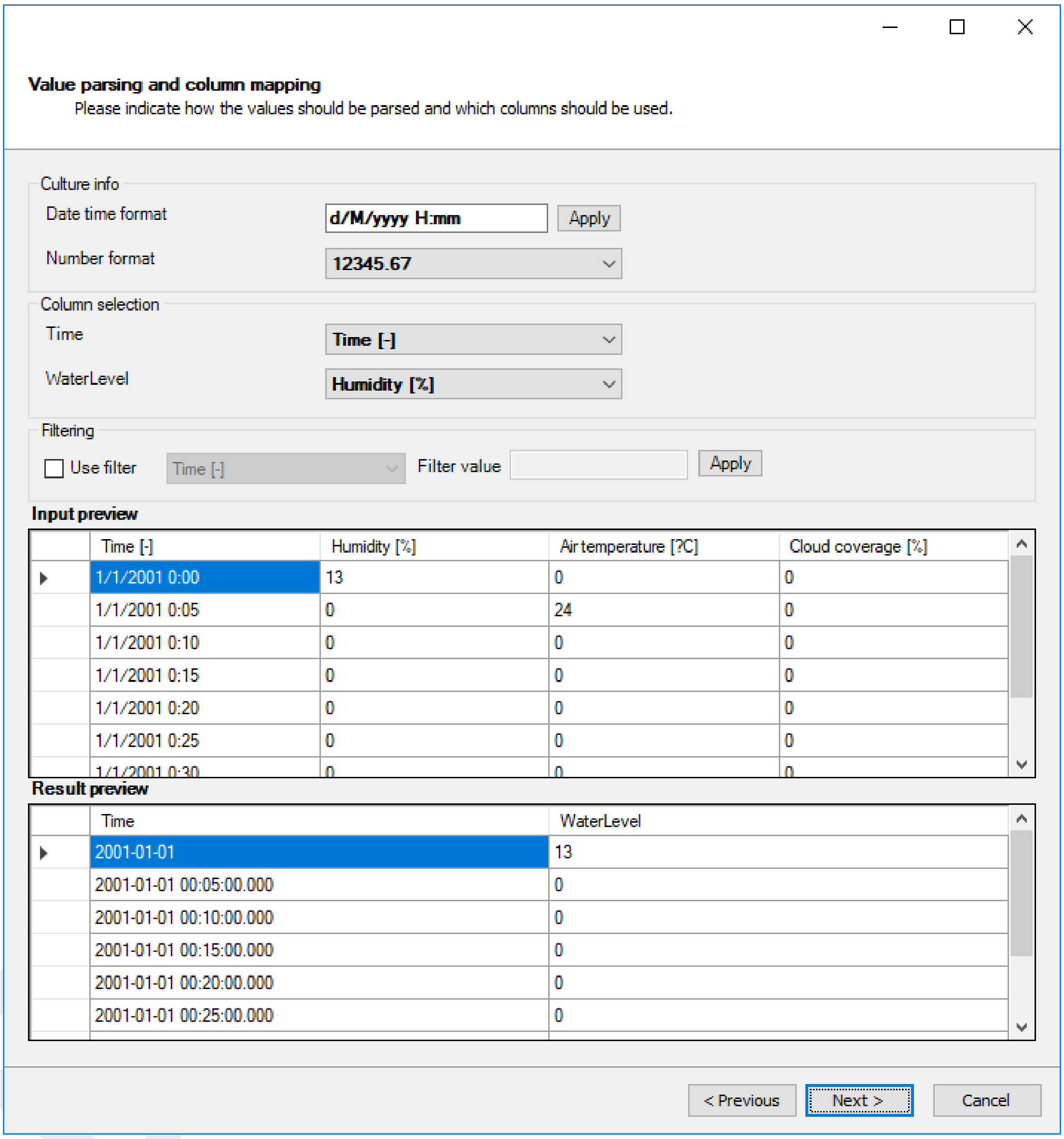
○ Tab ○ Semicolon ☑ Use first row as header

○ Space ● Comma ☑ Ignore empty lines

Data preview

Time [-]	Humidity [%]	Air temperature [?C]	Cloud coverage [%]
1/1/2001 0:00	13	0	0
1/1/2001 0:05	0	24	0
1/1/2001 0:10	0	0	0
1/1/2001 0:15	0	0	0
1/1/2001 0:20	0	0	0
1/1/2001 0:25	0	0	0
1/1/2001 0:30	0	0	0
1/1/2001 0:35	0	0	0
1/1/2001 0:40	0	0	0

Figure 5.54: *Clipboard/csv import wizard: specification of how data should be parsed into columns*



Value parsing and column mapping

Please indicate how the values should be parsed and which columns should be used.

Culture info

Date time format: d/M/yyyy H:mm [Apply]

Number format: 12345.67

Column selection

Time: Time [-]

WaterLevel: Humidity [%]

Filtering

□ Use filter: Time [-] Filter value: [Apply]

Input preview

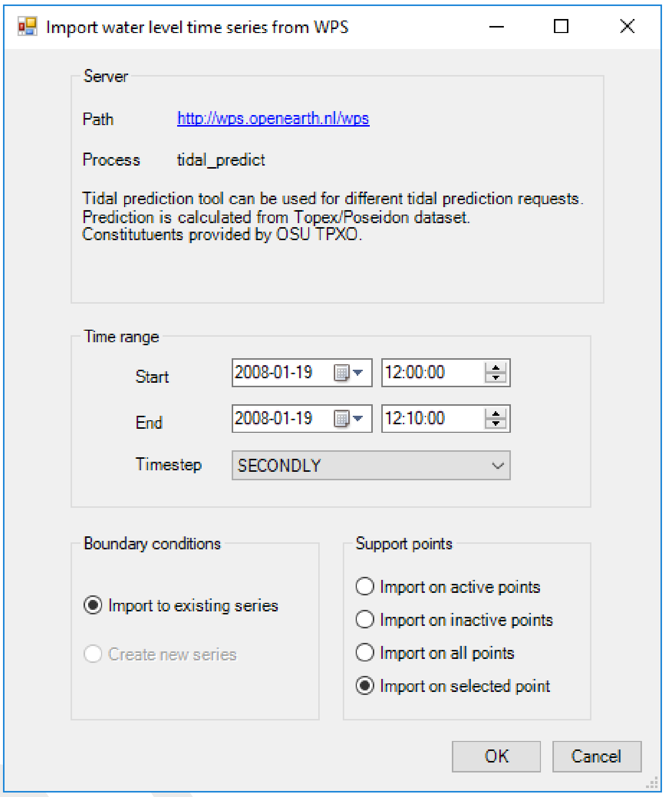
	Time [-]	Humidity [%]	Air temperature [?C]	Cloud coverage [%]
▸	1/1/2001 0:00	13	0	0
	1/1/2001 0:05	0	24	0
	1/1/2001 0:10	0	0	0
	1/1/2001 0:15	0	0	0
	1/1/2001 0:20	0	0	0
	1/1/2001 0:25	0	0	0
	1/1/2001 0:30	0	0	0

Result preview

	Time	WaterLevel
▸	2001-01-01	13
	2001-01-01 00:05:00.000	0
	2001-01-01 00:10:00.000	0
	2001-01-01 00:15:00.000	0
	2001-01-01 00:20:00.000	0
	2001-01-01 00:25:00.000	0

Figure 5.55: *Clipboard/csv import wizard: specification of how values should be parsed and columns should be mapped*

- ◇ Import from clipboard using the 'Clipboard import' button: a wizard will open in which the user can specify (1) how data should be parsed into columns (Figure 5.54) and (2) how the values should be parsed and mapped into columns (Figure 5.55).
- ◇ Import from Web Processing Service (WPS): with this service the user can download boundary forcing data (for now only water level time series) for a selected support point from an online database (TOPEX/Poseidon 7.2). Upon pressing the button 'Import from WPS ' a window will pop up as depicted in Figure 5.56. Here, the user can specify the time interval and time step for downloading the data. (**Note: Please note that this service is only available with an internet connection!**)



Import water level time series from WPS

Server

Path http://wps.openearth.nl/wps

Process tidal_predict

Tidal prediction tool can be used for different tidal prediction requests.
Prediction is calculated from Topex/Poseidon dataset.
Constitutuents provided by OSU TPXO.

Time range

Start 2008-01-19 12:00:00

End 2008-01-19 12:10:00

Timestep SECONDLY

Boundary conditions

● Import to existing series

○ Create new series

Support points

○ Import on active points

○ Import on inactive points

○ Import on all points

● Import on selected point

Figure 5.56: *Window for entering input to download boundary data from WPS*

- Import from attribute file <∗.bc>: with this button the user can import data from existing boundary conditions <∗.bc> file. The user has three options for importing:
  - Overwrite where matching (replace): only overwrites the forcing data for matching support points in the <∗.bc>-file and GUI input.
  - Overwrite where missing (extend): only overwrites the forcing data for matching support points in the <∗.bc>-file and in the GUI input that did not contain data.
  - Overwrite all: overwrites the forcing data for all support points in the GUI (meaning that the forcing data for non-matching support points is emptied).

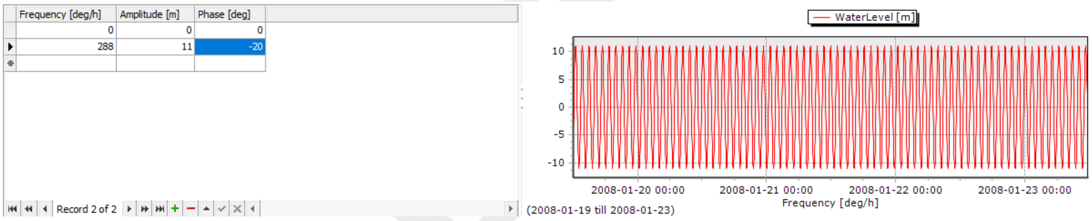
Harmonic components

The harmonic components are defined by a frequency, amplitude and phase (see Figure 5.57). By default the forcing data viewer shows the harmonic component for the time frame specified for the model simulation. The options to define the harmonic components are similar to the options for time series:

- Specify components step by step: the user can add or delete rows with the “plus”- and “minus”-signs below the table.
- Select (astronomical) components using the ‘Select components’ button (Figure 5.58): the user can select astronomical components which will be transformed in the corresponding frequencies.
- Import from csv using the ‘Csv import’ button: a wizard will open in which the user can (1) select a csv-file, (2) specify how data should be parsed into columns and (3) how the values should be parsed and mapped into columns.
- Import from clipboard using the ‘Clipboard import’ button: a wizard will open in which the user can specify (1) how data should be parsed into columns and (2) how the values should be parsed and mapped into columns.
- Import from attribute file <∗.bc>: with this button the user can import data from existing

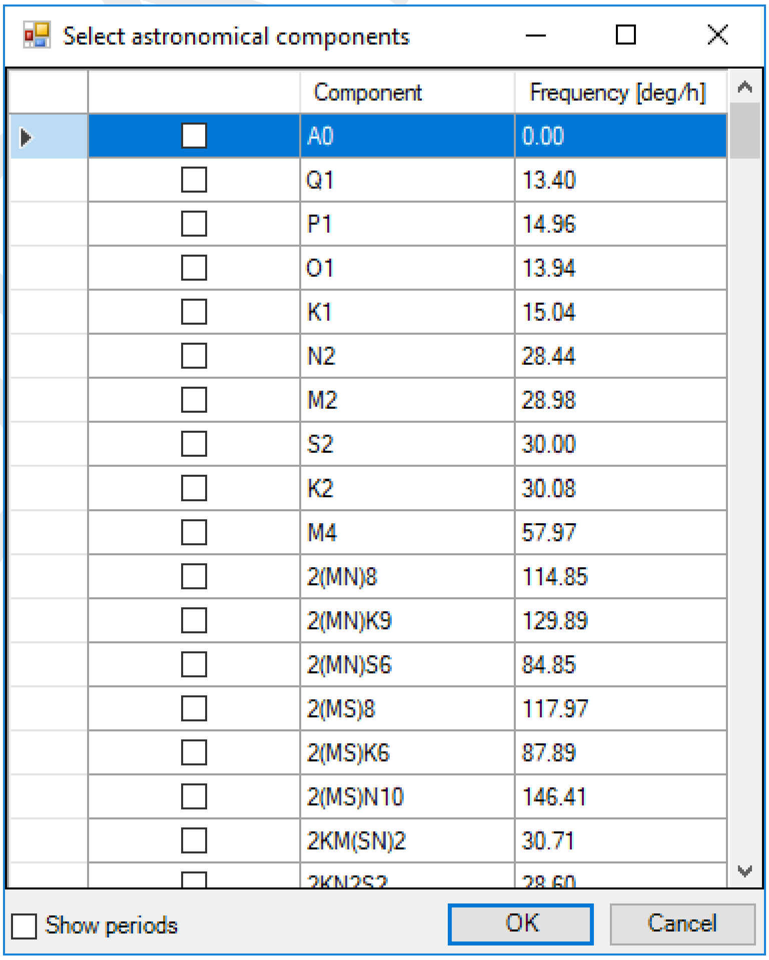
boundary conditions <∗.bc> file. The user has three options for importing:

- Overwrite where matching (replace): only overwrites the forcing data for matching support points in the bc-file and GUI input.
- Overwrite where missing (extend): only overwrites the forcing data for matching support points in the bc-file and in the GUI input that did not contain data.
- Overwrite all: overwrites the forcing data for all support points in the GUI (meaning that the forcing data for non-matching support points is emptied).



	Frequency [deg/h]	Amplitude [m]	Phase [deg]
	0	0	0
▶	288	11	-20
✻			

Figure 5.57: *Specification of harmonic components in boundary data editor*



		Component	Frequency [deg/h]
▶	□	A0	0.00
	□	Q1	13.40
	□	P1	14.96
	□	O1	13.94
	□	K1	15.04
	□	N2	28.44
	□	M2	28.98
	□	S2	30.00
	□	K2	30.08
	□	M4	57.97
	□	2(MN)8	114.85
	□	2(MN)K9	129.89
	□	2(MN)S6	84.85
	□	2(MS)8	117.97
	□	2(MS)K6	87.89
	□	2(MS)N10	146.41
	□	2KM(SN)2	30.71
	□	2KN2S2	28.60

□ Show periods

Figure 5.58: *Selection of astronomical components from list (after pressing 'select components')*

Astronomic components

Astronomical components are similar to harmonic components, with the exception that the frequency is prescribed. Instead of specifying the frequency the user can select astronomical components by name. Upon editing the component field in the table the user will get suggestions for components in a list (Figure 5.59). The most frequently used components (A0, Q1, P1, O1, K1, N2, M2, S2, K2 and M4) are put on top of the list, the other components are listed in alphabetic order. Instead of defining each component individually, the user can also make a selection of components by pressing the button 'select components' (Figure 5.58).

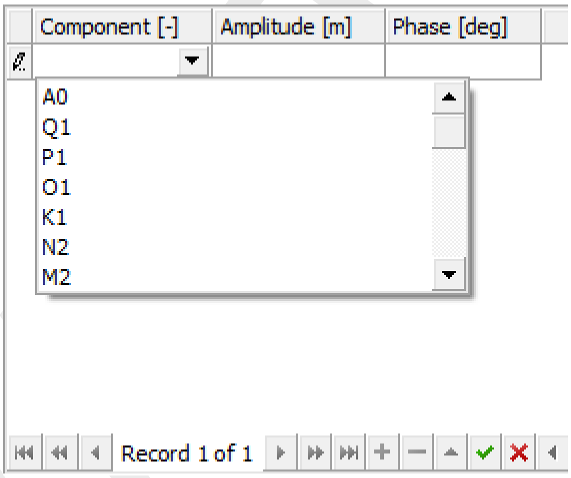
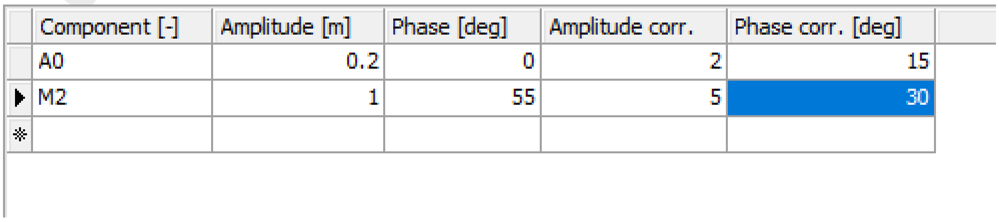


Figure 5.59: *Suggestions for astronomical components in list*

Harmonic or astronomic components with corrections

For calibration purposes the user can combine harmonic or astronomic components with corrections. The corrections are defined in terms of an amplitude (multiplication) factor and a phase difference (see Figure 5.60). This allows the user to keep track of both the original signal and the calibration coefficients. The effects of the corrections on the resulting signal are directly visualized in the forcing data viewer. **Note: Please note that the import functionality is not (yet) working properly for astronomic/harmonic boundary conditions with corrections.**

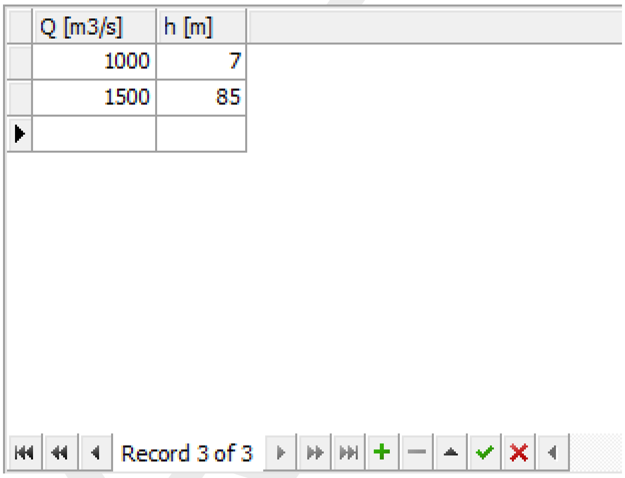


Component [-]	Amplitude [m]	Phase [deg]	Amplitude corr.	Phase corr. [deg]
A0	0.2	0	2	15
M2	1	55	5	30

Figure 5.60: *Editing harmonic/astronomic components and their corrections*

Q-h relation (only for water level)

The user can force the boundary with a Q-h relationship, but only for the quantity water level. This is a relationship between discharge and water level (see Figure 5.61). The relationship can only be prescribed per polyline, not per support point. (**Note: this functionality has not been tested extensively yet**)



Q [m3/s]	h [m]
1000	7
1500	85

Figure 5.61: *Specification of a Q-h relationship*

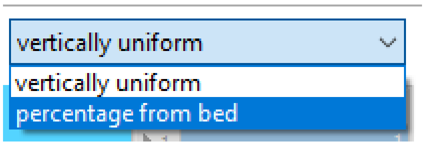
Exporting boundary conditions

With the *Export to files* button the user can export all boundary forcing data for the given polyline to a <∗.bc>-file.

5.4.8.2.1 3D boundary conditions

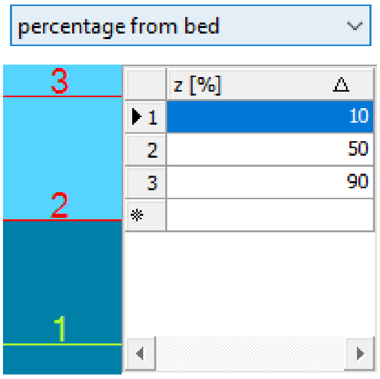
When the model is 3D (i.e. the number of layers is larger than 1), the user can specify three-dimensional boundary conditions for relevant quantities such as velocity, salinity, temperature and tracer concentration. The user can choose from vertically uniform or vertically varying boundary conditions (Figure 5.62), where the latter are defined as a percentage from the bed. In the boundary condition editor the layer view will appear (Figure 5.63). Here, the user can specify the vertical positions of the boundary conditions and view their position relative to the model layers. Please note that the number and position of vertical boundary conditions does not necessarily have to match the number and/or the exact position of the model layers. The computational core will interpolate the boundary forcing position to the number of model layers.

In case of non-uniform boundary conditions over the vertical, the number of columns in the boundary forcing data editor will increase correspondingly (see Figure 5.64). In this way the user can specify the conditions for all vertical positions in the same table and view the resulting signals in the forcing data viewer.



vertically uniform

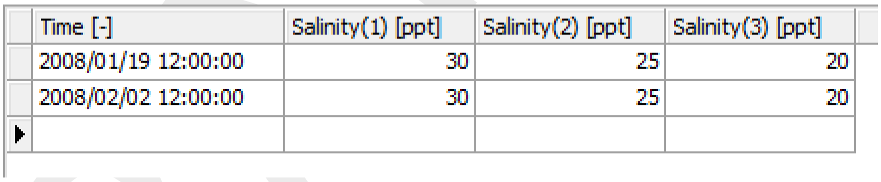
Figure 5.62: *Selection of vertically uniform or varying boundary conditions in case of a 3D model*



percentage from bed

	z [%]
▶ 1	10
2	50
3	90
✱	

Figure 5.63: *Overview of the layer view component of the boundary conditions editor. In the table the user can edit the vertical positions of the boundary conditions as a percentage from the bed. In the view left of the table, the user can see the vertical positions of the boundary conditions (indicated by number corresponding to the table) relative to the model layers.*

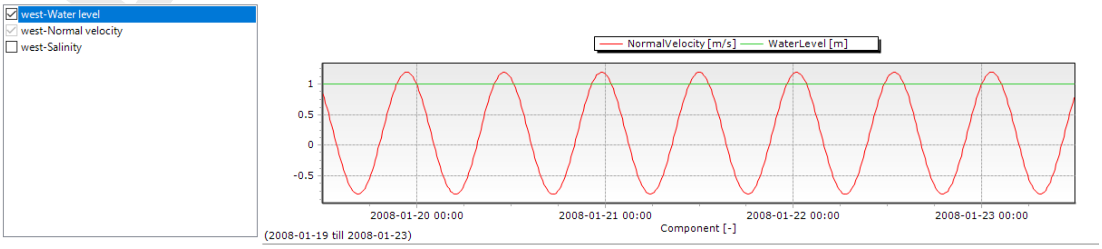


Time [-]	Salinity(1) [ppt]	Salinity(2) [ppt]	Salinity(3) [ppt]
2008/01/19 12:00:00	30	25	20
2008/02/02 12:00:00	30	25	20

Figure 5.64: *Specification of boundary forcing data (in this example for salinity) at 3 positions in the vertical*

View boundary data

All boundary data of the same quantity on a support point can be (pre-)viewed in the boundary data view in the lower-right panel. If multiple signals of the same process have been entered, the viewer will show the active signal in red and the total signal of all datasets in green (see Figure 5.65). In the view the user can zoom-in by dragging a box from top-left to bottom-right and zoom-out by dragging a box from bottom-right to top-left.



☑ west-Water level
☑ west-Normal velocity
□ west-Salinity

Figure 5.65: *Example of active and total signal for multiple water level data series on one support point*

To inspect multiple quantities at a support point at the same time (for example water level and normal velocity) the user can use the combined boundary data viewer by pressing the button 'Combined BC view'. **Note: This does not work properly yet.**

5.4.8.3 Import/export boundary conditions from the *Project* window

Apart from import and export functionality per individual boundary polyline in the boundary condition editor, the GUI offers the opportunity to import and export boundary locations (<∗.pli>) and forcing (<∗.bc>) on a higher level. Hereto, you have to click the right mouse button on "Boundary Conditions" in the *Project* window and select "Import" or "Export" (Figure 5.66). Imports and exports on "Boundary Conditions" apply to all the boundary conditions whereas import and exports on a boundary polyline apply only to that boundary condition.

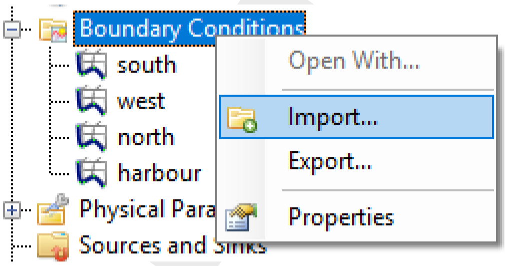
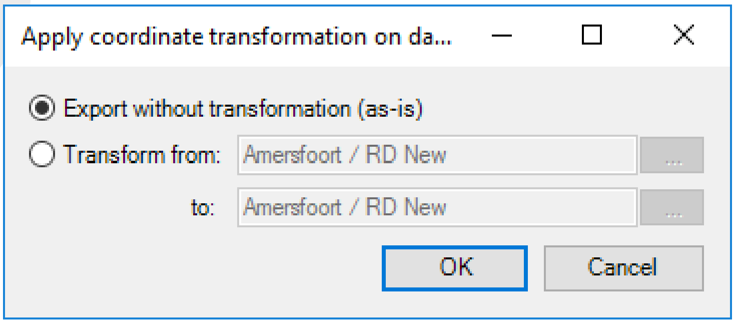


Figure 5.66: *Importing or exporting boundary features — both polylines <∗.pli> and forcing <∗.bc> — from the* Project *window using the right mouse button*

Import and export polylines

Upon importing a <∗.pli>-file with the same filename and the same polyline name(s) as the existing polyline names in the GUI, the existing polyline(s) will be replaced and all forcing data thereon will be deleted. Upon importing a polyline(s) with a different name(s), the polyline(s) will be added to the *Project* window without any forcing data on it/them. The user will be asked to import the data "as is" or to perform a coordinate transformation before the import (see Figure 5.67).

Alternatively, the user can exported created polylines to a <∗.pli>-file. Upon export the user will be asked to export the data "as is" or to perform a coordinate transformation before the export (see Figure 5.67).



● Export without transformation (as-is)

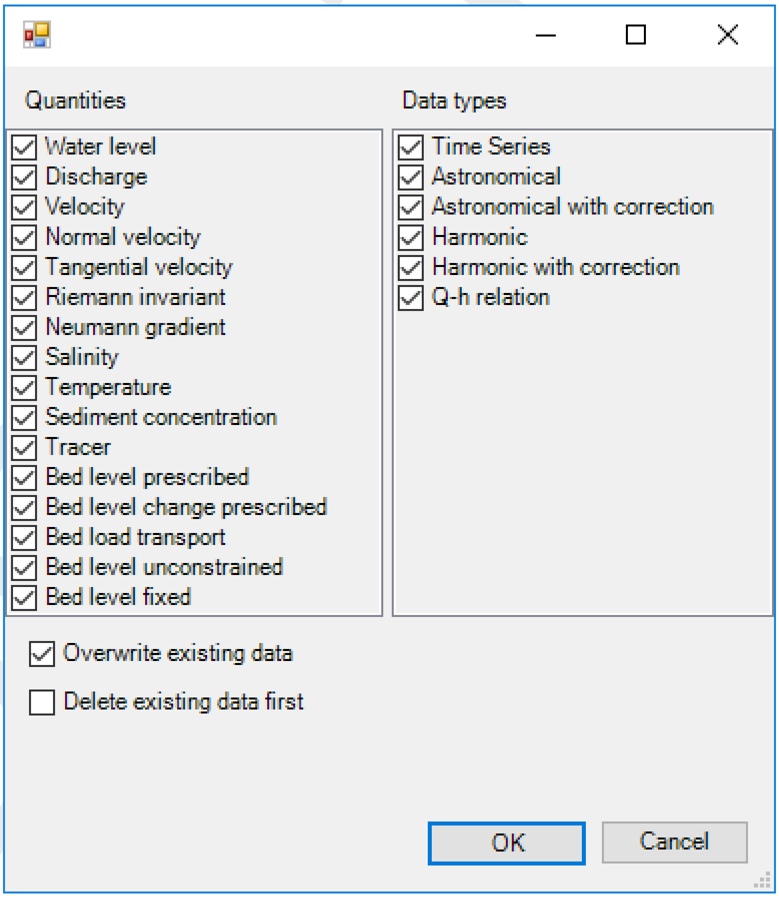
○ Transform from: Amersfoort / RD New

to: Amersfoort / RD New

Figure 5.67: *Import or export a <∗.pli>-file as is or with coordinate transformation.*

Import and export boundary forcing data

Similar to the polylines, you can import and export boundary forcing data from/to a <∗.bc>-file. To import forcing data the existence of a polyline with at least one matching support point is a prerequisite. Upon importing <∗.bc> data you can select which quantities and forcing types from the <∗.bc>-file should be imported and with which overrwrite options (see Figure 5.68). Similarly, you can export boundary forcing data. As an additional exporting feature you can select whether you would like to export: 1) all forcing data into one file, 2) as separate files per boundary, 3) as separate files per process or 4) as separate files per quantity (see Figure 5.69).



Quantities

☑ Water level
☑ Discharge
☑ Velocity
☑ Normal velocity
☑ Tangential velocity
☑ Riemann invariant
☑ Neumann gradient
☑ Salinity
☑ Temperature
☑ Sediment concentration
☑ Tracer
☑ Bed level prescribed
☑ Bed level change prescribed
☑ Bed load transport
☑ Bed level unconstrained
☑ Bed level fixed

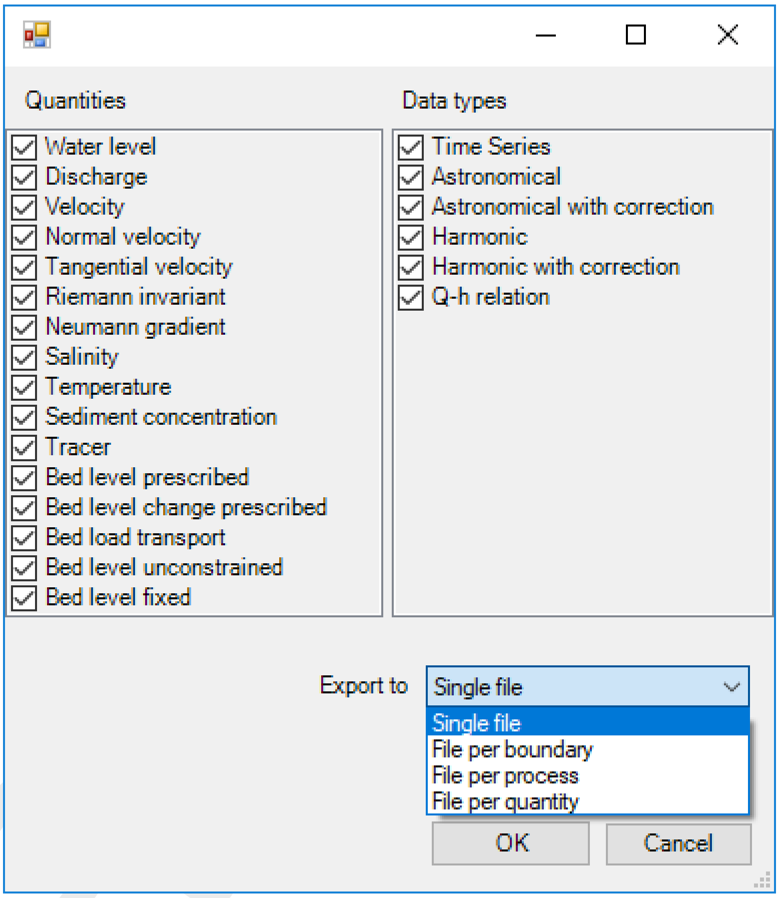
Data types

☑ Time Series
☑ Astronomical
☑ Astronomical with correction
☑ Harmonic
☑ Harmonic with correction
☑ Q-h relation

☑ Overwrite existing data

☐ Delete existing data first

Figure 5.68: *Import or export a <∗.pli>-file as is or with coordinate transformation.*



Quantities

☑ Water level
☑ Discharge
☑ Velocity
☑ Normal velocity
☑ Tangential velocity
☑ Riemann invariant
☑ Neumann gradient
☑ Salinity
☑ Temperature
☑ Sediment concentration
☑ Tracer
☑ Bed level prescribed
☑ Bed level change prescribed
☑ Bed load transport
☑ Bed level unconstrained
☑ Bed level fixed

Data types

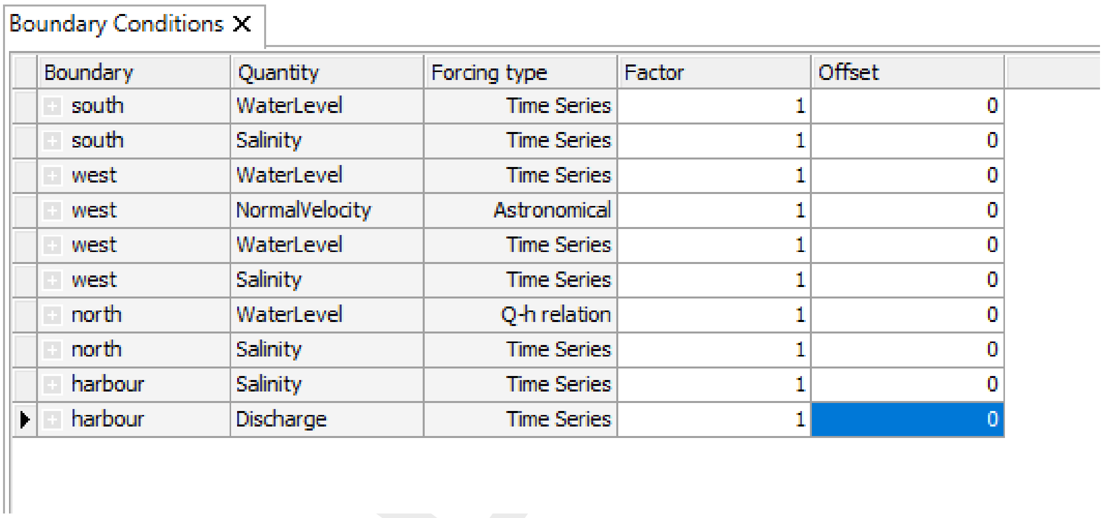
☑ Time Series
☑ Astronomical
☑ Astronomical with correction
☑ Harmonic
☑ Harmonic with correction
☑ Q-h relation

Export to: Single file

*Figure 5.69: Import or export a *.pli file as is or with coordinate transformation.*

5.4.8.4 Overview of boundary conditions in attribute table (non-editable)

The attribute table of the boundary conditions gives an overview of all specified boundary polylines and corresponding forcing (see Figure 5.70). This attribute table can be opened by double clicking ‘Boundary Conditions’ from the project or *Map* window. Most of the features in the attribute table are non-editable, except for the optional (multiplication) factor and offset per quantity per polyline. With these settings you can integrally multiply all data on a polyline with a factor and/or add an offset to it.



Boundary Conditions

Boundary	Quantity	Forcing type	Factor	Offset
south	WaterLevel	Time Series	1	0
south	Salinity	Time Series	1	0
west	WaterLevel	Time Series	1	0
west	NormalVelocity	Astronomical	1	0
west	WaterLevel	Time Series	1	0
west	Salinity	Time Series	1	0
north	WaterLevel	Q-h relation	1	0
north	Salinity	Time Series	1	0
harbour	Salinity	Time Series	1	0
harbour	Discharge	Time Series	1	0

***Figure 5.70:** Overview of all boundary conditions in attribute table*

5.4.9 Physical parameters

The physical parameters attribute is used to set all physical parameters of your model. When it is expanded in the *Project* window, it shows four attributes; roughness, viscosity, diffusivity and wind.

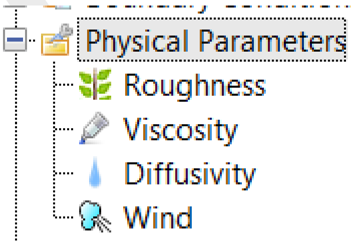


***Figure 5.71:** The physical parameters in the* Project *window*

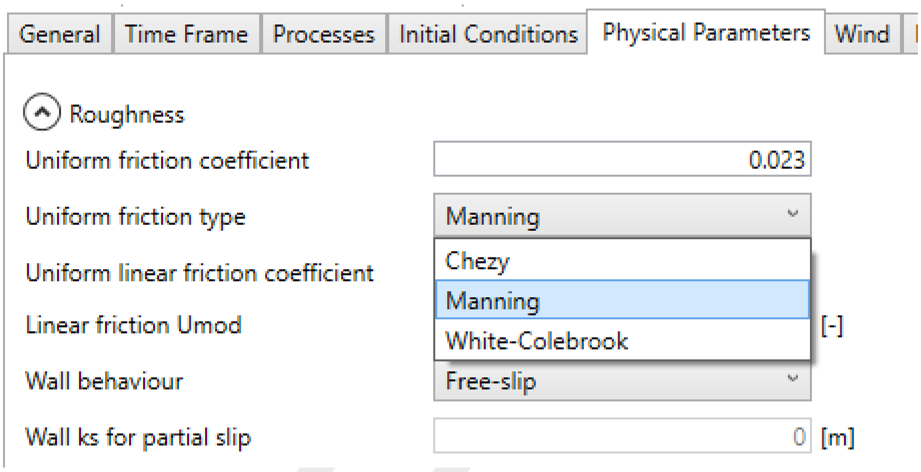
5.4.9.1 Constants

In the *Physical Parameters* tab, the user can adjust the value of *Gravity* [m/s^2] under *Others* and *Default water density* [kg/m^3] under *Density*.

5.4.9.2 Roughness

Bed roughness can be specified as a uniform value or as a coverage (e.g. a spatially varying field). The uniform values as well as the roughness formulation (i.e. Chézy, Manning, White-ColeBrook or Z_0) can be edited in the ‘Physical Parameters’ tab, which opens upon double clicking ‘Roughness’ in the *Project* window (Figure 5.72). **Note:** The latter is not yet working. In this tab you can also specify the linear friction coefficient, linear friction Umod, wall behaviour (free slip or partial slip) and wall ks for partial slip.

In case of spatially varying bed roughness you can double click the quantity in the *Project* window or select it from the dropdown box in the *Spatial Operations* ribbon (Figure 5.73). Then the spatial editor is activated, which you can use to edit spatially varying fields. For more information on how to use the spatial editor you are referred to Appendix H.



Roughness

Uniform friction coefficient: 0.023

Uniform friction type: Manning

Uniform linear friction coefficient

Linear friction Umod [-]

Wall behaviour: Free-slip

Wall ks for partial slip: 0 [m]

Figure 5.72: *The section of the 'Physical Parameters' tab where you can specify roughness related parameters and formulations.*

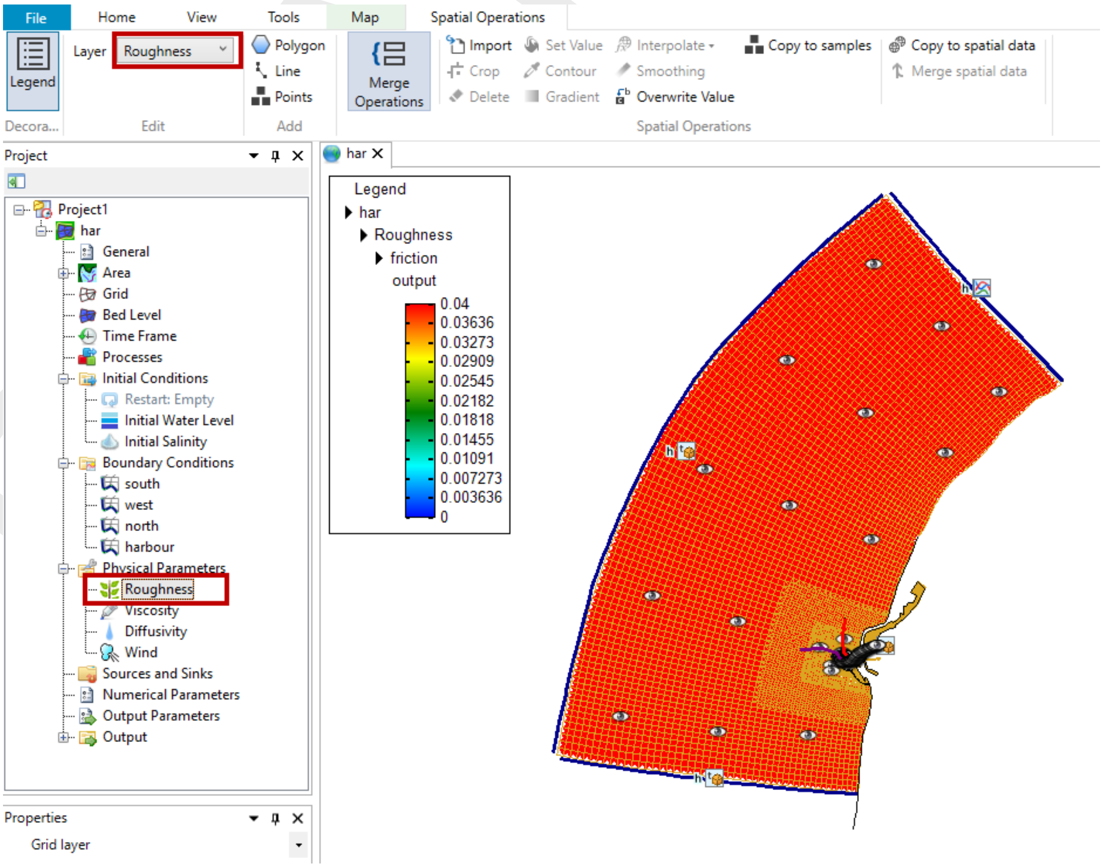
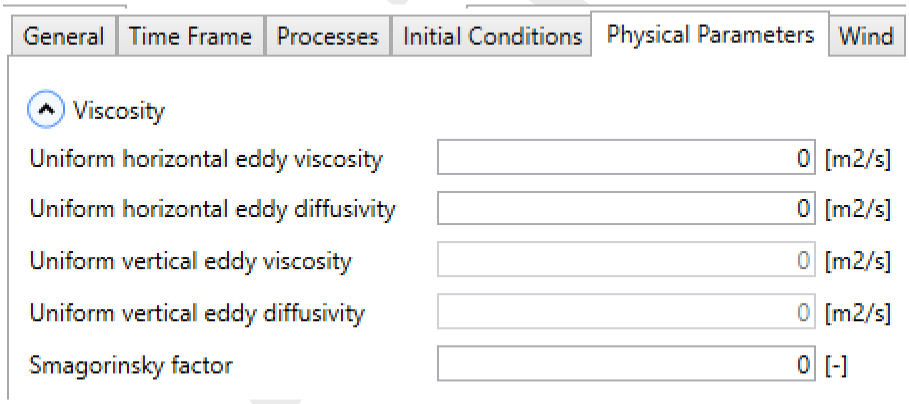


Figure 5.73: *Roughness activated in the spatial editor to create/edit a spatially varying field*

5.4.9.3 Viscosity

The eddy viscosity can be specified as a uniform value or as a coverage (e.g. a spatially varying field). In the 'Physical parameters' tab you can specify the uniform values for the horizontal and vertical (in case of 3D simulations) eddy viscosity and diffusivity (Figure 5.74).

In case of spatially varying viscosity you can double click the quantity in the *Project* window or select it from the dropdown box in the *Spatial Operations* ribbon (Figure 5.75). Then the spatial editor is activated, which you can use to edit spatially varying fields. For more information on how to use the spatial editor you are referred to Appendix H.



Viscosity

Uniform horizontal eddy viscosity	0	[m2/s]
Uniform horizontal eddy diffusivity	0	[m2/s]
Uniform vertical eddy viscosity	0	[m2/s]
Uniform vertical eddy diffusivity	0	[m2/s]
Smagorinsky factor	0	[-]

Figure 5.74: *The section of the'Physical Parameters' tab where you can specify (uniform) values for the horizontal and vertical eddy viscosity and diffusivity.*

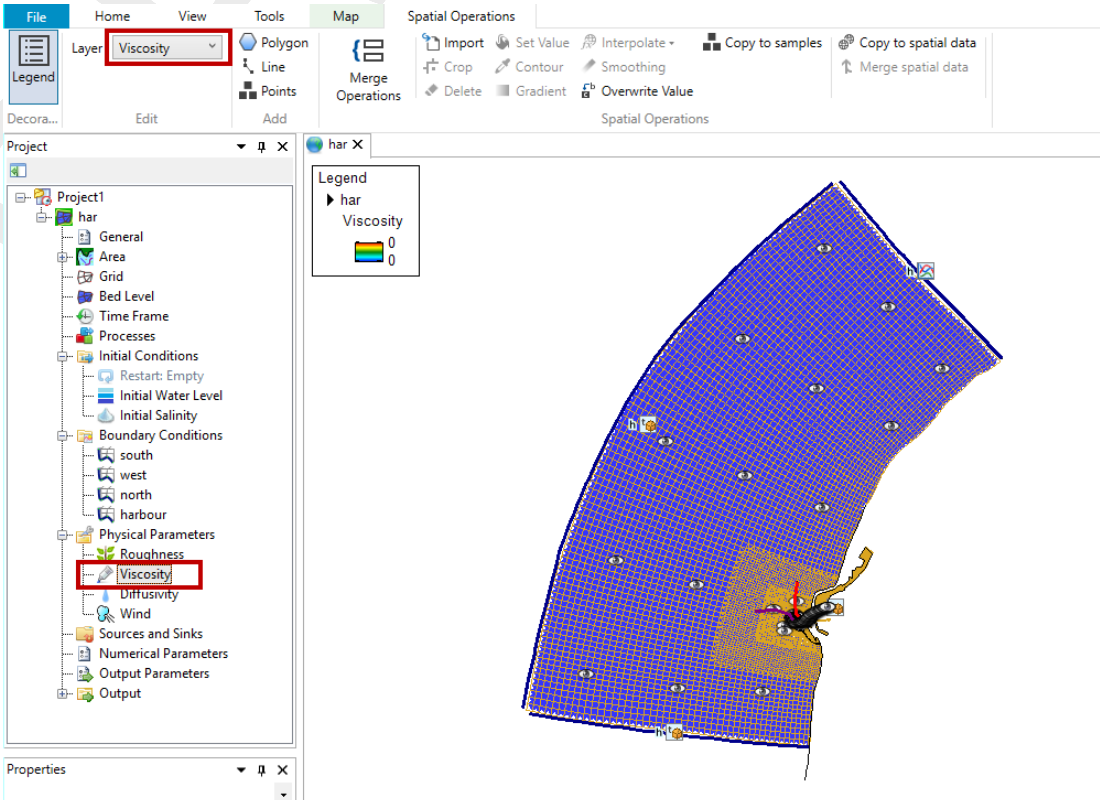
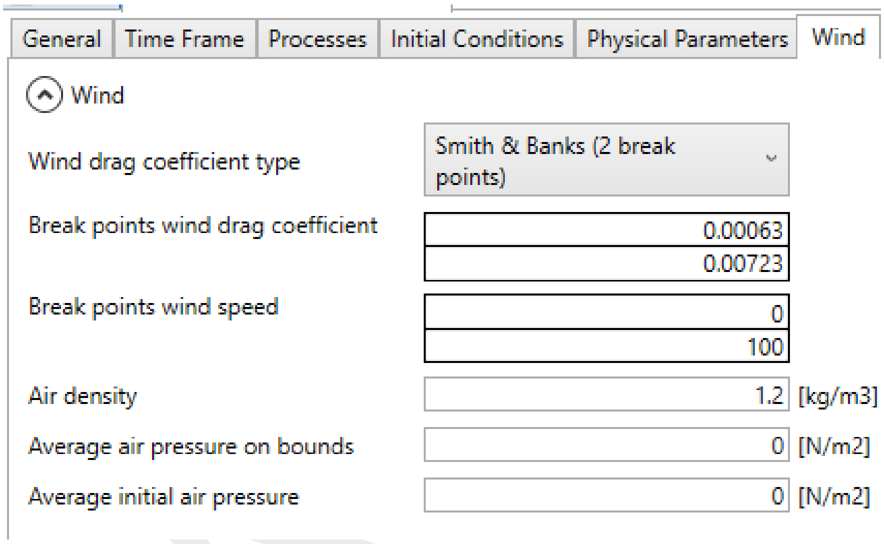


Figure 5.75: *Viscosity activated in the spatial editor to create/edit a spatially varying field*

5.4.9.4 Wind

All relevant parameters, described in chapter 11 can be adjusted in the following sub-tab.



Wind

Wind drag coefficient type: Smith & Banks (2 break points)

Break points wind drag coefficient: 0.00063, 0.00723

Break points wind speed: 0, 100

Air density: 1.2 [kg/m3]

Average air pressure on bounds: 0 [N/m2]

Average initial air pressure: 0 [N/m2]

***Figure 5.76:** Overview of parameters in sub-tab Wind*

5.4.9.5 Heat Flux model

The Heat flux model, described in chapter 10, needs only specification of which model to use. See the drop-down selection in Figure 5.36.

5.4.9.6 Tidal forces

The tide generating forces, described in section 8.12, can be enabled in the *Processes* tab, see Figure 5.36.

5.4.10 Sources and sinks

Sources and sinks (or: intake/outfall facilities) can be used to add/extract a discharge to/from the model or to redistribute water and constituents (such as temperature and salinity) within the model. Sources and sinks consists of a location (defined by a <∗.pli>-file) and time series describing the discharges (defined by a <∗.tim>-file). All the hydrodynamical considerations behind sources and sinks are discussed in section 8.10.

Sources and sinks locations

Sources and sinks can be added to the model using the corresponding icon from the *Map* ribbon (Figure 5.77). When the sources/sinks icon is active you can add them as polyline elements in the central map using the left mouse button. Each polyline element is closed by double clicking the left mouse button.

Note: Please note that the length of the polyline elements is not taken into account in the handling of sources and sinks (e.g. it is modeled as an instant redistribution of water and constituents without delays and friction losses).

Polyline elements starting outside the model domain and going inward are sources and, vice versa, polyline elements starting inside the model domain and going outward are sinks. Polyline elements starting and ending within the model are intake/outfall type of discharges.

The drawing direction determines the direction of the discharge indicated by an arrow (Figure 5.78). In case of a source the direction of the last polyline element determines the direction of flow momentum into the model.

Note: The ‘reverse’ line option — to switch the discharge direction without having to redraw the source/sink — is not implemented.

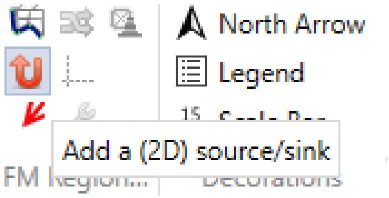


Figure 5.77: *Activate the sources and sinks editing icon in the* Map *ribbon*

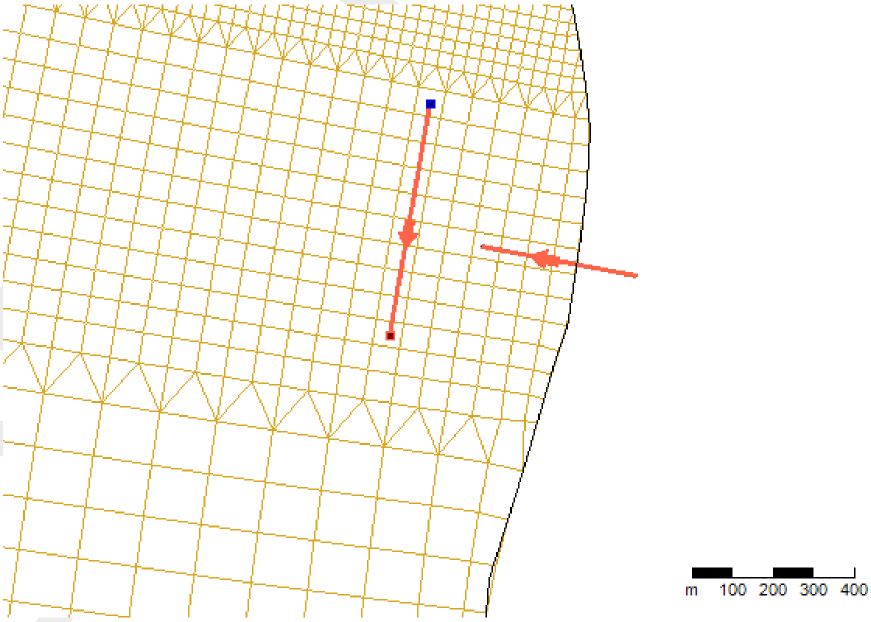


Figure 5.78: *Add sources and sinks in the central map using the ‘Sources and sinks’ icon.*

Sources and sinks time series

After drawing the sources/sinks locations in the central map, they appear under ‘Sources and sinks’ in the *Project* window (Figure 5.79). Either by double clicking the sources/sinks in the *Project* window or the line element in the central map, you can open the sources and sinks editor (Figure 5.80). In this editor you can specify the discharge time series as well as the corresponding constituents (for example salinity and temperature, depending on the active physical processes). The time series of water discharges are always defined as absolute values. For the time series of constituents the following applies:

- For sources the constituent time series are absolute values
- For sinks the constituent time series are determined by the modeled values
- For sources and sinks (intake/outfall relationships) the constituent time series are the excess values (e.g. on top of the modeled values)

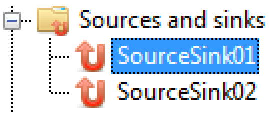
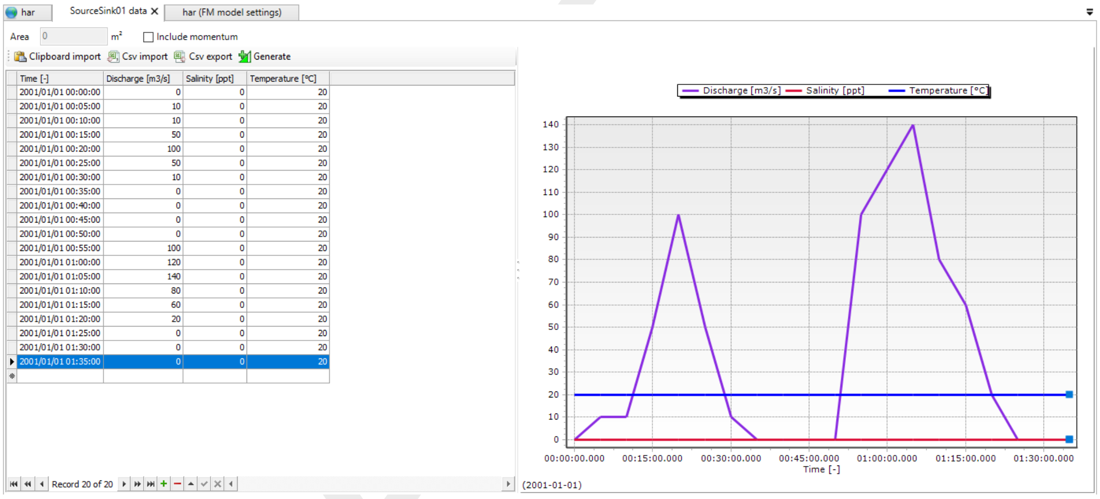


Figure 5.79: *Sources and sinks appearing in the* Project *window*



Area 0 m² □ Include momentum

Time [-]	Discharge [m3/s]	Salinity [ppt]	Temperature [°C]
2001/01/01 00:00:00	0	0	20
2001/01/01 00:05:00	10	0	20
2001/01/01 00:10:00	10	0	20
2001/01/01 00:15:00	50	0	20
2001/01/01 00:20:00	100	0	20
2001/01/01 00:25:00	50	0	20
2001/01/01 00:30:00	10	0	20
2001/01/01 00:35:00	0	0	20
2001/01/01 00:40:00	0	0	20
2001/01/01 00:45:00	0	0	20
2001/01/01 00:50:00	0	0	20
2001/01/01 00:55:00	100	0	20
2001/01/01 01:00:00	120	0	20
2001/01/01 01:05:00	140	0	20
2001/01/01 01:10:00	80	0	20
2001/01/01 01:15:00	60	0	20
2001/01/01 01:20:00	20	0	20
2001/01/01 01:25:00	0	0	20
2001/01/01 01:30:00	0	0	20
2001/01/01 01:35:00	0	0	20

Record 20 of 20

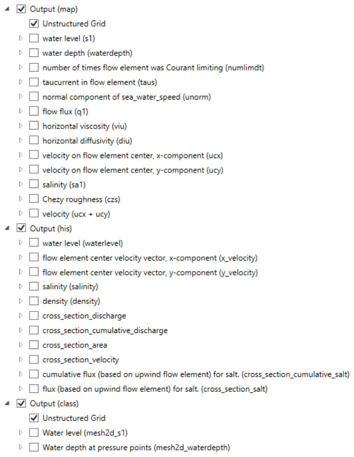
Figure 5.80: *Specifying time series for sources and sinks in the sources and sinks editor*

5.4.11 Numerical parameters

In the numerical parameters tab, all numerical parameters related to your computation can be set. The parameters that can be set are described in Table 5.1.

5.4.12 Output parameters

Model runs can produce various types of output files. The **Map** window shows the two most-used types: *(map* and *his(tory)* output (Figure 5.81).



- ☑ Output (map)
  - ☑ Unstructured Grid
  - □ water level (s1)
  - □ water depth (waterdepth)
  - □ number of times flow element was Courant limiting (numlimdt)
  - □ taucurrent in flow element (taus)
  - □ normal component of sea_water_speed (unorm)
  - □ flow flux (q1)
  - □ horizontal viscosity (viu)
  - □ horizontal diffusivity (diu)
  - □ velocity on flow element center, x-component (ucx)
  - □ velocity on flow element center, y-component (ucy)
  - □ salinity (sa1)
  - □ Chezy roughness (czs)
  - □ velocity (ucx + ucy)
- ☑ Output (his)
  - □ water level (waterlevel)
  - □ flow element center velocity vector, x-component (x_velocity)
  - □ flow element center velocity vector, y-component (y_velocity)
  - □ salinity (salinity)
  - □ density (density)
  - □ cross_section_discharge
  - □ cross_section_cumulative_discharge
  - □ cross_section_area
  - □ cross_section_velocity
  - □ cumulative flux (based on upwind flow element) for salt. (cross_section_cumulative_salt)
  - □ flux (based on upwind flow element) for salt. (cross_section_salt)
- ☑ Output (class)
  - ☑ Unstructured Grid
  - □ Water level (mesh2d_s1)
  - □ Water depth at pressure points (mesh2d_waterdepth)

Figure 5.81: Output Parameters *tab*

The history file contains output on specific locations: time series data on observation points (section 5.4.2.2), cross-sections (section 5.4.2.3) and structures (Sections 5.4.2.8 – 5.4.2.10). The map file contains flow quantities on the entire grid at specified time intervals, and can later

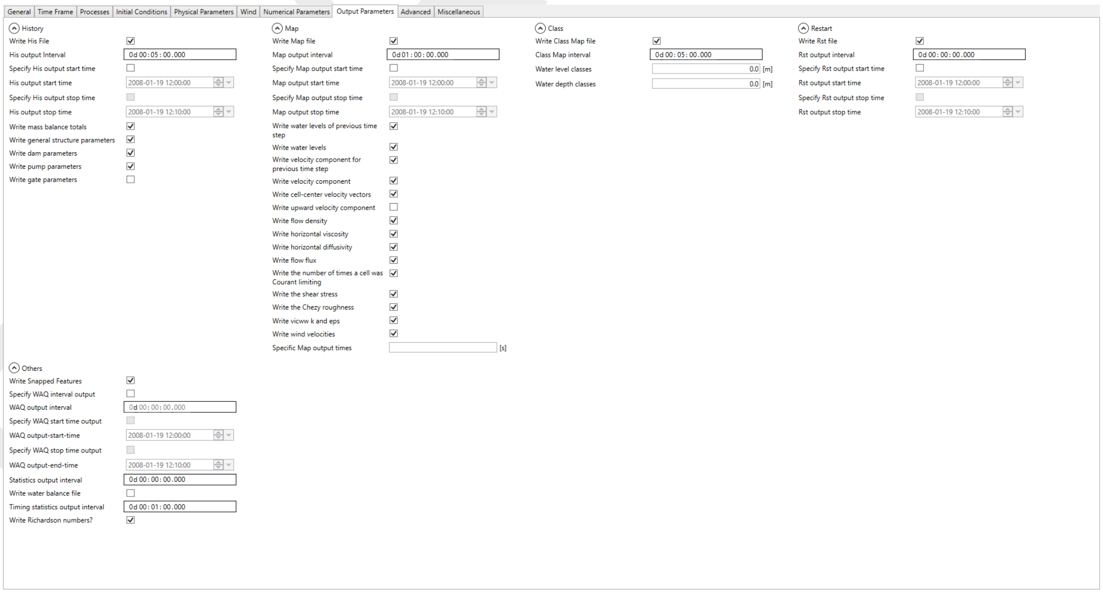
Table 5.1: Overview and description of numerical parameters

Parameter	Description
Max Courant number	The size of the time step, Δt, is computed by the computational kernel automatically, each time step by means of the maximum tolerable Courant number. By default, this value is 0.7.
Wave velocity fraction	The wave velocity fraction is related to stability of the computation. By using this fraction, the velocity of the flow is enlarged with the wave velocity times this fraction. The value of this fraction is 0.1 by default.
Advection type	This key depicts the ID of the advection scheme. By default this value is 33.
Water depth limiter type	The limiter type for waterdepth in continuity equation: 0 means no limiter (default), 1 is the minmod method, 2 the Van Leer method and 4 the monotone central method.
Advection velocity limiter type	The limiter type for the cell center advection velocity: 0 means no limiter, 1 the minmod method, 2 the Van Leer method and 4 the monotone central method (default).
Salinity transport limiter type	The limiter type for salinity transport: 0 means no limiter, 1 the minmod method, 2 the Van Leer method and 4 the monotone central method (default).
Solver type	Specification of the Poisson equation type solver for the pressure: 1 = sobekGS_OMP, 2 = sobekGS_OMPthreadsafe, 3 = sobekGS, 4 = sobekGS + Saadilud (default), 5 = parallel/global Saad, 6 = parallel/Petsc, 7 = parallel/GS.
Max degree in Gaussian elimination	The maximum degree in the Gauss elimination, part of the pressure solver. This key has the value 6 by default.
Fixed weir scheme	Or: fixed weir scheme. 0: none, 1: compact stencil, 2: whole tile lifted, full subgrid weir + factor.
Fixed weir contraction	Or: fixed weir contraction. This is the fixed weir flow width contraction factor, being the flow width = flow width times the fixed weir contraction.
Boundary smoothing time	Fourier smoothing time on waterlevel boundaries (s).
Linear continuity	Default 0; Set to 1 for linearizing d(Hu)/dx; link to AdvecType.
Threshold for drop losses	Apply droplosses only if local bed slope is larger than this specific value.
Theta of time integration	Level of explicitness/implicitness of time integration.
Downwind cell H on Q boundaries	Specifies the way of the upstream discharge boundary: 0 is original hu on qbnd, 1 is downwind hs on qbnd

be used for 2D and 3D visualizations of entire flow fields. The map file can typically turn out much larger than the history file, and it is therefore advised to use larger time intervals for map files than for his files.

Additionally, three more types of output can be requested: restart files, WAQ output, and timing statistics. Restart files are a special type of map file that can later be used as initial states in other runs. Restart files contain several additional flow quantities and are written at specified intervals into one file per each restart time. Typically, one selects a large restart interval in order not to waste disk space. WAQ output files are written by D-Flow FM and are intended to be used as input files to subsequent D-Water Quality runs. More details on water quality modelling can be found in chapter 19. Timing statistics can be produced both in the diagnostics file (via *Statistics output interval*, for viewing basic simulation progress), and in a separate detailed timings file (via *Timing statistics output interval*, for detailed performance analysis).

When double clicking the *Output Parameters* in the *Project* window, the *Output Parameters* tab is highlighted below the central map. All parameters related to the output of your model run are specified here (Figure 5.82). The most common output parameters to set are the parameters related to the water quality files, history files, map files and restart files.



History
Write His File ☑
His output Interval 0d 00:05:00.000
Specify His output start time ☐
His output start time 2008-01-19 12:00:00
Specify His output stop time ☐
His output stop time 2008-01-19 12:10:00
Write mass balance totals ☑
Write general structure parameters ☑
Write dam parameters ☑
Write pump parameters ☑
Write gate parameters ☐

Map
Write Map file ☑
Map output interval 0d 01:00:00.000
Specify Map output start time ☐
Map output start time 2008-01-19 12:00:00
Specify Map output stop time ☐
Map output stop time 2008-01-19 12:10:00
Write water levels of previous time step ☑
Write water levels ☑
Write velocity component for previous time step ☑
Write velocity component ☑
Write cell-center velocity vectors ☑
Write upward velocity component ☐
Write flow density ☑
Write horizontal viscosity ☑
Write horizontal diffusivity ☑
Write flow flux ☑
Write the number of times a cell was Courant limiting ☑
Write the shear stress ☑
Write the Chezy roughness ☑
Write vicww k and eps ☑
Write wind velocities ☑
Specific Map output times [s]

Class
Write Class Map file ☑
Class Map interval 0d 00:05:00.000
Water level classes 0.0 [m]
Water depth classes 0.0 [m]

Restart
Write Rst file ☑
Rst output interval 0d 00:00:00.000
Specify Rst output start time ☐
Rst output start time 2008-01-19 12:00:00
Specify Rst output stop time ☐
Rst output stop time 2008-01-19 12:10:00

Others
Write Snapped Features ☑
Specify WAQ interval output ☐
WAQ output interval 0d 00:00:00.000
Specify WAQ start time output ☐
WAQ output-start-time 2008-01-19 12:00:00
Specify WAQ stop time output ☐
WAQ output-end-time 2008-01-19 12:10:00
Statistics output interval 0d 00:00:00.000
Write water balance file ☐
Timing statistics output interval 0d 00:01:00.000
Write Richardson numbers? ☑

Figure 5.82: *Overview* Output Parameters *tab*

For each of the three files (water quality, history, map and restart) the input parameters are specified in the same manner (Figure 5.82). Taking his output as an example: when *Write His File* is checked, the output of history file is enabled. *His Output Interval* determines the interval at which the output data is stored in the file; the smaller the interval, the more detailed the output and the larger the file. By default, history output is written from the start until the end of the simulation. Optionally, output can be restricted to a certain time window: to specify different output start and/or stop times, check the box next to *Specify His Output Start Time* and/or *Specify His Output Stop Time* and enter the desired times next to the parameters *His Output Start Time* and *His Output Stop Time*.

When *Write Snapped Features* is activated, then shape files with snapped data will be generated for all quantities, such as fixed weirs and thin dams. For example, for fixed weirs a crest height is specified at both end of a fixed weir polyline. In between linear interpolation is applied, which can be checked via these shape files.

Below, the various output options are described in greater detail. *There is a special requirement on the output parameters for His and Restart files that the* Output Interval*,* Output Start Time *and* Output Stop Time *must be integer multiples of* User Time Step*. Optionally, the interval (*Output Stop Time−Output Start Time*) should be an integer multiple of* User Time Step*.*

Write his file

His Output Interval	Time interval for history time series.
His Output Start/Stop time	Restrict history output to a specified time window.
Write mass balance totals	Enable detailed mass balance time series output in the his file.
Write (misc.) structure parameters	Enable time series output across general structures, pumps, weirs and gates in the his file.

Write map file

Map Output Interval	Time interval for map field time series.
Map Output Start/Stop time	Restrict map output to a specified time window.
Specific Map Output Times	File containing specific time values at which to produce additional map output snapshots. It the value is not integer, it is firstly set to the least integer larger than or equal to this value. If the computational time does not hit the specified time value, the output snapshot is chosen to be at the time closest to the specified time value.
Write water levels, etc.	Several optionals for enabling/disabling certain quantities output in the map file.

Write restart file

Restart interval	Time interval for restart files.
Rst Output Start/Stop time	Restrict restart output to a specified time window.

Other output options

WAQ Output Interval	Time interval for D-Water Quality files in <DFM_DELWAQ_*mdu_name**.hyd>, etc.
Simulation statistics output interval	Interval for simulation progress output (on standard out and diagnostics file).
Timing statistics output interval	Interval for detailed timings output into <*mdu_name*_timings.txt> for expert performance analysis.

Example

One example of input and output parameters (in seconds) is given in Table 5.2. Table 5.3 shows the time (after *Reference Date* in seconds) when output files are generated. We explain this example as follows.

- The history ile has output interval 18 seconds. No parameters are specified for *His Output Start Time* and *His Output Stop Time*, which means that they are automatically set equal to *Start Time* and *Stop Time* of the simulation, respectively.
- The *Map Output Interval* is 6 seconds, and *Map Output Start Time* is 15 seconds. Since the *Map Output Stop Time* is not given, it is set to equal to *Stop Time*. Moreover, we hope to have output at time given in *Specified Map Output* as 30.5 and 42.1 seconds. These two values are firstly set to 31 and 43 seconds, respectively, in the simulation. Then there will be output snapshots if the computational time hit these two integers. Otherwise, as in this example, the output snapshots will be provided when the computational time hits the time that is greater and the closest to these integers, i.e. at 31.2 and 43.2 (as seen in Table 5.3).
- Three parameters are set for the output of the Rst files: the interval, start and stop time.

Input parameters		Output parameters	
Reference Date	2007-11-19 00:00:00	*His Output Interval*	00:00:18
User Time Step	00:00:00.3	*Map Output Interval*	00:00:06
Start Time	2007-11-19 00:00:03	*Map Output Start Time*	2007-11-19 00:00:15
Stop Time	2007-11-19 00:00:51	*Specific Map Output*	30.5, 42.1
		Rst Output Interval	00:00:09
		Rst Output Start Time	2007-11-19 00:00:12
		Rst Output Stop Time	2007-11-19 00:00:45

Table 5.2: *Input and output parameters of the example*

His file	3, 21, 39, 51
Map file	3, 15, 21, 27, 31.2, 33, 39, 43.2, 45, 51
Restart file	3, 12, 21, 30, 39, 45

Table 5.3: *Time (after* Reference Date *in seconds) of output files*

5.4.13 Miscellaneous

Within the miscellaneous sub-tab, various parameters in relation to waves and equatorial settings can be adjusted. Table 5.4 gives an overview and description of these parameters.

__Table 5.4:__ Overview and description miscellaneous parameters

Parameter	Description
Time step type	Leave at default.
Turbulence model	See **??**.
Turbulence advection	Leave at default.
Water level threshold	Max allowed water level difference between old and new time step in any cell. Run will abort if exceeded. (0 means disabled)
Velocity threshold	Max allowed velocity difference between old and new time step in any cell. Run will abort if exceeded. (0 means disabled)
Dry cell threshold	Flooding threshold at velocity points. Used in wetting and drying.

5.4.14 3D Layers

This tab is used to create a 3D model.

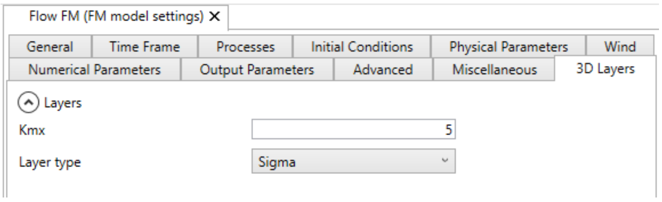
By default the model is 2D, which is specified by the parameter `Kmx` equal to 0.

A 3D model is created by specifying a positive number of layers (`Kmx` greater than 0).

The recommended type of vertical layering differs depending on the model application and the processes that you are interested in. There are two options:

- ◇ Sigma-layers: Layers in the σ-model increase or decrease in thickness as the water depth in the model increases or decreases. The relative thickness distribution of the different layers however remains fixed.
- ◇ Z-layers: Layers in the Z-model have a fixed thickness, which does not change as the water depth in the model varies. If the water depth drops below the cumulative thickness of all Z-layers, layer(s) will fall dry.

The most convenient way is to use sigma-layers, as follows.



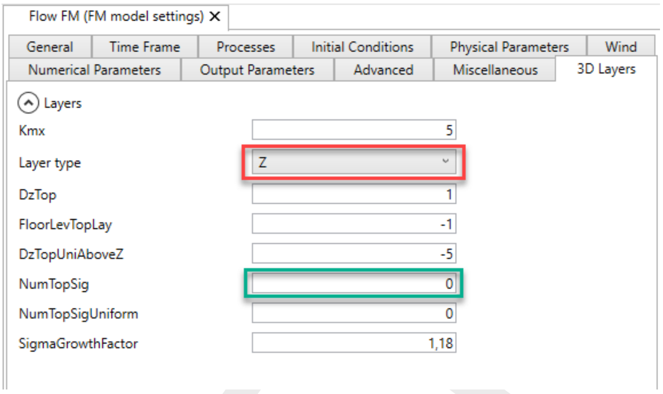
Kmx 5

Layer type Sigma

__Figure 5.83:__ Specification of sigma-layers

This way an evenly distributed vertical layering is specified (all layers are 20 percent of the water depth).

Z-layers are specified as follows.



Kmx	5
Layer type	Z
DzTop	1
FloorLevTopLay	-1
DzTopUniAboveZ	-5
NumTopSig	0
NumTopSigUniform	0
SigmaGrowthFactor	1,18

Figure 5.84: *Specification of Z-layers*

Table 5.5: *Overview and description of Z-layer parameters*

Parameter	**Description**
FloorLevTopLay	The floor level of the top layer
DzTop	z-layer thickness of layers above level DzTopUniAboveZ
DzTopUniAboveZ	The level above which the layers will have a uniform thickness of DzTop
NumTopSig	The number of sigma-layers on top of z-layers
NumTopSigUniform	Indicating whether the number of sigma-layers in a z-sigma-model is constant (=1) or decreasing (=0) (depending on local depth)
SigmaGrowthFactor	Layer thickness growth factor from DzTopUniAboveZ downwards

Note: By choosing NumTopSig greater than 0, a combination of Z- and sigma-layering is specified.

5.5 Save project, MDU file and attribute files

To save your D-Flow FM GUI project, navigate the menu ribbons to *File* and click *Save as*. Choose a location, specify a name and click *Save*. Your project will now be saved in a folder called <*name*.dsproj_data> and a <*name*.dsproj> file is written. Within this folder you will find all input ASCII input files of your model, output files of your model (if the model was run using the GUI) and zip folders containing your restart files. Be aware that the output files are stored within a separate folder in which the input files of your model are stored. The output folder on the same level as the folder containing model input files is empty.

To open a project, navigate the menu ribbons to *File* and click *Open*. Select the <∗.dsproj>-file of choice and click *Open*.

Importing model or data within a D-Flow FM GUI project can be achieved in two ways. Navigate the menu ribbons to *File* and click *Import*. A drop-down menu appears, allowing you to select what you want to import (Figure 5.85).

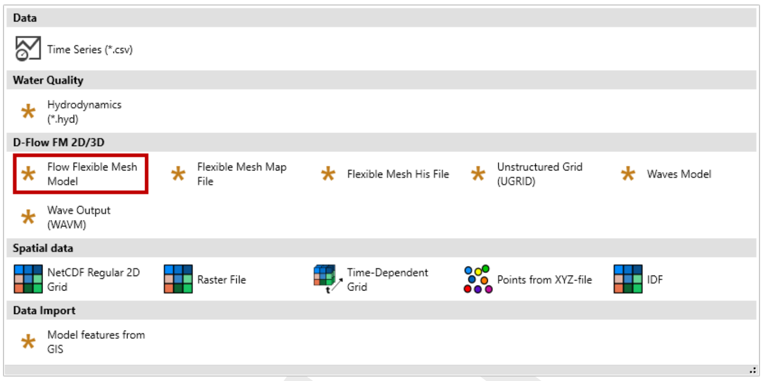


Figure 5.85: *Model/data import drop-down menu*

Alternatively, you can also right mouse click on the name of your project in the *Project* window and select *Import*. An import wizard appears, allowing you to select what you want to import (Figure 5.86).

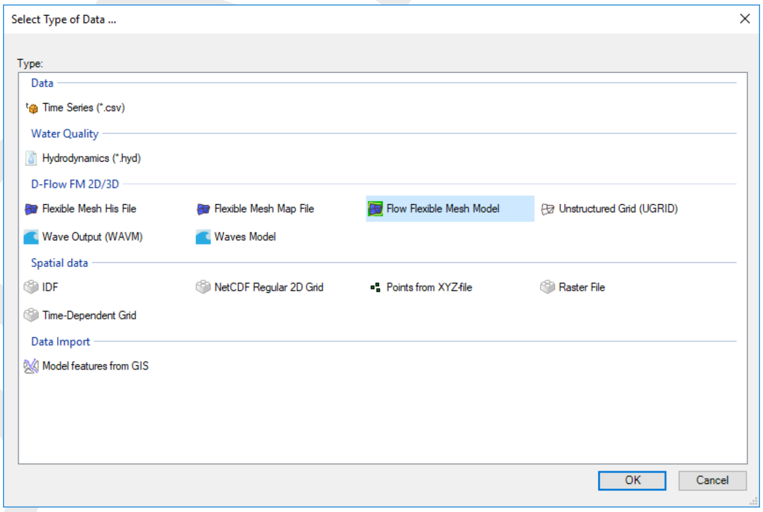


Figure 5.86: *Model/data import wizard*

Exporting your model can be achieved in the same fashion as importing your model. All model input files will be written to the folder you select. Be aware that model files exported to a folder in which other model files with the same names are present will be overwritten.

6 Running a model

6.1 Introduction

After defining the input for the D-Flow FM hydrodynamic simulation, the computation can be executed either via the D-Flow FM Graphical User Interface, GUI, (on Windows), see section 6.2, or using batch scripts (on Linux and Windows). Via the GUI, the status of the computation and possible messages are displayed in a separate messages window. When using a batch script, all messages are written to the diagnostics file (section F.1) and inside the terminal (stdout and stderr).

Use a batch script (see section 6.3) in the following cases:

1 Using MPI to run in parallel, see section 6.4
2 Using some queueing mechanism on a cluster
3 Running some unattended simulations, while continuing to work in the User Interface
4 Running on Linux.

6.2 Running a scenario using the User Interface

In the *Project* window, select the model you want to run by clicking the first sub-item of the desired model (Figure 7.1).

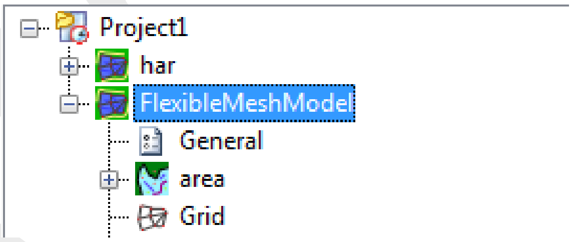


Figure 6.1: *Selecting the model you want to run in the* Project *window*

Starting the calculation can be achieved in two ways. You can navigate the menu ribbons; go to “Home” and in the group Run click on the button “Run Current”. To run all models that are opened in the *Project* window, click on “Run All” (Figure 6.2).

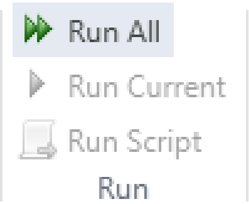
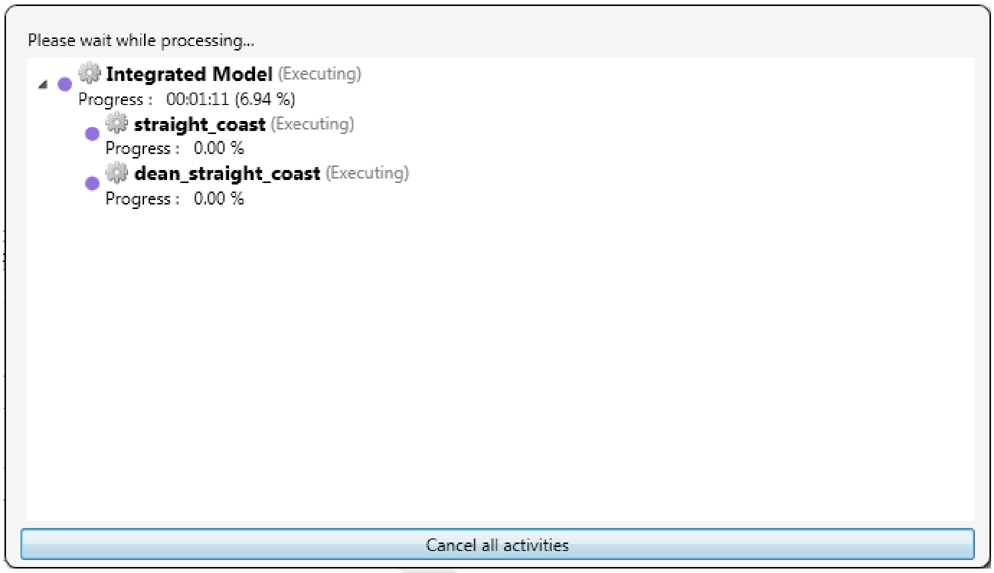


Figure 6.2: *Group Run in Home ribbon*

Alternatively, you can right mouse click the first attribute in the *Project* window of the model you want to run. Next, click “Run Model” to start the calculation. If you first select the model you want to run, the properties window (“View” “Properties”) will show several properties of the model you selected. If you set “ShowModelRunConsole” to true, the model run console of the computational core will be shown during the calculation, providing you with additional information during the model run (Figure 6.3).



Please wait while processing...

Integrated Model (Executing)
Progress : 00:01:11 (6.94 %)

straight_coast (Executing)
Progress : 0.00 %

dean_straight_coast (Executing)
Progress : 0.00 %

Figure 6.3: *Run console D-Flow FM User Interface*

If you cancel the run by clicking “Cancel all activities”, model results are stored up to the point where you cancel the run.

6.3 Running a scenario using a batch script

In general, the following steps have to be carried out:

1 Set-up a D-Flow FM model in the GUI on Windows.
Currently, this is only possible for 2DH (depth-averaged) model simulations. Validate the model.

2 Export the model to either a ’DIMR configuration’ or a ’Flow Flexible Mesh model’, as described in section 3.2 of this user manual. A ’DIMR configuration’ is necessary when the computation consists of a D-Flow FM model integrated with another kernel (e.g. D-RTC or D-Waves). A ’Flow Flexible Mesh model’ is optional for a standalone D-Flow FM computation.
In case of a 3D model, manually change the input for the 3D keywords (number of layers, layer percentages, . . .). **Note:** In the future a GUI for 3D modelling will become available.

3 Copy the complete set of model input files to the machine doing the computation (Windows or Linux).

4 In case of a parallel MPI computation: partition the model. Partitioning can be executed both on Windows (before copying the input files) and Linux (after copying the input files), see section 6.4.2.

5 Start the D-Flow FM model simulation. See section 6.3.2.

6 In case of a parallel MPI computation: after completion of the model simulation, map merging of the output files is often carried out in order to simplify the postprocessing. See section 6.4.4.2.

7 Use QUICKPLOT for postprocessing, see section 6.4.4 and the QUICKPLOT user manual.

6.3.1 Commandline executables

Executables should be called via run-scripts prepared by Deltares. To use these prepared scripts, you are supposed to write a one-lined script in your working directory.

Windows (example of a one-lined script that calls a Deltares-prepared script):
`call <installDir>\x64\dimr\scripts\run_dimr.bat`

Linux (example of a one-lined script that calls a Deltares-prepared script):
`<installDir>/lnx64/bin/run_dimr.sh`

Note: Example models, with one-line-scripts, are included in the (open source) code. After registering yourself at https://oss.deltares.nl/, you can have a look at:
https://svn.oss.deltares.nl/repos/delft3d/trunk/examples
/12_dflowfm/test_data/e100_f02_c02-FriesianInlet_schematic_FM

Advantages of this construction:

- ◇ The script will take care of preparations like setting environment parameters
- ◇ The script will call the correct binary with the correct arguments
- ◇ Uniform look and feel
- ◇ In case of multiple installations, there is no confusion about what version is being used. The prepared scripts, located in an installation directory, will always use the executables in that same installation directory.

Differences between the Windows and Linux prepared run-scripts:
On Windows:

- ◇ The location of the prepared scripts is `x64\<modulename>\scripts` inside
  `<Your installation base dir>\plugins\DeltaShell.Dimr\kernels\`
  where `<Your installation base dir>` is typically:
  `c:\Program Files\Deltares\Delft3D Flexible Mesh Suite HMWQ (xxxx.xx)`
- ◇ The file extension is <.bat> or <.cmd>.

On Linux:

- ◇ The location of the prepared scripts is `lnx64\bin` inside
  `<Your installation base dir>`
  where `<Your installation base dir>` is typically:
  `/opt/delft3dfm/xxxx.xx`
- ◇ The file extension is <.sh> or no extension at all.

6.3.2 Running D-Flow FM

Export the input files in the GUI, see section 3.2. Be sure that the GUI produces the 'DIMR configuration' file too, typically called <dimr_config.xml>. Write a one-line-script to call the 'run_dimr' script in the installation directory. The easiest Windows example <run.bat>:

```
call <installDir>\x64\dimr\scripts\run_dimr.bat dimr_config.xml
```

The easiest Linux example <run.sh>:

```
<installDir>/lnx64/bin/run_dimr.sh dimr_config.xml
```

Put the one-line-script in the same directory as file <dimr_config.xml> and execute it; on Windows double-click the script or execute it in a command box; on Linux execute it in a command box.

Note: Example models, with run scripts, are included in the (open source) code. After registering yourself at https://oss.deltares.nl/, you can have a look at:
https://svn.oss.deltares.nl/repos/delft3d/trunk/examples
/12_dflowfm/test_data/e100_f02_c02-FriesianInlet_schematic_FM

Remarks:

- Executing <dflowfm> instead of <dimr> is an option when running D-Flow FM stand-alone (not coupled to D-Water Quality, D-RTC, D-Waves etc.). To do this, replace 'run_dimr' by 'run_dflowfm' and the dimr config file by the mdu-file in the one-line-script.
- Calling 'run_dimr' with the argument `--help` will show the options
- Argument after the separator '`--`' are passed through to the D-Flow FM kernel (if the D-Flow FM kernel is the only kernel being started).

6.4 Parallel calculations using MPI

6.4.1 Introduction

This section describes parallel computing with D-Flow FM based on the *Message Passing Interface* system (MPI). This can be run both on computing clusters with distributed memory as well as shared memory machines with multiple processors and/or multiple CPU cores. A parallelMPI computation can use multiple machines in a Linux cluster. On Windows, only one machine can be used.

The goal of parallelization of D-Flow FM is twofold: faster computations and the ability to model problems that do not fit on a single machine. A less powerful, yet possibly attractive performance improvement is offered by D-Flow FM's OpenMP-based parallelization (section 6.5.1).

Technical backgrounds on the parallel algorithms in D-Flow FM are described in the Technical Reference Manual Deltares (2024a).

Workflow of a parallel run

A parallel MPI run divides the work between multiple processes. The model is split in submodels or partitions. A separate process is started for each partition, solving the equations on that partition. It will communicate with the neighbouring processes where needed and generate output only for it's own partition. Given a whole model, the workflow is as follows:

1. Partition the model (mesh and model definition file)
2. Execute the parallelMPI job on a multi-core machine or on several machines in a (Linux) cluster
3. Visualize the results from partitioned output files.

6.4.2 Partitioning a model

In D-Flow FM a model is defined by the model definition file, <.mdu>, the network (mesh) file, <_net.nc>, external forcing/boundary condition files, <.ext>, et cetera. Partitioning a model concerns partitioning of <.mdu>-file and the network file <_net.nc>. Other files are shared by all submodels and do *not* need to be partitioned, they should only be available to all processes.

Partitioning a model can be done by executing D-Flow FM via a command line call:

```
> dflowfm-cli.exe --partition:ndomains=n:icgsolver=i <mdu-file>
```

This basic command reads the name of the network file from mdu-file, and generates `n` subdomain network files by the METIS software package (See Deltares (2024a) and references mentioned therein). Then, it creates `n` subdomain mdu-files where the parallel Krylov solver `icgsolver` is set to `i`. Here, `i` can be 6, the PETSc solver(recommended), or, 7, the parallel CG with MILU block preconditioning.

It is recommended to use a runscript to partition the model, that takes care of paths and environment parameters:

◇ **Windows**
On Windows one can use batch file `run_dflowfm.bat`, please see the following example:

```
"<installDir>\x64\dflowfm\scripts\run_dfmoutput.bat" "--partition:ndomains=8:
icgsolver=7" example.mdu
```

Note that the double quotes, `"..."`, are used for the input arguments, in order to preserve equal-sign, =, when passing those input arguments to the batch file.

◇ **Linux**
On Linux one can use `run_dflowfm.sh`:

```
<installDir>/lnx64/bin/run_dflowfm.sh --partition:ndomains=8:icgsovler=6
example.mdu
```

The above commands result in 8 subdomain mdu-files, and 8 subdomain network files. The subdomain mdu-files have names as <example_000j.mdu>, j=0,1,...,7. The different items in, e.g. <example_0000.mdu> with respect to the original mdu-file are:

```
[geometry]
NetFile        = example_0000_net.nc

[numerics]
Icgsolver      = 6 (or 7)
```

Assume the specified network file in the original mdu-file is <example_net.nc>, then the 8 subdomain network files have names <example_000j_net.nc>, j=0,1,...,7. In other words, they are:

```
example_0000_net.nc  example_0003_net.nc  example_0006_net.nc
example_0001_net.nc  example_0004_net.nc  example_0007_net.nc
example_0002_net.nc  example_0005_net.nc
```

Moreover, by default a separate file <DFM_interpreted_idomain_example_net.nc> is generated, which includes partition domain information and cell information. One can switch off generating this file by adding in the original mdu-file in the chapter `output` the keyword `Writepart_domain = 0`.

Note: In all the partitioning commands shown in this subsection, one can specify a network file <example_net.nc>, instead of the mdu-file <example.mdu>. This will only partition the network file. However, we strongly suggest to use the mdu-file in the partitioning commands, which partitions both the mdu-file and the network file. The reason is, when a dry points or/and enclosure file is specified in the mdu-file, the partition tool will first delete dry areas based on these files, and then do the partition. If one specifies network file in the partitioning command, then the resulting subdomain network files still include dry areas. Then after running the parallel simulation, the output map files do not contain those dry areas. These will bring inconsistence and further yield a problem if merging the map files.

6.4.2.1 More options for the partitioning

The commands in section 6.4.2 partitions the model, including partitioning the network file. This section describes the full possibilities for partitioning a network (mesh). The mesh can be automatically partitioned with the METIS software package, or manually by supplying polygons that define the subdomains. They both produce a cell colouring of the original mesh. In each subdomain, the cells are assigned the same colour, and augmented with ghost cells. This information is saved in variable "idomain" in the resulting partitioning mesh files, e.g. <example_NNNN_net.nc>, where NNNN is a subdomain index.

Partitioning with METIS

The command:

```
> dflowfm-cli.exe --partition:ndomains=n:icgsolver=i <mdu-file>
```

is a basic command to partition with METIS. An advanced command, which enables more options, is:

```
> dflowfm-cli.exe --partition:ndomains=n[:method=0|1][:genpolygon=0|1]
[:contiguous=0|1][:ugrid=0|1][:seed=i] <mdu-file>
```

The explanation of these options is as follows:

The partition method can be chosen via setting `method=1`, the multilevel K-Way method (default), or `method=0`, the Recursive Bisection method. We refer to Karypis (2013) for more details about these two methods.

Option `genpolygon` specifies if the command generates a partition polygon file (`genpolygon=1`), or not (`genpolygon=0`, default).

By default FM enforces contiguous partitioning, i.e. `contiguous=1`. This can be switched off by setting `contiguous=0`. Only the multilevel K-Way method can enable the contiguous partition. If one requires the Recursive Bisection method, then it is not possible to enforce contiguous partitioning.

In the situation when one gets error message "*The above METIS error message is not a problem. It means that partitioning failed for k-way method with option contiguous=1 because the input graph is not contiguous. Retrying partitioning now with contiguous=0.*", it is no need to worry because it retries partitioning without enforcing contiguous.

Option `ugrid=1` allows to enforce UGRID output. This is required for running parallel 1D models (and map merging the output). Running parallel 2D models (and map merging) works for both non-UGRID and UGRID mesh files.

Option `seed=i` with `i` a user-defined integer allows to pass a random seed to the METIS partitioner. This allows the user to have reproducible partitioning result between separate runs.

Note: These options can be used with scripts `run_dflowfm.bat` and `run_dflowfm.sh`.
Note: Partitioning script `generate_parallel_mdu.sh` is outdated and should not be used.

Partitioning with manually supplying polygongs

If the user wants to manually partition the mesh instead of applying METIS, then he has to provide a polygon file <userpol.pol> which determines the mesh partition. In this situation, he can use the following command:

```
> dflowfm-cli.exe --partition:icgsolver=i <mdu-file> <userpol.pol>
```

Comparing with the command using METIS, this command does not use option `ndomains`, but adds the polygon file name at the end.

Note: Similarly, to use a script `run_dflowfm.bat` or `run_dflowfm.sh`, one can remove option `ndomains`, and add the polygon file name at the end of the command, as below:

- On Windows one can use batch file `run_dflowfm.bat`, please see the following example:

  ```
  "<installDir>\x64\dflowfm\scripts\run_dfmoutput.bat" "--partition:icgsolver=7"
  example.mdu part.pol
  ```

  Note that `" "` is used for the input arguments, in order to preserve = when passing those input arguments to the batch file.
- On Linux one can use `run_dflowfm.sh`:

  ```
  <installDir>/lnx64/bin/run_dflowfm.sh --partition:icgsovler=6
  example.mdu part.pol
  ```

Where, 'part.pol' is the name of a partition polygon file.

And the partitioning obeys the following rules:

- if the polygons have a z-value specified, it is considered a subdomain number,
- if the polygons have no z-value specified, its order determines the corresponding subdomain number,
- if a cell is not inside at least one polygon, it is assigned to subdomain 0,

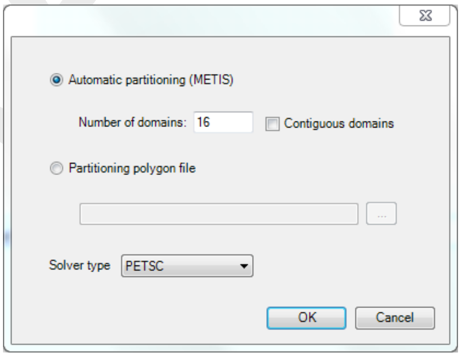
- ◇ if a cell is inside only one polygon, it is assigned to the subdomain defined by that polygon,
- ◇ if a cell is inside more than one polygon, it is assigned to the subdomain with the highest number.

In other words, the polygons may be overlapping and the largest subdomain number is taken in the overlapping regions. If the polygons have no z-value, the polygon order determines the corresponding subdomain number, i.e. the first polygon corresponds to subdomain 1 et cetera and there is no polygon defining subdomain 0.

Partitioning the mesh with METIS via the User Interface

A separate mesh partitioning is not needed when following the instructions for partitioning a complete model in section 6.4.2. This section describes the full posibilities for partitioning a mesh.

We will firstly focus on the METIS partitioner. Partitioning the mesh from within the User Interface is achieved by the following steps. In the *Project* window, select the model you want to run by means of clicking on the desired model (Figure 7.1). A right-mouse click will open the context menu, then select *Export...*. In the window **Select Type of Data...**, choose *Partition exporter* and the partitioning dialog will appear (Figure 6.4).



● Automatic partitioning (METIS)

Number of domains: 16 □ Contiguous domains

○ Partitioning polygon file

Solver type PETSC

Figure 6.4: *Partioning exporter dialog*

Enter the desired amount of subdomains, and typically leave the *contiguous* option switched off and the solver type at its default. After pressing *OK*, a file dialog will appear. Enter the name of the mdu-file, *without* any trailing '_000x' partition numbers: these will be added automatically.

Partitioning the mesh with the graphical user interface is achieved by the following steps: You are prompted for Contiguous domains are not necessary for parallel computations in D-Flow FM and often METIS produces contiguous subdomains without enforcing it. Still, some users may want to explicitly enforce contiguous domains. If you want to do so, make sure that your unpartitioned mesh is contiguous. Not being so may cause errors, Note that there is *no* polygon defining subdomain 0. Parts of the mesh not confined by any polygon are assumed to be in subdomain 0. In other words, there are at least $N - 1$ polygons defining N subdomains. In that case, regenerate the cell colors/domain numbers again by selecting *Operations* → *Generate domain numbers (polygon or METIS)*. Note that you will not be asked to specify the number of subdomains. The cell coloring/domain numbering is now based on the (modified) polygons and not produced by the METIS partitioner. Manual partitioning with user-specified polygons will be explained in the next section, The entered mesh filename is a *basename* that will be used to derive the partitioning filenames, e.g. <example_net.nc> will produce:

6.4.2.2 Partitioning of 1D2D models

The partitioning of models with a 1D network or even a combined 1D2D mesh is entirely the same as discussed before for a 2D mesh. This section contains some additional information about the resulting partitioned mesh files and the ghost cells in them.

The spatial discretization of the (momentum) advection and diffusion terms in the shallow water equations requires four levels of ghost cells beyond the partition boundary in each partition. More information can be found in (Deltares, 2024a, section 8.1.1). Figure 6.5 shows a partitioned 1D2D model with two partitions, where the four layers of ghost cells are clearly distinguishable in 2D. In the 1D channel, the 1D2D connections affect the ghost levels, but the partitioning algorithm accounts for this.

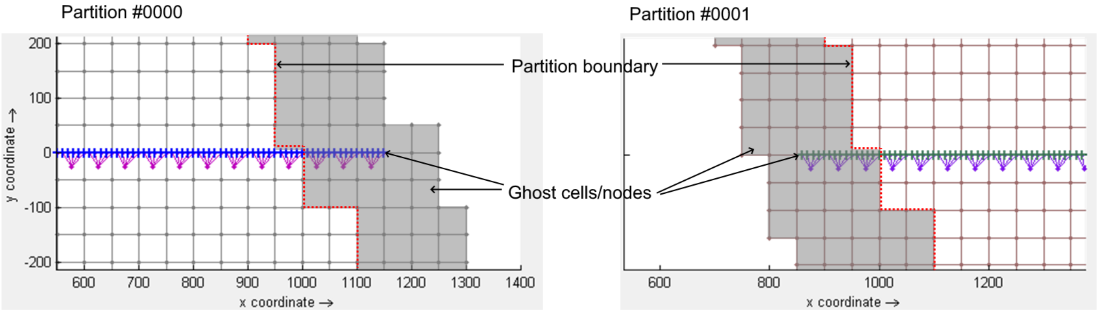


Figure 6.5: Partitioned 1D2D mesh with two partitions and illustrated ghost cells.

6.4.2.3 Partitioning the mdu-file

A separate mdu-partitioning is not needed when following the instructions for partitioning a complete model in section 6.4.2.

Partitioning script `generate_parallel_mdu.sh` is outdated and should not be used.

6.4.2.4 Remaining model input

A parallel run of a D-Flow FM model needs only partitioned <.mdu> and <_net.nc> files. All other model input is the same as for a standalone run, e.g., meteo forcings, boundary conditions, observation stations and more. These are generally copied to the working directory by the parallel job submission script.

6.4.3 Running a parallel job

On Windows, parallel jobs can only be executed using one node/machine/PC. On Linux, multiple nodes/machines can be used.

Note: Example models, with run scripts, are included in the (open source) code. After registering yourself at https://oss.deltares.nl/, you can have a look at:
https://svn.oss.deltares.nl/repos/delft3d/trunk/examples
/12_dflowfm/test_data/e02_f14_c040_westerscheldt

On Windows, the so-called *hydra* service must be running to enable parallel computations using MPI. The following instructions can also be found as remark in file <run.bat> in the example directory mentioned above:

➢ Preparation: Check that your Delft3D installation contains <...\x64\share\bin\hydra_service.exe>. Optionally copy it to a local directory (it will run as a service).

➢ *Microsoft Windows start button* → *Command Prompt* → *rightmouseclick on the related App* → *Run as Administrator*. A command box will appear.

➢ In this command box, go to the location of <hydra_service.exe> and execute:
`hydra_service -install`
When there is a *hydra* service already running on the machine, it must be ended first, using the Microsoft Task Manager or in the command box: `hydra_service -uninstall`

6.4.3.1 A parallel simulation via DIMR

Below is shown how to start a parallel simulation on Linux:

```
<installDir>/lnx64/bin/run_dimr.sh -c 3 dimr_config.xml
```

And for Windows:

```
call <installDir>\x64\dimr\scripts\run_dimr_parallel.bat 3 dimr_config.xml
```

This assumes that the model has already been partitioned in three partitions. When running via DIMR, in the file <dimr_config.xml>, in the section about the D-Flow FM component, the following two lines must be present:

```
<process>0 1 2</process>
<mpiCommunicator>DFM_COMM_DFMWORLD</mpiCommunicator>
```

where `<process>` must match exactly with the number of processes.

A Linux cluster usually has a queueing system with a scheduler. Example submission scripts are available upon request.

6.4.3.2 A parallel simulation on Linux via D-Flow FM

To run a parallel simulation with a D-Flow FM model on a Linux cluster you have to prepare a submission script with the specific options that the job scheduler on your cluster requires.

Below a simple example is shown of the D-Flow FM submission script for a Linux cluster with the Grid Engine scheduler:

```
#!/bin/bash
#$ -V
#$ -q test
#$ -cwd
#$ -N My_DFlowFM_JOB
#$ -m bea
#$ -M my.email@provider.net

export LD_LIBRARY_PATH=$DFLOWFM/lib:$LD_LIBRARY_PATH
export PATH=$DFLOWFM/bin:$PATH

mpiexec -np 4 dflowfm --autostartstop YOUR_MDU_FILE.mdu >out.txt 2>err.txt
```

Assumptions related to the simplified example above:

- ◇ The D-Flow FM simulation has already been partitioned into four domains.
- ◇ The location of the D-Flow FM binaries has been stored in environment parameter 'DFLOWFM'.
- ◇ D-Flow FM is used as a standalone executable, not as part of a DIMR computation.
- ◇ The example script is used in the actual submission command on a Linux cluster, together with a specificiation of the number of nodes to be used.

The options used above are:

-V Specify that all environment variables active within the qsub utility be exported to the context of the job.

-q Specify the queue 'test' to be used for this job, if absent default queue is used.

-cwd Execute the job from the current working directory.

-N Specify the name of the job.

-m Specifies which message type should be emailed (b=beginning of job, e=end of job, a=abort of job.

-M Specifies the email address to send the notification.

In order to submit more complicated, e.g. multi-node simulations, additional options that are scheduler dependent have to be added.

6.4.4 Visualizing the results of a parallel run

The map and history output files (as introduced in section 5.4.12) deserve special attention in parallel runs.

The history file — with time series for observation points, structures and more — is written only by process #0, and all model-global data has already been aggregated into that single file: <*mdu_name*_0000_his.nc>.

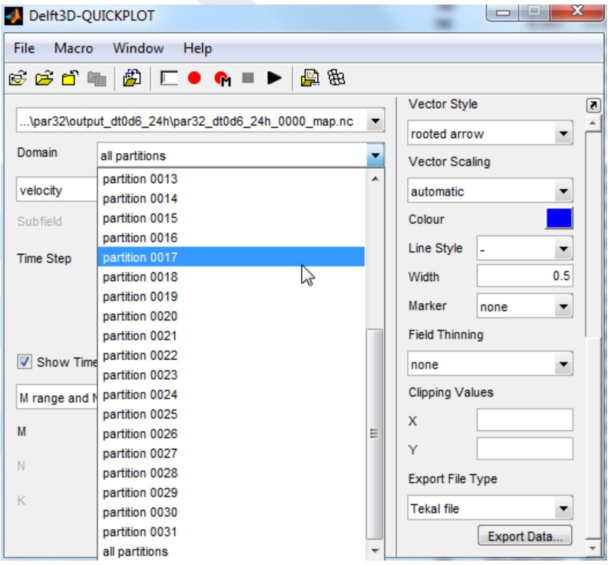
The map file — with full-grid output of flow fields — is written for each domain separately as <*mdu_name*_000X_map.nc>. This saves communication overhead during the parallel run.

The partitioned map files contain duplicate points, since each file also contains the domain's ghost nodes. For postprocessing these map files, two options are now available:

1 Direct plotting of the set of all map files in Delft3D-QUICKPLOT: the partitioned file series will be recognized automatically, and the partion results will be drawn on top of each other. For water levels this gives good results.
2 Merging the partioned map files into a single global map file with the `dfmoutput` tool. The resulting map file can then be loaded again in Delft3D-QUICKPLOT and other post-processing utilities.

6.4.4.1 Plotting all partitioned map files with Delft3D-QUICKPLOT

When opening one of the partitioned map files into Delft3D-QUICKPLOT, it will automatically detect that the map file is part of a series. An additional select list *Domain* appears, see Figure 6.6. Select either "all partitions", or a partition of your choice, and proceed with the plotting as normal (section 7.3).



...\par32\output_dt0d6_24h\par32_dt0d6_24h_0000_map.nc

Domain: all partitions

partition 0013
partition 0014
partition 0015
partition 0016
partition 0017
partition 0018
partition 0019
partition 0020
partition 0021
partition 0022
partition 0023
partition 0024
partition 0025
partition 0026
partition 0027
partition 0028
partition 0029
partition 0030
partition 0031
all partitions

velocity

☑ Show Time

M range and N

Vector Style: rooted arrow

Vector Scaling: automatic

Colour: 🟦

Line Style: -

Width: 0.5

Marker: none

Field Thinning: none

Clipping Values
X:
Y:

Export File Type: Tekal file

Figure 6.6: *Domain selector in Delft3D-QUICKPLOT for partitioned map files.*

6.4.4.2 Merging multiple map files into one

The partitioned map files of a parallel model run can be merged into a single global map file with the `dfmoutput` tool. It cuts off ghost nodes, and concatenates all grid points, taking care of correct global renumbering. Usage:

```
dfmoutput.exe mapmerge --infile FILE1 FILE2 [--outfile DSTFILE]
```

where `FILE1`, `FILE2` are the input files, e.g. <*mdu_name*_0000_map.nc>, <*mdu_name*_0001_map.nc>, and you can add more input files here. And `DSTFILE` is an optional output file name (the default is <*mdu_name*_merged_map.nc>).

It is recommended to use a runscript, that takes care of paths and environment parameters. Please see the following:

- **Windows**
  On Windows one can use batch file <run_dfmoutput.bat>:

  ```
  "<installDir>\x64\dflowfm\scripts\run_dfmoutput.bat" mapmerge
  --infile FILE1 FILE2 [--outfile DSTFILE]
  ```

  Remarks:
  - Note that for this script the number of arguments is limited to 9. Therefore, considering 2 arguments `mapmerge` and `--infile`, only maximal 7 map-file names could be passed, which means that only maximal 7 map-files can be merged if using the option `--infile`.
  - If one wants to merge more than 7 map-files using this batch file, then the user should use `--listfile` instead of `--infile`, then one can use the command option `--listfile` (see the list of more advanced options below), instead of `--infile`. An example is:

    ```
    dfmoutput.exe mapmerge --listfile list.txt
    ```

    where file `list.txt` contains a list of names of all input map files, as shown below, with one file name on a single line.

    ```
    model_0000_map.nc
    model_0001_map.nc
    model_0002_map.nc
    model_0003_map.nc
    model_0004_map.nc
    model_0005_map.nc
    model_0006_map.nc
    model_0007_map.nc
    ```

- **Linux**
  On Linux one can use <run_dfmoutput.sh>:

  ```
  <installDir>/lnx64/bin/run_dfmoutput.sh -- mapmerge --infile FILE1 FILE2
  [--outfile DSTFILE]
  ```

  Note, in this command, there is a space between "`--`" and "`mapmerge`". Moreover, one can also use option "`--listfile`" instead of "`--infile`".

Remark:

- Since a restart file is a special type of map-file, partitioned restart files can also be merged using the above command.

The built-in help gives a list of more advanced options:

```
> dfmoutput mapmerge --help
usage:  dfmoutput mapmerge [--infile FILE1 [FILE2 FILE3...]] [--listfile LISTFILE]
        [--outfile DSTFILE] [--force] [--help] [--version]

Merge multiple map files from parallel run into one.

Optional switches:
   --infile FILE1 [FILE2...], -i FILE1 [FILE2...]
          default value
          One or more input files.
   --listfile LISTFILE, -F LISTFILE
          Pass contents of LISTFILE as input files.
   --outfile DSTFILE, -o DSTFILE
          Write output to file DSTFILE. Default: <model>_merged_map.nc
   --force, -f
          default value .false.
          Force overwriting of existing output file.
   --help, -h
```

```
          Print this help message
   --version, -v
          Print version

Examples:
   dfmoutput mapmerge         --infile model_0000_map.nc model_0001_map.nc
   dfmoutput max_running_mean --infile hisfile.nc
   dfmoutput max25            --infile hisfile.nc
   dfmoutput genfilter        --infile hisfile.nc  --intval 6
```

Merging 2D map files

2D (or 3D) map file merging has a few special aspects:

- All data variables and underlying spatial coordinate variables are merged into one file, cutting off all ghost cells.
- Near partition boundaries, the data written to the merged file is taken from the input partitioned file that "owns" the grid locations near those boundaries. For grid cells, this is the partition in whose interior the grid cells lie. For shared mesh nodes and edges the partition is based on the lowest partition number owning the grid cell that they are part of.
- The order of the merged data values is in a so-called concatenated way. This means that all non-ghost points are concatenated domain-by-domain, keeping the local ordering for each partitioned submodel. This means that the output map file is likely to have a different ordering than the single map file from a sequential model run. This should not be an issue for any postprocessing, since all CF- and UGRID- domain variables have been merged in a consistent way, so the file remains correctly self-contained.
- Any user-defined vector dimensions for multidimensional data are also copied, if they are necessary for variables defined on the spatial domain (for example velocity components, tracer substances, or vertical layering).

Merging 1D map files

1D map files can also be merged by the `dfmoutput` tool using the above command. There are a few special aspects to 1D map merging:

- The network data, those variables whose names start with `network1d`[1], are copied from only one of the input map files, since these have not been partitioned at all and as such need no special merging. These variables contain all information on network topology and geometry, as well as branch and node ids, some of which are needed for the valid definition of the 1D computational mesh.
- Variables of the 1D computational mesh, i.e., those variables whose names start with `mesh1d`[2], are merged in a way that cuts off the ghost nodes, just like for 2D. These variables include topology data of the 1D mesh (e.g. `mesh1d_edge_x`, `mesh1d_edge_nodes`) and data variables that are located on the 1D mesh (e.g. `mesh1d_s1`, `mesh1d_u1`). The order of the merged data depends on the topological location as follows:
  - All variables that are located on 1D nodes (pressure points), e.g., `mesh1d_node_x` and `mesh1d_s1`, are merged in the order corresponding to the global numbering of the nodes (see variable `mesh1d_flowelem_globalnr`).
  - Variables on other locations, e.g., `mesh1d_edge_x` and `mesh1d_u1`, are merged in a concatenated way, the same as merging 2D map files. This means that all non-ghost points are concatenated domain-by-domain, keeping the local ordering for each

[1] The exact detection of a network variable is not based on a particular name, but generically on `:cf_role=mesh_topology` and the presence of an `:edge_geometry` attribute. See Section B.2 for the specifications.

[2] The exact detection of a 1D mesh variable is not based on a particular name, but generically on `:cf_role=mesh_topology` and `:topology_dimension=1`.

partitioned submodel.

Merging 1D2D map files

A 1D2D map file can contain both a 1D mesh and a 2D mesh, and/or mesh contacts between them. Merging 1D2D map files could be done by the `dfmoutput` tool using the above same command. Here are a few special aspects to 1D2D map merging:

- The variables on the 1D mesh and on the 2D mesh are merged separately, using the methods that are described in the above sections "Merging 1D map files" and "Merging 2D map files", respectively.
- A 1D2D map file can contain zero, one, or more mesh contacts. A mesh contact is a group of the so-called 1D2D links, where each 1D2D link connects a 1D node and a 2D face. For each mesh contact in the map file, there are a mesh contact variable, and some data variables that are located on the mesh contact, such as velocity and discharge on the 1D2D links.
  A mesh contact variable has attribute `:cf_role=mesh_topology_contact`, and it gives, for each 1D2D link, the index number of its connected 1D node and 2D face.
  If a mesh contact variable exists in input map files, then it is merged using the following way:
  - the 1D2D links are merged in the concatenated way (see section "Merging 2D map files"). Here, the domain number of a 1D2D link is chosen to be the minimal domain number of the 1D node and 2D face that are connected by this 1D2D link.
  - For a certain 1D2D link in the merged file, the index numbering of its connected 1D node and 2D faces are the global numbers, that are read directly from variable "_flowelem_globalnr" in the input map files. NOTE: when both 1D and 2D meshes are present in an input file, the global numbering of 1D nodes includes the numbering of 2D faces as well. In the merged map file, the global numbering of 1D nodes is done separately for 1D nodes, and separately for the 2D faces.
  - Variables that locate on the mesh contact (1D2D links) are merged in the concatenated way (see section "Merging 2D map files").
- If there are more than one mesh contacts in input map files, then they are merged one by one, using the above method.

Supported situations

The map merge tool supports the following different merging situations:

- All input map files are in the old format, or are in the ugrid format.
- All input map files have only a 1D mesh, or only a 2D mesh.
- All input map files have both a 1D mesh and a 2D mesh.
- All input map files have both a 1D mesh and a 2D mesh, and at least one mesh contact.
- Some input map files have both a 1D mesh and a 2D mesh, some have only a 1D mesh or a 2D mesh.
- Some input map files have only 1D mesh, some have only 2D mesh.

6.5 Run time

The actual run time of a model can vary considerably depending on a variety of factors such as:

- ◇ The problem being solved, characterised by the number of active grid points, the number of layers in the vertical or the number of processes taken into account.
- ◇ The length of the simulation in time and the time step being used.
- ◇ The hardware configuration that is used and the work load of the processor.

For this reason, only some general considerations are given to determine the run time of a hydrodynamic simulation. On a PC or a workstation without separate I/O-processors the CPU time is the sum of the processor time and the I/O time.

The *processor time* required for a simulation is primarily determined by:

- ◇ The model definition, i.e., the number of active grid points and the number and type of the processes taken into account.
- ◇ The length of the simulated period in terms of the number of time steps executed.

The *I/O time* is determined by:

- ◇ The number of times the computed data are written to history, map, restart files and other communication files for water quality or wave model couplings.
- ◇ The number of observation points, cross-sections and the number of output parameters.

The simulation performance is defined as the CPU time per grid point per time step per constituent:

$$\text{simulation performance} = \frac{\text{CPU time}}{\text{Dnt} \cdot \text{Ndx}} \quad \text{[system seconds]}$$

where:

Dnt is the number of time steps executed
Ndx is the number of flow nodes

The simulation performance is written to the diagnostic file at the end of the simulation.

6.5.1 Multi-core performance improvements by OpenMP

D-Flow FM has built-in support for multi-core parallellism using OpenMP[3]. This speeds up calculations by employing multiple processor cores in a single (shared-memory) computer, e.g., a modern-day notebook. OpenMP-parallellism in D-Flow FM does not scale as well as MPI-parallellism (section 6.4), but it comes for free (not any change to model input necessary) and can give a welcome performance improvement (approximately double speed on an Intel quadcore CPU). It is strongly advised to limit the number of OpenMP-threads to one less than the number of physical cores in your machine, thus also ignoring any hyperthreading. Mixing OpenMP with MPI is possible, but the optimum with respect to computation time is expected with only MPI parallellisation.

An OpenMP example on Windows:

[3] http://www.openmp.org

```
set OMP_NUM_THREADS=3
call <installDir>\x64\dimr\scripts\run_dimr.bat dimr_config.xml
```

An OpenMP example on Linux:

```
export OMP_NUM_THREADS=3
<installDir>/lnx64/bin/run_dimr.sh dimr_config.xml
```

6.6 Files and file sizes

For estimating the required disk space the following files are important:

- ◇ history file
- ◇ map file
- ◇ restart file

6.6.1 History file

The size of the history file is determined by:

- ◇ The number of monitoring points (observation points + cross-sections): H1.
- ◇ The number of quantities stored: H2.
- ◇ The number of additional process parameters, such as salinity, temperature, constituents and turbulence quantities, taken into account in the simulation: H3.
- ◇ The number of time the history data is written to the history file: H4.

You can estimate the size of a history file (in bytes) from the following equation:

size history file = $H1 \cdot (H2 + H3) \cdot H4 \cdot 8$ bytes.

As a first approximation you can use H2 = 8.

Example

For a 2D simulation with density driven currents (salinity and temperature), a simulated period of 12 hrs 30 min, a time integration step of 5 minutes, 30 monitoring points and each time step being stored, the size of the history file will be of the order of 384 kBytes. For the same model but now with 10 layers in the vertical the file size will increase to about 4 MBytes. These estimates show that history files are rather small. Unless the number of monitoring points is excessively large the history files are typically much smaller than the map output files.

6.6.2 Map file

The size of the map file is determined by:

- ◇ The size of the model, i.e. the number of grid cells multiplied by the number of layers (Ndxi · Kmax): M1n, and the number of flow links (open grid cell edges) multiplied by the number of layers (Lnx · Kmax): M1l.
- ◇ The number of quantities stored on grid cells and flow links: M2n, M2l, respectively.
- ◇ The number of process parameters taken into account, such as salinity, temperature, constituents and turbulence quantities: M3.
- ◇ The number of time steps for which the map file is written: M4.

Remark:

◇ For a more refined estimate you should distinguish between parameters that depend or not on the number of layers used (such as the water level). For a 3D simulation the latter quantities can be neglected, for a 2D simulation they must be accounted for. As a first estimate we double the number of quantities M2 in a 2D simulation.

As a first approximation you can use M2n = 5, M2l = 5 for a 3D simulation and M2n = 8, M2l = 5 for a 2D simulation.

You can estimate the size of a map file (in bytes) from the following equation:

size map file =[M1n · (M2n + M3) + M1l · M2l] · M4 · 8 bytes.

Example

For a 2D simulation with 6800 grid cells and 13000 flow links, simulation results stored for a period of 7 days, and the file is written with an interval of 60 minutes the size of the map file will be about 161 MBytes. For larger models the map file can easily become excessively large, as result the map file is less frequently written, for instance every 2 or 3 hours.

6.6.3 Restart file

A restart file is a special type of map file, where only one time snapshot per file is saved (i.e, M4 = 1). No grid or flow geometry information is stored in a restart file, except for the flow cell/link information (denoted by M5). Moreover, the restart files obtained after a parallel run contain some necessary information about parallelization (denoted by M6). Similarly to the equation of size map file above, one can write the estimating equation as follows:

size rst file = [M1n · (M2n + M3) + M1l · M2l + M5 + M6] · 8 bytes,

where M6 = 0 for a sequential run.

6.7 D-Flow FM command-line arguments

Sometimes it's necessary to run the D-Flow FM binary from the command line, without using any prepared script. The binary is named `dflowfm-cli.exe` (`dflowfm` on Linux). A basic non-interactive run is started by:

```
> dflowfm-cli --autostartstop MDUFILE
```

In the box below, a full list of command-line options and arguments is shown:

```
> dflowfm-cli --help
 Usage: dflowfm-cli [OPTIONS] [FILE]...

 FILE may be one of the following types:
  *.mdu                 model definition file
  *.cfg                 display settings file
  *.pol, *.xyz, etc.    additional model files

 Options:
   --autostart MDUFILE
       Auto-start the model run, and wait upon completion.

   --autostartstop MDUFILE
```

```
      Auto-start the model run, and exit upon completion.

  --noautostart MDUFILE
      Disable any AutoStart option in the MDU file (if any).

  --partition:OPTS (MDUFILE|NETFILE) [POLFILE]
      Partitions the unstructured model+grid in MDUFILE into multiple files.
             or: the unstructured grid in NETFILE into multiple files.

      POLFILE is an optional polygon file which defines the partitions.
      Only used when ndomains in OPTS is undefined or 0.

      OPTS is a colon-separated list opt1=val1:opt2=val2:...
        ndomains  = N     Number of partitions.
        method    = [123] Partition method: K-Way(1, default), Recursive Bisect
ion(2), Mesh-dual(3).
        genpolygon= [01]  Generate partition polygon(1), or not (0).
        contiguous= [01]  Enforce contiguous grid cells in each domain.
                          Only available when K-Way is enabled (method=1).
        icgsolver = [67]  Parallel CG solver (When MDUFILE is specified).
                          6: PETSc (default for parallel), 7: parallel GS+CG.
        ugrid     = [01]  write cf UGRID 0.8 (0, default) or UGRID 1.0 (1)
        seed      = i     User-defined random seed value, for reproducible
                          partitionings. Only used when i not equal to 0.

  -t N, --threads N
      Set maximum number of OpenMP threads. N must be a positive integer.

  --processlibrary PROCESSLIBRARYFILE
      Specify the process library file to be used for water quality processes.

  --openprocessdllso OPENPROCESSDLLSOFILE
      Specify the open process dll/so file with additional subroutines to be
      used for water quality processes.

  --bloomspecies BLOOMSPECIESFILE
      Specify the BLOOM species definition file to be used for water quality
      processes.

  --refine:OPTS NETFILE
      Refine the unstructured grid in NETFILE from commandline.
      OPTS is a colon-separated list opt1=val1:opt2=val2:...
          hmin=VAL
          dtmax=VAL
          maxlevel=M
          connect=[01]
          directional=[01]
          outsidecell=[01]
          drypointsfile=<filename (*.pol, or cutcellpolygons.lst)>

  --make1d2dlinks[:OPTS] NETFILE [-o OUTPUTFILE]
      Make 1d2d links for the given NETFILE and save the resulting net.
      OPTS is a colon-separated list opt1=val1:opt2=val2:...
        method       = (1to1 | 1ton_emb | 1ton_lat | long)  Coupling method.
        linktype     = N    The link type (kn3) that will be used for all links
                            (only for method=1to1).
        connect1dend = VAL  The search distance for coupling 1D endpoints.
      OUTPUTFILE is the name under which the file will be saved.
        When not specified, the original NETFILE will be overwritten.

 --cutcells NETFILE
      Cut the unstructured grid in NETFILE with the polygons specified
      in a file called 'cutcellpolygons.lst'.

 --no-geom-cache
      Do not load nor save cache file with geometry information.

  -q, --quiet
      Minimal output: Only (fatal) errors are shown.

  --verbose[:level_stdout[:level_dia]], e.g., --verbose:INFO:DEBUG
      Set verbosity level of output on standard out and in diagnostics file.
      where level is in: {DEBUG|INFO|WARNING|ERROR|FATAL}
      Levels are optional, default is INFO on screen, DEBUG in dia file.

  -h, --help
      Display this help information and exit.
```

```
    -v, --version
        Output version information and exit.
```

6.8 Restart a simulation

D-Flow FM allows to restart a simulation, not from the original starting time, but from a user-specified time with all the information at that time. In other words, for a model which has a long spinup time from t_{start}, instead of restarting from t_{start}, one can restart another simulation of the same model from a later time t_{rst}, where $t_{\mathrm{start}} \leq t_{\mathrm{rst}} < t_{\mathrm{stop}}$. And the results for times $t_{\mathrm{rst}} \leq t \leq t_{\mathrm{stop}}$ are the same as in the original simulation run.

This can be achieved by following steps:

1 In order to obtain restart files <_rst.nc>, this needs to be enabled in the output parameters (see section 5.4.12 for more details about how to set output parameters).
2 Run the original model from t_{start} to t_{stop}, resulting in restart files.
3 To restart the simulation from time t_{rst}, in the mdu-file modify `TStart` to t_{rst}, *and* fill in the name of corresponding restart file in `RestartFile` of the `[restart]`-section. Moreover, place the corresponding restart file relative to the directory of this mdu-file.

To restart a parallel simulation, one can use any of the following methods:

Method 1 On each subdomain use its own restart file.
Method 2 Merge all the subdomain restart files into a single global restart file (see section 6.4.4.2), and then use it for all the subdomains. This means to put the name of this merged file as `RestartFile` in all subdomain mdu-files. This method is supported when restarting with the same, or a different domain partitioning. Moreover, such a merged restart file can also be used to restart a sequential run.

The above two methods require taking care of the name of the restart file in subdomain mdu-files. D-Flow FM automatically fills in the correct names, when partition an mdu-file to subdomain mdu-files (see section 6.4.2), in the following way:

For Method 1 When <*mduname*_YYMMDD_HHMMSS_rst.nc> is filled in `RestartFile` of the original mdu-file, partition this mdu-file gives subdomain mdu-files with `RestartFile` as <*mduname*_000X_YYMMDD_HHMMSS_rst.nc> for each subdomain <000X>.
For Method 2 If the string word "merged" is contained in the name of `RestartFile` in the original mdu-file, then this file name is kept to all the subdomain mdu-files after partitioning.

It is also possible to use a map file as a restart file, following the above approaches. Notice that in this case, one needs to specify the `RestartDateTime` in mdu-file as well. However, we strongly recommend to use <_rst.nc> file instead of a map file.

6.9 Inspecting a model with the volume tool

With the use of the volume tool the user can generate a Netcdf file containing aggregated volumes and surface areas on 1D gridcells.

The aggregation can be performed on a complete model, on branches, on gridpoints or all three. Two matrices are written to the Netcdf file. For each level in an array of levels they contain:

1 The total volume below the level.
2 The surface area at the level.

The first index of the two matrices corresponds with the levels array. Levels start at the lowest point in the model and stops at the highest point in the model. Together with a given increment, levels are defined by successive increments from the lowest point until the highest point is reached.

The second index of the two matrices will correspond to a list with IDs. For the total volume for the complete model the ID-name will be 'Total'. For the branches the branch IDs will be used. For the gridpoints the node IDs will be used, written in a separate file also containing 1D geometry information.

The volume tool is called by:

```
usage:  dfm_volume_tool --mdufile FILE --increment INCREMENT [--output OUTPUT_TYPE]
                        [--outputfile OUTPUT_FILE] [--gridoutputfile GRID_OUTPUT_FILE]
                        [--help] [--version]
```

Required switches:

◇ `--mdufile FILE, -f FILE`
Name of the model definition file.
◇ `--increment INCREMENT, -i INCREMENT}`
Increment between volume table levels.

Optional switches:

◇ `--output OUTPUT_TYPE, -o OUTPUT_TYPE`
Output type: "`Total`", "`Branches`", "`Gridpoints`", "`All`".
◇ `--outputfile OUTPUT_FILE, -p OUTPUT_FILE`
Name of the output file.
◇ `--gridoutputfile GRID_OUTPUT_FILE, -g GRID_OUTPUT_FILE`
Name of the output file for the Gridpoints output option.
◇ `--help, -h`
Print help message.
◇ `--version, -v`
Print version.

Note: In some models storage nodes are defined with a street level of 9999.0 m. This is to denote there is no street storage, and in that case the highest level will be 9999 m. As a result the levels in the volume tables will go up to 9999 m.

Warning:

- ◇ Currently, running `dfm_volume_tool` will overwrite any existing output files (<*_map.nc>, <*_his.nc>, <*_clm.nc><*_rst.nc>). Make a backup copy first if you want to keep existing model results.

6.10 Frequently asked questions

This chapter aims to help you with common questions that may arise while using D-Flow FM.

1 **Question**
My model does not run/crashes. What's wrong?
Answer
The diagnostics file is the starting point for finding out what went wrong. See section F.1 for a detailed description of the contents of this file, and the order of model run output. Globally, ask yourself the following questions:

- ◇ Was the mdu-file found?
- ◇ If yes, was the model successfully loaded? Common mistakes are missing boundary or meteorological forcings file.
- ◇ If yes, was the time loop successfully started? Possible errors are non-writable output files.
- ◇ If yes, does the dia file contain any messages from during the time loop? Possible errors are solver convergence errors.
- ◇ If not, did the run end successfully?

2 **Question**
I get a warning that my network is non-orthogonal. Can I loosen the orthogonality treshold?
Answer
Unfortunately, no. Orthogonality is very important for accuracy: advised orthogonality values for your grid are around 0.01, preferrably lower. The current treshold is already very high at a value of 0.5. Use RGFGRID to improve your grid orthogonality (and smoothness).

3 **Question**
When running a computation with a batch script, should I call `DIMR` or `dflowfm-cli`?
Answer
When running D-Flow FM online in combination with for example D-RTC, you have to run via DIMR. When running a standalone D-Flow FM computation, you can choose. Deltares aims on one method for running computations and that is via DIMR. But not all D-Flow FM functionality is yet available via DIMR.

7 Visualize results

7.1 Introduction

A model run will produce two types of output:

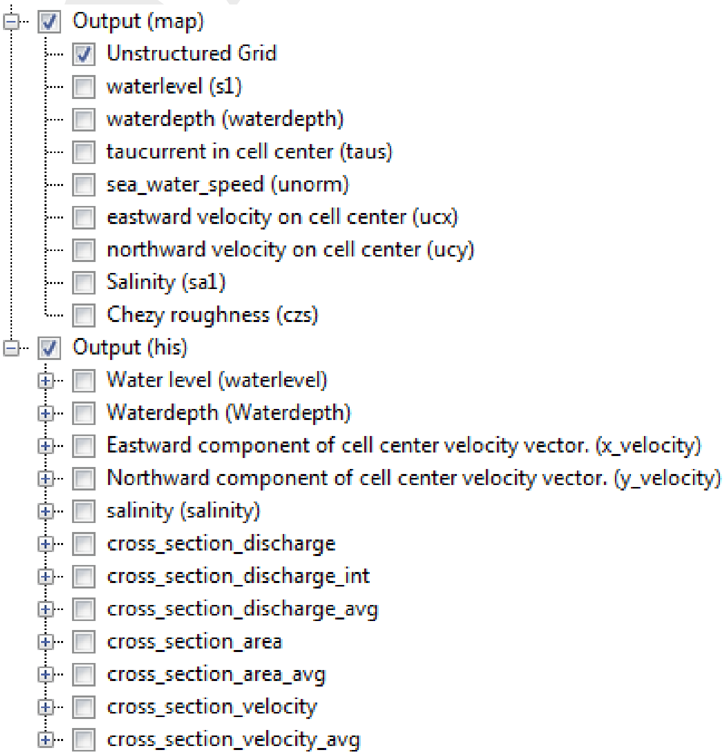
1 a graph or history (<∗.his>-file) for a specific quantity on a location
2 a map (<∗.map>-file) for a specific quantity

Both are stored in files within the project, as described in section 6.6.

D-Flow FM provides basic visualization of the model and the model results. Advanced and tailor-made visualization is possible by the export of the his- and/or map-files, and inspection with dedicated visualization applications. Some are provided and described below.

7.2 Visualization with the User Interface

When your model run has finished, a new folder called “Output” has appeared at the bottom of the attributes of your model in the *Map* window. Inside this folder, all output quantities of the his-, and map-files can be found, as well as a folder called “States”, in which you find all restart files written during the model run. Output folders have also appeared in the *Map* window; “Output (map)” and “Output (his)”. Both the results of your map and his files can be viewed in the central map by means of enabling the desired quantities from the *Map* window (Figure 7.1).



- ☑ Output (map)
  - ☑ Unstructured Grid
  - □ waterlevel (s1)
  - □ waterdepth (waterdepth)
  - □ taucurrent in cell center (taus)
  - □ sea_water_speed (unorm)
  - □ eastward velocity on cell center (ucx)
  - □ northward velocity on cell center (ucy)
  - □ Salinity (sa1)
  - □ Chezy roughness (czs)
- ☑ Output (his)
  - □ Water level (waterlevel)
  - □ Waterdepth (Waterdepth)
  - □ Eastward component of cell center velocity vector. (x_velocity)
  - □ Northward component of cell center velocity vector. (y_velocity)
  - □ salinity (salinity)
  - □ cross_section_discharge
  - □ cross_section_discharge_int
  - □ cross_section_discharge_avg
  - □ cross_section_area
  - □ cross_section_area_avg
  - □ cross_section_velocity
  - □ cross_section_velocity_avg

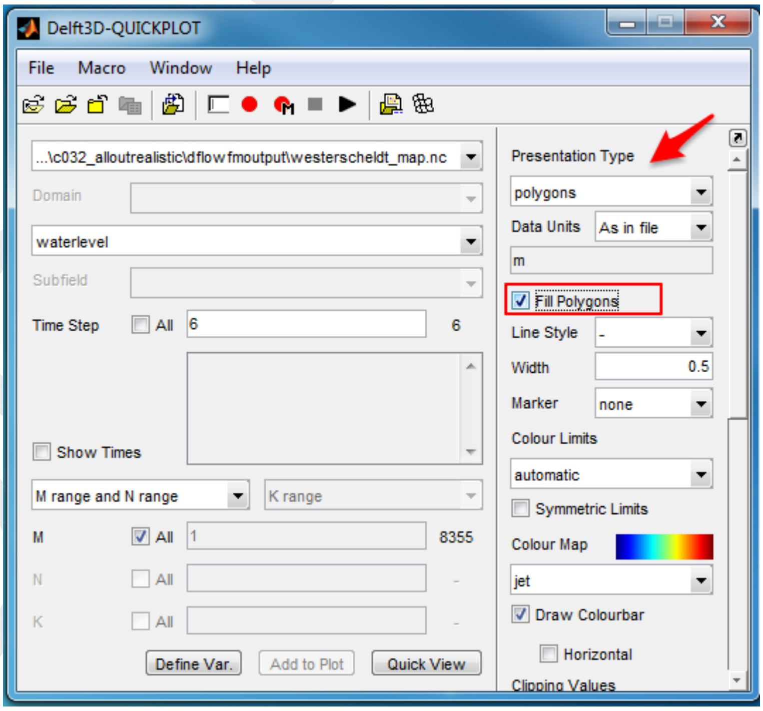
Figure 7.1: *Example of setting output (in)visible in the* Map *window*

The time navigator can be used to slide through the different time steps in the output files. If your model is relatively large, the drawing performance of the interface can be improved dramatically by enabling QuadTree visualizations in the properties window of the layer you want to visualize. To this end, select the layer you want to visualize in the ...

7.3 Visualization with Delft3D-QUICKPLOT

The interface of the Delft3D-QUICKPLOT allows to open the netCDF output files from a D-Flow FM simulation. In the interface you can select data fields, make a selection of time steps, stations or specify a preferred figure presentation options. After selection of a certain options you can visualize your data by using the “Quick View” button. To add another plot to an existing figure the “Add to Plot” button should be used.

When plotting the map output files from a D-Flow FM simulation, by default the presentation type “markers” will be selected. To smoothly visualize your results it is recommended to change presentation type into “polygons” and then select the option “Fill Polygons” (see the example on the Figure 7.2).



...\c032_alloutrealistic\dflowfmoutput\westerscheldt_map.nc

Domain

waterlevel

Subfield

Time Step ☐ All 6 6

☐ Show Times

M range and N range

K range

M ☑ All 1 8355

N ☐ All -

K ☐ All -

Presentation Type

polygons

Data Units As in file

m

☑ Fill Polygons

Line Style -

Width 0.5

Marker none

Colour Limits

automatic

☐ Symmetric Limits

Colour Map

jet

☑ Draw Colourbar

☐ Horizontal

Clipping Values

Figure 7.2: Useful map visualization options in the Delft3D-QUICKPLOT

See the Delft3D-QUICKPLOT User Manual for full details and a description of the routines and their use.

7.4 Visualization with Muppet

The interface of the Muppet allows to “Add Dataset” from a netCDF output file from a D-Flow FM simulation. It is possible to create a timeseries for a selected variable in from the history files. Additionally Muppet offers the option to visualize the map output files from a D-Flow FM simulation (see Figure 7.3).

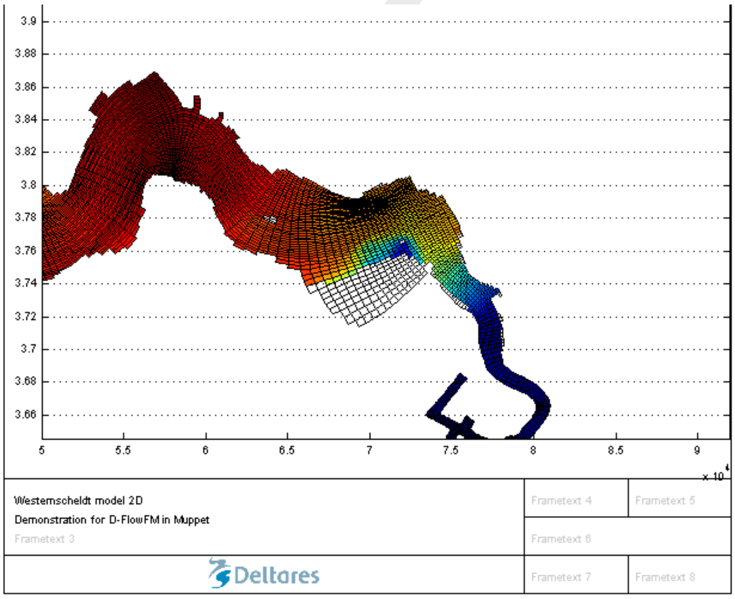


Figure 7.3: *Example of the Muppet visualization of the D-Flow FM map output file*

In the Muppet interface it is possible to specify figure settings, plot descriptions, labels and many more.

7.5 Visualization with Matlab

In the OpenEarthTools you can find a lot of Matlab scripts and functions that allow you to load, process and visualize your D-Flow FM output. For detailed guidance on getting started and using these tools effectively, visit https://publicwiki.deltares.nl/display/OET/MATLAB.

7.6 Visualization with Python

History-files can be easily visualized using *xarray* (https://docs.xarray.dev/en/stable/). Similarly, map-files can be visualized using *xugrid* (https://deltares.github.io/xugrid/). *Xugrid* is a UGRID wrapper around *xarray* that exposes many plotting options and processing options like coordinate transformation, re-indexing, rasterizing and regridding. If you want to do more advanced map-file visualization like horizontal and vertical (2DV cross section) slicing you can use *dfm_tools* (https://deltares.github.io/dfm_tools/), this package uses *xarray* and *xugrid* where possible and adds several extra features to read, process and visualize the history- and map-files of D-Flow FM. It can also works with multiple partitions and with older D-Flow FM mapformats.

8 Hydrodynamics

8.1 Introduction

Increasing awareness of environmental issues has focused the attention of scientists and engineers on the problem of predicting the flow and dispersion of contaminants in water systems. Reliable information on water flow, waves, water quality, sediment transport and morphology can be obtained from appropriate mathematical models. In general the first step in such modelling activities concerns the simulation of the flow itself. Whether the problem is related, for example, to the stability of a hydraulic structure, to salt intrusion, to the dispersion of pollutants or to the transport of sediment, flow simulations usually form the basis of the investigations to be carried out.

The *Delft3D Flexible Mesh Suite* is the integrated modelling system of Deltares for the aquatic environment. D-Flow Flexible Mesh, the flow module of this system, provides the hydrodynamic basis for other modules such as water quality, ecology, waves and morphology. For steady and non-steady modelling of the far-field water quality and ecology, it is coupled with the far-field water quality module D-Water Quality. For the interaction between waves and currents the flow module may be coupled with the short-waves model D-Waves. To control structures, the flow module is coupled to the D-Real Time Control module.

D-Flow FM is flexible by using an unstructured grid in the horizontal plane. In the vertical direction D-Flow FM offers two different vertical grid systems: a so-called σ co-ordinate system (σ-model) introduced by Phillips (1957) for ocean models and the Cartesian z-co-ordinate system (Z-model).

This section gives some background information on the conceptual model of the D-Flow FM module. Most of the concepts and algorithms are applicable to both the σ-model and Z-model.

8.2 General background

8.2.1 Range of applications of D-Flow FM

The hydrodynamic module D-Flow FM simulates two-dimensional (2DH, depth-averaged) or three-dimensional (3D) unsteady flow and transport phenomena resulting from tidal and/or meteorological forcing, including the effect of density differences due to a non-uniform temperature and salinity distribution (density-driven flow). The flow model can be used to predict the flow in shallow seas, coastal areas, estuaries, lagoons, rivers and lakes. It aims to model flow phenomena of which the horizontal length and time scales are significantly larger than the vertical scales.

If the fluid is vertically homogeneous, a depth-averaged approach is appropriate. D-Flow FM is able to run in two-dimensional mode (one computational layer), which corresponds to solving the depth-averaged equations. Examples in which the two-dimensional, depth-averaged flow equations can be applied are tidal waves, storm surges, tsunamis, harbor oscillations (seiches) and transport of pollutants in vertically well-mixed flow regimes.

Three-dimensional modelling is of particular interest in transport problems where the horizontal flow field shows significant variation in the vertical direction. This variation may be generated by wind forcing, bed stress, Coriolis force, bed topography or density differences. Examples are dispersion of waste or cooling water in lakes and coastal areas, upwelling and downwelling of nutrients, salt intrusion in estuaries, fresh water river discharges in bays and thermal stratification in lakes and seas.

8.2.2 Physical processes

The numerical hydrodynamic modelling system D-Flow FM solves the unsteady shallow water equations in two (depth-averaged) or in three dimensions. The system of equations consists of the horizontal equations of motion, the continuity equation, and the transport equations for conservative constituents. The equations are formulated in orthogonal curvilinear co-ordinates or in spherical co-ordinates on the globe. In D-Flow FM models with structured grid are considered as a simplified form of an unstructured grid. In Cartesian co-ordinates, the free surface level and bathymetry are related to a flat horizontal plane of reference, whereas in spherical co-ordinates the reference plane follows the Earth curvature.

The flow is forced by tide at the open boundaries, wind stress at the free surface, pressure gradients due to free surface gradients (barotropic) or density gradients (baroclinic). Source and sink terms are included in the equations to model the discharge and withdrawal of water.

The D-Flow FM model includes mathematical formulations that take into account the following physical phenomena:

- Free surface gradients (barotropic effects).
- The effect of the Earth rotation (Coriolis force).
- Water with variable density (equation of state).
- Horizontal density gradients in the pressure (baroclinic effects).
- Turbulence induced mass and momentum fluxes (turbulence closure models).
- Transport of salt, heat and other conservative constituents.
- Tidal forcing at the open boundaries.
- Space and time varying wind shear-stress at the water surface.
- Space varying shear-stress at the bed.
- Space and time varying atmospheric pressure on the water surface.
- Time varying sources and sinks (e.g. river discharges).
- Drying and flooding of tidal flats.
- Heat exchange through the free surface.
- Evaporation and precipitation.
- Tide generating forces.
- Effect of secondary flow on depth-averaged momentum equations.
- Lateral shear-stress at wall.
- Vertical exchange of momentum due to internal waves.
- Influence of waves on the bed shear-stress (2D and 3D).
- Wave induced stresses (radiation stress) and mass fluxes.
- Flow through hydraulic structures.
- Wind driven flows including tropical cyclone winds.

8.2.3 Assumptions underlying D-Flow FM

In D-Flow FM the 2D (depth-averaged) or 3D non-linear shallow water equations are solved. These equations are derived from the three dimensional Navier-Stokes equations for incompressible free surface flow. The following assumptions and approximations are applied:

- In the σ co-ordinate system the depth is assumed to be much smaller than the horizontal length scale. For such a small aspect ratio the shallow water assumption is valid, which means that the vertical momentum equation is reduced to the hydrostatic pressure relation. Thus, vertical accelerations are assumed to be small compared to the gravitational acceleration and are therefore not taken into account.
- The effect of variable density is only taken into account in the pressure term (Boussinesq approximation).
- In the σ co-ordinate system, the immediate effect of buoyancy on the vertical flow is

not considered. In D-Flow FM vertical density differences are taken into account in the horizontal pressure gradients and in the vertical turbulent exchange coefficients. So the application of D-Flow FM is restricted to mid-field and far-field dispersion simulations of discharged water.

- For a dynamic online coupling between morphological changes and flow the 2D sediment and morphology feature is available.
- In a Cartesian frame of reference, the effect of the Earth curvature is not taken into account. Furthermore, the Coriolis parameter is assumed to be uniform unless specifically specified otherwise.
- In spherical co-ordinates the inertial frequency depends on the latitude.
- At the bed a slip boundary condition is assumed, a quadratic bed stress formulation is applied.
- The formulation for the enhanced bed shear-stress due to the combination of waves and currents is based on a 2D flow field, generated from the velocity near the bed using a logarithmic approximation.
- The equations of D-Flow FM are capable of resolving the turbulent scales (large eddy simulation), but usually the hydrodynamic grids are too coarse to resolve the fluctuations. Therefore, the basic equations are Reynolds-averaged introducing so-called Reynolds stresses. These stresses are related to the Reynolds-averaged flow quantities by a turbulence closure model.
- In D-Flow FM the 3D turbulent eddies are bounded by the water depth. Their contribution to the vertical exchange of horizontal momentum and mass is modelled through a vertical eddy viscosity and eddy diffusivity coefficient (eddy viscosity concept). The coefficients are assumed to be proportional to a velocity scale and a length scale. The coefficients may be specified (constant) or computed by means of an algebraic, k-τ or k-ε turbulence model, where k is the turbulent kinetic energy, τ is the turbulent time scale and ε is the dissipation rate of turbulent kinetic energy.
- In agreement with the aspect ratio for shallow water flow, the production of turbulence is based on the vertical (and not the horizontal) gradients of the horizontal flow. In case of small-scale flow (partial slip along closed boundaries), the horizontal gradients are included in the production term.
- The boundary conditions for the turbulent kinetic energy and energy dissipation at the free surface and bed assume a logarithmic law of the wall (local equilibrium).
- The eddy viscosity is an-isotropic. The horizontal eddy viscosity and diffusivity coefficients should combine both the effect of the 3D turbulent eddies and the horizontal motions that cannot be resolved by the horizontal grid. The horizontal eddy viscosity is generally much larger than the vertical eddy viscosity.
- For large-scale flow simulations, the tangential shear-stress at lateral closed boundaries can be neglected (free slip). In case of small-scale flow partial slip is applied along closed boundaries.
- For large-scale flow simulations, the horizontal viscosity terms are reduced to a bi-harmonic operator along co-ordinate lines. In case of small-scale flow the complete Reynold's stress tensor is computed. The shear-stress at the side walls is calculated using a logarithmic law of the wall.
- In the σ co-ordinate system, D-Flow FM solves the so-called long wave equation.

  The pressure is hydrostatic and the model is not capable of resolving the scales of short waves. Therefore, the basic equations are averaged in analogy with turbulence introducing so called radiation stresses. These stresses are related to the wave quantities of Delft3D-WAVE.
- It is assumed that a velocity point is set dry when the actual water depth is below half of a user-defined threshold. If the point is set dry, then the velocity at that point is set to zero. The velocity point is set wet again when the local water depth is above the threshold. The hysteresis between drying and flooding is introduced to prevent drying and flooding in two consecutive time steps.

The drying and flooding procedure leads to a discontinuous movement of the closed boundaries at tidal flats.

- A continuity cell is set dry when all surrounding velocity points at the grid cell faces are dry or when the actual water depth at the cell centre is below zero (negative volume).
- The flux of matter through a closed wall and through the bed is zero.
- Without specification of a temperature model, the heat exchange through the free surface is zero. The heat loss through the bed is always zero.
- If the total heat flux through the water surface is computed using a temperature excess model the exchange coefficient is a function of temperature and wind speed and is determined according to Sweers (1976). The natural background temperature is assumed constant in space and may vary in time. In the more advanced heat flux formulation the fluxes due to solar radiation, atmospheric and back radiation, convection, and heat loss due to evaporation are modeled separately.
- The effect of precipitation on the water temperature is accounted for.

8.3 Hydrodynamic processes

D-Flow FM solves the Navier Stokes equations for an incompressible fluid, under the shallow water and the Boussinesq assumptions. In the vertical momentum equation the vertical accelerations are neglected, which leads to the hydrostatic pressure equation. In 3D models the vertical velocities are computed from the continuity equation. The set of partial differential equations in combination with an appropriate set of initial and boundary conditions is solved on an unstructured finite volume grid.

In the horizontal direction D-Flow FM uses orthogonal unstructured grids. Two coordinate references are supported:

1. Cartesian co-ordinates
2. Spherical co-ordinates

The boundaries of a river, an estuary or a coastal sea are in general curved and are not smoothly represented on a structured grid. The boundary becomes irregular and may introduce significant discretization errors. To reduce these errors unstructured grids are used. The unstructured grids also allow local grid refinement in areas with large horizontal gradients.

In the vertical direction D-Flow FM offers two different vertical grid systems: the σ coordinate system (σ-model) and the Cartesian z-co-ordinate system (Z-model). In the σ model, the vertical grid consists of layers bounded by two σ planes, which are not strictly horizontal but follow the bed topography and the free surface. Because the σ-model is boundary fitted both to the bed and to the moving free surface, a smooth representation of the topography is obtained. The number of layers over the entire horizontal computational area is constant, irrespective of the local water depth. The distribution of the relative layer thickness is usually non-uniform. This allows for more resolution in the zones of interest such as the near surface area (important for e.g. wind-driven flows, heat exchange with the atmosphere) and the near bed area (sediment transport). Please note that in D-Flow FM, unlike Delft3D-FLOW, the σ coordinate is equal to zero on the bed is 1 on the water surface.

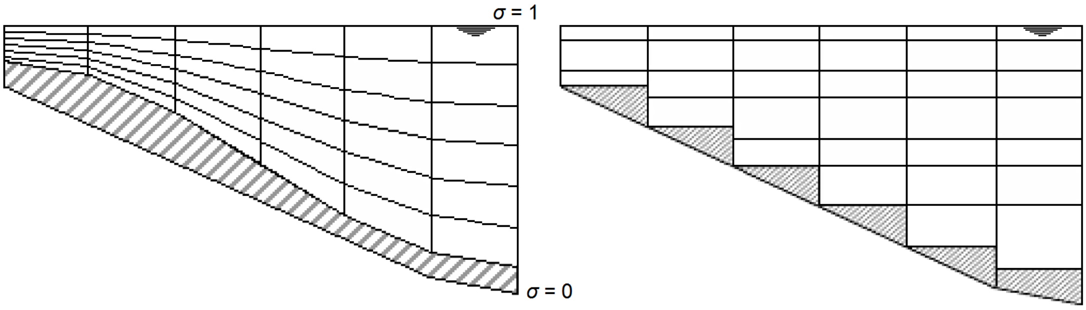


Figure 8.1: *Example of σ-model (left) and Z-model (right).*

Although the σ-grid is boundary fitted (in the vertical), it will not always have enough resolution around the pycnocline. The co-ordinate lines intersect the density interfaces that may give significant errors in the approximation of strictly horizontal density gradients (Leendertse, 1990; Stelling and Van Kester, 1994). Therefore, Z-model was introduced in D-Flow FM for 3D simulations of weakly forced stratified water systems. The Z-model has horizontal co-ordinate lines that are (nearly) parallel with density interfaces (isopycnals) in regions with steep bed slopes. This is important to reduce artificial mixing of scalar properties such as salinity and temperature. The Z-model is not boundary-fitted in the vertical. The bed (and free surface) is usually not a co-ordinate line and is represented as a staircase (zig-zag boundary).

8.3.1 Topological conventions

A computational cell in a D-Flow FM grid (sometimes referred to as a 'network') consists of corner nodes and edges connecting the corner nodes. Such a grid cell should contain at least three corner nodes and at most six corner nodes. The following topological conventions are used:

- netnodes: corners of a cell (triangles, quadrangles, ...),
- netlinks: edges of a cell, connecting netnodes,
- flownodes: the cell circumcentre, in case of triangles the exact intersection of the three perpendicular bisectors and hence also the centre of the circumscribing circle,
- flowlinks: a line segment connecting two flownodes.

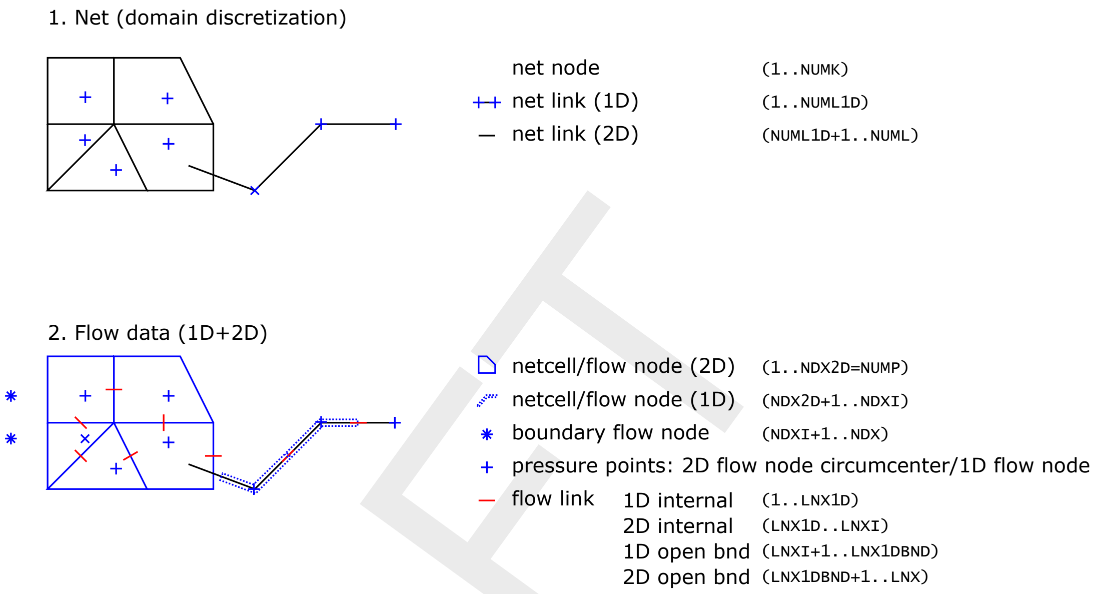

1. Net (domain discretization)
net node (1..NUMK)
net link (1D) (1..NUML1D)
net link (2D) (NUML1D+1..NUML)
2. Flow data (1D+2D)
netcell/flow node (2D) (1..NDX2D=NUMP)
netcell/flow node (1D) (NDX2D+1..NDXI)
boundary flow node (NDXI+1..NDX)
pressure points: 2D flow node circumcenter/1D flow node
flow link
1D internal (1..LNX1D)
2D internal (LNX1D..LNXI)
1D open bnd (LNXI+1..LNX1DBND)
2D open bnd (LNX1DBND+1..LNX)

Figure 8.2: *Flexible mesh topology*

This mesh topology is illustrated in Figure 8.2. The 'center' of a cell can be defined in multiple ways. To illustrate this, two conventional cell center definitions for a triangle are highlighted in Figure 8.3. The two displayed cell centers have different properties:

1 the circumcenter is the location within the triangle which is the center point of a circle that intersects the triangle at each corner node of the triangle; as a consequence, the orthogonal projection of the center point to each face of the triangle divides each face into two exactly equidistant pieces,

2 the mass center (or centroid) is the center of gravity; as a consequence, a line through a corner node and the mass center divides a face into two exactly equidistant pieces under an angle not necessarily equal to 90°.

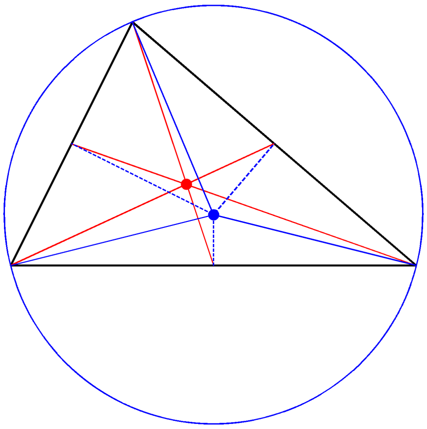


Figure 8.3: *Two conventional definitions of the cell center of a triangle: the circumcenter and the mass center.*

D-Flow FM utilizes the **circumcenter** as the basis of the definition of the elementary flow variables 'water level' and 'flow velocity'. The water level is defined at the circumcenter, whereas the face normal flow velocity is defined at the orthogonal projection of the circumcenter onto the cell face, i.e., the midpoint of the cell face.

Important properties of the mesh are the *orthogonality* and *smoothness*. The *orthogonality* is defined as the cosine of the angle φ between a flowlink and a netlink. Ideally 0, angle $\varphi = 90°$. The *smoothness* of a mesh is defined as the ratio of the areas of two adjacent cells. Ideally 1, the areas of the cells are equal to each other. A nearly ideal setup is shown in Figure 8.4.

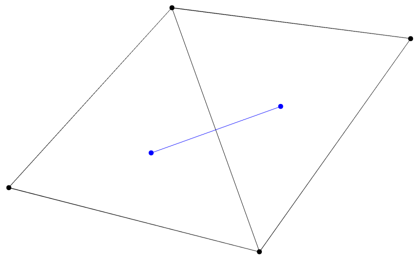
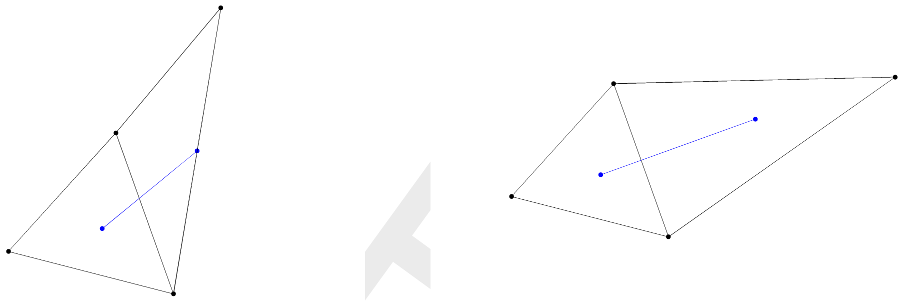


Figure 8.4: *Perfect orthogonality and nearly perfect smoothness along the edge connecting two triangles. Black lines/dots are network links/nodes, blue lines/dots are flow links/nodes.*

It is quite easy (and therefore dangerous) to generate meshes that violate the orthogonality and smoothness requirements. In Figure 8.5, two different setups of two gridcells are shown with different mesh properties.

The left picture of Figure 8.5 shows how orthogonality can be detoriated by skewing the right triangle with respect to the left triangle. While having the same area (perfect smoothness), the mutually oblique orientation results in poor orthogonality. In this particular case, the centre of the circumscribing circle is in principle located outside the right triangle. Such a triangle is denoted as an 'open' triangle, which is bad for computations.

The opposite is shown in the right picture of Figure 8.5 in which the right triangle has strongly been elongated, disturbing the smoothness property. However, the orthogonality is nearly perfect. Nonetheless, both meshes need to be improved to assure accurate model results.



(a) *Perfect smoothness, but poor orthogonality.*

(b) *Perfect orthogonality, but poor smoothness*

Figure 8.5: *Poor mesh properties due to violating either the smoothness or the orthogonality at the edge connecting two triangles. Black lines/dots are network links/nodes, blue lines/dots are flow links/nodes.*

8.3.2 Notation

The used symbols in this chapter are described in this section.

Time variables:

n Time index of the current time step.
Δt_n The computational time step size of time step n.

Computational grid and primary shallow water variables:

k Computational grid index for a pressure point ("flow node"),
j Computational grid index for a velocity point ("flow link"),
V_k Volume of water in grid cell k,
A_k Wet area of the horizontal water surface in grid cell k.
A_{u_j} Wet cross-sectional area ("flow area") of flow link j.
A_{1j}/A_{2j} Face-based cross-sectional area flow link j based on left or right side, respectively.

When a grid cell k is mentioned, this is essentially the same as a grid point (or calculation point) k. The term 'grid cell' better captures the concept of a cell-integrated value (such as volume), compared to a cell-averaged point value (such as water level).

Lateral discharges:

ℓ Index of a lateral discharge,
$\mathcal{L}_{\mathrm{lat},\ell}$ The set of grid cells selected for lateral discharge ℓ.
$Q_{\mathrm{lat},\ell,k}$ The discharge of lateral ℓ for grid cell k,
$Q_{\mathrm{lat},k}$ Total of all lateral discharges for grid cell k,
$Q_{\mathrm{in},k}$ Total of all source terms for grid cell k,

8.3.3 Conservation of mass and momentum

In this section, we will present in detail the governing equations for mass and momentum conservation. These equations are indicated by a continuity equation and momentum equations.

8.3.3.1 Continuity equation

D-Flow FM solves the depth-averaged continuity equation, derived by integration the continuity equation, for incompressible fluids ($\nabla \cdot \boldsymbol{u} = 0$) over the total depth, taken into account the kinematic boundary conditions at water surface and bed level, and is given by:

$$\frac{\partial h}{\partial t} + \frac{\partial Uh}{\partial x} + \frac{\partial Vh}{\partial y} = Q \tag{8.1}$$

with U and V the depth averaged velocities. Q is representing the contributions per unit area due to the discharge or withdrawal of water, precipitation and evaporation:

$$Q = \int_0^h (q_{in} - q_{out})\, dz + P - E \tag{8.2}$$

with q_{in} and q_{out} the local sources and sinks of water per unit of volume [1/s], respectively, P the non-local source term of precipitation and E non-local sink term due to evaporation. We remark that the intake of, for example, a power plant is a withdrawal of water and should be modelled as a sink. At the free surface there may be a source due to precipitation or a sink due to evaporation.

8.3.3.2 Momentum equations in horizontal direction

The momentum equations in x- and y-direction are given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} - fv = -\frac{1}{\rho_0}\frac{\partial P}{\partial x} + F_x + \frac{\partial}{\partial z}\left(\nu_V \frac{\partial u}{\partial z}\right) + M_x \tag{8.3}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} + fu = -\frac{1}{\rho_0}\frac{\partial P}{\partial y} + F_y + \frac{\partial}{\partial z}\left(\nu_V \frac{\partial v}{\partial z}\right) + M_y \tag{8.4}$$

Where ν_V is the vertical eddy viscosity coefficient. Density variations are neglected, except in the baroclinic pressure terms, $\partial P/\partial x$ and $\partial P/\partial y$ represent the pressure gradients.

The forces F_x and F_y in the momentum equations represent the unbalance of horizontal Reynolds stresses.

M_x and M_y represent the contributions due to external sources or sinks of momentum (external forces by hydraulic structures, discharge or withdrawal of water, wave stresses, etc.).

The effects of surface waves on the flow as modelled in D-Flow FM are described in **??**.

8.3.3.3 Vertical velocities

The vertical velocity w in the adapting σ co-ordinate system is computed from the continuity equation:

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} + \frac{\partial vh}{\partial y} + \frac{\partial w}{\partial z} = h\left(q_{in} - q_{out}\right) \tag{8.5}$$

At the surface the effect of precipitation and evaporation is taken into account. The vertical velocity w is defined at the iso σ-surfaces. w is the vertical velocity relative to the moving σ-plane. It may be interpreted as the velocity associated with up- or downwelling motions.

8.3.4 The hydrostatic pressure assumption

Under the shallow water assumption, the vertical momentum equation is reduced to a hydrostatic pressure equation. Vertical accelerations due to buoyancy effects and due to sudden variations in the bed topography are not taken into account. So:

$$\frac{\partial P}{\partial z} = -\rho g h \tag{8.6}$$

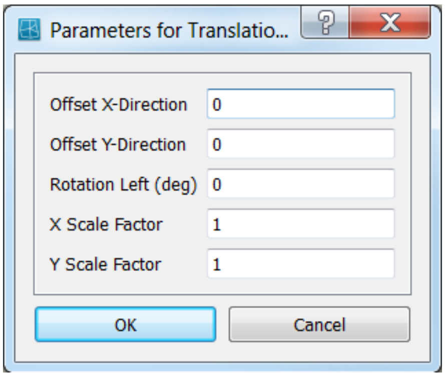
For water of constant density and taking into account the atmospheric pressure, it includes gradients of the free surface level, called barotropic pressure gradients. The atmospheric pressure is included in the system for storm surge simulations. The atmospheric pressure gradients dominate the external forcing at peak winds during storm events. Space and time varying wind and pressure fields are especially important when simulating storm surges.

In case of a non-uniform density the pressure gradients includes not only barotropic pressure gradient, but also vertical pressure gradient, the so called baroclinic pressure gradient. The baroclinic pressure gradient is the result of variable distribution of density and temperature in the vertical direction.

In the horizontal gradient a vertical derivative is introduced by the σ co-ordinate transformation. In estuaries and coastal seas the vertical grid may deteriorate strongly in case of steep bed slopes. In order to avoid artificial flow the numerical approximation of the baroclinic pressure terms requires a special numerical approach. The treatment of D-Flow FM to avoid the artificial mixing due to σ co-ordinates are discussed in section 8.5, see also Stelling and Van Kester (1994).

8.3.5 The Coriolis force

The Coriolis parameter f depends on the geographic latitude ϕ and the angular speed of rotation of the earth, Ω ($f = 2\Omega \sin\phi$). For a grid the user should specify the space varying Coriolis parameter, using a suitable projection. This can be done by selecting *Coordinate System* in RGFGRID, and selection of the option for *Spherical Coordinate*. The parameters for translation and rotation can be given as shown in Figure 8.6.



Parameters for Translatio...

Offset X-Direction: 0
Offset Y-Direction: 0
Rotation Left (deg): 0
X Scale Factor: 1
Y Scale Factor: 1

Figure 8.6: *Input for map projection for specifying Coriolis parameter on the grid.*

8.3.6 Diffusion of momentum

The forces F_x and F_y in the horizontal momentum equations represent the unbalance of horizontal Reynolds stresses. The Reynolds stresses are modelled using the eddy viscosity concept, (for details e.g. Rodi (1984)). This concept expresses the Reynolds stress component as the product between a flow as well as grid-dependent eddy viscosity coefficient and the corresponding components of the mean rate-of-deformation tensor. The meaning and the order of the eddy viscosity coefficients differ for 2D and 3D, for different horizontal and vertical turbulence length scales and fine or coarse grids. In general the eddy viscosity is a function of space and time.

For 3D shallow water flow the stress tensor is an-isotropic. The horizontal eddy viscosity coefficient, ν_H, is much larger than the vertical eddy viscosity ν_V ($\nu_H \gg \nu_V$). The horizontal viscosity coefficient may be a superposition of three parts:

1 a part due to "sub-grid scale turbulence",
2 a part due to "3D-turbulence" see Uittenbogaard *et al.* (1992) and
3 a part due to dispersion for depth-averaged simulations.

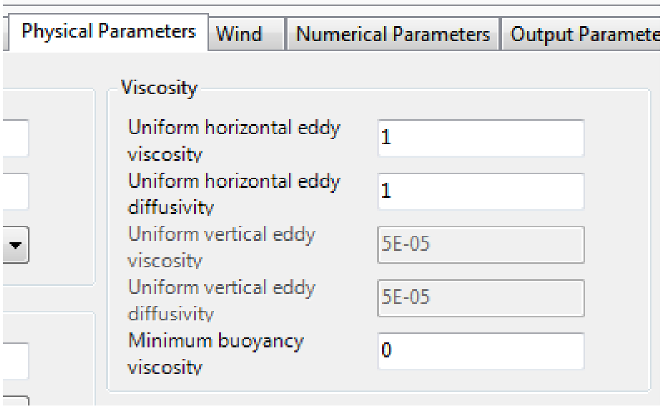
In simulations with the depth-averaged momentum and transport equations, the redistribution of momentum and matter due to the vertical variation of the horizontal velocity is denoted as dispersion. In 2D simulations this dispersion is not simulated as the vertical profile of the horizontal velocity is not resolved. Then this dispersive effect may be modelled as the product of a viscosity coefficient and a velocity gradient. The dispersion term may be estimated by the Elder formulation.

If the vertical profile of the horizontal velocity is not close to a logarithmic profile (e.g. due to stratification or due to forcing by wind) then a 3D-model for the transport of matter is recommended.

The horizontal eddy viscosity is mostly associated with the contribution of horizontal turbulent motions and forcing that are not resolved by the horizontal grid ("sub-grid scale turbulence") or by (a priori) the Reynolds-averaged shallow-water equations. in 3D, in the vertical direction, ν_V is referred to as the three-dimensional turbulence and in it is computed following a 3D-turbulence closure model.

Therefore, in addition to all turbulence closure models in D-Flow FM a constant (space and time) background mixing coefficient may be specified by the user, which is a background value for the vertical eddy viscosity in the momentum Equation (8.3) and Equation (8.4) consequently.

The horizontal and vertical eddy viscosities can be set by user defined value under `Physical Parameters` shown in Figure 8.7.



Viscosity

Uniform horizontal eddy viscosity: 1
Uniform horizontal eddy diffusivity: 1
Uniform vertical eddy viscosity: 5E-05
Uniform vertical eddy diffusivity: 5E-05
Minimum buoyancy viscosity: 0

Figure 8.7: *Input parameters for horizontal and vertical eddy viscosities.*

The default approach for the computation of the horizontal viscosity in D-Flow FM is the so-called Smagorinsky approach, which results into time and space varying horizontal viscosity coefficients. Due to the explicit time integration of the horizontal viscosity term a time step criterion is applied in order to ensure stable model results. If this criterion is violated, then the horizontal viscosity coefficient at a certain flow link is reduced. Next, in the diagnostic file a warning is generated as shown below.

```
    ** WARNING: Viscosity coefficient was limited for 3309 links.
    ** WARNING: Viscosity coefficient was limited for 3269 links.
    ** WARNING: Viscosity coefficient was limited for 3411 links.
```

Per time step is shown in how many flow links the horizontal viscosity coefficient has been reduced. After ten warnings this message is stopped, in order to prevent that the diagnostic file will become very large. The following message is written in the diagnostic file at the end of the simulation:

```
    ** INFO   : Viscosity coefficient/Horizontal transport flux were limited on some links in the course of computation.
```

8.3.7 Momentum equation for 1D networks

By default, D-Flow FM always uses the two-dimensional momentum equation in the horizontal plane. This has the advantage that the 1D and 2D flow solutions are consistent and head losses at bends are included implicitly. A validation of the bend losses using the experiments of Malone and Parr (2008) showed good agreement with measurements, but there could be situations in which one might like to deviate from this default approach, outside the range of validation. A keyword `Pure1D` has been added to the model settings which allows to enable a purely one-dimensional solution for the momentum equation.

Situations that may require such an approach are for example:

1 When converting from an existing 1D SOBEK/D-Flow 1D schematisation and desiring to obtain very similar results.
2 When the resolution of your model is too coarse to represent the change in curvature accurately (see Figure 8.8).
3 When running simulations that include morphodynamic evolution.

For the first case, the situation could be that you have an existing SOBEK model, which is already calibrated, and possibly already includes local friction losses. By converting this schematisation and using the standard `Pure1D = 0` option, you may be accounting for the same losses more than once. To prevent a recalibration, you could also opt for using the `Pure1D = 1` option.

Secondly, in certain situations the grid resolution may be too coarse to accurately capture the curvature. Figure 8.8 shows that the angle between two subsequent segments can be much larger when using a coarse resolution, than in the actual river branch geometry. In the real geometry the change is more gradual, and although some extra energy losses may be expected, they are not as large as when using the coarse grid setup. The choice of a coarse grid representation may therefore lead to mild curvature variations wrongfully being characterized as strong curvature variations, thereby giving unwanted extra water level setup.

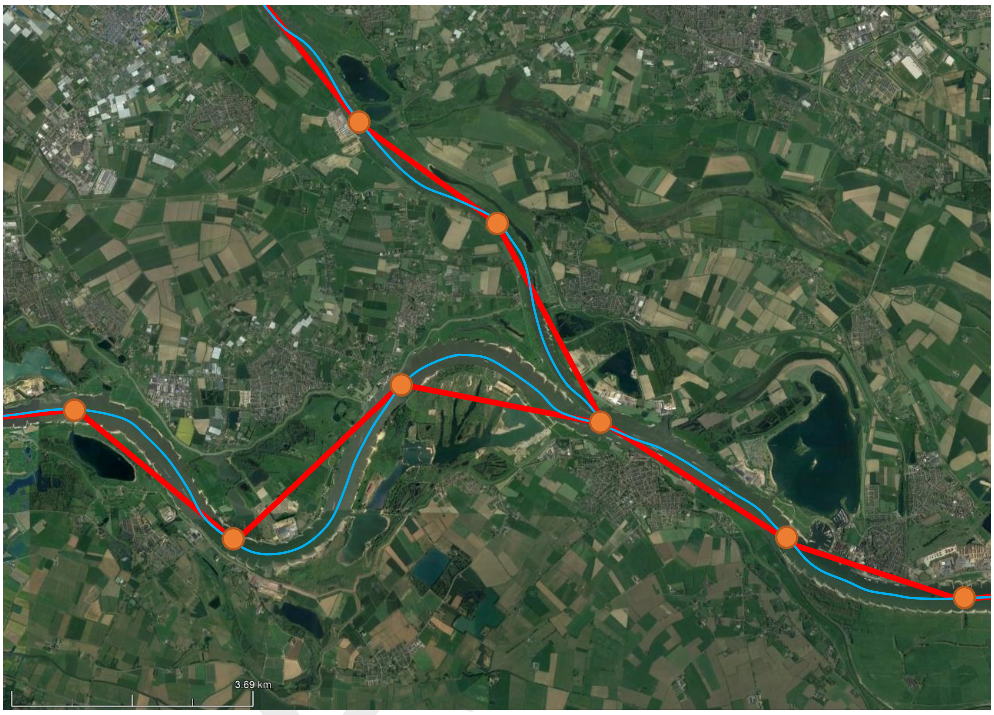


Figure 8.8: *Example of river network with gradual change in curvature along the river, and resulting coarse model representation with strong curvature changes.*

Lastly, when running simulations including a morphodynamic evolution of the bed, the `Pure1D = 1` approach is recommended. The reason for this is that the energy loss which occurs at the bend manifests itself in a single node. The local changes in the velocity are amplified in the sediment transport (typically the velocity raised to a certain power) which governs the sedimentation. The consequence when running simulations including morphology is that locally a spike in sedimentation occurs at the single node, which does not match the real behaviour.

The above `Pure1D` setting currently affects simple bends with one incoming and one outgoing channel at a node. Future versions will include additional functionality at channel junctions and confluences. See the settings description in Appendix A.

8.4 Hydrodynamics boundary conditions

In section 5.4.8, the boundary conditions are discussed from the viewpoint of the user interface. In the user interface, the user can specify the locations at which particular boundary conditions are to be imposed. Using section 5.4.8 as a backdrop, the present section discusses the underlying files and file formats and the way these are interpreted by the computational kernel. Three types of boundary conditions are discussed in this section, namely open boundaries (in section 8.4.1), vertical boundaries (in section 8.4.2) and closed boundaries (in section 8.4.3).

Lateral discharges are a special type of input that also act like boundary conditions, and are discussed in section 8.4.4.

8.4.1 Open boundary conditions

The proper prescription of an open boundary condition along a certain (part of the) rim of the grid can be achieved by considering four elements of the model:

1 the location where boundary are imposed, defined by either a polyline file (extension `.pli`), or for 1D networks the network end node id,
2 a boundary conditions file (extension `.bc`), containing the actual *forcing signals*, such as the time dependent information of the quantity under consideration and the physical nature of the quantity itself,
3 an external forcing file (extension `.ext`), in which the connection is laid between the location (polyline file or node id) and the boundary conditions file,
4 the name of the external forcing file in the master definition file (extension `.mdu`).

These four elements are subsequently discussed in the following.

8.4.1.1 The location of open boundaries

The flow engine needs the location for the boundary conditions. In 2D models, this is done by providing a polyline with support points (also see section 5.4.8.1). By default, these support points are a means to construct a series of virtual cell centers along the boundary rim of the grid. Figure 8.9 provides an image of the concept of virtual boundary cells in 2D.

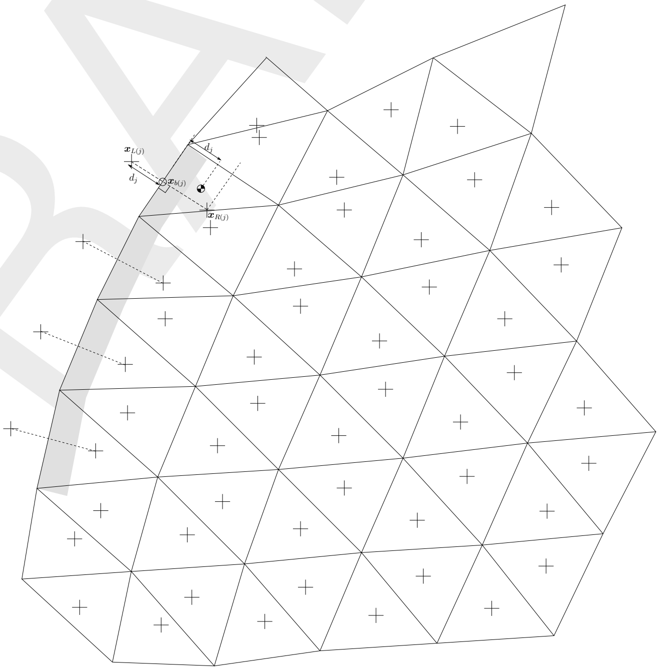


Figure 8.9: *Virtual boundary 'cells' near the shaded boundary; $\boldsymbol{x}_{Lj}$ is the virtual 'cell' pressure point near boundary edge j; $\boldsymbol{x}_{R(j)}$ is the inner cell's circumcenter; $\boldsymbol{x}_b(j)$ is the point on edge j that is nearest to the inner cell's circumcenter*

The support points are stored as one single polyline file per boundary condition, marking the rim along which the boundary conditions should hold. The polyline should be drawn in the vicinity of the rim. The user can specify the size of this 'vicinity' by means of the MDU-file keyword `OpenBoundaryTolerance`. The keyword specifies the search tolerance factor between the boundary polyline and the grid cells. The unit of this keyword is the cell size unit (i.e., not metres). By default, this value is 3, which loosely means that in the vicinity of $3\Delta x$ of the grid rim is searched of a boundary condition polyline.

In 1D, each boundary location can also be defined as a polyline in the vicinity of a network end point, but alternatively it can also be directly placed on that end point, by referring to its node id, using `nodeId` in the <.ext> file. A 1D boundary is obviously defined on a single point, and receives also one single virtual point. The precise details about the placement of this mirrored virtual boundary point follow in the text below.

The actual location of a specific boundary location point can be computed in two different ways, dependent on the user's choice for the keyword `izbndpos` in the mdu-file:

1 `izbndpos = 0`: construction of the boundary condition point by means of orthogonal mirroring of the closest cell center (default),
2 `izbndpos = 1`: construction of the boundary condition point as the orthogonal projection of the closest cell center onto the grid rim.

The following explanation uses both the mass center and the circumcenter, which are often different in 2D grid cells. For 1D gids, these two are identical.

Option `izbndpos = 0` 'mirrors' the internal cell closest to the boundary outward, to produce the virtual boundary point. The mirroring of the closest cell center is conducted as follows. First, the orthogonal distance from the cell's mass center to the actual rim of the grid is computed: d_j in Figure 8.9. Second, the cell circumcenter (pressure point) $\boldsymbol{x}_{L(j)}$ is projected orthogonally onto the grid cell outer edge. The virtual boundary cell's pressure point ($\boldsymbol{x}_{R(j)}$) is defined at a distance $\frac{1}{2}d_j$ starting at that projected point $\boldsymbol{x}_{b(j)}$. Sometimes the flat area of the cell, say A , at the rim can give rise to an adaptation of this distance. If $\frac{1}{2}\sqrt{A} > d_j$, then the center of the virtual cell is located at the a distance $\frac{1}{2}\sqrt{A}$ away from the grid.

The first part in Figure 8.10 shows how in a regular grid the mass center and circumcenter coincide, with an exactly mirrored virtual boundary point. Also, the resulting placement of the pressure points in the 2D and 1D example grid is identical.

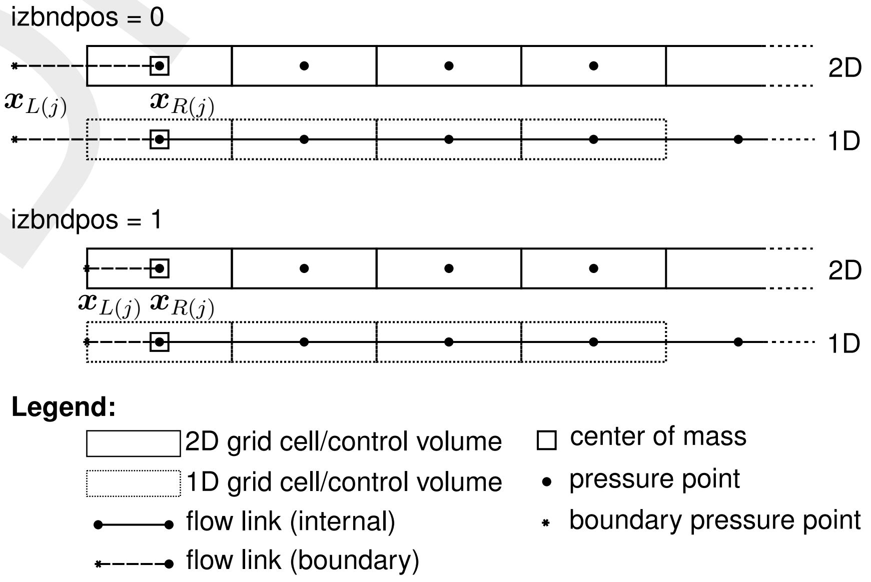


***Figure 8.10:** Position of virtual boundary point, depending on `izbndpos` setting.*

Option `izbndpos = 1` is mainly intended to force a water level boundary exactly on the outer edge of a 2D grid cell. The second part in Figure 8.10 shows how the pressure point now lies on that 2D edge. In 1D networks, the virtual pressure point is placed *half* an edge length outward (instead of a full edge length for `izbndpos = 0`). This ensures that it is possible to create a 2D gridded channel and an exactly equivalent 1D channel branch, with the same pressure point locations.

8.4.1.1.1 Geometry of open boundaries

The virtual boundary points as introduced above inherit some additional geometric parameters from their neighbouring internal grid point:

- the bed level of the boundary point is set equal to the bed level of the internal pressure point: $bl_{L(j)} = bl_{R(j)}$;
- the boundary flow link width is different for 1D and 2D boundaries:
  - 2D: The boundary flow link width is equal to the 2D edge width: $w_{u_j} = w'_{u_j}$
  - 1D: The boundary flow link j inherits the entire cross section geometry of the first neighbouring internal flow link j' (that is: width, area, perimeter). If no such cross section is available on link j', then the default uniform width `UniformWidth1D` is used.

Alternatively, for 1D boundaries, the bed level and flow link width can be customized per boundary. This can be useful at boundaries where the boundary water level is below the interior bed level, for example for an urban sewer outflow into open surface water. The following two options are available in the external forcings file (see also section 8.4.1.3):

- `bndBlDepth`: Custom bed level *depth below* initial water level for boundary point: $bl_{L(j)} = \zeta_{R(j)} - \texttt{bndBlDepth}$;
- `bndWidth1D`: Custom width for boundary flow link: $w_{u_j} = \texttt{bndWidth1D}$, and area and wetted perimeter are based on a rectangular profile with this width.

8.4.1.2 Forcing information

The <.bc>-file is used to store the actual boundary forcing signals. Multiple types of boundary conditions can be prescribed, configured via the `Quantity` keywords. The following subsections contain details for several types of flow boundaries.

The <.bc>-file may also contain the forcing signals for multiple boundary locations, configured via the `Name` keyword. When boundary locations have been specified by a polyline (`locationFile` in the <.ext> file), then the `Name` must refer to the polyline name, followed by the polyline point number, e.g., `Name = upstream_0001`. When instead a `nodeId` was used in the <.ext> file, the `Name` must be the same as that node id.

Water level

A water level signal is applied at the cell center of the virtual cell outside the grid (ghost cells or mirror cells). Water levels can be imposed as a timeseries or as a harmonic signal. In case of a harmonic signal, the period of the signal can be given as an astronomic component acronym.

If a timeseries is applied to a the first support point of a polyline named `arbitraryname`, prescribed in some polyline file, then the header of the `bc`-file is:

```
[forcing]
Name                  = arbitraryname_0001
Function              = timeseries
Time-interpolation    = linear
Quantity              = time
Unit                  = minutes since YYYY-MM-DD
```

```
Quantity               = waterlevelbnd
Unit                   = m
 [two-column data]
```

The data is to be inserted as a two-column array containing the time (in minutes) and the water level itself (in meters w.r.t. the reference level). In case of harmonic components, the header of the `bc`-file is:

```
[forcing]
Name                    = arbitraryname_0001
Function                = harmonic
Quantity                = harmonic component
Unit                    = minutes
Quantity                = waterlevelbnd amplitude
Unit                    = m
Quantity                = waterlevelbnd phase
Unit                    = degrees
 [three-column data]
```

The data is to be inserted as a three-column array containing the period (in minutes), the amplitude (in meters) and the phase (in degrees). If the period is T minutes, the amplitude is A meters and the phase is φ degrees, then the signal that is applied reads

$$h(t) = B + A\ \cos(2\pi t/T - \varphi). \tag{8.7}$$

The parameter B can be prescribed by an additional signal with a period specified equal to 0 minutes. As an example, the data series:

```
  0.0  0.5  0.0
745.0  2.0  0.0
```

represents the signal $h(t) = 0.5 + 2.0\ \cos(2\pi t/745)$, with the time t in minutes.

Discharge

A discharge boundary condition is applied at the face-center of the virtual boundary cell. For this, the face-normal velocity is used in combination with the face-center water depth. The face-based water depth in the evaluation of the flow area can optionally be set to a downwind approximation (for an inflowing discharge boundary) with the option `jbasqbnddownwindhs = 1` We remark that `jbasqbnddownwindhs = 0` is the default value.

Discharges can be imposed as a timeseries or as a harmonic signal. In case of a harmonic signal, the period of the signal can be given as an astronomic component acronym.

If a timeseries is applied to a the first support point of a polyline named `arbitraryname`, prescribed in some polyline file, then the header of the `bc`-file is:

```
[forcing]
Name                    = arbitraryname_0001
Function                = timeseries
Time-interpolation      = linear
Quantity                = time
Unit                    = minutes since YYYY-MM-DD
```

```
Quantity              = dischargebnd
Unit                  = m3/s
 [two-column data]
```

The data is to be inserted as a two-column array containing the time (in minutes) and the water level itself (in meters w.r.t. the reference level). In case of harmonic components, the header of the bc-file is:

```
[forcing]
Name                  = arbitraryname_0001
Function              = harmonic
Quantity              = harmonic component
Unit                  = minutes
Quantity              = dischargebnd amplitude
Unit                  = m3/s
Quantity              = dischargebnd phase
Unit                  = degrees
 [three-column data]
```

The data is to be inserted as a three-column array containing the period (in minutes), the amplitude (in cubic meters per second) and the phase (in degrees).

Velocity

A velocity boundary condition is applied at the face-center of the virtual boundary cell. Values provided are interpreted as face-normal velocities. Velocities can be imposed as a timeseries or as a harmonic signal. In case of a harmonic signal, the period of the signal can be given as an astronomic component acronym.

If a timeseries is applied to a the first support point of a polyline named arbitraryname, prescribed in some polyline file, then the header of the bc-file is:

```
[forcing]
Name                  = arbitraryname_0001
Function              = timeseries
Time-interpolation    = linear
Quantity              = time
Unit                  = minutes since YYYY-MM-DD
Quantity              = velocitybnd
Unit                  = m/s
 [two-column data]
```

The data is to be inserted as a two-column array containing the time (in minutes) and the water level itself (in meters w.r.t. the reference level). In case of harmonic components, the header of the bc-file is:

```
[forcing]
Name                  = arbitraryname_0001
Function              = harmonic
Quantity              = harmonic component
Unit                  = minutes
Quantity              = velocitybnd amplitude
Unit                  = m/s
Quantity              = velocitybnd phase
Unit                  = degrees
 [three-column data]
```

The data is to be inserted as a three-column array containing the period (in minutes), the amplitude (in meters per second) and the phase (in degrees).

Water level gradient

Besides the option to prescribe actual water levels as a boundary condition, D-Flow FM facilitates the prescription of water level *gradients*. Such a Neumann-type boundary condition for the water level can be assigned through the keyword neumannbnd. The value of the water level gradient is applied at the face-center.

Water level gradients can be imposed as a timeseries or as a harmonic signal. In case of a harmonic signal, the period of the signal can be given as an astronomic component acronym. If a timeseries is applied to a the first support point of a polyline named arbitraryname, prescribed in some polyline file, then the header of the bc-file is:

```
[forcing]
Name                   = arbitraryname_0001
Function               = timeseries
Time-interpolation     = linear
Quantity               = time
Unit                   = minutes since YYYY-MM-DD
Quantity               = neumannbnd
Unit                   = -
 [two-column data]
```

The data is to be inserted as a two-column array containing the time (in minutes) and the water level itself (in meters w.r.t. the reference level). In case of harmonic components, the header of the bc-file is:

```
[forcing]
Name                   = arbitraryname_0001
Function               = harmonic
Quantity               = harmonic component
Unit                   = minutes
Quantity               = neumannbnd amplitude
Unit                   = -
Quantity               = neumannbnd phase
Unit                   = degrees
 [three-column data]
```

The data is to be inserted as a three-column array containing the period (in minutes), the amplitude (in meters per second) and the phase (in degrees). The water level gradient is interpreted positive in the outward-normal direction.

Riemann invariant

At a Riemann boundary we do not allow any outgoing perturbation with respect to some reference boundary state to reflect back from the boundary. This is achieved by prescribing the incoming Riemann invariant. Using the convention of a positive inward normal at the boundary, this can be put as $u_b + 2\sqrt{gH_b}$, with u_b the velocity at the boundary and H_b the total water depth at the boundary. In this expression, we take the boundary values u_b and H_b as the reference boundary state. While applying Riemann boundaries, directional effects are disregarded.

Using linearization and the assumption of a flow field that is initially at rest, the Riemann boundary is rewritten such that is takes the form:

$$\zeta_b = 2\zeta_i - u\sqrt{\frac{H}{g}} - \zeta_0 \tag{8.8}$$

with

ζ_b the surface level elevation at the boundary,
ζ_i the water level of the perpendicularly incoming wave at the boundary,
H the total water depth,
g the gravitational acceleration,
u the flow velocity (positive inward) and
ζ_0 the initial water level elevation.

The user specifies as boundary signal ζ_i, the water level of the perpendicularly incoming wave at the boundary. Formulated in this way, the incoming wave height is independent of local bathymetry.

Following linear wave theory, one finds $u\sqrt{\frac{H}{g}}$ for a propagating wave. One will obtain a water level at the boundary of when there are no waves travelling from inside the model back to the boundary.

Example

As an example, a simple uniform channel with two Riemann boundaries, with 1 m amplitude on the left and 0 m on the right will show a 1 m amplitude wave travelling from left to right. At the right side, the wave passes the boundary with minor reflection only. Practically no waves travel from right to left since at the right boundary no incoming wave amplitude was specified.

The water level of the incoming wave for the Riemann boundary can be prescribed as a timeseries or as a harmonic signal. In case of a harmonic signal, the period of the signal can be given as an astronomic component, (e.g. M_2), or as a period in minutes, amplitude in m and phase in degrees. If a timeseries is applied to the first support point of a polyline named *arbitraryname*, prescribed in some polyline file, then the header of the <bc>-file read:

```
[forcing]
Name                    = arbitraryname_0001
Function                = timeseries
Time-interpolation      = linear
Quantity                = time
Unit                    = minutes since YYYY-MM-DD
Quantity                = riemannbnd
Unit                    = m
 [two-column data]
```

The data is to be inserted as a two-column array containing the time (in minutes) and the water level for Riemann boundary (in meters). In case of harmonic components, the header of the `bc`-file is:

```
[forcing]
Name                    = arbitraryname_0001
Function                = harmonic
Quantity                = harmonic component
```

```
Unit                    = minutes
Quantity                = riemannbnd amplitude
Unit                    = m
Quantity                = riemannbnd phase
Unit                    = degrees
 [three-column data]
```

The data is to be inserted as a three-column array containing the period (in minutes), the amplitude (in meters per second) and the phase (in degrees).

Suppose, as an example, that we have:

- ◇ a flow domain with a bathymetry at a bed level equal to 0 m w.r.t. the reference level,
- ◇ an initial water level equal to 10 m w.r.t. the reference level,
- ◇ a local initial disturbance of the initial water level,
- ◇ an initial flow field at rest and
- ◇ that we aim to prevent reflections at the boundary.

In that case, it follows from Equation (8.8) that $\zeta_b = 10$ m w.r.t. the reference level is to be prescribed. Remark that this application only holds for small disturbances of ζ from ζ_b.

Discharge-water level dependency

If a relation between the discharge and the local water level is known on beforehand, then this relation can be provided to the flow model as a table by means of the keyword `qhtable`. This table, provides the water level ζ as a function of the computed discharge Q along the entire polyline defining the boundary. Worded differently, only one relation is given for each polyline. Note that this is opposite to other boundary types in which, for instance, a time series is given for each support point of a polyline.

If a Qh-table is applied to a polyline named `arbitraryname`, prescribed in some polyline file, then the header of the `bc`-file is:

```
[forcing]
Name                    = arbitraryname
Function                = qhtable
Quantity                = qhbnd discharge
Unit                    = m3/s
Quantity                = qhbnd waterlevel
Unit                    = m
 [two-column data]
```

The data is to be inserted as a two-column array containing the discharge (in cubic meters per second) and the water *level* (in meters w.r.t. the reference level).

8.4.1.3 External forcings file

The external forcings file <.ext> collects all boundary condition information. The file format for boundaries is described in section C.5.2.1. It contains one or more `[Boundary]` blocks, one for each boundary condition. Each block must specify the boundary type via the `quantity` keyword. The previous section already listed some boundary quantity names. Next, each block must specify the boundary location specification. This is either a polyline file name, via the `locationFile` keyword, or a 1D network node id, via the `nodeId` keyword. Finally, the actual forcing data file must be specified, via the `forcingFile` keyword. Section 8.4.1.5 contains an example of this.

8.4.1.4 External forcings in model definition file

The final step in specifying boundary conditions, is referring to the <.ext> file in the <.mdu> file. This is done via the keyword ExtForceFileNew under the MDU block [external forcing].

8.4.1.5 Example

Recalling the four elementary actions from the beginning of section 8.4.1, the administration in files can be illustrated by means of an example. Suppose, we have a channel that is geometrically modelled by means of the grid shown in Figure 8.11. The domain is 10000 m long and 500 m wide. The bottom left corner coincides with the origin of the geometrical frame of reference.

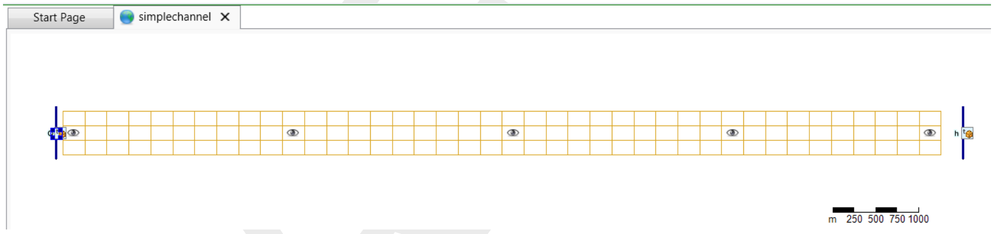


Figure 8.11: *User Interface view of a simple channel covered by a straightforward Cartesian grid. Boundary conditions are prescribed at the left hand side and the right hand size of the domain.*

The domain shown in Figure 8.11 contains two boundaries: on the left hand, a discharge boundary condition is present to present flow entering the domain, on the right hand, a water level boundary condition is set. Having two boundaries, we have two polyline files, in this example: left.pli and right.pli.

The contents of left.pli are:

```
boundaryleft
    2     2
    -80    -50
    -80    550
```

whereas the contents of right.pli are:

```
boundaryright
    2     2
  10250     -50
  10250     550
```

Notice that both polyline files contain the name of the polyline, namely boundaryleft and boundaryright, respectively. In this example, both the polylines contain *two* support points. For each of these support points, timeseries can be prescribed. In this example, we restrict ourselves to homogeneous boundary conditions. This means that we have to prescribe physical information for the first support points, i.e. for boundaryleft_0001 and boundaryright_0001.

The physical information should be provided in the bc-file. In this example, we use the following bc-file:

```
[forcing]
Name                 = boundaryleft_0001
Function             = timeseries
Time-interpolation   = linear
Quantity             = time
Unit                 = minutes since 2001-01-01
Quantity             = dischargebnd
Unit                 = m3/s
  0.000000    2500.0
120.000000    3000.0
240.000000    2500.0
360.000000    2000.0
480.000000    2500.0
600.000000    3000.0
720.000000    2500.0
840.000000    2000.0
960.000000    2500.0
1080.000000    3000.0
1200.000000    2500.0
1320.000000    2000.0
1440.000000    2500.0

[forcing]
Name                 = boundaryright_0001
Function             = timeseries
Time-interpolation   = linear
Quantity             = time
Unit                 = minutes since 2001-01-01
Quantity             = waterlevelbnd
Unit                 = m
  0.000000    2.50
1440.000000    2.50
```

This file couples the timeseries for the discharge to the support point named boundaryleft_0001. Likewise, the water level boundary, being constant in time, is coupled to the support point named boundaryright_0001.

The final specification of the boundary conditions is wrapped up in the external forcing file, with extension .ext. In this file, the connection is laid between the quantity of the boundary condition, the name of the polyline file (in which the name of the polyline itself is given) and the forcing file (in which the physical information is provided). In our example, the file simplechannel.ext has the following contents:

```
[boundary]
quantity     = dischargebnd
locationfile = left.pli
forcingfile  = simplechannel.bc

[boundary]
quantity     = waterlevelbnd
locationfile = right.pli
forcingfile  = simplechannel.bc
```

The final step is to let the model know that the external forcings file simplechannel.ext is the one that should be used. This is to be achieved in the MDU-file:

```
[external forcing]
ExtForceFile                        =
ExtForceFileNew                     = simplechannel.ext
```

Notice that the name is specified at the keyword `ExtForceFileNew`. The other keyword, `ExtForceFile`, should be kept empty, unless *deprecated* `.tim`-files and/or `.cmp`-files are used to prescribe the physical information for the boundaries.

8.4.1.6 Miscellaneous

In the previous sections, the most essential information on the application of boundary conditions is described. Some remaining aspects are discussed in this section.

Artificial boundary layers

In general, the velocity at the boundary is prescribed as having only a component normal to the boundary (`cstbnd = 0`). This could lead to an artificial boundary layer along the boundary at offshore boundaries. To resolve this behaviour, the advection terms containing gradients perpendicular to the boundary can be switched off, by setting the keyword `cstbnd = 1` in the `[numerics]` block in the MDU-file. By default this keyword `cstbnd = 0`.

Smoothing parameter boundary conditions

The solution of the shallow water equations is uniquely determined by a set of initial and boundary conditions. The boundary conditions represent the external forcing and determine the steady state solution. The deviation between the initial condition and the steady state solution generates a transient (mass spring system analogy).

In D-Flow FM, the initial conditions for the water level and velocities are obtained from:

◇ The results of a previous run (warm start).
◇ User-prescribed (space varying or uniform) input fields (cold start).

The initial values are usually inconsistent with the boundary conditions at the start time of the simulation. This will generate a transient solution consisting of waves with eigen frequencies of the model domain. These waves may be reflected at the boundaries and generate a standing wave system. The waves should be dissipated completely by bottom friction and viscosity terms or leave the domain through the open boundaries. The damping of the transient solution determines the spin-up time of the numerical model.

To reduce the amplitude of the transient wave and the spin-up time of a model, D-Flow FM has an option to switch on the boundary forcing gradually by use of a smoothing period (parameter T_{smo}). This can by specifying the total number of seconds required for the transition using the keyword `Tlfsmo` in the `[numerics]` block in the MDU-file. With $F_i(t)$ the initial value at the boundary, $F_b(t)$ the boundary condition signal and $F_b^{smo}(t)$ the boundary condition after smoothing, the boundary forcing is given by:

$$F_b^{smo}(t) = \alpha F_i(t) + (1-\alpha)F_b(t), \tag{8.9}$$

with:

$$\alpha = \frac{T_{smo} - t}{T_{smo}} \tag{8.10}$$

if $t < T_{smo}$. In case $t \geq T_{smo}$, then the smoothing parameter is set to zero: $\alpha = 0$.

Smoothing is possible both for a warm and a cold start. If the initial conditions are consistent with the boundary conditions at the start time of the simulation then the smoothing time should be set to zero. When restarting a simulation (warm start), the smoothing time should be reduced by the difference between the original smoothing time (i.e., in the cold start) and the restart time, and it shoulf be set to zero if the restart time is larger than the original smoothing time. This is done using `[time]` keyword `TStartTlfsmo`, which is the start time for applying the smoothing time with respect to the reference time. By default it is set to the start time of the simulation (i.e., smoothing is applied since the beginning), and it should be set to the start time of the original simulation when restarting.

Secondary boundary conditions

In section 8.4.1.2, several types of boundary conditions are discussed. In addition, two types of boundary conditions can be applied on top of the canonical types: *normal* velocities and *tangential* velocities. The associated keyword are `normalvelocitybnd` and `tangentialvelocitybnd`, respectively.

Bed level at open boundaries

For some evaluations, the solver requires the bed level at the open boundary. This value depends on the bed level in the ghost cell outside the model domain. By default, the bed level in the ghost cell is mirrored from the internal grid cell next to the boundary, which means that a zero bed slope is assumed. While this is perfectly acceptable for most real-life applications, it hinders obtaining perfect normal-flow solutions in idealized models when using `DpuOpt` = 2, i.e. the mean bed level imposed at the cell interfaces. For these specific cases, an option has been implemented to extrapolate the bed level outside the domain. When setting `[geometry]` keyword `ExtrBl` = 1, the bed levels at the ghost cells are extrapolated in such a way that the flow depth at velocity points can be exact in idealized cases.[1] This option is only available when the bed level is specified at the cell centres (`BedlevType` = 1).

Furthermore, for obtaining perfect normal flow, the downstream water level needs to be specified at the ghost cell (`izbndpos` = 0) and the conveyance needs to be computed based on a uniform bed level based on the surrounding net nodes (`Conveyance2D` = -1). Note that the boundary condition must be manually shifted by $s\Delta x$, being s the slope, from the exact value expected at the domain boundary (cell edge).[2]

Suppose a case in which a specific discharge q of 1 m^2/s is fed into a rectangular channel of constant width and constant slope s equal to -7.1356×10^{-4}. Wall friction is neglected. Friction is using Chézy with parameter C equal 3.74357×10^{1} m$^{1/2}$/s, such that the normal flow depth, given by $h = (q^2/C^2/s)^{1/3}$, becomes 1 m. A grid with streamwise space step Δx of 20 m is employed. The bed level at the dowstream end of the domain (the cell edge) is −1 mAD, such that the water level boundary condition for obtaining perfect normal flow is 0 mAD at the cell edge. This condition is obtained by setting the water level at the ghost cell equal to $s\Delta x$ = −0.01427 (`izbndpos` = 0).

Figure 8.12 shows the cell-centre velocity of the idealized case described above when using and not using the bed level extrapolation. Figure 8.13 shows the flow depth at velocity points and Figure 8.14 the flow depth at cell centres. When extrapolating the bed level, the flow depth at velocity points and the velocity at cell centres (and velocity points) are equal to the desired normal-flow case values. However, the flow depth at cell centres is smaller. This is because the D-Flow FM numerical scheme uses at the cell interfaces the upwind water level.

[1] The bed levels are extrapolated in line with the internal slope at water level boundaries, but with the reverse slope at velocity/discharge boundaries since the model uses the water level of the internal grid cell at the open boundary in the latter case.

[2] One would expect that the boundary condition should only be reduced by half that value, but the water level at the cell interfaces is determined by the value of the upwind cell centre. The water level to be specified at the ghost cell should be consistent with that trend.

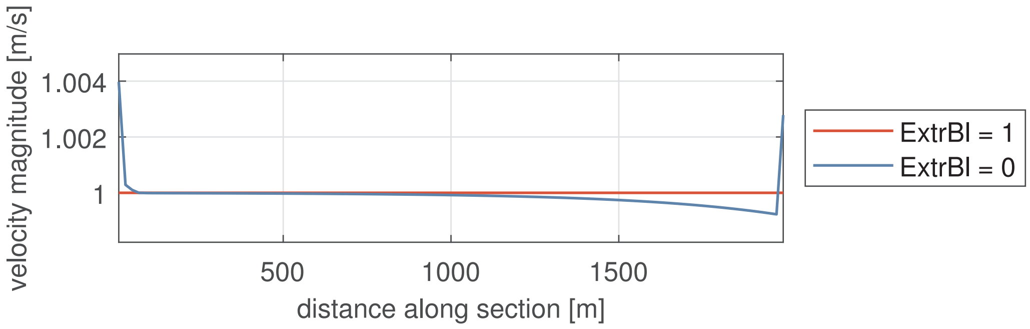


Figure 8.12: *Cell-centre velocity magnitude along the streamwise direction of an idealized normal-flow case.* `ExtrBl` *= 1 extrapolates the bed levels at the boundaries and perfect normal flow is obtained.*

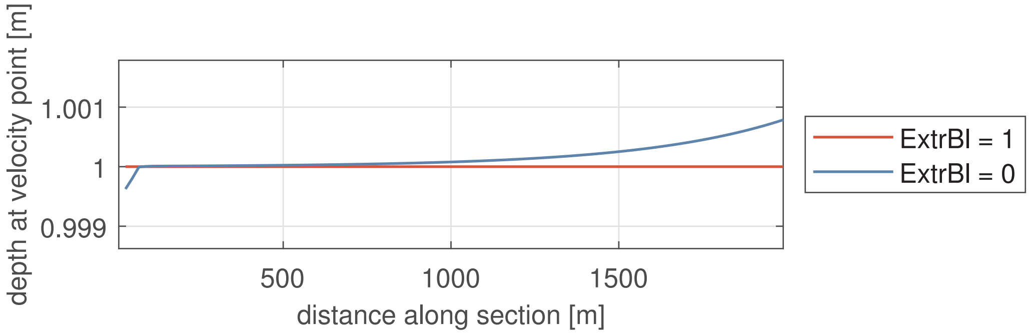


Figure 8.13: *Flow depth at velocity points along the streamwise direction of an idealized normal-flow case.* `ExtrBl` *= 1 extrapolates the bed levels at the boundaries and perfect normal flow is obtained.*

If the water depth at the interface matches the normal-flow water depth, then the water depth at the upstream cell centre will be smaller since the same water level is combined with a higher cell-centre bed level.

8.4.2 Vertical boundary conditions

The vertical boundary conditions close the system of shallow water equations at the bed level and the free surface level. Without precipitation, evaporation or ground water flow, there is by definition no flow through the bottom or top surface of the water column. The vertical boundary conditions for the continuity equation 8.1 are most easily expressed when the σ co-ordinate is used for the vertical direction. In that case the free surface ($\sigma = 1$, or $z = \zeta$) and the bottom ($\sigma = 0$, or $z = z_b$) are σ co-ordinate surfaces. ω is the vertical velocity relative to the σ-plane. The impermeability of the surface and the bottom is taken into account by prescribing the following kinematic conditions:

$$\omega|_{\sigma=1} = 0 \text{ and } \omega|_{\sigma=0} = 0 \tag{8.11}$$

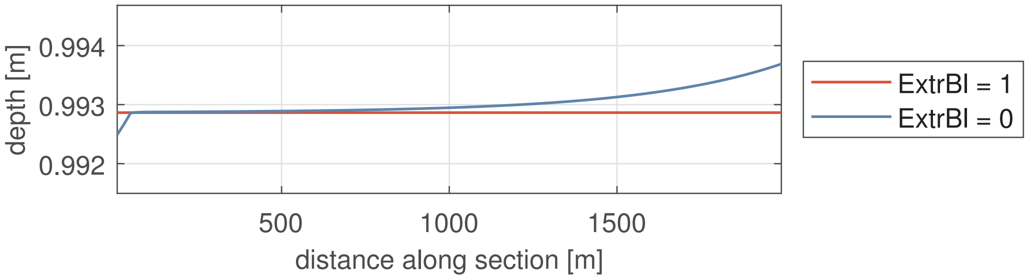


Figure 8.14: *Cell-centre flow depth along the streamwise direction of an idealized normal-flow case.* `ExtrBl` *= 1 extrapolates the bed levels at the boundaries and perfect normal flow is obtained.*

8.4.2.1 Bed friction and conveyance

The boundary conditions for the momentum equations 8.3 and 8.4 at the bed ($z = z_b$) are:

$$\left.\frac{\nu_V}{H}\frac{\partial u}{\partial \sigma}\right|_{\sigma=0} = \frac{1}{\rho_0}\tau_{bx} \tag{8.12}$$

and

$$\left.\frac{\nu_V}{H}\frac{\partial v}{\partial \sigma}\right|_{\sigma=0} = \frac{1}{\rho_0}\tau_{by} \tag{8.13}$$

with τ_{bx} and τ_{by} the components of the bed stress in x- and y-direction, respectively. These equations hold in 2D and 3D parts of the model, in 1D parts the equation reduces to a single component along the channel.

For 2D depth-averaged flow the shear-stress at the bed induced by a turbulent flow is assumed to be given by a quadratic friction law:

$$\boldsymbol{\tau}_b = \frac{\rho_0 g \boldsymbol{U}\left|\boldsymbol{U}\right|}{C_{2D}^2}, \tag{8.14}$$

where $|\boldsymbol{U}|$ is the magnitude of the depth-averaged horizontal velocity. In D-Flow FM we support the following equations to compute the C_{2D} coefficient. Since these equations can also be applied in 1D, we are using the 1D formulation based on hydraulic radius R which is typically equivalent to the water depth H in 2D. The numbers and strings following the name of the roughness formulation indicate the number and name by which the formulations are selected in the input files.

◇ **Chézy formulation**: 0, `Chezy`

$$C_{2D} = \text{Chézy coefficient [m}^{1/2}\text{/s]}. \tag{8.15}$$

◇ **Manning's formulation**: 1, `Manning`
As formulated by Manning (1891), and also known as Gauckler-Manning-Strickler formulation.

$$C_{2D} = \frac{\sqrt[6]{R}}{n} \tag{8.16}$$

where:

R is the hydraulic radius
n is the Manning coefficient [m$^{-1/3}$s].

- **Log law of the Wall**: 2, `WallLawNikuradse`

$$C_{2D} = \frac{\sqrt{g}}{\kappa} \ln \left(1 + \frac{30R}{ek_n} \right) \tag{8.17}$$

where:

R is the hydraulic radius
k_n is the Nikuradse roughness length [m].

- **White-Colebrook's formulation**: 3, `WhiteColebrook`
  As formulated by Colebrook and White (1937) and Colebrook (1939).

$$C_{2D} = 18\ {}^{10}\!\log \left(\frac{12R}{k_n} \right) \tag{8.18}$$

where:

R is the hydraulic radius
k_n is the Nikuradse roughness length [m].

- **Strickler's formulation based on Nikuradse**: 7, `StricklerNikuradse`
  As formulated by Strickler (1923).

$$C_{2D} = 25 \sqrt[6]{\frac{R}{k_n}} \tag{8.19}$$

where:

R is the hydraulic radius
k_n is the Nikuradse roughness length [m].

- **Strickler's formulation**: 8, `Strickler`
  As formulated by Strickler (1923).

$$C_{2D} = k_s \sqrt[6]{R} \tag{8.20}$$

where:

R is the hydraulic radius
k_s is the Strickler roughness coefficient [m$^{1/3}$/s].

- **de Bos & Bijkerk's formulation**: 9, `deBosBijkerk`
  As formulated by De Bos and Bijkerk (1963).

$$C_{2D} = \gamma \sqrt[3]{H} \sqrt[6]{R} \tag{8.21}$$

where:

H is the total water depth
R is the hydraulic radius
γ is the De Bos & Bijkerk coefficient [1/s].

Alluvial bed properties and land use data can be used to determine the roughness and flow resistance. Specific functionality has been implemented to make the translation of bed and land use properties into roughness and flow resistance characteristics. See section 15.2 for details.

8.4.2.2 Roughness definition input

The roughness definition (or: friction values) is different for 1D network and 2D grid parts:

- ◇ 1D network parts are typically defined using branch locations and cross section definitions. Therefore, the description of this is placed in section 8.7.6.
- ◇ 2D grid parts may use a global friction value, specified in the MDU file (Appendix A) under `[physics]`, keyword `UnifFrictCoef`. Note that the available `UnifFrictType` values are limited to 0, 1, 2, 3.
- ◇ Alternatively, 2D grid parts may have spatially varying friction values defined via the parameter field file, specified in the MDU file under `[geometry]`, keyword `IniFieldFile`. The spatially varying friction values have to be provided in a separate data file, and interpolation of that data onto the 2D grid is controlled by several parameters in the `IniFieldFile`. All details are given in section D.2.

8.4.2.3 Conveyance in 2D

Bed friction often plays a major role in the discharge capacity and expected maximum water levels of channels and gullies. If we model a trapezoidal channel with sloping sidewalls on a grid with 6 grid cells across the width of the channel, a typical cross section using the standard bed representation with uniform depth appears per cell (Figure 8.15a). In D-Flow FM, we allow for representation of a locally sloping bed as shown in Figure 8.15b.

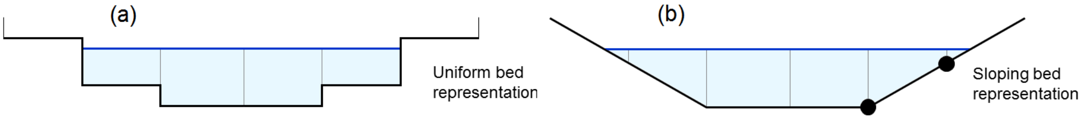


Figure 8.15: *Bed representation with uniform depth levels (a), and locally sloping bed (b).*

The most left and the most right cell are not yet wet in the uniform bed representation. In the sloping bed representation, these outer cells are partly wet, yielding a more accurate description of the total wet cross sectional area. The user can select whether to apply this more accurate description or not. This can only be done only in combination with a net-node based bathymetry description, i.e. using the keyword `BedlevType = 3` in the `.mdu` file.

For the bed friction in 2D models, one implicitly assumes a fully developed vertical velocity profile, using a logarithmic function of the water depth for the White-Colebrook bed friction formulation, or a one-sixth power function of the water depth for the Manning formulation. In a sloping cell, the local water depth is varying over the width of the cell. In the deeper part flow velocities will be higher than in the shallower part. Bed stresses will vary over the width of a cell with water depth and with the velocity. Integrating the stresses over the width of a cell one can derive the resulting total stress.

The bed friction term is not only a function of the normal velocity component in the direction of the flow link itself, but also depends on the tangential velocity component and with that on the total velocity. For each of the four components water depth, normal velocity, total velocity and Chézy parameter, we assume a linear variation over the width (Figure 8.16).

If we have in the `.mdu` file `Conveyance2D = 3`, the 2D analytic conveyance description using these four linearly varying components is applied. This option shows best grid convergence behaviour. Good grid convergence implies that the converged answer can be achieved on a coarser grid, thus saving computational costs.

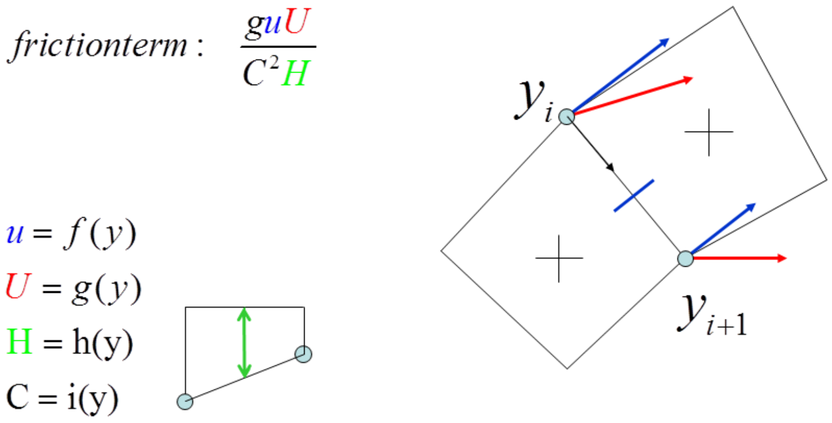


Figure 8.16: A shematic view of the linear variation over the width for calculating the flow parameters.

If one sets `Conveyance2D = 2`, the tangential velocity component is assumed zero. This method is only applicable on curvilinear grids that are aligned with the flow direction.

If one sets `Conveyance2D = 1`, both the normal and tangential velocity component are assumed constant over the width. Effectively one obtains the so called 'lumped' bed friction approach, with hydraulic radius $R = A/P$, A being the wet cross sectional area and P the wet perimeter. This method works equally well as methods 2 and 1, provided that there is sufficient resolution of a gully in the grid. It is found that when a gully is resolved by more than about 10 or 12 cells, it provides almost identical answers as method 2, while saving some 10 % computational overhead compared to method 2.

Setting `Conveyance2D = -1`, the hydraulic radius is based on a uniform bed level at the velocity point taken as the average bed level of the two surrounding net nodes.

Setting `Conveyance2D = 0`, the hydraulic radius is based on a uniform bed level at the velocity point taken as the average bed level of the two surrounding water level points. This option is not advisable because cell bed levels are taken as the minimum value of the bed levels of attached link. So a min-max operator is invoked, which is not suitable for accuracy.

Keyword `Conveyance2D` deals with the bed level at the velocity points (i.e., at flow links or cell edges) in the case of a net-node based bathymetry description (i.e. `BedlevType = 3`). When using a cell-centre based bathymetry description (i.e. `BedlevType = 1`) `[geometry]` keyword `Dpuopt` can be used to this end. Using the default value `Dpuopt = 1`, the bed level at links is the maximum of the adjacent cell-centre bed levels (i.e. of the bed levels connected by a link). Setting `Dpuopt = 2` computes the bed level at links as the mean value between the adjacent cell-centre bed levels.

Please note that setting `Dpuopt = 2` should not be confused with a piece-wise linear interpretation of the bed level. The bed level for `BedlevType = 1` is considered piece-wise constant and that is still the way in which the volumes are computed. The keyword `Dpuopt` only changes the way in which the bed level at the interface between two tiles is computed.

The default maximum bed level at links causes the flow depth at links to be minimum, which guarantees that there is no flooding of a dry cell of which the bed level is higher than the water lower of the wet cell. On the contrary, using the mean bed level at links may cause a positive flow depth at the link while the bed level of the dry cell is higher than the water level of the wet cell.

Computing the bed level at links as the mean value of the adjacent cell-centres (i.e. `Dpuopt = 2`) is particularly relevant for morphodynamic simulations. This increases the accuracy of the results, as the default maximum option causes a first-order upwind/downwind reconstruction while the mean ensures a second-order central reconstruction.

8.4.3 Shear-stresses at closed boundaries

A closed boundary is situated at the transition between land and water. At a closed boundary, two boundary conditions have to be prescribed. One boundary condition has to do with the flow normal to the boundary and the other one with the shear-stress along the boundary. A closed sidewalls is always considered as impermeable. For the shear stress along the boundary, the possible conditions to be prescribed are free-slip (zero tangential shear-stress), partial-slip and full-slip.

For large-scale simulations, the influence of the shear-stresses along closed boundaries can be neglected. Free slip is then applied for all closed boundaries. For simulations of smallscale flow (e.g. laboratory scale), the influence of the side walls on the flow may no longer be neglected. This option has been implemented for closed boundaries and dry points *but not for thin dams*. The reason is that the shear stress at a thin dam is dependent on the side (double valued).

Along the side walls, the tangential shear stress is calculated based on a logarithmic law of the wal. The friction velocity u_* is determined by the logarithmic law for a rough wall, with side wall roughness y_0 and the velocity in the first grid point near the side wall. Let Δy_s be the grid size normal to the sidwall, then:

$$|\vec{u}_{\text{sidewall}}| = \frac{u_*}{\kappa} \ln\left(1 + \frac{\Delta y_s}{2y_0}\right) \tag{8.22}$$

The choice for the way tangial shear stress are computed can be inserted through the key `irov` in the MDU-file (under `[physics]`):

- the case `irov = 0` treats the side walls as free-slip walls (default),
- the case `irov = 1` treats the side walls as partial-slip walls,
- the case `irov = 2` treats the side walls as no-slip walls.

If the choice for partial-slip walls is made, then a specific wall roughness value should be prescribed in the MDU-file. For this, the keyword `wall_ks` should be used (under `[physics]` in the MDU-file). The computational engine interpretes this value as Nikuradse roughness k_s. The value for y_0 is computed as $y_0 = k_s/30$.

8.4.4 Lateral discharges

Lateral discharges prescribe additional inflow (or outflow) into the system on specific locations in the model. Lateral discharges are merely adding (or extracting) volume to the model and no momentum. As such, they differ from discharge boundaries (section 8.4.1.2) and source-sinks (section 8.10), which *can* add momentum as well. The input of lateral discharges is via the external forcings file (section C.5.2.2); the following subsections describe the numerical treatment, the grid placement and the forcing information.

8.4.4.1 Numerical treatment of lateral discharges

Lateral discharges are merely a source of volume, and as such they are added to the right-hand-side source term of the continuity equation (Equation (8.1)). First all lateral contributions are summed for each grid cell. This allows that multiple different laterals are contributing to the same grid point:

$$Q_{\mathrm{lat},k} = \sum_{\ell} \sum_{k \in \mathcal{L}_\ell} Q_{\mathrm{lat},\ell,k}, \tag{8.23}$$

where $\mathcal{L}_{\mathrm{lat},\ell}$ is the subset of grid cells affected by lateral ℓ.

Next, positive discharges, i.e., inflows, are always fully added to the source term $Q_{\mathrm{in},k}$. Negative discharges, i.e., outflows are limited to the available water volume in the grid cell in each timestep.

$$Q_{\mathrm{in},k} \mathrel{+}= \begin{cases} Q_{\mathrm{lat},k}, & \text{if } Q_{\mathrm{lat},k} \geq 0, \\ -\min\left(\left|Q_{\mathrm{lat},k}\right|, \frac{1}{2} V_k / \Delta t\right), & \text{if } Q_{\mathrm{lat},k} < 0. \end{cases} \tag{8.24}$$

The relaxation factor of $\frac{1}{2}$ limits the negative source term contribution when little water is available.

Once all source terms have been collected in $Q_{\mathrm{in},k}$, the real calculation of new water levels is done. A positive source term is always added to the right hand side of the continuity equation (8.1), and the resulting water level at that grid point is calculated implicitly by the matrix solver. A negative source term $Q_{\mathrm{in},k}$ is solved explicitly in the continuity equation.

8.4.4.2 Grid snapping and distribution of discharge

The spatial location of lateral discharges can be a single point or a set of grid points, and both 1D and 2D points are supported (also combined). The grid snapping depends on the type of input:

- ◇ Single point input via a 1D network node id, or branch id/chainage. The given location is snapped to the nearest 1D pressure point.
- ◇ Single point input via a single x, y-point. The given location is snapped to the grid cell (1D or 2D) in which that x, y-point lies.
- ◇ Polygon area input via a sequence of x, y-points. The given area selects all pressure points that lie inside this polygon. If the input contains multiple polygons, it is sufficient for pressure points to lie in at least one of these polygons. Note that no "nesting" of polygon rings is detected, polygons can not be used to *exclude* regions from the lateral discharge.

The latter two types also allow the user to limit the grid point snapping to either 1D or 2D, via the keyword `locationType`, but this is optional.

When more than a single grid point (cell) is selected (only possible for polygon input), the prescribed lateral discharge $Q_{\mathrm{lat},\ell}$ is distributed across all selected grid cells based on the grid cell areas. Specifically, each grid cell k in the set $\mathcal{L}_{\mathrm{lat},\ell}$ of selected grid cells receives $Q_{\mathrm{lat},\ell,k}$, defined by:

$$Q_{\mathrm{lat},\ell,k} = \frac{Q_{\mathrm{lat},\ell}}{A_{\mathrm{lat,tot},\ell}} A_k, \tag{8.25}$$

where the total area affected by this lateral is defined as:

$$A_{\mathrm{lat,tot},\ell} = \sum_{k \in \mathcal{L}_{\mathrm{lat},\ell}} A_k. \tag{8.26}$$

This area-weighting ensures that the rate of change for the water levels in all selected grid cells is the same, as far as the forcing by this lateral discharge is concerned. Thus, a lateral discharge with a polygon shape could even be used to model local rainfall.

8.4.4.3 Forcing information for lateral discharges

The forcing information for lateral discharges is the intended flow rate for each of them. It can be defined in three ways, all starting with the keyword `discharge`:

- ◇ A constant value, e.g., `discharge = 10.0`, giving a constant flow rate of 10 m^3/s.
- ◇ A prescribed time series, stored in a <.bc> or <.tim> file, e.g., `discharge = forcing.bc`. This is described for boundaries in section 8.4.1.2. For laterals, the data format is the same, with one exception: one field `Quantity` should have the value `lateral_discharge`.
- ◇ An externally-provided discharge, i.e., `discharge = realtime`. More information about this type is given in the chapter on lumped rainfall-runoff coupling (chapter 18), which uses lateral discharges.

8.4.4.4 Output for lateral discharges

The lateral discharges can also be included in the output files for a model run. The total mass balance contributions can be included in the history file (section F.3.1.6). This is the total sum of lateral in- and outflows (possibly limited, see right-hand side in Equation (8.24)).

Flow data on each of the laterals can be written to the history file, see (section F.3.1.7). This can be controlled by setting `Wrihis_lateral = 0/1` in the MDU file under `[output]` (cf. Table A.1).

8.5 Artificial mixing due to sigma-coordinates

The σ-transformation is boundary-fitted in the vertical. The bottom boundary and free surface are represented smoothly. The water column is divided into the same number of layers independent of the water depth. In a σ-model, the vertical resolution increases automatically in shallow areas.

For steep bottom slopes combined with vertical stratification, σ-transformed grids introduce numerical problems for the accurate approximation of horizontal gradients both in the baroclinic pressure term and in the horizontal diffusion term. Due to truncation errors artificial vertical mixing and artificial flow may occur, Leendertse (1990) and Stelling and Van Kester (1994). This artificial vertical transport is sometimes called "creep".

Let z_b be the position of the bed and H the total water depth. If we consider the transformation from Cartesian co-ordinates to σ co-ordinates, defined by:

$$x = x^*\,,\; y = y^*\,,\; \sigma = \frac{z - z_b}{H} \quad (H = \zeta - z_b) \tag{8.27}$$

as result $\sigma = 0$ at the bed level and $\sigma = 1$ at the free surface. The horizontal pressure gradient reads:

$$\frac{\partial p}{\partial x} = \frac{\partial p^*}{\partial x^*}\frac{\partial x^*}{\partial x} + \frac{\partial p^*}{\partial \sigma}\frac{\partial \sigma}{\partial x} = \frac{\partial p^*}{\partial x^*} - \frac{1}{H}\left(\frac{\partial z_b}{\partial x} + \sigma\frac{\partial H}{\partial x}\right)\frac{\partial p^*}{\partial \sigma}. \tag{8.28}$$

In case of vertical stratification near steep bottom slopes, small pressure gradients at the left-hand side may be the sum of relatively large terms with opposite sign at the right-hand side.

Small truncation errors in the approximation of both terms result in a relatively large error in the pressure gradient.

This artificial forcing produces artificial flow.

The truncation errors depend on the grid sizes Δx and Δz.

Observations of this kind has led to the notion of "hydrostatic consistency", see also Figure 8.17. In the notation used by Haney (1991) this consistency relation is given by:

$$\left|\frac{\sigma}{H}\frac{\partial H}{\partial x}\right| < \left|\frac{\partial \sigma}{\partial x}\right| \tag{8.29}$$

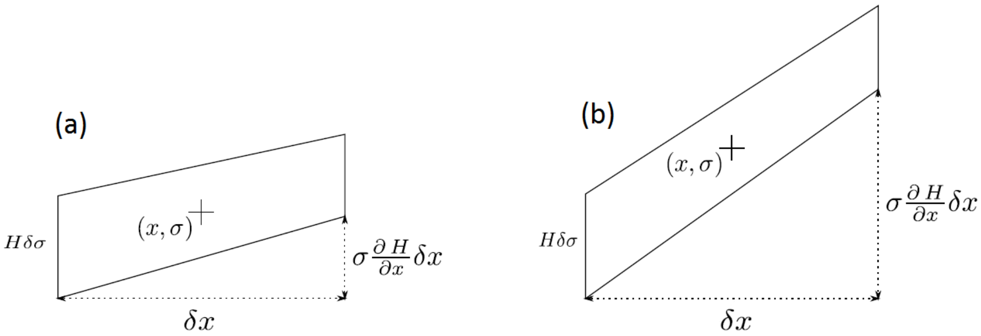

(a)
$H\delta\sigma$
(x,σ)
$\sigma\frac{\partial H}{\partial x}\delta x$
δx
(b)
$H\delta\sigma$
(x,σ)
$\sigma\frac{\partial H}{\partial x}\delta x$
δx

Figure 8.17: *Example of a hydrostatic consistent and inconsistent grid; (a)* $H\delta\sigma > \sigma\frac{\partial H}{\partial x}\delta x$*, (b)* $H\delta\sigma < \sigma\frac{\partial H}{\partial x}\delta x$

From this equation, it can be seen that by increasing the number of σ-levels the consistency condition will eventually be violated.

Similarly, for the horizontal diffusion term, the transformation from Cartesian co-ordinates to σ co-ordinates leads to various cross derivatives. For example, the transformation of a simple second order derivative leads to:

$$\frac{\partial^2 c}{\partial x^2} = \frac{\partial^2 c^*}{\partial x^{*2}} + \left(\frac{\partial \sigma}{\partial x}\right)^2 - \frac{\partial^2 c^*}{\partial \sigma^2} + 2\frac{\partial \sigma}{\partial x} - \frac{\partial^2 c^*}{\partial x^* \partial \sigma} + \frac{\partial^2 \sigma}{\partial x^2} - \frac{\partial c^*}{\partial \sigma} \tag{8.30}$$

For such a combination of terms it is difficult to find a numerical approximation that is stable and positive, see Huang and Spaulding (1996). Near steep bottom slopes or near tidal flats where the total depth becomes very small, truncations errors in the approximation of the horizontal diffusive fluxes in σ-co-ordinates are likely to become very large, similar to the horizontal pressure gradient.

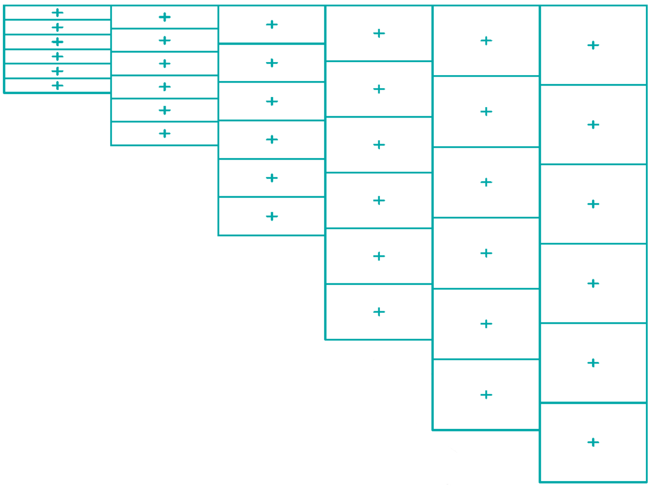


Figure 8.18: Finite Volume for diffusive fluxes and pressure gradients

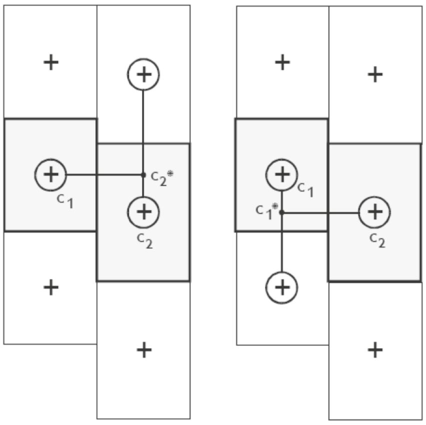


Figure 8.19: Left and right approximation of a strict horizontal gradient

In D-Flow FM the stress tensor is redefined in the σ co-ordinate system assuming that the horizontal length scale is much larger than the water depth (Mellor and Blumberg, 1985) and that the flow is of boundary-layer type. The horizontal gradients are taken along σ-planes. This approach guarantees a positive definite operator, also on the numerical grid (Beckers *et al.*, 1998).

If the same approach is used for the horizontal diffusion operator in the transport equation:

$$\frac{\partial^2 c}{\partial x^2} \approx \frac{\partial^2 c^*}{\partial x^{*2}} \tag{8.31}$$

Horizontal diffusion will lead to vertical transport of matter through vertical stratification interfaces (pycnocline) which is nonphysical. A more accurate, strict horizontal discretization is needed.

In D-Flow FM an option is available that minimises artificial vertical diffusion and artificial flow due to truncation errors. A method has been implemented which gives a consistent, stable and monotonic approximation of both the horizontal pressure gradient and the horizontal diffusion term, even when the hydrostatic consistency condition equation is not fulfilled. This "anti-creep" option is based upon a Finite Volume approach; see Figure 8.18. The horizontal diffusive fluxes and baroclinic pressure gradients are approximated in Cartesian co-ordinates by defining rectangular finite volumes around the σ-co-ordinate grid points. Since these boxes are not nicely connected to each other, see Figure 8.19, an interpolation in z co-ordinates is required to compute the fluxes at the interfaces.

Since the centres of the finite volumes on the left-hand side and right-hand side of a vertical interval are not at the same vertical level, a z-interpolation of the scalar concentration c is needed to compute strictly horizontal derivatives. The values obtained from this interpolation are indicated by c_1^* and c_2^* respectively in Figure 8.19. (Stelling and Van Kester, 1994) apply a non-linear filter to combine the two consistent approximations of the horizontal gradient,

```
$s_1 = (c_2^* - c_1)/\Delta x$ and $s_2 = (c_2 - c_1^*)/\Delta x$
If $s_1 \times s_2 < 0$ Then
    $\frac{\Delta c}{\Delta x} = 0$
Else
    $\frac{\Delta c}{\Delta x} = sign(s_1) \times \min(|s_1|, |s_2|)$
Endif
```
(8.32)

If an interval has only grid boxes at one side, the derivative is directly set to zero. The horizontal fluxes are summed for each control volume to compute the diffusive transport. The integration of the horizontal diffusion term is explicit with time step limitation:

$$\Delta t \leq \frac{1}{D_H}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) \tag{8.33}$$

The derivatives are used in the integral for the baroclinic pressure force in the momentum equation:

$$P_x(x,z) = \frac{1}{\rho_0}\int_z^{\zeta} g\frac{\partial \rho(x,s)}{\partial x}ds \tag{8.34}$$

Originally, this approach was implemented in Delft3D-FLOW. Slørdal (1997) stated that the above approximation may sometimes produce errors of the same sign which leads to a systematic underestimation of the baroclinic pressure term. This underestimation can be ascribed to the non-linear filter, which selects the minimum of the two gradients under consideration. This limiter is fully analogous to the min-mod limiter used for the construction of monotone advection schemes (Hirsch, 1990). Since the same approximation of the horizontal gradient is used for the horizontal diffusion flux, it is important to ensure that the difference operator is positive definite in order to get physically realistic solutions. The maximum and minimum of a variable being transported by diffusion do not increase or decrease (min-max principle). By taking the minimum of the gradients, Stelling and Van Kester (1994) show that, the min-max principle is fulfilled. Beckers *et al.* (1998) show that any nine-point consistent linear discretization of the horizontal diffusion on the σ-grid does not fulfil the min-max principle. From numerical tests Slørdal (1997) concluded that the underestimation is reduced by increasing the vertical resolution, but is sometimes enhanced by increasing the horizontal resolution.

Let s_4 be a consistent approximation of the horizontal gradient $s_4 = (s_1 + s_2)/2$. Slørdal (1997) suggested to take s_4 as approximation of the horizontal gradient. He calls his approach the "modified Stelling and Van Kester scheme". It is equivalent to linear interpolation at a certain z-level before taking the gradient. It is more accurate than taking the minimum of the absolute value of the two slopes s_1 and s_2 but it does not fulfil the min-max principle for the diffusion operator. It may introduce wiggles and a small persistent artificial vertical diffusion (except for linear vertical density distributions). Due to the related artificial mixing, stratification may disappear entirely for long term simulations, unless the flow is dominated by the open boundary conditions.

By introducing an additional approximation of the horizontal gradient in the filter algorithm defined by $s_3 = (c_2 - c_1)/\Delta x$, the stringent conditions of the minimum operator can be relaxed somewhat. The drawback of underestimation of the baroclinic pressure force reported by Slørdal (1997) can be minimised without loosing that the method fulfils the min-max principle. This third gradient s_3, which is consistent for $\min(|s_1|, |s_2|) < s_3 < \max(|s_1|, |s_2|)$, has point-to-point transfer properties and therefore leads to a positive scheme for sufficiently small time steps. The following non-linear approach presently available in D-Flow FM is both consistent and assures the min-max principle:

```
If $s_1 \times s_2 < 0$ Then
    $\frac{\Delta c}{\Delta x} = 0$
Elseif $|s_4| < |s_3|$ Then
    $\frac{\Delta c}{\Delta x} = s_4$
Elseif $\min(|s_1|, |s_2|) < |s_3| < \max(|s_1|, |s_2|)$ Then          (8.35)
    $\frac{\Delta c}{\Delta x} = s_3$
Else
    $\frac{\Delta c}{\Delta x} = \text{sign}(s_1) \min(|s_1|, |s_2|)$
Endif
```

The method requires a binary search to find the indices of neighbouring grid boxes, which is time consuming. The increase in computation time is about 30 %.

If the streamlines are strictly horizontal, transport of matter discretised on a σ co-ordinate grid may still generate some numerical vertical diffusion by the discretisation of the advection terms.

8.6 Secondary flow

The flow in a river bend is basically three-dimensional. The velocity has a component in the plane perpendicular to the river axis. This component is directed to the inner bend near the riverbed and directed to the outer bend near the water surface, see Figure 8.20.

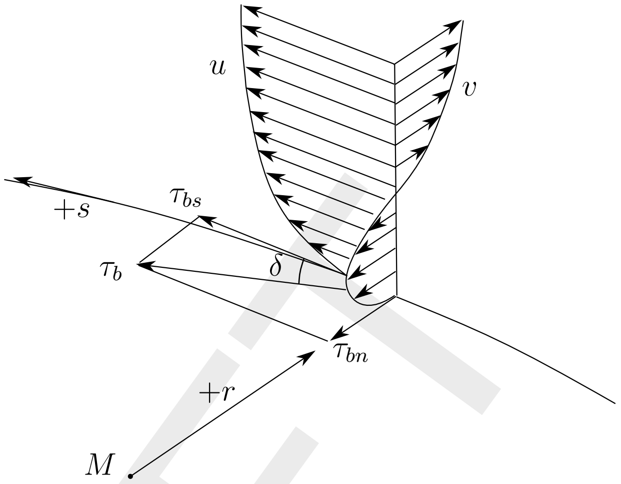


Figure 8.20: *Vertical profile secondary flow (v) in river bend and direction bed stress*

This so-called "secondary flow" (spiral motion) is of importance for the calculation of changes of the riverbed in morphological models and the dispersion of matter. In a 3D model the secondary flow is resolved on the vertical grid, but in 2D depth-averaged simulations the secondary flow has to be determined indirectly using a secondary flow model. It strongly varies over the vertical but its magnitude is small compared to the characteristic horizontal flow velocity.

8.6.1 Definition

The secondary flow will be defined here as the velocity component $v\left(\sigma\right)$ normal to the depth-averaged main flow. The spiral motion intensity of the secondary flow I is a measure for the magnitude of this velocity component along the vertical:

$$I=\int_0^1 \left|v\left(\sigma\right)\right| d\sigma \tag{8.36}$$

A vertical distribution for a river bend is given in Figure 8.20. The spiral motion intensity I leads to a deviation of the direction of the bed shear stress from the direction of the depth-averaged flow and thus affects the bedload transport direction. This effect can be taken into account in morphological simulations.

The component of the bed shear stress normal to the depth-averaged flow direction τ_{br} reads:

$$\tau_{br}=-2\rho\alpha^2\left(1-\frac{\alpha}{2}\right)\left|\boldsymbol{u}\right| I \tag{8.37}$$

where α is defined in Equation (8.48) and $\left|\boldsymbol{u}\right|$ is the magnitude of the depth-averaged velocity. To take into account the effect of the secondary flow on the depth-averaged flow, the depth-averaged shallow water equations have to be extended with:

- ◇ An additional advection-diffusion equation to account for the generation and adaptation of the spiral motion intensity.
- ◇ Additional terms in the momentum equations to account for the horizontal effective shear stresses originating from the secondary flow.

8.6.2 Depth-averaged continuity equation

The depth-averaged continuity equation is given by:

$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = Q \tag{8.38}$$

where u and v indicate the depth-averaged velocities along Cartesian axis.

8.6.3 Momentum equations in horizontal direction

The momentum equations in x- and y-direction are given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} - fv = -\frac{1}{\rho_0}\frac{\partial P}{\partial x} - \frac{gu\sqrt{u^2+v^2}}{C_{2D}^2 h} + F_x + F_{sx} + M_x \tag{8.39}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + fu = -\frac{1}{\rho_0}\frac{\partial P}{\partial y} - \frac{gv\sqrt{u^2+v^2}}{C_{2D}^2 h} + F_y + F_{sy} + M_y \tag{8.40}$$

The fourth term in the right-hand side represents the effect of the secondary flow on the depth-averaged velocities (shear stresses by depth-averaging the non-linear acceleration terms).

8.6.4 Effect of secondary flow on depth-averaged momentum equations

To account for the effect of the secondary flow on the depth-averaged flow, the momentum equations have to be extended with additional shear stresses. To close the equations these stresses are coupled to parameters of the depth-averaged flow field. The main flow is assumed to have a logarithmic velocity profile and the secondary flow originates from a multiplication of a universal function with the spiral motion intensity, see Kalkwijk and Booij (1986). Depth averaging of the 3D equations leads to correction terms in the depth-averaged momentum equations for the effect of spiral motion:

$$F_{sx} = \frac{1}{h}\left(\frac{\partial hT_{xx}}{\partial x} + \frac{\partial hT_{xy}}{\partial y}\right) \tag{8.41}$$

$$F_{sy} = \frac{1}{h}\left(\frac{\partial hT_{xy}}{\partial x} + \frac{\partial hT_{yy}}{\partial y}\right) \tag{8.42}$$

with the shear-stresses, resulting from the secondary flow, modelled as:

$$T_{xx} = -2\beta uv \tag{8.43}$$

$$T_{xy} = T_{yx} = \beta(u^2 - v^2) \tag{8.44}$$

$$T_{yy} = 2\beta uv \tag{8.45}$$

and:

$$\beta = \beta^* \frac{h}{R_s^*} \tag{8.46}$$

$$\beta^* = \beta_c\left(5\alpha - 15.6\alpha^2 + 37.5\alpha^3\right) \tag{8.47}$$

$\beta_c \in [0,1]$, correction coefficient specified by you,

$$\alpha = \frac{\sqrt{g}}{\kappa C_{2D}} < \frac{1}{2} \tag{8.48}$$

with R_s^* the effective radius of curvature of a 2D streamline to be derived from the intensity of the spiral motion and κ the Von Kármán constant. The spiral motion intensity is computed by Equation (8.49). The limitation on α, Equation (8.48), is set to ensure that the length scale L_a in Equation (8.56) is always positive. For $\beta_c = 0$, the depth-averaged flow is not influenced by the secondary flow.

Remark:

◇ Equation (8.48) effectively means a lower limit on C_{2D}.

8.6.5 The depth averaged transport equation for the spiral motion intensity

The variation of the spiral motion intensity I in space and time, is described by a depth-averaged advection-diffusion equation:

$$\frac{\partial hI}{\partial t} + \frac{\partial uhI}{\partial x} + \frac{\partial vhI}{\partial y} = h\frac{\partial}{\partial x}\left(D_H\frac{\partial I}{\partial x}\right) + h\frac{\partial}{\partial y}\left(D_H\frac{\partial I}{\partial y}\right) + hS \tag{8.49}$$

with:

$$S = -\frac{I - I_e}{T_a} \tag{8.50}$$

$$I_e = I_{be} - I_{ce} \tag{8.51}$$

$$I_{be} = \frac{h}{R_s}\left|\boldsymbol{u}\right| \tag{8.52}$$

$$I_{ce} = f\frac{h}{2} \tag{8.53}$$

$$\left|\boldsymbol{u}\right| = \sqrt{u^2 + v^2} \tag{8.54}$$

$$T_a = \frac{L_a}{\left|\boldsymbol{u}\right|} \tag{8.55}$$

$$L_a = \frac{(1 - 2\alpha)\,h}{2\kappa^2\alpha} \tag{8.56}$$

and R_s the radius of curvature of the stream-line defined by:

$$\frac{u_s}{R_s} = -\frac{\partial u_r}{\partial s} \tag{8.57}$$

with u_s and u_r the components along and perpendicular to the streamline. The effective radius of curvature to be used for the evaluation of the coefficient β, Equation (8.47), reads:

$$R_s^* = \frac{h\left|\boldsymbol{u}\right|}{I} \tag{8.58}$$

To guarantee stability the effective radius of curvature is bounded by the following empirical relation:

$$R_s^* \geq 10h \tag{8.59}$$

The above formulas account for two sources of secondary flow:

◇ The centrifugal force in case of curved streamlines, I_{be}.
◇ The effect of the Coriolis force, I_{ce}.

8.7 1D cross sections

8.7.1 Introduction

The bathymetry of a 1D element is defined by means of a cross section.

The available cross section types in D-Flow FM are:

- YZ cross section (section 8.7.2.1)
- XYZ cross section (section 8.7.2.1)
- ZW (tabulated) cross section (section 8.7.2.2)
- ZWriver cross section (section 8.7.2.3)
- Circle cross section (section 8.7.2.4)
- Rectangle cross section (section 8.7.2.5)
- Template cross section (section 8.7.2.6)

These cross sections are used for calculating the conveyance and flow area of a channel at a flow velocity point and the total volume in water level points.

For the calculation of the conveyance D-Flow FM offers two different approaches.

- Vertically segmented approach (section 8.7.5.1).
- Lumped approach (section 8.7.5.2)

For YZ and XYZ type cross sections the user can choose for either the vertically segmented approach (default) or the lumped approach. For all other cross section types the lumped approach is used.

The ZWriver cross section is a mix between the lumped and a vertically segmented approach. For this cross section type the user can define three different sections. (i.e. 'Main', 'Floodplain1' and 'Floodplain2'). For each separate section a lumped conveyance is calculated and summed up to construct a total conveyance.

Cross sections can be defined at arbitrary locations on branches. The required data is then interpolated to the correct location. The interpolation is described in section 8.7.7.

8.7.2 Cross section types

8.7.2.1 YZ and XYZ cross section

In a YZ-type cross section the user specifies the bed level, i.e., Z-value at given Y-points (see Figure 8.21).

An XYZ-type cross section contains (x, y) coordinates of the bed level locations. In the computational core an XYZ-cross section is changed into a YZ-type cross section. The distance between the different (x, y) locations is used to construct the relative Y-values for the YZ cross section.

Remarks:

- The Y-values in the YZ cross section must be given as a metric value ([m AD]), regardless of the coordinate reference system of the model.
- The (x, y) coordinates in the XYZ cross section must given in the coordinate reference system of the model (that is, in metres for projected coordinate systems, or degrees for a spherical coordinate system).

◇ In case two subsequent Z-values are identical, the second value is raised with 10^{-3} m.

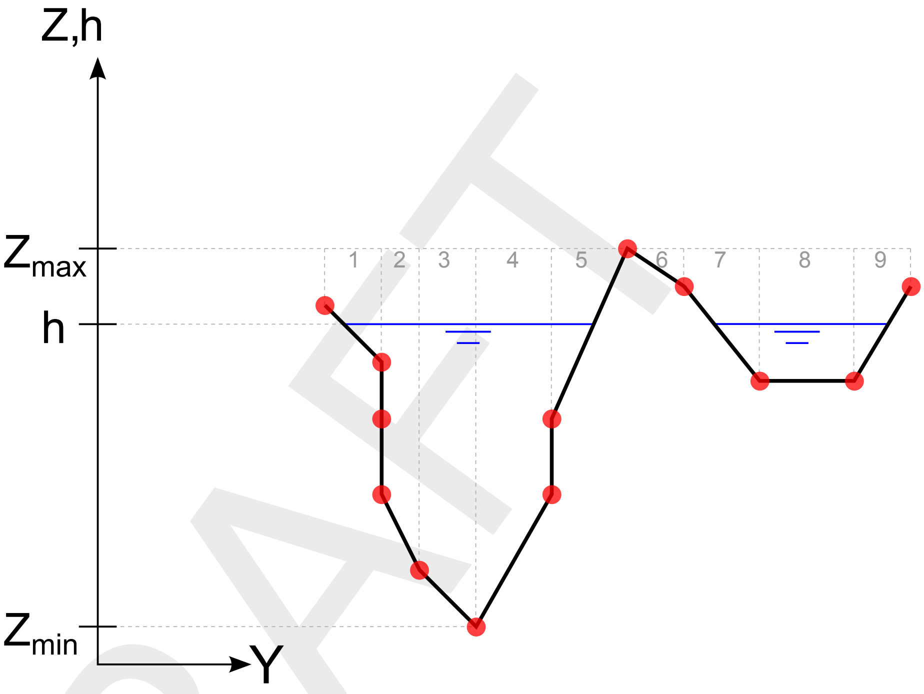


Figure 8.21: Example of constructed vertical segments in a Y-Z profile
13 (y_i, z_i) points red dots
9 vertical segments

8.7.2.2 ZW cross section

In a ZW-type cross section the user specifies the shape of the cross section by specifying the width at certain levels. For this cross section type it is possible to make a distinction between the flow area and the total area.

An input example for a ZW-cross section definition is given in Table 8.1. In Figure 8.22 the cross section for this example is shown. The blue shaded area shows the storage, i.e., non-flowing area.

Table 8.1: Example of a ZW cross section definition

Z	Width	Storage width
1	500	250
-4	450	330
-4.2	110	0
-10	33	0

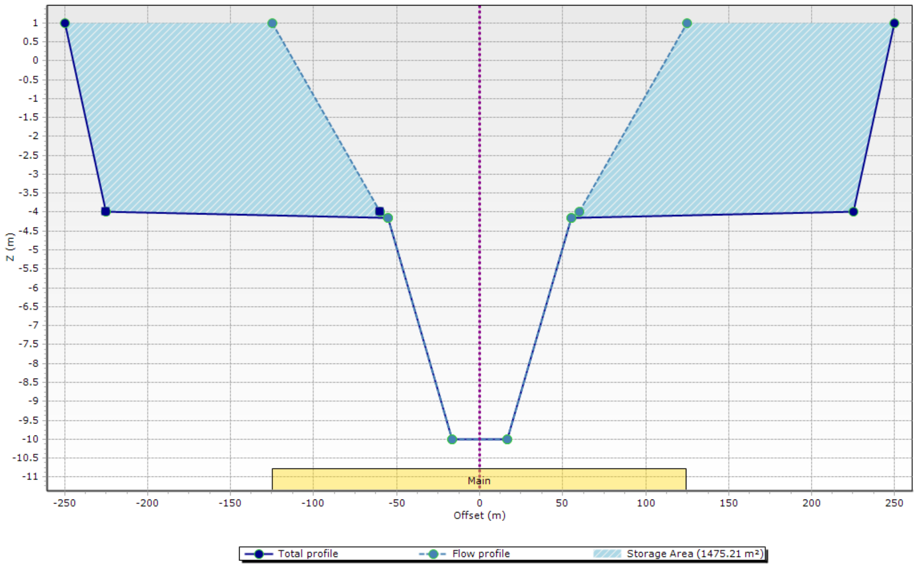


***Figure 8.22:** Example a ZW cross section*

8.7.2.3 ZWriver cross section

Not yet described

8.7.2.4 Circle cross section

A circular shaped cross section is defined by its diameter. Section C.16.1.1 describes the cross section definition input.

8.7.2.5 Rectangle cross section

A rectangular cross section is defined by its width and height and whether it is open on the top side or not. Section C.16.1.2 describes the cross section definition input.

8.7.2.6 Template cross sections

A 'Template cross section' is offered by the GUI to support a number of standard shapes. These are translated into the equivalent ZW-definition. The computational kernel then treats these as a tabulated ZW-type cross section. See also the optional keyword `template` in section C.16.1.4.

8.7.3 Preissmann slot and closed cross sections

Closed cross sections are cross sections that have a 0 m width above a certain level. Closed cross sections can be used on ordinary branches. An application for this functionality is the modelling of sewer systems, where the sewer pipes are modelled as ordinary branches with a closed cross section.

Note: Long culverts are also modelled as ordinary branches in D-Flow FM.

In case a branch with a closed cross section is completely filled, the water flow becomes a pressurised flow and not a free surface flow. Since D-Flow FM is not designed for simulations with pressurised flow, a Preissmann slot is added at the top of the cross section.

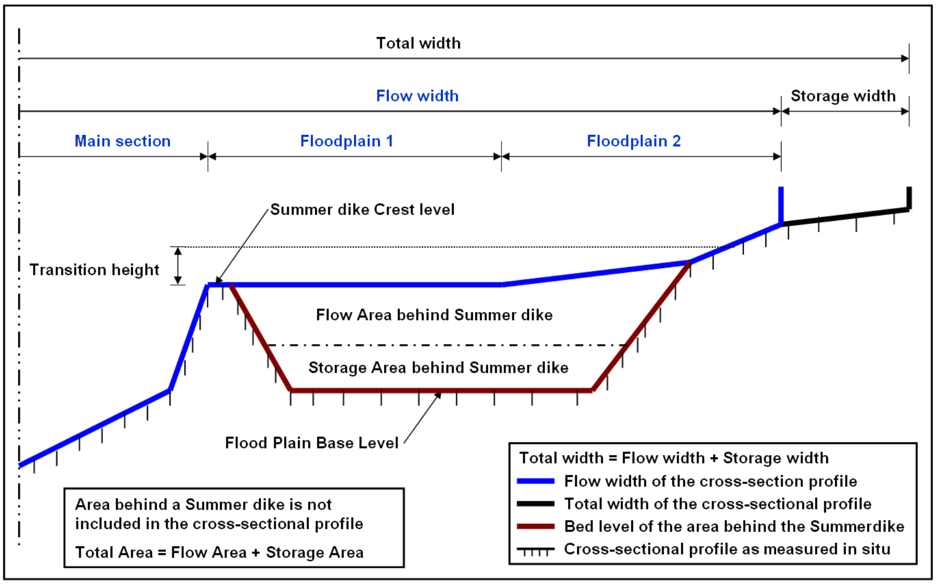

Total width
Flow width
Storage width
Main section
Floodplain 1
Floodplain 2
Summer dike Crest level
Transition height
Flow Area behind Summer dike
Storage Area behind Summer dike
Flood Plain Base Level
Area behind a Summer dike is not included in the cross-sectional profile
Total Area = Flow Area + Storage Area
Total width = Flow width + Storage width
Flow width of the cross-section profile
Total width of the cross-sectional profile
Bed level of the area behind the Summerdike
Cross-sectional profile as measured in situ

Figure 8.23: *levee option, available in a river profile*

A Preissmann slot is a narrow extension of the cross sectional area, above the part that is originally closed. As a result even closed pipes, will always have a small free surface area. This Preissmann slot is only used in calculating the amount of water in a water level point and in calculating the free surface area. The flow area and hydraulic radius are not affected by this Preissmann slot.

The Preissmann slot is set to 0.01 m.

Closed cross sections can also be used for the hydraulic structures, bridges and culverts. Since hydraulic structures do not have any effect on the storage of a model or the free surface area, no Preissmann slot is required for closed cross sections in hydraulic structures.

8.7.4 Levees

A flood plain might be separated from a river by means of a small levee, having a crest level lower than the crest level of the main dike (see Figure 8.23). In the Netherlands large floods usually occur in winter. The main dike and the small levee are, therefore, respectively referred to as the “winter dike” and the “summer dike”.

The presence of a levee implies that the cross-sectional profile as measured in situ (see shadowed line in Figure 8.23) does not increase monotonously with rising water levels. When the water level becomes slightly higher than the crest level of the levee, first the area behind the levee is to be filled, before water levels can become any higher. This implies a local attenuation of the flood wave at water levels just above the crest level of the levee. This hydraulic phenomenon can be modelled using the levee option available at a river profile.

In defining a levee in a river profile a distinction is to be made between:

◇ **The cross-sectional profile**
The cross-sectional profile comprises of a “flow area” and a “storage area”, which in Figure 8.23 are respectively located above the blue lines and the black lines. The summation of flow area and storage area is called the “total area”. Both flow area and total area are defined by its width as function of elevation. At each elevation in the cross-sectional profile

yields that storage width equals total width minus flow width. Hence, no storage is defined if total width equals flow width. The flow area of the cross-sectional profile can be divided in three separate flow sections, respectively called the (i) main section, (ii) floodplain 1 and (iii) floodplain 2 . Each such flow section can have a different roughness value and a different roughness formulation.

◇ **The area behind the summer-dike**
In analogy with the cross-sectional profile, the "total area behind a levee" is divided in a "flow area" and a "storage area". The "Flood Plain Base Level" coincides with the lowest part of the area behind the levee (see Figure 8.23).

Remark:

◇ Both the cross-sectional profile and the area behind the levee should be mutually consistent with the cross-sectional profile as measured in situ (see shadowed line in Figure 8.23). Hence, both the flow area and the storage area behind the levee should not be included in the cross-sectional profile.

The so-called "transition height" is used for avoiding numerical oscillations. More precisely, the transition height ensures that the flow area and storage area behind the levee are gradually taken into computation. Transition height for levees is a model-wide parameter.

Lets consider the following levee properties:

◇ flood plain base level (z_{base}) of 2 m,
◇ levee crest level (z_{sd}) of 4 m,
◇ transition height (T) of 0.5 m,
◇ storage area behind the levee of 200 m^2,
◇ flow area behind the levee (ExtraFlowArea) of 50 m^2, and
◇ total area behind the levee (ExtraTotalArea) of 250 m^2.

In explaining how a levee is taken into computation, we make a distinction between:

◇ The actual value of the (extra) flow area behind the levee,
◇ The flow velocity applied to the (extra) flow area behind the levee, and
◇ The actual value of the (extra) total area behind the levee.

The actual (extra) flow area is a unique function of the local water level (h) in the cross-sectional profile (see Figure 8.24). More precisely:

◇ If $\zeta < z_{\text{sd}}$; Flow area behind the levee is zero.
◇ If $z_{\text{sd}} \leq \zeta \leq z_{\text{sd}} + T$; Flow area behind the levee varies as a quadratic function of water level (ζ) between zero and ExtraFlowArea (= 50 m^2).
◇ If $\zeta > z_{\text{sd}} + T$; Flow area behind the levee is equal to ExtraFlowArea.

The flow velocity applied to the (extra) flow area behind the levee is the flow velocity in one of the three available flow sections. Hence, either the flow velocity in the main section, in floodplain 1 or in floodplain 2. The actual flow velocity applied to the (extra) flow area behind the levee is determined as follows:

1 Default the flow velocity in floodplain 2 is used.
2 However, if the wetted area of floodplain 2 becomes less than 0.4 m^2, the flow velocity in floodplain 1 is used,
3 Furthermore, if the wetted area of floodplain 1 becomes less than 0.4 m^2, the flow velocity of the main section is used.

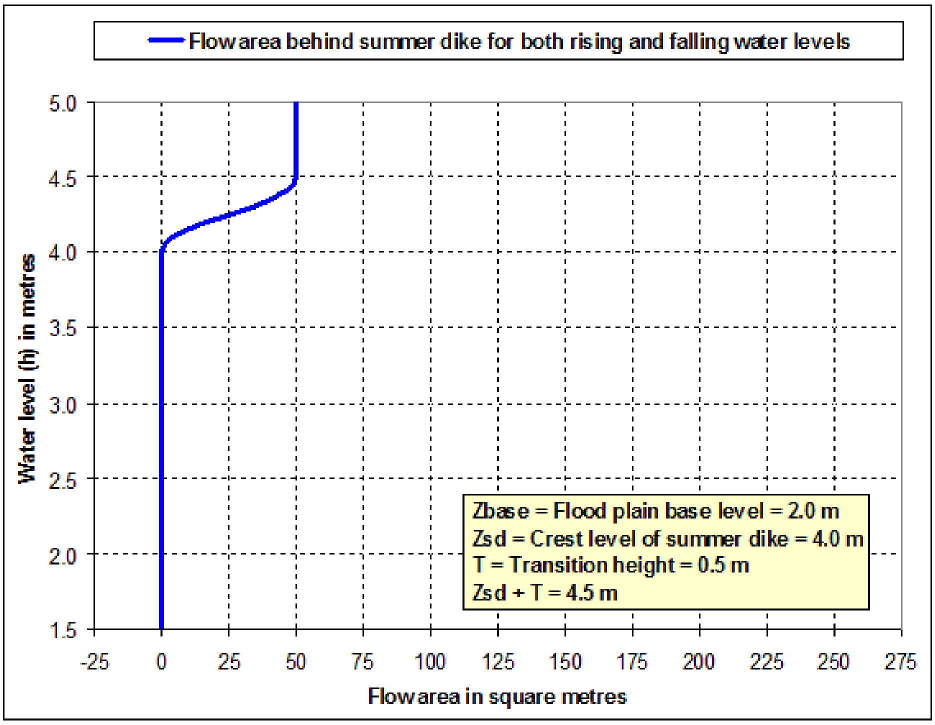


Figure 8.24: *Flow area behind the levee as function of local water levels.*

Remarks:

◇ Please contact Deltares if you like to use a different threshold value (i.e. not 0.4 m^2) for discerning which flow velocity is to be applied to the (extra) flow area behind the levee.

◇ The flow velocity applied to the (extra) flow area behind the levee is computed as follows

$$U_i = C_i\sqrt{R_i S} \qquad i \in \{\text{main section}, \text{floodplain 1}, \text{floodplain 2}\} \tag{8.60}$$

where:

U_i average flow velocity
C_i Chézy coefficient of the flow section
R_i $(A_i + A_{\text{sd}})/P_i$: hydraulic radius of the flow section
A_i wetted area of the flow section
A_{sd} extra flow area behind the levee
P_i wetted perimeter of the flow section
S water level slope

◇ The discharge including the (extra) flow area behind the levee is computed as follows

$$Q_i = U_i(A_i + A_{\text{sd}}) \tag{8.61}$$

Figure 8.25 depicts the actual (extra) total area behind the levee as function of the local water level (ζ) in the cross-sectional profile. The (extra) total area behind the levee is added to the total area of the cross-sectional profile. There is a hysteresis in the extra total area for water levels varying from flood plain base level ($z_{\text{base}} = 2.0\ m$) to crest level of levee plus transition height ($z_{\text{sd}} + T = 4.5\ m$). We call line ABC the rising limp and line CA the falling limp of the hysteresis. The so-called “hysteresis flag” determines which limp of the hysteris is to be followed. In case the hysteresis `flag=1` the rising limp of the hysteresis is followed, else the falling limp is followed. At the onstart of a computation yields that the hysteresis `flag=1`. The actual applied extra total area behind the levee is computed as follows:

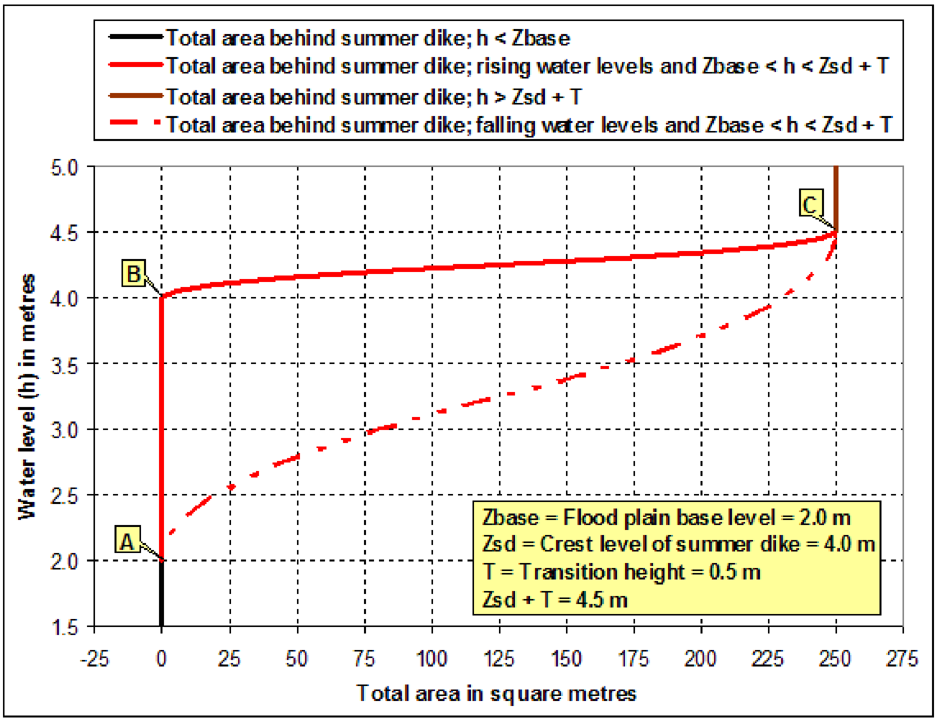


Figure 8.25: *Total area behind the levee as function of local water levels.*

1 If $\zeta < z_{\text{base}}$; Hysteresis flag is set to 1. Total area behind the levee is zero.
2 If $z_{\text{base}} \leq \zeta \leq z_{\text{sd}} + T$ and hysteresis `flag=1`;

- ◇ For $z_{\text{base}} \leq \zeta < z_{\text{sd}}$; Total area behind the levee is zero.
- ◇ For $z_{\text{sd}} \leq \zeta \leq z_{\text{sd}} + T$; Total area behind the levee varies as a quadratic function of water level (ζ) between zero and ExtraTotalArea (= 250 m^2)

3 If $z_{\text{base}} \leq \zeta \leq z_{\text{sd}} + T$ and hysteresis `flag=0`; Total area behind the levee varies as a quadratic function of water level (ζ) between zero and ExtraTotalArea.
4 If $\zeta > z_{\text{sd}} + T$; Hysteresis flag is set to 0. Total area behind the levee is equal to ExtraTotalArea.

8.7.5 Conveyance

The bed friction term in the 1D momentum equation is similar to the 2D bed friction in Equation (8.14):

$$\tau_b = \frac{\rho_0 g U \left|U\right|}{C^2}, \tag{8.62}$$

with Chézy coefficient C. In channels with flood plains, it is preferable to compute bed friction effects in vertical segments (more details follow), and for this purpose we rewrite the bed friction in terms of conveyance K. The conveyance is defined by

$$K = C A_F \sqrt{R}, \tag{8.63}$$

where A_F is the flow area and R is the hydraulic radius.

Substituting the conveyance in the bed friction term yields:

$$\tau_b = \frac{\rho_0 g A_F^2 U |U|}{K^2}. \tag{8.64}$$

In D-Flow FM two different approaches are implemented for calculating the 1D conveyance:

- Vertically segmented approach, this approach only applies to YZ cross section definitions. It is preferable for cross section shapes that have significant depth differences between large parts of the cross section, such as river like channels with large floodplains. (section 8.7.5.1)
- Lumped approach, this approach is preferable for smaller canals and sewer systems, also because it is better for including roughness of (nearly) vertical walls. (section 8.7.5.2)

8.7.5.1 Conveyance according to the vertically segmented approach

The vertically segmented conveyance approach is suited for modelling flow in rivers, since in this approach it is assumed that flow velocities vary significantly across the cross-sectional profile for a given water level. In line with this assumption, the cross-section is divided in vertical segments (see Figure 8.21).

A Y-Z profile is defined by a number (n) of (y_i, z_i) points. The total number of vertical segments equals: $n_{\text{seg}} = n - 1$. The bed or wall stress effects of nearly vertical walls is not well included in computing the conveyance of a Y-Z cross section. This underestimation of vertical wall stresses in computing the conveyance of a Y-Z profile, means that for a canal having a rectangular profile with a small ratio of width and water depth only, a substantial difference in discharge capacity can occur if such a canal is modelled as a rectangular cross-section (i.e. lumped conveyance approach) or as a rectangular shaped Y-Z profile (i.e. vertically segmented conveyance approach).

The conveyance (K_i) of each vertical segment is computed separately (for formulae, see conveyance per vertical segment further below). The conveyance of each inclined vertical segment is determined by solving an integral from y_i to y_{i+1}. The advantage of using an integral is that irrespective of the number of user-defined intermediate points, that lie on a straight line between (y_0, z_0) to (y_1, z_1), the same conveyance for the cross-sectional part from y_0 to y_1 is computed. The total conveyance (K_{tot}) at a particular water level is the summation of the conveyance of all the vertical segments $(K_{tot} = \sum K_i)$.

Conveyance tables, applied in the vertically segmented conveyance approach

Conveyance tables are a table representation of the conveyance integral values and are used for increasing the computational performance. A conveyance table gives the conveyance, flow width and flow area of a cross-section for a finite number of water levels. Conveyance tables are used for Y-Z cross sections only. These conveyance tables are made during initialisation, before the start of a hydrodynamic calculation. During the computation the conveyance at a certain water level is determined by means of linear interpolation between table values for the two surrounding support points. In constructing the conveyance table of each user-defined cross-section, it is ensured that the difference between the interpolated conveyance and the exact conveyance is less than 1 %.

Conveyance tables are made from the bed-level up to the highest (y_i, z_i) point, irrespective of the actual location of the highest (y_i, z_i) point within the cross-sectional profile.

Extrapolation of conveyance tables

When the water level (ζ) rises above the highest level ($n_{\max}$) of a conveyance table, the conveyance ($K(\zeta)$) is computed as

$$K(\zeta) = a(\zeta - z_{\min})^{b}, \tag{8.65}$$

where $z_{\min}$ is the lowest level in the cross-sectional profile and a, b are constants derived from the conveyances at levels $n_{\max}$ and $n_{\max} - 1$.

Conveyance per vertical segment

In this section, a single vertical segment i is considered, for notational simplicity defined by two (y_0, z_0) and (y_1, z_1) points, which do not lie on top of each other ($y_0 \neq y_1$) (see Figure 8.26). The local water depth $d(y)$ at a particular point on this segment is defined as:

$$d(y) = \zeta - z(y). \tag{8.66}$$

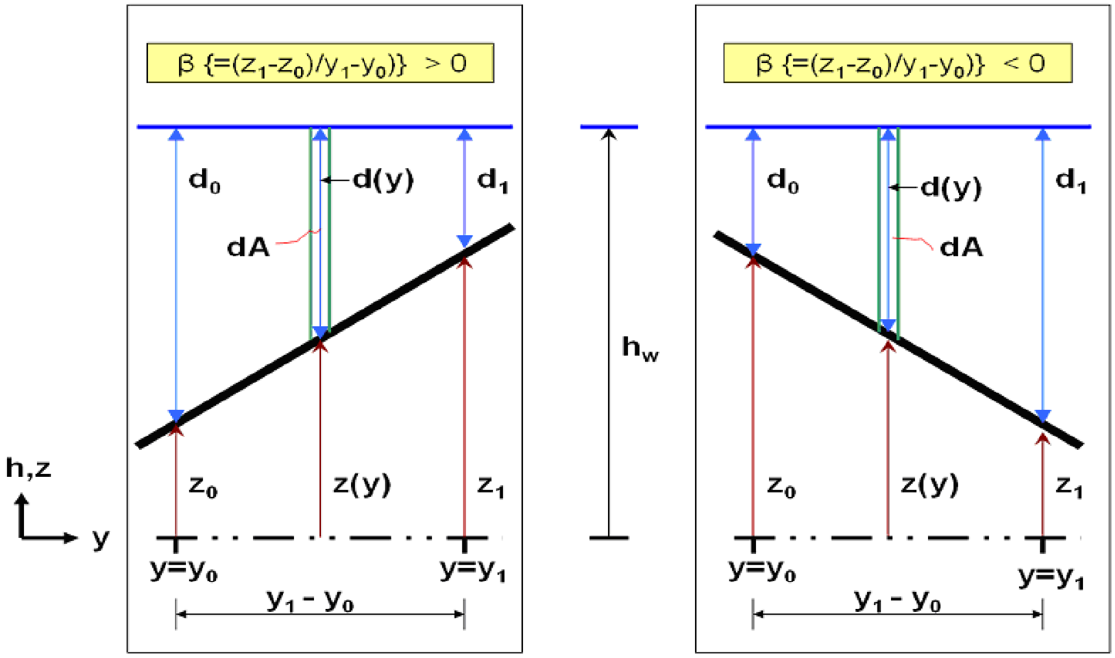


Figure 8.26: *Definition sketch of a vertical segment, considered in computing conveyance for a Y-Z profile and an Asymmetrical Trapezium cross-section*

The calculation of the conveyance integral for a vertical segment depends on the type of friction used. Below, the conveyance formulae are given for:

- Chézy roughness,
- Manning roughness,
- Strickler (k_n) roughness,
- Strickler (k_s) roughness,
- White-Colebrook (k_n) roughness, and
- Bos & Bijkerk (γ) roughness.

Chézy roughness conveyance for a vertical segment (see Figure 8.26):

```
$d = \max(d_0, d_1)$
if $d \leq 0$ then
    $K_i = 0$
else
    if $d < |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_1 = \frac{2C}{|\beta|(1+\beta^2)^{1/4}} \left(\frac{d}{2}\right)^{5/2}$
        elseif $|\beta| > 0.01$ then
            $K_i = k_2 = \frac{2C}{5|\beta|(1+\beta^2)^{1/4}} d^{5/2}$
    elseif $d \geq |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_3 = \frac{C}{(1+\beta^2)^{1/4}} \left(\frac{d_0+d_1}{2}\right)^{3/2} (y_1 - y_0)$
        elseif $|\beta| > 0.01$ then
            $K_i = k_4 = \frac{2C}{5|\beta|(1+\beta^2)^{1/4}} \left| d_0^{5/2} - d_1^{5/2} \right|$
        endif
    endif
endif
```

Manning roughness conveyance for a vertical segment (see Figure 8.26):

```
$d = \max(d_0, d_1)$
if $d \leq 0$ then
    $K_i = 0$
else
    if $d < |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_1 = \frac{2}{n|\beta|(1+\beta^2)^{1/4}} \left(\frac{d}{2}\right)^{8/3}$
        elseif $|\beta| > 0.01$ then
            $K_i = k_2 = \frac{3}{8n|\beta|(1+\beta^2)^{1/4}} d^{8/3}$
    elseif $d \geq |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_3 = \frac{1}{n(1+\beta^2)^{1/4}} \left(\frac{d_0+d_1}{2}\right)^{5/3} (y_1 - y_0)$
        elseif $|\beta| > 0.01$ then
            $K_i = k_4 = \frac{3}{8n|\beta|(1+\beta^2)^{1/4}} \left| d_0^{8/3} - d_1^{8/3} \right|$
        endif
    endif
endif
```

Strickler (k_n) roughness conveyance for a vertical segment (see Figure 8.26):

```
$d = \max(d_0, d_1)$
if $d \leq 0$ then
    $K_i = 0$
else
    if $d < |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
```

```
            $K_i = k_1 = \frac{50k_n^{-1/6}}{|\beta|(1+\beta^2)^{1/4}} \left(\frac{d}{2}\right)^{8/3}$
        elseif $|\beta| > 0.01$ then
            $K_i = k_2 = \frac{75k_n^{-1/6}}{8|\beta|(1+\beta^2)^{1/4}} d^{8/3}$
    elseif $d \geq |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_3 = \frac{25k_n^{-1/6}}{(1+\beta^2)^{1/4}} \left(\frac{d_0+d_1}{2}\right)^{5/3} (y_1 - y_0)$
        elseif $|\beta| > 0.01$ then
            $K_i = k_4 = \frac{75k_n^{-1/6}}{8|\beta|(1+\beta^2)^{1/4}} \left|d_0^{8/3} - d_1^{8/3}\right|$
        endif
    endif
endif
```

Strickler (k_s) roughness conveyance for a vertical segment (see Figure 8.26):

```
$d = \max(d_0, d_1)$
if $d \leq 0$ then
    $K_i = 0$
else
    if $d < |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_1 = \frac{2k_s}{|\beta|(1+\beta^2)^{1/4}} \left(\frac{d}{2}\right)^{8/3}$
        elseif $|\beta| > 0.01$ then
            $K_i = k_2 = \frac{3k_s}{8|\beta|(1+\beta^2)^{1/4}} d^{8/3}$
    elseif $d \geq |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_3 = \frac{k_s}{(1+\beta^2)^{1/4}} \left(\frac{d_0+d_1}{2}\right)^{5/3} (y_1 - y_0)$
        elseif $|\beta| > 0.01$ then
            $K_i = k_4 = \frac{3k_s}{8|\beta|(1+\beta^2)^{1/4}} \left|d_0^{8/3} - d_1^{8/3}\right|$
        endif
    endif
endif
```

White-Colebrook (k_n) roughness conveyance for a vertical segment (see Figure 8.26):

```
$d = \max(d_0, d_1)$
$c_1 = (1 + \beta^2)^{1/4} \ln(10)$
$c_2 = \frac{k_n}{12}$
if $d \leq 0$ then
    $K_i = 0$
else
    if $d < |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            if $\frac{6d}{k_n} \leq 1.0005$ then
                $K_i = k_1 = \frac{36}{|\beta|(1+\beta^2)^{1/4}} 0.00022 \left(\frac{d}{2}\right)^{5/2}$
            else
                $K_i = k_1 = \frac{36}{|\beta|(1+\beta^2)^{1/4}} \, {}^{10}\log\left\{\frac{6d}{k_n}\right\} \left(\frac{d}{2}\right)^{5/2}$
            endif
            elseif $|\beta| > 0.01$ then
                if $\frac{d}{c_2} \leq 1.495$ then
                    $K_i = k_2 = \frac{36}{5|\beta|c_1} d^{5/2} 0.00213$
                else
                    $K_i = k_2 = \frac{36}{5|\beta|c_1} d^{5/2} \left( \ln \left\{ \frac{d}{c_2} \right\} - \frac{2}{5} \right)$
                endif
        elseif $d \geq |z_1 - z_0|$ then
            if $|\beta| \leq 0.01$ then
                if $\frac{6(d_0+d_1)}{k_n} \leq 1.0005$ then
                    $K_i = k_3 = \frac{18}{(1+\beta^2)^{1/4}} 0.00022 (y_1 - y_0) \left( \frac{d_0+d_1}{2} \right)^{3/2}$
                else
                    $K_i = k_3 = \frac{18}{(1+\beta^2)^{1/4}} \, {}^{10}\log \left\{ \frac{6(d_0+d_1)}{k_n} \right\} (y_1 - y_0) \left( \frac{d_0+d_1}{2} \right)^{3/2}$
                endif
            elseif $|\beta| > 0.01$ then
                if $\frac{d_0}{c_2} \leq 1.495$ and $\frac{d_1}{c_2} \leq 1.495$ then
                    $K_i = k_4 = \frac{36}{5|\beta|c_1} \left| d_0^{5/2} 0.00213 - d_1^{5/2} 0.00213 \right|$
                elseif $\frac{d_0}{c_2} \leq 1.495$ and $\frac{d_1}{c_2} > 1.495$ then
                    $K_i = k_4 = \frac{36}{5|\beta|c_1} \left| d_0^{5/2} 0.00213 - d_1^{5/2} \left( \ln \left\{ \frac{d_1}{c_2} \right\} - \frac{2}{5} \right) \right|$
                elseif $\frac{d_0}{c_2} > 1.495$ and $\frac{d_1}{c_2} \leq 1.495$ then
                    $K_i = k_4 = \frac{36}{5|\beta|c_1} \left| d_0^{5/2} \left( \ln \left\{ \frac{d_0}{c_2} \right\} - \frac{2}{5} \right) - d_1^{5/2} 0.00213 \right|$
                else
                    $K_i = k_4 = \frac{36}{5|\beta|c_1} \left| d_0^{5/2} \left( \ln \left\{ \frac{d_0}{c_2} \right\} - \frac{2}{5} \right) - d_1^{5/2} \left( \ln \left\{ \frac{d_1}{c_2} \right\} - \frac{2}{5} \right) \right|$
                endif
            endif
        endif
    endif
endif
```

Bos & Bijkerk (γ) roughness conveyance for a vertical segment (see Figure 8.26):

```
$d = \max(d_0, d_1)$
if $d \leq 0$ then
    $K_i == 0$
else
    if $d < |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_1 = \frac{2\gamma}{|\beta|(1+\beta^2)^{1/4}} \left( \frac{d}{2} \right)^3$
        elseif $|\beta| > 0.01$ then
            $K_i = k_2 = \frac{\gamma}{3|\beta|(1+\beta^2)^{1/4}} d^3$
    elseif $d \geq |z_1 - z_0|$ then
        if $|\beta| \leq 0.01$ then
            $K_i = k_3 = \frac{\gamma}{(1+\beta^2)^{1/4}} \left( \frac{d_0+d_1}{2} \right)^2 (y_1 - y_0)$
        elseif $|\beta| > 0.01$ then
            $K_i = k_4 = \frac{\gamma}{3|\beta|(1+\beta^2)^{1/4}} \left| d_0^3 - d_1^3 \right|$
        endif
    endif
```

endif

8.7.5.2 Conveyance according to the lumped approach

The lumped conveyance for a given water depth h is computed as:

$$K(h) = A(h)C(h)\sqrt{R(h)} \tag{8.67}$$

where:

$K(h)$	Lumped conveyance for water depth h
$A(h)$	Cross-sectional area for water depth h
$C(h)$	Chézy friction value for water depth h
$P(h)$	Wetted Perimeter for water depth h
$R(h)$	Hydraulic radius for water depth h ($R(h) = A(h)/P(h)$)

Here, the water depth h follows from a water level ζ and a bed level z. More details follow in section 8.7.7.

If the water level in an open tabulated cross section rises above the highest level in a cross-sectional profile, a vertical wall is assumed, as a result:

- the flow width is set to the width at the highest level.
- the flow area is set to the total flow area of the cross section at the highest point plus the flow width at the highest level * water height above the highest point in the cross section definition.
- The wet perimeter is set to the wet perimeter at the highest point of the cross section plus 2 * the water height above the highest point in the cross section.

For a closed cross section the flow area and wet perimeter are equal to the corresponding highest level in the cross sectional profile.

8.7.6 1D friction definition

The previous section on conveyance uses friction coefficients for its calculations. The input of these friction values typically varies between the various locations in a 1D network. There are two places in the model input files where these values can be defined:

- Friction values (and types) directly in the cross section definition file.
- Friction values (and types) in friction files (separate of the cross sections), with the possibility to vary on the network branches.

These two types can also be combined in one model schematization. The following subsections contain further details.

The friction values are defined on certain locations on the network, but for conveyance calculations friction values are needed on other locations in the computational grid. The interpolation choices used for this are further explained in section 8.7.7.

8.7.6.1 Friction definition with cross section definitions

All available cross section definitions allow direct specification of the friction properties. This is done using the keywords `frictionType(s)` and `frictionValue(s)` in the cross section definition file section C.16.1.

Alternatively, a cross section definition may refer to separate friction definitions, using keyword frictionId(s). In this case, each referenced frictionId should be present in one of the friction files, described in the following section.

Friction sections

Friction sections allow for different friction values in subsegments of a single cross section definition. They are defined using multiple frictionIds (or frictionValues and frictionTypes) for a single cross section. This is only possible for the following cross section definition types: tabulated river, xyz and yz. All other cross section definitions only offer the singular keywords frictionId, frictionType and frictionValue.

8.7.6.2 Friction definition in friction files

When friction values are not directly specified in the cross section definitions, they may alternatively be specified on the network branches or in named friction classes. This is done using MDU keyword FrictFile under [geometry], and all definitions must then be placed in one or more 1D roughness definition INI (friction) files (section C.15).

Each friction file defines one or more frictionIds with a global friction value under [Global]. When defining more than one frictionId in a single file, only the use of [Global] is allowed, not [Branch]. An example use of this is to define multiple friction 'classes' for various materials, e.g., pvc or concrete. When multiple frictionIds are needed that have friction values on branches, each frictionId must be placed in its own friction file.

In addition to one [Global] value, different friction values may also be defined on branch locations under [Branch] blocks using keywords branchId and chainage. Spatially varying friction *within* a branch is defined using numLocations > 1, and the friction interpolation settings are listed in Table C.21.

Time-dependent friction

Also, time-dependent friction values on branches are possible by supplying a <*.bc> file with the time series for the friction coefficient. A branch with functionType = timeSeries can refer to exactly one timeSeriesId, which must be an existing name in that <*.bc> file. As a result the friction coefficient is constant along the branch and varying in time.

In section 8.7.5.1 is described how a conveyance table for YZ- and XYZ-cross section is computed. When the friction values are time-dependent, these tables need to be recalculated. The generation of a conveyance table is time consuming, therefore it is recommended to limit the number of updates.

In the MDU file the parameter UpdateRoughnessInterval under [time] can be specified (default is 1 day). The time dependent friction parameters are updated at the start of the computation and subsequently at UpdateRoughnessInterval intervals. This offers the user the possibility to limit the number of updates of the friction parameters and as a result also the number of conveyance tables that have to be recomputed is reduced. In case of a (X)YZ-cross section with one or more friction sections with time dependent friction values, two conveyance tables are computed, one for the start of the interval (with the corresponding friction parameters for this time instance) and one for the end of the interval (UpdateRoughnessInterval later). At intermediate times during the computation, a linear time interpolation of the conveyance is performed. For all the other types of cross sections the friction values are determined using a linear interpolation in time, using the friction values at the beginning of the update interval and at the end of the update interval, and the conveyance is directly calculated using that interpolated friction value.

8.7.7 Interpolation of cross sections

Cross sections can be placed at arbitrary locations on the branches of a 1D model. As a result a cross section might not be available at the required location for the computational core. In those cases an interpolation is required. The interpolation method is explained, using the example network from Figure 8.27

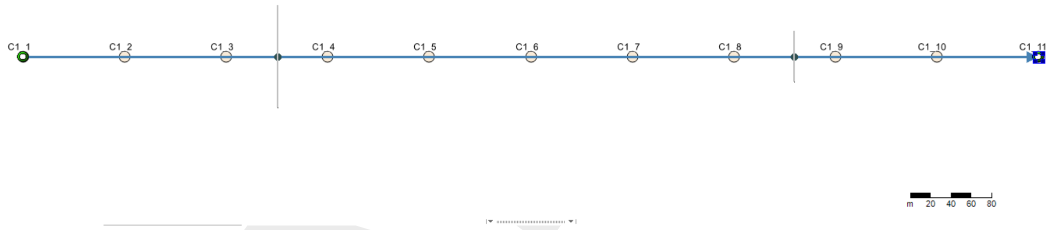


Figure 8.27: *Example of a 1D-model.*

This example contains 2 cross sections and eleven water level points.

The bedlevels are treated differently from the cross section dimensions. The interpolation of a circular cross section is different from the other cross section types. And also the conveyance is treated in a different way.

Remark:
- ◇ In D-Flow FM the user can specify interpolation over branches. The interpolation methods, described in this subsection, still holds for this case.

Interpolation of bed levels

The bed levels are interpolated to the water level points. In this example:

- ◇ C1_1 until C1_3 get the same bed level as the bed level of the first cross section.
- ◇ The bed levels for C1_4 until C1_8 will be the result of a linear interpolation between both cross sections.
- ◇ C1_9 until C1_11 get the same bed level as the bed level of the second cross section.

The absolute bed levels at cross section locations is determined by the absolute bed level of the cross section definition (in [m AD]) plus an optional shift at the cross section location (in [m]).

Interpolation of cross sectional flow data

The flow area and wet perimeter is required at the flow velocity points.

The computational core calculates a water depth for a velocity point. In the next step the flow area and the wet perimeter for the two adjacent cross sections are calculated, using the water depth from that velocity point. The two different values (both for flow area and wet perimeter) are then linearly interpolated back to the velocity point.

Interpolation of cross sectional total data

The volume of a 1D grid cell is calculated by adding up half the volume in each connected flow link: $0.5\Delta x A_T(h)$, and any additional storage volume in the node itself.

The water depth is calculated by subtraction of the bed level from the current water level. This water depth is used to calculate the total area and total width for the two adjacent cross sections. The two values for each item are then interpolated to the **velocity point**.

Interpolation of circular cross sections

Circular cross sections are handled differently from other cross sections. Only the diameter is interpolated from adjacent cross sections to the velocity point. The quantities total and flow area and wet perimeter are then calculated once using that interpolated diameter and the local water depth.

Calculation of the conveyance for YZ-type cross sections

In addition to the cross section data, also the Chézy value is required for the calculation of the conveyance. The user defined roughness values can be spatially varying, in the 1D roughness files (section C.15). In particular, these roughness locations need not be the same as the cross section locations. Therefore, the equivalent Chézy value is interpolated from the roughness locations to the location of the YZ cross sections. The conveyance is then calculated for the two adjacent cross sections of a velocity point. These conveyance values for the two cross sections are then interpolated to the velocity point.

Calculation of the conveyance for non-YZ-type cross sections

The conveyance at a velocity point for non-YZ-type cross sections is based on the interpolated flow area and wet perimeter, as described before. The Chézy value is directly interpolated from the roughness locations to the velocity point (i.e., not first to the cross section locations). Subsequently, these parameters are used to calculate the conveyance at the velocity point.

8.8 Drying and flooding

Estuaries and coastal embayments contain large, shallow, and relatively flat areas separated and interleaved by deeper channels and creeks. When water levels are high, the entire area is covered by water. But as tide falls, the shallow areas are exposed, and ultimately the flow is confined only to the deeper channels. The dry tidal flats may occupy a substantial fraction of the total surface area. The accurate reproduction of covering or uncovering of the tidal flats is an important feature of numerical flow models based on the shallow water equations.

Many rivers have compound channels, consisting of a main channel that always carries flow (the summer bed) and one or two flood plains which only convey flow during extreme river discharges (the winter bed). The summer bed is surrounded by low dykes, which could be overtopped if the river discharge increases. The winter-bed is surrounded by much higher dykes, which are designed to protect the polders against water levels due extreme river discharges. The flooding of the flood plains increases the drainage capacity of the river and reduces the local water level gradients.

In a numerical model, the process of drying and flooding is represented by removing grid points from the flow domain that become *dry* when the tide falls and by adding grid points that become *wet* when the tide rises. Drying and flooding is constrained to follow the sides of grid cells. In this section, we specify the algorithms which have been used to determine the moment when a grid cell (water level point) or cell boundary (velocity point) becomes dry or wet. Drying and flooding gives a discontinuous movement of the closed boundaries and may generate small oscillations in water levels and velocities. The oscillations introduced by the drying and flooding algorithm are small if the grid sizes are small and the bottom has smooth gradients.

Essential elements of the wetting and drying algorithm are the definition of the water level, the definition of the bed level and the criteria for setting a velocity and/or water level point wet or dry. In the following subsections, these three items will be discussed.

8.8.1 Definitions

In section 8.3.1, the locations of the primary flow variables (water level and flow velocity) have been described. Through Figure 8.3 it has been clarified that the water level is computed at the location of the circumcenter (cell center), whereas the face normal flow velocity is computed at the midpoint of each cell face (face center).

For the computation of these two primary variables, the level of the bed must be known both at the cell center and at the face center. The user can specify the way these values are interpreted from the available bathymetry by means of the MDU-file keywords `BedlevType`, `DpuOpt` and `conveyance2D`. For a proper understanding of the possibilities of D-Flow FM, Figure 8.28 is provided with a three-dimensional representation of two adjacent triangular cells.

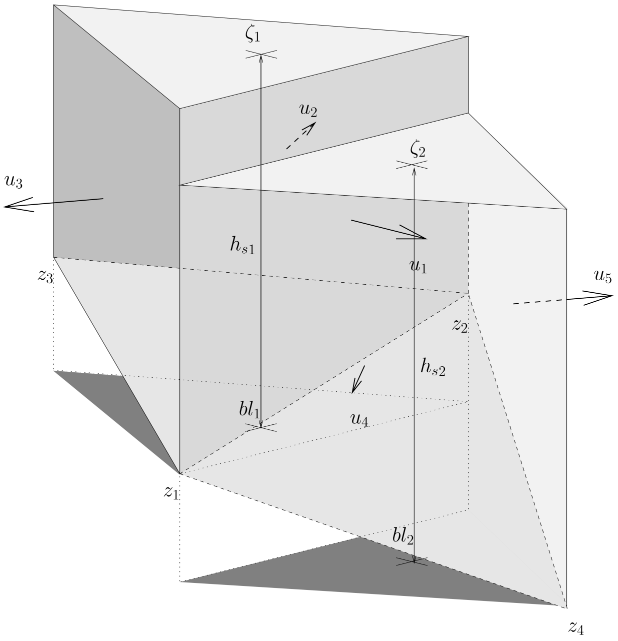


Figure 8.28: Definition of the water levels, the bed levels and the velocities in case of two adjacent triangular cells.

To the keyword `BedlevType`, the values `1`, `2`, `3`, `4`, `5` and `6` can be assigned. For the keyword `conveyance2D`, the values `-1`, `0`, `1`, `2` and `3` are available. The keyword `DpuOpt` can be either `1` or `2`; the latter option is currently only available in combination with D-Morphology.

The most common case is the definition of the bed levels at the corner nodes of the cell and a choice for the keyword `BedlevType = 3`. Depending on the choice for `conveyance2D`, the face center bed levels and cell center bed levels are computed.

8.8.1.1 Piecewise constant approach for the bed level

In principle, the bed levels are considered as piecewise constant in space. This approach is followed if `conveyance2D` < 1. The keyword `BedlevType` can be used in combination with the piecewise constant approach as highlighted in the table below. Note that the bed level is defined with respect to the reference level (not to be confused with a water depth given as a positive value downwards). With `conveyance2D` < 1, we have:

Keyword	**Value**	**Cell center bed level**	**Face center bed level**
`BedlevType`	1	user specified	the highest[3] cell center bed level considering the two cells next to the face
`BedlevType`	2	the lowest face center bed level considering all the faces of the cell	user specified
`BedlevType`	3	the lowest face center bed level considering all the faces of the cell	the mean corner bed level considering the two corner nodes the face is connecting
`BedlevType`	4	the lowest face center bed level considering all the faces of the cell	the lowest corner bed level considering the two corner nodes the face is connecting
`BedlevType`	5	the lowest face center bed level considering all the faces of the cell	the highest corner bed level considering the two corner nodes the face is connecting
`BedlevType`	6	the mean corner bed level considering all the corners of the cell	the highest[3] cell center bed level considering the two cells next to the face

The former Delft3D-FLOW version, running on curvilinear meshes, has utilized the piecewise constant approach for bed levels as well. The treatment of the bed level itself is, however, different from D-Flow FM. In Delft3D-FLOW, the user can distinctly specify the treatment type for the cell center bed level and the face center bed level. In D-Flow FM, the choice for the one implies the choice for the other. A strict, *exact* match of settings for which Delft3D-FLOW and D-Flow FM treat the bed similarly, is not facilitated. Hence, the user himself should take care in comparing the Delft3D-FLOW results and D-Flow FM results when it comes to the bed level treatment settings.

[3] This is the default behaviour in case `DpuOpt`=1. When `DpuOpt`=2, the mean value of the two cells next to the face is used. This option is only available in combination with D-Morphology.

8.8.1.2 Piecewise linear approach for the bed levels

A piecewise linear bed level approach can be chosen for through setting `conveyance2D` ≥ 1. In this case, only the approach for `BedlevType` equal to 3, 4 and 5 is affected. With `conveyance2D` ≥ 1, we have:

Keyword	Value	Cell center bed level	Face center bed level
`BedlevType`	3, 4, 5	the lowest corner node bed level considering all the nodes of the cell	linearly varying from the bed level at the one corner node to the bed level at the other corner node

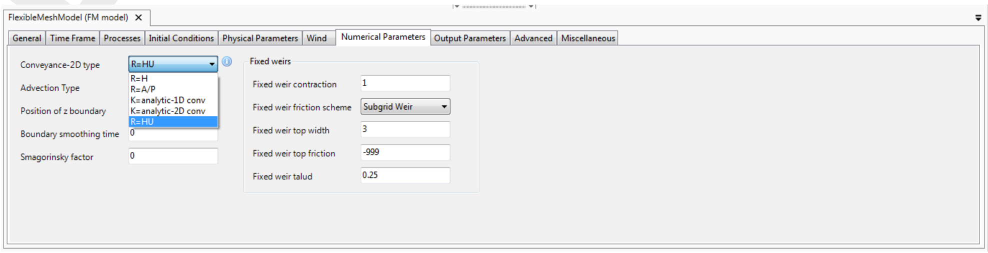
Notice that the choice for either `BedlevType` equal to 3, 4 or 5 does not imply different bed level treatment approaches. For the case `conveyance2D` ≥ 1, the bed is assumed linearly varying within a face only to compute the wet cross-sectional area of the vertical face; it should, however, be remarked that for the computation of the water column volume in a cell, this linear variation of the bed is *not* taken into account.

8.8.1.3 Hybrid bed level approach

In addition to the previously described bed level treatment approaches, D-Flow FM facilitates a hybrid approach for the computation of the cell center bed level by means of the keywords `blminabove` and `blmeanbelow`. In this approach, the cell center bed level is computed as the mean of the associated face center bed levels, be it only below the user specified level `blmeanbelow`. For levels above the user specified level `blminabove`, the minimum value of the associated face center bed levels is used. In between these two user specified levels, the bed levels are constructed by means of a linear interpolation between the two approaches' results.

8.8.2 Specification in the User Interface

The specification of the treatment of the bed level locations can be achieved through the tab fields 'Numerical Parameters' and 'Physical Parameters'.

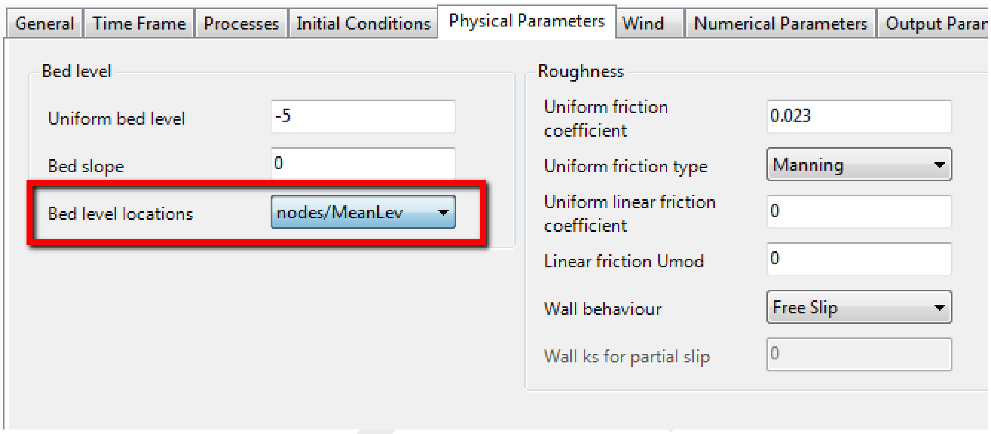


Conveyance-2D type: R=HU
Advection Type
Position of z boundary
Boundary smoothing time: 0
Smagorinsky factor: 0

Fixed weirs
Fixed weir contraction: 1
Fixed weir friction scheme: Subgrid Weir
Fixed weir top width: 3
Fixed weir top friction: -999
Fixed weir talud: 0.25

Figure 8.29: *Specification of the conveyance option in the User Interface.*

The tab field 'Numerical Parameters' contains the choice for the conveyance options: see Figure 8.29).



Bed level

Uniform bed level: -5

Bed slope: 0

Bed level locations: nodes/MeanLev

Roughness

Uniform friction coefficient: 0.023

Uniform friction type: Manning

Uniform linear friction coefficient: 0

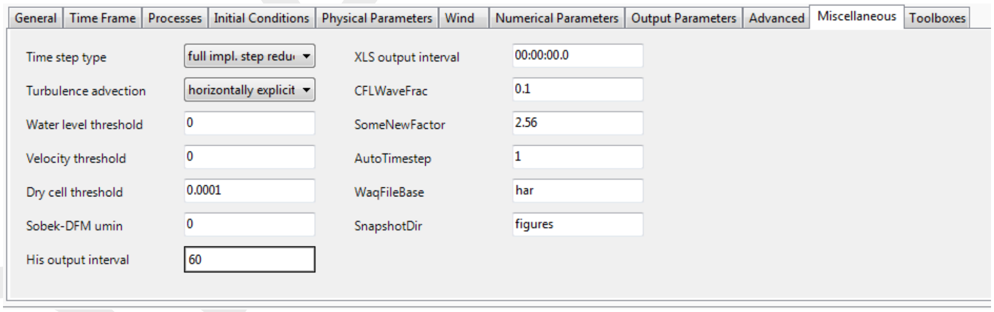
Linear friction Umod: 0

Wall behaviour: Free Slip

Wall ks for partial slip: 0

__Figure 8.30:__ Specification of the bed level treatment type in the User Interface.

The tab field ‘Physical Parameters’ contains the choice for the Bed level locations. Five options are facilitated: `BedlevType` numbers 1 up to 5; the option 6 is not facilitated in the user interface.



Time step type: full impl. step redu

Turbulence advection: horizontally explicit

Water level threshold: 0

Velocity threshold: 0

Dry cell threshold: 0.0001

Sobek-DFM umin: 0

His output interval: 60

XLS output interval: 00:00:00.0

CFLWaveFrac: 0.1

SomeNewFactor: 2.56

AutoTimestep: 1

WaqFileBase: har

SnapshotDir: figures

***Figure 8.31:** Specification of the hybrid bed options (with keywords `blminabove` and `blmeanbelow`).*

By means of the tab field ‘Miscellaneous’, the hybrid options with the keywords `blminabove` and `blmeanbelow` can be enabled.

8.8.3 Dry/wet criterion at cell faces

The parameter ε_{hu} (c.f. `Epshu` Table A.1) describes the threshold depth at cell faces. When the water depth at cell faces exceeds this ($h_u > \varepsilon_{hu}$) value the cell is considered wet and otherwise it is dry.

8.8.4 Dry/wet criterion at cell circumcenters

A research keyword `TestDryingFlooding` (Table A.1) is available for determining if the water depth is positive at cell circumcenters. To compute this, a variable d_{trsh} is introduced and when the water depth exceeds this value a cell is considered to be wet ($h > d_{trsh}$).

The default value is `TestDryingFlooding = 0` (D-Flow FM), and considers the threshold for positivity as $d_{trsh} = 0$. Option 1 is the Delft3D 4 default:

$$d_{trsh} = \max(10^{-9}, \min(\varepsilon_{hu}, 10^{-3})).$$

The last option 2 sets ($d_{trsh} = 0$) and furthermore considers a minimum volume for the transport solver based on the product of ε_{hu} and the flow cell area. The last setting is introduced to improve the stability of the transport computations, and can be activated if the default option leads to instabilities in the constituents.

8.9 1D2D coupling

The discretization of the model domain and the shallow water equations defined on that domain can be done in one, two and three dimensions. In a combined 1D2D model, some parts are modelled as 1D branches (e.g., channel flow or pipe flow) and other parts as 2D horizontal grids. The coupling between these 1D and 2D domain parts can use several 1D2D link types, depending on the application.

The following 1D2D link types can be distinguished:

1D2D lateral A sideways link between the 1D channel and 2D flood plains on the side(s), for example for rivers. 1D and 2D domain parts do not overlap. Further described in section 8.9.3.

1D2D embedded Very similar to the lateral link, but intended for smaller 1D channels, where the 2D grid cells overlap the 1D channels, for example in rural or marshland areas. Also described in section 8.9.3.

1D2D longitudinal The 1D channel is linked to the 2D grid part in the flow direction (i.e., not lateral), for example to model an upstream 1D river feeding into a 2D river or estuary. Further described in section 8.9.4.

1D2D vertically stacked A vertical link from 2D planes to 1D underground sewer networks. Further described in section 8.9.5.

8.9.1 1D2D model input

A coupled 1D2D model should contain the definition of the 1D2D links as "mesh contacts" in the network netCDF file. The details are described in section B.2.11.8.

All other model input that normally applies to regular flow links, can in principle also be applied to 1D2D links. For example, an observation cross section (section 5.4.2.3), or a fixed weir polyline (section 14.1) is allowed to cross 1D2D links as well.

8.9.2 Definitions

The following sections make intensive use of the mesh topology definitions in (Deltares, 2024a, section 6.1). Also of importance are the concepts of "bed level type" and "conveyance type", introduced in the same Technical Reference in Sections 6.1.2 and 6.2.2, respectively. The definitions below are specific for the 1D2D context.

Topology and connectivity

The following definitions are used for the connectivity at 1D2D links: which neighbouring cells (1D and 2D) are involved and which mesh nodes.

- 1D2D link j is a special type of flow link that connects a 1D mesh node with a 2D mesh cell,
- for longitudinal links, link j refers to two mesh nodes $l(j) = l(j')$ and $r(j) = r(j')$, where j' is the 2D vertical face (i.e., the cell edge) that is crossed by 1D2D link j,
- for all other types of 1D2D links, link j refers to two mesh nodes $l(j)$ and $r(j)$ that are simply the two end nodes of the corresponding mesh edge,
- 1D2D link j contains neighbors cells $L(j)$ and $R(j)$, corresponding to the start and end point, respectively. The side that connects to the 2D cell refers to a 2D mesh cell, and the other side to the 1D cell around the 1D mesh node.
- 1D2D link j contains neighbors cells $L(j)$ and $R(j)$, corresponding to the start and end point, respectively. The side that connects to the 2D cell refers to a 2D mesh cell, and the other side to the 1D cell around the 1D mesh node.
- For convenience, new notation is introduced to directly refer to the 1D cell and 2D cell of a 1D2D link: $C_{1D}(j)$ and $C_{2D}(j)$, respectively.

Bed levels

The effective bed levels depend on the type of 1D2D connection, therefore those details can be found in the following sections. Here, we only introduce the concept of *preliminary cell- or node-based bed levels* $bl_{*,l/r(j)}$, which are defined as:

$$bl_{*,l/r(j)} = \begin{cases} bl_{L/R(j)}, & \text{bed level type} \in \{1, 6\},\\ z_{l/r(j)}, & \text{bed level type} = 3,\\ bl_{u_j}, & \text{otherwise.} \end{cases} \tag{8.68}$$

Storage volume

The storage volume in a 1D2D link is contributed half to the left and half to the right mesh cell, as follows:

$$\begin{aligned} V_{L(j)} &\mathrel{+}= 0.5\Delta x_j A_{1j},\\ V_{R(j)} &\mathrel{+}= 0.5\Delta x_j A_{2j}, \end{aligned} \tag{8.69}$$

where Δx_j is the link length. This length depends on the type of 1D2D link, therefore those details can be found in the following sections. Next, the face-based cross-sectional areas $A_{1/2_j}$ are defined as the cross sectional area for the two face-based water depths, respectively:

$$\begin{aligned} h_{1j} &= \zeta_{L(j)} - bl_{1j},\\ h_{2j} &= \zeta_{R(j)} - bl_{2j}. \end{aligned} \tag{8.70}$$

The area calculation for $A_{1/2_j}$ depends on the cross sectional shape, which differs for the various types of 1D2D links. More details can be found in the following sections.

Remark:

- The face-based cross-sectional areas $A_{1/2_j}$, which determine storage, differ slightly from the wet flow area A_{u_j}, which determines conveyance. The first is based on the water depth from Equation (8.70), whereas the latter is based on water depth h_{u_j}, which is the water depth based on an upwind reconstruction (Deltares, 2024a, Equation (6.14)).

8.9.3 1D2D internal connections (lateral and embedded)

The 1D2D lateral and embedded connections serve two purposes, but are treated numerically in one and the same way: via so-called 1D2D *internal* links. Both can be used to model 'sideways' exchange of water between a 1D channel and neighbouring 2D grid cells. The difference between lateral and embedded connections is that for lateral connections, the 2D grid cells are supposed to *not* overlap with the 1D computational cells, whereas for embedded connections the 2D grid cells overlap the 1D channels. This may result in double storage above the extent of the 1D computational cells, but this may sometimes be preferrable over fully aligning 2D model grids with relatively small 1D channels.

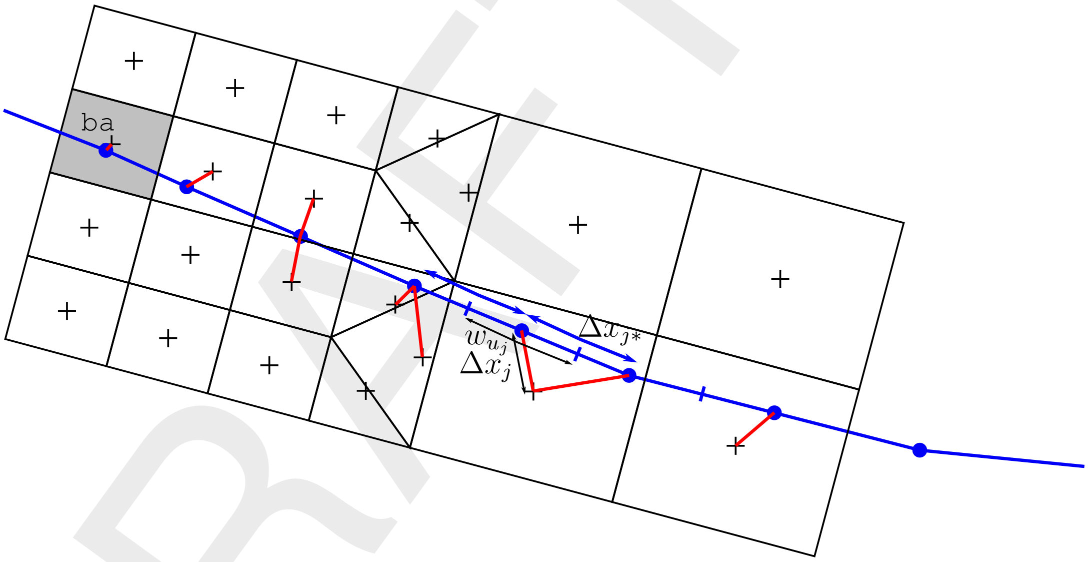


Figure 8.32: *Discretization of internal 1D2D links.*

Lengths and widths

The distance between the 1D and 2D pressure points for an internal connection j is defined as:

$$\Delta x_j = \max\left(\|\boldsymbol{x}_{R(j)} - \boldsymbol{x}_{L(j)}\|,\ 0.5\sqrt{\mathtt{ba}}\right) \tag{8.71}$$

where ba represents the area of the 2D grid cell connected to the 1D2D connection.

The width of a 1D2D internal flow link corresponds with either the "length" of the embankment on the 1D channel side or the width of the 2D cell. This length at the 1D channel side is approximated by the average of all regular 1D links connected to the 1D cell:

$$w_{u_{1D}} = \sum_{j^* \in \mathcal{J}_{1\text{D}}(j)} \Delta x_{j^*} / N_{1\text{D}}(j), \tag{8.72}$$

where the subset of regular 1D links neighbouring 1D2D link j is defined as:

$$\begin{aligned} \mathcal{J}_{1\text{D}}(j) &= \{ j^* \in \mathcal{J}(\mathcal{C}_{1\text{D}}(j)) \,|\, j^* \text{ is a regular 1D link} \}, & (8.73)\\ N_{1\text{D}}(j) &= |\mathcal{J}_{1\text{D}}(j)|. & (8.74) \end{aligned}$$

In case the 1D2D link intersects the 2D cell, the width ($w_{u_{2D}}$)at the 2D side of the 1D2D link is set to the width of the intersected face of the 2D cell, otherwise $\sqrt{ba}$ is used. Finally w_{u_j} is set to the minimum of $w_{u_{1D}}$ and $w_{u_{2D}}$.

Figure 8.32 shows the most common situation of a straight 1D channel: 1D2D link j has two neighbouring regular 1D links, so the width of j is half of the two flow link lengths of them added together.

Bed levels

The 1D and 2D grid cells connected by the link each begin with their own bed level, as defined in the usual way via the `BedlevType` keyword (see section 8.8.1). The flow link's bed levels are defined solely by the bed level of the 2D cell of the link:

$$
\begin{aligned}
bl_{1_j} &= bl_{C_{2D}(j)} \\
bl_{2_j} &= bl_{C_{2D}(j)}.
\end{aligned}
\tag{8.75}
$$

In the usual sense, the cell-based bed level on the left and right mesh cell are minimized by the flow link bed levels such that link bed levels are never below their connected cell's bed levels:

$$
\begin{aligned}
bl_{L(j)} &= \min(bl_{L(j)}, bl_{1j}), \\
bl_{R(j)} &= \min(bl_{R(j)}, bl_{2j}).
\end{aligned}
\tag{8.76}
$$

Storage volume

The storage volume in a 1D2D internal link differs from the usual approach in Equation (8.69). Instead, the link volume is contributed half to the 1D cell and a quarter to the 2D mesh cell, as follows:

$$
\begin{aligned}
V_{C_{2D}(j)} &\mathrel{+}= 0.5 \cdot 0.5 \Delta x_j A_{1j}, \\
V_{C_{1D}(j)} &\mathrel{+}= 0.5 \Delta x_j A_{2j},
\end{aligned}
\tag{8.77}
$$

The quarter volume contribution is explained in Figure 8.33. On the 1D side a rectangular control volume of $\frac{1}{2}\Delta x_j$ is used.[4] When multiple 1D2D links are connected to the same 2D mesh cell, the volume contribution of these 1D2D links on the 2D side ($\Delta V_{C_{2D}(j)}$) might overlap in the corners. Therefore, half the storage of such rectangles (or in fact: the depicted triangular volumes) is added.

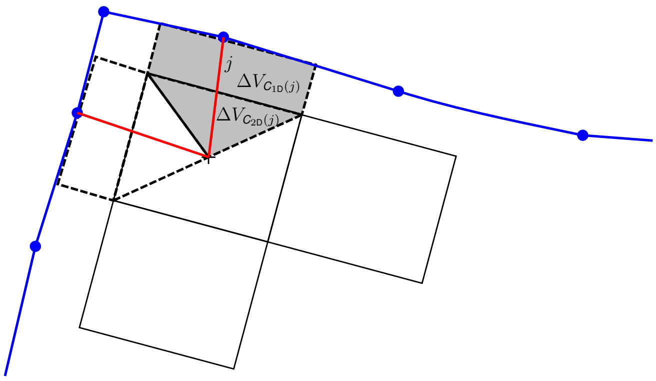


Figure 8.33: *Lateral 1D2D links and the volume contribution to the 1D and 2D cell.*

[4] Currently, the code has an additional factor $\frac{1}{2}$ also on the 2D side of the flow link, similar to the 1D side. This needs a closer look.

Face-based water depth h_{u_j}

The water depth on a 1D2D internal link j is defined as:

$$h_{u_j} = \zeta_{u_j} - \max(bl_{1_j}, bl_{2_j}) \tag{8.78}$$

Where the upwind water level ζ_{u_j} is defined according to (Deltares, 2024a, Equation (6.15)).

Wet cross-sectional area A_{u_j}

The wet cross-sectional area A_{u_j} is calculated as a rectangular cross section with width w_{u_j} and water depth h_{u_j}.

Remark:

- ◇ When MDU option `Conveyance2D` is switched on, the 2D bed slope is included in the area calculation. This is not yet documented here.

Advection treatment

For internal 1D2D connections advection scheme 8 is used, as described in Deltares (2024a). For 1D and 1D2D connections with type 8 advection scheme a flow limiter is used. This flow limiter is described in section 8.15.2.

In case fixed weirs are located on lateral 1D2D connections, it is also possible to use a weir formula to compute the flow over these connections. See section 14.1 for more information.

Conveyance

The conveyance on the 1D2D link is calculated as described in section 8.7.5.2. The Chézy roughness value is computed from a default Manning's roughness value of 0.035.

8.9.4 1D2D longitudinal connection

The 1D2D longitudinal connection type can be used to model 1D channel flow feeding into a downstream 2D domain part, or vice versa. The coupling is based on conveyance interpolation and the coupling is oriented perpendicular to the 2D grid cell edges, which should typically approximate the 1D channel flow direction.

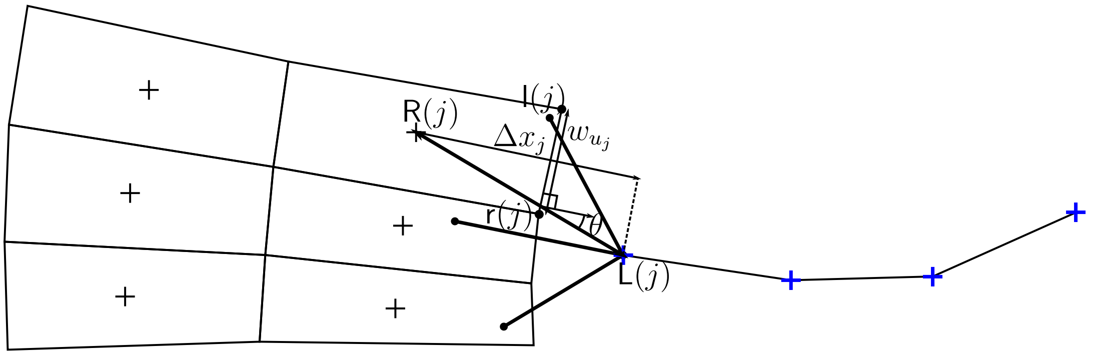


Figure 8.34: *Discretization of longitudinal 1D2D links.*

Lengths and widths

Longitudinal connections often come in the form of a 'fan-shaped' set of several 1D2D links ending up in the same number of 2D grid cells. The flow link length for each of them is intended to reflect 'unidirectional' flow perpendicular to the crossed 2D grid cell edge. As a result, the distance between the 1D and 2D pressure points for a link j is defined as:

$$\Delta x_j = \|\boldsymbol{x}_{R(j)} - \boldsymbol{x}_{L(j)}\| \cos\theta, \tag{8.79}$$

where θ is the angle between the direct 1D2D link $L(j) \rightarrow R(j)$ and the normal of the crossed 2D grid cell edge $l(j) \rightarrow r(j)$. This definition overrides the default length definition as it is used in (Deltares, 2024a, Figure 6.6).

The width (i.e., maximum flow width) of link j is defined as:

$$w_{u_j} = w_{u'_j}, \tag{8.80}$$

i.e., equal to the 2D grid cell edge's width. The latter is introduced in (Deltares, 2024a, Figure 6.4).

Bed levels

The 1D and 2D grid cells connected by the link each have their own bed level, as defined in the usual way via the `BedlevType` keyword (see section 8.8.1). The flow link's bed levels are defined by:

$$\begin{aligned} bl_{1_j} &= bl_{*,l(j)} \\ bl_{2_j} &= bl_{*,r(j)}, \end{aligned} \tag{8.81}$$

where the preliminary cell- or node-based bed levels $bl_{*,l/r(j)}$ were already defined in Equation (8.68). In other words: the flow link's bed levels are directly inherited from the left and right side. This overrides the 'face-based' bedlevel definition for regular flow links as given in (Deltares, 2024a, Equation (6.10)).

Remark:

◇ Note that the above definition 8.81 is not satisfactory for bed level type=1. It is not intended to use both the 1D and 2D bedlevels bl, but only the 2D bed level should be used. The code is subject to change on this aspect.

Face-based water depth h_{u_j}

The water depth on a 1D2D link j is defined as:

$$h_{u_j} = \begin{cases} \zeta_{u_j} - \min(bl_{1_j}, bl_{2_j}) & \text{if conveyance type} > 1 \\ \zeta_{u_j} - \max(bl_{1_j}, bl_{2_j}) & \text{otherwise,} \end{cases} \tag{8.82}$$

where the upwind water level ζ_{u_j} is defined according to (Deltares, 2024a, Equation (6.15)).

Wet cross-sectional area A_{u_j}

The wet cross-sectional area A_{u_j} is inherited from the 2D grid cell edge, and as such is computed according to Algorithm 5 in Deltares (2024a). This under the condition that all geometrical (re-)definitions in the preceding sections are used as input to the algorithm.

Advection treatment

TODO ***TODO 8.1:*** *Advection treatment*

8.9.5 1D2D vertically stacked connection

The 1D2D vertically stacked connection can be used to model the interaction between a 1D sewer system and the 2D domain above this sewer system. The vertically stacked connection can particularly be used for street inlets and manhole cover openings. Figure 8.35 shows two examples of this.

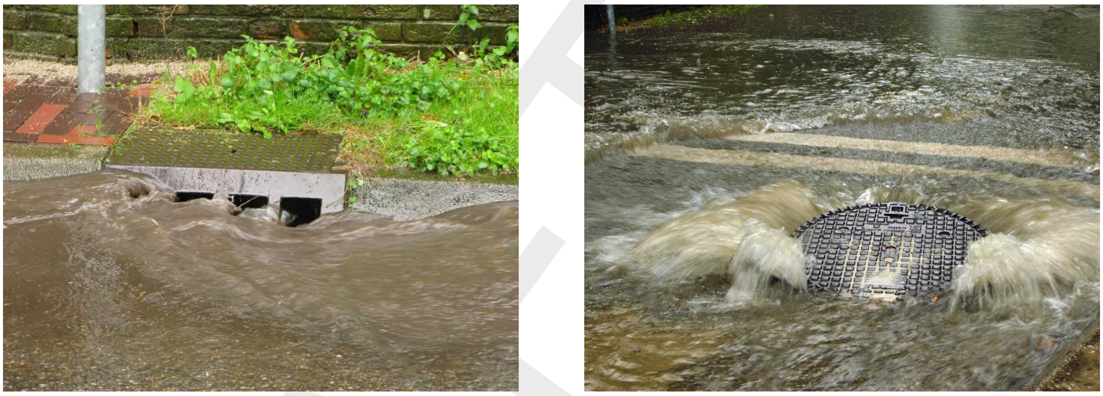


Figure 8.35: *Examples uses for 1D2D vertically stacked connections. Left: street inlet. Right: manhole cover. (Source: GAW/Stichting RIONED)*

Lengths and widths

The distance between the 1D and 2D pressure points for a connection j is defined as:

$$\Delta x_j = \max\left(\|\boldsymbol{x}_{R(j)} - \boldsymbol{x}_{L(j)}\|,\ 0.5\sqrt{\mathtt{ba}}\right) \tag{8.83}$$

where `ba` represents the area of the 2D grid cell connected to the 1D2D connection.

By default, all street inlets will have the same rectangular cross section with a width of 0.2 m and height of 0.1 m. These default global values for the dimensions can be changed by specifying `Uniformwidth1Dstreetinlets` and `Uniformheight1Dstreetinlets` in the `[geometry]` section of the MDU file. Alternatively, individual street inlets can be configured with different dimensions, using the 1D-2D link file, see section C.18.

In addition to street inlets, also manhole cover openings can be modelled by a 1D2D vertically stacked connection. The default shape for vertically stacked connections is rectangular. Even when the manhole opening would be circular, the connection is typically a vertical pipe, as opposed to street inlet-pipes, which are almost horizontal. As a result, the flow from street level into the manhole will occur simultaneously from all sides. Modelling this vertically stacked connection with a rectangular profile, i.e., with a flat bed, is a good choice, because the full profile width can start to flow once there is a bit of water at the street level. It is *not* recommended to set this rectangle's width to the circle circumference, since that will overestimate the pipe's capacity. The reason is that once the pipe flow is completely filling up the pipe, it is the pipe's conveyance that is limiting the flow.

A more sensible choice for the profile dimensions of the manhole cover is a width value that yields an equivalent shape in terms of lumped conveyance, i.e., with the same hydraulic radius.

Assuming a circle with radius r, the desired hydraulic radius equals:

$$R = A/P = \pi r^2/(2\pi r) = r/2. \tag{8.84}$$

A square profile of width w and the same height $h = w$, has hydraulic radius:

$$R = A/P = w^2/(4w) = w/4. \tag{8.85}$$

So, one could set the `openingWidth` and `openingHeight` both to $2r$, i.e., the circle diameter.

Bed levels

The 1D and 2D mesh cells that are connected by the connection each have their own bed level, as defined in the usual way via the `BedlevType` keyword (see section 8.8.1). The flow link's bed levels are defined by:

$$bl_{1j}/_{2j} = \begin{cases} z_{l/r(j)}, & \text{if and only if } z_{l/r(j)} \text{ is defined, and } l/r(j) \text{ is an endpoint, respectively.} \\ bl_{L/R(j)}, & \text{if and only if } L/R(j) \text{ is the 2D side, respectively,} \\ bl_{*,l/r(j)}, & \text{otherwise.} \end{cases} \tag{8.86}$$

where the preliminary cell- or node-based bed levels $bl_{*,l/r(j)}$ were already defined in Equation (8.68). In other words: the flow link's bed levels are directly taken from the mesh node levels, if they are available. Otherwise, the 2D side of the flow link gets the 2D bed level, and the 1D side gets the preliminary cell- or node-based bed levels, which effectively comes down to the 1D mesh node level for most models. This overrides the 'face-based' bedlevel definition for regular flow links as given in (Deltares, 2024a, Equation (6.10)).

In the usual sense, the cell-based bed level on the left and right mesh cell are lowered (if needed) to the flow link bed levels such that link bed levels are never below their connected cell's bed levels:

$$\begin{aligned} bl_{L(j)} &= \min(bl_{L(j)}, bl_{1j}), \\ bl_{R(j)} &= \min(bl_{R(j)}, bl_{2j}). \end{aligned} \tag{8.87}$$

Storage volume

The storage volume in a 1D2D vertically stacked link is computed slightly different from the standard way in Equation (8.69). Only the 1D mesh node receives the regular storage contribution (that is, half the link length), and the 2D mesh cell does not. The code is subject to change on this point.

The cross sectional area is computed for a rectangular shape, with the above-mentioned width and height.

Remarks:

- When the nonlinear "Nested Newton" option is switched off ($\texttt{nonlin1d} < 2$) the cross section is not a closed rectangle, but an open rectangular-shape channel. As a result, this might lead to undesirable large storage in the 1D2D pipes in the sewer network, since the pipe's storage area contribution to the 1D mesh node ($= 0.5 w_{u_j} \Delta x_j$) can store water even above the pipe diameter.
- Additionally, for numerical robustness, some extra storage is added by a Preissmann slot of uniform width. The default value is 0.001 m, and this can be changed by setting the MDU keyword `Slotw1D` under `[geometry]`.
- The option `setHorizontalBobsFor1d2d` in MDU section `[numerics]` makes it possible to override the bed levels of the flow links by:

$$bl_{1_j} = bl_{2_j} = \max(bl_{1_j}, bl_{2_j}). \tag{8.88}$$

As a result, not any storage in the 1D2D sewer connection pipe is applied until the water level in the sewer network exceeds the bed level of the 2D area.
This setting has no effect on the friction calculation, since the face based water depth h_{u_j} is already equal to $\zeta_{u_j} - \max(bl_{1_j}, bl_{2_j})$.

Face-based water depth h_{u_j}

The water depth on a 1D2D vertically stacked link j is defined as:

$$h_{u_j} = \zeta_{u_j} - \max(bl_{1_j}, bl_{2_j}) \tag{8.89}$$

Where the upwind water level ζ_{u_j} is defined according to (Deltares, 2024a, Equation (6.15)).

Wet cross-sectional area A_{u_j}

The wet cross-sectional area A_{u_j} is calculated for a rectangular cross section using the minimized water depth $\min(h_{u_j}, \text{pipe height})$.

Advection treatment

For vertically stacked connections advection scheme 8 is used, as described in Deltares (2024a). For 1D and 1D2D connections with type 8 advection scheme a flow limiter is used. This flow limiter is described in section 8.15.2.

Conveyance

The conveyance on the 1D2D link is calculated as described in section 8.7.5.2. The Chézy roughness value is computed from a default Manning's roughness value of 0.035.

8.10 Intakes, outfalls and coupled intake-outfalls

Many engineering studies concern the design of intakes and outfalls. For instance, studies about positioning of waste water diffusers or coupled intake-outfall design for cooling water at power plants. Intakes and outfalls can be modeled using sinks and sources. A source is a point in the model where a discharge Q in [m^3/s] is prescribed by a time series ASCII file (section C.4) with at least two columns and, depending on the model settings, possibly more. For each optional constituent that is modelled, an extra column is required. Best known constituents are salinity and temperature and the values in the time series file should be specified as salinity increase in [ppt] and temperature increase in [°C].

1. When salinity and temperature are not used in the computation:
   Model expects two columns, where the first column is the time in minutes, the second is the discharge in [m^3/s].
2. When either salinity or temperature is used in the computation:
   Model expects three columns, where the first column is the time in minutes, the second is the discharge in [m^3/s], the third column contains either the salinity (increase) in [ppt] or the temperature (increase) in [°C]. Note that third column can be either salinity or temperature depending which constituent is used in the model and which is not.
3. In general, when other constituents are also modelled (e.g. one or more sediment fractions, passive tracers), the column order is as follows: Time, Discharge, Salinity?, Temperature?, (Sed.frac 1, . . ., Sed.frac n)?, Spiral flow intensity?, (Tracer 1, . . ., Tracer n)?.

The location of the source or sink is specified in a polyline file (section C.2), containing a polyline with either multiple points or just one point. If two or more points are specified, a coupled source sink pair is made, the first point is the FROM (or sink) point, the last point is the TO (or source) point. Three variants may occur:

1. Sink point side lies inside a grid cell A, source point also lies inside a grid cell B (A may equal B, but that is rarely useful): water is extracted from cell A and transported to B.
2. Sink point lies outside of the grid, source point lies in a grid cell B: water is discharged into B, a bit like an inflow discharge boundary condition.
3. Sink side lies inside a grid cell A, source side lies outside of the grid: water is extracted from A, a bit like an outflow discharge boundary condition.

Specifying a negative discharge value effectively interchanges the role of the source and sink points. If only one point is in the <∗.pli> file, it is assumed to be a source point. Specifying a negative discharge turns the source into a sink.

For 3D computations, the polyline file should have a third column with z-values. It is good practice to change the <∗.pli> file into a <∗.pliz> file. The z-values are used to determine in which vertical grid cell the source and/or sink lie. The layer number can vary in time in sigma models.

In the case of a coupled pair of sink source points (variant 1), the third and fourth column with the salinity and temperature specification are interpreted as delta salinity and delta temperature. So the values at the source point become the values at the sink point plus the specified delta values.

The sources and sinks are specified in the <∗.ext> file in a way similar to the boundary conditions:

```
QUANTITY=discharge_salinity_temperature_sorsin
FILENAME=chan1_westeast.pli  # A file 'chan1_westeast.tim', with same basename
                             # as the polyline should be present
FILETYPE=9
METHOD  =1
OPERAND =O
AREA    =1.5
```

The specified area in the last line of this file determines (together with the specified discharge) the amount of momentum uQ that is released at the source point. This is currently only implemented in advection scheme 33 (cf. Deltares (2024a) on Momentum advection). Omitting this line or specifying a zero area switches off the release of momentum at the source point. The direction of the discharged momentum is in direction of the last two points of the polyline. So a one point polyline is momentumless. Both in 2D and 3D, the momentum can only be directed in horizontal direction.

Sources and sinks are treated explicity in the numerical scheme. This implies that the actual discharged or extracted amounts of water are limited by the velocity Courant condition $Q\Delta t/V < 1$. Not doing so could lead to severe timestep restrictions. Consider for instance a specified extraction in case the extraction point has fallen dry. In that case, a real pump would not be able to extract water and the specified extraction is more likely an input error than an actual description of a physically feasible situation. So, we limit the specified discharges to the local velocity Courant restriction. By specifying observation cross-sections (section 5.4.2.3), one can compare prescribed discharges to the discharges that were actually realised in the model. In case of large differences, the discharge should probably be distributed over a larger number of gridcells, or the extraction channel that feeds the extraction point should be dredged.

When modelling freshwater inflow from a river into a sea, the vertical distribution of the discharge that one should specify depends on the amount of detail available for modelling the saline water - fresh water interface. If the river is modeled with sufficient detail, and the river is well mixed at the upstream model boundary, the mixing process can be part of the modelling and the river can be discharged over the whole vertical. If the mixing process cannot be resolved in the model, for instance if the river is modeled as a point discharge adjacent to a closed boundary, make sure that the whole river is discharged into the top layer or in the top layers, depending on the estimated thickness of the fresh water plume at the discharge cell. If one would have distributed the river discharge over the whole vertical, the size of the fresh water plume would be underestimated because of too much mixing at the river discharge cell.

8.11 Equations of state for the density

The density of water ρ is a function of salinity (s) and temperature (t).

In D-Flow FM we copied the implementation from Delft3D-FLOW of the formulation derived by Eckart (1958) that is based on a limited number of measurements dating from 1910 (only two salinities at 5 temperatures). In the original equation the pressure is present, but at low pressures the effect on density can be neglected.

Eckart formulation

The Eckart formulation is given by (Eckart, 1958):

Range: $0 < T < 40$ °C, $0 < s < 40$ ppt

$$\rho = \frac{1\,000 P_0}{\lambda + \alpha_0 P_0}, \tag{8.90}$$

where:

$$\lambda = 1\,779.5 + 11.25T - 0.074\,5T^2 - (3.80 + 0.01T)\,s, \tag{8.91}$$

$$\alpha_0 = 0.698\,0, \tag{8.92}$$

$$P_0 = 5\,890 + 38T - 0.375T^2 + 3s. \tag{8.93}$$

with the salinity s in [ppt] and the water temperature T in [°C].

The keyword that selects the equation of state in the mdu file is called Idensform. The influence of salinity and or temperature on water motion through the baroclinic pressure can be switched off by setting Idensform=0

UNESCO formulation

The UNESCO formulation is given by (UNESCO, 1981a):

Range: $0 < T < 40$ °C, $0.5 < s < 43$ ppt

$$\rho = \rho_0 + As + Bs^{3/2} + Cs^2 \tag{8.94}$$

where

$$\begin{aligned}\rho_0 = {} & 999.842\,594 + 6.793\,952 \times 10^{-2}T - 9.095\,290 \times 10^{-3}T^2 + \\ & + 1.001\,685 \times 10^{-4}T^3 - 1.120\,083 \times 10^{-6}T^4 + 6.536\,332 \times 10^{-9}T^5\end{aligned} \tag{8.95}$$

$$\begin{aligned}A = {} & 8.244\,93 \times 10^{-1} - 4.089\,9 \times 10^{-3}T + 7.643\,8 \times 10^{-5}T^2 + \\ & - 8.246\,7 \times 10^{-7}T^3 + 5.387\,5 \times 10^{-9}T^4\end{aligned} \tag{8.96}$$

$$B = -5.724\,66 \times 10^{-3} + 1.022\,7 \times 10^{-4}T - 1.654\,6 \times 10^{-6}T^2 \tag{8.97}$$

$$C = 4.831\,4 \times 10^{-4} \tag{8.98}$$

with the salinity s in [ppt] and the water temperature T in [°C].

Note: The UNESCO formulation is set as default.

Remarks:

- Equation (8.96) is known as the International Equation of State for Seawater (EOS80) and is based on 467 data points. The standard error of the equation is 3.6×10^{-3} kg/m^3 (Millero and Poisson, 1981).
- The Practical Salinity Scale (UNESCO, 1981b, Par. 3.2) and the International Equation of State are meant for use in all oceanic waters. However, these equations should be used with caution in waters that have a chemical composition different from standard seawater. In such waters, densities derived with the methods based on practical salinity measurements and the International Equation of State may deviate measurably from the true densities. However, in water masses different in composition from standard seawater the *differences* in densities derived by the new equations involve only very small errors.

Recommendation

The UNESCO formulae serve as an international standard. Further, the UNESCO formulae show the correct temperature of 4 degrees Celsius where fresh water has its maximum density. The latter is of importance for thermal stratification in deep lakes in moderate climate zones. Therefore we recommend the application of the UNESCO formulae and to select the Eckart formulation only for consistence with previous projects in which it has been used. At this moment the default is UNESCO.

Nevertheless, the UNESCO formulae have their limitations as they are based on the general properties of seawater mixed with fresh water. Particularly in cases where marginal density differences play a role, typically lakes, a variable mineral content of the water may create density differences not detected by just temperature and salinity. For such dedicated cases, you are warranted to check the accuracy of the UNESCO formulae against experimental density-relations derived from the (lake) water. In case of deviations or other constituents determining the water density, we then recommend to contact the Helpdesk for further assistance such as the implementation of a more dedicated density formulation.

8.12 Tide generating forces

Numerical models of tidal motion in coastal seas generally do not account for the direct local influence of the tide generating forces. The amount of water mass in these models is relatively small and the effect of these forces on the flow can be neglected. In that case, tidal motion can often be reproduced with sufficient accuracy just by prescribing the tidal forcing along open model boundaries.

In models covering larger seas or oceans, the contribution of the gravitational forces on the water motion increases considerably and can no longer be neglected. In a global model, open boundaries are absent and the tidal motion can only be induced by including tide generating forces. Because tidal forces only play a significant role in the larger models, and because the exact position of each computational point on earth must be known, we have implemented these forces only in combination with spherical coordinates. By default, tide generating forces is switched on in spherical models. You can switch it off by setting in the mdu file: `Tidalforcing = 0`

The tide generating forces originate from the Newtonian gravitational forces of the terrestrial system (Sun, Moon and Earth) on the water mass. The equilibrium tide is the tide that would result from the gravitational forces if a solid earth would be completely covered by an ocean of about 21.5 km deep, such that the wave propagation speed would match the speed of the celestial forces over the ocean surface. The interacting frequencies that result can be grouped as diurnal, semi diurnal and long periodic. The total number of tide generating frequencies that is evaluated in the computation is a tradeoff between required accuracy and computational cost. Per default, we use a set of 60 frequencies.

A smaller set of 11 main frequencies as used in Delft3D-FLOW can also be selected by setting in the MDU file :

```
Doodsonstart =  57.555
Doodsonstop  = 275.555
Doodsoneps   =   0.030
```

The full set of 484 components can be selected by specifying:

```
Doodsonstart =  55.565
Doodsonstop  = 375.575
Doodsoneps   =   0.000
```

For the set of 60 components these numbers are:

```
Doodsonstart =  55.565
Doodsonstop  = 375.575
Doodsoneps   =   0.030
```

The earth itself is deformed by the celestial forces, this is called the solid earth tide. An estimate of this influence is based on the work of Love (1927), is included in the total tide generating potential.

We have gratefully applied the Fortran subroutines written by E.J.O. Schrama. Details on the implementation of tide generating forces can be found in his lecture notes Schrama (2007).

In order to save computation cost, the tidal potential is not computed for each computational point, but on a grid with a resolution of 1 degree in both the South-North and West-East direction.

The tidal and surge motions of the seas and ocean also deform the earth , which is called tidal loading. Next to that, the water at some point is attracted by all moving water elsewhere on the globe. This is called self attraction. The combined influence of these two processes is implemented in a beta version.

8.13 Advection at open boundaries

In hydrodynamic models, it is common practice to apply high-resolution grids in domains of limited extent, for instance to investigate the influence of a dam design on local flow patterns in a coastal area. Then, the challenge is to prescribe appropriate boundary conditions for such a detailed model. Tidal motion is often important and therefore water levels have to be prescribed. If an overall model with good reproduction of water levels and velocities is available, then it is common practice to start with water levels from the overall model on the open alongshore boundary and on both cross-shore boundaries of the detail model. Then, often the reproduction of water levels is accurate, but the reproduction of flow velocities might be less accurate. In particular, the residual flow velocities might be inaccurate. Obtaining a good velocity reproduction by prescribing only water levels often only works for the larger models (e.g. > 30 km) so that bed friction, pressure gradient and acceleration can balance each other. In smaller models, however, this might not be the case. Even small water level differences on the open boundaries may introduce unrealistic high or low velocities. A remedy might be to apply velocity boundaries or other types of boundary conditions (Neumann-type, . . .) on the cross-shore boundaries. However, this often reduces reproduction quality of water levels or might introduce strange flow patterns. Also the existence of stratification may also lead to unrealistic currents near open boundaries.

Therefore, in D-Flow FM the keyword `Zerozbndinflowadvection` can be activated in order to suppress unphysical currents near open boundaries. This applies for all water level open boundaries. This keyword has to be specified in group [numerics] and the options are `Zerozbndinflowadvection`=0,1 or 2. A value of 0 results into the default Neumann approach for currents, yielding that the gradient in velocity at the open boundary is zero. This approach is the default and applied when keyword `Zerozbndinflowadvection` is not specified. Applying a value of 1 forces cell center velocities to zero on the open boundary at *inflow* for all water level boundaries. A value of 2 forces cell center velocities to zero on the open boundary for both *inflow and outflow* for all water level boundaries.

Another option is to apply a so-called *advection* boundary, which can be specified via keyword `uxuyadvectionvelocitybnd`, in group [numerics]. For this option, both the x- and y-components of the velocity vector at the boundary have to be specified in the boundary condition file, in bc format (`.bc`). To be more precise, one needs to prescribe the x- and y-components of the velocity vector in the circumcenter of the virtual boundary cells (outside the domain boundary). It is not necessary to rotate these components in the direction normal to some local boundary. They are from West to East and from South to North components. An advection boundary is only applied at inflow. At outflow, a Neumann-type approach is applied by using current from inside the model domain.

8.14 Storage nodes

Storage nodes can be used in the 1D parts of a model schematization to define custom storage dimensions. Two example uses for storage nodes are:

1 Urban manhole compartments, where bed levels, areas, street level and storage area are relevant.
2 Lumped storage areas (typically open water) tied to a 1D node, where the storage levels and areas may be provided in a table.

The file format details for the storage node file are described in section C.17. The sections below further describe the numerical details. The storage areas defined for these nodes are added to the normal 1D storage as defined by all connected 1D channels (or pipes) and their cross section shapes. When a model schematization also contains meteo forcings for precipitation, it is important to realize that the additional storage areas are *not* used to capture extra precipitation. Only the open water area of the grid cells (or 1D channels) is used for that.

8.14.1 Notation

The geometric and hydraulic parameters relevant for storage nodes are listed below:

$bl_{\mathrm{st},k}$	Bed level (or bottom) of storage node k;
$bl_{\mathrm{gr},k}$	Street (or ground) level of storage node k;
$A_{\mathrm{st}}(\Omega_k)$	Bottom area of storage node k (computational cell Ω_k);
$V_{\mathrm{st,max},k}$	Maximum storage volume of storage node k (without any pipes)

8.14.2 Urban manhole compartments

Manhole compartments on 1D nodes form a basic modelling element for all urban schematisations. A manhole is in fact a small reservoir that interconnects two or more sewer pipes or is an end node. Its geometry and storage volume is describe in the following subsections.

8.14.2.1 Storage volume for urban manhole compartments

The available storage volume in and above an urban manhole compartment is defined by the bed level $bl_{\mathrm{st},k}$ of node k, the street (or ground) level $bl_{\mathrm{gr},k}$ at that node and the bottom area $A_{\mathrm{st}}(\Omega_k)$ of the manhole node. The maximum storage volume of the node *within* the manhole compartment (without any pipes) equals:

$$V_{\mathrm{st,max},k} = (bl_{\mathrm{gr},k} - bl_{\mathrm{st},k}) \cdot A_{\mathrm{st}}(\Omega_k) \tag{8.99}$$

Next, the *type* of the storage node defines whether and how excess water can be stored above the street level. Two storage types can be chosen. Figure 8.36 shows the differences between them.

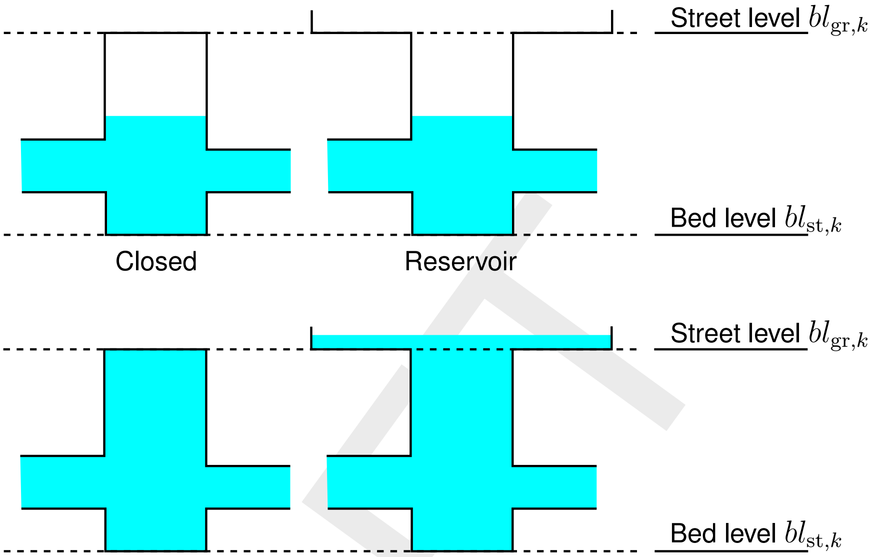

Street level $bl_{\mathrm{gr},k}$
Bed level $bl_{\mathrm{st},k}$
Closed
Reservoir
Street level $bl_{\mathrm{gr},k}$
Bed level $bl_{\mathrm{st},k}$

***Figure 8.36:** Difference between manhole storage nodes of type "closed" and "reservoir"*

- **Reservoir** In case the storage type "reservoir" is chosen, there will be an open connection between the manhole and the street surface. This means that, if the water level in the manhole exceeds the street level, it will inundate the "street storage area", which can be set by `streetArea`. Additionally, to easily switch from a 1D model with lumped street storage to a 1D2D model with excess storage on 2D grid cells at the street level, the global setting `useStreetStorage` can be changed to `false`. In this case, storage above ground level continues with the same bottom area $A_{\mathrm{st}}(\Omega_k)$.
- **Closed** In case the storage type "closed" is chosen, there will be no storage volume available above street level. For example, when a manhole has a fixed lid at the street level.

8.14.2.2 Network topology for urban manhole compartments

A single urban manhole may have multiple compartments. In that case, the network file in the model schematization must have one network node for each compartment. Figure 8.37 shows an example manhole with two compartments.

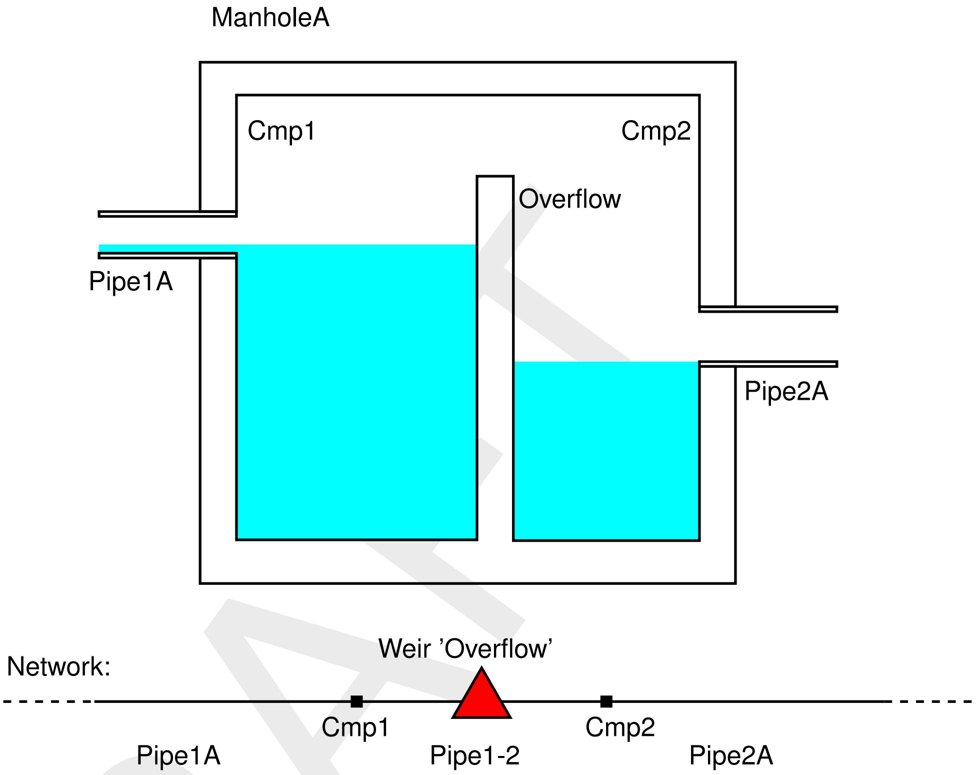


Figure 8.37: *Side view of a manhole with two compartments. At the bottom the matching network topology.*

The two sewer pipes become edges in the network and additionally, a third network edges must be present in the network to model the connection between compartment 'Cmp1' and 'Cmp2'. The type of connection between the compartments determines which type of hydraulic structure is best set on that edge. In this example, a sewer overflow structure is modelled as a weir with a crest level and a crest width. The two compartments become network nodes, and each network node will have a computational grid point placed on it.

The length of pipes is normally equal to the distance between the manhole compartment centers, but in this special case of a connection between two compartments within one manhole, it can be useful to override this Euclidean distance by a custom length. This edge length is defined as part of the network topology in the net file (see section B.2.4.2).

8.14.2.3 Vertical levels for urban networks

The vertical levels that define the placement of manhole compartments and connecting pipes originate from two types of input:

- ◇ A manhole compartment corresponds with a 1D pressure point and the bed level $bl_{\mathrm{st},k}$ and street (or ground) level $bl_{\mathrm{gr},k}$ are directly defined in the node file (section C.17).
- ◇ The invert levels of a sewer pipe are defined via the cross section location file (section C.16.2). The relative `shift` in that file is effectively the absolute invert level, because the closed cross sections (typically `circle` or `rectangle`) have no own absolute level (i.e., 0 m AD).

8.14.3 Storage tables for lumped water storage

Lumped water storage on 1D nodes can also be defined in a table that defines storage areas depending on water levels. The table uses blockwise interpolation.

$$i_{\mathrm{st},k}(\zeta) = \begin{cases} \max i|_{h_{\mathrm{st},k,i+1}<\zeta} & \text{if } \zeta > h_{\mathrm{st},k,2}, \\ 0 & \text{otherwise.} \end{cases} \tag{8.100}$$

$$V_{\mathrm{st},k}(\zeta) = \sum_{i=1}^{i_{\mathrm{st},k}(\zeta)} [(h_{\mathrm{st},k,i+1} - h_{\mathrm{st},k,i}) \cdot A_{\mathrm{st},k,i}] + (\zeta - h_{\mathrm{st},k,i_{\mathrm{st},k}(\zeta)}) \cdot A_{\mathrm{st},k,i_{\mathrm{st},k}(\zeta)} \tag{8.101}$$

Table 8.2: *Example storage table relating water levels to storage areas.*

$\mathbf{i = 1, N_{st}}$	$\mathbf{h_{st,k,i}}$	$\mathbf{A_{st,k,i}}$
1	-2.0	120
2	-1.5	950
3	4.3	1050

8.14.4 Manhole losses

In an urban model sewage water is transported through sewage pipes. In general, between these sewage pipes manholes are located. During the flow through a manhole from one pipe to another, an energy loss might occur. In this section it is explained how these losses can be added to a model.

The symbols, used in this paragraph are:

D_i Diameter of pipe i (see Figure 8.38).
v_i Flow velocity in pipe i (see Figure 8.38).
θ Angle between pipes 1 and 2 (see Figure 8.38).
ΔH Energy loss.
K_{loss_type} Loss coefficient.
g Gravity acceleration.

Subscript 1 denotes the pipe, where the flow is directed into the manhole. Subscript 2 denotes the pipe, where the flow is directed out of the manhole.

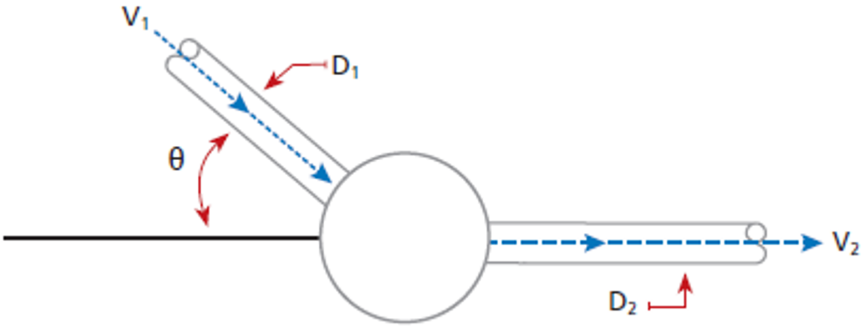


Figure 8.38: *Example of a manhole configuration. In this example the water from pipe 1 flows into the manhole and from the manhole the water flows into pipe 2*

During the flow through a manhole from one pipe to another, an energy loss might occur. There may be several causes for this energy loss. In D-Flow FM the following potential energy losses can be taken into account:

◇ *Exit loss*
The exit loss can be used to model the energy loss due to the expansion and mixing of the flow into the manhole.

$$\Delta H_{exit\_loss} = K_{exit\_loss}\left(\frac{v_1^2}{2g}\right) \tag{8.102}$$

◇ *Entrance loss*
The entrance loss can be used to model the energy loss due to the contraction of the flow from the manhole into the sewer pipe.

$$\Delta H_{entrance\_loss} = K_{entrance\_loss}\left(\frac{v_2^2}{2g}\right) \tag{8.103}$$

◇ *expansion loss*
The expansion loss can be used to model the expansion of the flow from pipe 1 to pipe 2.

$$\Delta H_{expansion\_loss} = K_{expansion\_loss}\left(\frac{v_1^2}{2g} - \frac{v_2^2}{2g}\right) \tag{8.104}$$

◇ *contraction loss*
The contraction loss can be used to model the contraction of the flow from pipe 1 to pipe 2.

$$\Delta H_{contraction\_loss} = K_{expansion\_loss}\left(\frac{v_2^2}{2g} - \frac{v_1^2}{2g}\right) \tag{8.105}$$

◇ *bend loss*
The bend loss is used to calculate the energy loss due to a change of direction of the flow, within a manhole. The user can specify a table that defines a piecewise linear function of the angle loss as a function of the angle θ, i.e. $K_{bend\_loss}(\theta)$

$$\Delta H_{bend\_loss} = K_{bend\_loss}(\theta)\left(\frac{v_1^2}{2g}\right) \tag{8.106}$$

In most cases only a subset of these types of loss terms will be applied, because some of them are interchangeble.

The total energy loss is computed by adding up all individual losses. An extra condition for the total energy loss is, that this total energy loss must not be less than $0.05v_2^2/(2g)$ and may not be greater than $v_1^2/(2g) + 0.5v_2^2/2g$.

Remarks:

- These losses can be applied at storage nodes of nodeType `compartment`.
- The default behaviour is that no manhole losses are applied.
- It is possible to define the loss coefficients for all `compartment` type storage nodes at once (*global*) and/or for a storage node specific. The individual defined loss coefficients will override the global loss coefficients for a specific manhole.
- The exit loss and the bend loss are taken into account on pipe 1.
- The entrance loss is taken into account on pipe 2.
- The components for expansion loss and contraction loss are put on either side of the manhole.
- Regarding the expansion and contraction loss for manholes with more than 2 pipes connected, a total flow area of the pipes that discharges into the manhole and the total flow area of the pipes that flows out of the manhole is computed. These two total areas determine if there is an expansion or contraction. Assume the flow velocity $v > 0$ in case the flow of a pipe is directed into the manhole. E.g. in case of an expansion the total loss is:

$$\Delta H_{expansion\_loss} = K_{expansion\_loss} \left(\sum_{j|v_j>0} \frac{v_j^2}{2g} - \sum_{j|v_j<0} \frac{v_j^2}{2g} \right) \tag{8.107}$$

- Regarding the bend loss for manholes with more than 2 pipes connected the bend losses are calculated in a different way. The pipe with the largest discharge from the manhole into the pipe is considered the reference pipe. The angle with respect to this reference pipe is determined for each pipe with a flow into the manhole, and the corresponding energly loss is applied on this same pipe.

8.15 Flow limiters

8.15.1 Slope limiter for 1D

This subsection describes the slope limiter for the pressure gradient in case the water level drops below the invert level of a channel or pipe.

The pressure gradient in Equation (8.3) and Equation (8.4) for 1D with constant density can be rewritten into

$$\frac{1}{\rho_0}\frac{\partial P}{\partial x} = \frac{\partial \zeta}{\partial x} \tag{8.108}$$

In the numerical discretization as summarized in (Deltares, 2024a, Algorithm 16) this term becomes:

$$\frac{\partial \zeta}{\partial x} \approx -\texttt{slope} = -\frac{\zeta_{L(j)} - \zeta_{R(j)}}{\Delta x(j)} \tag{8.109}$$

In sewer systems the bed level of a manhole can be lower than the invert levels of the incoming pipes, as is shown in Figure 8.37. This situation can also occur in open channel models, but this is less common, therefore we will focus on sewer systems here. In case the downstream water level drops below the invert level of the pipe, the pressure gradient in Equation (8.109) is not correct anymore. The downstream water level should in this case be equal to the downstream invert level of the pipe plus the critical depth. Since calculating the critical depth for an arbitrary shaped cross section requires a lot of computational effort the critical depth is neglected.

In the remainder of this section the upstream water level is assumed at $L(j)$, which means that the 'drop' on the $R(j)$-side has to be limited. In case the upstream water level is at $R(j)$, L and R have to be swapped in the following equations.

In case `drop1D=1` in `[numerics]` the water level slope is limited to:

$$\texttt{slope} = \frac{\zeta_{L(j)} - \max\left(\zeta_{R(j)}, z_{c,R(j)}\right)}{\Delta x(j)} \tag{8.110}$$

where $z_{c,R(j)} = bl_{2j}$ is the invert level of the pipe at the downstream cell of face j.

Remarks:

- ◇ Keep in mind that this limiter is based on the previous time step:

$$\begin{aligned}\texttt{slope} = \theta\frac{\zeta^{n+1}_{L(j)} - \zeta^{n+1}_{R(j)}}{\Delta x(j)} + (1-\theta)\frac{\zeta^{n}_{L(j)} - \zeta^{n}_{R(j)}}{\Delta x(j)} \\ + \frac{\zeta^{n}_{R(j)} - \max\left(\zeta^{n}_{R(j)}, z_{c,R(j)}\right)}{\Delta x(j)}\end{aligned} \tag{8.111}$$

- ◇ As a result, in a dynamic situation, the limiter is not completely exact.

8.15.2 Slope limiter for 1D or 1D2D connections with advection type 8

The slope limiter that is applied for 1D and 1D2D connections with advection type 8 differs slightly from the regular approach. The main difference with the limiter, described in section 8.15.1 is in the way $z_{c,R(j)}$ is computed. Repeating Equation (8.110):

$$\texttt{slope} = \frac{\zeta_{L(j)} - \max\left(\zeta_{R(j)}, z_{c,R(j)}\right)}{\Delta x(j)}, \tag{8.112}$$

where now for this slope limiter:

$$\begin{aligned} z_{c,R(j)} &= \max\left(bl_{1j}, bl_{2j}\right) + \tfrac{2}{3}h_{u_j} \\ h_{u_j} &= \zeta_{u_j} - \max\left(bl_{1_j}, bl_{2_j}\right) \end{aligned}$$

In other words: the critical waterdepth $\frac{2}{3}h_{u_j}$ is added to the limiting level.

8.16 Volume tables for 1D

8.16.1 Introduction

Volume tables contain the total volume for each 1D cell as a function of the water depth. These volume tables are used by the solver to compute the volume in a grid cell for the current water level, via table lookup and simple interpolation. Using the volume tables will improve the performance of a model, since this evaluation of the volume in a cell is more efficient than recalculating the volume every time using the surrounding channel cross section information directly.

The volume tables are pre-calculated based on the cross sections at the location of the 1d cell (see section 8.7), the bed level information and – when present – storage nodes (see section 8.14).

The generation of the volume tables requires some extra initialization time, but during the time integration the use of volume tables is much more efficient in comparison with the default computation. As a result the total required computation time is reduced. The initialization costs can be avoided by storing the volume tables in a file, and reading them directly from this file in subsequent runs for the same model.

8.16.2 Technical information

Computing the volume in a cell without the use of volume tables requires the computation of the total area for each connected flow link, i.e., a full integration of the cross section shapes every time, for every cell. This is a time costly operation.

In a volume table only one interpolation in this table is required per grid cell. And since the volume tables use equidistant increments for the waterdepth, the lookup of the required location in the table is very simple.
The use of volume tables has a drawback. In between two table entries, any variations in cross sectional shape or storage shape are simplified to a linear variation, which makes the volume calculation slightly less accurate.

The equidistant height (increment h_{inc}) between two table entries can be defined by the user in the MDU file (Appendix A). The volume table starts at index 1 (depth = 0) and ends at the depth for the heighest nearby cross section level. A smaller increment results in more accurate results, but also more table entries. When the size of the tables increases, the result will be (1) more computation time and (2) larger memory usage.

For each grid cell k the number of levels N_k is computed:

- ◇ For all cross sections that contribute to the storage of grid cell k the highest level of the cross section definition is used to compute the maximum height $h_{k,\max}$.
- ◇ The number of levels of the volume table in grid cell k depends on the highest level $h_{k,\max}$:

$$N_k = \lceil h_{k,\max}/h_{\text{inc}} \rceil + 1 \tag{8.113}$$

The volumes V_k^i for all table levels $i = 1, 2, \ldots, N_k$ are computed using the cross section information on the connected links and the storage node information on that grid cell. During the simulation, a linear interpolation is used for computing the volume in a grid cell for a given waterdepth. As a result the water surface area is presumed to be constant between two increments, and also constant above the highest level $h_{k,\max}$. The water surface area (A_k^i) is also stored in the volume tables:

$$A_k^i = \begin{cases} \frac{V_k^{i+1} - V_k^i}{h_{\text{inc}}} & \text{for } i = 1, 2, \ldots, N_k - 1, \\ A_k(N^k \cdot h_{\text{inc}}) & \text{for } i = N^k. \end{cases} \tag{8.114}$$

The calculation of the volume for a grid cell k with a given depth h_k using tables is now as follows:

$$V_k(h_k) = V_k^i + (h_k - (i-1) \cdot h_{\text{inc}}) \cdot A_k^i, \tag{8.115}$$

where

$$i = \max(0, \lfloor h_k/h_{\text{inc}} \rfloor) + 1. \tag{8.116}$$

9 Transport of matter

9.1 Introduction

In D-Flow FM, transport is formulated as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} c\,\mathrm{d}V + \int_{\partial V(t)} c(\boldsymbol{u}-\boldsymbol{v})\cdot\boldsymbol{n}\,\mathrm{d}S = \int_{\partial V(t)} (K\nabla c)\cdot\boldsymbol{n}\,\mathrm{d}S + \int_{V(t)} s\,\mathrm{d}V, \tag{9.1}$$

where $V(t)$ is a three-dimensional control volume, c is a concentration, $\boldsymbol{u}$ the flow velocity field, $\boldsymbol{v}$ the velocity of the (vertically) moving control volume, K is a diagonal matrix $K = diag(\kappa, \kappa, \kappa_z)$ with diffusion coefficients and s a source term. For two-dimensional (depth-averaged) flow, we obtain

$$\frac{\partial hc}{\partial t} + \nabla\cdot(h\boldsymbol{u}c) = \nabla\cdot(h\kappa\nabla c) + hs, \tag{9.2}$$

where h is the water depth. In case of three-dimensional (layer-averaged) flow, with Δz a layer thickness from $z_0(x,y,t)$ to $z_1(x,y,t)$, we obtain

$$\begin{aligned}\frac{\partial \Delta z\, c}{\partial t} + \nabla\cdot(\Delta z\, \boldsymbol{u}c) + \left[\omega_{z_1} c\right]_{z=z_1} - \left[\omega_{z_0} c\right]_{z=z_0} = \nabla\cdot(\Delta z\, \kappa\nabla c) + \\ \left[\kappa_z\frac{\partial c}{\partial z} - \kappa\nabla z_1\cdot\nabla c\right]_{z=z_1} - \left[\kappa_z\frac{\partial c}{\partial z} - \kappa\nabla z_0\cdot\nabla c\right]_{z=z_0} + \Delta z\, s,\end{aligned} \tag{9.3}$$

where by $\boldsymbol{u}$ and ∇ still the horizontal components are meant, i.e. $\boldsymbol{u} = (u,v)^{\mathrm{T}}$ and $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)^{\mathrm{T}}$ and κ_z is the vertical diffusion coefficient. Furthermore ω_{z_0} and ω_{z_1} are the velocity component normal and relative to the moving $z = z_0$ and $z = z_1$ layer interfaces respectively. Note that taking $c = 1$ yields

$$\omega_{z_1} + \frac{\partial z_1}{\partial t} = \omega_{z_0} - \nabla\cdot(\Delta z\boldsymbol{u}) + \frac{\partial z_0}{\partial t}, \tag{9.4}$$

which, combined with a zero-flux condition at the bed, recursively defines ω_{z_0} and ω_{z_1} for all layers.

We apply Equation (9.2) and Equation (9.3) to transport of

- salinity,
- temperature,
- suspended sediment,
- tracers,
- other, intentionally not mentioned,

and ultimately to the water itself (the continuity equation) to obtain the relative layer interface velocities as expressed by Equation (9.4).

9.2 Transport processes

Looking at the equations that govern transport of matter, we can identify three processes, namely advection, diffusion and sources/sinks. The next sections will describe the relevant user settings.

9.2.1 Advection

Advection of matter is expressed by the term $\nabla \cdot (h\boldsymbol{u}c)$ in Equation (9.2) in the case of two-dimensional modelling.

In three dimensions, we make a distinction between horizontal advection $\nabla \cdot (\Delta z\, \boldsymbol{u}c)$ and vertical advection $\left[\omega_{z_1} c\right]_{z=z_1} - \left[\omega_{z_0} c\right]_{z=z_0}$ in Equation (9.3). A higher-order numerical approximation of these terms can be obtained by setting their "limiter type" to an appropriate value. Setting the limiter type to "0" reduces the numerical approximation to a low-order upwind method (i.e. severe limiting).

Although not restricted to the advection of salinity only, the choices for *all* matter are set with keyword limtypsa in the mdu-file, for example

```
[numerics]
  limtypsa = 0         # first-order upwind, or
```

or

```
[numerics]
  limtypsa = 4         # MC limiter
```

In three-dimensional modelling horizontal advection is similarly as in two dimension, but now vertical advection can be selected with the keyword Vertadvtypsal in the mdu-file, i.e.

```
[numerics]
  Vertadvtypsal = 0        # no vertical advection
```

or

```
[numerics]
  Vertadvtypsal = 6        # default
```

9.2.2 Diffusion

Diffusion of matter is expressed by the term $\nabla \cdot (h\kappa\nabla c)$ for two-dimensional modelling and a similar and additional terms in three dimensions, see Equation (9.3).

Clearly, we have horizontal diffusivity κ and vertical diffusivity κ_z. The horizontal diffusivity κ is a summation of

◇ the molecular diffusivity κ_l. We use the following values:

$$\kappa_l = \begin{cases} \frac{1}{700}\nu_l, & \text{salt,} \\ \frac{1}{6.7}\nu_l, & \text{temperature,} \\ 0\nu_l, & \text{sediment,} \\ 0\nu_l, & \text{tracers,} \end{cases} \tag{9.5}$$

where $\nu_l = 10^{-6}\ m^2/s$ is the kinematic viscosity,

◇ a background diffusivity, user-specified as

  - spatially varying values by `horizontaleddydiffusivitycoefficient` in the ext-file, or
  - a uniform value `Dicouv` in the mdu-file,

  in that order of precedence, and
- a contribution from turbulent transport expressed as ν_t/σ_t, where ν_t is the eddy viscosity coefficient and σ_t is the turbulent Prandtl-Schmidt number for which the following values are used:

$$\sigma_t = \begin{cases} 0.7, & \text{salt,} \\ 0.7, & \text{temperature,} \\ 1.0, & \text{sediment,} \\ 1.0, & \text{tracers.} \end{cases} \tag{9.6}$$

Horizontal diffusion is turned off when `Dicouv=0`.

The user has to be aware of the following. The explicit nature of the horizontal diffusion terms in the time-integration method imposes a condition on the time step for numerical stability. Recall that we have a similar condition due to explicit horizontal advection. Although we always decrease the time step to satisfy the advection-related time step criterion, we do not do so for diffusion. Instead, diffusion is limited such that our time step is restricted by advection only, or by a user-specified maximum time step if it is smaller. For further details, consult Deltares (2024a).

Be aware that the modelled *diffusion may be be smaller than anticipated. However, the actual* effective *diffusion encountered may be (much) larger than anticipated due to numerical diffusion of the advection scheme.*

Not considering suspended sediment, and similar to the horizontal diffusivity, the vertical diffusivity κ_z is a summation of

- the molecular diffusivity κ_l, see Equation (9.5),
- a background vertical diffusivity, user-specified as a uniform value `Dicoww` in the mdu-file, and
- a contribution from turbulent transport, similar to its horizontal counterpart, but with the horizontal viscosity coefficient now replaced by the vertical (eddy) viscosity coefficient.

Time integration is performed with a method that does not impose an additional constraint on the time step. Unlike horizontal diffusion which is limited to ensure numerical stability while maintaining the time-step size, vertical diffusion is *not* limited.

Vertical diffusion is turned off when, again not considering suspended sediment:

- if `Dicoww=0`,
- if the water depth is smaller than a threshold $10^{-2}\ m$.

For the vertical diffusivity of suspended sediment, see D-Morphology UM (2019).

9.2.3 Sources and sinks

Sources and sinks may be provided by an entry in the ext-file as follows:

```
quantity=discharge_salinity_temperature_sorsin
```

This simultaneously prescribes sources and sinks of

- water volume itself (i.e. discharge),
- salinity, and
- temperature, and.
- any other constituents that are transported.

See section 8.10 for more details.

9.2.4 Forester filter

The central vertical advection scheme for `Vertadvtypsal = 4` or `Vertadvtyptem = 4` may cause nonphysical oscillations, or wiggles, especially near regions of large gradients. The user has the option to suppress salt and temperature wiggles with a filter inspired by the so-called Forester filter. This filter also penalizes physically unstable stratification. The maximum number of iterations in the filter is controlled with keywords `Maxitverticalforestersal` for salt (default 100) and `Maxitverticalforestertem` for temperature (default 0, i.e. no filtering).

9.2.5 Horizontal diffusion in transport equations

The advection and diffusion terms in the transport equation are integrated explicitly in time. However, for 2D models with large horizontal diffusion coefficients this will lead to a large reduction of the time step. Therefore, an implicit time integration scheme has been implemented for the horizontal diffusion term as well.To that purpose `TransportAutoTimestepdiff` has been introduced. Via `TransportAutoTimestepdiff = 3` the implicit time integration scheme is activated. The explicit time integration can be switched on via keyword `TransportAutoTimestepdiff = 0`. In this case the horizontal diffusion coefficient is probably reduced in order to satify the time step condition for stability. If so, then the following message is written in the diagnostic file:

```
** INFO   : Horizontal transport flux was limited for XX links.
```

with XX the number of links in which the horizontal diffusion coefficient has been reduced. Per time step, a warning message displays the number of flow links in which the horizontal diffusivity coefficient has been reduced. In order to prevent the diagnostic file becoming very large, the warning is shown a maximum of ten times. The following message is written to the diagnostic file at the end of the simulation:

```
** INFO   : Viscosity coefficient/Horizontal transport flux were limited on some links in the course of computation.
```

Another approach is to apply `TransportAutoTimestepdiff = 1`. Then, the horizontal diffusion coefficient isn't reduced, but the time step is probably reduced in order to satify the stability condition. If so, then the simulation will require more computation time. Depending on the application the user should choose the most suitable for for keyword `TransportAutoTimestepdiff`.

The options described above represent so-called prognostic approaches, in which the transport quantities are updated at each time step. A different approach is so-called diagnostic modelling of transport processes. Then, the transport quantities aren't updated during the simulation. In this way, the initial conditions are being used and the transport concentrates are frozen during the simulation. In case of horizontal density gradients the pressure term is relevant for the computation of water levels. Via the diagnostic approach the density gradient is taken into account in a simplified way. This avoids a possibly time step reduction for the transport equation in case of large horizontal diffusion coefficients. The latter are often required in depth-averaged simulations for modelling of salinity intrusion of estuaries. We remark that in 3D this problem does not occur, because a turbulence closure model is applied and the horizontal diffusion coefficients are small. In summary, a prognostic modelling of transport processes is to be preferred, but in some applications, in particular in depth-averaged models with large horizontal diffusion coefficients, diagnostic modelling can be a work-around. The latter can be activitated via keyword `DiagnosticTransport=1`.

9.3 Transport boundary and initial conditions

The equations that govern transport of matter are complemented with boundary and initial conditions. We make the distinction between "horizontal" boundaries, that are either "open" or "closed", and vertical boundaries, only relevant for three-dimensional modelling.

9.3.1 Open boundary conditions

At "horizontal" open boundaries the following boundary conditions are applied, with or without diffusion on the open boundary:

- salt, temperature and tracers:
  - inflow: user-specified Dirichlet condition,
  - outflow: homogeneous Neumann condition,
- suspended sediment: see D-Morphology UM (2019).

Horizontal diffusive transport through open boundaries is switched on by default and can be switched off. The diffusive transport can be enabled or disabled by specifying the keyword `Diffusiononbnd` in the mdu-file, and is valid for all open boundary types. Enable diffusive transport on open boundaries (default)

```
[Numerics]
Diffusiononbnd=1
```

Disable diffusive transport on open boundaries

```
[Numerics]
Diffusiononbnd=0
```

The user-specified Dirichlet conditions are supplied in the usual manner through the ext-file, for salt

```
quantity=salinitybnd
```

and for temperature

```
quantity=temperaturebnd
```

Boundaries conditions for multiple tracers may be defined by appending their name to the `tracerbnd` keyword, for example

```
quantity=tracerbndMY_FIRST_TRACER
```

and

```
quantity=tracerbndMY_SECOND_TRACER
```

When boundary conditions for transported constituents need to be prescribed using a three-dimensional distribution (e.g. for stratified flows), the user can provide 3D boundary conditions. This is described in **??**.

9.3.2 Closed boundary conditions

At "horizontal" closed boundaries zero-flux conditions are applied.

9.3.3 Vertical boundary conditions

At the "vertical" boundaries, i.e. at the bed and at the water surface, zero-flux conditions are applied, except for temperature that is, see chapter 10 for further details on that matter.

9.3.4 Thatcher-Harleman boundary conditions

Consider (a part of) an open boundary where the flow reverts from outflowing to inflowing. According to section 9.3.1, at that very moment a Dirichlet condition becomes effective and a user-specified boundary value is prescribed. This value does in general not reflect the true condition at the boundary and causes a discontinuous temporal behaviour. The so-called Thatcher-Harleman boundary condition is intended to regularize this behaviour. The actual value applied at the boundary is smoothly transformed from the last value under outflow conditions to the user-specified value under inflow conditions within a user-specified *return time*, which has to be specified in seconds. This return time is prescribed with the keyword `return_time` in the “new style” external forcings file for boundary condition, for example

```
[boundary]
quantity     = salinitybnd
locationfile = tfl_01.pli
forcingfile  = tfl.bc
return_time  = 250
```

See section C.5 for more details on this format.

9.3.5 Initial conditions

We will only consider intitial conditions for salinity, temperature and tracers. For initial sediment concentrations, see D-Morphology UM (2019).

Initial conditions are specified in three possible ways, namely

- a horizontally spatially varying field in the usual way through the ext-file,
- a vertical profile in three dimensions, horizontally uniformly distributed, for salinity and temperature only in the ext-file, and
- uniform values for salinity and temperature in the mdu-file.

In the ext-file we have

```
quantity      = initialsalinity
```

for the intial salinity, and

```
quantity      = initialsalinitytop
```

for the initial salinity in the top layer in case of three-dimensional modelling. When specified, the initial salinity field is linearly distributed from the "initialsalinity" in the lowest layer to the "initialsalinitytop" in the top layer. When *not* specified, the initial salinity field is vertically uniformly distributed.

The initial temperature field is prescribed with

```
quantity      = initialtemperature
```

which in three dimensions is vertically uniformly distributed.

A horizontally uniformly distributed vertical profile of salinity and temperature can be prescribed with `initialverticalsalinityprofile` and `initialverticaltemperatureprofile` respectively, for example for salinity

```
QUANTITY=initialverticalsalinityprofile
FILENAME=inisal.pol
FILETYPE=10
METHOD=4
OPERAND=O
```

The polygon file contains (z, salinity) value pairs, where z is the vertical coordinate in meters. For example for linearly varying salinity from 30 to $20\ ppt$ from -10 to $0\ m$:

```
L1
2 2
-10 30
  0 20
```

The initial field of an arbitrary number of tracers are prescribed in the same way, except for the user-specified tracername that is added to the keyword initialtracer, similar to the tracer boundary conditions, for example

```
quantity=initialtracerMY_FIRST_TRACER
```

and

```
quantity=initialtracerMY_SECOND_TRACER
```

Default values for salinity and temperature are defined in the mdu-file with Initialsalinity and Initialtemperature respectively.

9.4 Introduction to sediment transport

The sediment transport and morphology module supports both bedload and suspended load transport of non-cohesive sediments and suspended load of cohesive sediments. For schematisation we distinguish "mud" (cohesive suspended load transport), "sand" (non-cohesive bedload and suspended load transport) and "bedload" (non-cohesive bedload only or total load transport) fractions. For a detailed description we refer to D-Morphology UM (2019).

10 Heat transport

This chapter is an almost integral copy of the Delft3D-FLOW manual. The difference is that in Delft3D-FLOW five heat flux models are implemented, whereas in D-Flow FM only two models are implemented. These are the most complete heat flux model, the so called Composite heat flux model (i.e. the Ocean heat flux model nr 5 in Delft3D-FLOW) and the most simple model, the Excess temperature model (model nr 3 in Delft3D-FLOW). In D-Flow FM, the parameter that sets the temperature model is called `Temperature` in the mdu-file. We kept the numbering of Delft3D-FLOW. When specifying `Temperature=1`, the temperature is taken into account in the transport solver and in the equation of state, but heat fluxes through the water surface are not taken into account. This may be useful when mixing is the primary factor that determines the temperature distribution.

The heat radiation emitted by the sun reaches the earth in the form of electromagnetic waves with wavelengths in the range of 0.15 to 4 μm. In the atmosphere the radiation undergoes scattering, reflection and absorption by air, cloud, dust and particles. On average neither the atmosphere nor the earth accumulates heat, which implies that the absorbed heat is emitted back again. The wavelengths of these emitted radiations are longer (between 4 and 50 μm) due to the lower prevailing temperature in the atmosphere and on Earth. Schematically the radiation process, along with the heat flux mechanisms at the water surface, is shown in Figure 10.1.

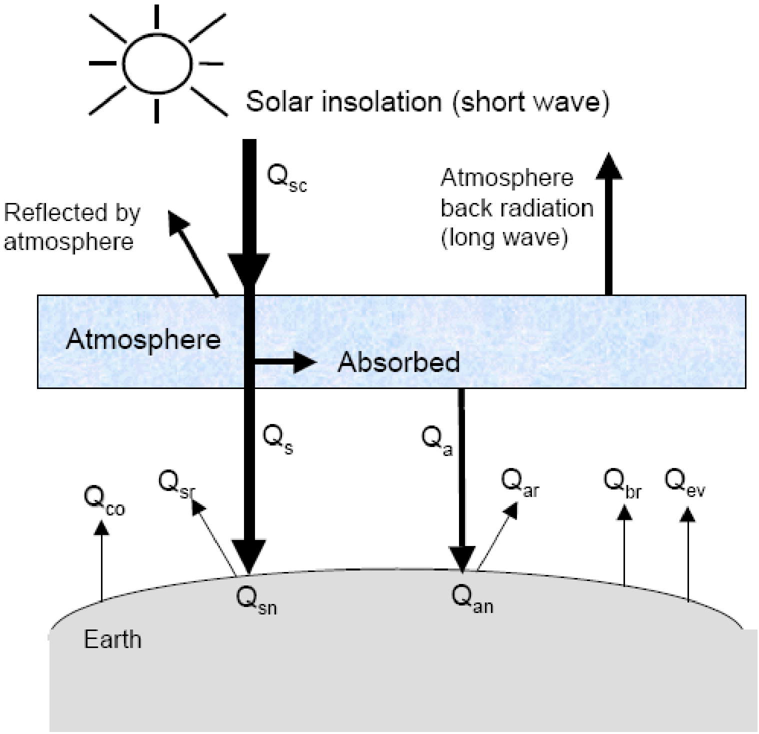


Figure 10.1: *Overview of the heat exchange mechanisms at the surface*

Legend for Figure 10.1:

Q_{sc}	radiation (flux) for clear sky condition in [J/m^{2}s]
Q_{co}	heat loss due to convection (sensible) in [J/m^{2}s]
Q_{sr}	reflected solar radiation in [J/m^{2}s]
Q_s	solar radiation (short wave radiation) in [J/m^{2}s]
Q_{sn}	net incident solar radiation (short wave), $= Q_s - Q_{sr}$

Q_a	atmospheric radiation (long wave radiation) in [J/m^{2}s]
Q_{an}	net incident atmospheric radiation (long wave)
Q_{ar}	reflected atmospheric radiation in [J/m^{2}s]
Q_{br}	back radiation (long wave radiation) in [J/m^{2}s]
Q_{ev}	heat loss due to evaporation (latent) in [J/m^{2}s]

In D-Flow FM the heat exchange at the free surface is modeled by taking into account the separate effects of solar (short wave) and atmospheric (long wave) radiation, and heat loss due to back radiation, evaporation and convection. The heat losses due to evaporation and convection are functions of the wind speed. In absence of wind, these terms become zero. However, since water vapor is lighter than air, water may be cooled by evaporation and convection even in a no wind situation. The terms are called Q_{evfree} and Q_{cofree} respectively.

Excess temperature model - heat flux model 3

In the Excess temperature model the heat exchange flux at the air-water interface is computed based upon the prescribed background air temperature, the computed water temperature of the top layer and the prescribed wind speed. This relatively simple model is sometimes used in intake–outfall design studies. It can be applied when the temperature mixing process itself is more relevant than the actual heat loss through the air water interface. The applied heat exchange coefficient is mainly a function of the windspeed and water surface temperature.

The excess temperature model 3 is based on Sweers (1976), the heat exchange flux is represented by a bulk exchange formula:

$$Q_{tot} = -\lambda \left(T_s - T_{back}\right), \tag{10.1}$$

with T_s the water temperature at the free surface and T_{back} the natural background temperature, both in °C.

The heat exchange coefficient λ is a function of the surface temperature T_s and the wind speed U_{10}. It is derived by linearization of the exchange fluxes for back radiation, evaporation and convection. The following relation was derived by Sweers (1976):

$$\lambda = 4.48 + 0.049T_s + f\left(U_{10}\right)\left(1.12 + 0.018T_s + 0.00158T_s^2\right). \tag{10.2}$$

Composite - heat flux model 5

The heat flux model 5 following Gill (1982) and Lane (1989) was calibrated for the North Sea and successfully applied for great lakes.

In the Composite heat flux model, the relative humidity in [%], air temperature in [°C] and cloudiness in [%] are prescribed.

These quantities may be either uniform or specially varying. In the external forcingsfile one may have:

```
QUANTITY =humidity_airtemperature_cloudiness
FILENAME =meteo.hac
FILETYPE =6
METHOD   =3
OPERAND  =O
```

For example input files see example directories:

➢ f20_heat_flux/**/

The effective back radiation and the heat losses due to evaporation and convection are computed by the model. Additionally, when air and water densities and/or temperatures are such that free convection occurs, free convection of latent and sensible heat is computed by the model.

Normally, solar radiation is computed based upon time of day, position on earth and cloudiness. However, if solar radiation was measured, it can also be prescribed via the solar radiation (in [W/m^2]) according to:

```
QUANTITY =humidity_airtemperature_cloudiness_solarradiation
FILENAME =meteo.hacs
FILETYPE =6
METHOD   =3
OPERAND  =O
```

We remark that the solar radiation doesn't contain the correction factor $(1 - albedo)$. Thus, the solar radiation values prescribed by the user will be multiplied by this factor.

In both heat flux models, the wind forcing may be uniform or spatially varying.

If wind is uniform, the wind speed and direction are prescribed, wind speed is in m/s, and direction follows nautical convention: 0 means wind coming from North, 90 means wind is coming from East. In the external forcings file specify, e.g.:

```
QUANTITY=windxy
FILENAME=zeg99-10.wnd
FILETYPE=2
METHOD=1
OPERAND=O
```

If wind is spatially varying, the air pressure is also prescribed:

```
QUANTITY =airpressure_windx_windy
FILENAME =CSM_2015.apwxwy
FILETYPE =6
METHOD   =3
OPERAND  =O
```

Air pressure is in [Pa], wind x- and y-components are given [m/s].

For the physical background of the heat exchange at the air-water interface and the definitions, we refer to Sweers (1976) for the Excess temperature model (Temperature=3), and to Gill (1982) and Lane (1989) for the Ocean heat flux model (Temperature=5).

10.1 Heat balance

The total heat flux through the free surface reads:

$$Q_{tot} = Q_{sn} + Q_{an} - Q_{br} - Q_{ev} - Q_{co} - Q_{evfree} - Q_{cofree}, \tag{10.3}$$

with:

Q_{sn}	net incident solar radiation (short wave)
Q_{an}	net incident atmospheric radiation (long wave)
Q_{br}	back radiation (long wave)
Q_{ev}	evaporative heat flux (latent heat)
Q_{co}	convective heat flux (sensible heat)
Q_{evfree}	evaporative heat flux (free convection latent heat)
Q_{cofree}	convective heat flux (free convection sensible heat).

The subscript n refers to a net contribution. Each of the heat fluxes in Equation (10.3) will be discussed in detail.

The change in temperature in the top layer T_s [°C] is given by:

$$\frac{\partial T_s}{\partial t} = \frac{Q_{tot}}{\rho_w c_p \Delta z_s}, \tag{10.4}$$

where Q_{tot} [J/m^{2}s] is the total heat flux through the air-water surface, c_p (= 3930 J kg^{-1} K) is the specific heat capacity of sea water, ρ_w is the specific density of water [kg/m^3] and Δz_s [m] is the thickness of the top layer. As in Delft3D-FLOW, the heat exchange at the bed is assumed to be zero. This may lead to over-prediction of the water temperature in shallow areas. Also the effect of precipitation on the water temperature is not taken into account.

Remarks:

- The temperature T is by default expressed in °C. However, in some formulas the absolute temperature $\bar{T}$ in K is used. They are related by:

  $$\bar{T} = T + 273.15. \tag{10.5}$$

- In Equation (10.4) the total incoming heat flux is absorbed with exponential decay as a function of depth. See the parameter Secchi-depth in the mdu-file.

10.2 Solar radiation

The short-wave radiation emitted by the sun that reaches the earth surface under a clear sky condition can be evaluated by means of:

- Applying Stefan-Boltzmann's law for radiation from a black-body:

  $$Q = \sigma \bar{T}^4 \tag{10.6}$$

  with σ = Stefan-Boltzmann's constant = 5.67×10^{-8} J/(m^{2}s K^4) and $\bar{T}$ the (absolute) temperature in K.

- Direct measurements.

The incoming short-wave solar radiation through a clear sky at ground level Q_{sc} is about 0.76 of the flux incident at the top of the atmosphere (Gill, 1982):

$$Q_{sc} = \begin{cases} 0.76 S \sin(\gamma), & \sin(\gamma) \geq 0, \\ 0.0, & \sin(\gamma) < 0. \end{cases} \tag{10.7}$$

The solar constant $S = 1\,368$ J/(m^{2}s) or W/m^2. This is the average energy flux at the mean radius of the Earth.

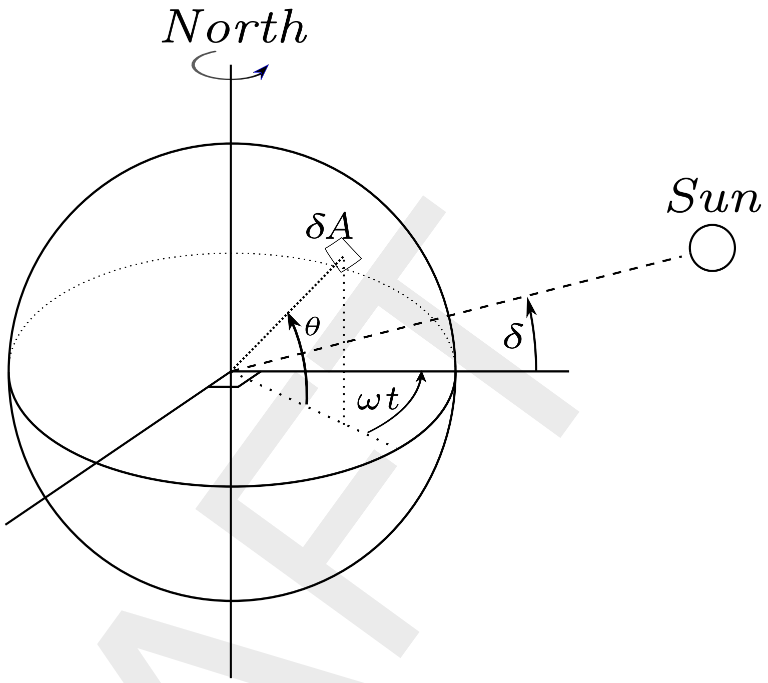


Figure 10.2: *Co-ordinate system position Sun*
δ: declination; θ: latitude; ωt: angular speed

The incoming energy flux at the water surface depends on the angle (declination) between the incoming radiation and the Earth's surface. This declination depends on the geographical position on the Earth and the local time. The Earth axis is not perpendicular to the line connecting the Sun with Earth. This tilting (angle δ) varies with the time of the year and it leads to a seasonal variation of the radiation flux. At June 21, the declination is maximal, 23.5 degrees. Of course, by the rotation of the Earth the solar radiation also varies during the day. Near twelve o'clock local time, the sun elevation above the horizon is maximal. For an overview of the angles used to determine the solar elevation angle γ, see Figure 10.2.

The temporal and latitude-dependent solar elevation angle γ is estimated by:

$$\sin(\gamma) = \sin(\delta)\sin\left(\frac{\pi\phi}{180}\right) - \cos(\delta)\cos\left(\frac{\pi\phi}{180}\right)\cos(\omega_1 t) \tag{10.8}$$

with:

$$\delta = \frac{23.5\pi}{180}\cos(\omega_0 t - 2.95), \tag{10.9}$$

where ω_0 is the frequency of the annual variation and ω_1 the frequency of the diurnal variation; ϕ is the latitude.

A part of the radiation that reaches the water surface is reflected. The fraction reflected or scattered (surface albedo) is dependent on latitude and season. Additionally, cloud cover will reduce the magnitude of the radiation flux that reaches the sea surface. The cloudiness is expressed by a cloud cover fraction F_c, the fraction of the sky covered by clouds. The correction factor for cloud cover is an empirical formula. The absorption of solar radiation is calculated (Gill, 1982) as the product of the net downward flux of short wave-radiation in cloudless conditions and factors correcting for reflection and cloud cover:

$$Q_{sn} = Q_s - Q_{sr} = (1-\alpha)\,Q_{sc}(1.0 - 0.4F_c - 0.38F_c^2), \tag{10.10}$$

with:

Q_{sn} net heat radiation (flux) from the Sun
Q_s solar radiation (short wave radiation) in [J/m^{2}s]
Q_{sr} reflected solar radiation in [J/m^{2}s]
Q_{sc} radiation (flux) for clear sky condition
α albedo (reflection) coefficient (=0.06)
F_c fraction of sky covered by clouds (user-defined input)

Finally, not all of the radiation is absorbed at the water surface. A part is transmitted to deeper water. Short waves can penetrate over a distance of 3 to 30 meters, depending on the clarity of the water, while the relatively longer waves are absorbed at the surface. The Secchi depth H_{Secchi} (m), is a measure of the transparency of the water.

The formulation for the (remaining) under-water light in the water column (W/m2) is based on the so-called Lambert-Beer Law, which can be derived from the following linear first-order ordinary differential equation (ODE):

$$\frac{dQ_{sn}}{dz} = -\gamma Q_{sn}, \tag{10.11}$$

with γ the extinction coefficient (1/m). The total extinction coefficient of visible light γ determines the visibility of a Secchi disk under water. The following approximation is used for the relation between γ and Secchi depth H_{Secchi} :

$$\gamma = \frac{1.7}{H_{Secchi}} \tag{10.12}$$

The Secchi depth is prescribed by the user at input (normal values are in the range from 2 to 30 meter). This results in an exponential function of the distance z from the water surface for the visible light (remaining) in the water column, the Lambert-Beer law (assuming constant γ):

$$Q_{sn}(z) = e^{-\gamma z} Q_{sn}^0 \,, \tag{10.13}$$

with:

Q_{sn}^0 the incoming nett solar radation Q_{sn} that reaches the water surface
γ extinction coefficient (measured) in m^{-1}, also related to the so-called Secchi-depth $\gamma = \frac{1.7}{H_{Secchi}}$
z distance to the water surface in meters.

The vertical coordinate z in Equation (10.13) is the distance to the water surface in meters. The absorbed flux of light (W/m2) in an internal computational layer k with coordinate z_{top} (upper vertical interface of the cell) and z_{bot} (lower vertical interface of the cell) is given by:

$$\begin{aligned} Q_{sn}(k) &= Q_{sn}(z_{top}) - Q_{sn}(z_{bot}) \qquad (10.14) \\ &= \left(e^{-\gamma z_{top}} - e^{-\gamma z_{bot}}\right) Q_{sn}^0 \qquad (10.15) \end{aligned}$$

Secchi disc measurements are, by nature of the observation method, a good proxy for the penetration of visible light in water. The penetration of visible light depends on the composition of the water (suspended and dissolves substances). The penetration of near-infrared (> 700 nm) light depends on water only, not on its composition. The Secchi depth is, therefore, not a good proxy for the penetration depth of near-infrared light. For that reason, we implemented the option to incorporate the penetration of solar radiation in the water column using a blend of two Lambert-Beer Law profiles, based on the work of Wunderlich (1972). This option requires a user-defined ratio β ($0 \leq \beta \leq 1$), which determines the portion of incoming solar radiation in the near-infrared spectrum, which is absorbed near the free-surface. The user can prescribe the thickness of this near-surface layer, i.e. a 'shallow' Secchi depth $H_{Secchi2}$. The rest $(1-\beta)$ is the visible light, absorbed with the (normal) user defined Secchi-depth H_{Secchi}.

Using this approach, the nett incoming solar radiation (after reduction through cloud coverage and surface reflection using the Albedo coefficient) is split into two contributions:

◇ βQ_{sn}, the near-infrared wave portion, which is absorbed at the surface ($H_{Secchi2}$) and
◇ $(1-\beta)\,Q_{sn}$, the remainder part, which is absorbed in deeper water (H_{Secchi}).

Finally, the absorbed flux of light (W/m2) in an internal computational layer k is then given by:

$$Q_{sn}(k) = \left((1-\beta)\left(e^{-\gamma z_{top}} - e^{-\gamma z_{bot}}\right) + \beta\left(e^{-\gamma_2 z_{top}} - e^{-\gamma_2 z_{bot}}\right)\right) Q_{sn}^{0} \tag{10.16}$$

with: γ_2 the 'shallow' extinction coefficient (measured) in m^{-1}, related to the 'shallow' Secchi-depth $\gamma_2 = \frac{1.7}{H_{Secchi2}}$

In the mdu-file, the user can switch on this functionality by specifying the $H_{Secchi2}$ and β using the following keywords:

◇ $H_{Secchi2}$ using `Secchidepth2` and
◇ β using `Secchidepth2fraction` (with $0 \leq \beta \leq 1$).

Note: It should be noted, that the term 'shallow' Secchi depth is formally not appropriate, since a Secchi disk would not be visible in infrared light, but we stick to the same naming to clarify users that the two depths H_{Secchi} and $H_{Secchi2}$ are related.

10.3 Atmospheric radiation (long wave radiation)

Atmospheric radiation is primarily due to emission of absorbed solar radiation by water vapour, carbon dioxide and ozone in the atmosphere. The emission spectrum of the atmosphere is highly irregular. The amount of atmospheric radiation that reaches the earth is determined by applying the Stefan-Boltzmann's law that includes the emissivity coefficient of the atmosphere ε. Taking into account the effect of reflection by the surface and reflection and absorption by clouds, the relation for the net atmospheric radiation Q_{an} reads (Octavio *et al.*, 1977):

$$Q_{an} = (1-r)\,\varepsilon\sigma\bar{T}_a^4 g\left(F_c\right), \tag{10.17}$$

where $\bar{T}_a$ is the air temperature (in K) and the reflection coefficient $r = 0.03$. The emissivity factor of the atmosphere ε may depend both on vapour pressure and air temperature. The emissivity of the atmosphere varies between 0.7 for clear sky and low temperature and 1.0. The presence of clouds increases the atmospheric radiation. This is expressed in the cloud function $g\left(F_c\right)$.

with T_a the air temperature (in °C). The cloud function $g\left(F_c\right)$ in Equation (10.17) is given by:

$$g\left(F_c\right) = 1.0 + 0.17F_c^2.2 - 9 \tag{10.18}$$

The linearisation of Equation (10.17) is carried out around $T_a = 15$ °C.

Remark:

◇ The atmospheric radiation is part of the total long-wave radiation flux, the so-called effective back radiation, see section 10.5.

10.4 Back radiation (long wave radiation)

Water radiates as a near black body, so the heat radiated back by the water can be described by Stefan-Boltzmann's law of radiation, corrected by an emissivity factor $\varepsilon = 0.985$ of water (Sweers, 1976; Octavio *et al.*, 1977) and the reflection coefficient for the air-water interface $r = 0.03$:

$$Q_{br} = (1 - r)\,\varepsilon\sigma\bar{T}_s^4, \tag{10.19}$$

with $\bar{T}_s$ the (absolute) water surface temperature in K.

10.5 Effective back radiation

The total net long wave radiation flux is computed. This is called the effective back radiation:

$$Q_{eb} = Q_{br} - Q_{an}. \tag{10.20}$$

The atmospheric radiation depends on the vapour pressure e_a, see section 10.6, the air temperature T_a and the cloud cover F_c. The back radiation depends on the surface temperature T_s.

The effective back radiation Q_{eb} is computed following:

$$Q_{eb} = \varepsilon\sigma\bar{T}_s^4\left(0.39 - 0.05\sqrt{e_a}\right)\left(1.0 - 0.6F_c^2\right), \tag{10.21}$$

with the actual vapour pressure e_a given by Equation (10.26).

10.6 Evaporative heat flux

Evaporation is an exchange process that takes place at the interface between water and air and depends on the conditions both in the water near the surface and the air above it. The evaporation depends on meteorological factors (wind-driven convection) and vapour pressures.

Forced convection of latent heat

The latent heat flux due to forced convection for the ocean heat flux model reads:

$$Q_{ev,\text{forced}} = L_V\rho_a f\left(U_{10}\right)\left\{q_s\left(T_s\right) - q_a\left(T_a\right)\right\}, \tag{10.22}$$

with q_s and q_a the specific humidity of respectively saturated air and remote air (10 m above water level):

$$q_s\left(T_s\right) = \frac{0.62e_s}{P_{atm} - 0.38e_s}, \tag{10.23}$$

$$q_a\left(T_a\right) = \frac{0.62e_a}{P_{atm} - 0.38e_a}. \tag{10.24}$$

The saturated and remote vapour pressures e_s and e_a are given by:

$$e_s = 10^{\frac{0.7859+0.03477T_s}{1.0+0.00412T_s}}, \tag{10.25}$$

$$e_a = r_{hum}10^{\frac{0.7859+0.03477T_a}{1.0+0.00412T_a}}. \tag{10.26}$$

With L_v the latent heat of vaporisation in J/kg water:

$$L_v = 2.5\ 10^6 - 2.3\ 10^3 T_s. \tag{10.27}$$

The wind function in Equation (10.22) is defined as:

$$f\left(U_{10}\right) = c_e U_{10}, \tag{10.28}$$

The default value for the Dalton number c_e in the Composite heat flux model reads $c_e = 0.0013$. This value should be close to the C_d coefficient that is used in the computation of wind stresses. The exchange coefficients of latent heat and momentum transfer are closely related. Specifying a negative Dalton number in the mdu file forces the use of the specified C_d coefficient, thus taking into account the specified dependency between windspeed and the C_d coefficient.

Here r_{hum} is the relative humidity in [-].

Remarks:

- ◇ The relative humidity r_{hum} is specified in the input files in percentages.
- ◇ When the computed E is negative, it is replaced by zero, assuming that it is caused by modelling misfit and not by the actual physical process of water condensation out of the air into the water. The same applies to the part associated with free convection.

For the excess temperature model, the wind speed function $f\left(U_{10}\right)$ following Sweers (1976) is used:

$$f\left(U_{10}\right) = \left(3.5 + 2.0 U_{10}\right) \left(\frac{5.0 \times 10^6}{S_{area}}\right)^{0.05}, \tag{10.29}$$

where S_{area} is the exposed water surface in m^2, defined in the input and fixed for the whole simulation. The coefficients calibrated by Sweers were based on the wind speed at 3 meter above the free surface; the coefficients in Equation (10.29) are based on the wind speed 10 meter above the water level.

Free convection of latent heat

Loss of heat due to evaporation occurs not only by forced convection, wind driven, but also by free convection. Free convection is driven by buoyant forces due to density differences (by temperature and/or water vapour content) creating unstable conditions in the atmospheric boundary layer. Evaporation due to free convection is important in circumstances where inverse temperature/density gradients are present and wind speeds are almost negligible so that the amount of forced convection is small. Neglecting free convection in this situation will lead to underestimating the heat loss. (Ryan *et al.*, 1974) developed a correction to the wind function, accounting for free convection. The derivation of evaporation by just free convection is based on the analogy of heat and mass transfer.

The latent heat flux due to free convection reads:

$$Q_{ev,\text{free}} = k_s L_V \overline{\rho}_a \left(q_s - q_a\right), \tag{10.30}$$

with the average air density:

$$\overline{\rho}_a = \frac{\rho_{a0} + \rho_{a10}}{2}, \tag{10.31}$$

and with the heat transfer coefficient defined as:

$$k_s = \begin{cases} 0 & \text{if} \quad \rho_{a10} - \rho_{a0} \leq 0 \\ c_{fr.\text{conv}} \left\{ \frac{g\alpha^2}{\nu_{air}\overline{\rho}_a} \left(\rho_{a10} - \rho_{a0}\right) \right\}^{1/3} & \text{if} \quad \rho_{a10} - \rho_{a0} > 0 \end{cases} \tag{10.32}$$

where the coefficient of free convection $c_{fr.\text{conv}}$ was calibrated to be 0.14, see (Ryan *et al.*, 1974). The viscosity of air ν_{air} is assumed to have the constant value 16.0×10^{-6} m^2/s. The molecular diffusivity of air α m^2/s is defined as

$$\alpha = \frac{\nu_{air}}{\sigma}, \tag{10.33}$$

with $\sigma = 0.7$ (for air) the Prandtl number. In Equation (10.30), the saturated air density is given by:

$$\rho_{a0} = \frac{\frac{100P_{atm} - 100e_s}{R_{dry}} + \frac{100e_s}{R_{vap}}}{T_s + 273.15}, \tag{10.34}$$

the remote air density (10 m above the water level):

$$\rho_{a10} = \frac{\frac{100P_{atm} - 100e_a}{R_{dry}} + \frac{100e_a}{R_{vap}}}{T_{air} + 273.15}, \tag{10.35}$$

where R_{dry} is the gas constant for dry air: 287.05 J/(kg K) and R_{vap} is the gas constant for water vapour: 461.495 J/(kg K). The specific humidity of respectively saturated air and remote air (10 m above the water level), q_s and q_a are given by Equation (10.23) and Equation (10.24). The saturated and remote vapour pressure e_s and e_a are defined in Equation (10.25) and Equation (10.26).

The total heat flux due to evaporation then results from adding the forced convection of latent heat in Equation (10.22) and the free convection of latent heat in Equation (10.30):

$$Q_{ev} = Q_{ev,\text{forced}} + Q_{ev,\text{free}}. \tag{10.36}$$

10.7 Convective heat flux

In the Ocean heat flux model, the convective heat flux is split into two parts, just as the evaporative heat flux. The convective heat flux is divided into a contribution by forced convection and a contribution by free convection.

Forced convection of sensible heat

The sensible heat flux due to forced convection is computed by:

$$Q_{co,\text{forced}} = \rho_a c_p g\left(U_{10}\right)\left(T_s - T_a\right), \tag{10.37}$$

with c_p the specific heat of air. It is considered constant and taken to be $1\,004.0$ J/(kg K). The wind-speed function $g\left(U_{10}\right)$ is defined following Gill (1982):

$$g\left(U_{10}\right) = c_H U_{10}, \tag{10.38}$$

with c_H the so-called Stanton number. The default value of the Stanton number reads $c_H = 0.0013$.

Free convection of sensible heat

$$Q_{co,\text{free}} = k_s \overline{\rho}_a c_p \left(T_s - T_a \right), \tag{10.39}$$

with the heat transfer coefficient k_s given by Equation (10.32).

The total heat flux due to convection then results from adding the forced convection of sensible heat in Equation (10.37) and the free convection of sensible heat in Equation (10.39):

$$Q_{co} = Q_{co,\text{forced}} + Q_{co,\text{free}}. \tag{10.40}$$

11 Wind

Various external influences can exert a force on the flow field. One of these influences is the wind. The force exerted by the wind is coupled to the flow equations as a shear stress. The magnitude is determined by the following widely used quadratic expression:

$$|\boldsymbol{\tau}_s| = \rho_a C_d U_{10}^2 \tag{11.1}$$

where:

ρ_a	the density of air.
U_{10}	the wind speed 10 meter above the free surface (time and space dependent).
C_d	the wind drag coefficient, dependent on U_{10}.

In order to specify the wind shear stress, a drag coefficient is required as well as the wind field in terms of velocity magnitude and wind direction. In this chapter, the backgrounds are provided of how wind fields should be imposed, in addition to section 5.4.9.4. Relevant definitions are addressed in section 11.1, whereas supported file formats are addressed in section 11.2.

The density of air in the computation of the wind stress is considered constant as a default, but it is also possible to use a time- and space-varying air density. The air density can be prescribed directly as a timeseries field or it can be computed from prescribed time- and space-varying air pressure, air temperature and dew point temperature (see section E.1.9).

The steps to compute the air density [kg m^{-3}] are (following ECMWF (2023)):

$$e = e_0 \exp(17.502(T_d - T_0)/(T_d - 32.19)) \tag{11.2}$$

$$q_v = \frac{e}{p + \varepsilon^*(p - e)} \tag{11.3}$$

$$\varepsilon^* = \frac{R_v}{R_d} - 1 \tag{11.4}$$

$$T_v = (1 + \varepsilon^* q_v)T \tag{11.5}$$

$$\rho_a = \frac{p}{R_d T_v} \tag{11.6}$$

where

e	water vapour saturation pressure [Pa].
e_0	water vapour saturation pressure over water at T_0 [611.21 Pa].
T_0	triple point temperature [K].
T_d	dew point temperature [K].
q_v	specific humidity [g kg^{-1}].
p	atmospheric pressure [Pa].
R_v	gas constant for water vapor [461.5249 J kg^{-1} K^{-1}].
R_d	gas constant for dry air [287.0596 J kg^{-1} K^{-1}].
T_v	virtual temperature [K].
T	air temperature [K].

11.1 Definitions

When imposing wind conditions, two definitions are respected: a definition for the wind direction (see section 11.1.1) and a definition regarding the drag coefficient (see section 11.1.2).

11.1.1 Nautical convention

The wind direction is defined according to the nautical definition, i.e. relative to true North and positive measured clockwise. In Figure 11.1 the wind direction is about +60 degrees, i.e. an East-North-East wind.

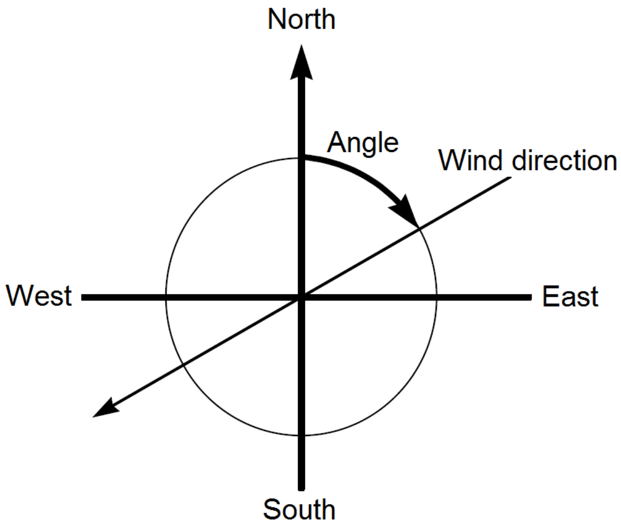


Figure 11.1: *Nautical conventions for the wind.*

11.1.2 Drag coefficient

The user can select how the wind drag coefficient should be computed, by specifying the type of wind drag formulation in the mdu-file. For this purpose, the keyword `ICdtyp` should be used. Nine options are available:

- `ICdtyp = 1` – constant drag coefficient,
- `ICdtyp = 2` – linearly varying drag coefficient (cf. Smith and Banke (1975)),
- `ICdtyp = 3` – piecewise linearly varying drag coefficient (cf. Smith and Banke (1975)),
- `ICdtyp = 4` – Charnock (1955) formulation,
- `ICdtyp = 5` – Hwang (2005a) and Hwang (2005b) formulation,
- `ICdtyp = 6` – Wüest and Lorke (2003) formulation,
- `ICdtyp = 7` – Hersbach (2011) formulation,
- `ICdtyp = 8` – Charnock (1955) formulation plus viscous effect,
- `ICdtyp = 9` – Garratt (1977) formulation.

If one of the Smith & Banke type dependencies is selected, the additional entries `Cdbreakpoints` and `Windspeedbreakpoints` come into play. In the following sections, these various options are described in more detail.

Smith & Banke type formulation

When specifying a Smith & Banke type dependency, the definition as sketched in Figure 11.2 should be kept in mind.

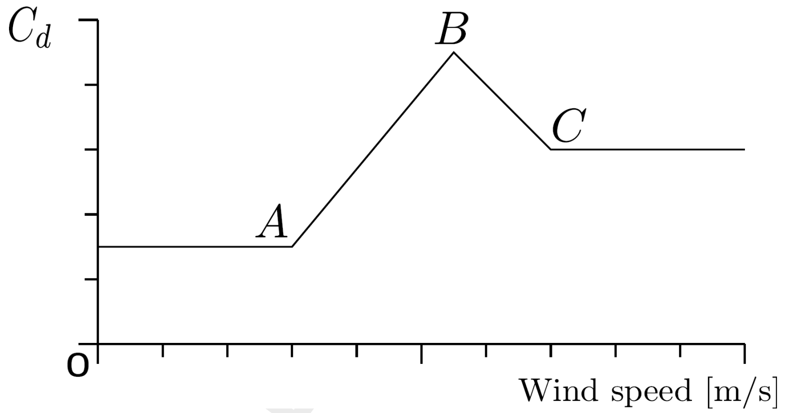


Figure 11.2: *Prescription of the dependency of the wind drag coefficient C_d on the wind speed is achieved by means of at least 1 point, with a maximum of 3 points.*

From this sketch, it can be seen that the wind drag is considered as dependent on the wind speed in a piecewise linear way. The options, that are facilitated in this respect, are:

◇ define *one* set of coordinates (breakpoint A), specifying a constant drag coefficient, valid for all wind speeds,
◇ define *two* sets of coordinates (breakpoints A and B), specifying a linearly varying dependency for one range of wind speeds,
◇ define *three* sets of coordinates (breakpoints A, B and C), specifying a piecewise linear dependency for two ranges of wind speeds.

Remark that for the latter two options, the drag coefficient is taken constant for wind speeds lower/higher than the lowest/highest specified wind speed, with a drag coefficient equal to the drag coefficient associated with the lowest/highest specified lowest/highest wind speed. In case of three breakpoints, the expression reads:

$$C_d\left(U_{10}\right) = \begin{cases} C_d^A, & U_{10} \leq U_{10}^A, \\ C_d^A + \left(C_d^B - C_d^A\right) \dfrac{U_{10} - U_{10}^A}{U_{10}^B - U_{10}^A}, & U_{10}^A \leq U_{10} \leq U_{10}^B, \\ C_d^B + \left(C_d^C - C_d^B\right) \dfrac{U_{10} - U_{10}^B}{U_{10}^C - U_{10}^B}, & U_{10}^B \leq U_{10} \leq U_{10}^C, \\ C_d^C, & U_{10}^C \leq U_{10}, \end{cases} \tag{11.7}$$

By means of the entries `Cdbreakpoints` and `Windspeedbreakpoints`, the coordinates of the breakpoints (see Figure 11.2) can be specified. Typical values associated with the Smith and Banke (1975) formulation are $C_d = 6.3 \times 10^{-4}$ for $U = 0$ m/s and $C_d = 7.23 \times 10^{-3}$ for $U = 100$ m/s. In this case, the entries in the mdu-file should be specified as follows:

```
[wind]
ICdtyp                          =  2
```

```
Cdbreakpoints                         =   0.00063   0.00723
Windspeedbreakpoints                  =   0.00000 100.00000
```

Relative wind

In D-Flow FM, there is the possibility to compute wind shear stress based on the *relative* wind velocity, i.e. relative to the flow velocity. This becomes important when the flow is predominantly forced by the wind and the flow velocity is in the order of the wind velocity. In this way, one can avoid that the wind still forces the flow, despite a zero (or small) difference between flow and wind speed. In case of relative wind Equation (11.1) is changed to

$$|\boldsymbol{\tau}_s| = \rho_a C_d \left(U_{10} - \alpha U_{\mathit{flow}}\right)^2 \tag{11.8}$$

where:

ρ_a	the density of air.
U_{10}	the wind speed 10 meter above the free surface (time and space dependent).
C_d	the wind drag coefficient, dependent on U_{10}.
U_{flow}	the flow velocity.
α	the relative wind factor.

This option can be switched on via keyword `Relativewind`. Then, a value between 0 and 1 has to be specified, which acts as a factor for relative wind, as shown in Equation (11.8).

Charnock formulation

The Charnock formulation (see Charnock (1955)) is based on the assumption of a fully developed turbulent boundary layer of the wind flow over the water surface. The associated wind speed profile follows a logithmic shape. In the Charnock formulation, the wind speed is considered at 10 meters above the free water surface, hence yielding the following expression:

$$\frac{U_{10}}{u_*} = \frac{1}{\kappa} \ln\left(\frac{z_{10}}{z_0}\right) \tag{11.9}$$

with κ the Von Kármán constant, z_{10} the distance to the water surface (equal to 10 m), u_* the friction velocity and U_{10} the wind speed at 10 m above the water surface. The drag coefficient C_d is defined as:

$$C_d = \frac{u_*^2}{U_{10}^2}. \tag{11.10}$$

Charnock (1955) has proposed to represent the friction of the water surface as z_0 according to:

$$z_0 = \frac{b\, u_*^2}{g}, \tag{11.11}$$

with g the gravitation acceleration and b a specific constant. Charnock (1955) has proposed $b = 0.012$. The value of the constant b can be specified in the mdu-file by the user by means of one single value for `Cdbreakpoints`. Since the above relation yields an implicit relation for u_*, the system is solved for iteratively. The user should be aware of interpretation of the specified wind field as the wind field at 10 m above the water surface. See section 11.2.4 for a space and time varying Charnock coefficient b.

Hwang formulation

The dynamic roughness could also be related to the steady state wave conditions of the flow field under consideration. The connection of the wave parameters with the drag coefficient as elaborated by Hwang (2005a) is available within D-Flow FM through `ICdtyp = 5`, given a wave field. The Hwang-formulation interprets the user defined wind speed as the wind speed at 10 m above the water surface.

The drag coefficient is computed as:

$$C_d = \left[\frac{1}{\kappa}\ln\left(\frac{k_p z_{10}}{k_p z_0}\right)\right]^{-2} \tag{11.12}$$

with $z_{10} = 10$ m, κ the Von Kármán constant. With wavelength scaling, $k_p z_0$ is the natural expression of the dimensionless roughness, where k_p is the wave number of the spectral peak, computed on the basis of the actual water depth and the provided peak period T_p as wave field. Further following Hwang (2005a),

$$k_p z_0 = \pi \exp\left(-\kappa C_{\lambda/2}^{-0.5}\right) \tag{11.13}$$

in which $C_{\lambda/2}$ is the drag coefficient at half the wavelength above surface. This parameter $C_{\lambda/2}$ is computed as:

$$C_{\lambda/2} = A_{10}\left(\frac{\omega_p U_{10}}{g}\right)^{a_{10}} \tag{11.14}$$

in which $A_{10} = 1.289 \times 10^{-3}$, $a_{10} = 0.815$, U_{10} the wind speed at 10 m above the water surface and ω_p the wave peak frequency ($\omega_p = 2\pi/T_p$). Thus, the drag coefficient C_d is defined.

Wuest formulation

Wüest and Lorke (2003) define the drag coefficient as

$$C_d = \begin{cases} 0.00063 + 0.000066 U_{10} & \text{if } U_{10} > 4 \\ 0.0044\max(0.1, U_{10})^{-1.15} & \text{if } U_{10} \leq 4 \end{cases} \tag{11.15}$$

Hersbach formulation

Hersbach (2011) defines the drag coefficient as

$$C_d = \left(\frac{\kappa}{b_n^{\text{fit}}}\right)^2 \tag{11.16}$$

where b_n^{fit} is split into a viscous b_n^{ν} and Charnock b_n^{α} contribution as

$$b_n^{\text{fit}} = \left[(b_n^{\nu})^p + (b_n^{\alpha})^p\right]^{1/p} \tag{11.17}$$

$$b_n^{\nu} = -1.47 + 0.93\ln R \tag{11.18}$$

$$b_n^{\alpha} = 2.65 - 1.44\ln A - 0.015(\ln A)^2 \tag{11.19}$$

in which

$$A = \frac{\alpha_{ch}}{g z_{10}}(\kappa U_{10})^2 \tag{11.20}$$

and

$$R = \frac{z_{10}}{\alpha_M \nu_{\text{air}}} \kappa U_{10} \tag{11.21}$$

where z_{10} equals the elevation at which the velocity is given, i.e. 10 m. The values of the constants α_{ch} and α_M can be specified in the mdu-file by the user by means of two values for `Cdbreakpoints`.

Charnock with viscous

When this option is selected, the drag coefficient is computed according Equation (11.10) where u_* is determined iteratively using

$$z_0 = \frac{b_1 u_*^2}{g} + \frac{b_2 \nu_{\text{air}}}{u_*} \tag{11.22}$$

and Equation (11.9). The values of the constants b_1 and b_2 can be specified in the mdu-file by the user by means of two values for `Cdbreakpoints`.

Garratt formulation

Garratt (1977) defines the drag coefficient as

$$C_d = \min(10^{-3}(0.75 + 0.067 U_{10}), 0.0035) \tag{11.23}$$

This formulation can also be represented as a Smith & Banke type formulation.

```
[wind]
ICdtyp                              =  2
Cdbreakpoints                       =  0.00075   0.0035
Windspeedbreakpoints                =  0.00000  41.0448
```

11.2 File formats

The wind field should be provided by means of an ascii-type file. This file should contain the grid on which the wind field is defined as well as the wind velocity vector(s).

D-Flow FM currently supports four types of wind field prescriptions, i.e. four grid types on which the wind field can be given. This wind grid does not need to be the same as the computational grid. The grid options to provide the wind data on are:

1. the computational grid — in this case, no specific wind grid is provided. The provided wind field is considered to be uniform over the entire model area. The wind field can be time dependent.
2. an equidistant grid — in this case, a wind field can be prescribed that varies both in space and in time. A Cartesian arcinfo-type grid should be provided on which the wind field is defined.
3. a curvilinear grid — this case is conceptually similar to the previous type (the equidistant grid) in the sense that a wind field can be imposed that both varies in space and time. However, a separate file should be provided in which a curvilinear grid is defined (a classic <∗.grd>-type file as known from Delft3D-FLOW) on which the wind field is defined.
4. a spiderweb grid — this type of wind specification is specially devoted to cyclone winds and is only available in combination with computational grids that are of spherical type. In this case, a cyclone wind field is given on a polar grid with the center ('eye') of the cyclone being the origin of the polar coordinate system. The location of this eye and the associated wind field usually varies in time.

Each of these filetypes can be assigned through the entry in the external forcings file (the <∗.ext>-file) named `FILETYPE`. In this chapter, the various types of wind field specifications are highlighted subsequently. Each of the options is illustrated by means of an example.

11.2.1 Defined on the computational grid

In D-Flow FM, the specification of the wind on the computational grid is equivalent to the specification of a uniform wind, since no separate wind grid is provided to the model. The specification of a uniform wind field can be done in two ways:

1 componentwise: as velocity in the longitudinal x-direction [m/s] and in the latitudal y-direction [m/s] — the associated `FILETYPE` in the external forcings file is depicted as `uniform`, which has `FILETYPE=1`.
2 by magnitude [m/s] and direction [degN] (see Figure 11.1) — the associated `FILETYPE` in the external forcings file is depicted as `unimagdir`, which has `FILETYPE=2`.

These two types are treated below separately.

11.2.1.1 Specification of uniform wind through velocity components

Since no particular wind grid is used, only timeseries for the x-component and the y-component of the wind need to be specified. The specification of these timeseries can be done separately (one single file for the x-component and one single file for the y-component) or jointly (one single file containing the x-component and the y-component of the wind).

Uniform wind should be provided as an <∗.wnd>-file containing either 2 colums (in case of separate specification of the x-component and y-component of the wind) or 3 columns (in case of joint specification of the velocity components). In either case, the first column contains the time in minutes with respect to the overall reference time.

Example

As an example, a uniform wind field is applied to a certain model. The uniform wind is provided in a file named `windxdirydir.wnd`. The contents of this wind file are:

```
  0.00000   10.00000   10.00000
 60.00000  -10.00000  -10.00000
```

The first column denotes the time in minutes with respect to the reference date (specified in the mdu-file). The second column denotes the wind velocity in x-direction, whereas the third column denotes the wind velocity in y-direction; both wind components are provided in one single file.

The connection with the flow model itself is laid through the external forcings file. The actual specification of the wind is in this case:

```
QUANTITY =windxy
FILENAME =windxdirydir.wnd
FILETYPE =1
METHOD   =1
OPERAND  =O
```

Since the two components are given in one single file, the `QUANTITY` is set to `windxy`. If two separate files would have been provided, the `QUANTITY` would have been set to `windx` and `windy` over two separate datablocks in the external forcings file.

11.2.1.2 Specification of uniform wind through magnitude and direction

Instead of specifying the separate components of the wind field, the uniform wind vector can also be prescribed through its magnitude and direction (see Figure 11.1).

This kind of specification should be done by means of one single file, containing three columns, representing the time (in minutes with respect to the reference date), the velocity magnitude [m/s], not necessarily positive, and the direction (nautical convention).

Example

As an example, the previous uniform wind case is reformulated as a case with magnitude and direction of the wind field prescribed. The unimagdir wind is provided in a file named <windinput.wnd>. The contents of this file are:

```
  0.00000   14.14213562373095  225.00000
 60.00000  -14.14213562373095  225.00000
```

The first column denotes the time in minutes with respect to the reference date (specified in the mdu-file). The second column denotes the wind velocity magnitude, whereas the third column denotes the wind direction. Note that there is a clear difference between the above case and a case in which the magnitude is kept positive (14.1421 m/s) and the direction varies (and hence *rotates*!) from 225 degN to 45 degN.

The connection with the flow model itself is laid through the external forcings file. The actual specification of the wind is in this case:

```
QUANTITY =windxy
FILENAME =windinput.wnd
FILETYPE =2
METHOD   =1
OPERAND  =O
```

11.2.2 Defined on an equidistant grid

The vector components of the velocity vectors can also be specified on a distinct grid, either of equidistant type or of curvilinear type. In both cases, the characteristics of the grid should be provided. In case of an equidistant grid, the grid is specified in arcinfo-style. That means, the constant grid sizes Δx and Δy should be specified such that a grid is spanned with respect to the location of the lower left corner of the grid (either the center of the lower left cel or the lower left corner of the lower left cell).

Example

As an example, a grid with $\Delta x = \Delta y = 100$ m is spanned, based on the center of the lower left cell, located at $x = y = 60$ m with respect to the origin. The input data for the x-component and the y-component should be specified separately, in two distinct files. The input of the x-component data should be given in an <$*$.amu>-type file, such as <windxdir.amu> as an example:

```
### START OF HEADER
### This file is created by Deltares
### Additional commments
FileVersion      =    1.03
filetype         =    meteo_on_equidistant_grid
NODATA_value     =    -9999.0
n_cols           =    5
n_rows           =    4
grid_unit        =    m
x_llcenter       =     60
y_llcenter       =     60
dx               =    110
dy               =    110
n_quantity       =    1
quantity1        =    x_wind
unit1            =    m s-1
### END OF HEADER
TIME = 0 hours since 2006-01-01 00:00:00 +00:00
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
TIME = 1 hours since 2006-01-01 00:00:00 +00:00
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
```

For the y-component data, a similar file (e.g. <windydir.amv>) should be provided. In addition, the pressure could be specified in a similar file (e.g. <pressure.amp>). Note that `x_llcorner` and `y_llcorner`, instead of `x_llcenter` and `y_llcenter`, are also supported.

Wind on an equidistant grid has been provided a filetype specification as `FILETYPE=4`. The connection with the flow model itself is laid through the external forcings file. The actual specification of the wind is in this case:

```
QUANTITY =windx
FILENAME =windxdir.amu
FILETYPE =4
METHOD   =2
OPERAND  =O

QUANTITY =windy
FILENAME =windydir.amv
FILETYPE =4
METHOD   =2
OPERAND  =O

QUANTITY =atmosphericpressure
FILENAME =pressure.amp
FILETYPE =4
METHOD   =2
OPERAND  =O
```

11.2.3 Defined on a curvilinear grid

In analogy with the wind specification on an equidistant grid, the wind can be specified on a curvilinear grid. This curvilinear grid should be provided as a classic <$*$.grd>-file as known from Delft3D-FLOW. A difference with the equidistant grid wind is the necessity to compile all data blocks (i.e. pressure, x-component and y-component) in one single file. This file should have the extension <$*$.apwxwy>. The sequence of this datablock is:

1 pressure,

2 x-velocity component,
3 y-velocity component.

Example

As an example, a curvilinear grid named <meteo.grd> is present, providing the underlying coordinates of the wind data field. The input data, comprising the atmospheric pressure, the x-velocity component *and* the y-velocity component, are given in one single file (as is compulsory). The contents of the example <meteo.apwxwy>-file is:

```
### START OF HEADER
### This file is created by Deltares
### Additional commments
FileVersion      =    1.03
filetype         =    meteo_on_curvilinear_grid
NODATA_value     =    -9999.0
grid_file        =    meteo.grd
first_data_value =    grid_llcorner
data_row         =    grid_row
n_quantity       =    3
quantity1        =    air_pressure
unit1            =    Pa
quantity2        =    x_wind
unit2            =    m s-1
quantity3        =    y_wind
unit3            =    m s-1
### END OF HEADER
TIME = 0.0 hours since 2006-01-01 00:00:00 +00:00
101325 101325 101325 101325 101325
101325 101325 101325 101325 101325
101325 101325 101325 101325 101325
101325 101325 101325 101325 101325
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
10 10 10 10 10
TIME = 1.0 hours since 2006-01-01 00:00:00 +00:00
101325 101325 101325 101325 101325
101325 101325 101325 101325 101325
101325 101325 101325 101325 101325
101325 101325 101325 101325 101325
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
-10 -10 -10 -10 -10
```

Note that `grid_llcenter`, instead of `grid_llcorner`, is also supported. On the contrary, `grid_column` is *not* supported instead of `grid_row`.

Wind on a curvilinear grid has been provided a filetype specification as `FILETYPE=6`. The connection with the flow model itself is laid through the external forcings file. The actual specification of the wind is in this case:

```
QUANTITY =airpressure_windx_windy
FILENAME =meteo.apwxwy
FILETYPE =6
```

```
METHOD   =3
OPERAND  =O
```

Notice that `METHOD=3` is chosen for wind on a curvilinear grid, instead of `METHOD=2` in case of wind on an equidistant grid.

11.2.4 Space and time varying Charnock coefficients

The value for the Charnock coefficient b in Equation (11.11) can be a constant given in the mdu-file, but also be given as a space and time varying field.

For a space and time varying Charnock coefficient, the user should provide a netCDF-file with meteorological forcing, including Charnock coefficients, and use the file specification `FILETYPE=11`.

The specification is in this case:

```
QUANTITY =airpressure_windx_windy_charnock
FILENAME =meteo.nc
FILETYPE =11
METHOD   =3
OPERAND  =O
```

11.2.5 Rainfall

Rainfall can be defined as `rainfall` and `rainfall_rate`. The difference is: `rainfall` is the cumulative precipitation and `rainfall_rate` is the instantaneous precipitation.

The specification is:

```
QUANTITY=rainfall_rate
FILENAME=rad_nl25_rac_mfbs_5min.nc
VARNAME =image1_image_data
FILETYPE=11
METHOD=3
OPERAND=O
```

If given in a netCDF file the unit should be given in attribute. If given in a tim-file, the unit is [mm/day].

11.2.6 Defined on a spiderweb grid

Cyclone winds can be imposed by means of a 'spiderweb'-like polar grid. The origin typically coincides with the cyclone eye and can move in time. Spiderwebs can only be used in combination with a spherical computational grid. The origin of the spiderweb should be given as longitude (for x_{eye}) and latitude (for y_{eye}). The number of rows (discretisation in radial direction) and the number of columns (discretisation in angular direction) should be given, as well as the radius of the grid (in meters). The definition of the spiderweb grid is illustrated in Figure 11.3.

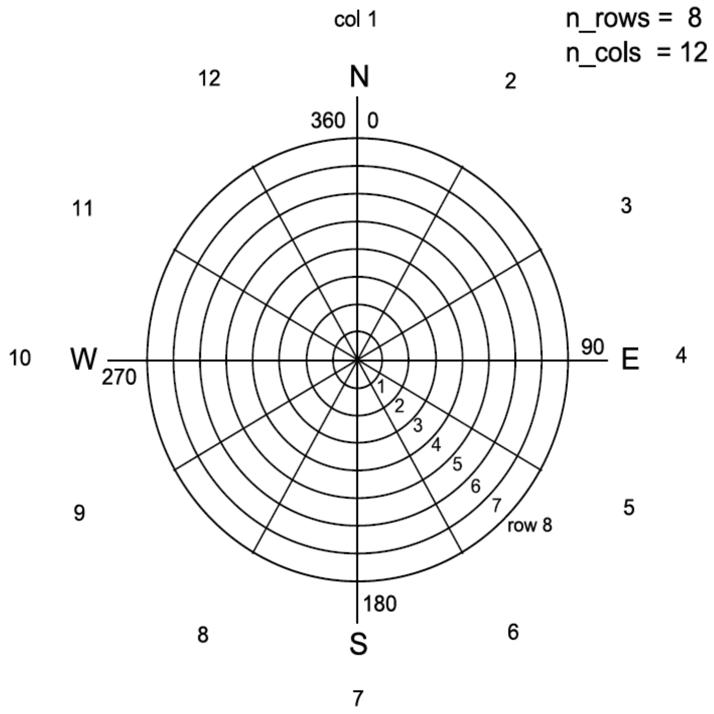


Figure 11.3: *Grid definition of the spiderweb grid for cyclone winds.*

The files containing the spiderweb data and metadata have the extension <∗.spw>. They consist of a global header containing properties that are not varying in time, followed by blocks of data for subsequent time levels. Each of these data blocks is headed by a set of properties for the corresponding time level. A detailed description of the spiderweb file format can be found in section C.13.3.

Through the specified unit for atmospheric pressure in the file, it can be specified whether the values should be interpreted as mbar (=hPa), instead of Pa, which is the default. Specifying the spiderweb merge fraction β in `spw_merge_frac` allows for linear fading of wind speed and pressure drop towards the outer rim. For a spiderweb with radius R, the weigth assigned to the spiderweb wind and pressure at a radius r is given by $(R - r)/\beta R$ for r between $(1 - \beta)R$ and R. The weight equals unity within the inner circle and zero beyond the outer rim.

Example

As an example, a spiderweb grid named <spwsimple.spw> is present, providing the underlying coordinates of the wind data field. The input data, comprising the atmospheric pressure drops, the wind velocity magnitudes (in [m/s]) *and* the wind directions (in [degN]), are given in one single file (as is compulsory). The contents of the example <spwsimple.spw>-file is:

```
### Spiders web derived from TRACK file: gonu.trk
### This file is created by Deltares
### All text on a line behind the first # is parsed as commentary
### Additional commments
FileVersion      = 1.03
filetype         = meteo_on_spiderweb_grid
### Spiders web derived from TRACK file: gonu.trk
### This file is created by Deltares
### All text on a line behind the first # is parsed as commentary
### Additional commments
NODATA_value     = -1001
n_cols           = 4
n_rows           = 4
spw_radius       = 600000.0
spw_merge_frac   = 0.75
spw_rad_unit     = m
### END OF HEADER
TIME             = 340000.00   minutes since 2005-01-01 00:00:00 +00:00
x_spw_eye        =  265.00
y_spw_eye        =   33.00
p_drop_spw_eye    = 7000.000
 5.000000  5.000000  5.000000  5.000000
10.000000 10.000000 10.000000 10.000000
15.000000 15.000000 15.000000 15.000000
20.000000 20.000000 20.000000 20.000000
   270.00      0.00     90.00    180.00
   270.00      0.00     90.00    180.00
   270.00      0.00     90.00    180.00
   270.00      0.00     90.00    180.00
  4000.00   4000.00   4000.00   4000.00
  3000.00   3000.00   3000.00   3000.00
  2000.00   2000.00   2000.00   2000.00
  1000.00   1000.00   1000.00   1000.00
TIME             = 380000.00   minutes since 2005-01-01 00:00:00 +00:00
x_spw_eye        =  275.00
y_spw_eye        =   18.00
p_drop_spw_eye    = 8000.000
 5.000000  5.000000  5.000000  5.000000
10.000000 10.000000 10.000000 10.000000
15.000000 15.000000 15.000000 15.000000
20.000000 20.000000 20.000000 20.000000
   270.00      0.00     90.00    180.00
   270.00      0.00     90.00    180.00
   270.00      0.00     90.00    180.00
   270.00      0.00     90.00    180.00
  4000.00   4000.00   4000.00   4000.00
  3000.00   3000.00   3000.00   3000.00
  2000.00   2000.00   2000.00   2000.00
  1000.00   1000.00   1000.00   1000.00
```

Wind on a spiderweb grid has been provided a filetype specification as `FILETYPE=5`. The connection with the flow model itself is laid through the external forcings file. The actual specification of the wind is in this case:

```
QUANTITY =airpressure_windx_windy
FILENAME =spwsimple.spw
FILETYPE =5
METHOD   =1
OPERAND  =O
```

Notice that METHOD=1 is chosen for wind on a spiderweb grid, instead of METHOD=2 in case of wind on an equidistant grid and METHOD=3 in case of wind on a curvilinear grid.

Alternatively, the spiderweb wind and pressure can be specified in the external forcings file as separate quantities referring to the same file (or specify one and omit the other):

```
QUANTITY =windxy
FILENAME =spwsimple.spw
FILETYPE =5
METHOD   =1
OPERAND  =O

QUANTITY =airpressure
FILENAME =spwsimple.spw
FILETYPE =5
METHOD   =1
OPERAND  =O
```

(NB. There is a slight difference between both specifications regarding their effect on the flow. The first approach requires an additional interpolation of wind on the velocity points, which in some cases of coarse computational grids and small scale wind could introduce smoothing.)

11.2.7 Combination of several wind specifications

The combination of the various wind specification types can *only* be achieved if the QUANTITY of the winds to be combined is the same, for instance QUANTITY=windx. The option OPERAND=+ can be used to add a wind field to an existing wind field.

Example

If the uniform wind is to be combined with a wind specified on an equidistant grid, then the wind field could be assigned in the external forcings file as follows:

```
QUANTITY =windx
FILENAME =windxdir.wnd
FILETYPE =1
METHOD   =1
OPERAND  =O

QUANTITY =windy
FILENAME =windydir.wnd
FILETYPE =1
METHOD   =1
OPERAND  =O

QUANTITY =windx
FILENAME =windxdir.amu
FILETYPE =4
METHOD   =2
OPERAND  =+

QUANTITY =windy
FILENAME =windydir.amv
FILETYPE =4
METHOD   =2
OPERAND  =+
```

The same is possible for e.g. a uniform wind and two spiderweb cyclones:

```
QUANTITY=windxy
FILENAME=uni.tim
FILETYPE=2
METHOD=1
OPERAND=O

QUANTITY =windxy
FILENAME =spwsimple.spw
FILETYPE =5
METHOD   =1
OPERAND  =+

QUANTITY =windxy
FILENAME =spwsimple2.spw
FILETYPE =5
METHOD   =1
OPERAND  =+

QUANTITY =atmosphericpressure
FILENAME =spwsimple.spw
FILETYPE =5
METHOD   =1
OPERAND  =+

QUANTITY =atmosphericpressure
FILENAME =spwsimple2.spw
FILETYPE =5
METHOD   =1
OPERAND  =+
```

In the above example, the wind is first prescribed as a spatially uniform time-varying field, onto which the spiderweb winds are added Note that the uniform wind has the `'O'` operand, which means that it overrides rather than adds. However, since the default wind is zero, one might just as well have used the `'+'` operand. If the spiderweb merge fraction is specified in the spiderweb file, a gradual transition between the spiderweb wind and the background wind is applied using the linearly varying weight as described above. The same is done for atmospheric pressure, if specified except that pressure is added to, whereas wind is averaged with the background values. Without any input, the background atmospheric pressure is set to `PavBnd` in the section `[wind]` in the mdu-file, if present and zero otherwise.

11.3 Masking of points in the wind grid from interpolation (‘land-sea mask’)

A mask can be supplied by the user to prevent selected points in the wind grid from contributing to the wind interpolation on velocity points, e.g. to exclude land points. This feature was included to conform to SIMONA and therefore implemented in the same way.

For each individual grid point for which to interpolate from the wind grid:

- Masked wind points are excluded from the interpolation.
- The total of the weight factors for the remaining wind points is determined.
- If this total falls below `1E-03`, the mask is ignored and the original bilinear weights are used.
- Otherwise, the weights for the remaining wind points are normalised again.

The effect of the mask, when applied as a land-sea mask, is that for velocity points close to shore the interpolated wind is no longer influenced by the wind over land (which would otherwise yield a zone of points with reduced wind near the shore).

Specification and format of the mask file

The name of the mask file, if any, is specified in the <.ext> file, labelled SOURCEMASK, directly following the FILENAME specification, e.g.:

```
QUANTITY   =windxy
FILENAME   =meteo.wxwy
SOURCEMASK =meteo_mask.asc
FILETYPE   =6
METHOD     =3
OPERAND    =O
```

The mask file itself has the same layout as the wind file, though the number of required header fields is reduced, e.g.:

```
FileVersion      =    1.03
.
.

unit1            =    Pa
### END OF HEADER
1       1       1       1       1
1       1       1       0       0
1       1       1       0       0
1       1       0       0       0
```

The lines in the header are ignored. The number of columns and rows in the matrix of ones and zeros should match those of a block (for a single variable and a single timestep) in the meteo files. Zeros signify the position of rejected points (and ones those of the accepted points) in the wind grid.

12 Ice

The ice functionality of D-Flow FM is currently limited to a non-moving ice cover. The ice cover may either be externally specified or dynamically computed based on the thermodynamics of the system. Frazil ice is not yet included.

The presence of an ice cover may influence

- ◇ Wind shear, see section 12.3
- ◇ Thermal exchange between water and atmosphere, see section 12.2.2
- ◇ Precipitation and evaporation, see section 12.6
- ◇ Wave growth and propagation, see section 12.7
- ◇ Water pressure, see section 12.4
- ◇ Surface resistance, see section 12.5

12.1 Forced ice cover

In the case of a forced ice cover, the ice thickness $h_i(x, y, t)$ and the ice cover fraction $A_i(x, y, t)$ must be specified by the user via the external forcings file. A snow layer is not yet supported in the forced ice cover mode. These two quantities may be specified via

- ◇ ASCII equidistant grid
- ◇ ASCII curvilinear grid
- ◇ NetCDF curvilinear grid

The quantity names for h_i and A_i are `sea_ice_thickness` and `sea_ice_area_fraction`, respectively.

12.2 Dynamic ice cover

The dynamic ice module consists of a thermodynamic model based on a single ice layer concept with snow on top (Semtner Jr., 1976). The ice model keeps track of three spatial quantities, namely the ice thickness $h_i(x, y, t)$, the snow thickness $h_s(x, y, t)$ and the ice cover fraction $A_i(x, y, t)$. Snowfall is prescribed via the precipitation input. In case of precipitation and air temperatures below zero, then this is considered as snowfall; see also section 12.6. Decay of the snow thickness is computed by the thermodynamic model.

12.2.1 Conservation laws

Although the current version does not yet include horizontal transport, it is foreseen that the functionality will be extended to include horizontal transport of the ice cover based on the elastic-viscous-plastic (EVP) sea-ice rheology of Hunke and Dukowicz (1997). Hence, the equations below include the ice velocity components $u_i(x, y, t)$ and $v_i(x, y, t)$. The equations for the conservation of the mass of ice and snow are given by

$$\frac{\partial(A_i h_i)}{\partial t} + \frac{\partial(u_i A_i h_i)}{\partial x} + \frac{\partial(v_i A_i h_i)}{\partial y} = S_{\text{ice}} + D_{\text{ice}} \tag{12.1}$$

$$\frac{\partial(A_i h_s)}{\partial t} + \frac{\partial(u_i A_i h_s)}{\partial x} + \frac{\partial(v_i A_i h_s)}{\partial y} = S_{\text{snow}} + D_{\text{snow}} + Q_{\text{snow}} \tag{12.2}$$

In these equations, the right-hand side contains thermodynamic terms S_* and diffusion terms D_*, with $*$ depending on the ice equation to be solved. The snowfall is represented by Q_{snow}.

The equation for ice cover fraction A_i reads

$$\frac{\partial A_i}{\partial t} + \frac{\partial (u_i A_i)}{\partial x} + \frac{\partial (v_i A_i)}{\partial y} = S_A + D_A \tag{12.3}$$

with $0 \leq A_i \leq 1$. The ice cover fraction or concentration represents the fraction of a computational cell that is covered by ice. Due to horizontal transport of the ice cover (from ice floes to icebergs), it is possible that only a part of a computational cell is covered by ice (and snow). The ice velocities u_i and v_i are computed via the momentum equations

$$M\left\{\frac{\partial u_i}{\partial t} + \frac{\partial (u_i u_i)}{\partial x} + \frac{\partial (v_i u_i)}{\partial y} - f v_i\right\} = Mg\frac{\partial \zeta}{\partial x} + \tau_a^x + \tau_w^x + F^x \tag{12.4}$$

$$M\left\{\frac{\partial v_i}{\partial t} + \frac{\partial (u_i v_i)}{\partial x} + \frac{\partial (v_i v_i)}{\partial y} + f u_i\right\} = Mg\frac{\partial \zeta}{\partial y} + \tau_a^y + \tau_w^y + F^y \tag{12.5}$$

with ζ the water elevation, f the Coriolis force and M the ice mass, g the acceleration due to gravity, F the internal stresses according to Hibler III (1979) and τ_a the air and τ_w the water stresses, respectively. The internal ice stress F includes a compressive ice strength according to

$$P = p^* h_i e^{-C(1-A_i)} \tag{12.6}$$

with p^* and C empirical constants of 2.5 × 10^4 and 20, respectively.

However, without horizontal movement, the ice velocity components $u_i(x, y, t)$ and $v_i(x, y, t)$ are currently always zero and the ice fraction A_i is always equal to zero (no ice in a computational cell) or one (computational cell completely filled with ice). The Equations 12.1 and 12.2 simplify to

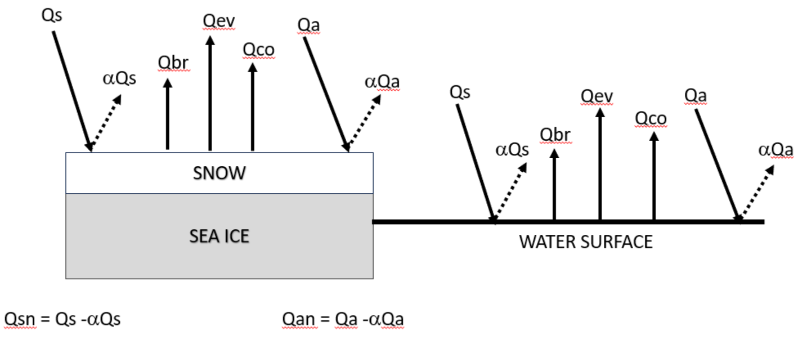
$$\frac{\partial h_i}{\partial t} = S_{\text{ice}} \tag{12.7}$$

$$\frac{\partial h_s}{\partial t} = S_{\text{snow}} + Q_{\text{snow}} \tag{12.8}$$

12.2.2 Thermodynamics of the ice/snow cover

The vertical growth rate (or decay) of ice thickness is determined by the ice thermodynamics. The model's thermodynamic interactions between snow, ice, water, and atmosphere are shown in the left panel of Figure 12.1. The heat fluxes through the ice are based on a simple one layer ice model with snow on top, according to Semtner Jr. (1976). The right panel in Figure 12.1 represents the (standard) temperature model in D-Flow FM when ice is absent.

Figure 12.1: *Conceptual diagram of the ice-growth (left) and temperature (right) model*



The total heat flux through the free surface reads

$$Q_{\text{tot}} = Q_{\text{sn}} + Q_{\text{an}} - Q_{\text{br}} - Q_{\text{ev}} - Q_{\text{co}} \tag{12.9}$$

with Q_{sn} the net incident solar radiation (short wave), Q_{an} the net incident atmospheric radiation (long wave), Q_{br} the back radiation (long wave), Q_{ev} the evaporative heat flux (latent heat) and Q_{co} the convective heat flux (sensible heat). The subscript n refers to a net contribution. Each of the heat fluxes in Equation (12.9) are described in detail in Chapter 10. Furthermore, the net atmospheric radiation Q_{an} and the back radiation Q_{br} have been combined into the effective back radiation Q_{ebr} according to

$$Q_{\text{ebr}} = Q_{\text{br}} - Q_{\text{an}} \tag{12.10}$$

The effective back radiation Q_{ebr} reads

$$Q_{\text{ebr}} = \alpha T_s^4 \tag{12.11}$$

with $\alpha = \varepsilon\sigma(0.39 - 0.05\sqrt{e_a})(1.0 - 0.6F_c^2)$ in J m^{-2} s^{-1} K^{-4}, in which e_a is the vapour pressure, F_c the cloudiness factor, ε the emissivity factor of the atmosphere and the air and σ the Stefan-Boltzman constant. In the numerical sea-ice model the temperature is assumed to vary linearly between the top and bottom of the ice layer. By further assuming a constant thermal conductive coefficient k_i for ice, the heat conduction through the ice is given by

$$Q_{\text{tot}} = -k_i \frac{T_a - T_f}{h_i} \tag{12.12}$$

which can be rewritten to

$$Q_{\text{tot without ebr}} - Q_{\text{ebr}} = -k_i \frac{T_a - T_f}{h_i} \tag{12.13}$$

in which the heat flux term $Q_{\text{tot without ebr}}$ no longer contains the effective back radiation flux and in which T_f is the freezing temperature of seawater. If ice (and possibly snow) are present, the effective back radiation heat flux is no longer computed by Equation (12.11) but via

$$Q_{\text{ebr}} = \alpha T_X^4 \tag{12.14}$$

in which T_X either the ice temperature T_i or the snow temperature T_s, which depends on the occurrence of snow on top of the ice layer (threshold thickness: 1 mm). In case of snow on the ice, the ratio of the conductive coefficient k_i over the ice thickness h_i in Equation (12.12) changes to $(k_i k_s)/(h_i k_s + h_s k_i)$, with h_s and k_s the thickness of the snow layer and the thermal conductive coefficient for snow, respectively. We will from here refer either ratio as k_*/h_*. A value of 2.04 W m^{-1} K^{-1} is applied for k_i and 0.31 W m^{-1} K^{-1} for k_s. Note that the above-described iterative procedure is often applied in sea-ice models, e.g.: Wang *et al.* (2005), Mellor and Kantha (1989). By defining $T_X = T_p + \Delta T$, with T_p the previous value of T_X and linearizing Equation (12.11) in ΔT via $T_X^4 = T_p^4 + 4T_p^3\Delta T$, the temperature T_X can be determined by iteration, via

$$Q_{\text{tot without ebr}} - \alpha T_p^4 - 4\alpha T_p^3 \Delta T = -\frac{k_*}{h_*} T_p + \frac{k_*}{h_*} \Delta T \tag{12.15}$$

Next, Equation (12.15) can be rewritten to

$$Q_{\text{tot without ebr}} - \alpha T_p^4 - \frac{k_*}{h_*} T_p = \Delta T \left(4\alpha T_p^3 + \frac{k_*}{h_*} \right) \tag{12.16}$$

resulting in an air temperature correction ΔT per iteration as follows

$$\Delta T = \frac{Q_{\text{tot without ebr}} - \alpha T_p^4 - \frac{k_*}{h_*} T_p}{4\alpha T_p^3 + \frac{k_*}{h_*}} \tag{12.17}$$

In subroutine `preprocess_icecover.f90` this is coded with the following variables (units in brackets)

$$\texttt{coef1} = \alpha T_X^4 \text{ [J m}^{-2}\text{ s}^{-1}\text{]} \tag{12.18}$$

$$\texttt{coef2} = 4\alpha T_X^3 \text{ [J m}^{-2}\text{ s}^{-1}\text{ K}^{-1}\text{]} \tag{12.19}$$

$$\texttt{Qlong\_ice} = \alpha \text{ [J m}^{-2}\text{ s}^{-1}\text{ K}^{-4}\text{]} \tag{12.20}$$

$$\texttt{conduc} = k_* \text{ [J}^{2}\text{ m}^{-2}\text{ s}^{-2}\text{ K}^{-2}\text{]} \tag{12.21}$$

$$\texttt{D\_ice} = h_* \text{ [J s}^{-1}\text{ K}^{-1}\text{]} \tag{12.22}$$

$$\texttt{qh\_air2ice} = Q_{\text{tot without ebr}} \text{ [J m}^{-2}\text{ s}^{-1}\text{]} \tag{12.23}$$

$$\texttt{Qlong} = Q_{\text{ebr}} \text{ [J m}^{-2}\text{ s}^{-1}\text{]} \tag{12.24}$$

After convergence the back radiation Q_{br} is computed via

$$Q_{\text{br}} = \texttt{coef1} + \texttt{coef2}\Delta T \tag{12.25}$$

Since $Q_{\text{tot without ebr}}$ is known in advance of the iteration process, the total heat flux between the air and the ice can now be computed; see Equations 12.12 and 12.13. In practice, we observed that only two or three iterations for are typically required for convergence. In the code maximally five iterations are applied.

So far the heat flux between the air and the ice or snow layer has been described in this section. Now we will focus on the computation of the heat flux beneath the ice. In grid cells in which ice is present, the heat flux between the ice and the water reads

$$Q_{\text{ice-water}} = -C_{Tz}(T_f - T_s) \tag{12.26}$$

where T_s is the water temperature at the uppermost grid layer near the surface. The heat transfer coefficient C_{Tz} is given by

$$C_{Tz} = \frac{u_*}{B_T + P_{rt}\ln(-z/z_0)/\kappa} \text{ with } B_T = 3P_{rt}\sqrt{\frac{z_0 u_*}{\nu}}P_t^{2/3} \tag{12.27}$$

with u_* the friction velocity, P_{rt} a turbulent Prandtl number, z is the vertical coordinate corresponding to the temperature T, z_0 the roughness length and κ the Von Karman constant. The molecular sublayer correction is represented by B_T, P_t is the molecular Prandtl number and ν is the kinematic viscosity of water.

The freezing temperature of seawater T_f depends on the salinity concentration (Fofonoff and Millard Jr., 1983) of the upper water layer has been taken from and reads

$$T_f = (-0.0575 + 0.001710523\sqrt{S} - 0.0002154996S)S \tag{12.28}$$

with S the salinity concentration in ppt. This formula is also used in NEMO (Madec *et al.*, 2022). Noted that for fresh water (S = 0 ppt) the freezing point equals 0 ℃.

12.3 Ice effect on wind drag

The wind drag assuming open water $C_{d,w}$ is computed as described in section 11.1.2. Subsequently, we have implemented 6 options to compute the effective wind drag including the ice cover. These options can be specified by setting the keyword `ModifyWindDrag` in the `[ice]` block of the mdu-file to the indicated string.

- `none`: In this case there is no effect of the ice cover.

$$C_{d,\text{eff}} = C_{d,w} \tag{12.29}$$

- `linear`: In this case the drag is assumed to be proportional to the water area fraction A_w.

$$C_{d,\text{eff}} = C_{d,w} A_w \tag{12.30}$$

- `cubic`: This option is consistent with the default "IceCube" option of ADCIRC (Chapman and Massey).

$$C_{d,\text{eff}} = \max(C_{d,w}, c_0 + c_1 A_i + c_2 A_i^2 + c_3 A_i^3) \tag{12.31}$$

  with c_0 = 0.000 75, c_1 = 0.0075, c_2 = –0.009, and c_3 = 0.002.
- `lupkes_birnbaum`: This option follows Lupkes and Birnbaum (2005)

$$C_{d,\text{eff}} = C_{d,w} A_w + C_{d,i} A_i + C_{d,f} \text{ where} \tag{12.32}$$

$$C_{d,i} = 0.0015 \tag{12.33}$$

$$C_{d,f} = 0.00034 A_i^2 \frac{A_w \left(A_w^{0.8} + \frac{1}{2}(1 - \frac{1}{2} A_i)^2\right)}{31 + 90 A_i A_w} \tag{12.34}$$

- `andreas`: This option follows Andreas et al (2010)

$$C_{d,\text{eff}} = c_0 + c_1 A_i A_w \tag{12.35}$$

  with c_0 = 0.0015, and c_1 = 0.002 233.
- `raysice`: This option is consistent with the "RaysIce" option of ADCIRC (Chapman et al).

$$C_{d,\text{eff}} = \max(C_{d,w}, c_0 + c_1 A_i A_w) \tag{12.36}$$

  with c_0 = 0.001 25, and c_1 = 0.005.

12.4 Ice effect on pressure

The ice cover exerts a pressure on the water column in addition to the atmospheric pressure, such that the total pressure P at the surface becomes

$$P = P_a + A_i \rho_i g h_i \tag{12.37}$$

where P_a is the atmospheric pressure [Pa], A_i is the fraction of the area covered by ice [-], ρ_i is the density of the ice [kg m^{-3}], g is the gravitational acceleration [m s^{-2}], and h_i is the thickness of the ice layer [m]. This effect can be switched off by setting `ApplyPressure` to false.

12.5 Ice effect on flow resistance

The vectorial boundary condition for the momentum equations 8.3 and 8.4 at the water-ice interface ($z = \zeta$) is:

$$\frac{\nu_V}{H}\frac{\partial \boldsymbol{u}}{\partial \sigma}\bigg|_{\sigma=1} = \frac{1}{\rho_0}\boldsymbol{\tau}_{s,\text{ice}} \tag{12.38}$$

where

$$\boldsymbol{\tau}_{s,\text{ice}} = \rho_0 A_i C_{d,\text{ice}} \left|\boldsymbol{U}_{\text{flow}} - \boldsymbol{U}_{\text{ice}}\right| \left(\boldsymbol{U}_{\text{flow}} - \boldsymbol{U}_{\text{ice}}\right) \tag{12.39}$$

where ρ_0 is the water density [kg m^{-3}], $C_{d,\text{ice}}$ is the drag coefficient [-] of the ice as given by the user via `IceFricValue`, and $\boldsymbol{U}_{\text{flow}}$ is the near-surface flow velocity [m s^{-1}] and $\boldsymbol{U}_{\text{ice}}$ is the drift velocity of the ice [m s^{-1}] (currently, always 0). This effect can be switched on by setting `ApplyFriction` to true.

12.6 Ice effect on surface exchange

The effect of ice on the surface heat exchange is included in the thermodynamic model and always switched on, see section 12.2.2. The effect on evaporation is to be documented. Snowfall is prescribed via the precipitation input. In case of precipitation and air temperatures below zero, then this is considered as snowfall. Decay of the snow thickness is computed by the thermodynamic model. Currently both rainfall and snowfall is applied is areas with ice in case of precipitation and air temperatures below zero. In future this will be improved. The switch `ReduceSurfExch` does not yet have an effect.

12.7 Ice effect on waves

This effect can be switched on by setting `ReduceWaves` to true. This effect is currently implemented as a reduction factor on the contribution of the wave forces to the momentum equation. Hence, it will not reduce the effect of waves at other places, such as wave streaming. It is foreseen that this implementation will be updated to more generally reduce the waves where an ice cover is present. Note, SWAN can also include the effect of an ice cover, so in coupled D-Flow FM-D-Waves simulation, it's better to include the effect of the ice cover also on the D-Waves side. Unfortunately, this option is only available for a forced ice cover, and not yet for a dynamic ice cover.

12.8 Ice input options

This section describes how to switch on ice modelling in D-Flow FM. The main mdu-input file of D-Flow FM contains several blocks of input, such as **geometry**, **numerics** and **output**. In case of ice modelling an extra block with ice input has to be added, **ice**, as illustrated below.

```
[ice]
IceCoverModel = Semtner          # Type of ice model (None, External, Semtner)
```

Currently, three options are available, namely:

- `None`: No ice cover included. This is the default setting.
- `External`: The ice cover thickness and ice area fraction are prescribed by the user via external forcings.
- `Semtner`: The ice thickness is dynamically computed by D-Flow FM, and varies in space and time based on the meteorological input (air temperature, wind, cloudiness, relative humidity, etc.). This option is based on the approach of Semtner Jr. (1976).

Examples for the latter two cases are given in the following sections. However, first we'll give an overview of all keywords that can be specified in the **ice** block.

Table 12.1: *Ice input keywords*

Keyword	**Record description**
ice	
IceCoverModel	specifies the ice cover model to be used
ApplyPressure	logical flag indicating whether the ice pressure term should be included (1 logical, default: true)
ApplyFriction	logical flag indicating whether the ice friction term should be included (1 logical, default: false)
ReduceSurfExch	logical flag indicating whether surface exchange should be reduced (1 logical, default: false)
ReduceWaves	logical flag indicating whether waves should be reduced (1 logical, default: false)
ModifyWindDrag	effect of ice on wind drag (1 string) ◇ none (default) ◇ linear ◇ cubic ◇ lupkes_birnbaum ◇ andreas ◇ raysice
IceDensity	density of the ice (1 float, default: 917 kg m^{-3})
IceAlbedo	albedo of the ice cover (1 float, default: 0.75)
SnowAlbedo	albedo of the snow cover (1 float, default: 0.9)
IceFricType	type of friction specification at interface between ice and water (1 string) ◇ cdrag (default)
IceFricValue	drag coefficient at the ice-water interface (1 float, default: 0.005)
AddIceToHis	logical flag indicating whether the ice quantities should be written to the his-file (1 logical, default: false)
AddIceToMap	logical flag indicating whether the ice quantities should be written to the map-file (1 logical, default: false)

12.8.1 Ice modelling via `External`

In case of a prescribed ice thickness, the (old format) external forcings file (*.ext) has to be extended with a reference to the ice model input. This file is connected in the input block `[external forcing]` in the MDU-file, as illustrated below:

```
[external forcing]
ExtForceFile                         = ice_example.ext
```

The ice-related forcing in the *.ext file could for example be:

```
QUANTITY=sea_ice_thickness
FILENAME=ice_forcing.HICE
FILETYPE=6
METHOD=3
OPERAND=O
```

in which the file ice_forcing.HICE contains a space- and time-varying ice thicknesses on a computational grid, similar to wind forcing as explained in Sections 12.2.2 and 12.2.3. The suffix of the filename is arbitrary, but we prefer a representative name like HICE (thickness of ice) or future options like AICE (area/fraction of grid cell covered by ice). An example is shown below. In this example a 5-by-5 grid and only two points in time are used (0 and 6 hours after T_start). The ice thickness in this example is 0.0, 1.0, 1.15 or 1.26 m. In this way, an external ice thickness can be prescribed. We remark that only an ice thickness can be prescribed and not a snow thickness. The latter can have a significant effect on the ice growth or thaw, but does not directly influence the water flow. Therefore, it is sufficient to only prescribe a fixed ice thickness.

```
### START OF HEADER
filetype         =    meteo_on_curvilinear_grid
NODATA_value     =    -999.000
grid_file        =    extdata.grd
n_quantity       =    1
### END OF HEADER
/* TIME (HRS)      0.0  20000101
     0.00     1.00     1.00     1.00     0.00
     0.00     1.15     1.15     1.00     0.00
     0.00     1.15     1.15     1.00     0.00
     0.00     1.00     1.00     1.00     0.00
     0.00     1.00     1.00     1.00     0.00
/* TIME (HRS)      6.0 20000101
     0.00     1.00     1.00     1.00     0.00
     0.00     1.26     1.26     1.00     1.00
     0.00     1.26     1.26     1.00     0.00
     0.00     1.00     1.00     1.00     0.00
     0.00     1.00     1.00     1.00     1.00
```

12.8.2 Ice modelling via `Semtner`

Via this option the ice thicknesses are computed by the computational engine of D-Flow FM, based on the model input prescribed for the Composite heat flux model in D-Flow FM (`temperature`=2 in the mdu-file). See Chapter 11 for a more detailed explanation of this model. At present, ice modelling in D-Flow FM is only possible in combination with the Composite heat flux model, because this is currently the only heat flux model in D-Flow FM that computes temperatures that can be compared with measurements (i.e. so-called absolute heat flux model).

With this heat flux model, no additional input is needed to compute ice growth and thaw. The input is the same as used for regular thermodynamic modelling with the Composite heat flux model. This means that the required input for ice modelling is air temperature, relative humidity and cloudiness. Furthermore, wind forcing input is used in the computation of ice thicknesses. Thus, just adding `IceCoverModel = Semtner` to the mdu-file is enough to include ice modelling in a simulation; see section 12.2.2. As the name reveals, the implemented ice module is based on Semtner Jr. (1976).

In D-Flow FM it is possible to compute temperature below zero. This is of use for ice modelling, because for example the freezing point of saline water with a salinity concentrations of 35 ppt is close to –2 °C; see also Eq. (17). To enable negative temperature in D-Flow FM the two keywords `Allowcoolingbelowzero = 1` and `Tempmin = -3` (or another negative value) have to be added to the input block `[physics]`.

The description above is sufficient for the computation of ice thicknesses without any occurrence of snow. However, snow can have a significant impact on the ice thickness. Snow on top of ice acts like an 'insulating blanket' and can significantly reduce the growth of ice thickness. In D-Flow FM, for the computation of snow thickness, rainfall input and air temperature input are used. If there is any rainfall in a model, ideally specified with space- and time-varying input, and if the air temperature is below zero, D-Flow FM assumes that this represents snowfall. In the example below, the snowfall (rainfall at air temperatures below zero) is 10 cm in one day. Note that the below file is a time series file (*.tim), as described in section C.4.

```
   0.000000     0.0000000   !  rainfall or snowfall in mm/day
 72000.0000     0.0000000
 72001.0000   100.0000000
 73441.0000   100.0000000
 73450.0000     0.0000000
217440.0000     0.0000000
```

To illustrate, Figure 12.2 shows ice thicknesses (in blue) and snowfall (in red) for a cold winter. In early December ice growth starts till a maximum thickness of 10 cm. This is followed by a period of thaw, that makes the ice almost completely disappear. From mid-December a second period of below-zero temperatures occurs, leading to a maximum ice thickness of about 25 cm with snow and 35 cm without any snow. This clearly illustrates that the ice growth is slowed down by a snow layer.

12.9 Postprocessing

Currently, ice quantities are only written to the map output file; this can be switched on by setting `AddIceToMap` to true. Ice quantities cannot yet be written to the history output file, i.e. the flag `AddIceToHis` does not yet have any effect. The ice quantities available on the map output file are listed in Table 12.2. The names on the output file are a combination of the names in the columns "long name" and "dimensions".

Note that time series can be retrieved by selecting model output in an individual grid cell. This was done to generate Figure 12.2. In general, the map output frequency will be much lower than the history output frequency. However, time variation of ice and snow is much slower than that of other quantities (water levels, currents, salinity, etc.).

Figure 12.2: *Illustration of computed ice thickness in case of no snow (in blue), ice thickness in case of snow (in green) and snow thickness (in red)*

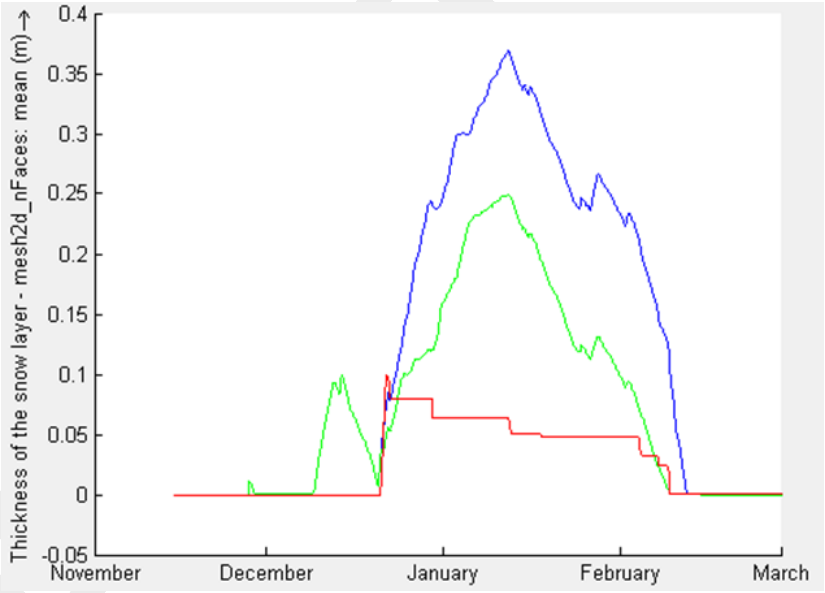


Table 12.2: *ice quantities on map output file*

long name	name on file	units	dimensions
Fraction of the surface area covered by floating ice		-	mesh2d_nFaces
Thickness of floating ice cover		m	mesh2d_nFaces
Pressure exerted by the floating ice cover		$\mathrm{N\,m^{-2}}$	mesh2d_nFaces
Thickness of the snow layer		m	mesh2d_nFaces

12.10 Limitations

Ice modelling in D-Flow FM currently has the following limitations:

- ◇ No history output for ice thickness and snowfall
- ◇ No horizontal transport of ice
- ◇ No open boundaries for ice
- ◇ No restart of ice

13 Hydrological processes

Hydrological processes describe how water moves across land and through the hydrologic cycle. For certain hydrodynamic simulations theses processes can not be ignored, for example for pluvial flooding. The actual flow over land is still described by shallow water hydrodynamics (chapter 8), but additionally it is possible to add hydrological parameter fields and boundary conditions to a model schematization.

13.1 Notation

The used symbols in this chapter build upon the hydrodynamics notation introduced in section 8.3.2. In this section, additional symbols are introduced for the hydrological processes.

Hydrology:

$A_{\mathrm{re},k}$ Bottom area of a grid cell for rain and evaporation.

The difference between wet area A_k and "rainfall-evaporation area" $A_{\mathrm{re},k}$ is quite subtle. While they are the same in a 2D-description[1], they usually differ in the 1D case. Figure 13.1 shows this by comparing two channels in the 1D and the 2D case.

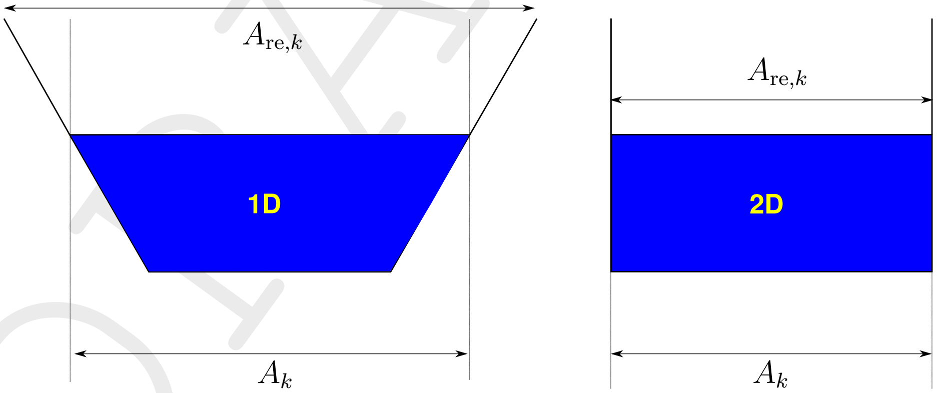


Figure 13.1: Vertical cross section view of wet area and hydrological area in a 1D and a 2D grid cell.

13.2 Precipitation

Precipitation falling onto the surface (land or water) acts as a source of water for the system. The state of precipitation (solid or liquid) is ignored, so all precipitation that falls, contributes directly to the mass balance. Hereafter, it is often referred to as rainfall.

13.2.1 Meteo forcing input

Rainfall is prescribed by a data file containing the rainfall intensities or amounts, and this file should be referenced in the external forcings file. An example rainfall block looks like:

Example:

```
[General]
fileVersion          = 2.01
fileType             = extForce
```

[1] This is not the case when Conveyance2D=3.

```
[Meteo]
quantity            = rainfall
forcingFile         = precip.nc
forcingFileType     = netcdf
targetMaskFile      = roofs.pol
targetMaskInvert    = true
interpolationMethod = nearestNb
```

The external forcings syntax for meteo is further described in section C.5.2.3. The supported data formats for the actual rainfall are:

- Global (spatially uniform) time series in <.bc> ascii format (section E.2.3);
- Time series on meteo stations in netCDF (section E.2.4);
- Gridded meteo data (space- and time-dependent) in netCDF (**??**);

13.2.2 Rainfall distribution across the model grid

The total rainfall amount is determined at the start of every timestep and added to the total volume source term for each grid cell. The total rainfall flux entering a grid cell is simply the local rainfall intensity times the maximum grid cell area, *if* that grid cell is allowed to capture rainfall. A grid cell is allowed to capture rainfall when it satisfies the following conditions:

- The grid cell is 2D[2];
- Or: the grid cell is 1D and is *not* overlapped by any other 2D grid cell. This excludes 1D embedded grid points and 1D sewer pipes below street level;
- Finally, the cell must be selected by the target mask if and only if such a target mask is present.

The maximum grid cell area used for catching rainfall is defined as:

- normal bed area for 2D grid cells;
- sum of maximum flow link areas for 1D open channel grid cells; these areas are based on the maximum flow width of the relevant cross sections and half the flow link length;
- the bottom area of a 1D storage node, if `useTable=false`;
- Note: the bottom area of a 1D storage node with `useTable=true` is yet undefined.

The rainfall intensity at each grid cell is interpolated from the source data to the cell centers. The type of interpolation depends on the settings supplied. Figure 13.2 shows nearest neigbour interpolation of three meteo stations onto the model grid.

Additionally, an optional target mask can be supplied in a polygon file. The grid cells whose center lies outside (or inside) of the polygon(s) are then excluded from receiving rainfall.

13.2.3 Rainfall intensities versus rainfall amounts

Some datasets represent the precipitation as an intensity (e.g., millimeters per day). The external forcings quantity name is then `quantity = rainfall_rate`. Such input timeseries can be interpolated in time as a linear function, so the resulting rainfall intensity during the simulation is continuous and piecewise linear. Other datasets may represent the precipitation as an amount per each data time interval (e.g. total millimeters for the past interval). Such input timeseries can be interpolated in time as a block function. The external forcings quantity name is then `quantity = rainfall`. The average intensity is derived as the total rain amount divided by the data time interval (which can also be non-equidistant in time). The

[2] In 3D models, "2D" means that the cell is 2D in the horizontal input grid.

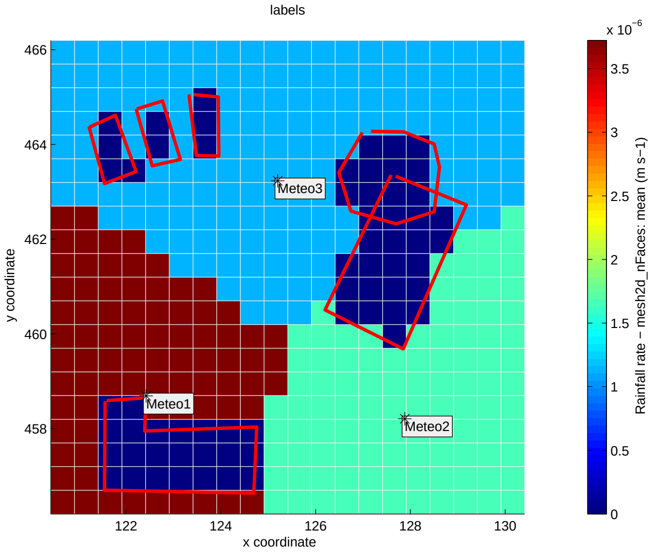


Figure 13.2: *Rainfall rates by nearest neighbour interpolation, with some areas masked out.*

resulting rainfall intensity during the simulation is then typically discontinuous and piecewise constant in time.

13.3 Interception

Interception in a hydrological context, refers to interception of rain such that it does not reach the ground. The precipitation can be intercepted by, e.g., leaves or roofs. Interception is modelled by maintaining a separate basket on each grid cell, that stores some water from precipitation. Once it is full, all precipitation continues to fall onto the ground directly. Interception can be enabled by setting `InterceptionModel=1` in the MDU file under `[hydrology]` (see Appendix A). The thickness of the interception layer is then prescribed via spatial input (see Initial field file in section D.2).

13.3.1 Numerical treatment of interception

Interception is treated explicitly. It is not handled in the matrix solver, but directly subtracted from current rain intensities. This behaviour is described by equations (13.1) and (13.2):

$$Q_{\text{icept},k} = \min\left(Q_{\text{in,rain},k}, A_{\text{re},k}/\Delta t \cdot \left(H_{\text{icept},k} - h_{\text{icept},k}\right)\right), \tag{13.1}$$

$$Q_{\text{in,rain},k} -= Q_{\text{icept},k}, \tag{13.2}$$

where $H_{\text{icept},k}$ is the thickness of the interception layer and $h_{\text{icept},k}$ is the water depth in the interception layer. Evaporation from the interception layer is treated in section 13.5.2.

13.4 Infiltration

Infiltration of water into the soil can be modeled in several ways. Three infiltration models are currently supported:

- ◇ Maximum (constant) infiltration capacity
- ◇ Function of pressure
- ◇ Horton's infiltration equation

13.4.1 Horton's equation

Horton's infiltration equation (Horton, 1933) is an empirical formula which describes the trend of the infiltration capacity. A decreasing infiltration capacity is described by

$$f_t = f_e + (f_b - f_e)e^{-k_a t}, \text{ if the surface is wet}$$

and a recovering infiltration capacity is described by

$$f_t = f_b - (f_b - f_e)e^{-k_h t}, \text{ if the surface is dry,}$$

where the symbols used are:

t	Time	[hr]
f_t	Infiltration capacity at t	[mm/hr]
f_b	Maximum infiltration capacity at $t = 0$	[mm/hr]
f_e	Minimum infiltration capacity	[mm/hr]
k_a	Time factor of decreasing infiltration capacity	[1/hr]
k_h	Time factor of recovering infiltration capacity	[1/hr]

It is assumed that at the beginning of each rainfall event, the infiltration capacity is at its maximum f_b. The infiltration capacity is decreasing as long as there is water stored on the surface. If the infiltration capacity is at its the maximum value $f_t = f_b$ and there is no water on the surface, it will remain at its maximum value. Vice versa, if the infiltration capacity is at its minimum value $f_t = f_e$, and there is still water on the surface, it will remain at its minimum value. The infiltration capacity will recover again as soon as the surface is dry and it is not raining.

13.5 Evaporation

Water loss via evaporation can be enabled by setting `Evaporation=1` in the MDU file under `[external forcing]`. The potential evaporation $E_{\mathrm{p},k}$ is then prescribed via spatial input (see Initial field file in section D.2). The actual evaporation is only limited by the available volume as described in section 13.5.1. Other hydrological factors such as soil moisture, are not modelled.

Note: Originally, evaporation was realized by specifying negative rainfall intensities. This functionality is kept for backwards-compatibility only and should not be used for new models.

13.5.1 Numerical treatment of evaporation

Evaporation is a negative source of volume and as such is subtracted from the right-hand side source term of the continuity equation (8.1). This is described by equation (13.3)

$$Q_{\mathrm{in},k} -= \min\left(\frac{1}{2}V_k/\Delta t + Q_{\mathrm{in,rain},k}, |E_{\mathrm{p},k} \cdot A_{\mathrm{re},k}|\right) \tag{13.3}$$

where $Q_{\mathrm{in,rain},k}$ describes the incoming rain. The negative evaporation flux is limited based on the available water volume in the grid cell in each timestep. The relaxation factor of $\frac{1}{2}$ limits the negative source term contribution when little water is available, to avoid negative water depths.

Once all source terms have been collected in $Q_{\mathrm{in},k}$, the real calculation of new water levels is done. The resulting water level at that grid point is then calculated implicitly by the matrix solver. Or, in the case of a negative source term $Q_{\mathrm{in},k}$, it is solved explicitly in the continuity equation.

13.5.2 Evaporation from interception layer

Evaporation from the interception layer can occur simultaneously with open water evaporation in the same grid cell. It is computed explicitly after precipitation was added (if any). The negative evaporation flux can simply be subtracted from the interception layer, under the condition that it is capped by volume available in the interception layer. This is described by equations (13.4) and (13.5)

$$
\begin{aligned}
Q_{\mathrm{evap,icept},k} &= \min\left(h_{\mathrm{icept},k} \cdot A_{\mathrm{re},k}/\Delta t, |E_{\mathrm{p},k} \cdot A_{\mathrm{re},k}|\right) && (13.4) \\
h_{\mathrm{icept},k} &-= Q_{\mathrm{evap,icept},k} \cdot \Delta t / A_{\mathrm{re},k} && (13.5)
\end{aligned}
$$

where $Q_{\mathrm{evap,icept},k}$ is the actual evaporation flux from the interception layer.

13.6 Lumped rainfall-runoff coupling

Rainfall-runoff processes can also be computed in a *lumped* way, instead of the *distributed* way in section 13.2. In this case, the rainfall runoff is computed elsewhere (for example by D-Rainfall Runoff) and translated into a set of single point lateral discharges in the D-Flow FM model schematization. This is further discussed in a separate chapter 18. The current chapter continues to focus on *distributed* hydrology instead.

13.7 Hydrological output

When hydrological processes are enabled in a model schematization, additional output can be produced in the various output files.

13.7.1 Spatial hydrological output in map file

The map file can contain extra variables for the current hydrological fluxes, for example rainfall intensities or actual evaporation. Table F.19 lists all available output quantities.

13.7.2 Time series hydrological output in history file

The history file can contain some extra hydrological volume totals that contribute to the overall mass balance. This output is enabled by one global setting in the MDU: `[output] Wrihis_balance = 1`. More details on this volume output is available section F.3.1.6.

14 Hydraulic structures

Obstacles in the flow may generate sudden transitions from flow contraction to flow expansion. The grid resolution is often low compared to the gradients of the water level, the velocity and the bathymetry. The hydrostatic pressure assumption may locally be invalid. Examples of these obstacles in civil engineering are: gates, barriers, dams, groynes and weirs. The obstacles generate energy losses and may change the direction of the flow.

The forces due to obstacles in the flow which are not resolved (sub-grid) on the horizontal grid, should be parameterised. The obstacles are denoted in D-Flow FM as hydraulic structures. In the numerical model the hydraulic structures should be located at velocity points of the grid. The direction of the forces should be specified at input. To model the force on the flow generated by a hydraulic structure, a quadratic energy or linear loss term is added to the momentum equation.

Note: Structure definitions can be made in Delta Shell, and will then be saved into a structures <∗.ini>-file (section C.12).

The user can insert the hydraulic structures by the means of polygons on the grid. By selecting the required structure and drawing a polygon on the computational grid, the location of structure can be defined (see Figure 14.1). The supported structures in D-Flow FM are

- Fixed Weirs
- Dambreaks
- (Adjustable) general structures, weirs, gates, orifices, and universal weirs,
- Culverts
- Long culverts
- Bridges
- Pumps
- Compound structures
- Thin dams

In practice, the word ’barrier’ is often used for structures. However, in D-Flow FM such structures are modelled as a gate or a weir in combination with a so-called Control Group (D-Real Time Control model).

Figure 14.1: Selection of structures (and other items) in the toolbar.

14.1 Fixed weirs

In D-Flow FM, a fixed weir is a fixed non-movable construction generating energy losses due to constriction of the flow. They are commonly used to model sudden changes in depth (roads, summer dikes) and groynes in numerical simulations of rivers. Such structures are applied to keep the river in its bed for navigation purposes. Others are built, for instance, to protect an area behind a tidal weir from salt intrusion. Upstream of a weir the flow is accelerated due to contraction and downstream the flow is decelerated due to expansion. The expansion introduces an important energy loss due to turbulence. The energy loss is dependent on the shape of the weir and its crest level.

Fixed weirs are located in D-Flow FM on the velocity points of the computational grids. For a description of the input of fixed weirs, we refer you to section 5.4.2.5 and C.8.

In D-Flow FM, three different approaches are applied to simulate the energy losses by fixed weirs.
First of all, a numerical approach has been implemented. Then, a special discretization of the advective terms around the fixed weir is applied. For a detailed description we refer to Deltares (2024a, section 6.7.1 – 6.7.4).

Next to the numerical approach, there is an empirical approach. Based on measurements in flume laboratories empirical formulae can be derived in order to match the measurements as accurately as possible. In this empirical approach, the energy loss due to a weir is described by the loss of energy height ($[m]$). The energy loss in the direction perpendicular to the weir is denoted as ΔE. This energy loss is added as an opposing force in the momentum equation by adding a term $-g\Delta E/\Delta x$ to the right hand side of the momentum equation, resulting in a jump in the water levels by ΔE at the location of the weir. For the computation fo the energy loss ΔE two options are availabe in D-Flow FM, namely the so-called 'Tabellenboek' and 'Villemonte' approaches. The two corresponding empirical formulas have been taken from the Simona software suite. For a detailed description of both formulas we refer to Deltares (2024a, section 6.7.5 – 6.7.6). In this empirical approach the discretization of the advective terms does not change, unlike the numerical approach in D-Flow FM for modelling fixed weirs.

The third approach is only available for fixed weirs that are located on 1D2D connections. For this approach a weir formula is used to compute the flow over the fixed weir. For a detailed description of this approach see Deltares (2024a, section 6.7.8). This approach can be turned on by putting the parameter `FixedWeirScheme1d2d` under the paragraph `[numerics]` in the MDU-file to 1. NOTE: The non-1D2D fixed weirs will not be affected by this parameter.

14.2 Dam break

The dam break is a structure that models a growing breach after a dam failure or levee breach. A dam break is not really a hydraulic structure, but since it shares several properties with other structures (in terms of input and output), it is included here. The calculation of flow through a breach is based on a fixed weir (section 14.1), which models energy losses across the crest. Therefore, one important requirement of a dam break is that, a fixed weir polyline must be present on the same location as the dam break polyline. The dam break structure lowers the crest level and widens the breach width over time.

The input file format for a dam break is described in section C.12.11 and the output options in section F.3.1.9.8.

In this section, firstly, the geometrical grid snapping of dam break polylines to 2D grid lines is discussed. Then the computation of breach growth of a dam break is discussed. This includes two empirical formulas for computing breach growth: Verheij–van der Knaap (2002) and van der Knaap(2000).

14.2.1 Grid snapping of a dam break

The potential location of a dambreak is defined by an input dam break polyline. This should coincide with (part of) a fixed weir polyline. When initializing a model, the geometrical snapping of these polylines to the grid cell edges is identical for both. The rule of this snapping is, if the polyline intersects with a flow link, then it is snapped to the cell edge that intersects with this flow link.

The user defines the start location of the dam breach by means of providing the coordinates (`startLocationX`, `startLocationY`). The actual starting point of the dam breach is determined by the following steps:

1 The nearest point on the dam break polyline to the start location is determined, by projecting the start location onto the polyline.
2 For all flow links in the dam break, the intersection of these flow links with the dambreak polyline are computed.
3 The flow link with its intersection that is closest to the the projected start location will be used as the actual midpoint location of the dam breach.

More details are available in the section about fixed weir snapping in (Deltares, 2024a, section 6.7.7). Note that a start location for a breach that is placed in the middle of a dam break, may not be located in the middle after snapping. This is most obvious when a dam break snaps to an even number of flow links/grid edges equal lengths. This is of particular importance when the breach width grows to about the full width of the dam break. See section 14.2.2.4 for more details.

Fixed weirs and dambreaks on a 2D grid

To correctly model the crest width of a fixed weir (and the breach width), the flow link widths at the grid-snapped polylines are adjusted. Assume that the fixed weir polyline intersects with a flow link j', associated with cell edge j. Then flow link width w_{u_j} is computed as:

$$w_{u_j} = \cos(\alpha)\ ||\mathit{Edge}||_j, \tag{14.1}$$

where

w_{u_j} is the flow link width,
$||\mathit{Edge}||_j$ is the original length of cell edge j (i.e., the original width of flow link j'), and
α is the angle between the fixed weir polyline and cell edge j.

In other words, w_{u_j} is the projection of the original width of edge j onto the fixed weir polyline. As a result, when summing w_{u_j} for all affected flow links j', the total width is similar to the length of the polyline.

Fixed weirs and dambreaks on 1D2D flow links

In case a fixed weir and/or a dambreak intersects a 1D2D flow link (lateral or embedded), no projection is applied. As a result the flow width w_{u_j} of this flow link is used. The calculation of the flow width of a 1D2D lateral or embedded flow link is described in Section 8.9.3.

14.2.2 Computation of breach growth

Two formulas are available to compute the breach growth of a dam break in D-Flow FM: Verheij–Van der Knaap(2002) and Van der Knaap (2000). These are discussed first. Alternatively, breach width and depth may be provided by the user as a time series. Finally, the details of a breach width spanning multiple flow links is discussed.

Both breach growth formulas regard the process of breach growth as two phases:

- ◇ **Phase 1**: during which the breach level, for a constant initial breach width, is lowering from its initial crest level z_{crest} (or dike level) to its minimal crest level z_{min}. In other words, this phase is a "lowering" phase. The breach level decreases while the breach width stays unchanged.
- ◇ **Phase 2**: during which the breach only grows in width under a certain condition depending on the formula in use. The Verheij–Van der Knaap (2002) formula increases the breach width based on the water level jump, as long as the flow velocity through the breach is larger than the critical flow velocity for eroding sediment/soil particles u_c. The Van der Knaap (2000) formula computes the breach width as a function of time and limits it by a maximal breach width. In other words, this phase is a "widening" phase. The breach level stays at the minimal level and the breach width increases.

14.2.2.1 The Verheij–Van der Knaap (2002) formula

One can choose to use the Verheij–Van der Knaap (2002) formula by setting the input parameter `algorithm=2` in Table C.18. The formula is as follows:

Phase 1:

For $T_{\text{start}} \leq t \leq T_{\text{start}} + t_{\text{Phase 1}}$

$$B(t) = B_0, \tag{14.2}$$

$$z(t) = z_{\text{crest}} - \left(\frac{t - T_{\text{start}}}{t_{\text{Phase 1}}}\right)(z_{\text{crest}} - z_{\text{min}}). \tag{14.3}$$

Phase 2:

For $t > T_{\text{start}} + t_{\text{Phase 1}}$ (hence $B(t) \geq B_0$):

$$z(t) = z_{\text{min}}, \tag{14.4}$$

$$B(t_{i+1}) = B(t_i) + \frac{\Delta t}{3600}\left(\frac{\partial B}{\partial t}\right)_{t_{i+1}}, \tag{14.5}$$

where

$$\left(\frac{\partial B}{\partial t}\right)_{t_{i+1}} \begin{cases} \frac{f_1 f_2}{\ln(10)} \frac{(g(h_{\text{up}} - \max(h_{\text{down}}, z_{\text{min}})))^{3/2}}{u_c^2} \frac{1}{1 + \frac{f_2 \ g \ (t_{i+1} - t_1)}{3600 \ u_c}} & \text{if} \quad |u| > u_c \\ 0 & \text{if} \quad |u| \leq u_c \end{cases}. \tag{14.6}$$

The notations in the above formula are:

f_1 (input parameter) Factor 1: constant factor [$-$],
f_2 (input parameter) Factor 2: constant factor [$-$],
$B(t)$ Width of the dam breach at time t [m],
B_0 (input parameter) Initial width of the dam breach [m],

$\left(\frac{\partial B}{\partial t}\right)_{t_i+1}$ Rate of breach width growth in time-step Δt from t_i to t_{i+1} [m/s],

g Acceleration due to gravity [m/s^2],

h_{up} Upstream water level at time t [m AD],

h_{down} Downstream water level at time t [m AD],

t Current model time [s],

$t_{\text{Phase 1}}$ (input parameter) Duration of phase 1, the time period over which the breach for a constant initial width (B_0) is lowered from its initial crest level z_{crest} (or dike level) to its minimal crest level z_{min} [s],

$t_1 = T_{\text{start}} + t_{\text{Phase 1}}$ The time [s] when the maximum breach-depth (z_{min}) is reached, i.e. when the Phase 1 finishes,

T_{start} (input parameter) Model time at which the development of the initial breach starts [s],

u Average flow velocity in the breach [m/s],

u_c (input parameter) Constant critical flow velocity for eroding sediment/soil particles.

$z(t)$ Crest level of the dam breach at time t [m AD],

z_{crest} (input parameter) Initial breach level at $t = T_{start}$ [m AD],

z_{min} (input parameter) Minimal breach level at $t = t_1$ [m AD],

Δt Computational time step [s], equal to $t_{i+1} - t_i$.

Settings of the input parameters can be found in section C.12.11.

Upstream and downstream water levels (h_{up} and h_{down})

By default the upstream and downstream water levels are computed by averaging the waterlevels upstream and downstream of the dambreak. Only grid cells that are connected to a flow link that is (partly) opened by the dambreak are included in the averaging. For the averaging of upstream and downstream water level the original flow width of the open flow links is used as a weight factor.

Another possibility is to define custom up and downstream water level locations. In Section C.12.11 the input settings for this option can be found.

Recommended values of input parameters in the Verheij–Van der Knaap (2002) formula

To use proper values, a list of recommended values as well as the range of values is given in Table 14.1.

Table 14.1: *Recommended values of input parameters in the Verheij–Van der Knaap (2002) formula.*

Parameter	recommended value	Range
f_1	1.3	0.5 - 5
f_2	0.04	0.01 - 1
B_0	10	1 - 100 m
$t_{\text{Phase 1}}$	0.1 hours	0.1 - 12 hours
u_c	0.2 m/s	0.1 - 10 m/s
z_{crest}	-	-
z_{min}	-	-

Characterising soil strength

As a recommendation, the parameter u_c, applied in the Verheij–Van der Knaap (2002) formula may be derived from Table 14.2, where τ_c is the critical shear stress.

Table 14.2: *Recommended values for u_c and τ_c.*

Soil type	**Bodemtype (Dutch)**	u_c [m/s]	τ_c [Pa]
grass, good	gras, goed	7	185
grass, moderate	gras, matig	5	92.5
grass, bad	gras, slecht	4	62
clay, very good (compacted)	klei, zeer goed (compact; $t_{\text{ongedraineerd}}$ = 80 – 100 kPa)	1.0	4
clay, good (firm)	klei met 60 % zand, (stevig; $t_{\text{ongedraineerd}}$ = 40 – 80 kPa)	0.8	2.5
clay, moderate (little structure)	goede klei met weinig structuur	0.7	2
clay moderate (considerable structured)	goede klei, sterk	0.6	1.5
clay, bad (weak)	slechte klei (slap; $t_{\text{ongedraineerd}}$ = 20 – 40 kPa)	0.4	0.65
sand with 17 % silt	zand met 17 % silt	0.225	0.20
sand with 10 % silt	zand met 10 % silt	0.20	0.15
sand with 0 % silt	zand met 0 % silt	0.16	0.10

Another way to compute u_c is using the critical shear stress τ_c. For example, if τ_c is known, then u_c can be calculated by

$$u_c = 0.5\sqrt{\tau_c}, \tag{14.7}$$

or

$$u_c = u_{c,\text{sand}}(1 + 0.01\alpha P_{\text{clay}}) + \beta(0.65 - v), \tag{14.8}$$

where using the defaults, factor $\alpha = 15$, factor $\beta = 1$, critical flow rate for sand $u_{c,\text{sand}} = 0.2$ m/s, P_{clay} is percentage of clay, and the voids ratio v can be computed by

$$v = n/(1 - n), \tag{14.9}$$

where n is the soils pore fraction (default $n = 0.4$).

For more details of formula Verheij–Van der Knaap (2002) we refer to Verheij (2002).

14.2.2.2 The Van der Knaap (2000) formula

One can choose to use the Van der Knaap (2000) formula by setting the input parameter `algorithm=1` in Table C.18. The formula is as follows:

Phase 1:

For $T_{\text{start}} \leq t \leq T_{\text{start}} + t_{\text{Phase 1}}$

$$B(t) = B_0, \tag{14.10}$$

$$z(t) = z_{\text{crest}} - \left(\frac{t - T_{\text{start}}}{t_{\text{Phase 1}}}\right)(z_{\text{crest}} - z_{\text{min}}). \tag{14.11}$$

Phase 2:

For $t > T_{\text{start}} + t_{\text{Phase 1}}$ (hence $B(t) \geq B_0$):

$$z(t) = z_{\text{min}}, \tag{14.12}$$

$$B(t) = \min\left(c_1 \log_{10}(\frac{t}{c_2}), B_{\text{max}}\right). \tag{14.13}$$

In Equation (14.13), the coefficients c_1, c_2 and allowed maximal breach width B_{max} depend on the material of the dam. Two types of materials are possible: clay and sand. (Currently only clay is supported in D-Flow FM.) The values of these coefficients are shown in Table 14.3.

Table 14.3: *Values of coefficients c_1, c_2 and B_{max} of the Van der Knaap (2000) formula for different material types.*

Material type	c_1	c_2	B_{max}
clay	20	288	75 m
sand	67	522	200 m

Other notations in the above formula are:

$B(t)$	Width of the dam breach at time t [m],
B_0	(input parameter) Initial width of the dam breach [m],
t	Current model time [s],
$t_{\text{Phase 1}}$	(input parameter) Duration of phase 1, the time period over which the breach for a constant initial width (B_0) is lowered from its initial crest level z_{crest} (or dike level) to its minimal crest level z_{min} [s],
T_{start}	(input parameter) Model time at which the development of the initial breach starts [s],
$z(t)$	Crest level of the dam breach at time t [m AD],
z_{crest}	(input parameter) Initial breach level at $t = T_{\text{start}}$ [m AD],
z_{min}	(input parameter) Minimal breach level at $t = t_1$ [m AD],

Settings of the input parameters can be found in section C.12.11.

For more details about formula Van der Knaap (2000) we refer to Van der Knaap (2000).

14.2.2.3 Breach crest level and width time series

One can choose to explicitly prescribe the breach's crest level and width as time series by setting the input parameter `algorithm=3` in Table C.18. The provided time series must be in a 3-column <*.tim> file (in minutes), specified by parameter `dambreakLevelsAndWidths`. The times in the file are relative to T_{start} (`t0`, in seconds).

14.2.2.4 Method when breach grows wider than one flow link

The breach starts at only one flow link, which is defined by setting `startLocationX` and `startLocationY` in the input, see section C.12.11. The breach widths during the simulation are computed by either the formulas for breach growth or by a time series. At a certain time, if the breach width is wider than the starting flow link, D-Flow FM must determine the affected links so that the width grows, from the starting flow link, to gradually include more edges. How the breach extends beyond the starting flow link is determined by the `breachGrowth` keyword in the <mdu-file>; three options have been implemented.

- `symmetric` In this case the breach grows symmetrically on both left and right sides of the breach. This is the original implementation. This approach triggers a warning when the breach reaches the end of the snapped dam break on *one* of the sides.
- `symmetric-asymmetric` In this case the breach is assumed to grow symmetrically on both left and right sides of the breach until it reaches the end of the dam break on one of the sides; thereafter it only growth on the other side. This is the current default. This approach triggers a warning only when the breach width is larger than the full length of the snapped dam break.
- `proportional` In this case the breach is assumed to grow proportionally to the available width on the two sides of the breach. This approach triggers a warning only when the breach width is larger than the full length of the snapped dam break.

Given the breach width on one of the sides, for instance a width on the left side w_{left}, links on the left side are opened one by one in sequence gradually moving further away from the starting flow link:

- **Situation 1:** If w_{left} is smaller than the width of the considered flow link, then this is end of the searching.
- **Situation 2:** If w_{left} is larger or equal to the width of the considered flow, the value of w_{left} is decreased by this flow link width and the process continues with the next flow link that is on the left side of this link. This is repeated until Situation 1 is achieved, or the end of the dam break is reached.

In both two situations, the width and bed level of the considered link is affected. The bed level of this link is adjusted to the maximal value of its original bed level and the crest level of the dam break. The width of this link is adjusted to the effective dam breach width, i.e. the minimum of the remaining breach width w_{left} and the width of the flow link.

14.3 General structure

The General structure combines weir and gate flow into one structure. Additionally, the General structure gives the highest degree of freedom in defining the the geometry of the hydraulic structure and coefficients affecting the flow. The geometrical shape is given in Figure 14.2 and Figure 14.3. The discharge through a General structure is computed based on upstream and downstream energy levels. The input file format for a general structure is described in section C.12.10.

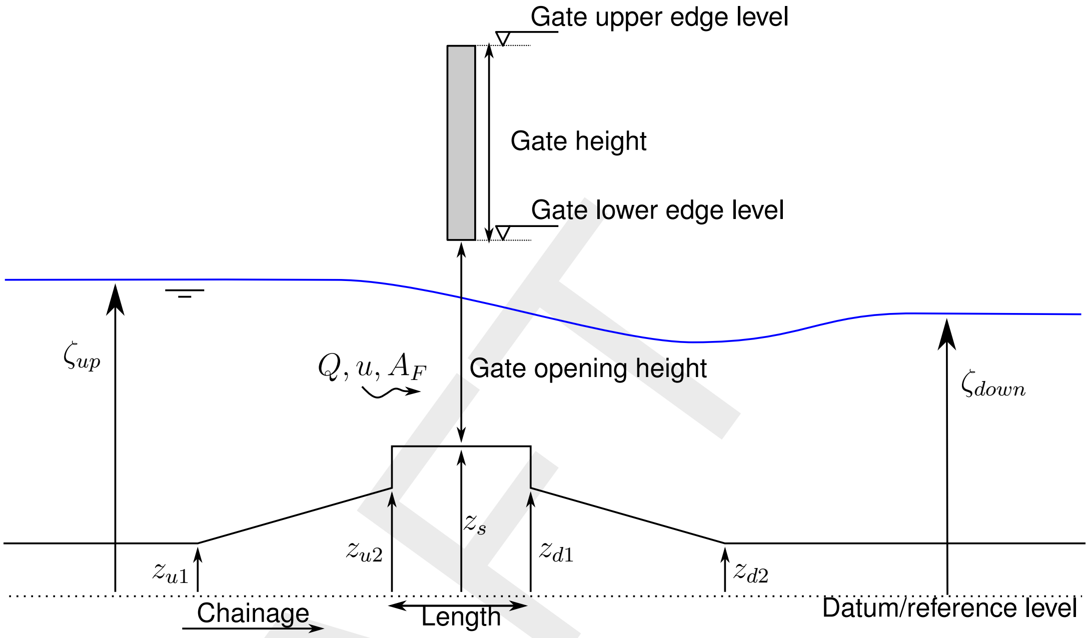


Figure 14.2: *General structure, side view*

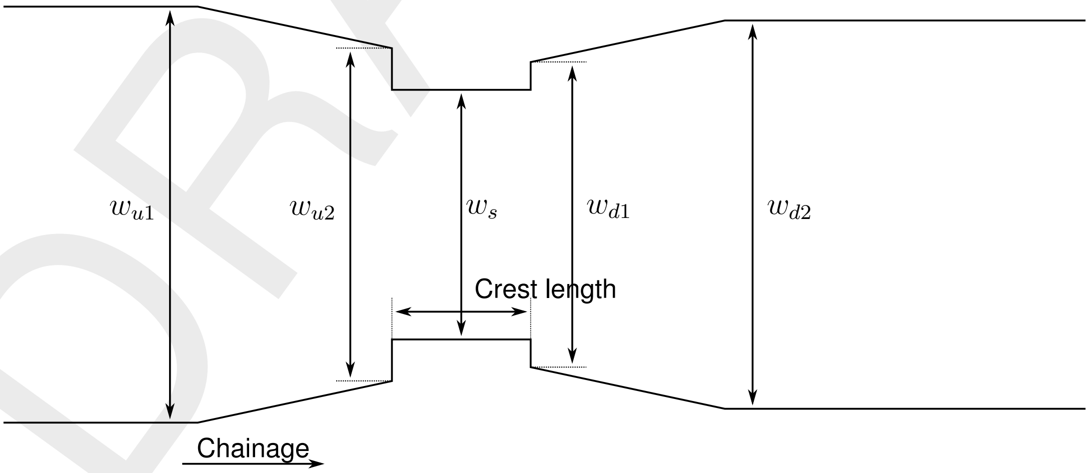


Figure 14.3: *General structure, top view*

The gate of the General structure can be configured as two doors with an opening in between as shown in Figure 14.4. When the modelled gate is a single door spanning the full width, the Gate Opening Width should be set to zero.

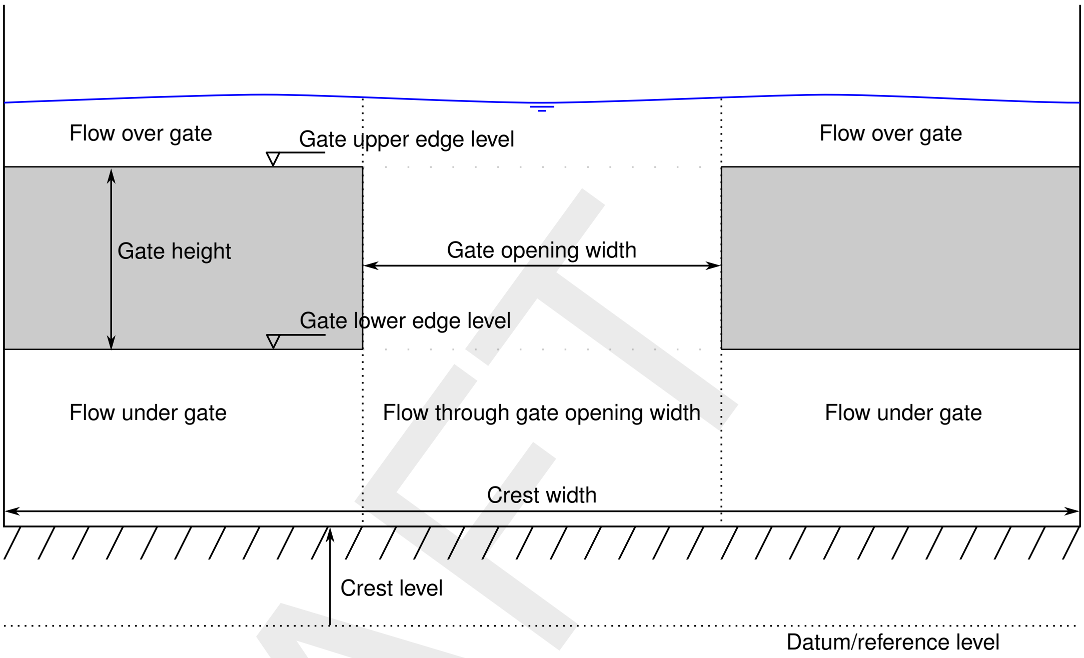


Figure 14.4: *General structure, front view*

Figure 14.4 also shows the three different geometrical parts where the General structure can have flow: under gate flow, over gate flow and between gate flow. The total discharge through the structure is calculated by adding up the discharge through the separate parts. For each part of the structure opening the downstream waterlevel is calculated and subsequently the flow type (gate or weir, free or submerged) is determined.

Remark:

◇ Please keep in mind that in the following part of this section the description for the General structure applies to the three different openings (with each its own flow conditions).

During the simulation a number of checks and possibly corrections on some dimensions of the general structure are performed.

◇ If the **crestLevel** is below the bed level of the surrounding water level points in the channel, the crestLevel (z_s) is raised to the highest bed level of these two points[1].

◇ If z_{u1} and/or z_{u2} are below the bed level of the upstream water level point in the channel, z_{u1} and/or z_{u2} are raised to the bed level of the upstream water level point in the channel.

◇ If z_{d1} and/or z_{d2} are below the bed level of the downstream water level point in the channel, z_{d1} and/or z_{d2} are raised to the bed level of the downstream water level point in the channel.

◇ All **widths** (w_s, w_{u1}, w_{u2}, w_{d1}, w_{d2}) are maximized with the corresponding flow width.

◇ If the **gateLowerEdgeLevel** is lower than the crestLevel, the gateLowerEdgeLevel is set to the crestLevel.

◇ If the **gateHeight** is less than or equal to 0 an error message is produced.

◇ If the **gateOpeningWidth** is less than 0, the gateOpeningWidth is set to 0. Else if the gateOpeningWidth is larger than the crestWidth then the gateOpeningWidth is set to the crestWidth.

[1] The bed level check is actually based on the face-based bed levels of the velocity point. In typical applications, this is equal to the maximum bed level of the two surrounding water level points. More information is available in (Deltares, 2024a, Equation (6.10)).

Flow across the General structure can be of the following types:

- free gate flow (i.e. supercritical gate flow),
- submerged gate flow (i.e. subcritical gate flow),
- free weir flow (i.e. supercritical weir flow), and
- submerged weir flow (i.e. subcritical weir flow).

The choice depends on the dimensions of the structure and the flow conditions. Whether or not the gate is in the flow or above the flow yields either gate or weir flow. Furthermore, the flow can be either subcritical (submerged) or supercritical (free). Both for incoming flow ("Flow") and for outgoing flow ("Reverse") contraction coefficients can be specified. These coefficients can be seen as tuning parameters for the user.

14.3.1 Notation

The used variables in this subsection are described in this paragraph.

The five geometric parameters as shown in Figure 14.2:

z_s	Crest level,
z_{u1}	Level upstream 1,
z_{u2}	Level upstream 2,
z_{d1}	Level downstream 1, and
z_{d2}	Level downstream 2

Similarly for the five values for the width of the General structure, see Figure 14.3:

w_s	Width at crest level,
w_{u1}	Width upstream 1,
w_{u2}	Width upstream 2,
w_{d1}	Width downstream 1, and
w_{d2}	Width downstream 2

Other variables used in the equations of this subsection:

C	Chézy value, representing the roughness at the downstream side of the General structure
c_{gf}	Correction coefficient for free gate flow
c_{gd}	Correction coefficient for submerged gate flow
c_{wf}	Correction coefficient for free weir flow
c_{wd}	Correction coefficient for submerged weir flow
E_1	Energy height upstream ($= \zeta_{up} + u^2/(2g)$)
g	Acceleration due to gravity
H_{s1}	Energy height with respect to the crest level ($= E_1 - z_s = \zeta_{up} + \frac{u^2}{2g} - z_s$)
h_g	Opening height of the gate ($= \texttt{gateLowerEdgeLevel} - z_s$)
L	Length of the expansion zone at the downstream side of the General structure
h_2	Water depth downstream of the structure with respect to the crest level ($= \zeta_{down} - z_{d2}$), see remark below.
ΔS_1	$= z_s - z_{d1}$
ΔS_2	$= z_{d1} - z_{d2}$
λ	Extra resistance parameter
ρ^*	$= \rho_2/\rho_1$
ρ_1	Density of the water at the upstream side of the General structure [kg/m^3]
ρ_2	Density of the water at the downstream side of the General structure [kg/m^3]
μ_{gf}	Contraction coefficient for free gate flow

μ_{gd}	Contraction coefficient for submerged gate flow ($\mu_{gd} = \mu_{gf}$)

Remarks:

◇ Downstream water depth h_2 is limited to a minimal value of $0.9 \cdot \frac{2}{3} \cdot H_{s1} \cdot \left(\frac{w_s}{w_{d2}}\right)^{\frac{2}{3}}$.

◇ The effect of this limiter might result in an incorrect flow state. In case the downstream level z_{d2} is similar to the crest level z_s and the downstream width w_{d2} is similar to the crest width w_s, the general structure might not reach the free flow state.

◇ The energy heights E_1, H_{s1} can be set to ordinary water levels in case the flag `UseVelocityHeight` is set to `False`. For more information, see section 14.3.4.

14.3.2 Geometry

The crest level, gate lower edge level and gate door opening width can also be given as time series.

A (2D) hydraulic structure is given as a single geometrical object by the user, but it may cross multiple 2D grid cells (or: flow links). In case a General structure is positioned on a 1D branch, a structure crosses only 1 link. Each flow link under the influence of this structure (i.e. structure link) is treated as a General structure with the correct geometrical parameters.

Remark:

◇ In case a General structure crosses multiple 2d flow links, w_{u1} , w_{u2}, w_{d1} and w_{d2} are all set per individual link to the crest width per link ($w_s(L)$).

Crest width

The crest width of an individual flow link, under the influence of this structure, cannot exceed the width of this flow link. As a result, when the crest width exceeds the sum of the widths of the structure links, the crest width is reduced to the sum of the widths of the flow links. When the crest width is omitted in the input the structure width will be set to the sum of the widths of the structure links.

Remark:

◇ Keep in mind that in case a polyline has e.g. an angle of 45 degrees with the grid, the sum of the flow widths might be a factor of $\sqrt{2}$ larger than the polyline. Therefore it is recommended to align the grid with the structure.

In case the crest width is smaller than the sum of the widths of the structure links, the following algorithm is applied:

◇ Use the next definitions

$w_u(L)$	is the flow width of a structure link
N	is the total number of structure links in a structure
w_s	is the total crest width
$w_s(L)$	is the crest width of a structure link
w_{closed}	$= \sum_{L=1}^{N} w_u(L) - w_s$

- Determine smallest p and largest q where

$$\left(\sum_{L=1}^{p} w_u(L)\right) - 0.5 \cdot w_{closed} > 0 \tag{14.14}$$

$$\left(\sum_{L=1}^{q} w_u(L)\right) - 0.5 \cdot w_{closed} < 0 \tag{14.15}$$

- The crest widths of the structure links 1 until q will be set to 0 (i.e. closed)
- The crest width of structure link $p = q - 1$ will become:

$$w_s(p) = \left(\sum_{L=1}^{p} w_u(L)\right) - 0.5 \cdot w_{closed} \tag{14.16}$$

- A similar approach is applied to the right side of the structure.
- A a result the opening of the structure will be right in the middle of the set of structure links.

Example for the left part of the structure:
Assume the total crest width is 550 [m] (w_s), the total flow width is 1000 [m] and the number of structure links is set to 5 (N). So each structure link has a width of 200 is [m] ($w_u(L)$), the total flow width is indeed 1000 [m] ($\sum_{L=1}^{N} w_u(L)$) and thus w_{closed} is 450 [m]. According Equation (14.14) we get $p = 2$, and according Equation (14.15) we get $q = 1$. The partial open/closed structure link $w_s(p)$ has a crest width of 175 [m], according Equation (14.16) ($175 = 400 - 225$). The total crest width is: $w_s(2) + w_s(3) + w_s(4) = 175 + 200 + 175 = 550$ [m]. Taken into account the symmetry between left and right side of the structure.

Gate opening

The `GateOpeningWidth` is a compulsory parameter for General structures. A single gate door with no horizontal movement should use `GateOpeningWidth`=0. A structure that has one or two gate doors that do have horizontal opening and closing movement should specify the width of the horizontal opening beteen the doors.

Depending on the gate door positions, some structure links may have only flow across the weir crest in the opening, whereas others may have gate flow, or a combination of both. A similar algorithm for applying the gate door to the individual flow is performed as described for the crest width. There is an exception, because the door might not close symmetrical. The parameter `GateOpeningHorizontalDirection` defines the way a gate door is divided over the structure links. Possible values are `symmetric`, `fromLeft`, `fromRight`.

At both ends of the gate opening structure links may be partially opened in such a way that the `gateOpeningWidth` exactly matches the sum of the width of the open flow links.

Extra resistance

The extra resistance parameter λ relates to the friction force in the impulse balance, that in case of submerged gate flow or submerged weir flow is solved for the water movement at the downstream side of the General structure. The extra resistance parameter λ is given by the expression:

$$\lambda = \frac{gL}{C^2} \tag{14.17}$$

For an explanation of the variables see section 14.3.1.

Either the extra resistance parameter, λ, can be specified, or the length, L, can be specified by the user. In case length is used, the Chézy value of the flow link is used for calculating λ.

14.3.3 Computation of the downstream water level

In case of submerged gate flow or submerged weir flow the water level at the sill or downstream of the gate is required. This level is computed by application of the impulse balance.

Remark:

◇ The equations in this paragraph apply to flow in positive direction for flow in negative direction replace the next variables in the equations:

- ◇ z_{d1} into z_{u2}
- ◇ z_{d2} into z_{u1}
- ◇ w_{d1} into w_{u2}
- ◇ w_{d2} into w_{u1}

Submerged gate-flow

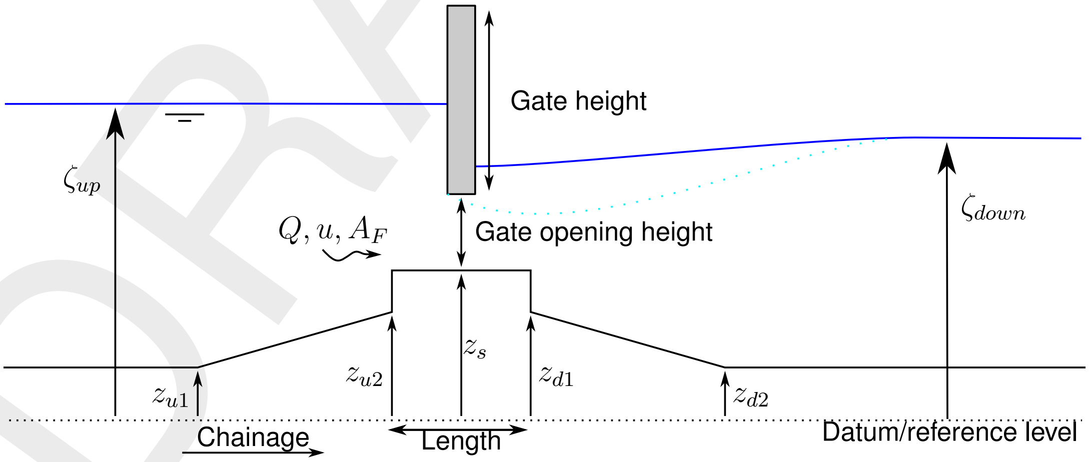


Figure 14.5: *Submerged gate flow*

The water depth on the crest h_s can be described by a second order algebraic equation:

$$A_g h_s^2 + B_g h_s + C_g = 0 \tag{14.18}$$

with:

$$A_g = (1+\rho^*)\left(\frac{w_{d1}}{3}+\frac{w_{d2}}{6}\right)+(1-\rho^*)\left(\frac{w_{d1}}{4}+\frac{w_{d2}}{12}\right) \tag{14.19}$$

$$\begin{aligned} B_g = {} & (1+\rho^*)\left((\Delta S_1+\Delta S_2-h_2)\left(\frac{w_{d1}}{3}+\frac{w_{d2}}{6}\right)+(2\Delta S_1+h_2)\frac{w_{d1}}{6}+\right. \\ & \qquad\qquad \left.(\Delta S_1+2h_2)\frac{w_{d2}}{6}\right)+ \\ & +(1-\rho^*)\left(\Delta S_1\frac{w_{d1}}{3}+(\Delta S_1+h_2)\frac{w_{d1}+w_{d2}}{6}\right)+ \\ & +\frac{4\rho^* c_{gd}^2\mu_{gd}^2 h_g^2 w_s^2}{w_{d2}h_2}(1+\frac{\lambda}{d_2})-4c_{gd}\mu_{gd}h_g w_s \end{aligned} \tag{14.20}$$

$$\begin{aligned} C_g = {} & (1+\rho^*)(\Delta S_1+\Delta S_2-h_2)\left(2\Delta S_1+h_2)\frac{w_{d1}}{6}+(\Delta S_1+2h_2)\frac{w_{d2}}{6}\right)+ \\ & +(1-\rho^*)\left((\Delta S_1)^2\frac{w_{d1}}{6}+(\Delta S_1+h_2)\frac{w_{d1}+w_{d2}}{12}\right)+ \\ & -\frac{4\rho^* c_{gd}^2\mu_{gd}^2 h_g^2 w_s^2 H_{s_1}}{w_{d2}h_2}(1+\frac{\lambda}{h_2})+4c_{gd}\mu_{gd}h_g w_s H_{s_1} \end{aligned} \tag{14.21}$$

For an explanation of the variables see section 14.3.1.

Equation (14.18) leads to:

$$h_s = \frac{-B_g+\sqrt{B_g^2-4A_gC_g}}{2A_g} \tag{14.22}$$

Submerged weir-flow

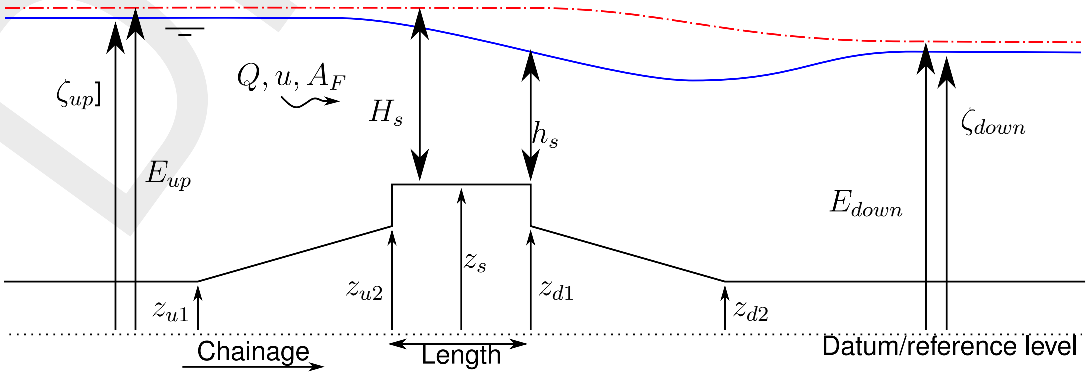


Figure 14.6: *Submerged weir-flow.*

The water depth at the sill h_s is described by a third order algebraic equation:

$$D_w h_s^3 + A_w h_s^2 + B_w h_s + C_w = 0 \tag{14.23}$$

with:

$$D_w = \frac{4\rho^* c_{wd}^2 w_s^2}{w_{d2} h_2}(1 + \frac{\lambda}{h_2}) \tag{14.24}$$

$$\begin{aligned} A_w = {} & (1+\rho^*)\left(\frac{w_{d1}}{3} + \frac{w_{d2}}{6}\right) + (1-\rho^*)\left(\frac{w_{d1}}{4} + \frac{w_{d2}}{12}\right) + \\ & - \frac{4\rho^* c_{wd}^2 w_s^2 H_{s_1}}{w_{d2} h_2}(1 + \frac{\lambda}{h_2}) - 4c_{wd} w_s \end{aligned} \tag{14.25}$$

$$\begin{aligned} B_w = {} & (1+\rho^*)\left((\Delta S_1 + \Delta S_2 - h_2)\left(\frac{w_{d1}}{3} + \frac{w_{d2}}{6}\right) + (2\Delta S_1 + h_2)\frac{w_{d1}}{6} + \right. \\ & \left. (\Delta S_1 + 2h_2)\frac{w_{d2}}{6}\right) + \\ & + (1-\rho^*)\left(\Delta S_1 \frac{w_{d1}}{3} + (\Delta S_1 + h_2)\frac{w_{d1} + w_{d2}}{6}\right) + 4c_{wd} w_s H_{s_1} \end{aligned} \tag{14.26}$$

$$\begin{aligned} C_w = {} & (1+\rho^*)(\Delta S_1 + \Delta S_2 - h_2)\left((2\Delta S_1 + h_2)\frac{w_{d1}}{6} + (\Delta S_1 + 2h_2)\frac{w_{d2}}{6}\right) + \\ & + (1-\rho^*)\left((\Delta S_1)^2 \ \frac{w_{d1}}{6} + (\Delta S_1 + h_2)^2 \ \frac{w_{d1} + w_{d2}}{12}\right) \end{aligned} \tag{14.27}$$

A direct method is applied to calculate h_s from equation Equation (14.23).

14.3.4 Turning off the computation of the energy levels

The input parameter `useVelocityHeight` in the <structures.ini> file (section C.12) can be used to change the default way of solving the flow through the structure. For certain situations the default setting might lead to incorrect results. The reason is that the general structure was developed for modelling structures in an open channel (river or rural applications), and the preceding approximations are based on certain assumptions. When applied to urban applications, some of these assumptions may not hold. Also important to realize is that the weir and orifice are based on the same formulations and assumptions.

The default setting, `useVelocityHeight=true`, is the situation as described in the previous section on water level computations. The upstream energy level is defined as

$$E_1 = \zeta_{up} + u^2/(2g) \tag{14.28}$$

and the downstream water depth (h_s) is determined as described in section 14.3.3.

The alternative setting, `useVelocityHeight=false`, simplifies the governing equations for E_1 and h_s. This can be of use when modelling a sewer system, where an overflow is also modelled as a general structure. Simple (gated) weir flow depending on simple water levels is preferred in such a case. The first difference compared to open water channels is that upstream horizontal velocities in an urban manhole compartment will typically be almost zero. The upstream energy height then simplifies to the upstream water level:

$$E_1 = \zeta_{up}. \tag{14.29}$$

The second difference is that the general structure's geometry simplifies to a long crest with $z_{u1} = z_{u2} = z_s = z_{d1} = z_{d2}$. The downstream water depth can no longer be derived from the accelaration across this (flat) crest, so it is simply based on downstream water level:

$$h_s = \zeta_{down} - z_s. \tag{14.30}$$

14.3.5 Discharge equations

The following discharge equations are applied during the computations.

- ◇ Free gate flow:

$$u_s = \mu_{gf} c_{gf} \sqrt{2g(E_1 - (z_s + \mu_{gf} h_g))} \quad (14.31)$$
$$A_f = w_s h_g \quad (14.32)$$
$$Q = u_s A_f = \mu_{gf} c_{gf} w_s h_g \sqrt{2g(E_1 - (z_s + \mu_{gf} h_g))} \quad (14.33)$$

- ◇ submerged gate flow:

$$u_s = \mu_{gd} c_{gd} \sqrt{2g(E_1 - (z_s + h_s))} \quad (14.34)$$
$$A_f = w_s h_g \quad (14.35)$$
$$Q = u_s A_f = \mu_{gd} c_{gd} w_s h_g \sqrt{2g(E_1 - (z_s + h_s))} \quad (14.36)$$
$$(14.37)$$

- ◇ Free weir flow:

$$u_s = c_{wf} \sqrt{\frac{2}{3} g(E_1 - z_s)} \quad (14.38)$$
$$A_f = w_s \ \frac{2}{3}(E_1 - z_s) \quad (14.39)$$
$$Q = u_s A_f = c_{wf} w_s \ \frac{2}{3} \sqrt{\frac{2}{3} g(E_1 - z_s)^{3/2}} \quad (14.40)$$

- ◇ submerged weir flow:

$$u_s = c_{wd} \sqrt{2g(E_1 - (z_s + h_s))} \quad (14.41)$$
$$A_f = w_s h_s \quad (14.42)$$
$$Q = u_s A_f = c_{wd} w_s h_s \sqrt{2g(E_1 - (z_s + h_s))} \quad (14.43)$$

For an explanation of the variables see section 14.3.1.

Remarks:

- ◇ The contraction coefficient has a maximum value $\mu = 1.0$. In case the user specifies a higher value, a warning will be generated.
- ◇ For all the coefficients two values must be specified. One for flow in positive direction and another for flow in negative direction.

Determining the flow type

The flow type is determined by this algorithm:

- ◇ Determine h_s based on submerged weir flow equations
- ◇ Use critical weir flow depth $h_c = \frac{2}{3}(E_1 - z_s)$
- ◇ Test weir flow
    - □ If ($h_s > h_c$ and $h_g > h_s$) then submerged weir flow
    - □ Else if ($h_s < h_c$ and $h_g > h_c$) then free weir flow
    - □ Else gate flow:

- ○ The water depth at the sill h_s is recalculated using the submerged gate flow equations. The critical depth h_c is now defined as $\mu_{gf} h_g$.
  - ▷ If ($h_s < h_c$ or if h_s cannot be solved) then free gate flow
  - ▷ Else submerged gate flow

Remark:

- ◇ In case upstream water level is above gate lower edge level, there can still be **submerged weir flow** if $\zeta_s > h_c$ and $h_g > \zeta_s$ or **free weir flow** if $\zeta_s < h_c$ and $h_g > \zeta_s$.

14.4 Weir

The weir structure is a construction generating energy losses due to constriction of the flow. The dimensions of a weir are described by a crest level and a crest width. Weirs are used to regulate the flow in a waterway. A weir is treated as a general structure whose conceptual description can be found in section 14.3. The input file format for a weir is described in section C.12.1.

Table 14.4 lists the mapping of the weir input parameters to the general structure parameters as well as default values for the remaining general structure parameters.

***Table 14.4:** Mapping table for weir input parameters to general structure parameters*

General structure parameters	Input variable/value for Weir
z_s	`crestLevel`
z_{u1}, z_{u2}	Upstream bedlevel of the channel
z_{d1}, z_{d2}	Downstream bedlevel of the channel
w_s, w_{u1}, w_{u2}, w_{d1}, w_{d2}	`crestWidth`
μ_{gf}, μ_{gd}	1.0
c_{gf}, c_{gd}, c_{wf}, c_{wd}	`corrCoeff`
`gateLowerEdgeLevel`	10^{10}
`gateHeight`	N/A
`gateOpeningWidth`	N/A

During the simulation a number of checks and possibly corrections on some dimensions of the weir are performed.

- ◇ If the given `crestLevel` is below the bedlevel of the surrounding waterlevel points in the channel, the `crestLevel` (z_s) is raised to the highest bedlevel of these two points (also see footnote 1).
- ◇ If the given `crestWidth` is larger than the corresponding maximum total flow width of the underlying flow link(s) then the `crestWidth` is set to that maximum flow width.

14.5 Orifice

The orifice structure can be used to simulate rectangle-shaped gates through which water within a channel or river is led. Orifices are used to regulate the water flow through or towards a channel or river. An orifice is treated as a general structure whose conceptual description can be found in section 14.3. The input file format for an orifice is described in section C.12.8.

Table 14.5 lists the mapping of the orifice input parameters to the general structure parameters as well as default values for the remaining general structure parameters.

***Table 14.5:** Mapping table for orifice input parameters to general structure parameters*

General structure parameters	Input variable/value for Orifice
z_s	`crestLevel`
z_{u1}, z_{u2}	Upstream bedlevel of the channel
z_{d1}, z_{d2}	Downstream bedlevel of the channel
w_s, w_{u1}, w_{u2}, w_{d1}, w_{d2}	`crestWidth`
μ_{gf}, μ_{gd}	1.0
c_{gf}, c_{gd}, c_{wf}, c_{wd}	`corrCoeff`
`gateLowerEdgeLevel`	`gateLowerEdgeLevel`
`gateHeight`	10^{10}
`gateOpeningWidth`	0.0

During the simulation a number of checks and possibly corrections on some dimensions of the orifice are performed.

◇ If the given `crestLevel` is below the bed level of the surrounding water level points in the channel, the `crestLevel` (z_s) is raised to the highest bed level of these two points (also see footnote 1).
◇ If the given `crestWidth` is larger than the corresponding maximum total flow width of the underlying flow link(s) then the `crestWidth` is set to that maximum flow width.
◇ If the given `gateLowerEdgeLevel` is below the `crestLevel`, the `gateLowerEdgeLevel` is set to that `crestLevel`.

Flow limiter

The total flow through an orifice can be limited using the input keywords `(use)LimitFlowPos` and `(use)LimitFlowNeg` (in positive and negative direction, respectively). In this case the discharge through this structure is limited to the maximum specified discharge. When an orifice is located on a 2D grid, the flow limiter is applied for each flow link individually. The maximum discharge through each individual orifice link is weighted by the individual sill width of a link ($w_s(L)$) divided by the total width at the sill (w_s):

$$Q_{\text{lim}}(L) = \min(Q(L), Q_{\text{lim}} \cdot w_s(L)/w_s), \tag{14.44}$$

where notation from section 14.3.2 is reused. As this limitation per link based on width is only an approximation, this can mean that $\sum_{L=1}^{N} Q_{\text{lim}}(L) \leq Q_{\text{lim}}$.

14.6 Culvert

The culvert is an underground structure that normally connects two open channels. The flow through a culvert is affected by its upstream and downstream invert levels (z_{c1} and z_{c2}, respectively), its length, the size and shape of its closed cross section, its inlet loss, its friction loss, its valve loss and its outlet loss. The input file format for a culvert is described in section C.12.3.

Figure 14.7 shows a side view of a culvert.

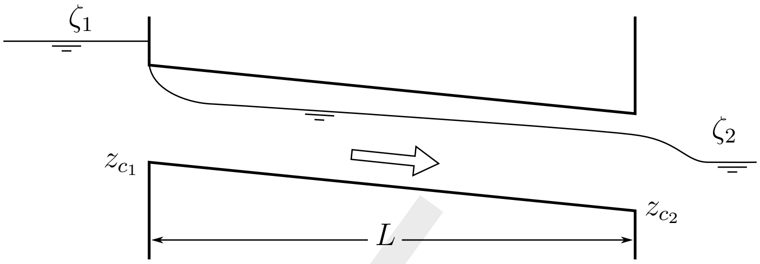


Figure 14.7: *Side view of a culvert*

Two flow conditions can occur:

Free flow when $\zeta_2 < z_{c2} + h_{c2}$

$$Q = \mu A_{fc} \sqrt{2g(\zeta_1 - (z_{c2} + h_{c2}))} \tag{14.45}$$

Submerged flow when $\zeta_2 \geq z_{c2} + h_{c2}$:

$$Q = \mu A_{fc} \sqrt{2g(\zeta_1 - \zeta_2)} \tag{14.46}$$

Where:

Q	Discharge through culvert [m^3/s]
μ	Discharge coefficient, derived from loss-coefficients [$-$]
A_{fc}	Discharge culvert flow area $\min(A_{fc1}, A_{fcvalve})$ [m^2] A_{fc1}: Flow area in the culvert at its upstream side [m^2] $A_{fcvalve}$: Flow area under the culvert valve [m^2]
g	Acceleration due to gravity [m/s^2] (≈ 9.81)
ζ_1	Upstream water level [m]
ζ_2	Downstream water level [m]
z_{c2}	Downstream culvert invert level [m]
h_{c2}	Critical culvert depth at the downstream side, $\sqrt[3]{Q^2/(gT_2^2)}$ [m]
T_2	Surface width in the culvert at its downstream side [m]

$$T_2 = \frac{A(h_{c2})}{h_{c2}} \tag{14.47}$$

$A(h_{c2})$	Flow area of the culvert at water depth (h_{c2}) [m^2]

Discharge coefficient (μ)

For numerical reasons the discharge coefficient (μ) is limited to a maximum of 1.0. The discharge coefficient (μ) is computed as follows:

$$\mu = \frac{1}{\sqrt{\xi_i + \xi_f + \xi_v + \xi_o}} \tag{14.48}$$

Where:

ξ_i	Inlet loss coefficient [$-$]
ξ_f	Friction loss coefficient [$-$]

ξ_v Valve loss coefficient $[-]$
ξ_o Outlet loss coefficient $[-]$

The inlet loss coefficient (ξ_i) can be defined as a constant value only.

The friction loss coefficient (ξ_f) is computed as follows:

$$\xi_f = \frac{2gL}{C_1^2 R} \tag{14.49}$$

Where:

L Length of the culvert $[m]$
C_1 Chézy coefficient in the culvert at its upstream side [m$^{1/2}$/s]
R Hydraulic radius $[m]$

$$R = \begin{cases} R_{c1} & \zeta_1 \geq z_{c1} + h_{valve} \\ R_{valve} & \zeta_1 < z_{c1} + h_{valve} \end{cases} \tag{14.50}$$

h_{valve} Valve opening height $[m]$
R_{c1} Hydraulic radius in the culvert at its upstream side $[m]$
R_{valve} Hydraulic radius based on actual valve opening height $[m]$
z_{c1} Upstream culvert invert level $[m]$

The valve loss coefficient (ξ_v) can be defined as a function of the ratio of the “Valve opening height” and the “maximum inner culvert height”.

Remarks:

◇ During the computations the actual valve loss coefficient (ξ_v) is derived from the user defined table, while using the ratio of the “actual valve opening height (h_{valve}) and the “actual maximum inner culvert height” ($h_{culvert}$):

$$\xi_v = f(h_{valve}/h_{culvert}) \tag{14.51}$$

Where f is a piecewise linear function with support points defined by the input fields `relOpening` and `lossCoeff`.

◇ A constant extrapolation is performed for values outside of the range of `relOpening`.

The outlet loss coefficient (ξ_o) is computed as follows:

◇ **Submerged flow** ($\zeta_2 = z_{c2} + h_{c2}$):

$$\xi_o = k\left(1 - \frac{A_{fc}}{A_{fr2}}\right)^2 \tag{14.52}$$

Where:

k User defined constant outlet loss coefficient $[-]$
A_{fr2} Flow area in the branch, adjacent to the downstream culvert side $[m^2]$
A_{fc} Culvert flow area $[m^2]$

$$A_{fc} = \begin{cases} A_{f_{cvalve}} & h_1 \geq z_{c1} + h_{gate} \\ A_{f_{c1}} & h1 < z_{c1} + h_{gate} \end{cases} \tag{14.53}$$

h_{valve} Valve opening height $[m]$

z_{c1} Upstream culvert invert level $[m]$
$A_{f_{cvalve}}$ Flow area under the culvert valve $[m^2]$
$A_{f_{c1}}$ Flow area in the culvert at its upstream side $[m^2]$

◇ **Free flow** ($\zeta_2 < z_{c2} + h_{c2}$):

$$\xi_o = 0 \tag{14.54}$$

Culvert cross sections, bed friction

For a culvert all available closed cross section types can be used. Culvert friction can be specified using any of the available bed friction formulations.

Culvert, Good modelling practice aspects

During the simulation a number of checks and possibly corrections on some dimensions of the culvert are performed.

◇ If the **upstream `invertLevel`** is below the bed level of the upstream water level point in the channel, the bed level of this water level point is lowered to the upstream `invertLevel`.
◇ If the **downstream `invertLevel`** is below the bed level of the downstream water level point in the channel, the bed level of this water level point is lowered to the downstream `invertLevel`.

Notice that the culvert checks are an exception to the other structure checks: the surrounding bed levels are adjusted, whereas for all other structure, the structure's own bed level(s) are adjusted.

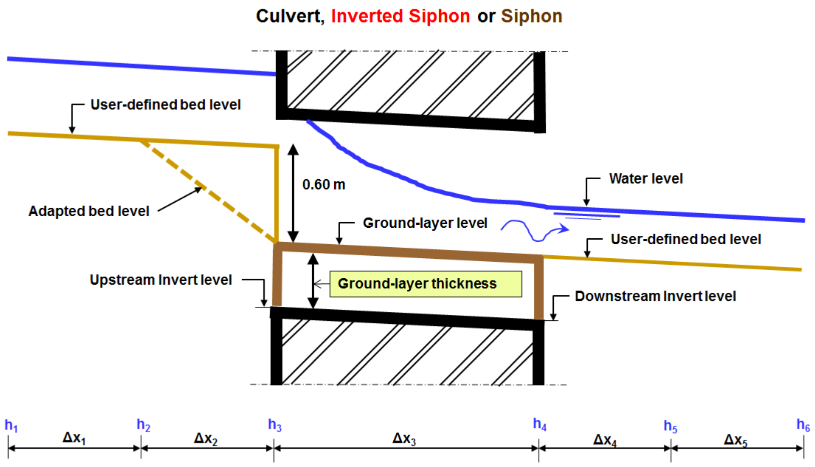


Figure 14.8: *Good modelling practice, culvert, Inverted Siphon and Siphon*

Inverted siphon

The inverted siphon is a special version of the culvert. An inverted siphon is a structure that normally connects two open channels, that are separated by a particular infrastructural work (e.g. dike, railroad). The inverted siphon makes an underground connection through such infrastructural work. An inverted siphon is assumed to be fully filled with water at its deepest point. In case this does not yield for your stucture, you can consider to use a culvert. The flow through an inverted siphon is affected by its upstream and downstream invert level, the size and shape of its closed cross section, its ground layer thickness, its entrance loss, friction loss, its valve loss, losses due to its bends (bend loss), and its exit loss.

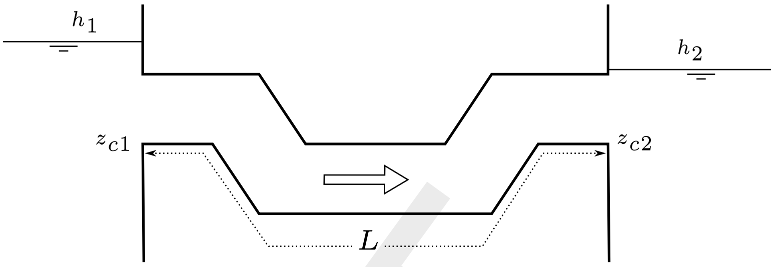


***Figure 14.9:** Side view of an inverted siphon*

The bend losses are taken into account by changing Equation (14.55) into:

$$\mu = \frac{1}{\sqrt{\xi_i + \xi_f + \xi_v + \xi_o + \xi_b}} \tag{14.55}$$

Where:

ξ_b bend losses coefficient $[-]$

14.7 Long Culvert

14.7.1 Long culvert

Remark:

◇ Long culvert functionality has beta status, and is still undergoing validation.

The long culvert is an underground pipe structure that normally connects two open waters modeled as 2D grid cells. The difference with the normal culvert is that the long culvert is modelled as an actual 1D pipe in the 1D network, instead of being modelled merely as a subgrid culvert, with only energy losses at the snapped flow link. The long culvert stores actual volume, and can optionally have multiple pressure points (which allows for a 'siphon-like' geometry with varying bed levels).

The flow through a long culvert is affected by its upstream and downstream invert levels (first and last z-coordinate, respectively), its total length, the size of its closed rectangular cross section, its friction loss, and its valve loss. The input file format for a long culvert is described in Section C.12.4.

Long culvert positioning

Figure 14.10 shows a side view of a long culvert. In the most simple case there will only be two points z_1 and z_2.

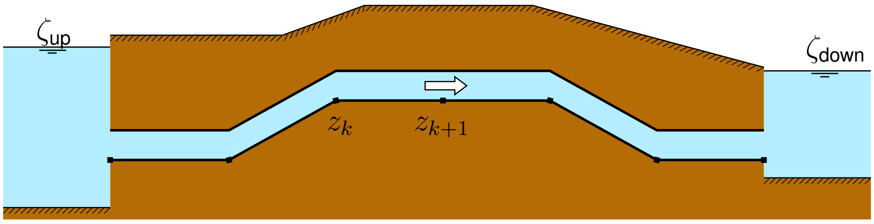


***Figure 14.10:** Side view of a culvert*

Figure 14.11 shows the discretization of long culverts in a top view. When the long culvert input polyline has more than two points, internal 1D computational pressure points are automatically generated and coupled via 1D2D links to the inlet and outlet 2D grid cells.

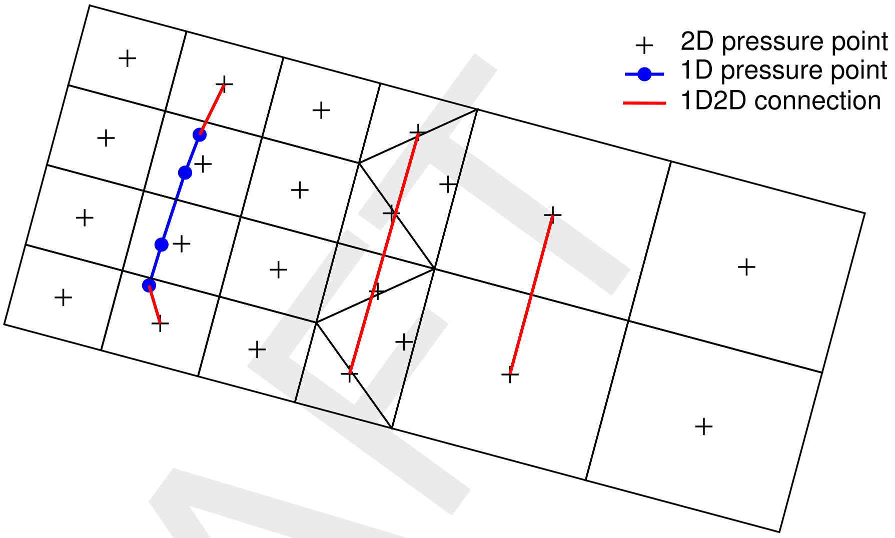


Figure 14.11: Discretization of three long culverts via 1D and 1D2D links.

The 1D2D links are of the vertically stacked type (see Section 8.9.5), such that pipe flow can go underneath the 2D grid cells. As a result, the flow link bed levels at the inlet and outlet side are fully determined by the long culvert's z-values (cf. Equation (8.86)).

Long culvert geometry

The 1D representation of long culverts results in regular 1D pipe flow, with a rectangular cross section with the user-defined dimensions (cf. Section 8.7).

Long culvert losses

The friction losses for long culverts are identical to the friction losses in regular 1D pipe flow (cf. Equation (8.64)). The valve losses are approximated by the user-defined relative valve opening, which is a dimensionless number between 0 and 1. The real discharge through the long culvert is reduced by a linear scaling of the maximum pipe cross sectional area:

$$A_{u_j,\text{culv}} := \theta_{\text{area,culv}} A_{u_j,\text{culv}}, \tag{14.56}$$

where j, culv is the first flow link at the inlet side of the long culvert and $\theta_{\text{area,culv}}$ is the relative flow area at the valve.

Long culverts part of input grid

In order to also support long culverts in parallel models with multiple partitions, a new implementation of the long culverts is available. The major difference is that the long culvert is no longer constructed based on an input polyline at runtime, but instead read from the netfile. So, the 1D grid points and 1D2D links must be present in the input grid (<_net.nc> file) already, via a preprocessing step. Because all long culvert flow node and link information is now contained in the net file, the partitioner is now able to partition the long culverts as well, enabling their use in parallel simulations.

Long culvert conversion

If it is desired to run a model containing long culverts in parallel (partitioned, using MPI), only 1D2D long culverts are supported. In order to facilitate this, a command line option is available to convert a model containing old-style long culverts to the 1D2D implementation.
This is done with the `--convertlongculverts` command line option, as follows:

```
dflowfm-cli.exe --convertlongculverts <prefix> <mdufile>
```

The `prefix` is a user defined input string that will be prepended to the new model input files generated by the conversion process. The `mdufile` is an MDU file pointing to a net file that does not yet contain the 1D2D culverts links, and also pointing to a structure file, where the long culverts are defined via polyline coordinates. The conversion process reads the polyline that defines the original long culvert, and creates a network branch representing it. The original long culvert implementation allows for short long culverts that consist of a single 2D2D link, see figure 14.11. As 2D2D links are not currently supported in the 1D2D kernel, instead of snapping the endpoints of the polyline to the neighboring 2D-flow nodes, the new implementation instead transforms *all* polyline points to new 1D grid points, and additionally creates two 1D2D links connecting the 1D culvert start and end point to the 2D grid cells containing these points. The branch ID is then written to the new <structures.ini> file and a cross definition file is generated to contain the rest of the culvert parameters. The process produces a new network file, mdu file, structures file and cross definition file.

14.8 Bridge

The bridge structure is a structure that can be used to model the flow underneath a bridge (and across). The bridge's geometry is defined by either or both of:

- ◇ An abutment definition (as a cross section definition), which determines its wetted flow area and hydraulic radius. The flow area of the bridge should be smaller than the surrounding flow area of the channel.
- ◇ A pillars definition.

A bridge definition has to consist of at least one geometry definition, so at least an abutment definition or pillars definition has to be given in the input.

Figure 14.12 shows an example bridge where flow area is restricted by abutments, the bridge deck and two bridge pillars.

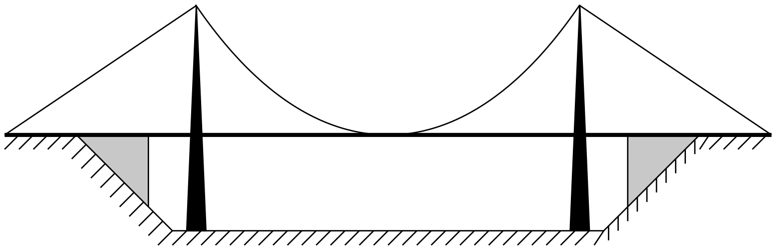


Figure 14.12: *A suspension bridge*

A limitation of the bridge structure is, that no free flow over the bridge can be computed. If free flow occurs over the bridge, it is advised to use a weir instead.

The general description of the discharge through a bridge is given by:

$$Q = \mu A_f \sqrt{2g(\zeta_1 - \zeta_2)} \tag{14.57}$$

Where

Q	Discharge through bridge [m^3/s]
μ	Discharge coefficient derived from loss-coefficients [-] (see Equation (14.58))
A_f	Wetted area [m^2] of flow through bridge at upstream side
g	Acceleration due to gravity [m/s^2] (≈ 9.81)
ζ_1	Upstream water level [m]
ζ_2	Downstream water level [m]

For the bridge the discharge coefficient is defined as:

$$\mu = \max\left(1, \frac{1}{\sqrt{\xi_i + \xi_f + \xi_o + \xi_y}}\right) \tag{14.58}$$

Where

ξ_i	Entrance loss coefficient.
ξ_f	Friction loss coefficient.
ξ_o	Exit loss coefficient.
ξ_y	Pillar loss coefficient.

Remark:

◇ For numerical reasons (e.g. validity of structure Equation (14.57)) the discharge coefficient (μ) is limited to a maximum of 1.0. In other words, if μ, according to Equation (14.58) becomes larger than 1.0, the actual applied discharge coefficient in Equation (14.57) is capped at 1.0. This means that for a given discharge, the water level difference over such bridge might be larger than anticipated, when considering the defined friction loss coefficients.

Abutment bridge

By defining an abutment bridge, the coefficients ξ_i, ξ_f and ξ_o in Equation (14.58) get a non-zero value. Also the flow area through the bridge is affected by the additional cross section definition.

When desired, the bridge deck can affect the flow through the bridge, for example by having the cross section closed. A limitation however, is that flow over the bridge deck cannot be taken into account.

The ξ_i loss coefficient is an input parameter

The ξ_o coefficient is defined as:

$$\xi_o = k\left(1 - \frac{A_f}{A_{f_2}}\right)^2, \tag{14.59}$$

where

k	Constant exit loss coefficient
A_f	Wetted area [m^2] of flow through bridge at upstream side, including the bridge pillars.

A_{f_2} Wetted area [m^2] of flow in branch at downstream side of bridge

The ξ_f coefficient is defined as:

$$\xi_f = \frac{2gL}{C^2 R}, \tag{14.60}$$

where

g Acceleration due to gravity [m/s^2] (≈ 9.81)
L Length of bridge [m]
C Chézy coefficient [m$^{1/2}$/s]
R Hydraulic radius [m]

During the simulation a check and possibly correction on the vertical position of the bridge can be performed, see MDU option `ChangeStructureDimensions`:

- If the bed level of the bridge is below the bed level of the surrounding water level points in the channel, the bedlevel of the bridge is raised to the highest bed level of the two water level points (also see footnote 1).
- In case the bed level is raised, the complete cross section is also raised.

Pillars

A bridge can have one or more pillars that affect the discharge through the bridge. By defining pillars, the coefficient ξ_y in Equation (14.58) get a non-zero value:

$$\xi_y = \beta \frac{\alpha_y}{A_f}, \tag{14.61}$$

where

β Parameter [$-$] depending on shape of pillar [s] (shape factor). Normally between 0.22 and 1.56
A_f Wetted area [m^2] of flow through bridge at upstream side
α_y Area [m^2] of wetted part of pillar [s] perpendicular to the flow direction, considered at upstream side

14.9 Universal weir

The universal weir structure is a construction generating energy losses due to constriction of the flow. The geometrical shape of the universal weir's crest can be defined as a Y-Z profile (see Figure 14.13).
It is assumed that the discharge across a universal weir is the summation of the discharges across each individual weir section. A weir section is the space between to subsequent support point of the Y-Z profile.

The crest level profile of a universal weir is divided into:

- *rectangular weir sections*, having a horizontal bed and
- *triangular weir sections*, having a sloping bed.

Rectangular weir sections are considered as a broad-crested weir. Triangular weir sections are considered as (the half of) a broad-crested weir with truncated triangular control section (Bos, 1989).

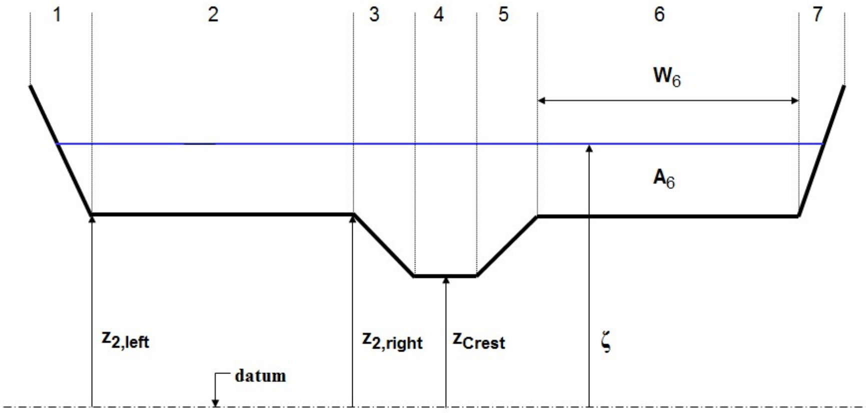


Figure 14.13: *Crest level (Y-Z) profile of a Universal weir, divided into three rectangular weir sections (2, 4 and 6) and four triangular weir sections (1, 3, 5 and 7)*

Parameters depicted in Figure 14.13 are:

A_i	Flow area of section i [m^2]
W_i	Width of weir section i [m]
$z_{i,left}$	Elevation at the left side of weir section i [mAD]
$z_{i,right}$	Elevation at the right side of weir section i [mAD]
z_{Crest}	Crest level of the Universal weir (output parameter only), equal to the lowest elevation of its crest level profile [mAD]
ζ	Water level [mAD]

Following equations yield for a universal weir:

$$Q = \sum_i^N (u_i A_i) = \sum_i^N Q_i \tag{14.62}$$

$$A = \sum_i^N A_i \tag{14.63}$$

$$U_{structure} = Q/A \tag{14.64}$$

where:

A	Flow area of the universal weir [m^2]
A_i	Flow area of weir section i [m^2]
N	Number of weir sections i [$-$]
Q	Discharge over the universal weir [m^3/s]
Q_i	Discharge over weir section i [m^3/s]
u_i	Flow velocity in weir section i [m/s]
$U_{structure}$	Average flow velocity over the universal weir [m/s]

Following equations yield for a rectangular weir section i:

$$z_{i,crest} = min(z_{i,left},\ z_{i,right}) \tag{14.65}$$

$$ml_{rectangular,i} = 2/3 \tag{14.66}$$

- ◇ Free rectangular weir flow if $\left(\frac{\zeta_2 - z_{i,crest}}{\zeta_1 - z_{i,crest}}\right) \leq ml_{rectangular,i}$

$$u_i = c_e\sqrt{\frac{2}{3}g}\sqrt{\zeta_1 - z_{i,crest}} \tag{14.67}$$

$$A_i = \frac{2}{3}W_i(\zeta_1 - z_{i,crest}) \tag{14.68}$$

$$Q_i = u_i\, A_i = c_e\, \frac{2}{3}\, \sqrt{\frac{2}{3}g} \quad W_i\, (\zeta_1 - z_{i,crest})^{3/2} \tag{14.69}$$

- ◇ Submerged rectangular weir flow if $\left(\frac{\zeta_2 - z_{i,crest}}{\zeta_1 - z_{i,crest}}\right) > ml_{rectangular,i}$

$$u_i = c_e\sqrt{2g(\zeta_1 - \zeta_2)} \tag{14.70}$$

$$A_i = W_i(\zeta_2 - z_{i,crest}) \tag{14.71}$$

$$Q_i = u_i\, A_i = c_e\, W_i\, (\zeta_2 - z_{i,crest})\, \sqrt{2g(\zeta_1 - \zeta_2)} \tag{14.72}$$

Following equations yield for a triangular weir section i:

$$z_{i,crest} = min(z_{i,left},\ z_{i,right}) \tag{14.73}$$

$$dz_i = |z_{i,left} - z_{i,right}| \tag{14.74}$$

$$ml_{triangular,i} = \begin{cases} \frac{4}{5} & \text{if } (\zeta_1 - z_{i,crest})) \leq 1.25\, dz_i \\ \frac{2}{3} + \frac{1}{6}\frac{dz_i}{(\zeta_1 - z_{i,crest})} & \text{if } (\zeta_1 - z_{i,crest})) > 1.25\, dz_i \end{cases} \tag{14.75}$$

- ◇ Free triangular weir flow if $\left(\frac{\zeta_2 - z_{i,crest}}{\zeta_1 - z_{i,crest}}\right) \leq ml_{triangular,i}$

$$u_i = c_e\sqrt{2g(1 - ml_{triangular,i})(\zeta_1 - z_{i,crest})} \tag{14.76}$$

$$A_i = \begin{cases} W_i\left(\frac{(ml_{triangular,i}\ (\zeta_1 - z_{i,crest}))^2}{2\ dz_i}\right) & \text{if } \frac{\zeta_1 - z_{i,crest}}{1/ml_{triangular,i}} \leq dz_i \\ W_i\left(\frac{\zeta_1 - z_{i,crest}}{1/ml_{triangular,i}} - \frac{dz_i}{2}\right) & \text{if } \frac{\zeta_1 - z_{i,crest}}{1/ml_{triangular,i}} > dz_i \end{cases} \tag{14.77}$$

$$Q_i = u_i\, A_i \tag{14.78}$$

- ◇ Submerged triangular weir flow if $\left(\frac{\zeta_2 - z_{i,crest}}{\zeta_1 - z_{i,crest}}\right) > ml_{triangular,i}$

$$u_i = c_e\sqrt{2g(\zeta_1 - \zeta_2)} \tag{14.79}$$

$$A_i = \begin{cases} W_i\left(\frac{(\zeta_2 - z_{i,crest})^2}{2\ dz_i}\right) & \text{if } \quad \zeta_2 - z_{i,crest} \leq dz_i \\ W_i\ (\zeta_2 - z_{i,crest} - \frac{dz_i}{2}) & \text{if } \quad \zeta_2 - z_{i,crest} > dz_i \end{cases} \tag{14.80}$$

$$Q_i = u_i\, A_i \tag{14.81}$$

where:

A_i Flow area of weir section i [m^2]
c_e Discharge coefficient, applicable to all rectangular weir sections as well as to all triangular weir sections of a universal weir [$-$]
dz_i Vertical distance between the elevation at the left side and the right side of a triangular weir section i [m]
g Acceleration due to gravity [m/s^2] (≈ 9.81)
$ml_{rectangular,i}$ *Water-level-based* modular limit of a rectangular weir section i. Its value is always equal to 2/3 [$-$]
$ml_{triangular,i}$ *Water-level-based* modular limit of a triangular weir section i. Its value depends on the actual water depth. [$-$]
Q_i Discharge over weir section i [m^3/s]
u_i Velocity in weir section i [m/s]
W_i Width of weir section i [m]
$z_{i,crest}$ Crest level of weir section i [mAD]
$z_{i,left}$ Elevation at the left side of weir section i [mAD]
$z_{i,right}$ Elevation at the right side of weir section i [mAD]
ζ_1 Water level at the upstream side of the universal weir [mAD]
ζ_2 Water level at the downstream side of the universal weir [mAD]

During the simulation a check and possibly correction on the crest level of the universal weir is performed.

- If the bed level of the bridge is below the bed level of the surrounding water level points in the channel, the bed level of the bridge is raised to the highest bed level of the two water level points (also see footnote 1).
- In case the bed level is raised, the complete weir profile is also raised.

14.10 Pump

The pump structure forces the flow of water in one direction at a certain discharge. This discharge is essentially a predescribed value for the pump, depending on the type of pump. A pump can either be a staged pump or a non staged pump. The input file format for a pump is described in section C.12.7.

A **staged pump** contains control rules to switch on or off the pump and to set the actual capacity depending on the water levels at both sides of the pump.

A **non-staged pump** has its capacity prescribed directly. The capacity can be defined as a constant value, a time series or controlled by RTC.

Both types of pumps can optionally have a **reduction factor** table defined. When a pump pumps against a bigger head difference, its effective discharge may be lower than the maximum capacity.

14.10.1 Notation

The used variable in this section are described below.

ζ_{ss}^n Water level at the suction side of the pump at the current time level.
ζ_{ds}^n Water level at the delivery side of the pump at the current time level.
h_{ss}^n Water depth at the suction side of the pump at the current time level.
V_{ss}^n Water volume at the suction side of the pump at the current time level.

14.10.2 Orientation

Water is always pumped from the suction side to the delivery side. The orientation of the pump in the model is defined by either the branch orientation (1D) or the direction of the polyline (2D, see section C.12). It may be convenient to orientate the pump opposite to the branch orientation of the channel. This is controlled by the input parameter `orientation`. `positive` means the suction side is at the (geometrical) upstream side of the pump and `negative` means that the suction side is at the (geometrical) downstream side of the pump.

14.10.3 Staged pump

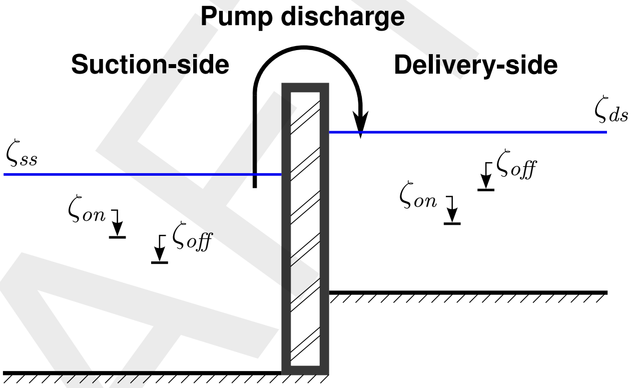


Figure 14.14: Pump station

Pump stages

A pump station may consist of multiple pump stages. Each stage can have a different pump capacity and switch on and switch off level. At each point in time, the pump can be in one pump stage only. The highest stage that is turned on, will be the actual stage, regardless of the state of the lower stages.

The pump can be controlled at either the **suction side** only, the **delivery side** only or on **both sides**.

The state of pump stage$_i$ is determined by:

- **suction side:**
  - **if** no suction side control **then** $s_{ss,i}^{n+1} = True$
  - **else if** $\zeta_{ss}^n > \zeta_{ss,on,i}$ **then** $s_{ss,i}^{n+1} = True$
  - **else if** $\zeta_{ss}^n < \zeta_{ss,off,i}$ **then** $s_{ss,i}^{n+1} = False$
  - **else** $s_{ss,i}^{n+1} = s_{ss,i}^n$
- **delivery side:**
  - **if** no delivery side control **then** $s_{ds,i}^{n+1} = True$
  - **if** $\zeta_{ds}^n < \zeta_{ds,on,i}$ **then** $s_{ds,i}^{n+1} = True$
  - **else if** $\zeta_{ds}^n > \zeta_{ds,off,i}$ **then** $s_{ds,i}^{n+1} = False$
  - **else** $s_{ds,i}^{n+1} = s_{ds,i}^n$
- **if** $s_{ss,i}^{n+1} == True$ **and** $s_{ds,i}^{n+1} == True$ **then** pump stage$_i$ is on

Where:

$s^{n+1}_{ss,i}$ Denotes whether the suction side control for the next time step is on ($True$) or off ($False$).

$s^{n+1}_{ds,i}$ Denotes whether the delivery side control for the next time step is on ($True$) or off ($False$).

$\zeta_{ss,on,i}$ Start level at suction side for stage i.

$\zeta_{ss,off,i}$ Stop level at suction side for stage i.

$\zeta_{ds,on,i}$ Start level at delivery side for stage i.

$\zeta_{ds,off,i}$ Stop level at delivery side for stage i.

Subsequently, the actual pump stage will be set to the stage with the highest sequence number that is turned on.

14.10.4 Non-staged pump

A non staged pump has either a constant capacity, a time series for the capacity or can be controlled by an external process, such as real time control.

14.10.5 Capacity reduction table

A capacity reduction table can be defined both for the staged and non-staged pump. The capacity reduction table contains capacity reduction factors as a function of the pump head (i.e., $\zeta_{ds} - \zeta_{ss}$). The pump heads (key 'head' in input) in a capacity reduction table should be in increasing order. Capacity reduction factors (key reductionFactor in input) should be equal or larger than 0 and equal or less than 1. The actual pump discharge equals the current pump capacity multiplied by the capacity reduction factor. When no capacity reduction table is defined, in effect a capacity reduction factor equal to 1 is applied.

Please note that the pump head at $t = t^n$ is used to determine the capacity reduction factor to be applied in the time-step from $t = t^n$ to $t = t^{n+1}$.

14.10.6 Distribution and relaxation of pump discharge

A single pump in a 2D model may span multiple flow links, in which case the pump discharge is distributed evenly over the included flow links. Flow links are included when the water depth at the suction side is above a certain threshold: $h^n_{ss} > 0.01$. A single, representative water level at the suction side is then determined by averaging the separate water levels at the suction side of the pump over the included flow links. The delivery side water level is determined similarly. These two values are used for staged pumps or, optionally, for a capacity reduction table.

The actual pump discharge may be limited (relaxed) even further when total potential pump volume is above the total available volume of water at the suction side. In other words, it can happen that the actual discharge through the pump is limited below capacity by water volume on suction side because the suction side has not enough volume of water to be sucked.

The actual discharge is computed by:

$$q^n_{\text{pump}} = \min(q_{\text{pump}}, 0.9 \cdot \sum_{\text{open pump links}} V^n_{ss}/\Delta t^n). \tag{14.82}$$

The pump capacity is then equally distributed over the open pump links.

14.11 Compound structure

A compound structure consists of several structures (structure elements) parallel to each other at the same location.
See Table C.7 for a list of possible structure types. The only exception is that compound structures cannot be nested.

Each structure element in a compound is treated independently. The discharge through each structure element is then summed up to the total discharge for the compound structure.

The user should keep in mind that geometric checks on structure parameters are performed for each structure element independently. For example the total structure width might be larger than the actual width of the channel, even though the widths of the underlying structures have been checked. As a result the user must be extra careful to check for inconsistencies in the input.

14.12 Outlet

Urban sewer systems contain outlets (or outfalls) to discharge excess water into open water, waste water treatment plants, or other ‘external’ downstream systems. Often, an outlet is combined with some sewer overflow structure that has a certain crest level. As long as the sewer water level does not reach this crest, no outflow occurs. In reality outlets are constructions part of the sewer system, but in the model schematization these outlets simply act as boundary conditions. In other words, there is *no* separate hydraulic structure type for the outlet.

As an example, the model schematization near an outlet can look as follows: the outlet node is a regular end node in the network (details in section B.2). A waterlevel boundary, associated via the `nodeId`, is set to the outside water level (details in section C.5.2.1). Next in the network, connected to the boundary node, can be a regular internal node. On the connection between these two nodes, the overflow structure can be placed, for example a weir (section 14.4) or a pump (section 14.10).

When the actual overflow structure is such that it only allows outflow and no inflow, the model schematization should use a hydraulic structure that has an `allowedFlowDir` setting, for example the universal weir (section 14.9).

14.13 Thin dams

Thin dams are similar to fixed weirs. The only difference between the thin dams and fixed weirs are in their crest levels. Thin dams, in principle, include infinitely high crest levels and hence, they do not allow water flux. Similar to the other structures, the thin dams can be selected from the toolbar and drawn by a polygon. D-Flow FM adjusts the polygon to the nearest velocity points. The input data for a thin dam is identical to those for fixed-weir, except for the crest level.

15 Bedforms and vegetation

The terrain and vegetation exert shear stresses on the passing flow. The magnitude of the shear stress of the bed is often characterised by means of roughness coefficient of type Chézy, Manning or White-Colebrook. Within the main stream flow the shear stresses are largely determined by the local conditions of the alluvial bed (bed composition and bedform characteristics). In other areas, such the floodplains of rivers and in the intertidal areas of estuaries, the flow resistance is determined by a combination of vegetation and an alluvial bedforms or even a non-alluvial bed. To accurately represent such conditions in the numerical model, D-Flow FM has been extended with a vegetation model. Another related feature known from Delft3D-FLOW is the bedform roughness predictors; these are not available in D-Flow FM yet. These types of flow resistance may be resolved in a 2D numerical model using the trachytope approach (see section 15.2).

15.1 Bedform heights

The dune height and Van Rijn (2007) bedform roughness predictors, known from Delft3D-FLOW, are not available yet in D-Flow FM. They will be in an upcoming release.

15.2 Trachytopes

This functionality allows you to specify the bed roughness and flow resistance on a sub-grid level by defining and using various land use or roughness/resistance classes, further referred to as trachytopes after the Greek word $\tau\rho\alpha\chi\acute{\upsilon}\tau\eta\varsigma$ for roughness. The input parameters and files to use the trachytopes functionality are described in section C.7.

At every time step (or less frequent as requested by the user) the trachytopes are converted into a representative bed roughness C, k or n and optional linear flow resistance coefficient λ per velocity point with index j.

$$M = -\frac{1}{2}\lambda_j u_j \left|\boldsymbol{u}_j\right| \tag{15.1}$$

To save computational time the user may choose to update the computed bed roughness and resistance coefficients less frequently than every time step. See section C.7 for a description of the keywords and input files associated with this feature.

The following two sections describe the various classes of trachytopes distinguished and the way in which they are combined, respectively.

15.2.1 Trachytope classes

Three base classes of trachytopes are distinguished: area classes, line classes and point classes. The area classes (type range 51–200) basically cover the whole area, therefore, they are generally the dominant roughness factor. The line classes (type range 201–250) may be used to represent hedges and similar flow resistance elements; it will add anisotropy to the roughness field. The point class (type range 251–300) represents a set of point flow resistance elements. The next six sections provide an overview of the various trachytope formulae implemented.

Special classes (1–50)

In addition to the three base classes two special trachytope classes have been defined: a flood protected area and a composite trachytope class. The first class represents a sub-grid area that is protected from flooding and thus does not contribute to the bed roughness; however, the effect on the flow resistance should be taken into account. The second class can be used to make derived trachytope classes that are a combination of two other trachytopes: an area fraction α of trachytope type T_1 and an area fraction β (often equal to $1-\alpha$) of trachytope type T_2.

FormNr	Name	Formula
Special classes (1–50)		
1	flood protected area	area fraction shows up as f_b in Eqs. 15.53 and 15.56
2	composite trachytope	fraction α of type T_1 and fraction β (generally $\beta = 1-\alpha$) of type T_2

Area trachytope classes (51–200)

The class of area trachytopes is subdivided into three types: simple (51–100), alluvial (101–150) and vegetation (151–200). Four simple area trachytopes have been implemented representing the four standard roughness types of flow module.

FormNr	Name	Formula
51	White-Colebrook value	k
52	Chézy value	C
53	Manning value	$C = \sqrt[6]{h}/n$
54	z_0 value	$k = 30z_0$

Six alluvial trachytopes have been implemented.

FormNr	Name	Formula
101	simplified Van Rijn	Equation (15.2)
102	power relation	Equation (15.3)
103	Van Rijn (1984)	Equations 15.4 to 15.12
104	Struiksma	Equations 15.13 to 15.16
105	bedforms quadratic	Equation (15.17)
106	bedforms linear	Equation (15.18)

The first alluvial roughness formula is a simplified version of the Van Rijn (1984) alluvial roughness predictor

$$k = Ah^{0.7}\left[1 - e^{-Bh^{-0.3}}\right] \tag{15.2}$$

it is obtained from Equation (15.4) by noting that $h_b \propto h^{0.7}$ and $L_b \propto h$ and ignoring the grain related roughness. The parameters A and B can be calibrated by the user. The second formula implemented is a straightforward general power law

$$C = Ah^B \tag{15.3}$$

where A and B are calibration coefficients. The Van Rijn (1984) alluvial roughness predictor reads

$$k = k_{90} + 1.1 h_b \left(1 - e^{-25 h_b / L_b}\right) \tag{15.4}$$

where the bedform height h_b and length L_b are given by

$$h_b = 0.11 h \left(\frac{D_{50}}{h}\right)^{0.3} \left(1 - e^{-T/2}\right)(25 - T) \tag{15.5}$$

$$L_b = 7.3 h \tag{15.6}$$

where h is the local water depth and the transport stage parameter T is given by

$$T = \frac{u'^2_* - u^2_{*,cr}}{u^2_{*,cr}} \tag{15.7}$$

where u'_* is the bed shear velocity given by

$$u'^2_* = g u^2 / C^2_{g,90} \tag{15.8}$$

where

$$C_{g,90} = 18\ {}^{10}\log(12h/k_{90}) \text{ and } k_{90} = 3 D_{90} \tag{15.9}$$

and $u_{*,cr}$ is the critical bed shear velocity according Shields given by

$$u^2_{*,cr} = g \Delta D_{50} \theta_c \tag{15.10}$$

given

$$\theta_c = \begin{cases} 0.24/D_* & \text{if } D_* \leq 4 \\ 0.14 D_*^{-0.64} & \text{if } 4 < D_* \leq 10 \\ 0.04 D_*^{-0.10} & \text{if } 10 < D_* \leq 20 \\ 0.013 D_*^{0.29} & \text{if } 20 < D_* \leq 150 \\ 0.055 & \text{if } 150 < D_* \end{cases} \tag{15.11}$$

where

$$D_* = D_{50} \left(\frac{g \Delta}{\nu^2}\right)^{1/3} \tag{15.12}$$

This predictor does not contain any calibration coefficients but requires D_{50} and D_{90} data from the morphology module. It does not include the advective and relaxation behaviour that is available by explicitly simulating the dune height as described in section 15.1 combined with trachytope number 106.

The second alluvial roughness predictor proposed by (Struiksma, pers. comm.) allows for a lot of adjustments, it reads

$$\frac{1}{C^2} = (1 - \xi) \frac{1}{C^2_{90}} + \xi \frac{1}{C^2_{min}} \tag{15.13}$$

where

$$C_{90} = A_1\ {}^{10}\log(A_2 h / D_{90}) \tag{15.14}$$

and

$$\xi = \frac{\max(0, \theta_g - \theta_c)}{\theta_m - \theta_c} \frac{\theta_m^2 - \theta_c \theta_g}{(\theta_m - \theta_c)\theta_g} \tag{15.15}$$

which varies from 0 at $\theta_g \leq \theta_c$ to 1 at $\theta_g = \theta_m$ where

$$\theta_g = \frac{u^2}{C_{90}^2 \Delta D_{50}} \tag{15.16}$$

and $A_1, A_2, \theta_c, \theta_m, C_{\min}$ are coefficients that the user needs to specify. This formula requires also D_{50} and D_{90} data from the morphology module. The fifth formula is based on Van Rijn (2007) and reads

$$k = \min(\sqrt{k_{s,r}^2 + k_{s,mr}^2 + k_{s,d}^2}, \frac{h}{2}) \tag{15.17}$$

It uses the roughness heights of ripples k_r, mega-ripples k_{mr} and dunes k_d. These formulae depend on sediment properties D_{50} and D_{90} data which may be either specified as part of the roughness type or obtained from the morphology module. The sixth formula is similar, but uses a linear addition

$$k = \min(k_{s,r} + k_{s,mr} + k_{s,d}, \frac{h}{2}) \tag{15.18}$$

Four vegetation based area trachytopes have been implemented. Two formulae (referred to as ‘Barneveld’) are based on the work by Klopstra *et al.* (1996, 1997) and two on the work by Baptist (2005).

FormNr	Name	Formula
151	Barneveld 1	Eqs. 15.19 – 15.28, $C_D = 1.65$
152	Barneveld 2	Eqs. 15.19 – 15.25, 15.29 – 15.31
153	Baptist 1	Eqs. 15.32 and 15.33
154	Baptist 2	Eqs. 15.34, 15.36 and 15.37

The formula by Klopstra *et al.* (1997) reads

$$C = \frac{1}{h^{3/2}} \left\{ \begin{array}{l} \frac{2}{\sqrt{2A}} \left(\sqrt{C_3 e^{h_v \sqrt{2A}} + u_{v0}^2} - \sqrt{C_3 + u_{v0}^2} \right) + \\ \qquad \frac{u_{v0}}{\sqrt{2A}} \ln \left(\frac{(\sqrt{C_3 e^{h_v \sqrt{2A}} + u_{v0}^2} - u_{v0})(\sqrt{C_3 + u_{v0}^2} + u_{v0})}{(\sqrt{C_3 e^{h_v \sqrt{2A}} + u_{v0}^2} + u_{v0})(\sqrt{C_3 + u_{v0}^2} - u_{v0})} \right) + \\ \qquad \frac{\sqrt{g(h-(h_v-a))}}{\kappa} \left((h - (h_v - a)) \ln \left(\frac{h-(h_v-a)}{z_0} \right) - a \ln \left(\frac{a}{z_0} \right) - (h - h_v) \right) \end{array} \right\} \tag{15.19}$$

where

$$A = \frac{n C_D}{2\alpha} \tag{15.20}$$

$$C_3 = \frac{2g(h - h_v)}{\alpha \sqrt{2A} (e^{h_v \sqrt{2A}} + e^{-h_v \sqrt{2A}})} \tag{15.21}$$

$$a = \frac{1+\sqrt{1+\frac{4E_1^2\kappa^2(h-h_v)}{g}}}{\frac{2E_1^2\kappa^2}{g}} \tag{15.22}$$

and

$$z_0 = ae^{-F} \tag{15.23}$$

where

$$E_1 = \frac{\sqrt{2A}C_3e^{h_v\sqrt{2A}}}{2\sqrt{C_3e^{h_v\sqrt{2A}}+u_{v0}^2}} \tag{15.24}$$

and

$$F = \frac{\kappa\sqrt{C_3e^{h_v\sqrt{2A}}+u_{v0}^2}}{\sqrt{g(h-(h_v-a))}} \tag{15.25}$$

Here, h is the water depth, h_v is the vegetation height, and $n = mD$ where m is the number of stems per square metre and D is the stem diameter. For the first implementation the parameter α in Equation (15.21) is given by

$$\alpha = \max(0.001, 0.01\sqrt{hh_v}) \tag{15.26}$$

and the velocity within the vegetation is approximated by $u_{v0}\sqrt{i}$ where

$$u_{v0}^2 = \frac{2g}{C_D n} \tag{15.27}$$

and i is the water level gradient. For emerged vegetation the first implementation reads

$$\frac{1}{C^2} = \frac{C_D nh}{2g} \tag{15.28}$$

The second implementation of Klopstra *et al.* (1996) is based on a modification by Van Velzen *et al.* (2003); it is identical except for the following modifications to Eqs. 15.26 – 15.28. The main difference between the two implementations is the inclusion of the roughness C_b of the bed itself (without vegetation). The parameter α in Equation (15.21) is now given by

$$\alpha = 0.0227h_v^{0.7} \tag{15.29}$$

and the velocity within the vegetation is approximated by $u_{v0}\sqrt{i}$ where

$$u_{v0}^2 = \frac{h_v}{\frac{C_D h_v n}{2g} + \frac{1}{C_b^2}} \tag{15.30}$$

and i is the water level gradient. For emerged vegetation the second implementation reads

$$\frac{1}{C^2} = \frac{C_D nh}{2g} + \frac{1}{C_b^2} \tag{15.31}$$

For large values of C_b the latter two equations simplify to the corresponding equations of the first implementation. The first implementation requires vegetation height h_v and density n as input parameters (the drag coefficient C_D is equal to 1.65); for second implementation you'll also need to specify the drag coefficient C_D and the alluvial bed roughness k_b (C_b in Equation (15.31) is computed as $18\ {}^{10}\log(12h/k_b)$).

The first implementation of the roughness predictor by Baptist (Baptist, 2005) reads for the case of submerged vegetation

$$C = \frac{1}{\sqrt{\frac{1}{C_b^2} + \frac{C_D n h_v}{2g}}} + \frac{\sqrt{g}}{\kappa} \ln(\frac{h}{h_v}) \tag{15.32}$$

where n is the vegetation density ($n = mD$ where m is the number of stems per square metre and D is the stem diameter). The second term goes to zero at the transition from submerged to emerged vegetation. At that transition the formula changes into the formula for non-submerged vegetation which reads

$$C = \frac{1}{\sqrt{\frac{1}{C_b^2} + \frac{C_D n h}{2g}}} \tag{15.33}$$

which is identical to the non-submerged case of the second implementation of the work by Klopstra *et al.* (1996) (see Equation (15.31)).

The drawback of the three vegetation based formulations above is that they parameterize the flow resistance by means of the bed roughness. Consequently, the presence of vegetation will lead to a higher bed roughness and thus to a higher bed shear stress and larger sediment transport rates in case of morphological computations. Therefore, we have included a $-\frac{\lambda}{2}u^2$ term in the momentum equation where λ represents the flow resistance of the vegetation. For the case of non-submerged vegetation $h < h_v$ the flow resistance and bed roughness are strictly separated

$$C = C_b \quad \text{and} \quad \lambda = C_D n \tag{15.34}$$

In the case of submerged vegetation $h > h_v$ the two terms can't be split in an equally clean manner. However, we can split the terms such that the bed shear stress computed using the depth averaged velocity u and the net bed roughness C equals the bed shear stress computed using the velocity u_v within the vegetation layer and the real bed roughness C_b.

$$\frac{u^2}{C^2} = \frac{u_v^2}{C_b^2} \tag{15.35}$$

With this additional requirement we can rewrite Equation (15.32) as

$$C = C_b + \frac{\sqrt{g}}{\kappa} \ln(\frac{h}{h_v}) \sqrt{1 + \frac{C_D n h_v C_b^2}{2g}} \tag{15.36}$$

and

$$\lambda = C_D n \frac{h_v}{h} \frac{C_b^2}{C^2} \tag{15.37}$$

which simplify to Equation (15.34) for $h = h_v$. Both formulae by Baptist require vegetation height h_v, density n, drag coefficient C_D and alluvial bed roughness C_b as input parameters.

Linear trachytope classes (201–250)

Two formulae have been implemented for linear trachytopes such as hedges or bridge piers.

FormNr	Name	Formula
201	hedges 1	Eqs. 15.38 to 15.40
202	hedges 2	Eqs. 15.41 to 15.43

The first implementation reads

$$\frac{1}{C^2} = \frac{h}{2g} \frac{L_{hedge}}{W_{cell} L_{cell}} \frac{1-\mu^2}{\mu^2} \tag{15.38}$$

where L_{hedge} is the projected length of the hedge, W_{cell} and L_{cell} are the width and length of the grid cell. The ratio L_{hedge}/W_{cell} may be interpreted as the number of hedges that the flow encounters per unit width. The second ratio is thus the inverse of the average distance between these hedges within the grid cell. The last term may be loosely referred to as the drag of the hedge, which is determined by the hedge pass factor μ given by

$$\mu = 1 + 0.175n \left(\frac{h}{h_v} - 2 \right) \tag{15.39}$$

if the hedge extends above the water level ($h_v > h$) and is given by

$$\mu = 1 - 0.175n \left(\frac{h}{h_v} \right) \tag{15.40}$$

if the hedge is fully submerged ($h > h_v$) where n is a dimensionless hedge density. The second implementation reads

$$\frac{1}{C^2} = \frac{C_D n L_{hedge} h}{2g L_{cell} W_{cell}} \tag{15.41}$$

or equivalently

$$C = \sqrt{\frac{2g L_{cell} W_{cell}}{h L_{hedge}}} \left(\sqrt{\frac{1}{C_D n}} \right) \tag{15.42}$$

for non-submerged conditions and

$$C = \sqrt{\frac{2g L_{cell} W_{cell}}{h L_{hedge}}} \left(\frac{h_v}{h} \sqrt{\frac{1}{C_D n}} + m_0 \sqrt{\frac{\left(\frac{h-h_v}{h}\right)^2}{1 - \left(\frac{h-h_v}{h}\right)^2}} \right) \tag{15.43}$$

for submerged conditions. We recognize the same ratio $L_{cell} W_{cell}/L_{hedge}$ that represents the average distance between hedges. Equation (15.41) can be directly compared to similar equations for area trachytopes (Equation (15.28)), point trachytopes (Equation (15.44)). Note that the formula for computing the loss coefficient for a bridge explicitly includes the reduction in the flow area and the resulting increase in the effective flow velocity, whereas the above mentioned trachytope formulae don't.

Point trachytope classes: various (251–300)

One formula for point trachytopes has been implemented. It may be used to represent groups of individual trees or on a smaller scale plants.

FormNr	**Name**	**Formula**
251	trees	Eqn. 15.44

The implemented formula reads

$$C = \sqrt{\frac{2g}{C_D n \min(h_v, h)}} \tag{15.44}$$

where $n = mD$ with m the number of trees per unit area and D the characteristic tree diameter, h_v is the vegetation height and h is the local water depth. The formula is identical to Equation (15.33) except for the fact that the point trachytope formula has no bed roughness and area associated with it. The generalization of Equation (15.44) to the submerged case ($h > h_v$) lacks the extra term in Equation (15.32).

15.2.2 Averaging and accumulation of trachytopes

Point and linear roughnesses are accumulated by summing the inverse of the squared Chézy values C_i.

$$\frac{1}{C_{pnt}^2} = \sum_i \frac{1}{C_{pnt,i}^2} \tag{15.45}$$

$$\frac{1}{C_{lin}^2} = \sum_i \frac{1}{C_{lin,i}^2} \tag{15.46}$$

The area roughnesses are accumulated weighted by the surface area fraction f_i. These roughnesses are accumulated as White-Colebrook roughness values and as Chézy values; for the latter values both the linear sum (“parallel”) and the sum of inverse of squared values (“serial”) are computed. Roughness values are converted into each other as needed based on the local water depth.

$$k_{area} = \sum_i f_i k_i \tag{15.47}$$

$$\frac{1}{C_{area,s}^2} = \sum_i f_i \frac{1}{C_i^2} \tag{15.48}$$

$$C_{area,p} = \sum_i f_i C_i \tag{15.49}$$

For the fraction of the grid cell area for which no roughness class is specified the default roughness is used, via keywords `UnifFrictCoef` and `UnifFrictType`.

The flow resistance coefficients are also accumulated proportionally to the surface area fraction f_i associated with the trachytope considered. For the fraction of the grid cell area for which no flow resistance is specified, obviously none is used.

$$\lambda = \sum_i f_i \lambda_i \tag{15.50}$$

The final effective bed roughness of the grid cell may be computed by either one of the following two methods.

Method 1

The total mean roughness is computed by summing the White-Colebrook values for the areas and line and point resistance features.

$$k_m = k_{area} + k_{lin} + k_{pnt} \tag{15.51}$$

where $k_{lin} = 12h10^{-C_{lin}/18}$ and $k_{pnt} = 12h10^{-C_{pnt}/18}$. The effect of the water free area fraction f_b is taken into account by means of the following empirical relation in which $C_m = 18\ {}^{10}\log(12h/k_m)$ is the mean Chézy value corresponding to the total mean White-Colebrook roughness value obtained from Equation (15.51).

$$f_b = \max(\min(0.843, f_b), 0.014) \tag{15.52}$$

$$C_{\text{total}} = C_m \left(1.12 - 0.25 f_b - 0.99\sqrt{f_b}\right) \tag{15.53}$$

The resulting C_{total} value is used in the computation. This method together with trachytope classes 1, 51, 101, 151 and 201 corresponds to the NIKURADSE option of the WAQUA/TRIWAQ flow solver.

Method 2

The total roughness is computed by first averaging over the serial and parallel averages of the Chézy values according

$$C_{area} = \alpha_s C_{area,s} + (1 - \alpha_s) C_{area,p} \tag{15.54}$$

where $\alpha_s = 0.6$ by default. Subsequenty the effect of the water free area fraction f_b is taken into account by means of the following empirical relation (identical to Equation (15.53) of method 1).

$$f_b = \max(\min(0.843, f_b), 0.014) \tag{15.55}$$

$$C_{area,corr} = C_{area} \left(1.12 - 0.25 f_b - 0.99\sqrt{f_b}\right) \tag{15.56}$$

Finally the Chézy value representing the total bed roughness is computed by accumulating the inverses of the squared Chézy values.

$$\frac{1}{C_{total}^2} = \frac{1}{C_{area,corr}^2} + \frac{1}{C_{lin}^2} + \frac{1}{C_{pnt}^2} \tag{15.57}$$

The resulting C_{total} value is used in the computation. This method together with trachytope classes 1, 51, 52, 53, 101, 152, 202 and 251 corresponds to the ROUGHCOMBINATION option of the WAQUA/TRIWAQ flow solver.

15.3 Dynamic vegetation model

15.3.1 Introduction

Next to the vegetation approaches in the previous sections of this chapter D-Flow FM also offers the possibility of a dynamic vegetation model. Then, an ecological model that has been developed in Python is coupled to D-Flow FM. By doing so a fast and flexible coupling platform is created to answer questions about the design and evaluation of Nature-based solutions (NbS), thereby enabling assessments that were considered too complicated and costly before. The software used is open software.

15.3.2 Philosophies of dynamic vegetation model

Faced with the problems of climate change and socio-economic pressure in many of the world's deltas and rivers, there is a rising interest in Nature-based solutions as a means to combine affordable and adaptive flood risk reduction with other ecosystem services. This has created a demand for tools that can quantify the development and performance of natural or nature-based systems like salt marshes, mangroves and river floodplains and the ecosystem services they provide. The NbS Dynamics modelling suite provides the tools to simulate interactions between hydrodynamics, sediment dynamics, morphodynamics and vegetation/ecology.

The ecological model can be as simple as a single habitat suitability rule, or endlessly complex involving multiple species, age classes and stressors. To facilitate the assessment of ecological processes, Delft3D Flexible Mesh has been expanded with summarizing statistics of ecologically relevant parameters (e.g. inundation time, bed shear stress). The ecological model is not restricted to vegetation only. Other biota that interact with hydrodynamics or morphology, such as mussels, algae and microfytobenthos, can also be computed with this modelling suite.

A substantial improvement over the earlier academic tools is the direct exchange of parameters through memory pointers, instead of via files. This is much faster, allows for flexible exchange intervals (i.e. only when really needed) and allows for relatively independent development of both parts of the coupled code.

15.3.3 Conceptual description

The vegetation module in Python is coupled to a hydrodynamic D-Flow FM model. The Python environment has two functions: it contains the vegetation module and it 'orchestrates' the interaction between the hydrodynamic and the vegetation modules. 'Orchestrating' means that Python is used to define when and via which parameters the models interact. To do so, it uses the Basic Model Interface (BMI) technique (Hutton et al., 2020; Peckham et al., 2013). The BMI interface of D-Flow FM is described in more detail in Section 17.3).

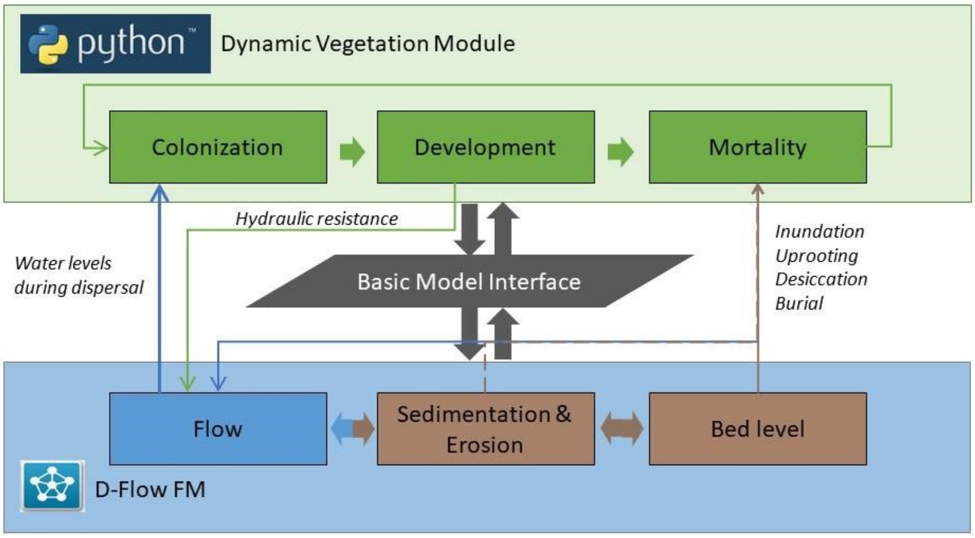


Figure 15.1: *Conceptual schematization of the modelling environment, visualizing the Basic Model Interface connecting the D-Flow FM model for hydrodynamic and morphodynamic computation with the Python environment for vegetation and ecological modelling*

The modelling environment is illustrated by a conceptual schematization in Figure 15.1. The hydrodynamic module D-Flow FM computes for example the water level, flow velocity and bed shear stress and passes these results to the vegetation module (Python). The vegetation module then computes the vegetation biomass, expressed as a stem density, stem height and stem diameter. These vegetation parameters are used as inputs for D-Flow FM in the next timestep.

15.4 (Rigid) three-dimensional vegetation model

The (rigid) 3D Vegetation model (Winterwerp and Uittenbogaard (1997)), as known from Delft3D-FLOW, is not available yet in D-Flow FM.

16 Calibration factor

The current chapter explains the effect of the calibration factor. The calibration factor is multiplier of the roughness. The calibration factor may be constant, discharge- or water-level-dependent. By assigning different areas to different calibration classes, each region can be independently calibrated. Calibration roughness definitions can also be combined by assigning multiple field definitions and a weighting for each gridcell edge (net link). This approach allows thus both abrupt and gradual transitions and the modeller can control how the calibration factor is going to be used. The approach is similar to the roughness and link definitions used in the trachytopes module for alluivial- and bedform-roughness (see chapter 15).

The calibration factor is defined by means of two files (see section C.9):

- ◇ Calibration factor definition file (CLD-file). This file defines the calibration factor (e.g. constant, water-level- or discharge dependent).
- ◇ Calibration factor area file (CLL-file). This file associates calibration factor defintions with the edges of the model grid and include a relative weighting.

The resulting weighted calibration factor is multiplied by the roughness type as expressed by the `UnifFrictCoef` keyword in the [physics] section in the .mdu file (see Appendix A). This implies that the effect of the calibration will be different depending on the definition of `UnifFrictCoef`. For example, if the `UnifFrictCoef`=0 (i.e. Chézy) a local calibration factor of 0.5 will imply an increase of the bed shear stress, whereas if `UnifFrictCoef`=2 (i.e. White-Colebrook) a reduction of the bed shear stress may be expected. A background calibration factor equal to one is applied if the sum of the weights at a single link is lower than one.

The calibration factor approach cannot be combined with multiple roughness types as specified through the external forcing file. This will lead to an error. Such a spatial variation in the roughness can be achieved by defining these areas through the trachytopes module.

It is good to be aware of the following known differences with the trachytopes module:

- ◇ The weighting of the calibration factors is done for all entries in the calibration area defintion file. Averaging for the trachytopes area file is only done for the last sequence of trachytope area defintions at one and the same location.
- ◇ When a calibration factor definition is imposed at a location outside of the grid, and this calibration factor definition depends on a water level station or cross-section which is also outside of the domain, the model crashes, however the trachytopes module allows this.

17 Coupling with D-RTC (FBC-Tools)

This chapter is on the *coupling* of hydrodynamics and real-time control — more specific: feedback control — of hydraulic structures.

17.1 Introduction

The D-Flow FM User Interface implements the concept of an *Integrated model* in order to couple different models, such as: hydrodynamics coupled with the controlling of structures, waves, morphology and/or water quality.

Two types of coupling are distinguished:

1 *offline* coupling and
2 *online* coupling.

Note: *Offline* is also referred to as *sequential* coupling and *online* as *parallel* coupling.

Offline

In case of an *Integrated model* with *offline* coupling, the entire hydrodynamic simulation is done first, i.e., *separately* from the second simulation. The file-based hydrodynamic output serves as input for the second simulation. As such, the hydrodynamic results drives the controlling of structures or the simulation of waves or water quality. In this offline case, there is no feedback from the waves or water quality to the hydrodynamic simulation. For many applications, this is good practice.

Online

An *online* coupling, on the other hand, exchanges data *every time* after computing a specified time interval. This tight coupling allows for direct feedback of the various processes on one another. This is crucial for controlling structures.

A coupled flow-rtc model can be run either as an Integrated Model from within the D-Flow FM User Interface, or from the commandline using the DIMR program (Deltares Integrated Model Runner).

In case of a flow-rtc coupling

- ◇ the flow model will exchange for example *Water levels*
- ◇ subsequently, the rtc model will prescribe for example the *Crest level* of a controlled structure, based on the exchanged *water levels*
- ◇ in case of an *online* coupling this *Crest level* will influence the hydrodynamic simulation.

17.2 Getting started

17.2.1 User interface: the first steps

First, the user must add a *real-time control* model to the *flow-fm* model. The **Project** window will look like this.

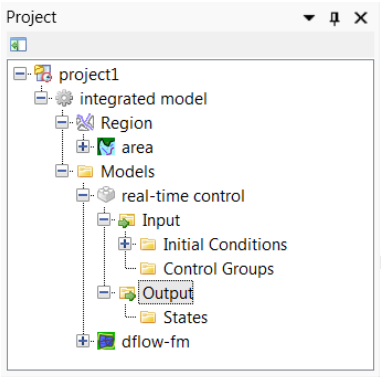


Figure 17.1: *An* Integrated model *in the* ***Project*** *window*

Secondly, the user will model the control flow for an hydraulic structure. This may look like this.

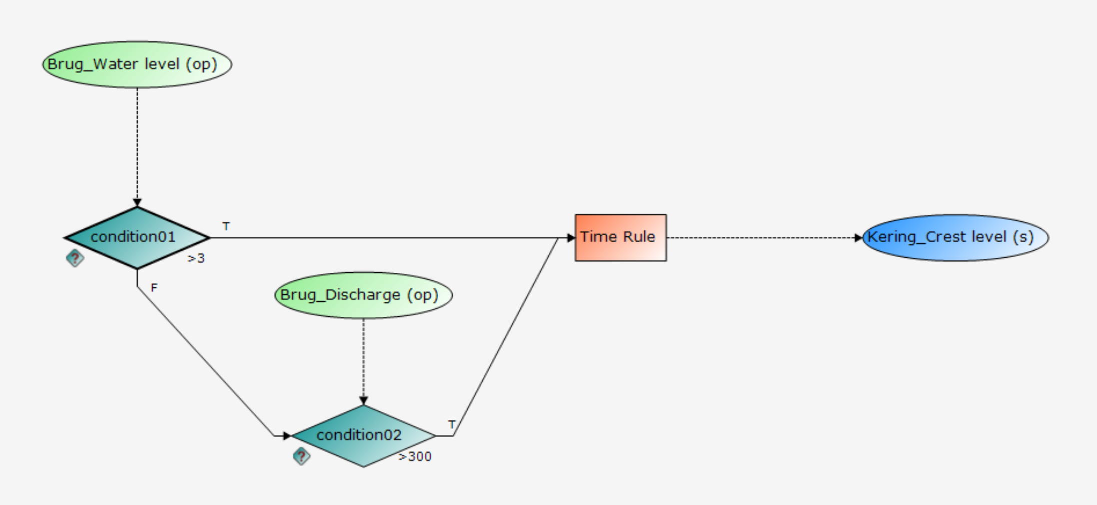


Figure 17.2: *Example of a* Control flow *in D-RTC*

Full documentation on D-RTC is available in its own User Manual. This chapter is limited to the details of running coupled flow-rtc models.

17.2.2 Input D-Flow FM

The hydraulic structures that are normally driven by scalar values or time series in <∗.tim> or <∗.bc> files, will now be fed by D-RTC. Replace the timeseries file name by the `realtime` keyword in the <structures.ini> file (section C.12):

```
[Structure]
type          = weir          # Type of structure
id            = weir01        # Name of the structure
polylinefile = weir01.pli    # *.pli; Polyline geometry definition for 2D structure
CrestLevel    = realtime      # Crest level in [m]
```

Also, the MDU file may optionally contain an observation file and/or a cross section file, such that D-RTC's triggers can be set to respond to the computed values at these locations.

```
[output]
ObsFile      = obs.xyn
CrsFile      = river_crs.pli
```

17.2.3 Input D-RTC

The input of D-RTC consists of several <∗.xml> files and a toplevel <settings.json>. We refer to the D-RTC User Manual (D-RTC UM, 2019) for further details.

17.2.4 Input DIMR

In **??**, details on a coupled flow-rtc-wave model are presented. This can serve as an example for a flow-rtc coupling, by leaving out all wave-related fragments.

17.2.5 Online process order

This is discussed in **??** for a coupled flow-rtc-wave model. This can serve as an example for a flow-rtc coupling, by leaving out all wave-related fragments.

17.3 Technical background

The D-Flow FM kernel shared library offers an API that implements the Basic Model Interface (BMI). The BMI allows one to initialize, run, and finalize a model component via standardized interface calls; furthermore it allows one to get and set various quantities such that model components can dynamically react to the computed results. General details on the BMI can be found in Peckham *et al.* (2013). This section discusses some of the D-Flow FM specifics.

17.3.1 BMI interface to interact with the D-Flow FM model state

Many variables in D-Flow FM's model state can be inquired and changed via BMI calls. Setting a variable can be done via a call to `set_var(varname, userdata)`, which copies the data from the `userdata` array to the D-Flow FM memory associated with the `varname` variable. Alternatively, one may use a call to `get_var(varname, xptr)`, which returns a pointer to the requested variable, and directly assign new values to the D-Flow FM memory from outside the running simulation. The latter approach may be faster because less data needs to be copied.

17.3.2 DIMR and the BMI

The DIMR program takes care of the exchange of data between components, and each component will be addressed using BMI calls. The data to be exchanged is defined in the <dimr.xml> configuration file, under the `<dimrConfig><coupler><item>` elements. More details on DIMR can be found in Deltares (2024).

The variable names specified there need to be supported by the addressed component. For the coupling to D-RTC typically scalar quantities need to be exchanged that are defined at specific locations (such as observation points or structures). Since BMI only identifies variables by a unique variable name without any location selection, D-Flow FM identifies both location and quantity by introducing *compound* variable names (not to be confused with a compound structure). A compound variable name takes the form:

```
varname = "groupname/locationid/fieldname"
```

For example the variable name "`weirs/Lith01/CrestLevel`" will select the crest level at the weir named "Lith01".

The `groupname` part of the variable name identifies the location or object type. Currently known group names are listed below. Unless noted otherwise, please check Section 17.3.3 for the list of quantities supported for those groups of locations.

- `pumps`
- `weirs`
- `orifices`
- `gates`
- `generalstructures`
- `culverts`
- `dambreak`
- `sourcesinks`
- `observations` (observation points, see Section 17.3.4)
- `crosssections` (observation cross sections, see Section 17.3.4)
- `laterals` (see Section 17.3.5)

The `locationid` part of the variable name is the unique identifier of the intended object. More details on the specific identifier used can be found in the subsequent sections on the various object groups. The `fieldname` identifies the quantity at the selected location. Which quantities are supported depends on the `groupname`.

17.3.3 BMI access to hydraulic structure data in D-Flow FM

To access quantities defined at structures use a compound variable name as explained in Section 17.3.2. The `locationid` part of the compound variable name is, for hydraulic structures, equal to the `id` key in the structure input file (Table C.7). Which quantities can be addressed via the BMI, depends on the structure type indicated by the `groupname`. Table 17.1 lists the supported field names for each hydraulic structure group.

Table 17.1: Supported compound variable fields for hydraulic structures in D-Flow FM via BMI.

Group name	Field name	Access	Remark
`pumps`			
	`Capacity`	read-write	Only for non-staged pumps. Prescribed capacity, not necessarily actual discharge.
`weirs`			
	`CrestLevel`	read-write	
`orifices`			
	`GateLowerEdgeLevel`	read-write	
`gates`			
	`CrestLevel`	read-write	
	`GateHeight`	read-write	
	`GateLowerEdgeLevel`	read-write	

(continued on next page)

(continued from previous page)

Group name	Field name	Access	Remark
	GateOpeningWidth	read-write	
generalstructures			
	CrestLevel	read-write	
	GateHeight	read-write	
	GateLowerEdgeLevel	read-write	
	GateOpeningWidth	read-write	
longculverts			
	ValveRelativeOpening	read-write	
culverts			
	ValveOpeningHeight	read-write	
dambreak			Currently only read-only, not settable.
	DambreakS1up	read-only	
	DambreakS1dn	read-only	
	DambreakBreach_depth	read-only	
	DambreakBreach_width	read-only	
	DambreakInstantaneous_discharge	read-only	
	DambreakCumulative_discharge	read-only	
sourcesinks			
	Discharge	read-write	Prescribed discharge, not necessarily actual discharge.
	ChangeInSalinity	read-write	Prescribed change, not necessarily actual change.
	ChangeInTemperature	read-write	Prescribed change, not necessarily actual change.

17.3.4 BMI access to observations in D-Flow FM

To access quantities defined at observation points and observation cross sections use a compound variable name as explained in Section 17.3.2. The locationid for observation points should match the name specified in the relevant input file (See Section F.2.2). All observation point names must be unique when accessing the data via BMI. The same holds for the locationid for observation cross sections (See Section F.2.4).

Table 17.2 lists the supported field names for each observation group. All supported variables/fields are read-only, evidently, as observations are only 'observing' the flow. Typically, these variables are used as input to hydraulic triggers in an RTC-coupling.

Table 17.2: *Supported compound variable fields for observations in D-Flow FM via BMI.*

Group name	Field name	Access	Remark
observations			Observation points
	water_level	read-only	

(continued on next page)

Group name	Field name	Access	Remark
			(continued from previous page)
	`water_depth`	read-only	
	`velocity`	read-only	
	`discharge`	read-only	
	`salinity`	read-only	Only when salinity transport is on.
	`temperature`	read-only	Only when temperature transport is on.
	`<tracername>`	read-only	Only when that named tracer is in the model.
`crosssections`			Observation cross sections
	`discharge`	read-only	Space-integrated along cross section.
	`velocity`	read-only	Space-averaged along cross section.
	`water_level`	read-only	Space-averaged along cross section.
	`water_depth`	read-only	Space-averaged along cross section.

17.3.5 BMI access to forcings in D-Flow FM

To access quantities defined as local forcings (currently only lateral discharges) use a compound variable name as explained in Section 17.3.2. The `locationid` for lateral discharges should match the `id` specified for them in the external forcings input (See Section C.5.2.2). Table 17.3 lists the supported field names for each forcing group.

Table 17.3: *Supported compound variable fields for forcings in D-Flow FM via BMI.*

Group name	Field name	Access	Remark
`laterals`			
	`water_discharge`	read-write	Prescribed discharge, not necessarily actual discharge (when negative).
	`water_level`	read-only	Point value of water level of the first pressure point affected by lateral.
	`number_of_layers`	read-only	Actual number of layers in the water column of the lateral. Dimension of the array for the following Field names.
	`water_discharge_per_layer`	read-write	Prescribed discharge, distributed over the layers.
	`incoming/<substance>`	read-write	Concentration of a transported quantity in incoming direction, array with 1 value per layer.
	`outgoing/<substance>`	read-only	Concentration of a transported quantity in outgoing direction, averaged value over the lateral, per layer.

Remark:

◇ <substance> can be salinity, temperature or <tracer>.

18 Coupling with D-Rainfall Runoff

This chapter is on the *coupling* of hydrodynamics and lumped rainfall-runoff.

18.1 Introduction

The Delta Shell framework implements the concept of an *Integrated model* in order to couple different models, for example: hydrodynamics coupled with lumped rainfall-runoff. In this case, only a so-called *online* coupling is relevant.

Online

An *online* coupling exchanges data *every time* after computing a specified time interval. This tight coupling allows for direct feedback of the various processes on one another. This is crucial for D-RR processes that are dependent on surface water levels computed by D-Flow FM.

A coupled D-Flow FM–D-RR model can be run either as an Integrated Model from within Delta Shell, or from the commandline using the dimr program (Deltares Integrated Model Runner).

In case of a D-Flow FM–D-RR coupling

◇ the D-Flow FM model might exchange for example *Water levels*;
◇ subsequently, the D-RR model will prescribe typically the *discharge* of one or more lateral discharges. Depending on the type of runoff model selected, this can be based on the exchanged *water levels*.

18.2 Getting started

18.2.1 User interface: the first steps

The GUI presents a coupled model as an *Integrated Model*, an example is shown in Figure 18.1.

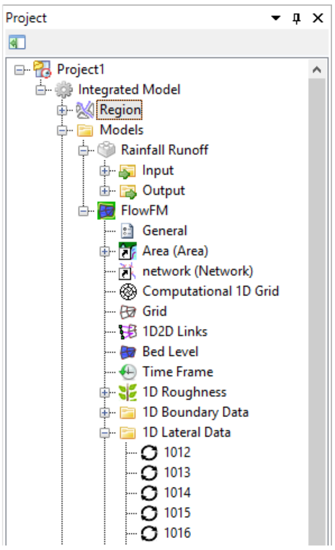
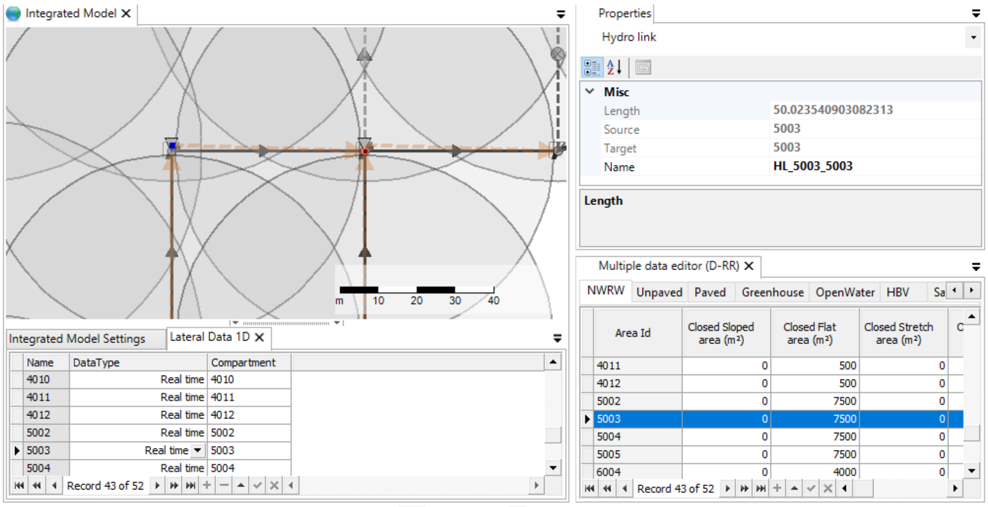


Figure 18.1: *Example of an* Integrated Model *with D-Flow FM and D-RR.*

The D-RR catchment nodes will yield runoff discharges that are to be coupled to D-Flow FM

(1D) lateral discharges. In the GUI, these couplings are called hydro links, as shown in Figure 18.2.



Name	DataType	Compartment
4010	Real time	4010
4011	Real time	4011
4012	Real time	4012
5002	Real time	5002
5003	Real time	5003
5004	Real time	5004

Record 43 of 52

Misc

Length	50.023540903082313
Source	5003
Target	5003
Name	HL_5003_5003

Length

Area Id	Closed Sloped area (m²)	Closed Flat area (m²)	Closed Stretch area (m²)	C
4011	0	500	0	
4012	0	500	0	
5002	0	7500	0	
5003	0	7500	0	
5004	0	7500	0	
5005	0	7500	0	
6004	0	4000	0	

Record 43 of 52

Figure 18.2: *Hydro links in the GUI: connection between D-RR catchments and D-Flow FM lateral discharges in 1D.*

Full documentation on D-RR is available in its own User Manual (D-RR_UM, 2019). The remainder of this chapter is limited to the details of running coupled D-Flow FM–D-RR models.

18.2.2 Input D-Flow FM

The lateral discharges that are normally driven by scalar values or time series in <∗.tim> or <∗.bc> files, will now be fed by D-RR. Replace the timeseries file name by the `realtime` keyword in the external forcings file (section C.5.2.2):

```
[Lateral]
id             = bnd_0230_007v
type           = discharge
nodeId         = NetworkNodeA
discharge      = realtime      # Discharge in [m3/s]
```

18.2.3 Input D-RR

The input of D-RR consists of the definition of catchments, and boundary conditions (and more). We refer to the D-RR User Manual (D-RR_UM, 2019) for further details.

18.2.4 Input DIMR

Both D-Flow FM and D-RR are used as dynamic libraries (DLL's on Windows, so's on Linux). DIMR is a small executable steering both dynamic libraries. Its input file, usually called "dimr_config.xml", looks like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dimrConfig xmlns="http://schemas.deltares.nl/dimrConfig"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://schemas.deltares.nl/dimrConfig
           content.oss.deltares.nl/schemas/dimr-1.2.xsd">
   <documentation>
      <fileVersion>1.2</fileVersion>
      <createdBy>Deltares, Coupling team</createdBy>
```

```
        <creationDate>2018-08-28T10:06:09.3197094Z</creationDate>
    </documentation>
  <control>
    <parallel>
      <startGroup>
        <time>0 60 7200</time>
        <start name="myNameRR"/>
        <coupler name="rr2flow"/>
      </startGroup>
      <start name="myNameDFlowFM"/>
    </parallel>
  </control>
  <component name="myNameDFlowFM">
    <library>dflowfm</library>
    <workingDir>dflowfm</workingDir>
    <inputFile>FlowFM.mdu</inputFile>
  </component>
  <component name="myNameRR">
    <library>rr_dll</library>
    <workingDir>rr</workingDir>
    <inputFile>Sobek_3b.fnm</inputFile>
  </component>
  <coupler name="rr2flow">
    <sourceComponent>myNameRR</sourceComponent>
    <targetComponent>myNameDFlowFM</targetComponent>
    <item>
      <sourceName>catchments/0230.007v/water_discharge</sourceName>
      <targetName>laterals/bnd_0230_007v/water_discharge</targetName>
    </item>
    <item>
      <sourceName>catchments/WR-8475_02/water_discharge</sourceName>
      <targetName>laterals/bnd_WR8475_02/water_discharge</targetName>
    </item>
    <logger>
      <workingDir>.</workingDir>
      <outputFile>to_dflowfm.nc</outputFile>
    </logger>
  </coupler>
</dimrConfig>
```

The meaning of all XML elements in this listing is documented on page 366.

18.2.5 Online process order

This is discussed in **??** for a coupled dflowfm-drtc-dwaves model. This can also serve as an example for a D-Flow FM +D-RR coupling, by leaving out all wave-related fragments and replacing D-RTC by D-RR.

18.3 Technical background

The technical background of the D-Flow FM–D-RR coupling is the same as for the D-Flow FM–D-RTC coupling, so for details refer to section 17.3. The only relevant BMI-exchangeable variable here is `laterals/bnd_0230_007v/water_discharge`, as listed in Table 17.3. Additionally, read-only access to `laterals/<lateralid>/water_level` is available.

19 Coupling with D-Water Quality (Delwaq)

19.1 Introduction

D-Water Quality is a multi-dimensional water quality model framework developed by Deltares over the past decades. It solves the advection-diffusion-reaction equation on a predefined computational grid and for a wide range of model substances. D-Water Quality offers flexible configuration of the substances to be included, as well as the processes to be evaluated. D-Water Quality is not a hydrodynamic model, so information on flow fields is obtained from hydraulic models such as D-Flow 1D (SOBEK 3) and D-Flow FM.

Here, only the *coupling* is discussed. Full documentation on D-Water Quality is available, see (Deltares, 2024e).

19.2 Offline versus online coupling

The D-Flow FM User Interface implements the concept of an *Integrated model* in order to couple different models, such as: hydrodynamics coupled with the controlling of structures, waves, morphology and/or water quality.

Two types of coupling are distinguished:

1 *offline* coupling and
2 *online* coupling.

Note: *Offline* is also referred to as *sequential* coupling and *online* as *parallel* coupling.

Offline

In case of an *Integrated model* with *offline* coupling, the entire hydrodynamic simulation is done first, i.e., *separately* from the second simulation. The file-based hydrodynamic output serves as input for the second simulation. As such, the hydrodynamic results drives the controlling of structures or the simulation of waves or water quality. In this offline case, there is no feedback from the waves or water quality to the hydrodynamic simulation. For many applications, this is good practice.

Online

An *online* coupling, on the other hand, exchanges data *every time* after computing a specified time interval. This tight coupling allows for direct feedback of the various processes on one another. This is crucial for controlling structures.

The *online* coupling can also be beneficial for the storage requirements of simulations, since in the online coupling the data is already in memory. Consequently, not every time step of the hydrodynamics has to be stored, and larger save intervals can be used. A downside of this method is that when the user wants to evaluate different water quality configurations, for each simulation not only the water quality simulation, but also the hydrodynamics need to be recomputed. This can imply recomputing the same hydrodynamic result for every water quality configuration, and can be expensive.

The *online* coupling is released as beta functionality. At this moment, this functionality is not supported in the user interface.

Similarities and differences between coupling modes for water quality

Both methods of coupling use the same processes library for water quality. The simulation of substances, processes and their interactions is identical. The key difference is the computation of the transport of substances. In the *offline* coupling, the transport is computed by D-Water Quality, giving the user a range of integration options to use for the tranport equation (see (Deltares, 2024e) for details). In the *online* coupling, the transport is computed by D-Flow Flexible Mesh. The user has less choice with respect to integration options.

19.3 Creating output for D-Water Quality

Creating output for D-Water Quality by D-Flow FM can be enabled in the User Interface in the main window under the tab ‘Output Parameters’. Change the “WAQ output interval” to a nonzero value. The MDU-equivalent is the keyword `[output]`, `WaqInterval`, where three numbers can be put and they are in the order of “WAQinterval”, “WAQ output-start-time”, “WAQ output-end-time”. Each number should be a whole number of seconds, and must be a multiple of the “User time step” (see tab “Time frame”). Moreover, if the start and end output time is not explicitly specified, they are automatically set to start and end time of the simulation, respectively.

D-Flow FM will create a special output folder named <DFM_DELWAQ_*mdu_name*>. In this folder a <*.hyd>-file will be created that gives an overview of the coupling with references to the files containing the hydrodynamic exchange data.

Load the <*.hyd>-file in the D-Water Quality GUI to prepare the input for a water quality calculation. For further information on how to run D-Water Quality please consult its user manual.

The coupling between D-Flow FM and D-Water Quality has currently been tested with good results for 2D and 3D (sigma-layer and z-layer) meshes.

The D-Water Quality GUI fully supports coupling with results from D-Flow FM. Postprocessing can be done in QUICKPLOT.

19.4 Limitations and remarks

When using multiple partitions (as described in Chapter 6), D-Flow FM will create a special output folder per partition named <DFM_DELWAQ_*mdu_name_nnnn*>. Since D-Water Quality can accept results for a single partition only, you have to merge the results into one using a tool called `waqmerge`.

It can be useful to model the water quality processes on a coarser mesh than the hydrodynamic grid, and for this grid aggregation would be useful. This can be done using the tools D-WAQ DIDO (which can be obtained by contacting support) and `agrhyd`.

20 Water Quality Processes

Note: The implementation of water quality processes is currently a β-functionality of D-Flow FM.

20.1 Introduction

Beside using D-Flow FM hydrodynamics in a file base coupling for modeling water quality processes with D-Water Quality as described in chapter 19: Coupling with D-Water Quality (Delwaq), it is also possible to use the processes in the D-Water Quality process library directly in D-Flow FM.

The file based coupling has several advantages. Hydrodynamic calculations can be time consuming, and one hydrodynamic data set can be used to run multiple water quality scenarios. However with the models becoming bigger and finer, the size of the coupling files is increasing. Also since D-Water Quality is only partly parallelised, cannot make use of MPI parallelism, and CPU frequencies aren't increasing any more for years, Delwaq runs have become very time consuming.

This was the reason that the possibility to use the processes of the D-Water Quality process library directly in D-Flow FM was developed. The coupling is implemented in such a way that only the processes from D-Water Quality are used, using the exact same code as used in D-Water Quality. For each substance that is define in the water quality processes setup, a tracer is defined. The advection-diffusion equations are solved by D-Flow FM as described in chapter 9: Transport of matter, which can make use of MPI parallelism. The processes them self do not need to make use of MPI parallelism, since the processes do not interact horizontally, but only in the vertical direction (e.g. sedimentation, light climate).

This approach seems to be useful for models with a high number of flow elements and/or layers, since this will avoid large hydrodynamic communications files. It also allows the use of MPI parallelism for the calculation of transport, and distribution of the water quality process calculations over the MPI nodes, which is possible without any MPI communication. The trade-off is of course that you have to recalculate the hydrodynamics for every run, which can be time consuming. In Table 20.1 you can see an overview of some of the pro's and con's of both approaches.

Table 20.1: *File base D-Water Quality versus D-Water Quality process in D-Flow FM.*

Model type	Pro	Con
File based	- run hydrodynamics once - simple models could be faster - aggregation is possible	- big coupling files - complex models could be slower
Integrated	- no coupling files - no coupling mistakes - use D-Flow FM's MPI capability	- rerun hydrodynamics every scenario - no aggregation possible - can't reuse old water quality setups

Water quality modelling with D-Water Quality within D-Flow FM is very flexible. Substances, water quality processes, output variables, etc. are all free for you to choose from the library, whcih can also be extended by new processes.

Since processes in D-Flow FM are a new development, currently there is no user interface support (except the PLCT to create sub-files). The user has to manual edit the mdu-file and ext-file to set up the processes and their inputs.

The basic steps in water quality modelling within D-Flow FM are:

1. Set up a hydrodynamic simulation and make it suitable your water quality simulation.
2. Define the substances and water quality processes you want to include using the PLCT.
3. Reference to the sub-file you have created in the mdu-file, and optionally to and additional history output file.
4. Choose a sensible processes time step (DtProcesses) and mass balance output interval (MbaInterval) in the mdu-file.
5. Define initial conditions, boundary conditions for your substances in the ext-file.
6. Define constant and spatial or temporal varying process parameters in the ext-file.
7. Define mass balance areas in the ext-file.
8. Run the simulation.
9. Check the output.

More information on this will follow in the next sections.

20.2 Definition of the water quality system

A water quality system is defined by its substances and the processes that influence these substances. These processes can range from simple decay to complex interactions between the substances.

The user can use all substances and processes that are available in D-Water Quality. A comprehensive description of the formulations and the input and output items of the processes can be found in Technical Reference Manual (Deltares, 2024d).

Processes can calculate actual fluxes between substances, but there are also 'informational' processes that e.g. calculate the light climate or the total bottom shear stress or the total resuspension flux. A setup (sub-file) can be created using the PLCT. The use of the Processes Library Configuration Tool (PLCT) is discussed in the User Manual for the Deltares (2024c). For D-Water Quality, this sub-file is read by the user interface, and the settings are incorporated in the D-Water Quality input file. D-Flow FM reads the sub-file directly.

The substance file is in plain text format, which allows manipulation by the user. The substance file contains:

- ◇ a list of substances that are transported (active)
- ◇ a list of substances that are not transported (inactive)
- ◇ a list of constants and their values (see: section 20.3.1)
- ◇ a list of list of outputs (see: ref below)
- ◇ a list of the processes that will be active

To use a sub-file, it has to be mentioned in the mdu-file. Use the keyword `[processes]`, `SubstanceFile`.

Example of the reference to a sub-file in the mdu-file which will activate the processes in D-Flow FM:

```
[processes]
```

```
SubstanceFile                     = sed_3fraction.sub
```

20.2.1 Substances

There a two types of substances in D-Water Quality: active and inactive.

1 Active substances are added to the constituents of D-Flow FM. You can define their initial conditions (or use the result from a previous run using a restart file), boundary conditions and sink-source concentrations in the same ways as for tracers in D-Flow FM.
2 Inactive substances usually reside only in the bottom water layer. They usually represent inorganic sediments or organic material at the bottom, or rooting vegetation.

The non-transported substances are not part of constituents in D-Flow FM. You can define their initial conditions using a special keyword in the ext-file, or use the result from a previous run using a restart file. You cannot define boundary conditions for them, since they are not transported over boundaries. Also, you do not need to add concentrations for them when you use sources and sinks, only for the transported substances.

An actual list of the constituents in the model and their order is written to the diagnostics file.

Format for a substance definition in the sub-file:

```
substance <substance name> <active/inactive>
   description        <decription>
   concentration-unit <unit>
   waste-load-unit    <waste load unit>
end-substance
```

Excerpt of the substance definition in a sub-file:

```
substance 'IM1' active
   description        'inorganic matter (IM1)'
   concentration-unit '(gDM/m3)'
   waste-load-unit    '-'
end-substance
substance 'IM1S1' inactive
   description        'IM1 in layer S1'
   concentration-unit '(gDM/m2)'
   waste-load-unit    '-'
end-substance
```

20.2.2 Processes

The processes in the the D-Water Quality processes library cover a large number of water quality problems. An overview can be seen in Figure 20.1. The user can select which processes are relevant and switched on. D-Flow FM will evaluate if all required input is available to actually calculate the processes. A report of this will be available in the lsp-report file.

All processes from D-Water Quality can be used in D-Flow FM. Processes are evaluated at larger time steps than flow and transport using fractional stepping. A time step for the evaluation of processes must be set in the mdu-file. Each DtProcesses, the processes are evaluated, which leads to fluxes. The effect is directly applied to the concentration field, after which the transport continues without any further interaction between the substances until the next water quality time step.

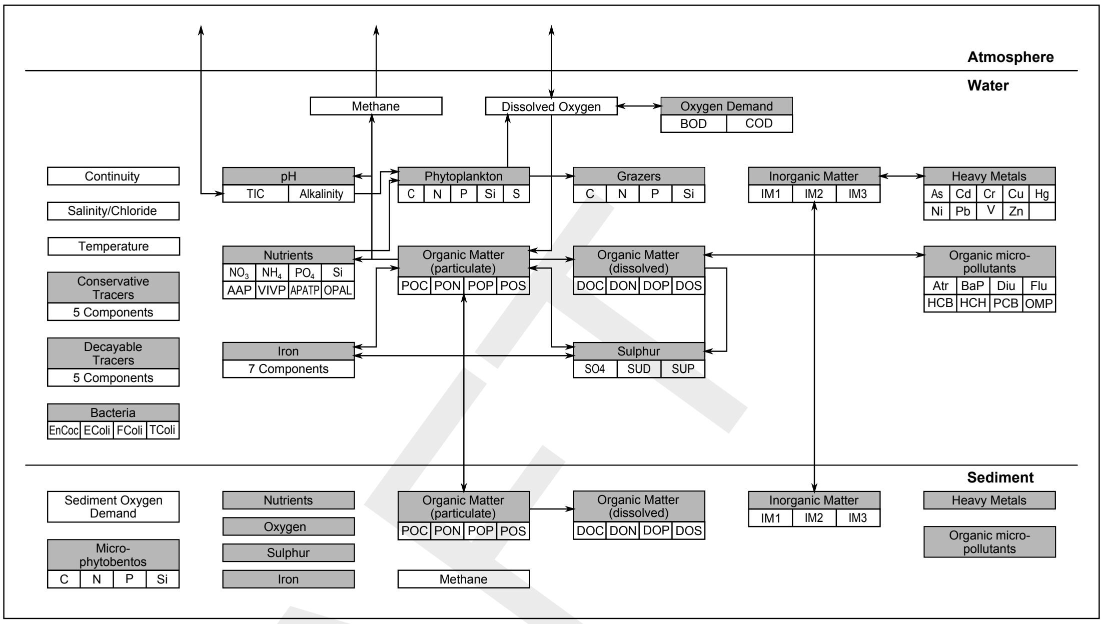


Figure 20.1: *General overview of substances included in D-Water Quality. Substances are organised in functional groups indicated by a grey header, except for some substances that form a group of their own. Major links between substances are indicated by arrows; note that many links are omitted.*

Limitation: `DtProcesses` must be a multiple of the `dtuser`, and can be specified in the processes section of the mdu-file using the MDU keyword `[processes]`, `DtProcesses`. If DtProcesses is negative, water quality processes are calculated with every hydrodynamic time step. A list of the processes to be switched on is given in the sub-file.

Example of a processes time step definition in the mdu-file:

```
[processes]
DtProcesses    =    600.0
```

Format for a processes list in the sub-file:

```
active-processes
   name <name 1> <decription 1>
   name <name 2> <decription 2>
         :
   name <name n> <decription n>
end-active-processes
```

Example of a processes list in the sub-file

```
active-processes
   name  'Compos' 'Composition'
   name  'Nitrif_NH4' 'Nitrification of ammonium'
   name  'DecFast' 'Mineralization fast decomp. detritus POC1'
         :
   name  'Daylength' 'Daylength calculation'
end-active-processes
```

20.2.3 Dry segments

When cells only contain a limited amount of water, or no water at all, some processes can behave peculiar, and cause negative concentrations or instabilities. To prevent this, processes receive a signal when segments are dry. Several processes consciously ignore the dry label and will always do calculations in dry cells because they deal with it properly, or e.g. are for processes in the sediment that continue in dry cells.

By default, segments are considered dry when the volume drops below 0.001m^3 or the depth of the layer drops below 0.001m. The same defaults are used in D-Water Quality. The user can adjust these settings by using the following keywords in the [processes] section of the mdu-file:

```
[processes]
VolumeDryThreshold = 1.0e-3
DepthDryThreshold = 1.0e-3
```

20.3 Definition of processes parameters

In D-Water Quality there are four types of parameters, constant in time and space, varying in time, varying in space, and varying in both time and space. Currently, only the first three types are supported. In Table 20.2, you can see an overview of these types, with the D-Water Quality term for them in italics, and the way they can be specified in D-Flow FM.

Table 20.2: *Various types of parameter inputs for water quality models in D-Flow FM.*

	Constant in time	**Varying in time**
Constant in space	*constant* define a `parameter` in the sub-file (section 20.3.1)	*function* define a `waqfunction` in the ext-file (section 20.3.3)
Varying in space	*parameter* define a `waqparameter` in the ext-file (section 20.3.2)	*segment function* not supported yet (section 20.3.4)

20.3.1 Constant parameter input

While in D-Water Quality constants are part of 'block 7' in the input file, in D-Flow FM they are moved to the substance file, where (confusingly) they are called `parameter`. A constant value in the substance file can however be trumped by a spatial or temporal definition in the ext-file.

Format for a *constant* definition in the sub-file (in which they are called parameter):

```
parameter <parameter name>
   description        <decription>
   unit               <unit>
   value              <value>
end-substance
```

An example of a *constant* definition in the sub-file (in which they are called parameter):

```
parameter 'V0SedIM1'
   description   'sedimentation velocity IM1'
   unit          '(m/d)'
   value         7.20000E-00
end-parameter
parameter 'TaucRS1DM'
   description   'critical shear stress for resuspension DM layer S1'
   unit          '(N/m2)'
   value          0.2000E+00
end-parameter
```

20.3.2 Spatial parameter input

Spatially varying input for a process parameter is called a parameter in D-Water Quality. In D-Flow FM, these are defined in the ext-file using `QUANTITY=waqparameter<name>`. At the moment, data can only be sepecified in 2D, and is copied to all layers.

```
QUANTITY=waqparameterV0SedIM1
FILENAME=para1.pol
FILETYPE=10
METHOD=4
OPERAND=O
VALUE=3.6

QUANTITY=waqparameterV0SedIM1
FILENAME=para2.pol
FILETYPE=10
METHOD=4
OPERAND=O
VALUE=7.2
```

20.3.3 Temporal parameter input

Temporal varying input for a process parameter is called a function in D-Water Quality. In D-Flow FM, these are defined in the ext-file using `QUANTITY=waqfunction<name>`.

```
QUANTITY=waqfunctionV0SedIM1
FILENAME =V0SedIM1.tim
FILETYPE =1
METHOD   =1*
OPERAND  =O

QUANTITY=waqfunctionV0SedIM2
FILENAME =V0SedIM2.fun
FILETYPE =1
METHOD   =0*
OPERAND  =O
```

* file containing a time series for this parameter;
0 = block, 1 = linear interpolation

20.3.4 Spatial and temporal parameter input

Parameter input data that is both spatial and temporal varying is called a segment function in D-Water Quality. In an integrated model, you can specify external data sources, but also connect to hydrodynamic and meteorological data that is already available within D-Flow FM in your model.

Defining external data sources for segment functions

In D-Flow FM, you can specify segment functions in the ext-file using `QUANTITY=waqsegmentfunction<name>`. At the moment, only data specified on a curvilinear grid in a netCDF is supported (filetype=11). We hope to support temporal varying sample files in the near future. Data can only be sepecified in 2D, and is copied to all layers.

```
QUANTITY=waqsegmentfunctionIM1
FILENAME=f34_sediments.nc
VARNAME=IM1
FILETYPE=11
METHOD=3
OPERAND=O
```

Selecting internal data to make it available for the processes

Within D-Flow FM it is possible to connect several parameters to the water quality processes. If you mention the name of certain parameters in the sub-file, they will be replaced by data from D-Flow FM, when available. Table 20.3 shows which data can be connected by which parameters. Which data is actually connected is reported in the dia-file.

Table 20.3: *Data form D-Flow FM that is available for water quality processes*

D-Flow FM data	D-Water Quality name in sub-file
horizontal surface area	Surf (default)
bottom shear stress	Tau or TauFlow*
flow element center velocity	Velocity
salinity	Salinity
temperature	Temp
wind velocity magnitude	VWind
wind direction	WindDir
fetch length and fetch depth	Fetch and/or InitDepth**
solar radiation	RadSurf
rain (mm/day)	Rain (mm/h)

*The bottom shear stress can be connected to Tau or TauFlow. When you connect to TauFlow, you can use the CalTau to calculate a Tau for the water quality processes with an additional bottom shear stress. Please be aware that if the D-Flow FM switch jawaveSwartDelwaq has been set to anything other than zero or D-Waves is coupled, there is already wave bottom shear stress included in TauFlow. Adding extra wave bottom shear stress with CalTau would lead to a doubling of wave bottom shear stress.
**Fetch length and fetch depth are both connected when either Fetch and/or InitDepth is mentioned in the sub-file.

If do not want to use the available data from D-Flow FM but want to specify your own input, you must not mention the parameter in the sub-file. You can add spatial or temporal input with possibly with just one uniform number if you actually wanted to provide a constant value.

20.4 Initial conditions, boundary conditions and sources and sinks

Most substances in the model will have certain initial conditions, boundary conditions and sources and sinks (the equivalent of D-Water Quality waste loads). An integrated D-Flow FM and D-Water Quality model makes uses of D-Flow FM tracers, therefore you have to make use of the D-Flow FM methods to prescribe them (See also: section 9.3). The options will be briefly discussed here.

20.4.1 Initial condition

The initial conditions for transported water quality substances are handled in exactly the same manner as those for any other constituent, i.e. you can specify a horizontally spatially varying field in the usual way through the ext-file using `QUANTITY=initialtracer<name>`.

For non-transported water quality substances you have to use `QUANTITY=initialwaqbot<name>`.

You can also refer to a restart file in the mdu-file. Restart conditions are read from the restart file by substance name, not by order in the file.

```
QUANTITY=initialtracerIM1
FILENAME=initracer.pol
FILETYPE=10
METHOD=4
OPERAND=O
VALUE=5.0

QUANTITY=initialwaqbotIM1S1
FILENAME=initracer.pol
FILETYPE=10
METHOD=4
OPERAND=O
VALUE=100.0
```

20.4.2 Boundary conditions

All of the D-Flow FM options for constituents are also available for transported water quality substances. You can specify boundary conditions for all these substances at all open inflow boundaries in the ext-file using `QUANTITY=tracerbnd<name>`. When modelling in three dimensions you may choose to specify boundary concentrations that have a uniform, linear, or step distribution over the vertical. You may also choose to specify a “Thatcher-Harleman” return time to simulate the re-entry of material that flowed out of the model after the flow reverses direction. When no boundary concentration is prescribe the concentration is presumed to be zero.

```
QUANTITY=tracerbndIM1
FILENAME=leftIM1_sed.pli*
FILETYPE=9
METHOD=3
OPERAND=O

QUANTITY=tracerbndIM2
FILENAME=leftIM2_sed.pli*
FILETYPE=9
METHOD=3
OPERAND=O
```

* tim-file (time series) with the same name is read

20.4.3 Sources and sinks

Sources and sinks are the equivalent of waste loads in D-Water Quality. They may be provided using using `QUANTITY=discharge_salinity_temperature_sorsin` in the ext-file as follows:

```
QUANTITY=discharge_salinity_temperature_sorsin
FILENAME      =WWTP.pliz*
FILETYPE      =9
METHOD        =3
OPERAND       =O
AREA          =0
```

* tim-file (time series) with the same name is read

The tim-file simultaneously prescribes sources and sinks of

- water volume itself (i.e. discharge),
- salinity (when switched on in the model), and
- temperature (when switched on in the model), and.
- any other constituents that are transported.

See section 8.10 for more details.

The number of columns in the time file is equal to 2 (time and discharge) + 0/1/2 (depending on whether salinity and/or temperate are switched on) + <the number of constituents>. The diagnostic file contains a list of the actual order of the constituents. The non-transported substances are not part of the constituents, and do not need values in the tim-file.

Warning:
Be aware that the order in which you specify boundary conditions, initial conditions in the ext-file and the substances in the sub-file might influence the order of the constituents. This can lead to another interpretation of the tim-file than you would have expected, because the columns in the tim file cannot be labeled with names.

Warning:
Another, somewhat subtler, aspect of the current set-up of sources and sinks is that all data in the tim-file are interpolated linearly, so that the resulting waste load is *quadratically* dependent on time, rather than linearly. With large time steps between records in the tim-file and rapidly changes discharge rates and concentrations this may cause noticeable discrepancies between the mass you expect and what is actually put into the model. The easiest solution is probably to use block-wise constant concentrations and let the discharge rate vary linearly, for example (assuming a single substance and no salinity or temperature in the model):

```
1000.0   10.0     1.0
2000.0   20.0     1.0
2001.0   20.0     0.0     <-- Extra record to make the transition
3000.0   20.0     0.0
4000.0   20.0     0.0
4001.0   20.0     1.0     <-- Ditto
5000.0   10.0     1.0
```

Because D-Flow FM is a hydrodynamic model, the water discharge is always added to the model. Dry waste loads are not possible yet, but a workaround might be that you divided the discharges by a factor 1000 or 10^6, and multiply the concentrations by the same factor.

20.5 Output options

20.5.1 Map and history output

All output mention in the sub-file is written to both the map-output and his-output of D-Flow FM. This might not always be desirable. To distinguish between map and history output, it is possible to move a selection of outputs from the sub-file to a seperate file that must be mentioned in the mdu-file. The map-file will only contain outputs mention in the sub-file. The his-file will contain all outputs mention in both the sub-file and the AdditionalHistoryOutput-File. Use the MDU keyword `[processes]`, `AdditionalHistoryOutputFile`.

Example of this setting in the mdu-file:

```
[processes]
AdditionalHistoryOutputFile = addhisout.eho
```

Format for an output definition in the sub-file or eho-file:

```
output <output name>
   description        <decription>
end-output
```

Example of the output definition in the sub-file or eho-file:

```
output 'Tau'
   description   'total bottom shear stress'
end-output
output 'Surf'
   description   'Horizontal surface'
end-output
output 'TotalDepth'
   description   'Total depth of water column'
end-output
```

20.5.2 Mass balance areas

Define mass balance areas using polygons with `QUANTITY=waqmassbalancearea<name>` in the mdu-file. If there is any overlap in the polygons when using multiple mass balance areas, the last definition will overrule any previous definitions. If there are cells remaining that are not included in any mass balance area, they will be grouped in an extra mass balance area named *Remaining cells*. This will cut up the whole model area in non-overlapping mass balance areas.

D-Flow FM will produce mass balances for the defined areas, that gives an overview of the fluxes between the various mass balance area's, over the model boundaries, and the processes fluxes in each mass balance area. The mass balances will be written to text and binary files at a given interval, and at the end of the run. Use the MDU keyword `[output]`, `MbaInterval` to set the interval (this used to be `[processes]`, `DtMassBalance`).

It is possible to group four groups of terms in mass balance area output: Exchange between the areas, exchange over boundaries, Source/sinks and process fluxes. This will reduce the output in case you are not interested in all the details. Use the following mdu-keyword in the `[output]` section with (1: yes, 0: no):

- MbaLumpFromToMba to lump the from/to other areas
- MbaLumpBoundaries to lump the boundaries
- MbaLumpSourceSinks to lump the source/sinks
- MbaLumpProcesses to lump the processes

The mass balance areas output can also be written to csv-files, one for total mass, one for the mass balance terms. These files can easily be imported and used by post-processing tools. Use the MDU keyword `[output]`, `MbaWriteCsv` to turn this on when mass balance areas output is active.

Limitation: MbaInterval must be a multiple of the dtuser

Example of this setting in the mdu-file:

```
[output]
MbaInterval                     = 86400.0
MbaLumpFromToMba                = 0
MbaLumpBoundaries               = 0
MbaLumpSourceSinks              = 1
MbaLumpProcesses                = 0
MbaWriteCsv                     = 1
```

Example of the mass balance area definition in the ext-file.

```
QUANTITY=waqmassbalanceareaBalArea1
FILENAME=barea1.pol
FILETYPE=10
METHOD=4
OPERAND=O
VALUE=1.0*

QUANTITY=waqmassbalanceareaBalArea2
FILENAME=barea2.pol
FILETYPE=10
METHOD=4
OPERAND=O
VALUE=1.0*
```

*Ignored

20.5.3 Statistical output

It is possible to have statistical output similar to stand alone D-Water Quality in an integrated run. You can specify an stt-file in the same format as used in D-Water Quality. Use the MDU keyword `[processes]`, `StatisticsFile` to provide the name. For a description of the stt-file please refer to chapter 10 of the D-Water Quality Input File Manual (Deltares, 2024b).

20.6 Running D-Flow FM with processes

When running D-Flow FM with processes, it needs to read a processes data base that contains input/output information of all the processes. The processes database is closely related to the code in the executable, and generally it is advised to use the process library that comes with the D-Flow FM executable. Using the open processes library that uses additional subroutines in a seperate dll-file (Windows) or so-file (Linux) is also supported. When using the BLOOM algae model, D-Flow FM also needs to read a BLOOM algae species file. All these options must be given as command line arguments in the following way:

```
> dflowfm-cli <mdu-file> --autostartstop --processlibrary <proc_def>
    --openprocessdllso <openprocessess dll/so> --bloomspecies <bloom.spe>
```

```
 --processlibrary PROCESSLIBRARYFILE
     Specify the process library file to be used for water quality processes.

 --openprocessdllso OPENPROCESSDLLSOFILE
     Specify the open process dll/so file with additional subroutines to be
     used for water quality processes.

 --bloomspecies BLOOMSPECIESFILE
     Specify the BLOOM species definition file to be used for water quality
     processes.
```

20.7 Output

The water quality output can be found in the D-Flow FM map and history files according to the user specifications (see: section 20.5).

The output for the mass balance areas is written into two text files named after the mdu-file, and appended with '_bal.txt' and '_baltot.txt'. They contain an overview mass balance areas, the active an inactive substances and the fluxes that affect the substances and thier stochiometry factor.

What follows for each mass balance area (and for each mass balance interval in the case of the _bal.txt-file), is a water and mass balance for each substance, followed by a water and mass balance for each substance for the whole area.

20.8 Known issues and limitations

The processes functionality has the following limitations: - although the sinks and source are working for tracers and thus water quality substances, it is a bit difficult to add concentrations to these sinks and sources. In the tim-file that accompanies the sinks en sources, besides the discharge the user has to specify the concentrations of constitutes in unnamed columns. But the order is determined by the order that constituents are defined in firstly the ext-file and then the sub-file (for substances that have no initial conditions or boundaries. Adding a substance means you'll immediately have to fix the tim-file. The dia-file now contains an actual list of the order of the constituents - there is a significant performance difference when substances are defined in a sub-file too (without any processes), compared to when they are only defined by boundary and initial conditions that should not be there.

And we would like to make the following improvements in the near future:

- ◇ Tim files are using time in minutes since reference date. We would like to see support for absolute dates too.

- ◇ In D-Water Quality, the user could specify multiple concentrations for boundary conditions in the same table, and use 'USEFOR' statements for flexible boundary definitions.
- ◇ The binary mass balance output should be written in a netCDF format
- ◇ Write (mass balances, and) average concentrations and other outputs for monitoring areas

20.9 Particle tracking

Note: Modelling of particle tracks is not possible with D-Flow FM. Currently, the DELFT3D-PART module is extended so that hydrodynamic model results on both structured grids and unstructured grids can be applied for particle tracking. This is currently α-functionality and is therefore not available yet.

21 Coupling with Nearfield data via COSUMO

D-Flow FM can be coupled to the external program called COSUMO, created by Deltares, for modelling subgrid processes. A typical example is a discharge via a diffuser. D-Flow FM delivers ambient data to COSUMO (flow velocities, concentrations) and COSUMO delivers discharge data to D-Flow FM. Contact Deltares for more information.

Restriction:

◇ When D-Flow FM runs integrated with COSUMO, D-Flow FM can not run in parallel yet.

21.1 Getting started

When D-Flow FM is coupled with COSUMO, DIMR must be used to run a simulation, managing the following two dlls/so's:

◇ D-Flow FM as a dll, named <dflowfm.dll>
◇ <cosumo_bmi.dll> which takes care of the communication with a running COSUMO instance

The following files are involved:

◇ <dimr_config.xml>, see section 21.1.2. This file connects D-Flow FM to cosumo_bmi and specifies the quantities to be exchanged between them.
◇ <COSUMOsettings.xml>, see section 21.1.3, configures COSUMO and is identical to the one used in Delft3D-FLOW.
◇ <FF2NF.xml>, see section 21.1.4, contains FarFieldToNearField data, is created on the fly by cosumo_bmi, is read by COSUMO and is identical to the one used in Delft3D-FLOW.
◇ <NF2FF.xml>, see section 21.1.5, contains NearFieldToFarField data, is created on the fly by COSUMO, is read by cosumo_bmi and is identical to the one used in Delft3D-FLOW.

The mdu-file may contain the following additional, optional keywords:

Keyword	Value	Description	Default
`NFEntrainmentMomentum`	1 I	In block [physics], 1: Momentum transfer in NearField related entrainment	0
`Wrimap_NearField`	1 I	In block [output], write near field parameters (1: yes, 0: no)	0

21.1.1 Run procedure

Obtain the COSUMO program from Deltares. Currently, only a Windows version is available. The exchange between COSUMO and D-Flow FM is via files. If D-Flow FM runs on Linux, a file system is needed that supports both reading/writing on Linux and on Windows.

A typical COSUMO instance checks a specified folder, e.g. FF2NF (FarFieldToNearField), for files to appear, placed there by cosumo_bmi (using data from D-Flow FM, obtained via DIMR), to be processed by COSUMO. When COSUMO has processed the FF2NF-file, it will produce a NF2FF-file. COSUMO will place this resulting file in a folder, e.g. NF2FF (NearFieldToFarField). cosumo_bmi checks this NF2FF-folder for files to appear. When they do, cosumo_bmi will read these files, transfer data to D-Flow FM via DIMR and the D-Flow FM computation will continue with updated data.

One COSUMO instance can serve multiple D-Flow FM computations at the same time. To distinct these files, D-Flow FM adds a uniqueId to the name of the file, consisting of two underscores with 6 capital letters, e.g. "_WGUPKA_". COSUMO will add this uniqueId also to the name of the resulting NF2FF-file.

When starting a simulation, be sure that COSUMO is running as a service. Then start DIMR, using a correct dimr_config file, referring to a correct COSUMO settings file.

Examples are available upon request.

21.1.2 Input DIMR

Both D-Flow FM and COSUMO are used as dynamic libraries (DLL's on Windows, so's on Linux). DIMR is a small executable steering both dynamic libraries. Its input file, usually called “dimr_config.xml”, looks like this when combining D-Flow FM with COSUMO:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<dimrConfig xmlns="http://schemas.deltares.nl/dimr"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://schemas.deltares.nl/dimr
                        http://content.oss.deltares.nl/schemas/dimr-1.0.xsd">
   <documentation>
      <fileVersion>1.00</fileVersion>
      <createdBy>Deltares, Coupling team</createdBy>
      <creationDate>2015-05-20T07:56:32+01:00</creationDate>
   </documentation>

   <control>
      <parallel>
         <startGroup>
            <time>0 1800 21600</time>
            <coupler name="flow2cosumo"/>
            <start name="COSUMO"/>
            <coupler name="cosumo2flow"/>
         </startGroup>
         <start name="FM"/>
      </parallel>
   </control>

   <component name="FM">
      <library>dflowfm</library>
      <workingDir>fm</workingDir>
      <inputFile>FlowFM.mdu</inputFile>
   </component>

   <component name="COSUMO">
      <library>cosumo_bmi</library>
      <workingDir>cosumo</workingDir>
      <inputFile>COSUMOsettings.xml</inputFile>
      <parameter key="skipUniqueID" value="yes"/>
   </component>

   <coupler name="flow2cosumo">
      <sourceComponent>FM</sourceComponent>
      <targetComponent>COSUMO</targetComponent>
      <item type="pointer">
         <sourceName>geometry/xcc</sourceName>
         <targetName>flow_xcc</targetName>
      </item>
      <item type="pointer">
         <sourceName>geometry/ycc</sourceName>
         <targetName>flow_ycc</targetName>
      </item>
      <item type="pointer">
         <sourceName>geometry/z_level</sourceName>
         <targetName>z_level_cc</targetName>
      </item>
      <item type="pointer">
         <sourceName>geometry/kbot</sourceName>
         <targetName>kbot</targetName>
      </item>
      <item type="pointer">
         <sourceName>geometry/ktop</sourceName>
         <targetName>ktop</targetName>
      </item>
      <item type="pointer">
         <sourceName>field/water_depth</sourceName>
         <targetName>water_depth_cc</targetName>
      </item>
      <item type="pointer">
         <sourceName>field/velocity_x</sourceName>
         <targetName>velocity_x_cc</targetName>
      </item>
      <item type="pointer">
         <sourceName>field/velocity_y</sourceName>
         <targetName>velocity_y_cc</targetName>
      </item>
      <item type="pointer">
         <sourceName>field/rho</sourceName>
         <targetName>rho_cc</targetName>
      </item>
      <item type="pointer">
         <sourceName>field/constituents</sourceName>
         <targetName>constituents</targetName>
      </item>
      <item>
         <sourceName>isalt</sourceName>
         <targetName>isalt</targetName>
      </item>
      <item>
         <sourceName>itemp</sourceName>
         <targetName>itemp</targetName>
      </item>
      <item type="pointer">
         <sourceName>runid</sourceName>
         <targetName>runid</targetName>
      </item>
      <item type="pointer">
         <sourceName>constituents_names</sourceName>
         <targetName>constituents_names</targetName>
      </item>
   </coupler>
   <coupler name="cosumo2flow">
      <sourceComponent>COSUMO</sourceComponent>
      <targetComponent>FM</targetComponent>
      <item type="pointer">
         <sourceName>nf_q_source</sourceName>
         <targetName>sourcesinks/COSUMO/nf_q_source</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_q_intake</sourceName>
         <targetName>sourcesinks/COSUMO/nf_q_intake</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_const</sourceName>
         <targetName>sourcesinks/COSUMO/nf_const</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_intake</sourceName>
         <targetName>sourcesinks/COSUMO/nf_intake</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_sink</sourceName>
         <targetName>sourcesinks/COSUMO/nf_sink</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_sour</sourceName>
         <targetName>sourcesinks/COSUMO/nf_sour</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_const_operator</sourceName>
         <targetName>sourcesinks/COSUMO/nf_const_operator</targetName>
      </item>
      <item type="pointer">
         <sourceName>nf_src_mom</sourceName>
         <targetName>sourcesinks/COSUMO/nf_src_mom</targetName>
      </item>
      <logger>
        <workingDir>.</workingDir>
        <outputFile>cosumo_to_dflowfm.nc</outputFile>
      </logger>
   </coupler>
</dimrConfig>
```

Description:

<control> Specifies the workflow of the deltaresHydro executable. It indicates which components are started in which order. If the data transfer is to be arranged by the main program DIMR, then a coupler should be included. The main <control> block is a sequential block; this means that each component is initialized, time stepped, and finalized before the next component starts. For each component/coupler listed inside the <control> block there will be a corresponding component/coupler specification block defined below.

<parallel> Within a <parallel> tag the components are started concurrently (if the mpi process id's listed per component don't overlap) or executed synchronously in sequence (first all initialize, then time stepping, and to conclude all finalization calls). The order of the components is retained.

<start> A <parallel> block contains exactly one <start> component, defining the start and end time of the simulation. This is the component inside the <parallel> block with the smallest time step and can be denoted as the "master-component". All other components must be defined with a <startGroup> and can be denoted as a "slave-component".

<startGroup> A <startGroup> should be used if a component (possibly including couplers) should only be executed at a subset of simulation time steps.

<time> Start-, step- and stop-time (in seconds) at which this slave-component should be executed. The times are relative to the times of the master-component. Thus a start-time of 0.0 always refers to the start time of the master-component and a stop-time of ”infinity” always refers to the end time of the master-component.

<component name="myComponentName"> Component specification block. ”myComponentName” is free to be defined by the user. It must match exactly with the reference in the <control> block above. The name of a component must be unique.

<library> Reference to the component to be executed. Currently ”flowfm”, ”wave” and ”FBCTools_BMI” are supported. The name must match exactly the name of the related dll/so (excluding prefixes (e.g. “lib”) and suffixes (e.g. “.dll” or “.so”)). The library should be located in the search path, or it may include an absolute or relative path. Multiple <component> blocks may refer to the same component to be executed.

<process> Optional list of the ids of the mpi processes that should be used to run the component. If not specified, then the component will run only in process “0” (i.e. non-parallel). The processes may be specified as a space separated list with series compressed using colons e.g. “16:31” represents processes “16 17 18” up to “31”.

<mpiCommunicator> D-Flow FM specific flag. Mandatory only when this D-Flow FM component should run a parallel (partitioned) model. Note that this is *unrelated* to the <parallel> tag as introduced above.

<workingDir> Specification of the working directory of this <component>, relative to the location of the “dimr_config.xml” file. The workingDir is the base/root directory of all input and output files for the component. All other files will be located RELATIVE TO this folder. If not specified, then workingDir will be equal to the folder of the configuration file.

<inputFile> Specification of the input file of this <component>, relative to <workingDir>: D-Flow FM: mdu-file, D-Waves: mdw-file, D-RTC: .

<coupler name="myCouplerName"> Coupler specification block. ”myCouplerName” is free to be defined by the user. It must match exactly with the reference in the <control> block above. The name of a coupler must be unique.

<sourceComponent> Identifies which component provides the data.

<targetComponent> Identifies which component needs to receive the data. The coupler runs on the superset of the processes configured for the source and target components.

<item> For each quantity to be exchanged, the name in the source component and the name in the target component are specified. To support recursive components, a directory syntax is used with forward slashes. If a name includes a forward slash, then it needs to be escaped using a backward slash, like \/; if a name includes a backward slash, then it needs to be escaped with a backward slash, like \\.

<sourceName> Identifies which parameter will be sent at the source component side.

<targetName> Identifies which parameter will be received at the target component side.

<parameter> DIMR way to provide parameters in the form of key - value pairs. For the COSUMO coupling, only the key `skipUniqueID` is relevant.

21.1.3 COSUMO config file

Below is an example <COSUMOsettings.xml>.

Remarks:

- ◇ Multiple diffusors can be specified in one file.
- ◇ Tag "<FF2NFdir>" specifies the folder where cosumo_bmi will put the FarFieldToNearField file. COOSUMO will monitor that folder, waiting for a file to be processed. COSUMO will generate a NearFieldToFarField file and put it in the neighbouring folder named "NF2FF".
- ◇ See COSUMO manual for more details.

```
<?xml version="1.0" encoding="utf-8"?>
<COSUMO>
  <fileVersion>0.3</fileVersion>
  <settings>
    <general>
      <subGridModel>cormix</subGridModel>
      <ID>Diffusor_1</ID>
      <farFieldModel>Delft3D</farFieldModel>
      <preProcessFcn></preProcessFcn>
      <postProcessFcn>postProcessFcn_coupleAtVerticalMixingSpreadSinks</postProcessFcn>
      <cmxFile>/path/to/cormix_file.cmx</cmxFile>
    </general>
    <comm>
      <FF2NFdir>COSUMO\FF2NF\</FF2NFdir>
      <FFrundir>rundir</FFrundir>
    </comm>
    <data>
      <XYdiff>550.0 350.0</XYdiff>
      <XYambient>823.0 344.8</XYambient>
      <XYambient>465.8 793.2</XYambient>
      <XYambient>587.4 509.2</XYambient>
      <XYintake>567.0 821.3453</XYintake>
      <discharge>
        <M3s>10.0</M3s>
             Operator: "absolute" values or "excess" (dT,dS,d..) -->
        <constituentsOperator>excess</constituentsOperator>
        <constituents>10.0 0.0 0.0</constituents>
      </discharge>
      <D0>2.5</D0>
      <H0>3.2</H0>
      <Theta0>15.0</Theta0>
      <Sigma0>0.0</Sigma0>
    </data>
  </settings>
  <settings>
    <general>
      <subGridModel>cormix</subGridModel>
      <ID>Diffusor_3</ID>
      <farFieldModel>Delft3D</farFieldModel>
      <preProcessFcn></preProcessFcn>
      <postProcessFcn>postProcessFcn_coupleAtVerticalMixingSpreadSinks</postProcessFcn>
      <cmxFile>/path/to/cormix_file.cmx</cmxFile>
    </general>
    <comm>
        <FF2NFdir>COSUMO\FF2NF\</FF2NFdir>
      <FFrundir>rundir</FFrundir>
    </comm>
    <data>
      <XYdiff>1350.0 850.0</XYdiff>
      <XYambient>1689.34 832.789</XYambient>
      <XYambient>1308.27 712.603</XYambient>
      <XYintake>1365.65 349.8342</XYintake>
      <discharge>
      <M3s>10.0</M3s>
          Operator: "absolute" values or "excess" (dT,dS,d..) -->
      <constituentsOperator>excess</constituentsOperator>
      <constituents>10.0 0.0 0.0</constituents>
     </discharge>
     <D0>2.5</D0>
     <H0>3.2</H0>
     <Theta0>15.0</Theta0>
     <Sigma0>0.0</Sigma0>
   </data>
  </settings>
</COSUMO>
```

21.1.4 FF2NF.xml file

Below is an example <FF2NF.xml> file generated by cosumo_bmi, to be read by COSUMO.

Remarks:

- ◇ Settings from the <COSUMOsettings.xml> file are echoed in here.
- ◇ See COSUMO manual for more details.

```
<?xml version="1.0" encoding="utf-8"?>
<COSUMO>
   <fileVersion>0.3</fileVersion>
   <comm>
      <Filename>COSUMO\FF2NF\FF2NF_UPKWVF_2dis_001_SubMod002_40.000.xml</Filename>
      <waitForFile>COSUMO\NF2FF\NF2FF_UPKWVF_2dis_001_SubMod002_40.000.xml</waitForFile>
      <FFrundir>/folder/to/workingDirectory</FFrundir>
      <FFinputFile>runid.mdf</FFinputFile>
      <FFuniqueID>UPKWVF</FFuniqueID>
   </comm>
   <SubgridModel>
      <SubgridModelNr>2</SubgridModelNr>
      <TIME>0.40000000000000000E+02</TIME>
      <constituentsNames>
         Temperature
         tracer
      </constituentsNames>
      <cormix>
         <HA>0.10044436851301324E+02 0.10087161007500129E+02</HA>
         <HD>0.10087687181911859E+02 0.10087687181911859E+02</HD>
         <UA>0.16860016896497312E+01 0.16839284598102089E+01</UA>
         <UorS>U U</UorS>
         <RHOAM>1022.882 1022.891</RHOAM>
         <STYPE>- -</STYPE>
         <RHOAS>- -</RHOAS>
         <RHOAB>- -</RHOAB>
         <HINT>- -</HINT>
         <DROHJ>- -</DROHJ>
         <Q0>0.10000000000000000E+02 0.10000000000000000E+02</Q0>
         <RHO0>0.10203153693903616E+04 0.10203153693903616E+04</RHO0>
         <D0>0.25000000000000000E+01 0.25000000000000000E+01</D0>
         <PHI>0.016 359.946</PHI>
         <S1>-0.87687181911859632E-01 -0.87687181911859632E-01</S1>
         <h_dps>0.10000000000000000E+02 0.10000000000000000E+02</h_dps>
         <x_diff>0.13500000000000000E+04 0.13500000000000000E+04</x_diff>
         <y_diff>0.85000000000000000E+03 0.85000000000000000E+03</y_diff>
         <taua>0.35998429477578065E+03 0.54386433951378876E-01</taua>
      </cormix>
      <FFDiff>
         <XYZ>
             0.13500000000000000E+04 0.85000000000000000E+03 0.50438435909559298E+00
             0.13500000000000000E+04 0.85000000000000000E+03 0.15131530772867789E+01
             0.13500000000000000E+04 0.85000000000000000E+03 0.25219217954779647E+01
             0.13500000000000000E+04 0.85000000000000000E+03 0.35306905136691502E+01
```

```
          0.13500000000000000E+04 0.85000000000000000E+03 0.45394592318603362E+01
          0.13500000000000000E+04 0.85000000000000000E+03 0.55482279500515217E+01
          0.13500000000000000E+04 0.85000000000000000E+03 0.65569966682427072E+01
          0.13500000000000000E+04 0.85000000000000000E+03 0.75657653864338927E+01
          0.13500000000000000E+04 0.85000000000000000E+03 0.85745341046250783E+01
          0.13500000000000000E+04 0.85000000000000000E+03 0.95833028228162647E+01
      </XYZ>
      <waterLevel>
         -0.87687181911859632E-01
      </waterLevel>
      <XYvelocity>
         0.18634752863723187E+01 -0.65423130314603509E-04
         0.18440910975357934E+01 -0.75752086320958088E-04
         0.18205156937121281E+01 -0.88198565822350397E-04
         0.17912584190512835E+01 -0.10340229548004782E-03
         0.17553731372697654E+01 -0.12157407948827463E-03
         0.17113303098565760E+01 -0.14292104796713966E-03
         0.16562845706940919E+01 -0.16767438429836248E-03
         0.15846208832750814E+01 -0.19590817071806193E-03
         0.14835159394973303E+01 -0.22652941333490347E-03
         0.13143232417856638E+01 -0.25073039928090045E-03
      </XYvelocity>
      <rho>
         0.10228927346557822E+04
         0.10228925528287078E+04
         0.10228925573324485E+04
         0.10228926721853319E+04
         0.10228928369848645E+04
         0.10228929327129407E+04
         0.10228929247694515E+04
         0.10228927989582477E+04
         0.10228925577392596E+04
         0.10228923390834573E+04
      </rho>
      <constituents>
         0.14992404423794710E+02 0.99999999999999956E+00
         0.14993260801272069E+02 0.99999999999999956E+00
         0.14993239589741927E+02 0.99999999999999967E+00
         0.14992698653658509E+02 0.99999999999999956E+00
         0.14991922455257015E+02 0.99999999999999933E+00
         0.14991471568012786E+02 0.99999999999999911E+00
         0.14991508982855612E+02 0.99999999999999933E+00
         0.14992101561293641E+02 0.99999999999999911E+00
         0.14993237673759108E+02 0.99999999999999900E+00
         0.14994267466736535E+02 0.99999999999999900E+00
      </constituents>
   </FFDiff>
   <FFIntake>
      <XYZ>
          0.13656500000000001E+04 0.34983420000000001E+03 0.50432409160155689E+00
          0.13656500000000001E+04 0.34983420000000001E+03 0.15129722748046710E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.25216204580077846E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.35302686412108981E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.45389168244140121E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.55475650076171252E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.65562131908202392E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.75648613740233523E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.85735095572264655E+01
          0.13656500000000001E+04 0.34983420000000001E+03 0.95821577404295795E+01
      </XYZ>
      <waterLevel>
         -0.86481832031138614E-01
      </waterLevel>
      <XYvelocity>
         0.18818301897771481E+01 -0.41131954003087163E-04
         0.18620821293796870E+01 -0.51252479552073700E-04
```

```
          0.18380816955559764E+01 -0.63488977655451589E-04
          0.18083027580156026E+01 -0.78497113307481159E-04
          0.17717742093295803E+01 -0.96507535602107604E-04
          0.17269543380078383E+01 -0.11776490852935577E-03
          0.16710270758615824E+01 -0.14255875243078517E-03
          0.15984082611615054E+01 -0.17106797450662620E-03
          0.14962939551442496E+01 -0.20245260897418325E-03
          0.13259941391355654E+01 -0.22864590576560917E-03
       </XYvelocity>
       <rho>
          0.10228854330432104E+04
          0.10228854437812954E+04
          0.10228854562015631E+04
          0.10228854708553444E+04
          0.10228854878069217E+04
          0.10228855070823315E+04
          0.10228855286756180E+04
          0.10228855525062595E+04
          0.10228855781668173E+04
          0.10228856032910751E+04
       </rho>
       <constituents>
          0.15026768296882313E+02 0.99999999999999978E+00
          0.15026717798282352E+02 0.99999999999999978E+00
          0.15026659388645221E+02 0.99999999999999967E+00
          0.15026590475116326E+02 0.99999999999999989E+00
          0.15026510755293693E+02 0.10000000000000000E+01
          0.15026420106616042E+02 0.99999999999999978E+00
          0.15026318556960202E+02 0.99999999999999956E+00
          0.15026206484863748E+02 0.99999999999999978E+00
          0.15026085806309688E+02 0.10000000000000000E+01
          0.15025967649284215E+02 0.10000000000000000E+01
       </constituents>
    </FFIntake>
    <FFAmbient>
       <XYZ>
           0.16893399999999999E+04 0.83278899999999999E+03 0.50222184256506619E+00
           0.16893399999999999E+04 0.83278899999999999E+03 0.15066655276951988E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.25111092128253309E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.35155528979554629E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.45199965830855948E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.55244402682157272E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.65288839533458596E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.75333276384759920E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.85377713236061243E+01
           0.16893399999999999E+04 0.83278899999999999E+03 0.95422150087362567E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.50435805037500647E+00
           0.13082700000000000E+04 0.71260299999999995E+03 0.15130741511250194E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.25217902518750321E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.35305063526250446E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.45392224533750571E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.55479385541250696E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.65566546548750830E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.75653707556250955E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.85740868563751071E+01
           0.13082700000000000E+04 0.71260299999999995E+03 0.95828029571251214E+01
       </XYZ>
       <waterLevel>
          -0.44436851301324229E-01
          -0.87161007500128396E-01
       </waterLevel>
       <XYvelocity>
          0.18661537625494253E+01 -0.49366424296911108E-03
          0.18471012743117909E+01 -0.49009395919670366E-03
          0.18239800958890944E+01 -0.48552181171841280E-03
          0.17952748109803749E+01 -0.47968468355273462E-03
```

```
         0.17599596219659430E+01 -0.47254837584717526E-03
         0.17163722300563427E+01 -0.46418653024982002E-03
         0.16614795392077935E+01 -0.45474119036623513E-03
         0.15893375561451455E+01 -0.44419780292137806E-03
         0.14865823068086272E+01 -0.43119016983203832E-03
         0.13137750651930324E+01 -0.40563453546634822E-03
         0.18656908317901597E+01 0.18170783410017626E-02
         0.18462618374506266E+01 0.17982146905653733E-02
         0.18226279443100606E+01 0.17750156766796000E-02
         0.17932667921863070E+01 0.17451632889557401E-02
         0.17571938867882322E+01 0.17063017989827256E-02
         0.17128557987268032E+01 0.16547532684783187E-02
         0.16574336733219286E+01 0.15843060342165156E-02
         0.15853619219655999E+01 0.14838805756680281E-02
         0.14839126603310422E+01 0.13324351571148145E-02
         0.13146716649400929E+01 0.10870739127617247E-02
      </XYvelocity>
      <rho>
         0.10228822123610827E+04
         0.10228821429023594E+04
         0.10228819887936504E+04
         0.10228817850054177E+04
         0.10228815897850097E+04
         0.10228814566462096E+04
         0.10228814297320561E+04
         0.10228815246915606E+04
         0.10228817189527225E+04
         0.10228819512385468E+04
         0.10228912677595583E+04
         0.10228912655445724E+04
         0.10228912692015710E+04
         0.10228912765903282E+04
         0.10228912848435131E+04
         0.10228912904465891E+04
         0.10228912911869041E+04
         0.10228912861229883E+04
         0.10228912758342422E+04
         0.10228912642271267E+04
      </rho>
      <constituents>
         0.15041909278602565E+02 0.10000000000000007E+01
         0.15042235704076928E+02 0.10000000000000007E+01
         0.15042959930419553E+02 0.10000000000000009E+01
         0.15043917587557827E+02 0.10000000000000009E+01
         0.15044834944004041E+02 0.10000000000000007E+01
         0.15045460552575548E+02 0.10000000000000002E+01
         0.15045587017923388E+02 0.10000000000000000E+01
         0.15045140815082185E+02 0.10000000000000000E+01
         0.15044227978676414E+02 0.10000000000000002E+01
         0.15043136415270279E+02 0.10000000000000007E+01
         0.14999312349324843E+02 0.99999999999999933E+00
         0.14999322778559403E+02 0.99999999999999978E+00
         0.14999305559622911E+02 0.99999999999999944E+00
         0.14999270769702697E+02 0.99999999999999911E+00
         0.14999231909566548E+02 0.99999999999999889E+00
         0.14999205527433457E+02 0.99999999999999889E+00
         0.14999202041651518E+02 0.99999999999999900E+00
         0.14999225885148908E+02 0.99999999999999911E+00
         0.14999274329731929E+02 0.99999999999999900E+00
         0.14999328981733916E+02 0.99999999999999889E+00
      </constituents>
   </FFAmbient>
</SubgridModel>
<settings>
   <general>
      <subgridmodel>cormix</subgridmodel>
```

```
         <id>Diffusor_1</id>
         <farfieldmodel>Delft3D</farfieldmodel>
         <preprocessfcn></preprocessfcn>
         <postprocessfcn>postProcessFcn_coupleAtVerticalMixingSpreadSinks</postprocessfcn>
         <cmxfile>/path/to/cormix_file.cmx</cmxfile>
      </general>
      <comm>
         <ff2nfdir>COSUMO\FF2NF\</ff2nfdir>
         <ffrundir>rundir</ffrundir>
      </comm>
      <data>
         <xydiff>550.0 350.0</xydiff>
         <xyambient>823.0 344.8</xyambient>
         <xyambient>465.8 793.2</xyambient>
         <xyambient>587.4 509.2</xyambient>
         <xyintake>567.0 821.3453</xyintake>
         <discharge>
            <m3s>10.0</m3s>
            <constituentsoperator>excess</constituentsoperator>
            <constituents>10.0 0.0 0.0</constituents>
         </discharge>
         <d0>2.5</d0>
         <h0>3.2</h0>
         <theta0>15.0</theta0>
         <sigma0>0.0</sigma0>
      </data>
   </settings>
   <settings>
      <general>
         <subgridmodel>cormix</subgridmodel>
         <id>Diffusor_3</id>
         <farfieldmodel>Delft3D</farfieldmodel>
         <preprocessfcn></preprocessfcn>
         <postprocessfcn>postProcessFcn_coupleAtVerticalMixingSpreadSinks</postprocessfcn>
         <cmxfile>/path/to/cormix_file.cmx</cmxfile>
      </general>
      <comm>
         <ff2nfdir>COSUMO\FF2NF\</ff2nfdir>
         <ffrundir>rundir</ffrundir>
      </comm>
      <data>
         <xydiff>1350.0 850.0</xydiff>
         <xyambient>1689.34 832.789</xyambient>
         <xyambient>1308.27 712.603</xyambient>
         <xyintake>1365.65 349.8342</xyintake>
         <discharge>
            <m3s>10.0</m3s>
            <constituentsoperator>excess</constituentsoperator>
            <constituents>10.0 0.0 0.0</constituents>
         </discharge>
         <d0>2.5</d0>
         <h0>3.2</h0>
         <theta0>15.0</theta0>
         <sigma0>0.0</sigma0>
      </data>
   </settings>
</COSUMO>
```

21.1.5 NF2FF.xml file

Below is an example <NF2FF.xml> file generated by COSUMO, to be read by cosumo_bmi.

Remark:

◇ See COSUMO manual for more details.

```
<?xml version="1.0" encoding="utf-8"?>
<NF2FF>
  <fileVersion>0.3</fileVersion>
  <discharge>
   <M3s>10.0</M3s>
       Operator: "absolute" values or "excess" (dT,dS,d..) -->
   <constituentsOperator>excess</constituentsOperator>
   <constituents>10.0 0.0 0.0</constituents>
  </discharge>
  <velocity>
       COSUMO should produce a file that ONLY contains a discharge block OR a velocity block -
  </velocity>
  <NFResult>
   <sinks>
  550.000 350.087 9.700 1.000 0.000 0.000
  552.500 350.048 9.700 1.600 0.250 0.380
  555.000 350.008 9.700 1.900 0.500 0.750
  557.500 350.958 9.700 2.100 0.750 1.120
  560.000 350.919 9.700 2.300 1.000 1.500
  562.500 350.869 9.700 2.400 1.250 1.880
  565.000 350.830 9.570 2.600 1.500 2.250
  567.500 350.790 9.220 2.700 1.750 2.620
  570.000 350.741 8.860 2.800 2.000 3.000
  572.500 350.701 8.510 2.900 2.250 3.380
  575.000 350.662 8.150 3.000 2.500 3.750
  577.500 350.612 7.800 3.100 2.750 4.120
  580.000 350.573 7.440 3.200 3.000 4.500
  582.500 350.533 7.090 3.300 3.250 4.880
  585.000 350.483 6.730 3.400 3.500 5.250
  587.500 350.444 6.370 3.500 3.750 5.620
  590.000 350.394 6.020 3.600 4.000 6.000
  592.500 350.355 5.660 3.600 4.250 6.380
  595.000 350.315 5.310 3.700 4.500 6.750
  597.500 350.266 4.950 3.800 4.750 7.130
  600.000 350.226 4.600 3.900 5.000 7.500
  602.500 350.187 4.240 3.900 5.250 7.880
  605.000 350.137 3.890 4.000 5.500 8.250
  607.500 350.097 3.530 4.100 5.750 8.620
  610.000 350.058 3.180 4.100 6.000 9.000
  612.500 350.008 2.820 4.200 6.250 9.380
  615.000 350.969 2.470 4.300 6.500 9.750
  617.500 350.919 2.120 4.300 6.750 10.120
  620.000 350.880 1.760 4.400 7.000 10.500
  622.500 350.840 1.410 4.400 7.250 10.880
  625.000 350.791 1.050 4.500 7.500 11.250
  627.500 350.751 0.950 4.600 7.500 11.620
  630.000 350.711 0.840 4.600 7.500 12.000
  632.500 350.662 0.740 4.700 7.500 12.380
  635.000 350.622 0.630 4.700 7.500 12.750
  637.500 350.583 0.530 4.800 7.500 13.130
  640.000 350.533 0.420 4.800 7.500 13.500
  642.500 350.494 0.320 4.900 7.500 13.880
  645.000 350.444 0.210 4.900 7.500 14.250
  647.500 350.405 0.110 5.000 7.500 14.630
      </sinks>
   <sources>
  650.000 350.365 -0.000 5.000 7.500 15.000
   </sources>
  </NFResult>
</NF2FF>
```

22 Calibration and data assimilation

Note: Calibration of D-Flow FM with OpenDA is a β-functionality.

22.1 Introduction

A flow model in D-Flow FM will generally benefit from parameter *calibration* to closer match observation data. When the model runs in an operational system *data assimilation* can be used to into the running model. For the automatic calibration and data assimilation, the open source toolbox **OpenDA** is available.

OpenDA basically provides three types of building blocks: an optimisation algorithm that performs the calibration or data assimilation, communication routines for passing information between OpenDA and D-Flow FM, and methods for handling observation data. The communication between OpenDA and D-Flow FM is realised using a Black Box approach. A number of wrapper objects (so called dataObjects) are available for reading and writing D-Flow FM input and output files.

This chapter contains a description of how the OpenDA toolbox could be deployed to apply calibration and data assimilation.

The OpenDA tools can run D-Flow FM models and analyze the model results. More information on OpenDA can be found on the website http://www.openda.org.

General information on the installation of OpenDA to get started is provided in section 22.2. section 22.4 gives an overview of the black box wrapper for D-Flow FM. section 22.4 describes the OpenDA configuration and the related D-Flow FM files. The generation of noise and how this noise is added to forcings and boundaries is given in section 22.5. Examples for calibration and data assimilation using the ensemble Kalman filter are described in section 22.6.

22.2 Getting started with OpenDA

The required D-Flow FM wrapper is enclosed in the official OpenDA release since version 3.0.3. The following three elements are needed for a calibration or data assimilation run with D-Flow FM:

1 The D-Flow FM Command Line Interface (CLI) installation. All OpenDA algorithms start D-Flow FM with a shell script `start_dimr.sh` or a batch script `start_dimr.bat`. The `start_dimr` scripts are positioned in the .
`bin` directory of the various examples (see the section 22.6 section.). They specify the path to the `run_dimr`) scripts that actually perform the computation, and are part of the Delft3D Flexible Mesh Suite installation. The name `_dimr` refers to the Deltares Integrated Model Runner, the main application that starts a D-Flow FM run.

2 An OpenDA installation including the OpenDA core, the D-Flow FM specific wrapper code and examples.

3 A Java Runtime Environment (JRE) version 8 (or higher) is needed to run OpenDA (3.0.3). OpenDA can use one of the system installed JREs or alternatively an JRE can be installed directly in the OpenDA directory at the same level as the <bin> directory.

For Linux it is required to run `source settings_local.sh` to setup OpenDA. The OpenDA GUI can be started from the <bin> directory using `oda_run_gui.bat` (Windows) or `oda_run.sh` with the `-gui` command line option (Linux).

22.3 The OpenDA black box model wrapper for D-Flow FM

For a D-Flow FM model to function within the OpenDA toolbox we need to establish two things:

1 the control to propagate the model over time and
2 access to the model state, physical parameters, boundary conditions and external forcings.

In the black box approach the standard D-Flow FM command line executable is used to propagate the model over time. Access to the model state, physical parameters, boundary conditions and external forcings is obtained by reading from and writing to the D-Flow FM input and output files. Inside OpenDA all data is available in the form of exchange items which all have an unique identifier.

Calibration and data assimilation typically need multiple model evaluations with altered parameters, forcings, boundary conditions or the initial model state. In the black box approach this is achieved by creating multiple work directories containing altered model input files and starting the D-Flow FM executable in each work directory. D-Flow FM model results are than read from the work directories and compared to observation data.

For calibration this is an iterative process. Results from model evaluations $1, \dots, n$ are required to obtain a better estimate for parameter values, which are then evaluated in run $n+1$.

In case of an ensemble Kalman filter (EnKF) run, the D-Flow FM computations (one run for each ensemble member) is stopped each time observations are available. The input files for each ensemble member are modified according to the ensemble Kalman filter algorithm, after which the D-Flow FM run is restarted.

Next to the D-Flow FM model configuration OpenDA has its own configuration for selecting the algorithm, observations and interfacing with the D-Flow FM input an output files.

22.4 OpenDA configuration

22.4.1 Main configuration file and the directory structure

The OpenDA main configuration file has the `.oda` extension. All OpenDA configuration files use the xml format. It is advised to use a schema aware xml editor, when making changes to the OpenDA configuration. These editors provide direct access to the documentation that is stored in the schema and can validate the correctness of the xml files.

All other xml config file names and directories are configurable. However, there is a commonly used directory layout and naming convention. All the examples are configured using this convention. For example, the directory structure for the simple_waal_kalman example is given in Table 22.1.

All the work directories are available in the <stochModel> directory, after performing a run with OpenDA they contain the D-Flow FM results.

It is a good practice to name the main configuration file corresponding to the algorithms executed by OpenDA. The following files are used in the provided examples:

- ◇ <Simulation.oda>: performs a regular D-Flow FM simulation run, only the executable is started by OpenDA. This algorithm is useful to check the configuration.
- ◇ <Dud.oda>: Dud (Doesn't Use Derivative) is one of the optimisation algorithms available for calibration purposes.

```
<simple_waal_kalman>
├─ <algorithm> ................. calibration method or data-assimilation algorithm
│  ├─ <enkfAlgorithm.xml>....................algorithm specific configuration
│  └─ ...
├─ <stochModel>..........................model and its uncertainty description
│  ├─ <bin> ......................... .bat and .sh scripts for calling D-Flow FM
│  ├─ <input_dflowfm>.....................D-Flow FM template configuration
│  │  └─ ...
│  ├─ <work0/> ............................ work directory for propagated mean
│  ├─ <work1/>........................work directory for first ensemble member
│  ├─ ...
│  ├─ <dflowfmModel.xml>......................exchange items configuration
│  ├─ <dflowfmStochModel.xml> ............... predictor, state and parameter
│  ├─ <dflowfmWrapper.xml>............actions and data objects configuration
│  └─ ...
├─ <stochObserver>............................observations and uncertainty
│  ├─ <noosObservations.xml> ................. observations and uncertainty
│  ├─ <waterlevel_Obs01.noos> .........................raw observation file
│  └─ ...
├─ <Enkf.oda> ...................................... main configuration file
└─ ...
```

Table 22.1: *Directory structure of the OpenDA Ensemble Kalman filtering configuration for the simple Waal D-Flow FM model.*

- ◇ <SequentialSimulation.oda>: performs a D-Flow FM simulation run through which the executable is stopped and restarted by OpenDA at the moments for which observed data are available.
- ◇ <Enkf.oda>: performs an ensemble Kalman filtering.

The main configuration for an OpenDA application has three mandatory components, which make up an OpenDA application: `stochModelFactory`, `stochObserver` and `algorithm`. Each component is configured by specifying its `className` attribute, `workingDirectory` and `configFile/configString`. There are optional components to enable writing OpenDA results, to define OpenDA restart input and output files and to enable timings. The `resultwriter` is typically useful for writing stochastic properties that are only available to OpenDA and not in D-Flow FM.

The main configuration file <Enkf.oda> for the `estuary_kalman_FMSuite2019.01_bcfile` example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<openDaApplication xmlns="http://www.openda.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.openda.org/openDaApplication.xsd">
   <stochObserver className="org.openda.observers.NoosTimeSeriesStochObserver">
      <workingDirectory>./stochObserver</workingDirectory>
      <configFile>noosObservations.xml</configFile>
   </stochObserver>
   <stochModelFactory className="org.openda.blackbox.wrapper.BBStochModelFactory">
      <workingDirectory>./stochModel</workingDirectory>
      <configFile>dflowfmStochModel.xml</configFile>
   </stochModelFactory>
   <algorithm className="org.openda.algorithms.kalmanFilter.EnKF">
      <workingDirectory>./algorithm</workingDirectory>
      <configString>EnkfAlgorithm.xml</configString>
   </algorithm>
   <timingSettings doTiming="true"></timingSettings>
   <resultWriter className="org.openda.resultwriters.PythonResultWriter">
      <workingDirectory>.</workingDirectory>
      <configFile>Enkf_results.py</configFile>
      <selection>
         <resultItem id="pred_f"/>
         <resultItem id="pred_f_0"/>
         <resultItem id="pred_f_1"/>
         <resultItem id="pred_a"/>
         <resultItem id="pred_a_0"/>
         <resultItem id="pred_a_1"/>
         <resultItem id="pred_f_std"/>
         <resultItem id="pred_f_central"/>
         <resultItem id="pred_a_central"/>
         <resultItem id="pred_a_linear"/>
         <resultItem id="analysis_time"/>
         <resultItem id="obs"/>
      </selection>
   </resultWriter>
</openDaApplication>
```

Note that the directory layout in this section is created by setting the workingDirectory elements in stochModelFactory, stochObserver and algorithm parts of the configuration. Each of these components have their own configuration, which are described in the following sections.

22.4.2 The algorithm configuration

All provided methods for calibration and data assimilation are configurable through an xml file. The convention is to include the algorithm name in the file name e.g <EnkfAlgorithm.xml>. For data assimilation algorithms the configuration typically specifies the ensemble size and the option to use stochastic parameters, forcing and initialisation. For calibration algorithms the configuration typically contains definition of the cost function and tolerances and stopping criteria. For a list of algorithms and their configuration options see the general OpenDA documentation.

22.4.3 The stochObserver configuration

The access to observations is standardized in OpenDA using a stochObserver object. The configuration and the observation data are placed in the <stochObserver> directory. OpenDA contains a number of different stochObserver objects that can handle different types of data files. In this manual, we discuss the NoosTimeSeriesStochObserver and the more generic IoObjectStochObserver. For a more complete list of available stochObservers see the OpenDA web site.

22.4.3.1 NoosTimeSeriesStochObserver

The NOOS file format is used to store time series. The format is created for use by the members of the North West European Shelf Operational Oceanographic System http://www.noos.cc/. An example of (a part of) a NOOS file is:

```
#------------------------------------------------------
#------------------------------------------------------
# Location : station01
# Position : (0.0,0.0)
# Source : twin experiment DFlowFM
# Unit : waterlevel
# Analyse time: null
# Timezone : null
```

```
#------------------------------------------------------------
199101010000   1.0000
199101010100   0.8944
199101010200   0.6862
199101010300   0.5956
199101010400   0.3794
199101010500   0.1372
199101010600   -0.1300
199101010700   -0.3044
199101010800   -0.3963
199101010900   -0.3739
199101011000   -0.1930
...
```

The file contains a header with meta data specifying among others the location name (Location) and the quantity (Unit). The data is written in two columns, the first gives the time of the observation using the ‘YYYYMMDDhhmm’ format and the second column gives the measured value.

The file <noosObserver.xml> defines a number of time series, each coupled to a NOOS file containing the measurements.

```
<?xml version="1.0" encoding="UTF-8"?>
<noosObserver xmlns="http://www.openda.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.openda.org/observation/noosObservations.xsd">
    <timeSeries status="use" standardDeviation="0.005"
       minDateTime="199101010000" maxDateTime="199101030000">
      waterlevel_station01.noos
    </timeSeries>
    <timeSeries status="use" standardDeviation="0.005"
       minDateTime="199101010000" maxDateTime="199101030000">
      waterlevel_station02.noos
    </timeSeries>
    <timeSeries status="use" standardDeviation="0.005"
       minDateTime="199101010000" maxDateTime="199101030000">
      waterlevel_station03.noos
    </timeSeries>
</noosObserver>
```

OpenDA creates an exchange item for each observation time series. The default exchange item id (identifier) is created using the location (Location) and the quantity (Unit). The standard deviation (measurement error) is specified with standDeviation attribute.

22.4.3.2 IoObjectStochObserver

This observer uses a dataObject for accessing the file(s) containing the measurements. All exchange items that are provided by a specific dataObject can be used by the observer. For instance the NetcdfDataObject can read and write to netCDF files, and has an exchange item for each variable in the netCDF file. The lake_kalman example uses this approach to use the *.his file from a D-Flow FM run as synthetic observations for a ensemble Kalman filter run. The <dflowfmStochObsConfig.xml> file reads:

```
<?xml version="1.0" encoding="UTF-8"?>
<ioObjectStochObserver
   xmlns="http://www.openda.org"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation
     ="http://www.openda.org http://schemas.openda.org/openDaStochObserver.xsd">
 <uncertaintyModule
     workingDirectory="."
```

```
    className="org.openda.uncertainties.UncertaintyEngine">
  <arg>stochObsUncertainties.xml</arg>
 </uncertaintyModule>
 <ioObject
    workingDirectory="."
    className="org.openda.exchange.dataobjects.NetcdfDataObject">
  <fileName>lake2d_his.nc</fileName>
 </ioObject>
</ioObjectStochObserver>
```

In the configuration of `UncertaintyEngine` OpenDA object, a selection of these exchange items `id`'s is made and a standard deviation is specified.

```
<?xml version="1.0" encoding="UTF-8"?>
<uncertainties
   xmlns="http://www.wldelft.nl"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation
     ="http://www.wldelft.nl http://schemas.openda.org/uncertainties.xsd"
   version="1.0">
   <uncertaintyType>ProbabilityDistributionFunction</uncertaintyType>
   <probabilityDistributionFunction id="S1.waterlevel" isActive="true">
      <normal mean="0" stdv="0.05" stdvIsFactor="false"/>
   </probabilityDistributionFunction>
   <probabilityDistributionFunction id="S2.waterlevel" isActive="true">
      <normal mean="0" stdv="0.05" stdvIsFactor="false"/>
   </probabilityDistributionFunction>
   <probabilityDistributionFunction id="S3.waterlevel" isActive="true">
      <normal mean="0" stdv="0.05" stdvIsFactor="false"/>
   </probabilityDistributionFunction>
   <probabilityDistributionFunction id="S4.waterlevel" isActive="true">
      <normal mean="0" stdv="0.05" stdvIsFactor="false"/>
   </probabilityDistributionFunction>
</uncertainties>
```

22.4.4 The `stochModel` configuration

The `stochModel` configuration usually consists of three <∗.xml> files in the <stochModel> directory. These are called:

- <dflowfmWrapper.xml>: This file specifies the actions to perform in order to run a D-Flow FM simulation and list the D-Flow FM input and output files that can be used to let OpenDA interact with the model. For each file the dataObject is specified which is used for handling the specified file.
- <dflowfmModel.xml>: This file contains a list of the exchange items which are provided by the configured dataObject. The model time information is constructed by the `timeInfoExchangeItems` element. It also contains a number of alias values which can be used in three stochModel configuration files.
- <dflowfmStochModel.xml>: In this file defines the predictor, the selection of observations which are compared to the model results. For calibration this file also specifies which parameters can be changed. For data assimilation this files contains the definition of the model state and the noise (uncertainty) specification for the boundaries and forcings.

22.4.5 D-Flow FM files and the OpenDA dataObjects configuration

The dataObjects for reading and writing provide so-called exchange items that allow OpenDA to manipulate specific parts of the files of D-Flow FM. The D-Flow FM files that can be manipulated by OpenDA and the corresponding OpenDA class names are given in Table 22.2.

Table 22.2: *D-Flow FM files that can be manipulated and the corresponding OpenDA class names to be used in the <dflowfmWrapper.xml> file.*

D-Flow FM filetype	OpenDA dataObject & exchange items
<∗.mdu>	`org.openda.model_dflowfm.DFlowFMTimeInfo` IDs: `start_time, end_time`
<∗.amu>	`org.openda.model_dflowfm.DFlowFMMeteoFile` ID: `x_wind`
<∗.amv>	`org.openda.model_dflowfm.DFlowFMMeteoFile` IDs: `y_wind`
<∗.amp>	`org.openda.model_dflowfm.DFlowFMMeteoFile` ID: `air_pressure`
<∗.tim>	`org.openda.model_dflowfm.DFlowFMTimeSeriesDataObject` IDs: `BOUNDARY_ID.#:QUANTITY`
<∗.xyz>	`org.openda.model_dflowfm.DFlowFMXyzFile` IDs: `FILENAME_#`
<∗_his.nc>	`org.openda.exchange.dataobjects.NetcdfDataObject` IDs: `STATION_ID.VARIABLE_NAME`
<∗_map.nc>	`org.openda.model_dflowfm.DFlowFMRestartFileWrapper` IDs: `VARIABLE_NAME`

<∗.cld>	`org.openda.model_dflowfm.DFlowFMCalibrationFactorFile`
	IDs: `CalFactor-CALIBRATION_DEFINTION_NUMBER`,
	`CalFactor-CALIBRATION_DEFINTION_NUMBER-q{DISCHARGE}`,
	`CalFactor-CALIBRATION_DEFINTION_NUMBER-h{WATERLEVEL}`

<∗.ttd>	`org.openda.model_dflowfm.DFlowFMTrachytopeFile`
	IDs: `RoughNr_{ROUGHNESSNR}_FormulaNr{FORMULANR}_{FORMULAPARAMETER}`,
	`RoughNr_{ROUGHNESSNR}_DISCHARGE{DISCHARGE}_FormulaNr{FORMULANR}_{FORMULAPARAMETER}`,
	`RoughNr_{ROUGHNESSNR}_WATERLEVEL{WATERLEVEL}_FormulaNr{FORMULANR}_{FORMULAPARAMETER}`

All the dataObjects and their configuration are described tn the following sections.

22.4.5.1 Start and end time in the model definition file (.mdu)

The start and end time are set in <∗.mdu>-file the by OpenDA using the `start_time` and `end_time` exchange items. These are provided by the `DFlowFMTimeInfo` data object.

D-Flow FM reference	Exchange Item Id	Remark
`TStart`	`start_time`	`RefDate` and `Tunit` needed for interpretation
`TStop`	`end_time`	`RefDate` and `Tunit` needed for interpretation

22.4.5.2 Spatial external forcings (.xyz)

All D-Flow FM external forcings are specified via the <∗.ext> forcings file. Here a spatial forcing can be defined by using a `.xyz`-file (e.g. the bed friction coefficients). For instance the file <nikuradse.xyz> contains:

```
x1 y1 0.9994357525438934
x2 y2 0.9994357525438934
x3 y3 2.0021214673505600
x4 y4 2.0021214673505600
x5 y5 2.0021214673505600
x6 y6 2.0021214673505600
```

When performing calibration of a spatial field it is often required to group points in a select number of regions. The calibration then does not change the individual values but applies a factor to all values in the group. The best approach is to create a file with multipliers (e.g <friction_multiplier.xyz>) which are initially all equal to one.

```
x1 y1 1.0
x2 y2 1.0
x3 y3 1.0
x4 y4 1.0
x5 y5 1.0
x6 y6 1.0
```

The multiplication (or addition) with the values in `nikuradse.xyz` should be configured in the `*.ext` file.

Group from keywords in file

One option to construct groups is to use keywords directly in the <friction_multiplier.xyz> file:

```
x1 y1 1.0 #friction_3
x2 y2 1.0 #friction_3
x3 y3 1.0 #friction_1
x4 y4 1.0 #friction_1
x5 y5 1.0 #friction_4
x6 y6 1.0 #friction_4
```

In the OpenDA wrapper config the dataObject is than configured as:

```
<ioObject className="org.openda.model_dflowfm.DFlowFMXyzFile">
  <file>friction_multiplier.xyz</file>
  <id>frictionCoefFile</id>
  <arg>idsFromKeywordsInFile</arg>
</ioObject>
```

This will create exchange items with identifier `friction_3`, `friction_1` and `friction_4`.

Group from template file

An other options is to use a template file (<friction_multiplier_template.xyz>) with exactly the same (x,y) coordinates as in (<friction_multiplier.xyz>). The third column is used to define groups by using these values as group numbers:

```
x1 y1 3.0
x2 y2 3.0
x3 y3 4.0
x4 y4 4.0
x5 y5 1.0
x6 y6 1.0
```

In the OpenDA wrapper config the dataObject must be configured as:

```
<ioObject className="org.openda.model_dflowfm.DFlowFMXyzFile">
  <file>friction_multiplier.xyz</file>
  <id>frictionCoefFile</id>
  <arg>idsFromTemplateFile=friction_multiplier_template.xyz</arg>
</ioObject>
```

This will create an exchange item for each group with identifier 'FILE_BASENAME + _ + number from template file', e.g. `friction_multiplier_3`, `friction_multiplier_4` and `friction_multiplier_1`.

22.4.5.3 Boundary time series (`.tim`)

Boundary conditions are specified as a combination of a <∗.pli> file and one or more `.tim` files. The `DFlowFMTimeSeriesDataObject` dataObject creates exchange items for all boundary conditions. It starts with reading the name of the external forcing file name from the <.mdu>-file (key `ExtForceFile`). The <∗.ext>-file contains formatted blocks, one for each forcing. Forcings are defined along polylines, given in `.pli`-files. A <∗.pli>-file is accompanied by a <∗.cmp>- or a (number of) <∗.tim>-file(s).

Noise can be added by means of an extra block in the `.ext`-file. As an example, noise is added to a boundary with a discharge imposed as:

```
QUANTITY =dischargebnd
FILENAME =sw_east_dis.pli
FILETYPE =9
METHOD =3
OPERAND =O

QUANTITY =dischargebnd
FILENAME =sw_east_dis_noise.pli
FILETYPE =9
METHOD =3
OPERAND =+
```

The discharge is set by the first block (operand=O), the information in the <∗.pli>-files is identical and noise is added as a time series: the <_noise.pli> file is always accompanied by a (number of) <∗.tim> file(s). The `location`-information on the first line of the `.pli`-file combined with the quantity is used to construct the exchange item identifier: `location.1.dischargebnd`. The numbering is used to discern between multiple <∗.tim>-files possibly linked to a single <∗.pli>-file.

22.4.5.4 Meteorological boundary conditions (<∗.amu>, <∗.amv>, <∗.amp>)

OpenDA can read and write to the D-Flow FM <∗.amu>, <∗.amv> and <∗.amp> files using the `org.openda.model_dflowfm.DFlowFMMeteoFile` dataObject. These contain the x- and y components of the wind and the air pressure at the free surface on an equidistant grid. In a typical data assimilation use noise fields are added to the wind. For the this purpose OpenDA can generate a spatial noise field on an equidistant grid (see section 22.5). D-Flow FM can combine fields defined in files on different to single field on the computational grid.

22.4.5.5 Boundary conditions in bc format (`.bc`)

Boundary conditions can also be specified in a <∗.bc>-file. OpenDA can read and write to the D-Flow FM <∗.bc> files using the `org.openda.model_dflowfm.org.openda.model_dflowfm.BcFile` dataObject. This is being done in the `estuary_kalman_FMSuite2019.01_bcfile` example. A specific bc file is used for noise on the boundary and in this example `waterlevel_noise.bc` looks like:

```
[forcing]
Name = eastboundary_0001
Function = timeseries
FunctionIndex = 2
Time-interpolation = linear
Quantity = time
Unit = seconds since 1991-01-01 00:00:00
Quantity = waterlevelbnd
Unit = m
0 0
3600 0
```

22.4.5.6 Result time series (<∗_his.nc>)

The <∗_his.nc>-file contains time series with D-Flow FM model results for a number of stations. The generic `org.openda.exchange.dataobjects.NetcdfDataObject` is used for handling this type of files. The `NetcdfDataObject` expects a NetCDF-file that contains dimensions `time` and `stations` plus a variable `station_id` of type string and dimension `stations`. For each variable in this NetCDF-file with dimensions `(time,stations)` an exchange item is created, that can be referred to as `station_id(i).variablename`. For instance:

```
dimensions:
time = UNLIMITED ; // (2882 currently)
stations = 3 ;
station_name_len = 40 ;
variables:
char station_id(stations, station_name_len) ;
(containing strings Obs1, Obs2, Obs3)
double waterlevel(time, stations) ;
(containing the computed values of the waterlevel)
```

results in three exchange items (a 1D vector in this case) with identifiers: `Obs1.waterlevel`, `Obs2.waterlevel` and `Obs3.waterlevel`.

22.4.5.7 Restart file (<*_map.nc>)

OpenDA provides the `org.openda.model_dflowfm.DFlowFMRestartFileWrapper` for reading and writing the <*_map.nc>-file. This file contains all information needed to restart a D-Flow FM computation. Not all variables are relevant for manipulation by OpenDA: variable names that start with 'time', 'NetLink', 'BndLink', 'FlowLink', 'NetElem', 'FlowElem', 'NetNode', 'wgs84' and 'projected_coordinate_system' are ignored. Variables of other types than float or double are also ignored. For all other variables an exchange item is created, where the name of the variable in the NetCDF is used as the exchange item id.

Note: for Kalman filtering it is essential that the model starts from the <*_map.nc> file. It is not possible to specify an initial field by setting the `initialwaterlevel` or `initialsalinity` quantities in the <*.ext> file. If you want to set an initial field using these settings, make a custom D-Flow FM run where `TStop` is equal to `TStart` and use the created <*_map.nc> file for the starting point of the Kalman filtering. Also make sure that in the mdu file in the [output] section MapFormat = 1 so the mapfile stays in the correct format for restart.

22.4.5.8 Calibration factor definition file (<*.cld>)

The calibraction factor definition file has the following layout. This file includes examples of the three different types of calibration definition factors and in comments the resulting names of the exchange items for OpenDA.

```
# [FileInformation]
#   FileType      = CalibrationFactorsDefinitionFile
#   FileVersion   = 1.0
# [CalibrationFactors]

# Non-Q-or-h-dependent
#
# calibration-class-nr   calibration-factor
#
34 0.9
#
# (will lead to exchange item CalFactor-34)

# Q-dependent calibration factors
#
# calibration-class-nr DISCHARGE "Cross-section name"
# calibration-class-nr Q1 ConstantValueAtQ1
# calibration-class-nr Q2 ConstantValueAtQ2
# calibration-class-nr Q3 ConstantValueAtQ3
# calibration-class-nr Q4 ConstantValueAtQ4
#
601 DISCHARGE "cross-section name"
601 0100 1.0
601 1000 1.0
601 5500 1.0

# (will lead to exchange items:
#  CalFactor-601-q0100, CalFactor-601-q1000, CalFactor-601-q5500) etc.

# Waterlevel dependent calibration factor
# calibration-class-nr WATERLEVEL "Observation station name"
# calibration-class-nr H1 ConstantValueAtH1
# calibration-class-nr H2 ConstantValueAtH2
# calibration-class-nr H3 ConstantValueAtH3
# calibration-class-nr H4 ConstantValueAtH4
#
3 WATERLEVEL "water-level station name"
3 0.45 1.0
3 0.9  1.0
```

```
# (will lead to exchange items:
# CalFactor-3-h0.45 and CalFactor-3-h0.9)
```

OpenDA provides the `org.openda.model_dflowfm.DFlowFMCalibrationFactorFile` for reading and writing the <∗.cld>-file. In the OpenDA wrapper config `<dflowfmWrapper.xml>` the dataObject can be configured as:

```
<ioObject className="org.openda.model_dflowfm.DFlowFMCalibrationFactorFile">
   <file>FlowFM.cld</file>
   <id>calibFactorFileID</id>
</ioObject>
```

In the `dflowfmModel.xml` a listing of the exchange items which are to be used can be found. To select all of the available exchange items from the calibration factor definition file, the following code can be used:

```
<exchangeItems>
  <vector id="allElementsFromIoObject" ioObjectId="calibFactorFileID"/>
</exchangeItems>
```

To select just a few of the exchange items, these can also be selected individually:

```
<exchangeItems>
  <vector id="CalFactor-601-q0100" ioObjectId="calibFactorFileID"
    elementId="CalFactor-601-q0100"/>
  <vector id="CalFactor-601-q1000" ioObjectId="calibFactorFileID"
    elementId="CalFactor-601-q1000"/>
  <vector id="CalFactor-601-h0.45" ioObjectId="calibFactorFileID"
    elementId="CalFactor-601-h0.45"/>
  <vector id="CalFactor-601-q0.9" ioObjectId="calibFactorFileID"
    elementId="CalFactor-601-h0.9"/>
</exchangeItems>
```

22.4.5.9 Trachytopes roughness definition file (<∗.ttd>)

The trachytopes definition file has the following layout. This file includes examples of the three different types of calibration definition factors and in comments the resulting names of the exchange items for OpenDA.

```
# [FileInformation]
#   FileType     = TrachytopesRoughnessDefinitionFile
#   FileVersion  = 1.0
# [RoughnessDefinitions]
#
# Constant roughness definitions
# roughness-definition-code formula-number formula-parameters
# Nikuradse 0.2
7     51      0.2
# (Leads to exchange-item RoughNr_7_FormulaNr_51_A)

# Linear trachytopes: wooden barrier
9     201     5.0      1.25
# (Leads to exchange-items RoughNr_9_FormulaNr_201_A
# and RoughNr_9_FormulaNr_201_B})

#     Uniform chezy value 25
10    52      25
# (Leads to exchange-item RoughNr_10_FormulaNr_52_A)

# Discharge dependent trachytopes
# roughness-definition-code DISCHARGE cross-section-name
# roughness-definition-code Q1 formula-number formula-parametersatQ1
# roughness-definition-code Q2 formula-number formula-parametersatQ2
11     DISCHARGE    "m=1"
11     0.0      51       0.2
11     200.0    51       0.18
11     1000.0   51       0.1
%# (Leads to exchange-items RoughNr_11_DISCHARGE0.0_FormulaNr_51_A,
# RoughNr_11_DISCHARGE200.0_FormulaNr_51_A,
# RoughNr_11_DISCHARGE1000.0_FormulaNr_51_A)

# Water-level dependent trachytopes
# roughness-definition-code WATERLEVEL observation-station-name
# roughness-definition-code ZS1 formula-number formula-parametersatZS1
# roughness-definition-code ZS2 formula-number formula-parametersatZS2
11     WATERLEVEL    "obs1"
16     -1.2    151      4.0       0.1
16     -1      151      4.0       0.3
16      2      151      5.0       0.1
# (Leads to exchange-items RoughNr_16_WATERLEVEL-1.2_FormulaNr_151_A,
# RoughNr_16_WATERLEVEL-1.2_FormulaNr_151_B,
# RoughNr_16_WATERLEVEL-1_FormulaNr_151_A,
# RoughNr_16_WATERLEVEL-1_FormulaNr_151_B,
# RoughNr_16_WATERLEVEL2_FormulaNr_151_A,
# RoughNr_16_WATERLEVEL2_FormulaNr_151_B)
```

OpenDA provides the org.openda.model_dflowfm.dflowfm.DFlowFMTrachytopeFile for reading and writing the <∗.cld>-file. In the OpenDA wrapper config <dflowfmWrapper.xml> the dataObject can be configured as:

```
<ioObject className="org.openda.model_dflowfm.dflowfm.DFlowFMTrachytopeFile">
<file>FlowFM.ttd</file>
<id>trachytopesFileID</id>
</ioObject>
```

In the dflowfmModel.xml a listing of the exchange items which are to be used can be found. To select all of the available exchange items from the calibration factor definition file, the following code can be used:

```
<exchangeItems>
<vector id="allElementsFromIoObject" ioObjectId="trachytopesFileID"/>
</exchangeItems>
```

To select just a few of the exchange items, these can also be selected individually:

```
<exchangeItems>
  <vector id="RoughNr_7_FormulaNr_51_A" ioObjectId="trachytopesFileID"
  elementId="RoughNr_7_FormulaNr_51_A"/>
  <vector id="RoughNr_11_DISCHARGE0.0_FormulaNr_51_A" ioObjectId="trachytopesFileID"
  elementId="RoughNr_11_DISCHARGE0.0_FormulaNr_51_A"/>
  <vector id="RoughNr_16_WATERLEVEL-1.2_FormulaNr_151_A" ioObjectId="trachytopesFileID"
  elementId="RoughNr_16_WATERLEVEL-1.2_FormulaNr_151_A"/>
  <vector id="RoughNr_16_WATERLEVEL-1.2_FormulaNr_151_B" ioObjectId="trachytopesFileID"
  elementId="RoughNr_16_WATERLEVEL-1.2_FormulaNr_151_B"/>
</exchangeItems>
```

22.5 Generating noise

To add uncertainty to the external forcings and the boundary conditions, OpenDA has functionality for generating noise time series and spatial fields. The noise generation can be specified within the state definition in <dflowfmStochModel.xml>. For instance in the example estuary_kalman_FMSuite2019.01_bcfile, a noise time series is created (ID waterlevelnoise), which is added to the inflow discharge (ID eastboundary_0001.waterlevelbnd).

```
<?xml version="1.0" encoding="UTF-8"?>
<blackBoxStochModel xmlns="http://www.openda.org"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.openda.org/blackBoxStochModelConfig.xsd">
  <modelConfig>
    <file>./dflowfmModel.xml</file>
  </modelConfig>
  <vectorSpecification>
    <state>
      <noiseModel id="boundaryNoiseModelSurge"
       className="org.openda.noiseModels.TimeSeriesNoiseModelFactory"
       workingDirectory=".">
        <configFile>BoundaryNoiseSurge.xml</configFile>
        <exchangeItems>
          <exchangeItem id="waterlevelnoise"
           operation="add"
           modelExchangeItemId="eastboundary_0001.waterlevelbnd"/>
        </exchangeItems>
      </noiseModel>
      <vector id="s1"/>
      <vector id="unorm"/>
    </state>
    <predictor>
      <vector id="station01.waterlevel"/>
      <vector id="station02.waterlevel"/>
      <vector id="station03.waterlevel"/>
    </predictor>
  </vectorSpecification>
</blackBoxStochModel>
```

The config file <BoundaryNoiseSurge.xml> contains the details of the noise:

```
<?xml version="1.0" encoding="UTF-8"?>
<timeSeriesNoiseModelConfig xmlns="http://www.openda.org"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://schemas.openda.org/noiseModels/timeSeriesNoiseModel.xsd">
   <simulationTimespan timeFormat="dateTimeString">
    199101010000,199101010100,...,199101050000</simulationTimespan>
   <timeSeries id="waterlevelnoise" location="westboundary" quantity="waterlevel"
    standardDeviation="0.05" timeCorrelationScale="1.0"
    timeCorrelationScaleUnit="hours" initialValue="0.0"/>
</timeSeriesNoiseModelConfig>
```

A noise time series is created (ID `waterlevelnoise`) with a correlation time of 1 hour and a standard deviation of 0.05. **Note:** currently OpenDA does not support the creation of D-Flow FM files, all files should exist in the <input_dflowfm>. In this case the boundary is defined in <waterlevel_surge.bc>, but the noise is in a separate file <waterlevel_noise.bc>. Initially it contains zeros, but is filled with the noise values at run time. The time points in the file need to match the `simulationTimespan` in the OpenDA config. The waterlevel at the boundary is constructed by D-Flow FM where the noise is added to original boundary values (see <FlowFM_bnd.ext>).

To generate a spatial correlation field the same approach can be used. In the `lake_kalman` example a noise field is added the wind. Instead of `timeSeries` a `noiseItem` is specified which contains the grid definition.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapsNoiseModelConfig>
   <simulationTimespan timeFormat="dateTimeString">
     200106240000,200106240100,...,200106270000
   </simulationTimespan>
   <noiseItem id="2DNoise" quantity="wind-x" unit="m/s" height="10.0"
    standardDeviation="20.0"
    timeCorrelationScale="12.0" timeCorrelationScaleUnit="hours"
    initialValue="0.0"
    horizontalCorrelationScale="10" horizontalCorrelationScaleUnit="km">
      <grid type="cartesian" coordinates="XY">
         <x>0,3000,...,63000</x>
         <y>0,3000,...,63000</y>
      </grid>
   </noiseItem>
</mapsNoiseModelConfig>
```

The calculation of spatially correlated noise on a cartesian grid is quite fast as the x and y-direction are independent. The interpolation from an equidistant grid to the computational grid is performed within D-Flow FM. Again, note that an zero valued wind files should be present <input_dflowfm> folder, where the time stamps match with the ones given in `simulationTimespan` in the OpenDA configuration.

22.6 Examples of the application of OpenDA for D-Flow FM

In this section, some examples are elaborated for both the calibration of a model and the ensemble Kalman filtering (abbreviated as ‘EnKF’) of a model. All the examples can be found in the directory <examples/model_dflowfm_blackbox>.

22.6.1 Example 1: Calibration of the roughness parameter

The automatic calibration of a model needs two main choices from the user:

1 Which model parameters may be modified during the calibration process?
2 Which model results need to be compared to observations, to judge the model quality?

The remainder of this section will be in the form of a tutorial, to directly illustrate all steps in an example model. In this example you will use a small river model ‘simple_waal’ and use the bed roughness to calibrate this model for its three water level observation stations.

Step 1: Inspect the model

All the required model files can be found in the directory <simple_waal_calibration_roughness>. Consider the following steps:

1 Start D-Flow FM (standalone) in directory <input_dflowfm/>.
2 Select *Files* → *Load MDU-file*.
3 Load the model: select <simple_waal.mdu>.
4 You can run the model if you like (right mouse button).

The basic model is built to simulate a simple two-dimensional river with a spatially varying bed friction coefficient. It is driven by two boundary conditions: an upstream discharge inflow at the eastern boundary and a downstream water level at the western boundary. Inspect the model forcing in the following way:

1 Open the external forcings file <simple_waal.ext> in a text editor.
2 Notice how, in addition to the two boundaries, there are two blocks for the friction coefficient. The first one is a spatially varying roughness field in the <sw_nikuradse.xyz> file. The second refers to the <sw_frcfact_all.xyz> file that contains multipliers for the original friction coefficients. A third file <sw_frcfact_template.xyz> is present, which is used to define a number of subdomains.
3 In D-Flow FM, select *Files* → *Load sample file* and select <sw_frcfact_template.xyz>.
4 Notice how the loaded samples have three distinct values 1, 2 and 3, which act as identifiers: they approximately define the corner points of three subdomains of the entire river

stretch. For each subdomain, a different roughness can be calibrated.

Step 2: Select the model parameters

Currently, the only calibratable parameters are the time-independent parameters in the external forcings file that use the `.xyz` sample file format. The most obvious parameter is the bed friction coefficient.

Step 3: Select the model results

The example directory <simple_waal_calibration_roughness> contains all the necessary configuration files for the so-called Black Box model wrapper for D-Flow FM to run a 'twin experiment'. In a twin experiment a model setup with given solution (the synthetic observations) is perturbed after which OpenDA is applied to re-estimate the original settings. The effects of the parameter variations may be judged by comparing time series output in the <∗_his.nc> history file to observed data in NOOS time series format. The model output and observation data are compared by calculating a cost function. Which cost function is used is configured in the <dudAlgorithm.xml> in the <algorithm> directory. See the OpenDA documentation for the available options for the cost function.

The D-Flow FM model simulates a 1D river flow. The input files for D-Flow FM are located in the directory <simple_waal_calibration_roughness/stochModel/input>. D-Flow FM allows the user to specify regions with a different bed friction coefficient (constant for each region) and is able to handle the interpolation of the coefficients between these regions. In the experiment we try to re-estimate the values of the bed friction coefficients of an earlier run. As observations, the waterlevel at three locations (stations) along the river is used. These results are written to the main output file (<∗_his.nc>) as time series.

In the directory <simple_waal_calibration_roughness>, there are two main configuration files of OpenDA present:

◇ `Simulation.oda`: runs a single run of the model, this configuration is mainly used to test the black box configuration files.
◇ `Dud.oda` runs a calibration experiment with algorithm DUD (Doesn't Use Derivative).

These files configure the main ingredients of an OpenDA run:

1 the stochObserver (`org.openda.observers.NoosTimeSeriesStochObserver`). Observations for this experiment were created by extracting the timeseries for all stations from the netcdf-file and convert them to NOOS format (script nchis2noos.sh) het script schrijft nog tijd sinds begin situatie.
2 the stochObserver (`org.openda.observers.NoosTimeSeriesStochObserver`). Observations for this experiment were created by extracting the time series for all stations from the netcdf-file and convert them to NOOS format (script nchis2noos.sh)
3 the stochModelFactory (org.openda.blackbox.wrapper.BBStochModelFactory). A black box model configuration consist of three configuration files that are described in more detail below.
4 the algorithm (org.openda.algorithms.Dud). Dud is a well known algorithm in calibration experiments, more information about it can be found on the OpenDA website or in the literature.

The configuration files for these 3 components are located in different sub directories to reflect the Object Oriented architecture of OpenDA. The fourth block in the configuration file specifies the result writer (org.openda.resultwriters.MatlabResultWriter). The resulting m-file may be loaded in Matlab to visualize results of the OpenDA run.

In this example, the data exchange between OpenDA and D-Flow FM is limited to the bed friction coefficients and the computed waterlevel at observation locations. The waterlevel at a observation location is expected to be written to NetCDF-file with the following features

1 dimension ‘time’ and ‘stations’ are defined
2 there exists a variable ‘station_id(stations)’ defined that contains strings with the station_id

For NetCDF-files that satisfy these two conditions OpenDA creates an ‘exchange item’ for each variable that has the dimensions (time, stations). The exchange item is referred to as ‘station id(nr)’.’name of variable’.

22.6.2 Example 2: EnKF with uncertainty in the tidal components

The geometry for this test case is the same as used in the Delft3D-FLOW model example for calibration that was presented in a Deltares webinar (recording, slides and all configuration files for this example are available at the OpenDA website).

Regularly, D-Flow FM uses 1 component file to specify all tidal component (one component at a line). In order to add different noise models to different components, you must split the component file and add one <∗.tim>-files for noise for each component.

Again, all configuration files are available, but not much effort has been put into the exact configuration of the EnKF algorithm or the noise model specifications. The results of the SequentialSimulation show that this test case suffers much less from the inexact restart. The whole workflow is highlighted in more detail below.

A few remarks are made:

◇ the directories <bin> and <jre> should be on the same level,
◇ the computation is started through running the file oda_run_gui.bat,
◇ the bare D-Flow FM model is located in the directory <input_dflowfm>,
◇ the observations are in `.noos`-format, and are located in the directory <stochObserver>.

Step 2: Start the EnKF computation

The OpenDA run is launched through the core `oda_run_gui.bat` file. Once having opened this file, a user interface appears. Within the user interface, an <∗.oda>-file can be opened from a certain case directory (in this case, we have <estuary_kalman_FMSuite2019.01_bcfile>). One can choose `Enkf.oda`, `SequentialSimulation.oda` or `Simulation.oda`. In this case, we choose for `Enkf.oda`.

Step 3: Examine the applied noise

The basic necessary component of an EnKF computation comprises the noise applied to some variable. In this case, the noise is applied to the waterlevel boundary representing the tidal motion. Within the directory <input_dflowfm>, this noise is explicitly declared through a separate polyline and a separate data file for the boundary. The configuration of the noise is accomplished through two <∗.xml>-files in the directory <stoch_model>.

Step 4: Run the EnKF computation

By means of the user interface, the EnKF computation can be launched. After having opened the file <EnKF.oda>, the *Run*-button can be pressed. The computation is being performed. Along the computation's duration, multiple <work> directories are generated in the directory <stochModel>, i.e. <work0>, <work1>, <work2>, etc.

Step 5: Evaluate the outcomes

After having run the computation, output files have been generated in Matlab-format. The relevant files are placed in the directory <estuary_kalman>. The data are stored in the Matlab file <Enkf_results.m>. Visualisation could be accomplished like shown in Figure 22.1.

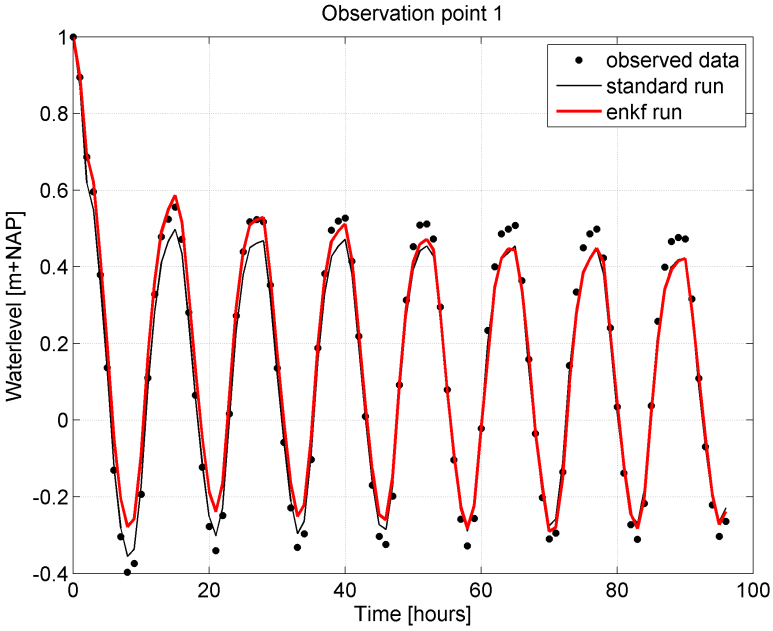


Figure 22.1: *Visualisation of the EnKF computation results from OpenDA for a certain observation point. The dots represent the observed data, the black line represents the original computation with D-Flow FM (without Kalman filtering) and the red line represents the D-Flow FM computation with Kalman filtering.*

22.6.3 Example 3: EnKF with uncertainty in the inflow velocity

The geometry in this example is the same as for the calibration example: a two-dimensional river model, the initial waterlevel is zero, the river bed is filled gradually due to the boundary conditions. At the inflow boundary, a constant discharge is prescribed (along the line <sw_east_dis_0001.pli>), whereas at the outflow boundary, a constant water level is prescribed (along the line <sw_west_wlev_0001.cmp>).

There are 3 observation locations along the river (Obs01, Obs02 and Obs03). The matlab script <plot_results_hisfile.m> is available to plot the water level as a function of time for these 3 stations. Simulation time span is 100 days (Start: 199208310000, End: 199212090000). The noise contribution is found in file <sw_east_dis_noise.pli>. Initially, no noise is present, so the file contains zeroes in directory <input_dflowfm>.

22.6.4 Example 4: EnKF with uncertainty in the inflow condition for salt

The geometry in this example and the boundary conditions for waterlevel and velocity are exactly the same as in `simple_waal_kalman`. The transport of salt is added to the computation by a discharge boundary condition `sw_east_dis_sal_001.pli` and a noise component added to this boundary.

22.6.5 Example 5: EnKF with uncertainty on the wind direction

This example is also converted from a Delft3D-FLOW test case (`d3d_lake_2d`): a lake forced by an uniform wind field. This example is a twin experiment with spatially correlated 2D-noise added to the wind field. The noise is created on cartesian (equidistant grid), the noise realisations are written by OpenDA to the <lake2d_windx_noise.amu>. The 2D noise field is interpolated at computational grid by D-Flow FM and added to the uniform wind field. The <SequentialSimulationNoise.oda> can be used to perform a single D-Flow FM run where wind with noise is used as forcing. The resulting <lake2d_his.nc> file can be is used as syntethic observations by the <stochObserver>.

22.6.6 Example 6: EnKF with the DCSM v5 model and uncertainty on the wind direction

This example is converted from the SIMONA DCSM v5 model with spatially correlated 2D-noise added to the wind field.

References

Baptist, M. J., 2005. *Modelling floodplain biogeomorphology*. Ph.D. thesis, Delft University of Technology.

Beckers, J. M., H. Burchard, J. M. Campin, E. Deleersnijder and P. P. Mathieu, 1998. "Another reason why simple discretizations of rotated diffusion operators cause problems in ocean models: comments on "isoneutral diffusion in a z co-ordinate ocean model"." *American Meteorological Society* 28: 1552–1559.

Bos, 1989. *Discharge Measuring Structures*. International Institute for Land Reclamation and Improvement/ILRI, Wageningen, The Netherlands.

Bos, W. de and C. Bijkerk, 1963. "Een nieuw monogram voor het berekenen van waterlopen." *Cultuurtechnisch Tijdeschrift* 3 (4): 149–155.

Charnock, H., 1955. "Wind-stress on a water surface." *Q. J. Royal Meteorol. Soc.* 81: 639–640.

Colebrook, C., 1939. "Turbulent flow in pipes, with particular reference to the transition region between the smooth and rough pipe laws." *Journal of the Institution of Civil Engineers* 11 (4): 133–156.

Colebrook, C. and C. White, 1937. "Experiments with Fluid Friction in Roughened Pipes." *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences* 161 (906): 367–381.

D-Morphology UM, 2019. *D-Morphology User Manual*. Deltares, 1.5 ed.

D-RR_UM, 2019. *SOBEK 3 / D-Rainfall Runoff User Manual*. Deltares, Delft. Version: 3.6.1.

D-RTC UM, 2019. *D-Real Time Control User Manual*. Deltares, 1.4 ed.

Deltares, 2024a. *D-Flow Flexible Mesh Technical Reference Manual*. Deltares, Delft, 1.1.124 ed.

Deltares, 2024b. *D-Water Quality Description of Input file*. Deltares, 2.00 ed.

Deltares, 2024c. *D-Water Quality Processes Library Configuration Tool User Manual*. Deltares, 0.01 ed.

Deltares, 2024d. *D-Water Quality Technical Reference Manual*. Deltares, 5.01 ed.

Deltares, 2024e. *D-Water Quality User Manual*. Deltares, 5.06 ed.

Deltares, 2024. *DIMR Technical Reference Manual*. Deltares, Delft.

Eckart, C., 1958. "Properties of water, Part II. The equation of state of water and sea water at low temperatures and pressures." *American Journal of Science* 256: 225–240.

ECMWF, 2023. *IFS DOCUMENTATION – Cy48r1, Operational implementation 27 June 2023, PART IV: PHYSICAL PROCESSES*. Tech. rep., ECMWF.

Fofonoff, N. P. and R. C. Millard Jr., 1983. *Algorithms for the computation of fundamental properties of seawater*. UNESCO Technical Papers in Marine Sciences 44, UNESCO, Paris, France.

Garratt, J. R., 1977. "Review of Drag Coefficients over Oceans and Continents." *Review Articles in Monthly Weather Review* 105 (7): 915–929.

Gill, A. E., 1982. *Atmosphere-Ocean dynamics*, vol. 30 of *International Geophysics Series*. Academic Press.

Haney, R. L., 1991. "On the pressure gradient force over steep topography in sigma coordinate models." *Journal of Physical Oceanography* 21: 610–619.

Hersbach, H., 2011. "Sea Surface Roughness and Drag Coefficient as Functions of Neutral Wind Speed." *Journal of Physical Oceanography* 41 (1): 247–251.

Hibler III, W. D., 1979. "A dynamic thermokdynamic sea ice model." *Journal of Physical Oceanography* 9 (4): 815–846.

Hirsch, C., 1990. *Numerical computation of internal and external flows*. John Wiley & Sons, New York.

Horton, R. E., 1933. "The Rôle of infiltration in the hydrologic cycle." *Eos, Transactions American Geophysical Union* 14 (1): 446-460. DOI: 10.1029/TR014i001p00446, URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/TR014i001p00446.

Huang, W. and M. Spaulding, 1996. "Modelling horizontal diffusion with sigma coordinate system." *Journal of Hydraulic Engineering* 122 (6): 349–352.

Hunke, E. C. and J. K. Dukowicz, 1997. "An elastic-ciscous-plastic model for sea ice." *Journal of Physical Oceanography* 27 (9): 1849–1867.

Hwang, P., 2005a. "Comparison of the ocean surface wind stress computed with different parameterization functions of the drag coefficient." *J. Oceanogr.* 61: 91–107.

Hwang, P., 2005b. "Drag coefficient, dynamic roughness and reference wind speed." *J. Oceanogr.* 61: 399–413.

Kalkwijk, J. P. T. and R. Booij, 1986. "Adaptation of secondary flow in nearly horizontal flow." *Journal of Hydraulic Research* 24 (1): 19–37.

Karypis, G., 2013. *METIS - A Software Package for Partitioning Unstructured Graph, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 5.1.0*. Tech. rep., Department of Computer Science and Engineering, University of Minnesota.

Klopstra, D., H. J. Barneveld and J. M. Van Noortwijk, 1996. *Analytisch model hydraulische ruwheid van overstroomde moerasvegetatie*. Tech. Rep. PR051, HKV consultants, Lelystad, The Netherlands. Commissioned by Rijkswaterstaat/RIZA, The Netherlands.

Klopstra, D., H. J. Barneveld, J. M. Van Noortwijk and E. H. van Velzen, 1997. "Analytical model for hydraulic roughness of submerged vegetation." In *The 27th IAHR Congress, San Francisco, 1997; Proceedings of Theme A, Managing Water: Coping with Scarcity and Abundance*, pages 775–780. American Society of Civil Engineers (ASCE), New York.

Knaap, F. Van der, 2000. *Breach growth as a function of time*. Tech. Rep. Q2655, WL | Delft Hydraulics, Delft, The Netherlands. Memos 2 and 3 of May 21 and September 5 respectively.

Lane, A., 1989. *The heat balance of the North Sea*. Tech. Rep. 8, Proudman Oceanographic Laboratory.

Leendertse, J. J., 1990. "Turbulence modelling of surface water flow and transport: part IVa." *Journal of Hydraulic Engineering* 114 (4): 603–606.

Love, A. E. H., 1927. *A Treatise on the Mathematical Theory of Elasticity*. Cambridge University Press, 4th ed.

Madec, G., M. Bell, R. Bourdallé-Badie, J. Chanut, E. Clementi, A. Coward, M. Drudi, I. Epicoco, C. Ethé, D. Iovino, D. Lea, C. Lévy, N. Martin, S. Masson, P. Mathiot, F. Mele, S. Mocavero, A. Moulin, S. Müller, G. Nurser, G. Samson and D. Storkey, 2022. *NEMO ocean engine*. Scientific Notes. Institut Pierre-Simon Laplace, Paris, France, v4.2 ed.

Malone, T. and A. D. Parr, 2008. *Bend Losses in Rectangular Culverts*. Tech. Rep. K-TRAN: KU-05-5, Kansas Department of Transportation and Kansas State University.

Manning, R., 1891. "On the flow of water in open channels and pipes." *Transactions of the Institution of Civil Engineers of Ireland* 20: 161–207.

Mellor, G. and A. F. Blumberg, 1985. "Modelling vertical and horizontal diffusivities with the sigma coordinate system." *Monthly Weather Review* 11: 1379–1383.

Mellor, G. L. and L. Kantha, 1989. "An ice-ocean coupled model." *Journal of Geophysical Research* 94 (8): 10937–10954.

Millero, F. J. and A. Poisson, 1981. "International one-atmosphere equation of state of sea water." *Deep-Sea Research* 28A (6): 625–629.

Octavio, K. A. H., G. H. Jirka and D. R. F. Harleman, 1977. *Vertical Heat Transport Mechanisms in Lakes and Reservoirs*. Tech. Rep. 22, Massachusetts Institute of Technology.

Peckham, S. D., E. W. Hutton and B. Norris, 2013. "A component-based approach to integrated modeling in the geosciences: The design of CSDMS." *Computers & Geosciences* 53: 3–12.

Phillips, N. A., 1957. "A co-ordinate system having some special advantages for numerical forecasting." *Journal of Meteorology* 14: 184–185.

RGFGRID UM, 2016. *Delft3D-RGFGRID User Manual*. Deltares, 5.00 ed.

Rijn, L. C. van, 1984. "Sediment transport, Part III: bed form and alluvial roughness." *Journal of Hydraulic Engineering* 110 (12): 1733–1754.

Rijn, L. C. van, 2007. "Unified View of Sediment Transport by Currents and Waves. I: Initiation of Motion, Bed Roughness, and Bed-Load Transport." *Journal of Hydraulic Engineering* 133 (6): 649–667.

Rodi, W., 1984. "Turbulence models and their application in Hydraulics, State-of-the-art paper article sur l'etat de connaissance." *IAHR* Paper presented by the IAHR-Section on Fundamentals of Division II: Experimental and Mathematical Fluid Dynamics, The Netherlands.

Ryan, P. J., D. R. F. Harleman and K. D. Stolzenbach, 1974. "Surface Heat Loss From Cooling Ponds." *Water Resources Research* 10 (5): 930–938.

Schrama, E., 2007. "Tides. Lecture Notes AE4-876."

Semtner Jr., A. J., 1976. "Numerical simulation of the Arctic Ocean circulation." *Journal of Physical Oceanography* 6 (4): 409–425.

Slørdal, L. H., 1997. "The pressure gradient force in sigma-co-ordinate ocean models." *International Journal Numerical Methods In Fluids* 24: 987–1017.

Smith, S. D. and E. G. Banke, 1975. "Variation of the sea surface drag coefficient with wind speed." *Quarterly Joournal of the Royal Meteorological Society* 101: 665–673.

Soulsby, R. L., L. Hamm, G. Klopman, D. Myrhaug, R. R. Simons and G. P. Thomas, 1993. "Wave-current interaction within and outside the bottom boundary layer." *Coastal Engineering* 21: 41–69.

Stelling, G. S. and J. A. T. M. van Kester, 1994. "On the approximation of horizontal gradients in sigma co-ordinates for bathymetry with steep bottom slopes." *International Journal Numerical Methods In Fluids* 18: 915–955.

Strickler, A., 1923. *Beiträge zur Frage der Geschwindigheits-formel und der Rauhigkeitszahlen fur Ströme, Kanäle und geschlossene Leitungen.* Tech. Rep. Mitteilungen des Eidgenössischen Amtes für Wasserwirtshaft, No. 16. (Some contributions to the problem of the velocity formula and roughnessfactors for rivers, canals, and closed conduits.).

Sweers, H. E., 1976. "A nomogram to estimate the heat exchange coefficient at the air-water interface as a function of windspeed and temperature; a critical survey of some literature." *Journal of Hydrology* 30: –.

Uittenbogaard, R. E., J. A. T. M. van Kester and G. S. Stelling, 1992. *Implementation of three turbulence models in 3D-TRISULA for rectangular grids*. Tech. Rep. Z81, WL | Delft Hydraulics, Delft, The Netherlands.

UNESCO, 1981a. *Background papers and supporting data on the international equation of state 1980*. Tech. Rep. 38, UNESCO.

UNESCO, 1981b. *The practical salinity scale 1978 and the international equation of state of seawater 1980*. Tech. Rep. 36, UNESCO. Tenth report of the Joint Panel on Oceanographic Tables and Standards (1981), (JPOTS), Sidney, B.C., Canada.

Velzen, E. H. van, P. Jesse, P. Cornelissen and H. Coops, 2003. *Stromingsweerstand vegetatie in uiterwaarden*. Tech. Rep. 2003.029, Rijkswaterstaat/RIZA.

Verheij, H., 2002. *Modification breach growth model in HIS-OM*. Tech. Rep. Q3299, WL | Delft Hydraulics, Delft, The Netherlands. (in Dutch).

Wang, J., Q. Liu, M. Jin, M. Ikeda and F. J. Saucier, 2005. "A Coupled Ice-Ocean Model in the Pan-Arctic and North Atlantic Ocean: Simulation of Seasonal Cycles." *Journal of Oceanography* 61: 213–233.

Winterwerp, J. C. and R. E. Uittenbogaard, 1997. *Sediment transport and fluid mud flow*. Tech. Rep. Z2005, WL | Delft Hydraulics, Delft, The Netherlands.

Wüest, A. and A. Lorke, 2003. "Small-Scale Hydrodynamics in Lakes." *Annual Review of Fluid Mechanics* 35 (1): 373–412.

Wunderlich, W. O., 1972. *Heat and Mass Transfer Between a Water Surface and the Atmosphere*. Water Resources Research Laboratory Report 14, TVA (Tennessee Valley Authority), Tennessee Valley Authority, Division of Water Control Planning, Engineering Laboratory, Norris, TN.

A The master definition file

A.1 Overview of keywords generated by GUI

The mdu-file contains the key information of the flow model. Besides the names of the relevant user specified files, such as the grid file and the external forcings file, the values of various model parameters should be specified in the MDU-file. In Table A.1 the model parameter settings are given as well as the associated default setting for these parameters that are generated by the Graphical User Interface of D-Flow FM. If a model parameter has a unit, then this unit will also be given in [ ] in the column of Description in this table.

The basename of the mdu-file, without `.mdu`, is also used as the model identification string, and is often denoted as *mdu_name* throughout this User Manual. It is equivalent to Delft3D-FLOW's *runid* concept.

Table A.1: *Standard MDU-file with default settings.*

Keyword	Default setting	Description
`[general]`		
`Program`	`D-Flow FM`	Program.
`Version`	`1.2.60.64623M`	Version number of computational kernel.
`fileType`	`modelDef`	File type. Do not edit this.
`fileVersion`	`1.09`	File version. Do not edit this.
`GuiVersion`	`1.5.4.45545`	Version number of GUI.
`AutoStart`	`0`	Autostart simulation after loading MDU or not (0=no, 1=autostart, 2=autostartstop).
`PathsRelativeToParent`	`0`	Whether or not (1/0) to resolve file names (e.g. inside the *.ext file) relative to their direct parent, instead of to the toplevel MDU working dir.
`[geometry]`		
`NetFile`		<*_net.nc>. See Appendix B.
`DryPointsFile`		Dry points file <*.xyz>, third column dummy z values, or polygon file <*.pol>.
`GridEnclosureFile`		Enclosure file <*.pol> to clip outer parts from the grid.
`StructureFile`		File <*.ini> containing list of hydraulic structures. See section C.12.
`GulliesFile`		Polyline file <*_gul.pliz>, containing lowest bed level along talweg x, y, z level.
`RoofsFile`		Polyline file <*_roof.pliz>, containing roofgutter heights x, y, z level.
`IniFieldFile`		Initial and parameter field file <*.ini>. See section D.2.
`WaterLevIniFile`		Initial water levels sample file <*.xyz>.
`LandBoundaryFile`		Only for plotting.
`ThinDamFile`		<*_thd.pli>, Polyline(s) for tracing thin dams.
`FixedWeirFile`		<*_fxw.pliz>, Polyline(s) x, y, z, z = fixed weir top levels (formerly fixed weir).
`PillarFile`		<*_pillar.pliz>, Polyline file containing four colums with x, y, diameter and Cd coefficient for bridge pillars.
`UseCaching`	`1`	Use caching for geometrical/network-related items (0: no, 1: yes) (section C.19).
`VertplizFile`		<*_vlay.pliz>), = pliz with x, y, Z, first Z = nr of layers, second Z = laytyp.

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
FrictFile		Location of the files with roughness data for 1D. See section C.15.
CrossDefFile		Cross section definitions for all cross section shapes. See section C.16.
CrossLocFile		Location definitions of the cross sections on a 1D network. See section C.16.
StorageNodeFile		File containing the specification of storage nodes and/or manholes to add extra storage to 1D models. See section C.17.
1D2DLinkFile		File containing the custom parameterization of 1D-2D links. See section C.18.
AllowBndAtBifurcation	0	Allow 1d boundary node when connecting branch leads to bifurcation (1: yes, 0: no).
ProflocFile		<*_proflocation.xyz>) x, y, z, z = profile refnumber.
ProfdefFile		<*_profdefinition.def>) definition for all profile nrs.
ProfdefxyzFile		<*_profdefinition.def>) definition for all profile nrs.
ManholeFile		File containing manholes (e.g. <*.dat>).
PartitionFile		<*_part.pol>, polyline(s) x, y.
WaterLevIni	0.	Initial water level [m AD].
Bedlevuni	-5.	Uniform bed level [m AD], (only if BedlevType>=3), used at missing z values in netfile.
Bedslope	0.	Bed slope inclination [-], sets zk = bedlevuni + x*bedslope and sets zbndz = xbndz*bedslope.
BedlevType	3	1: at cell center (tiles xz, yz, bl, bob=max(bl)), 2: at face (tiles xu, yu, blu, bob=blu), 3: at face (using mean node values), 4: at face (using min node values), 5: at face (using max node values), 6: with bl based on node values.
Blmeanbelow	-999.	if not -999d0, below this level [m] the cell centre bedlevel is the mean of surrouding netnodes.
Blminabove	-999.	if not -999d0, above this level [m] the cell centre bedlevel is the min of surrouding netnodes.
AngLat	0.	Angle of latitude S-N [deg], 0=no Coriolis.
AngLon	0.	Angle of longitude E-W [deg], 0=Greenwich Mean Time.
Conveyance2D	-1	-1:R=HU, 0:R=H, 1:R=A/P, 2:K=analytic-1D conv, 3:K=analytic-2D conv.
Nonlin1D	1	Non-linear 1D volumes, applicable for models with closed cross sections. 1=treat closed sections as partially open by using a Preissmann slot, 2=Nested Newton approach, 3=Partial Nested Newton approach. (For a description of these methods, see (Deltares, 2024a, Section 6.9).
Nonlin2D	0	Non-linear 2D volumes, only i.c.m. BedlevType=3 and Conveyance2D>=1.
slotw1D	0.001	Minimum slotwidth 1D [m].
slotw2D	0.001	Minimum slotwidth 2D [m].
dthdth1D	2.	Uniform width for 1D profiles and 1d2d internal links [m].
Uniformheight1D	3.	Uniform height for 1D profiles and 1d2d internal links [m].
Uniformwidth1Dstreetinlets	0.2	Uniform width for street inlets [m].

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
`Uniformheight1Dstreetinlets`	`0.1`	Uniform height for street inlets [m].
`Uniformtyp1Dstreetinlets`	`-2`	Uniform cross section type for street inlets (1: circle, 2: rectangle, -2: closed rectangle).
`Uniformwidth1Droofgutterpipes`	`0.1`	Uniform width for roof gutter pipes [m].
`Uniformheight1Droofgutterpipes`	`0.1`	Uniform height for roof gutter pipes [m].
`Uniformtyp1Droofgutterpipes`	`-2`	Uniform cross section type for type roof gutter pipes (1: circle, 2: rectangle, -2: closed rectangle).
`Sillheightmin`	`0.0`	Fixed weir only active if both ground heights are larger than this value [m AD].
`Makeorthocenters`	`0`	(1: yes, 0: no) switch from circumcentres to orthocentres in geominit.
`Dcenterinside`	`1.`	Limit cell center [-]; 1.0:in cell <-> 0.0:on c/g.
`Bamin`	`1E-06`	Minimum grid cell area [m^2], i.c.m. cutcells.
`OpenBoundaryTolerance`	`3.`	Search tolerance factor between boundary polyline and grid cells. [Unit: in cell size units (i.e., not meters)].
`RenumberFlowNodes`	`1`	Renumber the flow nodes (1: yes, 0: no).
`Kmx`	`0`	Number of vertical layers. NB. If kewyord SigmaGrowthFactor is used, then number of layers is determined by D-Flow FM [-].
`Layertype`	`1`	1= sigma-layers, 2 = z-layers, 3 = use VertplizFile.
`Numtopsig`	`0`	Number of sigma-layers on top of z-layers in case of z-sigma-layers.
`SigmaGrowthFactor`	`1.`	Growth factor of z-Layer thickness starting below the level specified by Dztopuniabovez till the bed [-].
`dxmin1D`	`0.001`	Minimum 1D link length [m].
`dxDoubleAt1DEndNodes`	`true`	Whether a 1D grid cell at the end of a network has to be extended with $0.5\Delta x$.
`ChangeVelocityAtStructures`	`false`	Ignore structure dimensions for the velocity at hydraulic structures, when calculating the surrounding cell centered flow velocities.
`ChangeStructureDimensions`	`true`	Change the structure dimensions in case these are inconsistent with the channel dimensions. ◇ `weirs`, `orifices`, `general structures`: 1. In case the crest width exceeds the surface width, the crest width is set to the surface width; 2. In case the crest level is lower than the bed level, the crest level is set to the bed level. ◇ `bridges`: 1. In case the crest width exceeds the surface width, the crest width is set to the surface width; 2. In case the flow area of the bridge exceeds the upstream flow area the flow area of the bridge is set to the upstream flow area. ◇ `universal weirs`: only the crest level is checked and changed. **NOTE: It is strongly advised not to change this parameter (`true`). Since turning this option off can lead to instabilities and unrealistic results.**
`[volumeTables]`		
`useVolumeTables`	`0`	Use volume tables for 1D grid cells (see section 8.16) (0: no, 1: yes).
`increment`	`0.1`	The height increment for the volume tables [m].

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
`useVolumeTableFile`	0	Read and write the volume table from/to file (0: no, 1: yes).
`[numerics]`		
`CFLMax`	0.7	Maximum Courant nr [-].
`AdvecType`	33	Adv type, 0=no, 33=Perot q(uio-u) fast, 3=Perot q(uio-u).
`AdvecCorrection1D2D`	0	Advection correction of 1D2D link volume (0: regular advection, 1: link volume au*dx, 2: advection on 1D2D switched off.)
`TimeStepType`	2	0=only transport, 1=transport + velocity update, 2=full implicit step_reduce, 3=step_jacobi, 4=explicit.
`maxNonlinearIterations`	100	Maximal iterations in non-linear iteration loop before a time step reduction is applied.
`setHorizontalBobsFor1d2d`	0	Bobs are set to 2D bedlevel, to prevent incorrect storage in sewer system (0: no, 1:yes).
`Limtyphu`	0	Limiter type for waterdepth in continuity eq., 0=no, 1=minmod,2=vanLeer,3=Koren,4=Monotone Central.
`Limtypmom`	4	Limiter type for cell center advection velocity, 0=no, 1=minmod,2=vanLeer,4=Monotone Central.
`Limtypsa`	4	Limiter type for salinity transport, 0=no, 1=minmod,2=vanLeer,4=Monotone Central.
`Pure1D`	0	Purely 1D advection (0: original advection using velocity vector, 1: pure 1D using flow volume vol1_f, 2: pure 1D using volume vol1) (section 8.3.7).
`Junction1D`	0	Advection at 1D junctions: (0: original 1D advection using velocity vector, 1 = same as along 1D channels using Pure1D=1) (section 8.3.7).
`Icgsolver`	4 or 6	Solver type, 4 = sobekGS + Saad-ILUD (default sequential), 6 = PETSc (default parallel), 7= CG+MILU (parallel).
`LogSolverConvergence`	0	Print time step, number of solver iterations and solver residual to diagnostic output (0: no, 1: yes).
`Maxdegree`	6	Maximum degree in Gauss elimination.
`FixedWeirScheme`	9	6 = semi-subgrid scheme, 8 = Tabellenboek, 9 = Villemonte (default).
`FixedWeirContraction`	1.	flow width = flow width*FixedWeirContraction. [-]
`Fixedweirtopwidth`	3	Uniform width of the groyne part of fixed weirs [m].
`Fixedweirtalud`	4	Uniform talud slope of fixed weirs.
`Fixedweirtopfrictcoef`		Uniform friction coefficient of the groyne part of fixed weirs [the unit depends on frictiontype].
`FixedweirRelaxationcoef`	0.6	Fixed weir relaxation coefficient for computation of energy loss.
`FixedweirScheme1d2d`	0	Fixed weir scheme for 1d2d links (0: same as fixedweirscheme, 1: lateral iterative fixed weir scheme).
`Fixedweir1d2d_dx`	50.0	Extra delta x for lateral 1d2d fixed weirs.
`Izbndpos`	0	Position of z boundary, 0=mirroring of closest cell (as in Delft3D-FLOW), 1=on net boundary.
`Tlfsmo`	0.	Fourier smoothing time on water level boundaries [s].
`Slopedrop2D`	0.	Apply droplosses only if local bottom slope > Slopedrop2D [m], <=0 =no droplosses.
`Drop1D`	0	Limit the downstream water level in the momentum equation to the downstream invert level, BOB_{down} ($\zeta^*_{down} = \max\left(\mathrm{BOB}_{down}, \zeta_{down}\right)$).

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
Chkadvd	0.1	Check advection terms if depth < chkadvd [m].
Teta0	0.55	Theta (implicitness) of time integration [-], 0.5 < Theta < 1.0.
cstbnd	0	Delft3D-FLOW type velocity treatment near boundaries for small coastal models (1) or not (0).
Maxitverticalforestersal	0	Forester iterations for salinity (0: no vertical filter for salinity, > 0: max nr of iterations).
Maxitverticalforestertem	0	Forester iterations for temperature (0: no vertical filter for temperature, > 0: max nr of iterations).
TransportAutoTimestepdiff	0	Auto Timestepdiff in Transport, (0 : lim diff, no lim Dt_tr, 1: no lim diff, lim Dt_tr, 2: no lim diff, no lim Dt_tr, 3: implicit (only 2D)).
Implicitdiffusion2D	0	Implicit diffusion in 2D (0: no, 1:yes).
Turbulencemodel	3	0=no, 1 = constant, 2 = algebraic, 3 = k-epsilon, 4 = k-tau.
Turbulenceadvection	3	0=no, 3 = horizontal explicit vertical implicit.
AntiCreep	0	Include anti-creep to suppress artifical vertical diffusion (0: no, 1: yes).
DiagnosticTransport	0	No update of transport quantities, also known as diagnostic transport (0: no, 1: yes).
Maxwaterleveldiff	0.	Upper bound [m] on water level changes, (<= 0: no bounds). Run will abort when violated.
Maxvelocitydiff	0.	Upper bound [m/s] on velocity changes, (<= 0: no bounds). Run will abort when violated.
Maxvelocity	0.	Upper bound [m/s] on velocity (<= 0: no bounds). Run will abort when violated.
Waterlevelwarn	0.	Warning level [m AD] on water level (<= 0: no check).
Velocitywarn	0.	Warning level [m/s] on normal velocity(<= 0: no check).
Velmagnwarn	0.	Warning level [m/s] on velocity magnitude (<= 0: no check).
MinTimestepBreak	0.	Smallest allowed timestep [s], checked on a sliding average of several timesteps. Run will abort when violated.
Epshu	0.0001	Threshold water depth for wetting and drying [m].
EpsMaxlev	1d-8	Stop criterium for non linear iteration.
EpsMaxlevm	1d-8	Stop criterium for Nested Newton loop in non-linear iteration.
TestDryingFlooding	0	Drying flooding algorithm (0: D-Flow FM, 1: Delft3D-FLOW, 2: Similar to 0, and volume limitation in the transport solver based on Epshu).
[physics]		
UnifFrictCoef	0.023	Uniform friction coefficient (0: no friction) [the unit depends on UnifFrictType].
UnifFrictType	1	Uniform friction type (0: Chezy, 1: Manning, 2: White-Colebrook, 3: White-Colebrook in WAQUA).
UnifFrictCoef1D	0.023	Uniform friction coefficient in 1D links (0: no friction) [the unit depends on UnifFrictType].
UnifFrictCoefLin	0.	Uniform linear friction coefficient (0: no friction) [m/s].
Vicouv	0.1	Uniform horizontal eddy viscosity [m^2/s].
Dicouv	0.1	Uniform horizontal eddy diffusivity [m^2/s].
Vicoww	1.d-6	Background vertical eddy viscosity [m^2/s].

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
`Dicoww`	`1.d-6`	Background vertical eddy diffusivity [m^2/s].
`Vicwminb`	`0.`	Minimum viscosity in production and buoyancy term [m^2/s].
`Xlozmidov`	`0.`	Ozmidov length scale [m], default=0.0, no contribution of internal waves to vertical diffusion.
`Smagorinsky`	`0.2`	Add Smagorinsky horizontal turbulence [-]: vicu = vicu + ((Smagorinsky*dx)**2)*S.
`Elder`	`0.`	Add Elder contribution [-]: vicu = vicu + Elder*kappa*ustar*H/6); e.g. 1.0.
`irov`	`0`	Wall friction, 0=free slip, 1 = partial slip using wall_ks.
`wall_ks`	`0.`	Nikuradse roughness [m] for side walls, wall_z0=wall_ks/30.
`Rhomean`	`1000.`	Average water density [kg/m^3].
`Idensform`	`2`	Density calulation (0: uniform, 1: Eckart, 2: UNESCO, 3=UNESCO83, 13=3+pressure).
`Ag`	`9.81`	Gravitational acceleration [m/s^2].
`TidalForcing`	`0`	Tidal forcing, if jsferic=1 (0: no, 1: yes).
`Doodsonstart`	`55.565`	Doodson start time for tidal forcing [s].
`Doodsonstop`	`375.575`	Doodson stop time for tidal forcing [s].
`Doodsoneps`	`0.0`	Doodson tolerance level for tidal forcing [s].
`VillemonteCD1`	`1.0`	Calibration coefficient for Villemonte [-]. Default = 1.0.
`VillemonteCD2`	`10.0`	Calibration coefficient for Villemonte [-]. Default = 10.0.
`Salinity`	`0`	Include salinity, (0: no, 1: yes).
`InitialSalinity`	`0.`	Initial salinity concentration [ppt].
`Sal0abovezlev`	`-999.`	Salinity 0 above level [m].
`DeltaSalinity`	`-999.`	uniform initial salinity [ppt].
`BackgroundSalinity`	`30.`	Background salinity for eqn. of state if salinity not computed [psu].
`Temperature`	`0`	Include temperature (0: no, 1: only transport, 3: excess model of D3D, 5: composite (ocean) model).
`InitialTemperature`	`6.`	Initial temperature [°C].
`BackgroundwaterTemperature`	`20.`	Background water temperature for eqn. of state if temperature not computed [°C].
`Secchidepth`	`2.`	Water clarity parameter [m].
`Stanton`	`0.0013`	Coefficient [-] for convective heat flux ( ), if negative, then Cd wind is used.
`Dalton`	`0.0013`	Coefficient [-] for evaporative heat flux ( ), if negative, then Cd wind is used.
`SecondaryFlow`	`0`	Secondary flow (0: no, 1: yes).
`BetaSpiral`	`0.`	Weight factor [-] of the spiral flow intensity on flow dispersion stresses (0d0 = disabled).
`BreachGrowth`	`symmetric-asymmetric`	Method for distributing dam breach width over dam break flow links: `symmetric`, `proportional`, or `symmetric-asymmetric`.
`[wind]`		
`ICdtyp`	`2`	Wind drag coefficient type (1: Const, 2: Smith&Banke (2 pts), 3: S&B (3 pts), 4: Charnock 1955, 5: Hwang 2005, 6: Wuest 2005, 7: Hersbach 2010 (2 pts), 8: 4+viscous).
`Cdbreakpoints`	`6.3d-4 7.23d-3`	Wind drag breakpoints [-], e.g. 0.00063 0.00723.

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
`Windspeedbreakpoints`	`0.  100.`	Wind speed breakpoints [m/s], e.g. 0.0 100.0.
`Rhoair`	`1.2`	Air density [kg/m^3].
`computedAirdensity`	`0`	Compute air density (0: no, 1: yes). Requires quantities airpressure, airtemperature and dewpoint in <ext>-file.
`Stresstowind`	`0.`	Switch between Wind speed (=0) and wind stress (=1) approach for wind forcing.
`Relativewind`	`0.`	Wind speed [kg/m^3] factor relative to top-layer water speed*relativewind (0d0=no relative wind, 1d0=using full top layer speed).
`Windpartialdry`	`1`	Reduce windstress on water if link partially dry, only for `BedlevType`=3, 0=no, 1=yes (default).
`PavBnd`	`0.`	Average air pressure on open boundaries [N/m^2], only applied if value > 0.
`PavIni`	`0.`	Initial air pressure [N/m^2], only applied if value > 0.
`[grw]`		
`Infiltrationmodel`	`0`	Infiltration method (0: No infiltration, 1: Interception layer, 2: Constant infiltration capacity, 3: model unsaturated/saturated (with grw), 4: Horton).
`UnifInfiltrationCapacity`	`0`	Uniform maximum infiltration capacity [m/s].
`[hydrology]`		
`InterceptionModel`	`0`	Interception model (0: none, 1: on, via layer thickness).
`[time]`		
`RefDate`	`20010101`	Reference date [yyyymmdd]. By default midnight is taken (00h00m00s)
`Tzone`	`0.`	Data Sources in GMT are interrogated with time in minutes since refdat-Tzone*60 [min].
`Tunit`	`S`	Time units in MDU [D, H, M or S].
`DtUser`	`300.`	User timestep in seconds [s] (interval for external forcing update & his/map output).
`DtNodal`	`21600.`	Time interval [s] for updating nodal factors in astronomical boundary conditions.
`DtMax`	`30.`	Maximum timestep in seconds [s].
`DtInit`	`1.`	Initial timestep in seconds [s].
`TStart`	`0.`	Start time with respect to RefDate [Tunit].
`TStop`	`86400.`	Stop time with respect to RefDate [Tunit].
`StartDateTime`		Computation start datetime (yyyymmddhhmmss), when specified, overrides `TStart`.
`StopDateTime`		Computation stop datetime (yyyymmddhhmmss), when specified, overrides `TStop`.
`UpdateRoughnessInterval`	`86400.`	Update interval for time dependent roughness parameters [s].
`TStartTlfsmo`	`TStart`	Start time with respect to RefDate of Fourier smoothing time on water level boundaries [Tunit].
`[restart]`		
`RestartFile`		Restart file, only from NetCDF-file, hence: either *_rst.nc or *_map.nc.

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
RestartDateTime		Restart time [yyyymmddhhmmss], only relevant but obligatory in case of restart from *_map.nc.
[external forcing]		
ExtForceFile		Old format for external forcings file *.ext, link with tim/cmp-format boundary conditions specification.
ExtForceFileNew		New format for external forcings file *.ext, link with bc-format boundary conditions specification. See section C.5.2.
Rainfall		Include rainfall, (0=no, 1=yes).
QExt		Include user Qin/out, externally provided, (0=no, 1=yes).
Evaporation		Include evaporation in water balance, (0=no, 1=yes).
WindExt		Include wind, externally provided, (0=no, 1=reserved for EC, 2=yes).
[hydrology]		
InterceptionModel	0	Interception model (0: none, 1: on, via layer thickness). See Section 13.3.
[trachytopes]		
TrtRou	N	Flag for trachytopes (Y=on, N=off).
TrtDef		File (*.ttd) including trachytope definitions.
TrtL		File (*.arl) including distribution of trachytope definitions.
DtTrt	60.	Interval for updating of bottom roughness due to trachytopes in seconds [s].
[output]		
Wrishp_crs	0	Writing cross sections to shape file (0=no, 1=yes).
Wrishp_dambreak	0	Writing dambreaks to shape file (0=no, 1=yes).
Wrishp_dryarea	0	Writing dry areas to shape file (0=no, 1=yes).
Wrishp_enc	0	Writing enclosures to shape file (0=no, 1=yes).
Wrishp_emb	0	Writing embankments file (0=no, 1=yes).
Wrishp_fxw	0	Writing fixed weirs to shape file (0=no, 1=yes).
Wrishp_gate	0	Writing gates to shape file (0=no, 1=yes).
Wrishp_genstruc	0	Writing general structures to shape file (0=no, 1=yes).
Wrishp_obs	0	Writing observation points to shape file (0=no, 1=yes).
Wrishp_pump	0	Writing pumps to shape file (0=no, 1=yes).
Wrishp_src	0	Writing sources and sinks to shape file (0=no, 1=yes).
Wrishp_thd	0	Writing thin dams to shape file (0=no, 1=yes).
Wrishp_weir	0	Writing weirs to shape file (0=no, 1=yes).
OutputDir		Output directory of map-, his-, rst-, dat- and timings-files, default: DFM_OUTPUT_<modelname>. Set to . for no dir/current dir.
WAQOutputDir		Output directory of Water Quality files.
FlowGeomFile		*_flowgeom.nc Flow geometry file in NetCDF format.
ObsFile		Space separated list of files, containing information about observation points. See section F.2.2.

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
CrsFile		Space separated list of files, containing information about observation cross sections. See section F.2.4.
FouFile		Name of attribute file that defines the *_fou.nc Fourier output file in NetCDF format, see C.14.
FouUpdateStep	0	Fourier output type: 0 = every user step ; 1 = every computational step ; 2 = equal to his output.
HisFile		*_his.nc History file in NetCDF format.
HisInterval	300.	History output, given as 'interval' 'start period' 'end period' [s].
XLSInterval	0.	Interval between XLS history [s].
MapFile		*_map.nc Map file in NetCDF format.
MapInterval	1200.	Map file output, given as 'interval' 'start period' 'end period' [s].
RstInterval	0.	Restart file output, given as 'interval' 'start period' 'end period' [s].
MapFormat	4	Map file format, 1: NetCDF, 2: Tecplot, 3: NetCDF and Tecplot, 4: NetCDF UGRID.
NcFormat	3	Format for all NetCDF output files (3: classic, 4: NetCDF4+HDF5).
NcMapDataPrecision	0	Precision for NetCDF data in map files (double or single).
NcHisDataPrecision	0	Precision for NetCDF data in his files (double or single).
NcCompression	0	Whether or not (1/0) to apply compression to NetCDF output files - NOTE: only works when NcFormat = 4.
NcNoUnlimited	0	Write full-length time-dimension instead of unlimited dimension (1: yes, 0: no). (Might require NcFormat=4.)
NcNoForcedFlush	0	Do not force flushing of map-like files every output timestep (1: yes, 0: no).
NcWriteLatLon	0	Write extra lat-lon coordinates for all projected coordinate variables in each NetCDF file (for CF-compliancy) (1: yes, 0: no).
Several His file options		See section F.3.1.
Wrihis_balance	1	Write mass balance totals to his file, (1: yes, 0: no).
Wrihis_structure_gen	1	Write general structure parameters to his file, (1: yes, 0: no).
Wrihis_structure_dam	1	Write dam parameters to his file, (1: yes, 0: no).
Wrihis_structure_pump	1	Write pump parameters to his file, (1: yes, 0: no).
Wrihis_structure_gate	1	Write gate parameters to his file, (1: yes, 0: no).
Wrihis_structure_weir	1	Write weir parameters to his file, (1: yes, 0: no).
Wrihis_structure_orifice	1	Write orifice parameters to his file, (1: yes, 0: no).
Wrihis_structure_bridge	1	Write bridge parameters to his file, (1: yes, 0: no).
Wrihis_structure_culvert	1	Write culvert parameters to his file, (1: yes, 0: no).
Wrihis_structure_longculvert	1	Write long culvert parameters to his file, (1: yes, 0: no).
Wrihis_structure_damBreak	1	Write dam break parameters to his file, (1: yes, 0: no).
Wrihis_structure_uniWeir	1	Write universal weir parameters to his file, (1: yes, 0: no).
Wrihis_structure_compound	1	Write compound structure parameters to his file, (1: yes, 0: no).
Wrihis_lateral	1	Write lateral data, (1: yes, 0: no).

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
Wrihis_velocity	1	Write velocity magnitude in observation point to his file, (1: yes, 0: no).
Wrihis_discharge	1	Write discharge magnitude in observation point to his file, (1: yes, 0: no).
Wrihis_sourcesink	1	Write sources-sinks statistics to his file, (1: yes, 0: no).
Wrihis_turbulence	1	Write k, eps and vicww to his file, (1: yes, 0: no).
Wrihis_wind	1	Write wind velocities to his file, (1: yes, 0: no).
Wrihis_rain	1	Write precipitation to his file, (1: yes, 0: no).
Wrihis_airdensity	0	Write air density to his file (1: yes, 0: no).
Wrihis_infiltration	1	Write infiltration to his file, (1: yes, 0: no).
Wrihis_temperature	1	Write temperature to his file, (1: yes, 0: no).
Wrihis_waves	1	Write wave data to his file, (1: yes, 0: no).
Wrihis_heat_fluxes	1	Write heat fluxes to his file, (1: yes, 0: no).
Wrihis_salinity	1	Write salinity to his file, (1: yes, 0: no).
Wrihis_density	1	Write density to his file, (1: yes, 0: no).
Wrihis_waterlevel_s1	1	Write water level to his file, (1: yes, 0: no).
Wrihis_bedlevel	1	Write bed level to his file, (1: yes, 0: no).
Wrihis_waterdepth	0	Write water depth to his file, (1: yes, 0: no).
Wrihis_velocity_vector	1	Write velocity vectors to his file, (1: yes, 0: no).
Wrihis_upward_velocity_component	0	Write upward velocity to his file, (1: yes, 0: no).
Wrihis_sediment	1	Write sediment transport to his file, (1: yes, 0: no).
Wrihis_constituents	1	Write tracers to his file, (1: yes, 0: no).
Wrihis_zcor	1	Write vertical coordinates to his file, (1: yes, 0: no).
Wrihis_taucurrent	1	Write mean bed shear stress to his file, (1: yes, 0: no).
Wrihis_velocity	1	Write velocity magnitude to his file, (1: yes, 0: no).
Several Map file options		See section F.3.2.
Wrimap_waterlevel_s0	1	Write water levels at old time level to map file, (1: yes, 0: no).
Wrimap_waterlevel_s1	1	Write water levels at new time level to map file, (1: yes, 0: no).
Wrimap_evaporation	0	Write evaporation to map file, (1: yes, 0: no).
Wrimap_velocity_component_u0	1	Write velocities at old time level to map file, (1: yes, 0: no).
Wrimap_velocity_component_u1	1	Write velocities at new time level to map file, (1: yes, 0: no).
Wrimap_velocity_vector	1	Write cell-center velocity vectors to map file, (1: yes, 0: no).
Wrimap_upward_velocity_component	0	Write upward velocity component to map file, (1: yes, 0: no).
Wrimap_density_rho	1	Write density to map file, (1: yes, 0: no).
Wrimap_horizontal_viscosity_viu	1	Write horizontal viscosity to map file, (1: yes, 0: no).
Wrimap_horizontal_diffusivity_diu	1	Write horizontal diffusivity to map file, (1: yes, 0: no).
Wrimap_flow_flux_q1	1	Write fluxes to map file, (1: yes, 0: no).
Wrimap_spiral_flow	1	Write spiral flow to map file, (1: yes, 0: no).
Wrimap_numlimdt	1	Write numlimdt to map file, (1: yes, 0: no).
Wrimap_taucurrent	1	Write bottom friction to map file, (1: yes, 0: no).

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
Wrimap_chezy	0	Write chezy roughness in flow elements to map file, (1: yes, 0: no)
Wrimap_chezy_on_flow_links	0	Write chezy roughness on flow links to map file, (1: yes, 0: no)
Wrimap_input_roughness	0	Write chezy input roughness on flow links to map file, (1: yes, 0: no).
Wrimap_turbulence	1	Write turbulence to map file, (1: yes, 0: no).
Wrimap_rain	0	Write rainfall rate to map file, (1: yes, 0: no).
Wrimap_wind	1	Write winds to map file, (1: yes, 0: no).
Wrimap_airdensity	0	Write air density to map file, (1:yes, 0:no).
Writek_CdWind	0	Write wind friction coefficients to tek file (1: yes, 0: no).
Wrimap_heat_fluxes	0	Write heat fluxes to map file, (1: yes, 0: no).
Wrimap_fixed_weir_energy_loss	0	Write energy losses of fixed weirs to map file, (1: yes, 0: no).
Wrimap_wet_waterdepth_threshold	2d-5	Waterdepth threshold above which a grid point counts as 'wet'. Defaults to $0.2 \cdot$Epshu. It is used for Wrimap_time_water_on_ground, Wrimap_waterdepth_on_ground and Wrimap_volume_on_ground.
Wrimap_time_water_on_ground	0	Write cumulative time when water is above ground level (only for 1D nodes) to map file (1: yes, 0: no).
Wrimap_freeboard	0	Write freeboard (only for 1D nodes) to map file (1: yes, 0: no).
Wrimap_waterdepth_on_ground	0	Write waterdepth that is above ground level to map file (only for 1D nodes) (1: yes, 0: no).
Wrimap_volume_on_ground	0	Write volume that is above ground level to map file (only for 1D nodes) (1: yes, 0: no).
Wrimap_total_net_inflow_1d2d	0	Write current total 1D2D net inflow (discharge) and cumulative total 1D2D net inflow (volume) to map file (only for 1D nodes) (1:yes, 0:no).
Wrimap_total_net_inflow_lateral	0	Write current total lateral net inflow (discharge) and cumulative total lateral net inflow (volume) to map file (only for 1D nodes) (1:yes, 0:no).
Wrimap_water_level_gradient	0	Write water level gradient to map file (only for 1D links) (1:yes, 0:no).
Wrimap_flow_analysis	0	Write flow analysis data to the map file (1:yes, 0:no).
Wrimap_volume1	0	Write volumes to map file (1: yes, 0: no).
Wrimap_waterdepth	1	Write water depths to map file (1: yes, 0: no).
Wrimap_waterdepth_hu	0	Write water depths on u-points to map file (1: yes, 0: no).
Wrimap_ancillary_variables	0	Write ancillary variables attributes to map file (1: yes, 0: no).
Wrimap_flowarea_au	0	Write flow areas au to map file (1: yes, 0: no).
Wrimap_velocity_magnitude	1	Write cell-center velocity vector magnitude to map file (1: yes, 0: no).
Wrimap_velocity_vectorq	0	Write cell-center velocity vectors (discharge-based) to map file (1: yes, 0: no).
Wrimap_flow_flux_q1_main	0	Write flow flux in main channel to map file (1: yes, 0: no).
Wrimap_interception	0	Write interception to map file (1: yes, 0: no).
Wrimap_windstress	0	Write wind stress to map file (1: yes, 0: no).
Wrimap_CdWind	1	Write wind friction coeffs to tek file (1: yes, 0: no).
Wrimap_DTcell	0	Write time step per cell based on CFL (1: yes, 0: no).

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
Wrimap_bnd	0	Write boundary points to map file (1: yes, 0: no).
Wrimap_Qin	0	Write sum of all influxes to map file (1: yes, 0: no).
Wrimap_every_dt	0	Write output to map file every computational timestep, between start and stop time from MapInterval, (1: yes, 0: no).
MapOutputTimeVector		File (.mpt) containing fixed map output times (s) w.r.t. RefDate.
FullGridOutput	0	Full grid output mode for layer positions (0: compact, 1: full time-varying grid layer data).
EulerVelocities	0	Write Eulerian velocities, (1: yes, 0: no).
ClassMapFile		Name of class map file.
WaterlevelClasses	0.	Series of values between which water level classes are computed.
WaterdepthClasses	0.	Series of values between which water depth classes are computed.
ClassMapInterval	0	Interval [s] between class map file outputs.
WaqInterval	0.	Interval [s] between DELWAQ file outputs.
StatsInterval	-60.	Interval [s] between screen step outputs in seconds simulation time, if negative in seconds wall clock time.
TimingsInterval	0.	Timings output interval TimingsInterval.
Richardsononoutput	0	Write Richardson number, (1: yes, 0: no).

A.2 Overview of additional keywords for 3D Modelling

In the current D-Flow FM Graphical User Interface not all keywords can be specified for 3D modelling yet. Currently, this involves the keywords Vicoww, Dicoww, Turbulencemodel, Turbulenceadvection, Anticreep. This means that still some keywords for 3D modelling have to be specified manually. These keywords are shown in Table A.2. If a keyword has a unit, then this unit will also be given in [ ] in the column of Description in this table.

Table A.2: *Keywords in MDU-file for 3D modelling not in the GUI yet.*

Keyword	Default setting	Description
[geometry]		
ZlayBot	-999	if specified, first z-layer starts from zlaybot [ ], if not, it starts from the lowest bed point.
ZlayTop	-999	if specified, highest z-layer ends at zlaytop [ ], if not, it ends at the initial water level.
StretchType	1	Stretching type for non-uniform layers, 1=user defined, 2=exponential, otherwise=uniform.
StretchCoef	1	coefficients for sigma layer, 1: Percentages of the layers, user defined, laycof(kmx), 2: Stretching level, and two coefficients for layers growth, laycof(3).
[numerics]		
HorizontalMomentumfilter	0	Filter for reduction of checkerboarding; 0=No, 1=yes.
Checkerboardmonitor	0	Flag for checkerboarding output on history file (only for sigma layers yet); 0=No, 1=yes.
Tspinupturblogprof	0.0	Spin up time [s] when starting with a parabolic viscosity profile in whole model domain.

(continued on next page)

(continued from previous page)

Keyword	Default setting	Description
`Vertadvtypmom`	6	Vertical advection type in momentum equation; 3: Upwind implicit, 6: centerbased upwind explicit.
`Vertadvtypsal`	6	Vertical advection type for salinity (0: none, 4: Theta implicit, 6: higher order explicit, no Forester filter).
`Vertadvtyptem`	6	Vertical advection type for temperature (0: none, 4: Theta implicit, 6: higher order explicit, no Forester filter).
`Zerozbndinflowadvection`	0	Switch for advection at open boundary (0: Neumann, 1=zero at inflow, 2=zero at inflow and outflow).

A.3 Overview of research keywords

In Table A.1 and Table A.2 in this appendix an overview of keywords in the master definition file is given that can be specified by the user or should be added manually, respectively.

In addition there are several research keywords in the computational kernel of D-Flow Flexible Mesh that should, in principle, not be used and not be changed by the user. These keywords haven't been documented yet and aren't tested in the D-Flow FM test benches yet. Therefore, using the default setting of these research keywords is strongly recommended. Because these keywords are listed in the diagnostic file, for the sake of completeness they are listed in the table below and also in Section 5.12 of the D-Flow FM Technical Reference Manual. If a keyword has a unit, then this unit will also be given in [ ] in the column of Description in this table.

Table A.3: *Overview of numerical research parameters with compulsory defaults.*

Keyword	Default setting	Description
`[numerics]`		
`Barocterm`	4	Various options for baroclinic pressure term.
`BaroctimInt`	4	Time integration baroclinic pressure, 1 = explicit, 4 = Adams Bashforth + dryfloodproof.
`Cffacver`	0.	Factor for including (1-CFL) in HO term vertical (0d0: no, 1d0: yes).
`Corioadamsbashfordfac`	0.5	Adams Bashforth factor for Coriolis term.
`Drop3D`	1	Apply drop losses in 3D if z upwind is below bob + 2/3 hu*drop3D (0: no, 1: yes).
`Horadvtypzlayer`	0	Horizontal advection treatment of z-layers for dam breaks (0: default, 2: sigma-like).
`Icoriolistype`	5	0=No, 1=yes, if jsferic then spatially varying, 2-5: ..., if icoriolistype==6 then constant (anglat).
`Jbasqbnddownwindhs`	0	0=original hu on qbnd, 1=downwind hs on qbnd.
`Filterorder`	2	First-order or second order filter to suppress checkerboarding.
`Keepstbndonoutflow`	0	Keep salinity and temperature signals on boundary also at outflow, (1: yes, 0: no) (copy inside value on outflow).
`Keepzlayeringatbed`	2	Z layering at bed: 0=original bed level, 1=adapted bed level, 2= Ztbml approach of Delft3D-FLOW.
`Logprofatubndin`	1	ubnds inflow: 0=uniform U1, 1 = log U1, 2 = user3D.
`Logprofkepsbndin`	0	inflow: 0=0 keps, 1 = log keps, 2 = user3D.
`Newcorio`	1	Temporary keyword for Coriolis improvement on open boundary; 0=No, 1=yes.

(continued on next page)

Keyword	Preferred setting	Description
		(continued from previous page)
Turbulenceadvection	3	Turbulence advection (0: none, 1=Upwind explicit, 2=Central explicit, 3: horizontally explicit and vertically implicit, 4=Central implicit.
Windhuorzwsbased	0	Wind approach; 0= Hu-based (Delft3D-FLOW), 1= finite volume based.
[calibration]		Roughness calibration. Details in Chapter 16.
UseCalibration	0	Activate calibration factor friction multiplier (0: no, 1: yes).
DefinitionFile		File (*.cld) containing calibration definitions. Details in Section C.9.1.
AreaFile		File (*.cll) containing area distribution of calibration definitions. Details in Section C.9.2.
[processes]		Online water quality processes. Details in Chapter 20.
SubstanceFile		Substance file name. Details in Section 20.2.1.
AdditionalHistoryOutputFile		Extra history output filename. Details in Section 20.5.1.
StatisticsFile		Statistics definition file. Details in Section 20.5.3.
ThetaVertical	0.0	Theta value for vertical transport of water quality substances [-].
DtProcesses	0.0	waq processes time step [s]. Must be a multiple of DtUser. If DtProcesses is negative, water quality processes are calculated with every hydrodynamic time step.
ProcessFluxIntegration	1	Process fluxes integration option (1: WAQ, 2: D-Flow FM).
Wriwaqbot3Doutput	0	Write 3D water quality bottom variables (0: no, 1: yes).
VolumeDryThreshold	1d-3	Volume [m^3] below which segments are marked as dry. Details in Section 20.2.3.
DepthDryThreshold	1d-3	Water depth [m] below which segments are marked as dry. Details in Section 20.2.3.
[veg]		Dynamic vegetation model, more details in Section 15.3.
Vegetationmodelnr	0	Vegetation model nr, (0: no, 1: Baptist DFM).
Clveg	0.8	Stem distance factor [-].
Cdveg	0.7	Stem Cd coefficient [-].
Cbveg	0.0	Stem stiffness coefficient [-].
Rhoveg	0.0	Stem Rho, if > 0, bouyant stick procedure [kg/m^3].
Stemheightstd	0.0	Stem height standard deviation fraction, e.g. 0.1 [-].
Densvegminbap	0.0	Minimum vegetation density in Baptist formula. Only in 2D. [1/m^2].
If expchistem < 0		
Expchistem	0	[-]
Expchileaf	0	[-]
Uchistem	0	[m/s]
		(continued on next page)

(continued from previous page)

Keyword	Preferred setting	Description
Uchileaf	0	[m/s]
Arealeaf	0	[m^2]
Cdleaf	0	[-]

A.4 Overview of deprecated and removed keywords

The following table lists in alphabetical order per data block the deprecated keywords (which are currently still supported by D-Flow FM, but may be removed in a future release) and the removed keywords (which are no longer processed). When the functionality of the keyword has is now supported via another keyword then the new keyword is specified. Check the description of the new keyword in Table A.1 for more details on how to use it.

Table A.4: *Overview of deprecated and removed keywords.*

<table>
<tr><th>Keyword</th><th>Action</th></tr>
<tr><td>[geometry]</td><td></td></tr>
<tr><td>BathymetryFile</td><td>Removed since March 2022.
See [geometry] keyword BedLevelFile in this table.</td></tr>
<tr><td>BedLevelFile</td><td>Removed since March 2022.
Use [geometry] keyword IniFieldFile instead. That keyword should point to a file with the following content where my_bedlevel_data.xyb is the name of your bed level samples file which was originally referenced by the removed keyword.
<pre>
[General]
    fileVersion = 2.00
    fileType   = iniField

[Initial]
    quantity   = bedlevel
    dataFile   = my_bedlevel_data.xyb
    dataFileType   = sample
    interpolationMethod = triangulation
</pre></td></tr>
<tr><td>BotLevUni</td><td>Removed since March 2022.
Use [geometry] keyword BedLevUni instead.</td></tr>
<tr><td>BotLevType</td><td>Removed since March 2022.
Use [geometry] keyword BedLevType instead.</td></tr>
<tr><td>ManholeFile</td><td>Removed since March 2022.
Use [geometry] keyword StorageNodeFile instead.</td></tr>
<tr><td>NoOptimizedPolygon</td><td>Removed since March 2022. Option no longer supported.</td></tr>
<tr><td>ThindykeFile</td><td>Removed since March 2022. Use [geometry] keyword FixedWeirFile instead.</td></tr>
</table>

(continued on next page)

Keyword	Action
	(continued from previous page)
`old format of general structure`	Removed since October 2023. Use new keywords instead. The old/new keywords are: widthleftW1/Upstream1Width, levelleftZb1/Upstream1Level, widthrightW2/Upstream2Width, levelrightZb2/Upstream2Level, widthleftWsdl/Downstream1Width, levelleftZbsl/Downstream1Level, widthrightWsdr/Downstream2Width, levelrightZbsr/Downstream2Level, widthcenter/CrestWidth, levelcenter/CrestLevel, gateheight/GateLowerEdgeLevel, gatedoorheight/GateHeight, door_opening_width/GateOpeningWidth, lower_edge_level/GateLowerEdgeLevel, opening_width/GateOpeningWidth, sill_level/CrestLevel, door_height/GateHeight, horizontal_opening_direction/GateOpeningHorizontalDirection, crest_level/CrestLevel.
`[numerics]`	
`Hkad`	Removed since March 2022. Option no longer supported.
`IThindykeScheme`	Removed since March 2022. Use `[numerics]` keyword `FixedWeirScheme` instead.
`ThindykeContraction`	Removed since March 2022. Use `[numerics]` keyword `FixedWeirContraction` instead.
`TransportTimeStepping`	Removed since August 2022. Option no longer supported.
`TransportMethod`	Removed since August 2022. Value 1 is the default and only available option, no further input needed. Value 2 for diagnostic transport is now under its own keyword. Use `[numerics]` keyword `DiagnosticTransport=1` instead.
`[output]`	
`WriteBalanceFile`	Removed since March 2022. Superseded by default information included in the history file.

B The net file for flexible meshes

The net file contains the full computational grid (mesh) information. It is a netCDF file that can contain 2D grids, as well as 1D networks, and combinations of these. The netCDF conventions used for this file are:

- CF-conventions ($\geq$ 1.7) and UGRID-1.0 for the 2D grids. More information on: http://ugrid-conventions.github.io/ugrid-conventions/.
- A Deltares proposal for an extension to the UGRID convention for 1D network specifics. This proposal follows in section B.2. It is part of the Deltares ($\geq$ 0.10) conventions. This proposal builds on the simple geometries concept introduced in CF-conventions version 1.8.
- As a result the global `Conventions` attribute will typically read `CF-1.8 UGRID-1.0 Deltares-0.10`.

B.1 2D grids in netCDF UGRID files

D-Flow FM reads net files that adhere to the 2D rules in the UGRID conventions. At the minimum this means that the file must contain the mesh node coordinates and for all grid cells ('faces') the corner node numbers in the face_node_connectivity table.

B.2 Proposal for netCDF conventions for 1D networks

This section contains the Deltares proposal for conventions for netCDF datasets containing 1D networks (graphs) and data defined on these networks. It is part of the Deltares-0.11 conventions. These newly proposed conventions should currently be seen as an extension on top of the regular CF-conventions ($\geq$1.8) and, in particular, on top of the UGRID (1.0) conventions. Key new aspects are:

- Definition of 1D mesh locations in a *network coordinate space*, instead of a regular two-dimensional coordinate space (e.g., Cartesian).
- Definition of the network branch geometries adhering to the proposal for geometries in the CF-1.8 conventions[1].

B.2.1 History

- Deltares-0.11 (2020-03-20): New `flag_bounds` attribute for packed data (Section B.2.11.9).
- Deltares-0.10 (2019-08-21): Consistent renaming of variables and attribute names, singular instead of plural.
- Deltares-0.9 (2019-04-25): Duplicate mesh nodes on network (connection) nodes discouraged.
- Deltares-0.8 (2017-04-21): Initial version for 1D network extension to regular UGRID.

B.2.2 Naming conventions

The 1D domains considered in this proposal are mainly described on two levels:

1. The *network* is a 1D subspace (typically in a 2D or 3D space) defined by a 1D network topology and geometry.
2. The *1D mesh* is a discrete mesh defined in the network coordinate space.

Both two levels can to a large extent be described following the UGRID conventions. As a result, the UGRID naming conventions are followed, amongst others:

[1] https://github.com/cf-convention/cf-conventions/blob/master/ch07.adoc#geometries

- ◇ A *node* is a point, possibly with some edges connected to it.
- ◇ An *edge* is a connection between two nodes.

Occasionally the term *branch* is used in this proposal. This is not an officially proposed attribute nor keyword, but is often shorthand notation for: a branch is an edge in the network topology. In hydrodynamics, the network domain is often defined by several branches, and the computational 1D mesh points are placed on these branches. All proposed netCDF variable attributes will continue to use the word *edge*, though.

B.2.3 Requirements 1D Network

1. In this document, all indexes are 1-based, but this can also be 0-based, specified by the UGRID attribute `start_index`.
2. Mesh nodes need not necessarily be numbered increasingly along a network branch, see Figure B.4.
3. Mesh edges need not necessarily be numbered increasingly along a network branch, see Figure B.4.

B.2.4 1D Network: Example based on Figure B.1

A simple example is given in this section, Figure B.1. Also the networks given in section B.2.6 need to be supported, with special attention to Figure B.4.

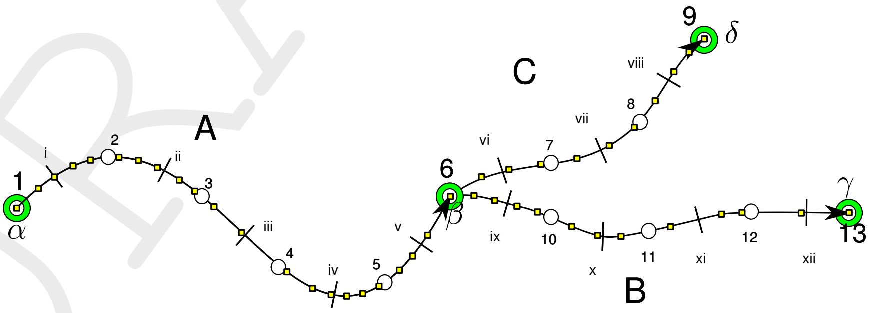


Figure B.1: *Simple computational 1D network:*
3 network edges (A, B, C),
4 network nodes (green circles, α, β, γ, δ),
46 network geometry nodes (yellow squares),
13 1D mesh nodes defined on the network (white circles, latin numbers),
12 1D mesh edges defined on the network (roman numbers)

B.2.4.1 Dimensions

network1d_nNodes = 4; network nodes α, β, γ, δ
network1d_nEdges = 3; branches A, B, C
network1d_nGeometryNodes = 46; yellow squares per branch (including both end points)
mesh1d_nNodes = 13 (13=6+5+4-(3-1); 3 branches are connected to node 6)
mesh1d_nEdges = 12 (12=5+4+3); mesh edges are between mesh nodes

```
dimensions:
    network1d_nNodes = 4 ;
    network1d_nEdges = 3 ;
    network1d_nGeometryNodes = 46 ;
    mesh1d_nNodes = 13 ;
    mesh1d_nEdges = 12 ;
    time = UNLIMITED ; // (2 currently)
    Two = 2 ;
```

B.2.4.2 1D-Network topology

The 1D network topology serves as the coordinate space in which a 1D mesh discretization will later be defined. The network is largely based on the UGRID conventions for its topology (i.e., nodes and edges) and additionally uses an optional `edge_geometry` to define the precise network branch geometries (more about this in the next Section).

```
uint network1d ;
    network1d:cf_role = "mesh_topology" ;
    network1d:edge_dimension = "network1d_nEdges" ;
    network1d:edge_geometry = "network1d_geometry" ;
    network1d:edge_length = "network1d_edge_length" ;
    network1d:edge_node_connectivity = "network1d_edge_nodes" ;
    network1d:long_name = "Network topology" ;
    network1d:node_coordinates = "network1d_node_x network1d_node_y" ;
    network1d:node_dimension = "network1d_nNodes" ;
    network1d:topology_dimension = 1 ;
```

network1d_node_x/y

```
double network1d_node_x(network1d_nNodes) ;
    network1d_node_x:standard_name = "projection_x_coordinate" ;
    network1d_node_x:long_name = "x-coordinate of network nodes" ;
    network1d_node_x:units = "m" ;
double network1d_node_y(network1d_nNodes) ;
    network1d_node_y:standard_name = "projection_y_coordinate" ;
    network1d_node_y:long_name = "y-coordinate of network nodes" ;
    network1d_node_y:units = "m" ;
```

```
network1d_node_x = -187.96667, 2195.7333, 4071.4928, 3445.4246 ;
network1d_node_y = 720.81667, 708.71667, 690.94861, 1540.1838 ;
```

network1d_edge_nodes

```
uint network1d_edge_nodes(network1d_nEdges, Two) ;
    network1d_edge_nodes:cf_role = "edge_node_connectivity" ;
    network1d_edge_nodes:long_name = "Start and end nodes of network edges" ;
    network1d_edge_nodes:start_index = 1 ;
```

```
 network1d_edge_nodes =
  1, 2,
  2, 3,
  2, 4 ;
```

network1d_edge_length

```
double network1d_edge_length(network1d_nEdges) ;
    network1d_edge_length:long_name = "Real length of branch geometries" ;
    network1d_edge_length:unit = "m" ;
```

```
network1d_edge_length = 2500, 2100, 1600 ;
```

B.2.4.3 1D-Network geometry

The network topology essentially defines a graph. Additionally, the actual geometrical shapes of connecting branches may be defined. This is done using the new geometry descriptions as introduced in CF-1.8[2]. The full network geometry typically consists of multiple *geometries* (i.e., branches) and each of them is defined by a list of geometrical *points*.

```
uint network1d_geometry ;
    network1d_geometry:geometry_type = "line" ;
    network1d_geometry:long_name = "1D Geometry" ;
    network1d_geometry:node_coordinates = "network1d_geom_x network1d_geom_y" ;
    network1d_geometry:node_count = "network1d_geom_node_count" ;
```

network1d_node_count

```
uint network1d_geom_node_count(network1d_nEdges) ;
    network1d_geom_node_count:long_name = "Number of geometry nodes per branch" ;
```

```
network1d_geom_node_count = 22, 13, 11 ;
```

This variable is needed, because all geometry is stored in contiguous ragged arrays. So the branches contain the following 46 geometry points (including the end points):

```
Branch A: 1, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
          18, 19, 20, 21, 22, 23, 24, 2;
Branch B: 2, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 3;
Branch C: 2, 36, 37, 38, 39, 40, 41, 42, 43, 44, 4.
```

Note that each list of geometry nodes also includes both end points of the branch, this implies that the connection node 2 appears three times in this array of 46 entries.

network1d_geom_x/y

```
double network1d_geom_x(network1d_nGeometryNodes) ;
    network1d_geom_x:standard_name = "projection_x_coordinate" ;
    network1d_geom_x:long_name = "x-coordinate of branch geometry nodes" ;
    network1d_geom_x:units = "m" ;
double network1d_geom_y(network1d_nGeometryNodes) ;
    network1d_geom_y:standard_name = "projection_y_coordinate" ;
    network1d_geom_y:long_name = "y-coordinate of branch geometry nodes" ;
    network1d_geom_y:units = "m" ;
```

```
network1d_geom_x = -187.96667, -127.96887, -53.84786, 56.174104, 187.13333,
    352.5, 529.96667, 683.23333, 828.43333, 892.96667, 993.8, 1082.5333,
```

[2] https://github.com/cf-convention/cf-conventions/blob/master/ch07.adoc#geometries

```
    1219.6667, 1405.2, 1492.2591, 1598.5306, 1712.2164, 1818.4835, 1912.3747,
    2007.9644, 2094.9, 2195.7333, 2195.7333, 2269.2456, 2381.5612, 2460.2412,
    2577.8276, 2774.8643, 2911.5187, 3029.1051, 3235.6759, 3438.7319,
    3582.0791, 3817.2519, 4071.4928, 2195.7333, 2275.9165, 2377.6129,
    2539.6914, 2739.9061, 2901.9847, 3041.8172, 3140.3355, 3226.4949,
    3321.8353, 3445.4246 ;
network1d_geom_y = y_1, ..., y_46
```

Note that the bold numbers are the begin and end nodes of a geometry.

B.2.4.4 Numerical discretization

The actual 1D mesh dataset — which goes as input into a model run, or results as output from a model run — is the discretized 1D mesh defined on the previously described 1D network. The definition is largely based on the UGRID conventions, but additionally uses location specification in the network 1D coordinate space using branch indexes and offsets.

```
uint mesh1d ;
    mesh1d:cf_role = "mesh_topology" ;
    mesh1d:long_name = "Topology data of 1D mesh" ;
    mesh1d:coordinate_space = "network1d" ;
    mesh1d:edge_dimension = "mesh1d_nEdges" ;
    mesh1d:edge_node_connectivity = "mesh1d_edge_nodes" ;
    mesh1d:edge_coordinates = "mesh1d_edge_branch mesh1d_edge_offset" ;
    mesh1d:node_coordinates = "mesh1d_node_branch mesh1d_node_offset" ;
    mesh1d:node_dimension = "mesh1d_nNodes" ;
    mesh1d:topology_dimension = 1 ;
```

mesh1d_edge_nodes

```
uint mesh1d_edge_nodes(mesh1d_nEdges, Two) ;
    mesh1d_edge_nodes:long_name = "Start and end nodes of mesh edges" ;
    mesh1d_edge_nodes:start_index = 1 ;
```

```
mesh1d_edge_nodes =
  1, 2,
  2, 3,
  3, 4,
  4, 5,
  5, 6,
  6, 7,
  7, 8,
  8, 9,
  6, 10,
  10, 11,
  11, 12,
  12, 13 ;
```

mesh1d_edge_branch

```
uint mesh1d_edge_branch(mesh1d_nEdges) ;
    mesh1d_edge_branch:long_name = "Index of branch on which mesh edges are located" ;
    mesh1d_edge_branch:start_index = 1 ;
```

```
mesh1d_edge_branch = 1, 1, 1, 1, 1, 3, 3, 3, 2, 2, 2, 2 ;
```

mesh1d_node_branch and mesh1d_node_offset

```
uint mesh1d_node_branch(mesh1d_nNodes) ;
    mesh1d_node_branch:long_name = "Index of branch on which mesh nodes are located" ;
    mesh1d_node_branch:start_index = 1 ;

double mesh1d_node_offset(mesh1d_nNodes) ;
    mesh1d_node_offset:_FillValue = 0. ;
    mesh1d_node_offset:long_name = "Offset along branch of mesh nodes" ;
    mesh1d_node_offset:units = "m" ;
```

```
mesh1d_node_branch = 1, 1, 1, 1, 1, 1, 3, 3, 3, 2, 2, 2, 2;
mesh1d_node_offset = 0, 500, 1000, 1500, 2000, 2500, 700, 1400, 2100, 400,
    800, 1200, 1600 ;
```

B.2.4.5 Duplicate or unique points

The association of discrete mesh locations with branch indexes needs to be unique in the dataset, but the network may allow multiple possible choices. Referring to Figure B.1, consider mesh point 6. It lies on the connection node between branches A, B and C. This proposal does not prescribe any particular choice for which branch index should be used for point 6. The application should choose this. The location at the end (offset) of branch A is equivalent to the start of branch B as well as the start of branch C. If the application needs duplicated mesh points on network connection nodes, the dataset should explicitly contain all these separate points.

In a similar way, the connectivity of mesh nodes is not defined in any special way by our proposal. Two mesh nodes are connected if and only if they appear as a tuple in the edge_node_connectivity table, as defined by the UGRID conventions.

Another related topic is the association of mesh edges with network branches. For example, when data is defined on mesh edges (e.g., flow rates) the dataset must contain two edge location variables, defining the network edge index and offset for each mesh edge. These variables can be referred to in the mesh's `edge_coordinates` attribute as well as the variable's `coordinates` attribute. Our proposal does *not* allow the association of mesh edges to network edges via an implicitly inferred way from the association of mesh nodes to network edges. Figure B.5 shows the need for this in branches B and D: the two distinct flow rates through the two u-points (one on each branch) can not be inferred from the surrounding mesh nodes, as the begin and end points of these two branches are the same.

B.2.5 1D Network, 1D2D Links as mesh contacts, 2D Meshes; Example based on Figure B.2

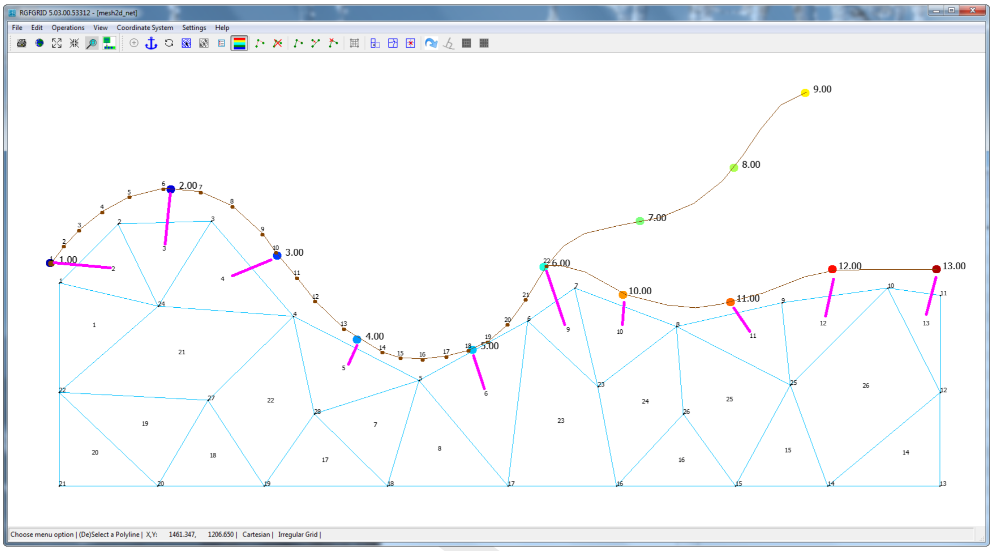

RGFGRID 5.03.00.53312 - [mesh2d_net]
File Edit Operations View Coordinate System Settings Help
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00
13.00
Choose menu option | (De)Select a Polyline | X,Y: 1461.347, 1206.650 | Cartesian | Irregular Grid |

Figure B.2: Composite mesh (1D Network, 1D2D Links and 2D Meshes),
1D Mesh*: coloured dots (=computational 1D mesh), ζ-points; numbered dots, 1d network geometry for branch A (Figure B.1).*
2D Mesh*: light blue with node and face numbers.*
1D2D Link*: links are pink coloured.*
Python script: `create_ugrid_1d2d_map.py`

B.2.5.1 1D Network

See section B.2.4

B.2.5.2 1D2D Links as mesh contacts

1D2D Links may be defined between the computational nodes of the 1D Network (ζ-points) and the faces of the 2D Meshes (ζ-points) using the concept of *mesh contacts* from a Deltares-proposed UGRID-extension[3], see the attribute `contact` of the link1d2d-array below.

```
uint composite_mesh ;
    composite_mesh:cf_role = "parent_mesh_topology" ;
    composite_mesh:meshes = "mesh1d mesh2d" ;
    composite_mesh:mesh_contact = "link1d2d" ;
uint link1d2d(nLink1D2D_edge, Two) ;
    link1d2d:cf_role = "mesh_topology_contact" ;
    link1d2d:contact = "mesh1d:node mesh2d:face" ;
    link1d2d:start_index = 1 ;
```

Table B.1: *1D2D Link table based on Figure B.2*

mesh1d:node	mesh2d:face
1	2
2	3
3	4

[3] https://publicwiki.deltares.nl/display/NETCDF/Mosaics+of+meshes

4	5
5	6
6	9
10	10
11	11
12	12
13	13

B.2.6 Figures with example numbering of nodes and edges

For each of the figures in this section a netCDF-file is available and also the corresponding Python script to generate the netCDF4 file.

- Figure B.2. Script `create_ugrid_1d2d_map.py`.
- Figure B.3. Script `create_ugrid_1d_network_mesh_1_node_edge.py`.
- Figure B.4. Script `create_ugrid_1d_network_mesh_2_node_edge.py`.
- Figure B.5. Script `create_ugrid_1d_network_mesh_3_node_edge.py`.
- Figure B.6. Script `create_ugrid_1d_network_mesh_4_node_edge.py`.
- Figure B.7. Script `create_ugrid_1d_network_mesh_5_node_edge.py`.
- Figure B.8. Script `create_ugrid_1d_network_mesh_6_node_edge.py`.

The branch lengths of a network are given by the array `edge_length` and can not fully determined by the `node_offset` array, especially as several branches are connect between the same network nodes, see Figures B.4 to B.8. The Figures B.5 to B.8 does contain branches without intermediate nodes, so some branches does only have a begin- and an end-node.

The vertical bars in the figures mark the location of the middle of an edge indicating the velocity point for a staggered grid as used by Deltares hydrodynamic modules D-Flow 1D and D-Flow FM.

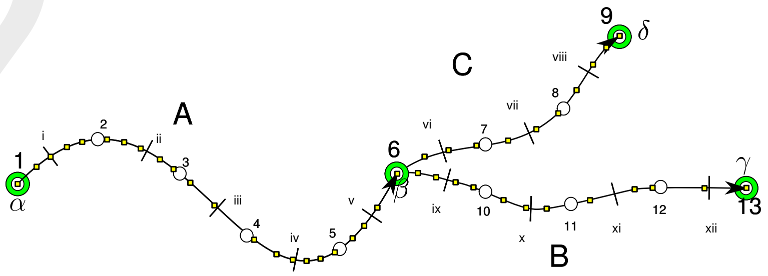


Figure B.3: Simple 1D network and mesh, all branches do have intermediate nodes; 3 branches (A, B, C), 4 network nodes (green circles, α, β, γ, δ), 13 nodes.

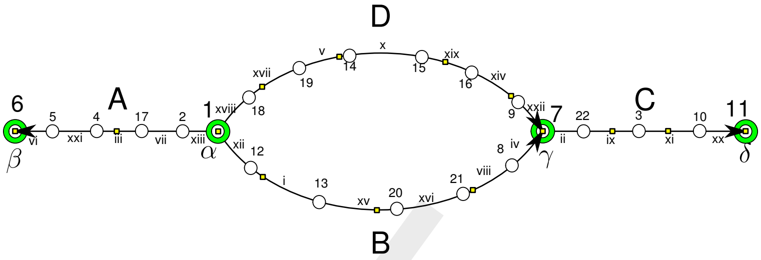


Figure B.4: *Simple 1D network and mesh, all branches do have intermediate nodes; 4 branches (A, B, C, D), 4 network nodes (green circles, α, β, γ, δ), 22 nodes (latin numbers), 22 edges (roman numbers).*

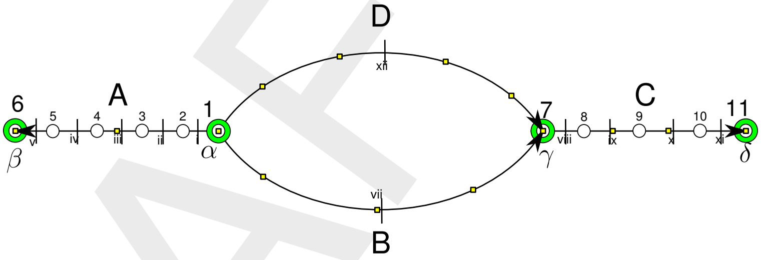


Figure B.5: *Simple 1D network and mesh, 2 branches do not have intermediate nodes (so only end points); 4 branches (A, B, C, D), 4 network nodes (green circles, α, β, γ, δ), 11 nodes (latin numbers), 12 edges (roman numbers).*

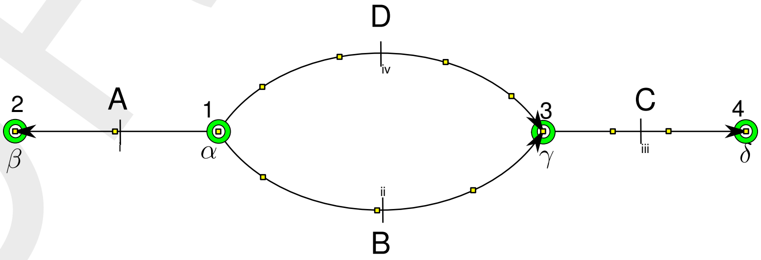


Figure B.6: *Simple 1D network and mesh, all branches do not have intermediate nodes (so only end points); 4 branches (A, B, C, D), 4 network nodes (green circles, α, β, γ, δ), 4 nodes (latin numbers), 4 edges (roman numbers).*

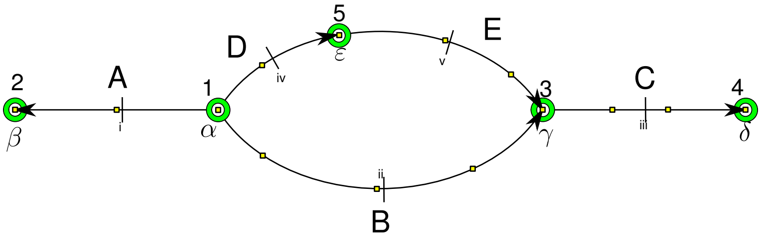


Figure B.7: *Simple 1D network and mesh, all branches do not have intermediate nodes (so only end points) and connection node ε does only have two branches connected. 5 branches (A, B, C, D, E), 5 network nodes (green circles, α, β, γ, δ, ε), 5 nodes (latin numbers), 5 edges (roman numbers).*
Network node 5 is special because it has only two edges attached. This is the same number as for a node which is not a network node.

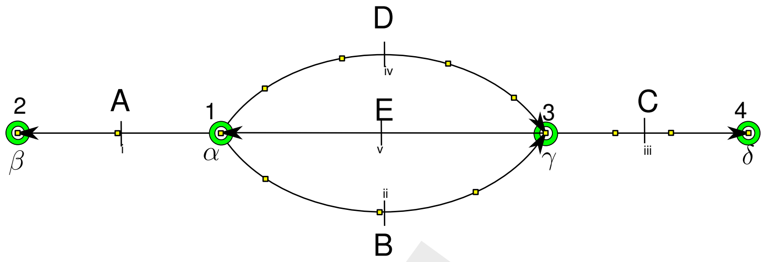


Figure B.8: *Simple 1D network and mesh, all branches do not have intermediate nodes (so only end points) and branch B, D and E have the same begin and end nodes. 5 branches (A, B, C, D, E), 4 network nodes (green circles, α, β, γ, δ), 4 nodes (latin numbers), 5 edges (roman numbers).*

In Figure B.8 the edges II, IV and V are special because they have the same begin and end node and should be separately identified by the array `mesh1d_edge_branch` array and also the node offset is not given in the array `mesh1d_node_offset` for these begin- and end-points. In the `mesh1d_node_offset` array the offset are given for the points 1 (α) and 3 (γ) which belongs to branch A and branch C.

B.2.7 Data defined on 1D meshes

Datasets that contain data variables defined on 1D meshes should define for each variable:

◇ The name of the mesh variable that defines the mesh, via the `mesh` attribute.
◇ The topological location on which the discrete data values are defined, via the `location` attribute. Allowed values are: `"node"` and `"edge"`.

This approach comes directly from the UGRID-conventions and is sufficient already.

As a result, the standard does not have to prescribe anything about assumed locations for any particular data variable. For example in hydrodynamics, relevant variables are water levels and velocities. It is up to the data producer (e.g., the simulation software) to define whether these are defined on nodes or edges, staggered or collocated. Below is an example definition of water level results on nodes (ζ-points) and velocities on edges (u-points).

```
...
double mesh1d_s1(time, mesh1d_nNodes) ;
    mesh1d_s1:location = "node" ;
    mesh1d_s1:long_name = "Water level" ;
    mesh1d_s1:mesh = "mesh1d" ;  //  this is UGRID location pointing to mesh1d topology
    mesh1d_s1:standard_name = "sea_surface_height_above_geoid" ;
    mesh1d_s1:units = "m" ;
double mesh1d_u(time, mesh1d_nEdges) ;
    mesh1d_u:location = "edge" ;
    mesh1d_u:long_name = "velocity along branch" ;
    mesh1d_u:mesh = "mesh1d" ;
    mesh1d_u:standard_name = "sea_water_speed" ;
    mesh1d_u:units = "m s-1" ;
...
```

B.2.8 1D Network history nc-file

To be added station locations, see for an example the D-Flow FM history file.

```
...
double s1(time=ntimes, stations=nstations);
    coordinates = "station_x station_y"
    long_name = "Water level"
    standard_name = "sea_surface_height_above_geoid"
    units = "m"
double u(time=ntimes, stations=nstations);
    coordinates = "station_xu station_yu"
    long_name = "Velocity along branch"
    standard_name = "sea_water_speed"
    units = "m s-1"
...
```

Note: The x- and y-coordinates of the station does not necessarily have the same coordinates as the nodes on the mesh or network.

B.2.9 Keyword definition for 1D Network

Table B.2: *1D network topology*

Required topology attributes	**Value**	**Example**
cf_role	"mesh_topology"	
topology_dimension	1	
edge_geometry	*variable name*	"network1d_geometry"
edge_node_connectivity	*variable name*	"network1d_edge_nodes"
node_coordinates	*variable name*	"network1d_node_x␣network1d_node_y"
Optionally required attributes		
edge_dimension	*dimension name*	"network1d_nEdges"
node_dimension	*dimension name*	"network1d_nNodes"
Optional attributes		
edge_length	*variable name*	"network1d_edge_length"
long_name		"Network topology"

Table B.3: *1D network geometry, referenced by the value of the attribute* **edge_geometry** *attribute as defined inTable B.2*

Required geometry attributes	**Value**	**Example**
geometry_type	"line"	
node_coordinates	*variable names*	"network1d_geom_x␣network1d_geom_y"
Optionally required attributes		
node_count	*variable name*	"network1d_node_count"
Optional attributes		

long_name		"1D Geometry"

Table B.4: 1D computational mesh

Required mesh attributes	**Value**	**Example**
cf_role	"mesh_topology"	
topology_dimension	1	
coordinate_space	*variable name*	"network1d", *defined by Table B.2*
edge_node_connectivity	*variable name*	"mesh1d_edge_nodes"
node_coordinates	*variable names*	"mesh1d_node_branch␣mesh1d_node_offset"
Optionally required attributes		
edge_coordinates	*variable names*	"mesh1d_edge_branch␣mesh1d_edge_offset"
Optional attributes		
edge_dimension	*dimension name*	"mesh1d_nEdges"
node_dimension	*dimension name*	"mesh1d_nNodes"
long_name		"Topology data of 1D mesh"

B.2.10 Keyword definition for 1D2D Links

Table B.5: Composite meshes, 1D2D Links

Required attributes	**Value**	**Example**
cf_role	"mesh_topology_parent"	
meshes	*variable names*	"mesh1d␣mesh2d"
mesh_contact	*variable name*	"link1d2d"
Optional attributes		
long_name		"Topology data of composite 1D2D mesh"

Table B.6: 1D2D Links, referenced by the value of the attribute mesh_contact *attribute as defined in Table B.5*

Required attributes	**Value**	**Example**
cf_role	"mesh_topology_contact"	
contact	*variable names*	"mesh1d:node␣mesh2d:face"
start_index	*integer*	1
Optional attributes		
long_name		"1D2D links"

B.2.11 Addendum: Deltares-specific agreements for 1D2D datasets

The preceding proposal was initiated by Deltares, but is intended to be generic for any 1D2D data producer or consumer. In this Section we will now additionally list a few Deltares-specific requirements that have been agreed upon as part of the functional design of the Delft3D Flexible Mesh Suite/D-HYDRO Suite software, abbreviated below as D-HYDRO.

B.2.11.1 Attributes for data variables

Variable units

The units of each variable are presented according the Climate and Forecast conventions, see http://cfconventions.org/Data/cf-standard-names/41/build/cf-standard-name-table.html, column "canonical units". Because the unit of Chézy does not fulfill the SI unit convention because the power of the length is not an integer ($C = [m^{\frac{1}{2}}/s] \equiv [m^{1/2}/s] \equiv [\sqrt{m}/s]$), we choose to represent the unit of Chézy in the netCDF file by: [m1/2 s-1]

long_name

Long names in a variable's `long_name` attribute should start with a capital character, because these names are used in plots as labels. See http://cfconventions.org/Data/cf-conventions/cf-conventions-1.7/cf-conventions.html#long-name

B.2.11.2 Non-unique relation between 1D mesh nodes and network topology edges

D-HYDRO uses one computation mesh point on each network topology node. When that network topology node is a network end point, the mesh node uniquely lies on that network topology edge (sometimes also referred to as a "‘*branch*"’). However, when the network topology node is not an end point (then often referred to as a "‘*connection node*"’), then the mesh node could equally well be associated with any of the connected network edges. For example, referring to Figure B.1, 1D mesh node 6 can be defined on branch A offset 2500, or on branch B offset 0, or on branch C offset 0. All are the same x, y locations. D-HYDRO has not yet defined a specific choice for network edge/branch selection, but this could for example be: the lowest branch number.

No choice has been made yet for models where the physics require multiple 1D mesh points on connection nodes (e.g. for morphology and 1D bed levels) nor where the physics require fluxes between all these multiple 1D mesh points on connection nodes (e.g., for water quality transport fluxes between all pairs of branches on a connection node).

B.2.11.3 Branch and node terminology

D-HYDRO models will often refer to network edges as branches, even though this is not in the standard. To this end, some additional attributes (and underlying variables) will be present for network edges in the net files:

◇ `network1d:branch_id` points to a variable name with the character IDs for all branches.
◇ `network1d:branch_long_name` points to a variable name with the character long names for all branches.
◇ `network1d:branch_order` points to a variable name with the integer branch orders for all branches, used in cross section interpolation.

The network nodes are also often referred to by their character string nodeId, for example in 1D boundary conditions (Section C.5.2.1). To this end, some additional attributes (and underlying variables) will be present for network nodes in the net files:

- `network1d:node_id` points to a variable name with the character IDs for all nodes.
- `network1d:node_long_name` points to a variable name with the character long names for all nodes.

Contrary to the above definitions for the *network* layer, no such variables are defined for the computational *mesh* layer. Locations should not be referenced by any computational grid point number. This is motivated by the fact that model schematizations may be changed with different computation grid resolutions, making any old reference to grid point numbers obsolete. Only using network branch and node ids or even x,y coordinates to specify locations makes all model input reusable.

B.2.11.4 Water system characteristics

Most of the definitions in this entire chapter deal with geometry, but more network characteristics may need to be stored in order to represent all relevant details of the 1D water system.

B.2.11.4.1 Branch type

Urban sewer systems typically distinguish between pipes draining only storm sewage, only domestic/industrial sewage (so-called dry weather flows), or both (mixed). D-HYDRO stores this information into the network file by means of an integer branch type variable with explanatory `flag_meanings` attribute:

```
int network_branch_type(network1d_nEdges);
    long_name = "Water type in branch (network edge)";
    flag_values = [1, 2, 3, 4, 5];
    flag_meanings = "dry_weather_flow storm_water_flow mixed_flow
                     surface_water transport_water";
    _FillValue = -999;
    valid_range = [1, 5];
    mesh = "network1d";
    location = "edge";
```

The water type "transport_water" represents transport pipes that connect two subparts of the total network. Often these are pressurized pipes behind a pump. This type is mainly used in the GUI to distinguish these pipes from regular sewer pipes and open water.

The water type of a manhole (compartment), in the numerical model represented by a storage node, can be derived from the network branches connected to the corresponding network node; it's therefore not stored separately. When all connected branches are of the same type, the node is also of that type. When the connected branches have different types, the node will be of type `mixed_flow`. The storage nodes file (section C.17) includes a `nodeType` keyword to indicate the purpose of the manhole; that type represents information that is complementary to the water type.

B.2.11.5 Network geometry

The presence of network geometry (section B.2.4.3) via the `edge_geometry` attribute is optional in a network topology variable. D-HYDRO however, will always write network edge geometry points, and will also assume the presence of these when reading in 1D data files. In particular, when network edge lengths are specified (thereby overriding the (shortest) lengths of the straight network edges), it makes sense to define the precise geometrical shape of the network edges. This is because the discrete 1D mesh points defined on each network edge using offsets can then be placed approximately at the correct x, y positions.

B.2.11.6 Positioning of 1D mesh nodes: from offset to x, y

Discrete 1D mesh points are defined in the dataset by the nodes's branch and offset variables. The corresponding x, y-coordinates of each 1D mesh node are determined as follows:

1 Determine the total length L_E of the network edge on which the mesh node lies:

- ◇ *If* a network edge length variable is defined via the network topology's `edge_length` attribute, read the length for that edge.
- ◇ *If not*, then calculate the direct length between the edge's endpoints, using the network topology's `edge_nodes` table, and its `node_coordinates` variables.

2 Interpolate the mesh node's offset d_n onto the network edge geometry:

- ◇ *If* an explicit network edge geometry is defined via the network topology's `edge_geometry` attribute:

  2.1 Calculate the total length L_{G_E} of the geometry line string of that edge.
  2.2 Scale the offset d_n of the mesh node on the network edge to a corresponding offset of that mesh node on the network edge geometry: $d_{G_E} = d_n \cdot L_{G_E} / L_E$.
  2.3 Find the edge geometry's line segment number i_{G_E} in which the calculated offset d_{G_E} lies.
  2.4 Interpolate the scaled offset d_{G_E} between edge geometry points i_{G_E} and $i_{G_E} + 1$ using the cumulative edge geometry's line segment lengths.

- ◇ *If not*:

  2.1 Calculate the relative offset of the mesh node on the network edge: $d_{n,\mathrm{rel}} = d_n / L_E$.
  2.2 Interpolate this relative offset between the network edge's start and end points (which have network node numbers $i_{E,1}$ and $i_{E,2}$, respectively): $x_n = (1 - d_{n,\mathrm{rel}}) \cdot x_{i_{E,1}} + d_{n,\mathrm{rel}} \cdot x_{i_{E,2}}$

B.2.11.7 Determine the chainage of nodes on a branch, ex. branch 1 of Figure B.4

(work in progress)
The branch offset for each node is given by the following array

```
mesh1d_node_branch = 1, 1, 3, 1, 1,
                     1, 3, 2, 4, 3,
                     3, 2, 2, 4, 4,
                     4, 1, 4, 4, 2,
                     2, 3 ;
mesh1d_node_offset =    0,  500, 1400, 1500, 2100,
                     2750, 3500, 3150, 2100, 2200,
                     3000,  700, 1400, 1050, 1400,
                     1750, 1000,  350,  700, 2100,
                     2800, 700 ;
```

The nodes 1, 6, 7 and 11 are network nodes, these nodes are assigned to a one of the branches who are connect to these nodes. In this example node 1 and 6 are assigned to branch A with offset 0 and 2750,and nodes 7 and 11 are assigned to branch C with offset of 0 and 3000.

How to determine the offset of a network node if not listed in the array `mesh1d_node_offset`

Note: The network edge length appears two times in the definition, one when defining the network edge length (array `network1d_edge_length`) and one in the array `mesh1d_node_offset`.

As example choose node 7.
For branch B, C and D the chainage value of node 7 is different:
3500.0 for branch B,
0.0 for branch C,
2500.0 for branch D
Which chainage value is assigned to this node is dependent on the branch you are looking to, the chainage for node 7 is 0.0 because node 7 is located on branch C, as given by array `mesh1d_node_branch` (C≡3).

Assume that we want to have the chainage values for branch B shown if Figure B.4.

- ◇ Look into the array `mesh1d_edge_branch` for the edges laying on branch B. The edges are: 1, 4, 8, 12, 15 and 16
- ◇ Look in the array `mesh1d_edge_nodes` for nodes who define the edges, i.e. 12–13, 8–7 , 21–8, 1–12, 13–20 and 20–21. These pairs are ordered, the positive direction is from the first to the second node, we need that in the next step.
- ◇ Sort these edges in positive direction, resulting in: 1–12, 12–13, 13–20, 20–21, 21–8 and 8–7. The first and last node are the begin and end node of the branch and have either zero or branch length as chainage.
- ◇ Node 1 is the begin of the branch and its chainage is set to 0.0 and node 7 is the end of the branch so its chainage is set to 3500.
- ◇ The chainage of the other nodes are given by array `mesh1d_node_offset`
- ◇ The chainage for branch B is given in Table B.7:

***Table B.7:** Chainage table for branch B, Figure B.4*

Node	Chainage
1	0000.0
12	0700.0
13	1400.0
20	2100.0
21	2800.0
8	3150.0
7	3500.0

The same algorithm can be used for a so called route, i.e. a sequence of branches. Note that for a route the length of the preceding branches should be taken into account.

Note: If the edge direction is in the same direction as the branch direction then the offset of an edge at the beginning or end of the branch can be taken from the array `network2d_edge_length`.

B.2.11.8 1D2D link types

1D2D links are defined as mesh contacts in the network file, which have been introduced in Section B.2.5.2. The *type* of 1D2D links is not included in the concept of mesh contacts, therefore this is via an additional variable in the file. In the example below, the variable `link1d2d_contact_type` can be identified via the mesh contact's attribute `link1d2d:contact_type`.

```
uint link1d2d(nLink1D2D_edge, Two) ;
    link1d2d:cf_role = "mesh_topology_contact" ;
    link1d2d:contact = "mesh1d:node mesh2d:face" ;
    link1d2d:contact_type = "link1d2d_contact_type" ;
    link1d2d:start_index = 1 ;

uint link1d2d_contact_type(nLink1D2D_edge) ;
    link1d2d:contact_type = "link1d2d_contact_type" ;
    link1d2d:flag_values = [3, 4, 5, 7] ;
    link1d2d:flag_meaning = "lateral_1d2d_link longitudinal_1d2d_link
                             street_inlet_1d2d_link roof_gutter_1d2d_link";
    link1d2d:valid_range = [3, 7] ;
    link1d2d:_FillValue = -1 ;
```

The integer link type codes determine how the 1D2D link is treated numerically. This is further described in Section 8.9. Note that the lateral and embedded link types are both coded as type 3.

B.2.11.9 Data compression by packing

This section is not specific for 1D network input, but applies to describing packed data on any mesh or network. An example use of this is the classmap output as described in Section F.3.3. Output data may be compressed by *packing*, which is explained in the CF-conventions as "altering the data in a way that reduces its precision" [4]. For example, floating point data may be compressed by categorizing it in consecutive intervals, each represented by an integer class number. All classes together span the full data range.

Here we propose a small extension to the CF-conventions concept of flag values and flag meanings. We introduce a new data variable attribute `flag_bounds`, which gives the name of a variable containing the bounds of each category. This flag bounds variable for a set of N contiguous intervals must be an array of shape $(N, 2)$, where N is the slowest varying dimension. When an interval has an open end, that bound should be set to the fill value. The following example packs a velocity magnitude variable into 8 intervals of non-uniform size, the last of which open-ended.

```
dimensions:
  Two = 2 ;
  mesh2d_nFaces = 40 ;
  time = UNLIMITED ; // (24 currently)
  class_ucmag = 8 ;
variables:
  byte mesh2d_ucmag(time, mesh2d_nFaces) ;
    mesh2d_ucmag:mesh = "mesh2d" ;
    mesh2d_ucmag:location = "face" ;
    mesh2d_ucmag:coordinates = "mesh2d_face_x mesh2d_face_y" ;
    mesh2d_ucmag:cell_methods = "mesh2d_nFaces: mean" ;
    mesh2d_ucmag:cell_measures = "area: mesh2d_flowelem_ba" ;
    mesh2d_ucmag:standard_name = "sea_water_speed" ;
    mesh2d_ucmag:long_name = "Flow element center velocity magnitude" ;
    mesh2d_ucmag:units = "m s-1" ;
    mesh2d_ucmag:grid_mapping = "projected_coordinate_system" ;
    mesh2d_ucmag:flag_values = 1b, 2b, 3b, 4b, 5b, 6b, 7b, 8b ;
    mesh2d_ucmag:flag_bounds = "class_bounds_ucmag" ;
    mesh2d_ucmag:flag_meanings = "0.000m_s-1_to_0.200m_s-1
     0.200m_s-1_to_0.400m_s-1 0.400m_s-1_to_0.600m_s-1
     0.600m_s-1_to_0.800m_s-1 0.800m_s-1_to_1.000m_s-1
     1.000m_s-1_to_1.500m_s-1 1.500m_s-1_to_2.000m_s-1 above_2.000m_s-1" ;

  double class_bounds_ucmag(class_ucmag, Two) ;
```

[4] http://cfconventions.org/cf-conventions/cf-conventions.html#packed-data

```
data:
 mesh2d_ucmag =
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
    1, 4, 2, 1, 1, 5, 4, 2, 1, 5, 4, 2, 5, 4, 5,
  //...
 class_bounds_ucmag =
  0, 0.2,
  0.2, 0.4,
  0.4, 0.6,
  0.6, 0.8,
  0.8, 1,
  1, 1.5,
  1.5, 2,
  2, _ ;
```

C Attribute files

C.1 Introduction

In the following sections we describe the attribute files used in the input file (MDU-file) of D-Flow FM. Most of these files contain the quantities that describe one specific item, such as the location of open boundaries, or time dependent data of fluxes discharged in the model area by discharge stations. Most of the attribute files can be generated by the D-Flow FM GUI after defining an input scenario. Some files can almost only be generated by utility programs such as the unstructured grid generated by RGFGRID. Still, we describe both type of files as it might be useful to know how the input data is structured to be able to generate (large) files, such as astronomic boundary conditions, or time-series for wind speed and direction by client specific tools.

C.2 Polyline/polygon file

D-Flow FM uses the same format for polylines and (closed) polygons. When used as a polygon file, there is *not* the requirement that the first and last point should be identical. It is good modelling practice to name files containing polygons with the extension <.pol>, and files containing polylines with the extension <.pli>. When the polylines have a third column with z-coordinates, the extension <.pliz> is advised.

File contents	The coordinates of one or more polylines. Each polyline (piecewise linear) is written in a single block of data.
Filetype	ASCII
File format	Free formatted
Filename	<*name*.pol>
Generated	RGFGRID, QUICKIN, D-Flow FM GUI, etc

Record description:

Record	Record description
	Preceding description records, starting with an asterisk ($*$), and will be ignored.
1	A non blank character string, starting in column one.
2	Two integers N_r, N_c representing the numbers of rows and number of columns for this block of data.
	Two reals representing the x, y or λ, ϕ-coordinate, followed by remaining data values at that location (if $N_c > 2$).

Example:

```
*
* Polyline L007
*
L007
6  2
            132400.0    549045.0
            132345.0    549030.0
            132165.0    549285.0
            131940.0    549550.0
            131820.0    549670.0
```

```
            131585.0    549520.0
*
* Polyline L008
*
L008
4  2
            131595.0    549685.0
            131750.0    549865.0
            131595.0    550025.0
            131415.0    550175.0
*
* Polyline L009
*
L009
6  2
            131595.0    549655.0
            148975.0    564595.0
            150000.0    564935.0
            152105.0    565500.0
            153150.0    566375.0
            154565.0    567735.0
```

C.3 Sample file

Sample file are used for several of D-Flow FM's input files.

File contents	The location and value of samples.
Filetype	ASCII
File format	Free formatted
Filename	<*name*.xyz>
Generated	Manually or Offline with QUICKIN or D-Flow FM GUI and data from digitised charts or GIS-database.

Record description:

Filetype	Record description
Free formatted	Location and sample value per row Two reals representing the x, y or λ, ϕ-coordinate and one real representing the sample value

Example:

Sample file with 12 sample values with their location (free formatted file).

```
   213813.2       603732.1      -4.053000
   214686.0       607226.1      -4.522000
   214891.7       610751.2      -5.000000
   210330.8       601424.1      -2.169000
   211798.0       604444.8      -2.499000
   212460.0       607475.7      -2.760000
   212436.9       610362.5      -2.865000
   185535.4       606607.9       1.360000
   186353.0       603789.4       1.122000
   187959.2       601197.6      0.9050000
   190193.0       599101.5      0.7050000
   208578.7       602513.7     -0.7990000
```

C.4 Time series file (ASCII)

Time series files are used for several of D-Flow FM's input files. There is no header, except for optional comment lines (starting with * or #). There should be two or more columns containing numbers (integer or floating point): the first column contains time in minutes since the model's reference date, all remaining columns contain the data values. Columns can be separated using one or more whitespace or tab characters. Each line contains a single time with its (space-uniform) values. A simple example is shown below:

```
* comments
0  1.23
60 1.5
* ...
1440 1.5
```

C.5 The external forcings file

Two definition files for the external forcings are supported, each with their own format.

C.5.1 Old style external forcings

The name of the file is specified in the MDU-file as ExtForceFile. External forcings are specified in in blocks key-value pairs, e.g.

```
* comments
* comments

QUANTITY =waterlevelbnd
FILENAME =tfl_01.pli
FILETYPE =9
METHOD   =3
OPERAND  =O

QUANTITY = ...
FILENAME = ...
...
```

The format is not case-sensitive, but key-value pairs are expected in the *particular order*

The majority of the keywords speak for themselves or are optional. The operand needs some explanation. This keyword determines the operation to be performed with a new connected forcing for a quantity already set by another forcing from a different source: 'O' to replace the existing value with the new one and '+' to add the new to the existing value. Though generally applicable to all external forcings, it is most commonly used when working with spatial fields (such as wind and atmospheric pressure) originating from different types of sources. For instance a spatially uniform wind field could be set as a timeseries, but overwritten locally by a values interpolated from regional atmospheric model output, upon which a cyclone is superimposed from a spiderweb-type file. This means that the <ext>-file likely has three entries for quantity wind in the aforementioned order: a uniform timeseries with an operand 'O', a netCDF file with operand 'O' and a spiderweb file with operand '+'. The netCDF file might not cover the entire flow grid. In the uncovered regions, the value from the uniform wind is left unaltered. The same is done in the case of the spiderweb file with operand 'O'. Regions for which data is missing in spatial fields are treated as non-existent in the file (currently no interpolation is by default active to fill gaps, therefore it is highly recommendable to complete these files in a preprocessing step). In netCDF files, missing data are recognized as values equal to the fill value designated in the `_FillValue`-attribute. In files for the Curvi-type (ASCII) it is the `NODATA`-value in the header.

A comprehensive list of keywords is given in the table below.

quantity	character	Name of the quantity, see the list below
filename	character	File associated with this forcing
varname*	character	Variable name used in filename associated with this forcing; some input files may contain multiple variables
sourcemask*	character	File containing a mask
filetype	integer	Indication of the filetype: 1. Time series (C.4) 2. Time series magnitude and direction 3. Spatially varying weather 4. ArcInfo 5. Spiderweb data (cyclones) (C.13.3) 6. Curvilinear data (C.13.2) 7. Samples (C.3) 8. Triangulation magnitude and direction 9. Polyline (<∗.pli>-file, C.2) 11. NetCDF grid data (e.g. meteo fields) 14. NetCDF wave data
method	integer	Method of interpolation: 1. Pass through (no interpolation) 2. Interpolate time and space 3. Interpolate time and space, save weights (+ optional extrapolation) 4. Interpolate space 5. Interpolate time 7. Interpolate/Extrapolate time
extrapolation_method*	integer	Optionally allow nearest neighbour extrapolation in space (0: no, 1: yes). Default off.
maxsearchradius*	float	Max search radius for nearest neighbor extrapolation in space.
operand	character	Overwriting or superimposing values already set for this quantity: 'O' Values are overwritten. '+' New value is superimposed.
value*	float	custom coefficients for transformation
factor*	float	
ifrctyp*	float	
averagingtype*	float	
relativesearchcellsize*	float	
extrapoltol*	float	
area*	float	Area for source/sink

Note: Keywords marked with ∗ are optional

Remark:

◇ A polyline file specified as FILENAME for a boundary should only contain a single polyline. If you need boundaries that consist of multiple unconnected polylines, provide a separate block for each of them.

C.5.2 New style external forcing

The name of this file is specified using the keyword ExtForceFileNew in the MDU-file (see Appendix A). External forcings are specified in the ini-file format. Table C.1 lists the structure of this file.

Table C.1: *Definition of external forcings.*

Keyword	Type	Default	Description
[General]			
fileVersion	String	2.01	File version. Do not edit this.
fileType		extForce	File type. Do not edit this.
[Boundary] definition(s)			See Section C.5.2.1.
[Lateral] definition(s)			See Section C.5.2.2.
[Meteo] definition(s)			See Section C.5.2.3.

This file may contain one or more [Boundary] and [Lateral] and [Meteo] blocks, each followed by a set of keywords that specify the location and forcing for that forcing. See the following sections for details.

Example:

```
[General]
fileVersion        = 2.01
fileType           = extForce

[Boundary]
quantity           = waterlevelbnd
locationFile       = tfl_01.pli
forcingFile        = tfl_01.bc

[Lateral]
id                 = Lateral1
name               = First Lateral
type               = discharge
locationType       = 1d
branchId           = Channel1
chainage           = 250.000
discharge          = 27.3

[Meteo]
quantity           = rainfall
forcingFile        = precip.nc
forcingFileType    = netcdf
targetMaskFile     = roofs.pol
targetMaskInvert   = true
interpolationMethod = nearestNb
```

C.5.2.1 Boundary definitions

***Table C.2:** Boundary definitions in new style external forcing file.*

Keyword	Type	Default	Description
(Part of new-style external forcings file, see Table C.2.)			
`[Boundary]`			*Repeat as needed*
`quantity`	String		Name of the quantity, see the list in Section C.5.3 (boundaries only)
`nodeId`	String		(Optional) Node id of a 1D network node.
`locationFile`	String		Name of boundary polyline <∗.pli>. This keyword is only used if `nodeId` is not specified.
`forcingFile`	String		Name of file containing the forcing for this boundary. The file may either be a <bc> file (see section E.2.3) or a netCDF-file (see section E.2.4).
`returnTime` (`return_time` in earlier versions)	Double	0	(Optional) Thatcher-Harleman return time [s]. See also section 9.3.4. The default value of 0 seconds means that the Thatcher-Harleman return time has no impact.
`tracerFallVelocity`	Double	0	(Optional) Fall velocity for this tracer [m/s]. The default value 0 means that settling is disabled for this tracer.
`tracerDecayTime`	Double	0	(Optional) Decay lifetime τ for this tracer [s]. The default value 0 means that tracer decay is disabled for this tracer.
`bndWidth1D`	Double		(Optional) Custom width for boundary flow link, to override default mirrored internal flow link width.
`bndBlDepth`	Double		(Optional) Custom bed level depth below initial water level for boundary point, to override default mirrored bed level from internal pressure point.

Remarks:

- ◇ The `tracerfallVelocity` is set *per tracer*, not per boundary location. Multiple boundaries for the same tracer should not specify different values, otherwise the first value is used. It is sufficient to set the `tracerFallVelocity` only once.
- ◇ The `tracerDecayTime` is also set *per tracer* and follows the same rules as in the previous remark.
- ◇ A polyline file specified as `locationFile` for a boundary should only contain a single polyline. If you need boundaries that consist of multiple unconnected polylines, provide a separate `[Boundary]+locationFile` for each of them.
- ◇ When multiple `[Boundary]` blocks are present with the same `quantity` name, the behavior depends on the `locationFile` name:
    - ◇ When also the `locationFile` full paths are the same, the boundaries are applied in the order in which they appear in the file, and are *added* on top of each other (comparable to implicitly setting `OPERAND=+` if this had been an old format external

forcings file, see Section C.5).

- ◇ When the `locationFile` full paths are different, the boundaries are applied in the order in which they appear in the file, and are *overwriting* each other (comparable to implicitly setting `OPERAND=O` if this had been an old format external forcings file, see Section C.5).

C.5.2.2 Lateral discharge definitions

Table C.3: *Lateral discharge definitions in new style external forcing file.*

Keyword	Type	Default	Description
(Part of new-style external forcings file, see Table C.1.)			
`[Lateral]`			*Repeat as needed*
`id`	String		Unique lateral discharge id.
`name`	String		Long name in the user interface.
`type`	String	`discharge`	(optional) Type of lateral forcing. Possible values: ◇ `discharge`: Flow is directly prescribed via the `discharge` keyword (see below).
`locationType`	String	`all`	(optional) Only when `xCoordinates` and `yCoordinates` are also specified. Possible values are: ◇ `1d`: only affect enclosed 1D grid points (the default and only valid value when `branchId` and `chainage` are given). ◇ `2d`: only affect enclosed 2D grid points. ◇ `all`: affect all enclosed grid points.
`applyTransport`	Int	0	Defines the way the concentrations of substances at lateral locations are taken into account. ◇ `0` Only water (so without transport of substances) is added to or removed from the model, depending on the sign of lateral discharge. For 3D models, lateral discharge is applied to the top layer of the model. ◇ `1` Transport of substance is included. This option works only in combination with an external library via BMI-Interface. In a dimr configuration file, users can specify any substances and layers of the lateral discharge. If the sign is positive, lateral discharge is passed to the model with a concentration set via the BMI-interface. When negative, discharge is passed from the model with a concentration equal to that of the ambient water in the model.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
nodeId	String		(optional) Node id of a 1D network node. The lateral will be connected to the grid point on/closest to that network node.
branchId	String		(optional) For a 1D point lateral: branch on which the lateral discharge is located. Alternative to nodeId.
chainage	Double		(optional) For a 1D point lateral: location on the branch (m) of the point-lateral discharge, in combination with branchId.
numCoordinates	int		(optional) number of values in xCoordinates and yCoordinates. This value should be ≥ 1.
xCoordinates	Double[]		(optional) x-coordinates of the lateral discharge location polygon. (number of values = numCoordinates).
yCoordinates	Double[]		(optional) y-coordinates of the lateral discharge location polygon. (number of values = numCoordinates).
discharge	*		The prescribed discharge for this lateral.

Remarks:

- ◇ Three types of location specifications are available.
  - ◇ When nodeId is given, it will be used to snap the lateral to the grid point closest to that network node (only for 1D).
  - ◇ When instead, branchId is given, the keywords branchId and chainage will be used for the location specification (only for 1D).
  - ◇ When not nodeId, nor branchId are given, the keywords numCoordinates, xCoordinates and yCoordinates will be used to define a polygon. When numCoordinates ≥ 3, the lateral discharge will be distributed over all grid points inside that polygon. When instead numCoordinates $= 1$, only the 2D grid cell in which that single point lies will fully receive the lateral discharge.
- ◇ Items marked with (*) can either be a scalar value, or a time series file (<*.tim>/<*.bc>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- ◇ If the discharge is defined via a <*.bc> file, that file should contain a [Forcing] block, with name equal to this lateral's id, and quantity equal to lateral_discharge.

C.5.2.3 Meteorological forcings

***Table C.4:** Meteorological forcings in new style external forcing file.*

Keyword	Type	Default	Description
(Part of new-style external forcings file, see Table C.1.)			
[Meteo]			*Repeat as needed*
			(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
quantity	String		Name of the quantity, see the list in Section C.5.3 (meteo only).
forcingFile	String		Name of file containing the forcing for this meteo quantity.
forcingFileType	String		Type of forcingFile. Supported types: ◇ bcAscii: Space-uniform time series in <*.bc> (Section E.2.3). ◇ uniform: Space-uniform time series in <*.tim> (Section C.4). ◇ netcdf: NetCDF, either with gridded data, or multiple station time series (Section E.2.4). ◇ TODO: meteo on grid and more.
locationFile	String		*Obsolete! Only needed for uniform inside polygon, superseded by the* targetMaskFile*.*
locationType	String		*Obsolete! Maybe needed in the future for multi-station data in* bcAscii *files.*
targetMaskFile	String		(optional) Name of <*.pol> file to be used as mask. Grid parts inside any polygon will receive the meteo forcing.
targetMaskInvert	Logical	no	(optional) Flag indicating whether the target mask should be inverted, i.e., outside of all polygons: no or yes.
interpolationMethod	String		Type of (spatial) interpolation. Supported methods: ◇ nearestNb, nearest neighbor, only with station data in forcingFileType=netcdf.

Remarks:

- ◇ Spatially uniform meteo forcing is available as forcingFileType=bcascii, when that file contains a scalar time series in a block with name=global.
- ◇ Multiple station time series is available as forcingFileType=netcdf, when that file contains a time series variable with a station dimension. The station feature is identified via the attribute :cf_role="timeseries_id". x- and y-coordinates must also be present, to support spatial interpolation.

C.5.3 Accepted quantity names

The list of accepted quantity names is subdivided into six categories:

- ◇ Boundary conditions
- ◇ Meteorological fields
- ◇ Structure parameters
- ◇ Initial fields
- ◇ Spatial physical properties
- ◇ Miscellaneous

Table C.5: *List of accepted external forcing quantity names.*

Quantity	pg.	Description
Boundary conditions (old and new ext):		
waterlevelbnd	166	Water level
neumannbnd	169	Water level gradient
riemannbnd	169	Riemann invariant
outflowbnd		
velocitybnd	168	Velocity
dischargebnd	167	Discharge
riemann_velocitybnd		Riemann invariant velocity
salinitybnd	239	Salinity
temperaturebnd	239	Temperature
sedfracbnd<*fractionname*>	see[1]	Suspended sediment concentration for fraction <*fractionname*>
uxuyadvectionvelocitybnd		
normalvelocitybnd, tangentialvelocitybnd	175	Normal and tangential velocity (only in 2D)
qhbnd	171	Discharge-water level dependency
tracerbnd<*tracername*>	239	User-defined tracer
Meteorological fields: (old ext only, *: also in new ext)		
windx, windy, windxy	262	Wind components, wind vector
airpressure, charnock		
airpressure_windx_windy	262	Atmospheric pressure and wind components
airpressure_windx_windy_charnock		
atmosphericpressure	262	Atmospheric pressure
rainfall* or rainfall_rate*	265	Precipitation
humidity, cloudiness		
airdensity, airpressure, airtemperature, dewpoint	255	Varying air density
humidity_airtemperature... ..._cloudiness	512	Combined heat flux terms
humidity_airtemperature... ..._cloudiness_solarradiation		
dewpoint_airtemperature... ..._cloudiness		
dewpoint_airtemperature... ..._cloudiness_solarradiation		
longwaveradiation	512	Long wave radiation
solarradiation	512	Solar radiation
discharge_salinity... ..._temperature_sorsin	237	Discharge, salinity and heat sources

[1] D-Morphology UM (2019)

Quantity	pg.	Description
Structure parameters: (old ext only)		Use preferred structure INI file instead (See Section C.12).
pump	318	Pump capacity
damlevel		
gateloweredgelevel		
generalstructure		
Initial fields: (old ext only)		*Deprecated, see section D.2.*
initialwaterlevel	506	
initialsalinity	506	
initialsalinitytop	506	
initialtemperature	506	
initialverticaltemperatureprofile	506	
initialverticalsalinityprofile	506	
initialtracer<*tracername*>	507	
initialsedfrac<*fractionname*>	see[1]	Initial suspended sediment concentration for fraction <*fractionname*>
Spatial physical properties: (old ext only)	507	*Deprecated, see section D.2.*
frictioncoefficient		
horizontaleddyviscositycoefficient		
horizontaleddydiffusivitycoefficient		
advectiontype		
ibotlevtype		
Miscellaneous:		
shiptxy		
movingstationxy	524	Moving observation point for output (time,x,y)
wavesignificantheight	??	Wave related input
waveperiod	??	Wave related input

C.6 Initial and parameter fields

Initial and parameter fields are time-independent spatial fields that can describe initial conditions and spatially varying model parameters. The name of this file is specified using the keyword IniFieldFile in the MDU-file (see Appendix A). This file supersedes the old-style external forcings file for these quantities. More details can be found in section D.2.

C.7 Trachytopes

The trachytope functionality allows for the usage of different types of roughness formulations at different locations within the computational domain. Multiple formulation may be active in the same grid cell. Several keywords in the MDU file influence the functioning. All keywords below should be placed underneath the `[trachytopes]` section in the MDU file.

Keyword	Value	Description	Default
`TrtRou`	Y or N	Trachytope option activated	N
`TrtDef`	<name.ttd>	Definition file trachytopes	
`Trtl`	<name.arl>	Area Roughness on Link file trachytopes	
`DtTrt`	pos. real	Time step in seconds for updating roughness and resistance coefficients based on trachytopes. Must be a multiple of `DtUser`.	1 `DtUser`
`Make spatially varying input calibration possible TrtCll`	*#name#*	Spatial calibration factor for trachytopes	uniform 1
`TrtMnH`	pos. real	Minimum water depth in roughness computation	.2 `Epshu`
`TrtMth`	1 or 2	Area averaging method	1
`TrtAsr`	real $\in [0, 1]$	Serial factor in averaging of area roughnesses	0.6
`TrtMxR`	integer	Maximum recursion depth for mixed trachytope definitions	8

Limitation: `DtTrt` must be a multiple of `DtUser`

C.7.1 Area Roughness on Links (ARL-file)

File contents	The Area Roughness on Links file (ARL-file) is the input file for the spatial distribution of the alluvial and vegetation roughness which are handled by the trachytopes module.
Filetype	ASCII
File format	Space separated file format
Filename	<*name*.arl>
Generated	At present: Conversion possible from structured Delft3D-FLOW input files with matlab conversion tool from Open Earth Tools (http://www.openearth.info/), see example below (section C.7.1.2).

The ARL file has the following properties:

- A single line comment, starts with a hash tag "#" or an asterisk "*".
- Each line has the format
  "`xu yu zu TrachytopesNr Fraction`",
  - where `xu`, `yu`, `zu` is the coordinate of the midpoint of the netlink,
  - `TrachytopeNr` is an integer corresponding to the number as described in the TrachyTopes Definition File (cf. section C.7.2),
  - and `Fraction` is a Fraction between `0.0` and `1.0`

- ◇ A line with a midpoint-netlink-coordinate which is the same as the preceding line (comments not considered) allows multiple types of trachytopes roughness definitions, provided the sum of the `Fraction` keywords does not add to a value greater than `1.0`.
- ◇ If the sum is less than `1.0` the background roughness prescribed in the `[physics]` chapter in the MDU file is prescribed for the remaining `Fraction`, cf. Appendix A. Only a single roughness type is allowed in combination with the Trachytopes module.
- ◇ If a midpoint-netlink-coordinate is repeated in the .arl file with the midpoint-netlink-coordinate `xu`, `yu`, `zu` in the preceding line being different, all previous instances of the specific `xu`, `yu`, `zu` are ignored.

C.7.1.1 Example

An example of the <arl> file is given below based on the <ttd> file given in section C.7.2. In this case the netlink with u point located at x_u = 10.542 and y_u = 11.6, which is covered for 30 %, 30 % 20 % and 20 % for `TrachytopeNr` = 1, 2, 4 and 3 respectively.

```
(continued)
...
10.542    11.6    0    1      0.3
10.542    11.6    0    2      0.3
10.542    11.6    0    4      0.2
10.542    11.6    0    3      0.2
...
(continued)
```

C.7.1.2 Conversion from Delft3D 4 input files

Open Earth Tools has different matlab tools available for the conversion of Delft3D-FLOW models to D-Flow FM models (e.g. dflowfmConverter.m and d3d2dflowfm.m).

An extra Matlab conversion script has been added to convert Delft3D-FLOW's <∗.aru> and <∗.arv> files to <∗.arl> files for D-Flow FM:
d3d2dflowfm_friction_trachytopes.m.

It can be called as follows:

```
oetsettings
d3d2dflowfm_friction_trachytopes(filgrd,filaru,filarv,filnet,filarl)
```

where the variables are defined as follows:

filgrd	Delft3D-FLOW grid filename (*.grd)
filaru	Delft3D-FLOW area definition file in u-direction (*.aru)
filarv	Delft3D-FLOW area definition file in v-direction (*.arv)
filnet	name of the dflowfm network file (*_net.nc)
filarl	name of the dflowfm trachytope file (*.arl))

C.7.2 Trachytope Definition file (TTD-file)

The trachytope definition file contains lines of the following format defining the different types of trachytopes. The types which are supported are general, discharge dependent and water-level dependent formats.

C.7.2.1 General format

The general format is formatted as follows:

```
TrachytopeNr    FormulaNr     ...Parameters...
```

where `...Parameters...` indicates a space separated list of formula specific parameters; the parameters required and their order are specified in section C.7.2.5. The user must specify for each trachytope (combination of formula number and parameters) a unique positive trachytope number. This trachytope number is used in the area files (see section C.7.1) to indicate the roughness types on a net link.

C.7.2.2 Example

An example of such a file where the first three codes 1–3 with a Nikuradse roughness height are defined, and the fourth (code 4) is Chézy roughness height.

```
1       51       0.1
2       51       0.2
3       51       0.3
4       52       35
```

C.7.2.3 Discharge dependent format

The discharge dependent format is formatted as follows:

```
TrachytopeNr    DISCHARGE    "Cross-section name"
TrachytopeNr    Q1      FormulaNr      ...Parameters...
TrachytopeNr    Q2      FormulaNr      ...Parameters...
TrachytopeNr    Q3      FormulaNr      ...Parameters...
TrachytopeNr    Q4      FormulaNr      ...Parameters...
...
```

which again expects a user defined `TrachytopeNr`. The first row contains the keyword “DISCHARGE” and subsequently a cross-section name occuring in CrsFile in the mdu-file.

The cross-section name can be enclosed by quotation marks, for instance when the cross-section name is made up of more than one word. The subsequent rows all have the same `TrachytopeNr` as the first row and furthermore every `FormulaNr` should be the same. The list of discharges Q1, Q2, Q3, Q4 should be monotonically increasing.

The `...Parameters...` for each formula type are specified in section C.7.2.5.

C.7.2.4 Water level dependent format

The water-level dependent format is similar to the discharge dependent format and is formatted as follows:

```
TrachytopeNr    WATERLEVEL    "Observation-station name"
TrachytopeNr    ZS1      FormulaNr      ...Parameters...
TrachytopeNr    ZS2      FormulaNr      ...Parameters...
TrachytopeNr    ZS3      FormulaNr      ...Parameters...
TrachytopeNr    ZS4      FormulaNr      ...Parameters...
...
```

which again expects a user defined `TrachytopeNr`. The first row contains the keyword “WATERLEVEL” and subsequently an observation-station name occuring in ObsFile in the mdu-file. The observation-statopm name can be enclosed by quotation marks, for instance when the observation-station name is made up of more than one word. The subsequent rows all have the same `TrachytopeNr` as the first row and furthermore every `FormulaNr` should be the same. The list of waterlevels ZS1, ZS2, ZS3, ZS4 should be monotonically increasing.

The `...Parameters...` for each formula type are specified in section C.7.2.5.

C.7.2.5 Supported roughness formulations

The roughness formulations specified in the table below are supported by D-Flow FM. These formulations are specified in the .ttd by `FormulaNr` and `...Parameters...` in the order in which they appear in the table below. The related formulae are presented in the Technical Reference Manual

FormulaNr	Description	Parameters
Special classes (1–50)		
1	flood protected area, reduces the effective area of the grid cell. It has no influence on the continuity equation (i.e. it does not decrease the surface area of the grid cell).	–
2	composite trachytope: fraction α of type T_1 and fraction β (generally $\beta = 1 - \alpha$) of type T_2	T_1, T_2, α, β
Area trachytope classes: simple type (51–100)		
51	constant White-Colebrook/Nikuradse value	k [m]
52	constant Chézy value	C [m$^{1/2}$/s]
53	constant Manning value	n [s/m$^{1/3}$]
54	constant z_0 value	z_0 [m]
61	constant White-Colebrook/Nikuradse values for ebb and flood	k_{ebb} [m], k_{flood} [m]
62	constant Chézy values for ebb and flood	C_{ebb} [m$^{1/2}$/s], C_{flood} [m$^{1/2}$/s]
63	constant Manning values for ebb and flood	n_{ebb} [s/m$^{1/3}$], n_{flood} [s/m$^{1/3}$]
64	constant z_0 values for ebb and flood	$z_{0,ebb}$ [m], $z_{0,flood}$ [m]
Area trachytope classes: alluvial type (101–150)		
101	simplified Van Rijn	A [m$^{0.3}$], B [m$^{0.3}$]
102	power relation	A [m$^{1/2}$/s], B [-]
103[2]	Van Rijn predictor	-
104[2]	Struiksma predictor	A_1 [m$^{1/2}$/s], A_2 [-], θ_c [-], θ_m [-], $C_{\min}$ [m$^{1/2}$/s]
105[2]	bedforms quadratic	-
106[2]	bedforms linear	-
Area trachytope classes: vegetation type (151–200)		
151	Barneveld 1	h_v [m], n [1/m]
152	Barneveld 2	h_v [m], n [1/m], C_D [-], k_b [m]
153	Baptist 1	h_v [m], n [1/m], C_D [-], C_b [m$^{1/2}$/s]
154	Baptist 2	h_v [m], n [1/m], C_D [-], C_b [m$^{1/2}$/s]
Linear trachytope classes: various (201–250)		
201	hedges 1	h_v [m], n [1/m]

[2] The alluvial roughness predictors 103, 104, 105 and 106 are not yet supported, because the coupling with the morphology module is not yet available.

FormulaNr	Description	Parameters
202	hedges 2	h_v [m], n [1/m]
Linear trachytope classes: various (251–300)		
251	trees	h_v [-], C_D [-]

C.8 Fixed weirs

A fixed weir has many quantitites. On a polyline several quantitites has to be specified. In the table below an overview is given of all these quantities.

Keyword	Description	Default value
`X-coordinate`	X-coordinate of polyline point	-
`y-coordinate`	Y-coordinate of polyline point	-
`Crest level`	Absolute crest level	- [mAD]
`Ground height left`	Difference between crest level and toe level at left side	0.0 [m]
`Ground height right`	Difference between crest level and toe level at right side	0.0 [m]
`Crest width`	Width of weir in perpendicular direction	3.0 [m]
`Slope left`	Slope of weir at left side	4.0 [-]
`Slope right`	Slope of weir at right side	4.0 [-]
`Roughness code`	Roughness code for vegetation on weir	0.0 [-]
`Weir type`	(v)illemonte or (t)abellenboek	v [-]

in which 'mAD' stands for 'meter Above Datum', which denotes a level relative to the vertical reference system. The last column with the weir type (Villemonte or Tabellenboek) is optional. In this way, individual polyline can be given another weir type. For example, if FixedweirType=9 (Villemonte) has been specified in the MDU-file, then individual polylines can be set to Tabellenboek by adding a "t" to all points in the polyline; see the example below.

The default value for the ground height levels (left and right) is 0.0 m. However, in case of the Tabellenboek a minimum value for the ground heights of 0.1 m is applied, because the empirical Tabellenboek relation doesn't allow values smaller than 0.1 m. Noted that in WAQUA also a minimum value of 0.1 m is applied in case of the Tabellenboek. For fixed weirs also a shape file can be generated, see section 5.4.12. In case of the Tabellenboek the ground heights in the shape file can be slightly different from the value in the input file, because a minimum value of 0.1 m is applied.

For some of the fixed weir input quantities upper and lower limits are applied, because realistic input values are required for an accurate computation of the energy losses of fixed weirs. This involves a maximum crest level of 10000 [m], a minimum slope of 0.0000001 [-], a maximum slope of 1000 [-], a minimum ground height of 0.0 [m] and a maximum ground height of 500 [m]. If one of these values isn't between the lower and upper limit, then an error message is written to the diagnostic file and the simulation will stop. Then, the user has to check the fixed weirs input file for erroneous values. Below an example is given of an error message for fixed weirs in the diagnostic file, in which a negative slope is applied.

```
     ** INFO   : Opened file : harlingen-1_fxw_error.pliz
 ** INFO   : Closed file : harlingen-1_fxw_error.pliz
 ** WARNING: Slope left -0.10000D+01 is smaller than the minimum limit   0.10000D-07.
 ** WARNING: Slope left is located at line 2 in PLI FixedWeir01 of file harlingen-1_fxw_error.pliz.
 ** ERROR  : Some fixed weirs have values outside of limits. See messages above.
```

Example

An example of a polyline with fixed weir input is given below. It consists of a polyline with two points

```
(continued)
...
weir
     2     10
     399.999420   99.997688    2.0   0.2   0.8   3.0   4.0   4.0   0   t
     399.999420  100.999542    2.1   0.3   0.9   3.0   4.0   4.0   0   t

...
(continued)
```

In general, fixed weir polylines do not coincide with the computational grid. The polylines are snapped to flow links, which is illustrated in Figure 5.15. This is computed by the computational kernel of D-Flow FM.

In practice, it is possible that a certain flow link contains multiple snapped fixed weirs. This is for example the case when polylines with fixed weirs cross each other. Then, the fixed weir with the highest crest level is taken, because the crest level is used in the drying-flooding algorithm. In this way, overtopping of a weir the water level only occurs if the water level is above the weir with the highest crest level.

C.9 Calibration Factors

The calibration factor functionality is a multiplier for the rouhgness at different locations within the computational domain. Multiple formulation may be active in the same grid cell. Several keywords in the MDU file influence the functioning. All keywords below should be placed under the `[calibration]` section in the MDU file.

Keyword	Value	Description	Default
`UseCalibration`	1 or 0	Calibration factor option activated	N
`DefinitionFile`	<name.cld>	Calibration factor definition file	
`AreaFile`	<name.cll>	Calibration factor area file	

C.9.1 Calibration factor definition file (CLD-file)

The calibration class definition file contains lines of the following format defining the different calibration classes. It supports constant values and values that depend on the discharge at a named cross section or the water level at a named location.

C.9.1.1 Header of the CLD-file

The file starts with the following header

```
# [FileInformation]
#   FileType     = CalibrationFactorsDefinitionFile
#   FileVersion  = 1.0
# [CalibrationFactors]
```

C.9.1.2 Constant values

The format for constant values is as follows:

```
CalibrationClassNr  ConstantValue
```

where ConstantValue indicates the calibration value to be used (use 1.0 to use the uncalibrated roughness). The user must specify for each calibration class (line in this file) a unique calibration class number CalibrationClassNr. This calibration class number is used in the calibration area <.cll> file to indicate the area of influence of this class.

C.9.1.3 Discharge dependent format

The discharge dependent line is formatted as follows:

```
CalibrationClassNr DISCHARGE "Cross-section name"
CalibrationClassNr Q1 ConstantValueAtQ1
CalibrationClassNr Q2 ConstantValueAtQ2
CalibrationClassNr Q3 ConstantValueAtQ3
CalibrationClassNr Q4 ConstantValueAtQ4
...
```

which again expects a user defined calibration class number CalibrationClassNr. The first line contains the keyword DISCHARGE and subsequently a cross-section name occurring in the CrsFile in the mdu-file. The cross-section name can be enclosed by quotation marks, for instance when the name contains spaces. The subsequent lines all start with the same calibration class number as the first line. The list of discharges Q1, Q2, Q3, and Q4 should be monotonically increasing. For each discharge value a calibration value should be specified. Between the discharges specified, the calibration values are linearly interpolated. Below the minimum and above the maximum discharge the first and last values are used respectively.

C.9.1.4 Water level dependent format

The water-level dependent format is similar to the discharge dependent format and is formatted as follows:

```
CalibrationClassNr WATERLEVEL "Observation-station name"
CalibrationClassNr ZS1 ConstantValueAtZS1
CalibrationClassNr ZS2 ConstantValueAtZS2
CalibrationClassNr ZS3 ConstantValueAtZS3
CalibrationClassNr ZS4 ConstantValueAtZS4
...
```

which again expects a user defined calibration class number CalibrationClassNr. The first line contains the keyword WATERLEVEL and subsequently an observation-station name occurring in ObsFile in the mdu-file. The observation-station name can be enclosed by quotation marks, for instance when the name contains spaces. The subsequent lines all start with the same calibration class number as the first line. The list of water levels ZS1, ZS2, ZS3, and ZS4 should be monotonically increasing. For each water level value a calibration value should be specified. Between the water levels specified, the calibration values are linearly interpolated. Below the minimum and above the maximum water level the first and last values are used respectively.

C.9.1.5 Example

The example file given below defines three calibration classes. In the area associated with the first class, the roughness values are used without any adjustment. In the area associated with the second class, the roughness value is decreased by 20%. In the area associated with the third and final class, the increase in the roughness value varies between 0% and 30% depending on the discharge at a specific location.

```
# [FileInformation]
#   FileType     = CalibrationFactorsDefinitionFile
#   FileVersion  = 1.0
# [CalibrationFactors]

# areas without any calibration adjustment
1 1.0

# areas with 20% decreased roughness value
2 0.8

# area with calibration depending on discharge at crsX
10 DISCHARGE 'crsX'
10 500  1.0
10 1000 1.1
10 3000 1.3
```

C.9.2 Calibration Class Area on Links (CLL-file)

C.9.2.1 Header of the CLL-file

The file starts with the following header

```
# [FileInformation]
#   FileType     = CalibrationAreasFile
#   FileVersion  = 1.0
# [CalibrationAreas]
```

C.9.2.2 Body of the CLL-file

The body of the CLL-file consists of comment lines (any line starting with # or *) or data lines of the following format:

```
xu yu zu CalibrationClassNr AreaFraction
```

where xu, yu, zu is the coordinate of the midpoint of the netlink, CalibrationClassNr is an integer corresponding to the number as described in the Calibration Class Definition File (cf. section C.9.1), and AreaFraction is an area fraction between 0.0 and 1.0.

All lines that list the same xu, yu, zu coordinates are combined to a weighted average of multiple calibration classes, provided the sum of the fractions does not add to a value greater than 1.0.

If the sum of the areas specified for a certain netlink is less than 1.0, then the a constant calibration factor of 1.0 is assumed for the remaining area (i.e. no calibration effect). The result is that if for a specific netlink no line is specified at all, then the roughness of this netlink will be unaffected by the calibration process.

C.9.2.3 Example

The example of the <CLL>-file is given below based on the <CLD>-file given in section C.9.1. In this case the roughness at netlink with midpoint located at xu = 10.542 and yu = 11.6 is completely affected by calibration class 1, the netlink with midpoint at xu = 11.120 and yu = 12.1 is affected for 60 % by calibration class 1 and 40 % by calibration class 10, and the final location listed uses only the value resulting from calibration class 10.

```
...
10.542 11.6 0 1 1.0
11.120 12.1 0 1 0.6
11.120 12.1 0 10 0.4
12.635 12.6 0 10 1.0
...
```

C.10 Sources and sinks

Sources and sinks (section 8.10) are defined as part of the <∗.ext> file, as follows:

```
QUANTITY=discharge_salinity_temperature_sorsin
FILENAME=chan1_westeast.pli  # A file 'chan1_westeast.tim', with same basename
                             # as the polyline should be present
FILETYPE=9
METHOD  =1
OPERAND =O
AREA    =1.5
```

The <∗.pli(z)> file is a polyline file (section C.2) with two or three or five columns. 2D models only need two (x, y) columns. A 3D model with extraction and discharge in a single layer needs three (x, y, z) columns. It is also possible to extract and/or discharge into a range of vertical layers. This must be specified in a five-column file $(x, y, z_{\text{min}}, z_{\text{max}}, 0)$, where the fifth column is a dummy unused value and $z_{\text{min}}, z_{\text{max}}$ define the range of layers across which the discharge is averaged. Along with the <∗.pli(z)>-file, there has to be a time series file (section C.4) with the same basename as the <∗.pli(z)>-file, and extension <.tim>. The columns in the <∗.tim> are as follows: time in minutes, discharge Q in [m^3/s], and *all* constituents in the model. The order of consituents is: salinity in [ppt], temperature T in [°C], sediment concentrations, spiral flow intensity and tracers. For example, if we only have temperature and two tracers we get 5 columns: time, discharge, temperature, tracer 1, tracer 2. For more information, see section 8.10.

C.11 Dry points and areas

Dry points can either be defined by a basic sample file (see section C.3), or a polygon file (see section C.2).

Special attention must be given to the optional third column of a polygon file: when the first point has a z-value of -1, the polygon mask is inverted, i.e., all points outside the polygon are set as dry points.

More details can be found in section 5.4.2.7.

C.12 Structure INI file

Hydraulic structures are defined in a <structures.ini> file. Each structure is defined in its own [Structure] section, followed by a set of key-value parameters that depend on the type of structure:

Table C.7: *General part of structure file.*

Keyword	Type	Default	Description
[General]			
fileVersion	String	3.01	File version. Do not edit this.
fileType	String	structure	File type. Do not edit this.
[Structure]			
id	String		Unique structure id (max. 256 characters).
name	String		Given name in the user interface.
branchId	String		(optional) Branch on which the structure is located.
chainage	Double		(optional) Chainage on the branch [m].
numCoordinates	int		(optional) Number of values in xCoordinates and yCoordinates. This value should be greater or equal 2.
xCoordinates	Double		(optional) x-coordinates of the location of the structure. (number of values = numCoordinates)
yCoordinates	Double		(optional) y-coordinates of the location of the structure. (number of values = numCoordinates)
type	String		Structure type. Possible values: ◇ weir (for specific input refer to C.12.1) ◇ universalWeir (for specific input refer to C.12.2) ◇ culvert (for specific input refer to C.12.3) ◇ longCulvert (for specific input refer to C.12.4) ◇ bridge (for specific input refer to C.12.6) ◇ pump (for specific input refer to C.12.7) ◇ orifice (for specific input refer to C.12.8) ◇ gate (for specific input refer to C.12.9) ◇ generalStructure (for specific input refer to C.12.10) ◇ dambreak (for specific input refer to C.12.11) ◇ compound (for specific input refer to C.12.12)
...			Structure type specific data

Remarks:
- The structure id/name may contain spaces, with no quoting needed.
- Two types of location specifications are available.
  - When branchId is given, the keywords branchId and chainage will be used for the location specification (only for 1D).

  - When branchId is *not* given, the keywords numCoordinates, xCoordinates and yCoordinates will be used to define a polyline. The structure will be split up over the intersecting flow links.

- All structure types require a location specification as above, except for the compound structure. The latter should not have a location specification; it is inherited from its underlying structures (Section C.12.12).
- When direction is of importance (e.g., for structures with an allowedFlowDir, and for all flux quantities output), the positive direction is oriented as follows:
  - When placed on 1D branch, in the direction of the branch.
  - When placed via a polyline (x/yCoordinates), the positive direction is oriented 90 degrees clockwise w.r.t. the polyline.
- The universal weir, culvert and bridge can only be used in a single 1D channel.
- The long culvert does not follow the location specifications described above. Instead, the polyline coordinates should lie on top of 1D network nodes, to define the location and length in the positive flow direction.
- All other structures will be split up over the intersecting flow links.

C.12.1 Weir

The data block for a weir structure in the <structures.ini> file is described in this section. The geometrical shape and hydraulic treatment is further described in Section 14.4.

Table C.8: *Weir definition.*

Keyword	Type	Default	Description
[Structure]			
type	String	weir	Structure type; must read weir.
allowedFlowdir	String	both	Possible values: both, positive, negative, none.
crestLevel	*		Crest level of weir [m AD].
crestWidth	Double	$\sum w_u$	Width of weir [m].
corrCoeff	Double	1.0	Correction coefficient [-].
useVelocityHeight	logical	true	Flag indicates whether the velocity height is to be calculated or not.

Remarks:

- Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- crestWidth is optional. Default is (the sum of) the width[s] of the flow link[s]. If the given value is larger than this sum, it is automatically lowered to this default.
- If the crestWidth of a 2d structure is smaller than the sum of the widths of the 2d links, the crestWidth of the individual structure elements is reduced symmetrical from the outside inwards.

C.12.2 Universal Weir

The data block for a universal weir structure in the <structures.ini> file is described in this section. The geometrical shape and hydraulic treatment is further described in Section 14.9.

Table C.9: *Universal Weir definition.*

Keyword	Type	Default	Description
[Structure]			
type	String	universal Weir	Structure type; must read universalWeir.
allowedFlowDir	String		Possible values: both, positive, negative, none.
numLevels	Int		Number of yz-Values.
yValues	Double[]		y-values of the cross section [m]. (number of values = numLevels)
zValues	Double[]		z-values of the cross section [m]. (number of values = numLevels)
crestLevel	Double		Crest level of weir [m AD].
dischargeCoeff	Double		Discharge coefficient c_e [-], as used in the formulas in Section 14.9.

Remark:

◇ The zValues should be considered as relative values, merely defining the shape of the cross section. The absolute z-values are determined by shifting the entire cross section such that the lowest point is exactly at crestLevel.

C.12.3 Culvert

The data block for a culvert structure in the <structures.ini> file is described in this section. The geometrical shape is described by the parameters in the following table and these refer to the symbols in Figure 14.7. The entrance and exit losses are further described in Section 14.6.

Table C.10: *Culvert*

Keyword	Type	Default	Description
[Structure]			
type	String	culvert	Structure type; must read culvert.
allowedFlowdir	String		Possible values: both, positive, negative, none.
leftLevel	Double		Left invert level of the culvert (z_{c1}) [m AD].
rightLevel	Double		Right invert level of the culvert (z_{c2}) [m AD].
csDefId	String		Id of cross section definition.
length	Double		Length (L) [m].
inletLossCoeff	Double		Inlet loss coefficient (ξ_i) [-].
outletLossCoeff	Double		Outlet loss coefficient (ξ_o) [-].
valveOnOff	Int		Flag for having valve or not (0=no valve, 1=valve).
valveOpeningHeight	*		Valve opening height (h_{valve}) [m].
			Table for loss coefficients of valve.
numLossCoeff	Int		Number of rows in table.
relOpening	Double[]		Relative valve opening (0.0 — 1.0) [-]. (number of values = numLossCoeff)

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
lossCoeff	Double[]		Loss coefficients (ξ_v) [-]. (number of values = numLossCoeff)
bedFrictionType	String		Friction type, possible values are listed in Section C.15.
bedFriction	Double		Friction Value.
subType	String	culvert	Defines the behaviour of the culvert. Possible values: culvert and invertedSiphon.
bendLossCoeff	Double		Bend loss coefficient (ξ_b) of the inverted siphon.

Remarks:
- ◇ Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- ◇ valveOpeningHeight is a relative height with respect of the invert level of the culvert.
- ◇ bendLosses is mandatory in case of an inverted siphon, but is not allowed in combination with a culvert.

C.12.4 Long culvert

The data block for a long culvert structure in the <structures.ini> file is described in this section. The geometrical shape is described by the parameters in the following table and these refer to the symbols in Figure 14.10. The geometrical details are further described in Section 14.7.1.

Table C.11: *Long culvert*

Keyword	Type	Default	Description
[Structure]			
type	String	longCulvert	Structure type; must read longCulvert.
numCoordinates, xCoordinates, yCoordinates			See Table C.7.
zCoordinates	Double		z-coordinates for the vertical placement of the structure. (number of values = numCoordinates) [m AD].
allowedFlowdir	String	both	Possible values: both, positive, negative, none.
width	Double		Width of the culvert's cross section shape [m].
height	Double		Height of the culvert's cross section shape [m].
frictionType	String		Roughness type associated with this cross section (see section C.15 for encoding).
frictionValue	Double		Roughness value; its meaning depends on the roughness type selected (only used if frictionType specified).
valveRelativeOpening	*	1.0	Relative valve opening $[0.0, 1.0]$ [-].

Remarks:
- ◇ Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.

- ◇ The location specification for a long culvert is only supported via x/yCoordinates, and these should define the actual placement of the pipe in positive flow direction (i.e., not perpendicular to the flow direction).
- ◇ When the optional friction parameters are not set, they default to the global MDU-settings `UnifFrictType` and `UnifFrictCoef1D`.
- ◇ `valveRelativeOpening` is *different* from the normal culvert's `relOpening`. The former directly reduces the discharge through a long culvert, with respect to the fully open state.

C.12.5 1D2D Long culvert

The data block for a 1D2D long culvert structure in the <structures.ini> file is described in this section. The geometrical shape is described by the parameters in the following table and these refer to the symbols in Figure 14.10. The geometrical details are further described in Section 14.7.1.

Table C.12: *Long culvert*

Keyword	Type	Default	Description
`[Structure]`			
`type`	String	`longCulvert`	Structure type; must read `longCulvert`.
`numCoordinates`, `xCoordinates`, `yCoordinates`			See Table C.7. Should match the coordinates of the branch.
`zCoordinates`	Double		z-coordinates for the vertical placement of the structure. (number of values = `numCoordinates`) [m AD].
`allowedFlowdir`	String	`both`	Possible values: `both`, `positive`, `negative`, `none`.
`width`	Double		Width of the culvert's cross section shape [m].
`height`	Double		Height of the culvert's cross section shape [m].
`branchId`	String		Branch ID of network branch belonging to culvert
`csDefId`	String		ID of cross definition belonging to culvert.
`valveRelativeOpening`	*	1.0	Relative valve opening $[0.0, 1.0]$ [-].

Remarks:

- ◇ Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or `realtime`, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- ◇ The location specification for a long culvert is only supported via x/yCoordinates, and these should define the actual placement of the pipe in positive flow direction (i.e., not perpendicular to the flow direction).
- ◇ When the optional friction parameters are not set, they default to the global MDU-settings `UnifFrictType` and `UnifFrictCoef1D`.
- ◇ `valveRelativeOpening` is *different* from the normal culvert's `valveOpeningHeight`. The former directly reduces the flow area through a long culvert, with respect to the fully open state, whereas the latter indirectly limits the flow via the table of loss coefficients.

C.12.6 Bridge

The data block for a bridge structure in the <structures.ini> file is described in this section. The geometrical shape and hydraulic treatment is further described in Section 14.8.

***Table C.13:** Bridge Structure definition.*

Keyword	Type	Default	Description
`[Structure]`			
`type`	String	`bridge`	Structure type; must read `bridge`.
`allowedFlowdir`	String		Possible values: `both`, `positive`, `negative`, `none`.
`Pillar definition:`			
`pillarWidth`	Double		Total width of pillars in cross direction [m].
`formFactor`	Double		Shape factor [-].
`Abutment definition:`			
`csDefId`	String		Id of Cross-Section Definition.
`shift`	Double		Vertical shift of the cross section definition [m]. Defined positive upwards.
`inletLossCoeff`	Double		Inlet loss coefficient [-], ξ_i in Equation (14.58).
`outletLossCoeff`	Double		Outlet loss coefficient [-], k in Equation (14.59).
`frictionType`	String		Friction type, possible values are listed in Section C.15.
`friction`	Double		Friction Value, used in friction loss in Equation (14.60).
`length`	Double		Length [m], L in Equation (14.60).

Remarks:

- ◇ At least one of the sections "Pillar definition" and "Abutment definition" must be present, but the other is optional.
- ◇ In case the "Pillar definition" is provided the effect of pillars are taken into account. In that case all of the items in the "Pillar definition" section are required parameters
- ◇ In case the "Abutment definition" is provided, the bridge results to a bridge with an own cross section definition. In that case all of the items in the "Abutment definition" section are required parameters
- ◇ The bridge treatment is equivalent to the one known from an *abutment bridge*, see Section 14.8 for details.
- ◇ The cross section is suggested to be an open cross section (that is, without bridge deck), but it can be of any type, because it is merely used to compute the wet area and perimeter.
- ◇ The `friction` and `frictionType` values are required in case of an abutment bridge, and will override the cross section's own roughness.

C.12.7 Pump

The data block for a pump structure in the <structures.ini> file is described in this section. The pump staging and triggering, and its hydraulic treatment is further described in Section 14.10.

***Table C.14:** Pump definition.*

Keyword	Type	Default	Description
`[Structure]`			
`type`	String	`pump`	Structure type; must read `pump`.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
orientation	String	positive	Pump orientation. Possible values: positive, negative.
controlSide	String		On which side[s] built-in pump triggering is active. *Only required/used when* numStages *> 0.* Possible values: ◇ suctionSide: Suction side control. ◇ deliverySide: Delivery side control. ◇ both: Suction and Delivery side control.
numStages	Int	0	Number of trigger stages in pump. Set to 0 to disable built-in triggering.
capacity	*		Pump capacity [m^3/s]. (number of values = max(1, numStages))
startLevelSuctionSide	Double[]		Start level suction side [m AD]. (number of values = numStages)
stopLevelSuctionSide	Double[]		Stop level suction side [m AD]. (number of values = numStages)
startLevelDeliverySide	Double[]		Start level at delivery side [m AD]. (number of values = numStages)
stopLevelDeliverySide	Double[]		Stop level at delivery side [m AD]. (number of values = numStages)
numReductionLevels	Int	0	Number of levels in reduction table.
head	Double[]		Head, the difference in waterlevel (delivery side - suction side) [m]. (number of values = numReductionLevels)
reductionFactor	Double[]		Reduction factor [-]. (number of values = numReductionLevels)

Remarks:

- Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- A pump's orientation defines the direction of positive capacity with respect to the spatial orientation. For example, when a pump is pumping against a network branch's direction, it may be convenient to set orientation = negative and use positive capacity values (as opposed to using negative capacity values).
- Built-in triggering based on stages can not be combined with time series or controlling by RTC. Therefore, if the capacity is a time series filename or is controlled by RTC, the number of stages (numStages) may only be equal to 0.
- When built-in triggering is on (numStages > 0), then all capacity values must be nonnegative (use orientation when needed). On the other hand, a (non-triggered) capacity as a scalar, or timeseries, or via RTC may be negative.
- The discharge of a 2D pump is equally distributed over the different flow links.

C.12.8 Orifice

The data block for an orifice structure in the <structures.ini> file is described in this section. The geometrical shape and hydraulic treatment is further described in Section 14.5.

Table C.15: *Orifice definition.*

Keyword	Type	Default	Description
[Structure]			
type	String	orifice	Structure type; must read orifice.
allowedFlowdir	String	both	Possible values: both, positive, negative, none.
crestLevel	*		Crest level [m AD]
crestWidth	Double	$\sum w_u$	Crest width [m]
gateLowerEdgeLevel	*		Position of gate door's lower edge. [m AD]
corrCoeff	Double	1.0	Correction coefficient [-]
useVelocityHeight	logical	true	Flag indicates whether the velocity height is to be calculated or not.
useLimitFlowPos	logical	false	Flag for setting the flow limiter for positive flow to either unlimited or limited (false = unlimited, true = limited).
limitFlowPos	Double		Maximum positive flow [m^3/s], only applied when uselimitflowpos is true.
useLimitflowNeg	logical	false	Flag for setting the flow limiter for negative flow to either unlimited or limited (false = unlimited, true = limited).
limitFlowneg	Double		Maximum negative flow [m^3/s], only applied when uselimitflowneg is true.

Remarks:

- Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- crestWidth is optional. Default is (the sum of) the width[s] of the flow link[s]. If the given value is larger than this sum, it is automatically lowered to this default.
- If the crestWidth of a 2d structure is smaller than the sum of the widths of the 2d links, the crestWidth of the individual structure elements is reduced symmetrical from the outside inwards.
- The maximum flow limitation by settings (use)LimitflowPos and (use)LimitflowNeg is further described in Table 14.5.

C.12.9 Gate

The data block for a gate structure in the <structures.ini> file is described in this section. This structure is currently only supported in 2D. For 1D models, consider using a general structure (Section C.12.10) or an orifice (Section C.12.8).

Table C.16: *Gate definition.*

Keyword	Type	Default	Description
[Structure]			
type	String	gate	Structure type; must read gate.
polylineFile	String		Filename of a <*.pli> file containing x-/y-points of the gate's position. (2D only)
crestLevel	*		Crest level. [m AD]
crestWidth	Double	$\sum w_u$	Crest width. [m]

(continued on next page)

(continued from previous page)

Keyword		Default	Description
gateLowerEdgeLevel	*		Position of gate door's lower edge. [m AD]
gateHeight	Double		Height of gate door. Needed for possible overflow across door. [m]
gateOpeningWidth	*	0.0	Opening width between gate doors, should be smaller than (or equal to) crestWidth. Use 0.0 for a vertical door. [m]
gateOpeningHorizontalDirection	String	symmetric	Horizontal opening direction of gate door[s]. Possible values are: symmetric, fromLeft, fromRight.

Remarks:

- Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or realtime, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- crestWidth is optional. Default is (the sum of) the width[s] of the flow link[s]. If the given value is larger than this sum, it is automatically lowered to this default.
- If the crestWidth of a 2d structure is smaller than the sum of the widths of the 2d links, the crestWidth of the individual structure elements is reduced symmetrical from the outside inwards.

C.12.10 General Structure

The data block for a general structure in the <structures.ini> file is described in this section. The geometrical shape is described by the parameters in the following table and these refer to the symbols in Figures 14.2 and 14.3. The hydraulic treatment is further described in Section 14.3.

Table C.17: *Specific input for the general structure.*

Keyword	Type	Default	Description
[Structure]			
type	String	generalStructure	Structure type; must read generalStructure.
allowedFlowdir	String	both	Possible values: both, positive, negative, none.
upstream1Width	Double	10.0	w_{u1} [m]
upstream1Level	Double	0.0	z_{u1} [m AD]
upstream2Width	Double	10.0	w_{u2} [m]
upstream2Level	Double	0.0	z_{u2} [m AD]
crestWidth	Double	10.0	w_s [m]
crestLevel	*	0.0	z_s [m AD]
crestLength	Double	0.0	The crest length across the general structure. When the crest length $>$ 0, the extra resistance for this structure will be $l_s \cdot g/(C^2 \cdot \mathrm{waterdepth})$. [m]
downstream1Width	Double	10.0	w_{d1} [m]
downstream1Level	Double	0.0	z_{d1} [m AD]
downstream2Width	Double	10.0	w_{d2} [m]

(continued on next page)

(continued from previous page)

Keyword	Type	Default	Description
`downstream2Level`	Double	0.0	z_{d2} [m AD]
`gateLowerEdgeLevel`	*	11.0	Position of gate door's lower edge. [m AD]
`posFreeGateFlowCoeff`	Double	1.0	Positive free gate flow corr.coeff. c_{gf} [-].
`posDrownGateFlowCoeff`	Double	1.0	Positive drowned gate flow corr.coeff. c_{gd} [-].
`posFreeWeirFlowCoeff`	Double	1.0	Positive free weir flow corr.coeff. c_{wf} [-].
`posDrownWeirFlowCoeff`	Double	1.0	Positive drowned weir flow corr.coeff. c_{wd} [-].
`posContrCoefFreeGate`	Double	1.0	Positive gate flow contraction coefficient μ_{gf} [-].
`negFreeGateFlowCoeff`	Double	1.0	Negative free gate flow corr.coeff. c_{gf} [-].
`negDrownGateFlowCoeff`	Double	1.0	Negative drowned gate flow corr.coeff. c_{gd} [-].
`negFreeWeirFlowCoeff`	Double	1.0	Negative free weir flow corr.coeff. c_{wf} [-].
`negDrownWeirFlowCoeff`	Double	1.0	Negative drowned weir flow corr.coeff. c_{wd} [-].
`negContrCoefFreeGate`	Double	1.0	Negative gate flow contraction coefficient μ_{gf} [-].
`extraResistance`	Double	0.0	Extra resistance [-].
`gateHeight`	Double	1d10	
`gateOpeningWidth`	*	0.0	Opening width between gate doors, should be smaller than (or equal to) `crestWidth`. [m]
`gateOpeningHorizontalDirection`	String	`symmetric`	Horizontal opening direction of gate door[s]. Possible values are: `symmetric`, `fromLeft`, `fromRight`.
`useVelocityHeight`	logical	true	Flag indicates whether the velocity height is to be calculated or not.

Remarks:

- For an extensive description on these parameters see Section 14.3.
- Items marked with (*) can either be a scalar value, or a time series file (<*.bc>, see more details in Section C.12.13/<*.tim>) or `realtime`, to indicate that this parameter gets it values supplied by, e.g., realtime control.
- `crestWidth` is optional. Default is (the sum of) the width[s] of the flow link[s]. If the given value is larger than this sum, it is automatically lowered to this default.
- If the `crestWidth` of a 2d structure is smaller than the sum of the widths of the 2d links, the `crestWidth` of the individual structure elements is reduced symmetrical from the outside inwards.

C.12.11 Dambreak

The data block for a dambreak in the <structures.ini> file is described in this section. A dambreak is not really a hydraulic structure, but it has so many similarities with hydraulic structure input that it is also placed in the same structure file. The dam break behavior and hydraulic treatment is further described in Section 14.2.

Table C.18: *Specific input for the dambreak.*

Keyword	Type	Default	Description
`[Structure]`			
`type`	String	`dambreak`	Structure type; must read `dambreak`.
`startLocationX`	Double		x-coordinate of breach starting point.
`startLocationY`	Double		y-coordinate of breach starting point.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
`algorithm`	Int		Breach growth algorithm. Possible values are: ◇ `1`: van der Knaap (2000) ◇ `2`: Verheij–van der Knaap (2002) ◇ `3`: Predefined time series, see `dambreakLevelsAndWidths`
`crestLevelIni`	Double		Initial breach level $z_{\text{crest level}}$ [m AD].
`breachWidthIni`	Double		Initial breach width B_0 [m].
`crestLevelMin`	Double		Minimal breach level z_{min} [m AD].
`t0`	Double		Breach start time T_{start} [s].
`timeToBreachToMaximumDepth`	Double		$t_{\text{Phase 1}}$ [s].
`f1`	Double		Factor f_1 [-].
`f2`	Double		Factor f_2 [-].
`uCrit`	Double		Critical flow velocity u_c for erosion [m/s].
`waterLevelUpstreamLocationX`	Double		(optional) x-coordinate of custom upstream water level point.
`waterLevelUpstreamLocationY`	Double		(optional) y-coordinate of custom upstream water level point.
`waterLevelDownstreamLocationX`	Double		(optional) x-coordinate of custom downstream water level point.
`waterLevelDownstreamLocationY`	Double		(optional) y-coordinate of custom downstream water level point.
`waterLevelUpstreamNodeId`	String		(optional) Node Id of custom upstream water level point.
`waterLevelDownstreamNodeId`	String		(optional) Node Id of custom downstream water level point.
`dambreakLevelsAndWidths`	String		(only when `algorithm=3`) Filename of <*.tim> file (Section C.4) containing the breach levels and widths.

Remarks:

- ◇ The location specification for a dambreak currently only supports polyline coordinates via `x/yCoordinates` (see Table C.7).
- ◇ On the same location as the dambreak polyline, a fixed weir polyline must be present (see Section C.8).
- ◇ The dambreak supports 2D grids, and also has basic support for 1D2D connections. For a 1D river with lateral couplings to 2D flood plains, the polyline must lie alongside the river, intersecting the 1D2D connections. Currently, the resolution of the 1D network must be similar to the 2D grid resolution, i.e., typically one 1D2D link per 2D grid cell.
- ◇ Breach start time `t0` is considered relative to the model's `[time] RefDate` (see Appendix A).
- ◇ `crestLevelIni` is also required for time series dambreaks.
- ◇ When `algorithm=1`, Van der Knaap(2000), currently only the default material type clay is supported. This means that the related coefficients in Equation (14.13) are for clay (see Table 14.3).
- ◇ When `algorithm=3`, the time series provided in the filename via `dambreakLevelsAndWidths` in minutes (also see Section C.4) are considered relative to breach start time `t0` (which is in in seconds).
- ◇ When `algorithm=2`, Verheij–van der Knaap (2002), the up- and downstream water levels h_{up} and h_{down} are computed as the average water levels for all grid cell[s] upstream or downstream

of the breach gap, respectively.
Alternatively, users can specify two points so that the water levels at these points are used for h_{up} and h_{down}. (This solves any unwanted dependency on local grid cell size.) These two points can be provided either by node Ids or coordinates. For instance, the point used for upstream water level can be given by `waterLevelUpstreamNodeId` or `waterLevelUpstreamLocationX`. (same for `Down`).

A example of a timfile for a dambreach is:

```
*  Time breachlevel breachwidth
     0     18.5          0
    60     17.5         20
   120     16.5         50
   240     15.0        100
   480     15.0        200
  2880     15.0        200
* end input
```

C.12.12 Compound Structure

The data block for a compound structure in the <structures.ini> file is described in this section. A compound structure consists of a number of structures defined in the structure file that are combined to one compound structure. The hydraulic treatment is further described in Section 14.11.

Table C.19: *Specific input for the compound structure.*

Keyword	Type	Default	Description
[Structure]			
type	String	compound	Structure type; must read compound.
numStructures	int		Number of individual structures in compound structure.
structureIds	String		Semicolon separated list of structure ids.

Remarks:

- ◇ `structureIds` is a semicolon separated list, where leading and trailing spaces will be trimmed from each structure id in the list.
- ◇ `structureIds` can contain spaces in the individual structure ids, no quotation is needed.
- ◇ A compound structure should *not* contain a location specification. This will be retrieved from the individual structures.
- ◇ A compound structure *does* require its own `id` and `name` fields, because the output in the history file will also contain total output for the compound's underlying structures together and this will appear under its own compound `id`.
- ◇ All individual structures in a compound structure must be snapped to exactly the same flow link[s].
- ◇ The referenced individual structures may appear anywhere in the structure file.

The structure INI file is the preferred format for hydraulic structure input, and supersedes some of the old structure quantity names in Table C.5 under "Structure parameters".

```
[structure]
type                    = weir                  # Type of structure
id                      = weir01                # Name of the structure
polylinefile            = weir01.pli            # *.pli; Polyline location for structure
CrestLevel              = weir01_crest.tim      # Crest level in [m]

[structure]
type                    = gate                  # Type of structure
id                      = gate01                # Name of the structure
```

```
polylinefile          = gate01.pli          # *.pli; Polyline location for structure
SillLevel             = gate01_crest.tim    # Sill (crest) level in [m]
SillWidth             = 39.5                # (Optional) sill width in [m], between
                                            # the fixed side walls.
DoorHeight            = 8.54                # Gate door height in [m], used for
                                            # detecting flow across door.
LowerEdgeLevel        = gate01_ledge.tim    # Position of gate's lower edge in [m],
                                            # used for flow underneath door.
OpeningWidth          = gate01_opening.tim  # (Optional) opening width between two
                                            # sideways closing gate doors.
HorizontalOpeningDirection = from_left/from_right/symmetric # (Optional) Opening
                                            # direction of the gate door(s).

[structure]
type                  = pump                # Type of structure
id                    = pump01              # Name of the structure
polylinefile          = pump01.pli          # *.pli; Polyline location for structure
Capacity              = pump01_cap.tim      # Pumping capacity in [m3/s]
```

C.12.13 Structure defined with a time series

Some quantities of a certain structure can be given as time dependent values, using a time series. We use an example of defining a pump's capacity as a time series, following the steps below:

- ◇ 1: in the structure INI file (which was specified as StructureFile in the MDU file), when defining a structure in one [Structure] block, specify a BC file as value of the quantity keyword. For example, the following [Structure] block defines a pump, whose capacity is specified by structureforcing.bc file.

```
[Structure]
id                    = Pump_bc_tim
name                  = Pump_bc_tim
branchId              = T3_B1
chainage              = 1050.000
type                  = pump
capacity              = structureforcing.bc
```

- ◇ 2: In the specified BC file, when defining a [forcing] block, keyword name should have the same name as the name of the structure in the structure *.ini file. Keyword function should be given as timeseries. Keyword quantity should have a quantity name, which must be equal to one of the quantity names in Table C.20, other quantity names will not be recognized. Using the pump example again, to specify a time series in the corresponding *.bc file, i.e. file structureforcing.bc, the [forcing] block can be defined as below.

```
[forcing]
name                  = Pump_bc_tim
function              = timeseries
time-interpolation    = linear
quantity              = time
unit                  = minutes since 2010-01-01 00:00:00
quantity              = pump_capacity
unit                  = m3 s-1
0 0
60 0
65 1
120 1
125 2.5
180 2.5
240 0
1440 0
```

Remark:

- ◇ Quantities of structures that can be specified as a time series are listed in Table C.20. Keyword quantity in the *.bc file must be exactly equal to one of the quantities in Table C.20. Otherwise it cannot be recoganized.

Table C.20: *Quantities that can be set by a time series.*

structure	quantity keywords in the .bc file
`pump`	`pump_capacity`
`culvert`	`culvert_valveOpeningHeight`
`weir`	`weir_CrestLevel`

`orifice`	`orifice_CrestLevel,`
	`orifice_GateLowerEdgeLevel`

`general structure`	`general_structure_CrestLevel,`
	`orifice_GateLowerEdgeLevel,`
	`general_structure_GateLowerEdgeLevel,`
	`general_structure_CrestWidth,`
	`general_structure_GateOpeningWidth`

C.13 Space varying wind and pressure

Warning:

◇ This functionality contains bugs and therefore the user is advised to check the space varying wind and pressure via the map output file. Compare a wind and pressure field, at a certain time stamp on the map file, with the wind and pressure field you supplied at that time stamp. If the space varying wind and pressure appears to be wrong for your application, then please contact Delft3D Support.

In many cases the space varying wind data is provided by a meteorological station. This data is often defined on a different grid than the computational grid used in D-Flow FM. Translating these files into files defined on the grid of the computational engine is often a lengthy process and can result in huge files. This feature facilitates the reading of the meteorological data on its own grid and interpolates the data internally to the grid of D-Flow FM. D-Flow FM can handle three types of meteorological input:

1 Time- and space-varying wind on an equistant grid.
2 Time- and space-varying wind on a curvilinear grid.
3 Time- and space-varying wind on a Spiderweb grid.
4 Time- and space-varying wind in a netCDF file.

C.13.1 Meteo on equidistant grids

File contents	Time-series of a space varying wind and atmospheric pressure defined on an equidistant (Cartesian or spherical) grid.
File format	Free formatted or unformatted, keyword based.
Generated	Some offline program.

Remark:

◇ The keywords are case insensitive.

Header description for the wind velocity files:

Keywords	**Value**	**Description**
`FileVersion`	1.03	version of file format
`Filetype`	meteo_on_equidistant_grid	meteo input on equidistant grid
`NODATA_value`	*free*	value used for input that is to be neglected

Keywords	**Value**	**Description**
`n_cols`	*free*	number of columns used for wind datafield
`n_rows`	*free*	number of rows used for wind datafield
`grid_unit`	m *or* degree	unit of distances on the grid in both x- and y-direction
`x_llcorner`	*free*	x-coordinate of lower left corner of lower left grid cell (in units specified in `grid_unit`
`y_llcorner`	*free*	y-coordinate of lower left corner of lower left grid cell (in units specified in `grid_unit`
`x_llcenter`	*free*	x-coordinate of centre of lower left grid cell (in units specified in `grid_unit`
`y_llcenter`	*free*	y-coordinate of centre of lower left grid cell (in units specified in `grid_unit`
`dx`	*free*	gridsize in x-direction in units specified in `grid_unit`
`dy`	*free*	gridsize in y-direction in units specified in `grid_unit`
`n_quantity`	1	number of quantities specified in the file
`quantity1`	x_wind *or* y_wind	the velocity component given in unit `unit1`
`unit1`	m s-1	unit of `quantity1`: metre/second

The user must specify the location of the equidistant grid on which the meteorological data is specified. While the user can specify the meteorological grid by means of grid cell's corners or grid cell's centres, the meteorological data is always specified in the cell centre. If one has the location of the lower left corner of the lower left grid cell, one can specify the starting point of the grid using keywords `x_llcorner` and `y_llcorner`. If one has the location of the cell centre of the lower left grid cell, one should use the keywords `x_llcenter` and `y_llcenter`. Using the first option, the first data value is placed at (x_llcorner+$\frac{1}{2}dx$, y_llcorner+$\frac{1}{2}dy$), which is the cell centre of cell (1,1). Using the latter option, the first data value is placed at (x_llcenter, y_llcenter), which is again the cell centre of cell (1,1), i.e. the data values are always placed at the cell centres of the meteorological grid. Note that the lower left grid cell is defined to be the grid cell with index (1,1). When using the option of meteorological data on a separate curvilinear grid, the origin and orientation of the data set can be chosen freely with respect to the grid on which it is specified, see section C.13.2 for details.

Time definition and data block description for the wind velocity files

Keywords	**Value**	**Description**
`Time`	*fixed format described below*	time definition string

The time definition string has a fixed format, used to completely determine the time at which a dataset is valid. The time definition string has the following format:

TIME *minutes/hours since* YYYY-MM-DD HH:MM:SS TIME ZONE, e.g.

```
360 minutes since 2008-07-28 10:55:00 +01:00
```

The format of the string is completely fixed. No extra spaces or tabs can be added between the different parts of the definition. The time definition is followed by the datablock of input values corresponding to the specified time. The data block contains values for the wind velocity in x- or y-direction for *n_cols* by *n_rows* points, starting at the top left point. The time definition and the data block are repeated for each time instance of the time-series.

The atmospheric pressure file

The header for the atmospheric pressure is similar to that of the wind velocity files, except for the following differences.

Keywords	Value	Description
`quantity1`	air_pressure	air pressure
`unit1`	Pa *or* mbar	unit of `quantity1`: Pascal or millibar

The specification of the time definition and the data block is fully conform the wind velocity files.

File version and conversion

The current description holds for `FileVersion` 1.03. The table below shows the latest modifications in the file format (and version number).

FileVersion	Modifications
`1.03`	Use of keyword `Value_pos` to indicate the position of the lower left corner of the grid replaced by use of the combination of keywords: `x_llcorner` and `y_llcorner` or `x_llcenter` and `y_llcenter`
`1.02`	No changes for this meteo input type, but for the meteo type *meteo_on_spiderweb_grid*
`1.01`	Changed keyword `MeteoType` to `FileType` Changed fixed value of input type (Keyword `Filetype`) from *ArcInfo* to *meteo_on_equidistant_grid*

Restrictions:

- ◇ The contents of the file will not be checked on its domain.
- ◇ Keywords are followed by an equal sign ’=’ and the value of the keyword.
- ◇ When a keyword has value *free*, the value of this keyword is free to choose by the user. When only one value is given for a keyword, this keyword has a fixed value and when 2 or more options are shown, the user can choose between those values.

- Times must be specified exactly according to the time definition. See the examples shown in this section.
- The atmospheric pressure file must use the same grid definition and time frame as the files for the wind velocity components.
- The unit of the meteo grid must be the same as the computational grid, i.e. both with `grid_unit` = [m] or both with `grid_unit` = [degree].
- Input items in a data block are separated by one or more blanks.
- The wind components are specified at the cell centres (water level points) of the numerical grid.
- The wind components are specified in the west-east (`x_wind`) and south-north directions (`y_wind`).

Remarks:

- The time definition in the meteo files contains the number of minutes or hours since a reference date and time in a certain time zone. The reference time and time zone may differ from those of the simulation. During a simulation the computational engine will search in the meteo file for the current simulation time and interpolate between neighbouring times if necessary. Possible differences in time zone will be accounted for by shifting the meteo input data. The reference times within the time definition string may vary in a meteo file, i.e. it is possible to attach new input with a different reference time, behind the last data block. Consecutive times must always be increasing in the input file.
- Comments can be added after pound signs (#). These are not read.

Example of a file containing wind in x-direction (west-east)

The data blocks in this example are the result of the following FORTRAN statements:

```
do j = nrows,1,-1
   write(out,*) (xwind(i,j),i=1,ncols)
enddo
```

The x-wind velocity file for a 3 (n_cols) by 4 (n_rows) grid has the following layout:

```
FileVersion      =    1.03
filetype         =    meteo_on_equidistant_grid
NODATA_value     =    -999.000
n_cols           =    3
n_rows           =    4
grid_unit        =    degree
x_llcenter       =    -12.000
y_llcenter       =     48.000
dx               =    0.12500
dy               =    0.083333333
n_quantity       =    1
quantity1        =    x_wind
unit1            =    m s-1
TIME =      0.0 hours since 2008-01-15 04:35:00 +00:00
2   3.0  3.6
3   4.5  2
2.2 1    2.3
1.2 0.7 -0.4
TIME =      6.0 hours since 2008-01-15 04:35:00 +00:00
-1.1 -2.3 -3.6
-3.2  0.8  1.1
 2.2 -1   -1.6
 1.2 -0.7 -0.4
```

This results in an x-component of wind velocity given - in [m/s] - on a spherical, 3 by 4, equidistant grid, with grid sizes given by *dx* and *dy* (in degrees) and where the centre point of the lower left cell of the grid lies in (longitude, latitude) (-12.0, 48.0) on the globe. Data is given at two times: 0 and 6 hours since January 15th, 2008, 4:35 AM, in UTC+0.

Remark:

◇ The layout of the data blocks is from north to south (whereas most of the other files in Delft3D-FLOW, such as the curvilinear grid file, are ordered from south to north).
Usually the signal corresponds to Mean Sea Level. One actually wants to prescribe an input signal corresponding to the local pressure prescribed by the space varying meteo input. To this end, it is possible to specify an average pressure - which should correspond to your input signal on the open boundaries - which is then used to determine local pressure gradients that need to be applied along the open boundaries to obtain an input signal that is consistent with the local atmospheric pressure. Using this keyword one can specify an average pressure that is used on all open boundaries, independent of the type of wind input. The pressure must be specified in N/m^2. An example:

C.13.2 Curvilinear data

File contents	Time-series of a space varying wind and atmospheric pressure defined on a curvilinear (Cartesian or spherical) grid.
File format	Free formatted or unformatted, keyword based.
Generated	Some offline program.

Remark:

◇ The keywords are case insensitive.

Header description for the wind velocity files:

Keywords	**Value**	**Description**
FileVersion	1.03	version of file format
Filetype	meteo_on_curvilinear_grid	meteo input on curvilinear grid
NODATA_value	*free*	value used for input that is to be neglected
grid_file	*free*.grd	name of the curvilinear grid file on which the data is specified
first_data_value	grid_llcorner *or* grid_ulcorner *or* grid_lrcorner *or* grid_urcorner	*see example below*
data_row	grid_row *or* grid_column	*see example below*
n_quantity	1	number of quantities specified in the file
quantity1	x_wind *or* y_wind	the velocity component given in unit unit1
unit1	m s-1	unit of quantity1: metres/second

Time definition and data block description for the wind velocity files

For a description of the time definition and data block see section C.13.1.

The atmospheric pressure file

For a description of the atmospheric file see section C.13.1.

File version and conversion

The current description holds for `FileVersion` 1.03. The table below shows the latest modifications in the file format (and version number).

FileVersion	Modifications
`1.03`	Fixed bug in interpolation of data from meteo grid to computational grid: Conversion script mirrored data set erroneously. This was treated correctly by meteo module. Fixed both the conversion script and the meteo module together: Required modification in meteo input file: Change `first_data_value` = grid_llcorner into grid_ulcorner or vice versa or Change `first_data_value` = grid_lrcorner into grid_urcorner or vice versa
`1.02`	No changes for this meteo input type, but for the meteo type *meteo_on_spiderweb_grid*
`1.01`	Changed keyword `MeteoType` to `FileType` Changed keyword `Curvi_grid_file` to `Grid_file` Changed fixed value of input type (Keyword `Filetype`) from *Curvi* to *meteo_on_curvilinear_grid*

Restrictions:

- The restrictions for space varying wind and pressure on a separate curvilinear grid are the same as for space varying wind and pressure on an equidistant grid, described in section C.13.1. A differerence is that the data values on the curvilinear grid are not specified in the cell centres, but in the grid points (cell corners).
- The unit of the meteo grid must be the same as the computational grid, i.e. both with `grid_unit` = [m] or both with `grid_unit` = [degree].

Remark:

- The remarks for space varying wind and pressure on a separate curvilinear grid are the same as for space varying wind and pressure on an equidistant grid, described in section C.13.1.

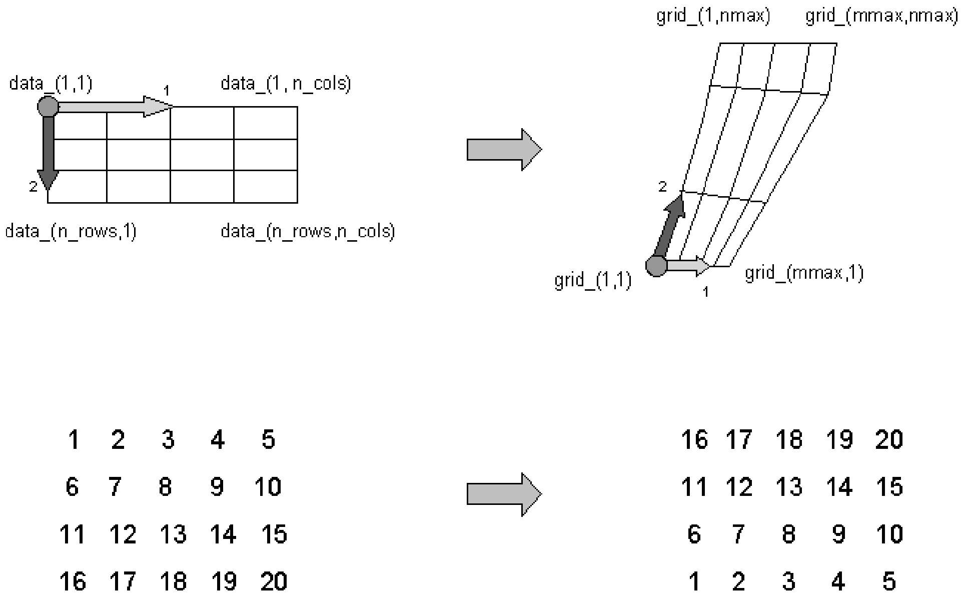


Figure C.1: *Illustration of the data to grid conversion for meteo input on a separate curvilinear grid*

Example:

A file for input of x-velocity (in west-east direction) on a 4 (n_rows) by 5 (n_cols) curvilinear grid, where the meteorogical data is mirrored vertically with respect to the grid, has the following layout:

```
FileVersion      =    1.03
filetype         =    meteo_on_curvilinear_grid
NODATA_value     =    -999.000
grid_file        =    curviwind.grd
first_data_value =    grid_llcorner
data_row         =    grid_row
n_quantity       =    1
quantity1        =    x_wind
unit1            =    m s-1
TIME =       0.0 minutes since 1993-06-28 14:50:00 -02:00
 1   2   3   4   5
 6   7   8   9  10
11  12  13  14  15
16  17  18  19  20
TIME =       600.0 minutes since 1993-06-28 14:50:00 -02:00
 1   2   3   4   5
 6   7   8   9  10
11  12  13  14  15
16  17  18  19  20
```

This results in an x-component of velocity given - in [m/s] - on the curvilinear grid specified in file <curviwind.grd>. The data set will be mirrored such that the first value of the data (upper left corner, in the example '1') corresponds to the lower left corner of the grid (point (1,1)) and a row of data corresponds to a row on the grid, see Figure C.1. Data is given at two times: 0 and 600 minutes since June 28th, 1993, 14:50 PM, in UTC-2.

C.13.3 Spiderweb data

Remarks:

- The spiderweb file format used in D-Flow FM forms a subset of the the format described below. The extensive format is accepted by D-Flow FM, but not all keywords are required and/or meaningful.
- The keywords grid_unit and air_pressure_reference are ignored.
- The keyword n_quantity is ignored; the number of quantities is always 3.
- The keywords quantity1, quantity1 and quantity1 are ignored, the order of the variables in the file is assumed to be wind speed, wind direction and pressure drop.
- The keywords unit1 and unit2 are ignored.
- The keyword unit3 is optional; if omitted or different from 'mbar', Pa is silently assumed

Cyclone winds are governed by a circular motion, combined with a cyclone track. This type of wind is generally very difficult to implement on a curvilinear grid. This feature facilitates the reading of the so-called Spiderweb files and interpolates the wind and pressure data internally to the computational grid. A special feature of the space varying wind and pressure on the Spiderweb grid is that it can be combined with one of the other meteorological input options described in this manual, i.e. to either uniform wind and pressure, or to one of the space varying wind and pressure options, see section C.13.

File contents	Time-series of a space varying wind and atmospheric pressure defined on a Spiderweb grid. This grid may be specified in Cartesian or spherical coordinates.
File format	Free formatted or unformatted, keyword based.
Generated	Some offline program.

Remarks:

- The keywords are case insensitive.
- Space varying wind and pressure on a Spiderweb grid is added to other wind input and the wind fields are interpolated and combined in and around the cyclone.

Header description of the Spiderweb wind and pressure file:

Keywords	Value	Description
FileVersion	1.03	version of file format
Filetype	meteo_on_spiderweb_grid	meteo input on Spiderweb grid
NODATA_value	*free*	value used for input that is to be neglected
n_cols	*free*	number of gridpoints in angular direction
n_rows	*free*	number of gridpoints in radial direction
grid_unit	m *or* degree	unit of the Spiderweb grid
spw_radius	*free*	radius of the spiderweb given in units given by spw_rad_unit
spw_rad_unit	m	unit of the Spiderweb radius
spw_merge_frac	[0.0,1.0]	fraction of the Spiderweb radius where merging starts of the background wind with the Spiderweb wind. Default is 0.5
air_pressure	air_pressure_default_from	Both keyword and value are too

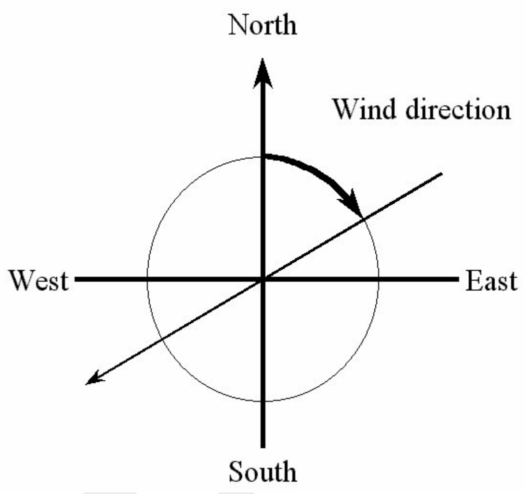


Figure C.2: *Wind definition according to Nautical convention*

Keywords	**Value**	**Description**
_reference	_computational_engine *or free*	long to fit on one line. Reference value related to p_drop is the default air pressure of the computional engine *or* the value specified. If missing, p_drop is extracted from the actual atmospheric pressure.
n_quantity	3	number of quantities specified in the file
quantity1	wind_speed	wind speed given in unit unit1
quantity2	wind_from_direction	direction where the wind is coming from given in unit unit2
quantity3	p_drop	drop in atmospheric pressure given in unit unit3
unit1	m s-1	unit of quantity1: metres/second
unit2	degree	unit of quantity2: degrees
unit3	Pa *or* mbar	unit of quantity3: Pascal or millibar

Time definition and data block description

For a description of the time definition see section C.13.1.

Cyclone track information:

For each time in the time series of space varying wind and pressure on a Spiderweb grid, the position of the cyclone eye (and thus also the spiderweb grid) must be given, as well as the drop of atmospheric pressure in the cyclone eye.

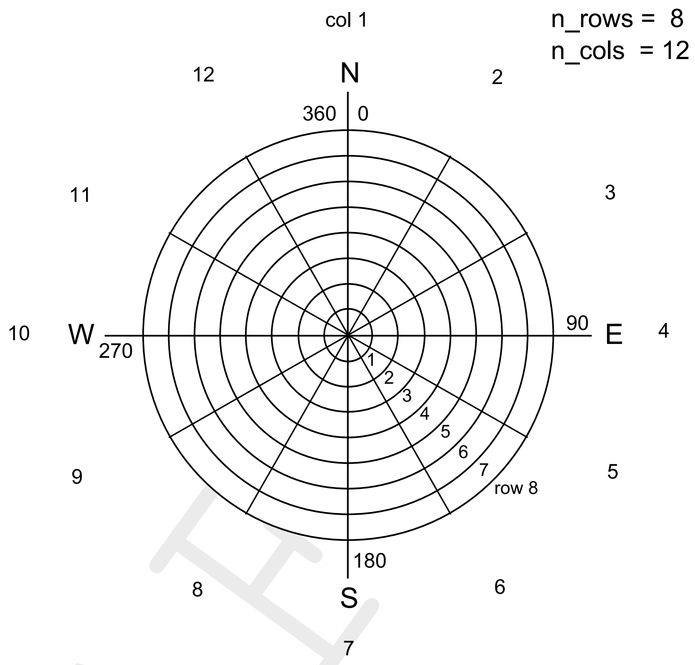


Figure C.3: Spiderweb grid definition

File version and conversion

The current description holds for `FileVersion` 1.03. The table below shows the latest modifications in the file format (and version number).

FileVersion	Modifications
`1.03`	No changes for this meteo input type
`1.02`	Changed the use of keyword `n_rows` and `n_cols`. The radius of the cyclone is divided in *n_rows* rings of width: spw_radius/n_rows [m] and the circle is divided in *n_cols* parts of $2\pi/n\_cols$ [rad].
`1.01`	Changed keyword `MeteoType` to `FileType` Changed fixed value of input type (Keyword `Filetype`) from *Spiderweb* to *meteo_on_spiderweb_grid*

Restriction:

- The restrictions for space varying wind and pressure on a Spiderweb grid are the same as for space varying wind and pressure on an equidistant grid, described in section C.13.1.

Remarks:

- The remarks for space varying wind and pressure on a separate curvilinear grid are the same as for space varying wind and pressure on an equidistant grid, described in section C.13.1.
- The Spiderweb grid is circular and the definitions of the number of rows `n_rows` and the number of columns `n_cols` is therefore different then for the other meteo input formats. For the Spiderweb grid, the number of rows determines the grid size in radial direction. The number of columns defines the grid size in angular direction. See Figure C.3.
- The wind is specified according to the nautical convention, i.e. wind from the true North has direction zero and the wind turns clockwise with an increasing angle. See Figure C.2.

Example:

A file for input of space varying wind and pressure on a 5x3 Spiderweb grid, has the following layout:

```
FileVersion            = 1.03
filetype               = meteo_on_spiderweb_grid
NODATA_value           = -999.000
n_cols                 = 3
n_rows                 = 5
grid_unit              = degree
spw_radius             = 600000.0
spw_rad_unit           = m
air_pressure_reference = air_pressure_default_from_computational_engine
n_quantity             = 3
quantity1              = wind_speed
quantity2              = wind_from_direction
quantity3              = p_drop
unit1                  = m s-1
unit2                  = degree
unit3                  = Pa
TIME          =    0.0 hours since 1997-07-14 03:00:00 -06:00
x_spw_eye     = 115.1
y_spw_eye     = 18.9
p_drop_spw_eye = 5300.0
1.38999       1.38261       1.38315
1.28251       1.34931       1.22571
1.27215       1.31214       1.32451
1.38999       1.86592       2.87732
1.43899       1.24912       2.21519
60.0000       180.0000      270.0000
28.7500       20.0000       31.2500
42.5000       53.7500       65.0000
49.3400       60.2400       81.5200
51.4100       62.0000       43.1200
5301.280      5294.490      5156.240
5043.460      5112.040      5264.020
5140.020      5202.520      5411.210
5294.730      5285.760      5235.250
5242.530      5156.190      5124.240
TIME          =      1.0 hours since 1997-07-14 03:00:00 -06:00
x_spw_eye     = 114.8
y_spw_eye     = 18.8
p_drop_spw_eye = 5250.0
1.35763       1.35763       1.35763
1.35763       1.87273       2.24784
1.92214       2.47836       2.17266
1.87662       2.72116       2.82375
1.26585       2.24146       2.38722
159.0000      346.5200      290.6400
342.3200      282.1400      20.2400
10.7500       25.5300       36.4500
61.8400       81.6200       45.5100
49.5250       56.7500       75.1300
5314.520      5104.490      5287.240
5124.240      5285.760      5252.420
5152.460      5247.040      5222.020
5242.020      5223.520      5475.210
5244.270      5211.210      4998.110
```

This results in the following set of meteo data. Velocities given in [m/s] and pressure drops in [Pa] on a Spiderweb grid which is given in spherical coordinates (`grid_unit` = degree). The cyclone and spiderweb grid have a radius of 600 km. The grid is 5x3, which means the radius is divided in five parts of 120 km and the 360 degrees are divided in 3 parts of 120 degrees each. Wind speeds, wind directions and pressure drops are given at two times: 0 and 1.0 hour since July 14th, 1997, 03:00 AM, in UTC-6. Between these two times the cyclone eye moves from (longitude, latitude) (115.1, 18.9) to

(114.8, 18.8) on the globe and the pressure drop in the cylcone eye decreases from 5300.0 [Pa] to 5250.0 [Pa].

C.13.4 Meteo on a netCDF file

This item will be added in near future

Remarks:

◇

◇

C.14 Statistical output (Fourier, minimum, maximum and average)

File contents	All parameters required to execute an online statistical analysis for specified quantities, a specified period and for specified frequencies. This analysis can be a Fourier analysis or another form of statistical analysis e.g. computation of the minimum, maximum or average value.
Filetype	ASCII
File format	Fix formatted for text variables, free formatted for real and integer values.
Filename	<*name*.fou>
Generated	Manually offline

Each line in the file represents a record.

Record description:

Record	Record description
each line	1 Id of the quantity on which the analysis is to be performed: bs bed stress (1D/2D field) cn n-th constituent in the MDU-file cs salinity, will be skipped if temperature is not modelled ct temperature, will be skipped if temperature is not modelled eh energy height (1D/2D field) fb freeboard (1D only) q1 discharge through a flow link sul water levels at a flow links (1D/2D field) ux velocity in x-direction uxa velocity in x-direction, column average (1D/2D field) uy velocity in y-direction uya velocity in y-direction, column average (1D/2D field) uc velocity magnitude vog volume on ground (1D only) wd water depth (1D/2D field) wdog waterdepth on ground (1D only) wl water level (1D/2D field) ws wind magnitude (1D/2D field) 2 Analysis start time, `tstart`, in Tunit [seconds, minutes, hours] after the simulation Reference Date. Use -1 to indicate the simulation start time. 3 Analysis stop time, `tstop`, in Tunit [seconds, minutes, hours] after the simulation Reference Date. Use -1 to indicate the simulation stop time. 4 Integer number of cycles (i.e. mode) within the analysis time frame. Alternatively, the length of the running mean filter. 5 Nodal amplification factor. 6 Phase shift [degrees]. 7 Layer number for the analysis of 3D quantities. This number should if and only if the selected quantity is not marked as a 1D or 1D/2D field. When specified, it should be a valid layer number. However, this value is currently not used. In 3D models, the analysis is done on the whole 3D field, ignoring the layer number. 8 Flag specifying the type of the analysis: ◇ default (no keyword): perform Fourier analysis ◇ `min/max/avg`, requests the computation of the minimal, maximal or average of the selected quantity over the selected period. The number of cycles now is the length of an optional running mean filter. An additional quantity id can be supplied to report the requested quantity when the `min/max` of that additional quantity occurs (see last record of Example 2, extensions for min/max). ◇ `last`, requests the last value, e.g. when anticipating convergence to a steady-state solution. The number of cycles now is the number of time steps before the end of the analysis period to average. ◇ `time below` / `time above` followed by threshold value, requests the computation of the total time passed during which the selected quantity strictly below or above the specified threshold.

Remarks:

- ◇ The analysis is performed for all grid points individually.
- ◇ If the number of cycles is set equal to 0, the *mean* value of the quantity over the interval specified by the start and stop time is determined.
- ◇ The computed Fourier amplitudes differ slightly from the amplitude of the corresponding tidal component. When comparisons with co-tidal maps have to be made, this factor can be deter-

mined using the subsystem ASCON of Delft3D-TIDE, the tidal analysis package of Deltares.

- The computed Fourier phases are by default associated with the reference date/time of the FLOW simulation. For comparisons with co-tidal maps the user may specify a non-zero phase shift.
- For computational cells which are dry during all Fourier cycles, no values are associated, and hence they contain a missing value.
- The layer number should not be specified for 1D or 1D/2D quantities such as water levels. In 3D models, the analysis is done on the whole 3D field, ignoring the layer number.
- In case of `max` and `min`, this keyword maybe followed by the keyword `time` to indicate that also the time at which the maximal or minimal value occurs, should be written to file.
- In case of `max` computation for water level, water depth or energy height, an additional keyword `inidryonly` can be specified to perform the analysis only for points that are initially dry.
- Depending on the flag `FouUpdateStep` the analysis is done every user time step (`FouUpdateStep = 0`, default), or every computation time step (`FouUpdateStep = 1`), or with the same time step as the history output (`FouUpdateStep = 2`). The keyword `FouUpdateStep` is defined in the chapter `[output]` of the mdu-file.

Restrictions:

- If `FouUpdateStep = 0 or 2`, the start and stop times of the analysis frame specified must be a multiple of the user time step or the history output time step.
- The start and stop times of the analysis frame specified must be a valid time, i.e. must fit in the simulation time frame of the FLOW computation.
- Items in a record must be separated by one or more blanks.
- The quantity name (first item on a line) must start on position one.
- The starting time of the simulation is *excluded* from the analysis.

Running mean

The minimum and maximum values may be filtered with a running mean filter, to minimize the effect of spikes. Column 4 is used for this and is the length of the filter in fourier output time steps. Therefore, this option can only be used if `FouUpdateStep = 0` or `2`.

The running mean filter is defined by:
Suppose we have a time serie $x[1], x[2], ...x[N]$.
Then the running mean is:
$\overline{x}[i] = (x[i+1-L] + x[i+2-L] + .. + x[i])/L$, with L the length of the filter and $i \geq L \wedge L > 0$.

The minimum of the running mean is defined as: $\min \overline{x}[i], i \in [\text{tstart}, \text{tstop}]$, the same holds for other functions.

The running mean filter is applied for all points in the model. Note that we need a buffer of length L times the number of grid points to calculate this. This buffer is needed for each quantity in the fou-file using the running mean, except if you ask both the $\min$ and $\max$ for the same field with the same start and stop time. Then the same buffer is used.

It is possible to get the velocity at the moment that we have the highest water level, or the time of the maximum water level (see the last two records of Example 2, extensions for min/max). Or just the latest time(s), in case you are running to a steady state solution (check out the `last` option).

Example 1

A statistical analysis is requested for:

- Water level: mean value and the first two harmonics.
- Velocity in x-direction: first harmonic, in the top layer.
- Velocity in y-direction: first harmonic, in the top layer.
- Velocity magnitude: first harmonic, in the top layer.
- Temperature: first harmonic, in the third layer.
- Salinity: mean value of the third layer.
- Four constituents: mean value for a slightly shifted time period in the top layer for 3 constituents; and the maximum for the fourth constituent.

```
wl     720.    1440.     2    1.000   0.000
wl     720.    1440.     1    1.000   0.000
wl     720     1440.     0    1.000   0.000
ux     720.    1440.     1    1.000   0.000      1
uy     720.    1440.     1    1.000   0.000      1
uc     720.    1440.     1    1.000   0.000      1
ct     720.    1440.     1    1.000   0.000      3
cs     720.    1440.     0    1.000   0.000      3
c1     710.    1430.     0    1.000   0.000      1
c2     710.    1430.     0    1.000   0.000      1
c3     710.    1430.     0    1.000   0.000      1
c4     710.    1430.     0    1.000   0.000      1    max
```

Remark:

- In this example the start and stop times are in minutes because the Tunit of this model (as defined in the MDU file) is in minutes.

Example 2, extensions for min/max

A statistical analysis is requested for:

- Water level: maximum with a filter with length 25, starting t = 720.
- Water level: minimum with a filter with length 13, stopping at t = 1440.
- Water level: average for the last 5 steps of the simulation.
- Water level: maximum with a filter with length 25, with extra output of the time at which this maximum occurs.
- Velocity magnitude as the water level reaches this maximum, starting at t = 720 and stopping at t = 1440.

```
wl     720.    -1         25   1.000   0.000      max
wl     -1       1440.     13   1.000   0.000      min
wl     -1       -1         5   1.000   0.000      last
wl     -1       -1        25   1.000   0.000      max time
uc     720.    1440.       1   1.000   0.000      1 max wl
```

Remarks:

- In this example the start and stop times are in minutes because the Tunit of this model is in minutes. A start time of -1 means start of the simulation; A stop time of -1 means the end of the simulation.
- In this example the columns 5 and 6 are completely ignored.
- Quantities that give combined results, must have the same dimension. Therefore we introduced sul (water level at flow links) to compute q1 (discharge through flow link) at the time of the maximum water level. Note that this combination does not work for 3D applications.

C.15 1D roughness INI file

The roughness settings for the 1D network may be specified directly in the cross section definitions (see section C.16.1) or by reference to roughness variables defined here. The keyword `FrictFile` should be used to point to one or more roughness files as defined below (see Appendix A). Each file defines one roughness variable. The following roughness types have been implemented

◇ `Chezy`: Chézy C [m$^{1/2}$/s]
◇ `Manning`: Manning n [s/m$^{1/3}$]
◇ `wallLawNikuradse`: Nikuradse k_n [m]
◇ `WhiteColebrook`: Nikuradse k_n [m]
◇ `StricklerNikuradse` = Nikuradse k_n [m]
◇ `Strickler`: Strickler k_s [m$^{1/3}$/s]
◇ `deBosBijkerk`: De Bos-Bijkerk γ [1/s]

See section 8.4.2.1 for details on these formulations.

The file description is given below.

Table C.21: *Roughness definition.*

Keyword	Type	Default	Description
[General]			
fileVersion	String	3.01	File version. Do not edit this.
fileType	String	roughness	File type. Do not edit this.
frictionValuesFile	String		Name of <*.bc> file containing the time-series with friction values. Only needed for functionType = timeSeries.
[Global]			*One global block per file (see Remarks for an exception)*
frictionId	String		Name of the roughness variable
frictionType	String		The global roughness type for this variable which is used if no branch specific roughness definition is given. See the table at the beginning of this section for the encoding of the different roughness types.
frictionValue	Double		The global default value for this roughness variable
[Branch]			*Optional: one block per branch*
branchId	String		The name of the branch
frictionType	String		The roughness type to be used on this branch. See the table at the beginning of this section for the encoding of the different roughness types.
functionType	String	constant	Function type for the calculation of the value, possible values ◇ constant: Constant (in time) ◇ timeSeries: Time series (see remark below) ◇ absDischarge: Function of absolute discharge (not dependent on direction) ◇ waterLevel: Function of water level

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
timeSeriesId	String		Refers to a data block in the <*.bc> frictionValuesFile. Only if functionType = timeSeries.
numLevels	Int		Number of levels in table. Only if functionType is not constant.
levels	Double[]		Discharge (m^3/s) or water level [m AD] values. Only if functionType is not equal to constant.
numLocations	Int	0	Number of locations on branch. The default 0 value implies branch uniform values.
chainage	Double[]		Space separated list of locations on the branch [m]. Locations sorted by increasing chainage. The keyword must be specified if numLocations>0.
frictionValues	Double[][]		numLevels lines containing space separated lists of roughness values: numLocations values per line. If the functionType is equal to constant, then a single line is required. For a uniform roughness per branch (numLocations = 0) a single entry per line is required. The meaning of the values depends on the roughness type selected (see frictionType).

Remarks:

- ◇ If the roughness varies along a branch, the roughness values are interpolated *linearly* along the branch in between roughness locations and assumed constant beyond the first and last location specified. If the roughness varies as a function of discharge or water level, the roughness values are interpolated *linearly* between the specified values and assumed constant below the first and above the last value.
- ◇ Multiple roughness variable definitions that don't vary spatially (i.e. that don't require Branch blocks) may be combined in one file. In that case there will one General block, multiple instances of the Global block and no Branch blocks in the file. See the second example.
- ◇ The function of discharge has been implemented as a function of absolute discharge.
- ◇ The frictionId may contain spaces, with no quoting needed.
- ◇ The branchId may contain spaces, with no quoting needed.
- ◇ This file does not include an option to specify roughness locations via x and y.
- ◇ The number of frictionValues to be specified per line equals the number of locations on that branch. If numLocations is not specified or equal to 0, one value is expected which will be applied uniformly along the branch.
- ◇ The number of data lines following frictionValues is one for functionType = constant and equal to numLevels for the other function types.
- ◇ When at least one [Branch] block has frictionType = timeSeries, then:
  - ◇ the [General] block must specify a <*.bc> file via frictionValuesFile. The <*.bc> may contain multiple time series (in [Forcing] blocks);
  - ◇ the branch keyword timeSeriesId must refer to a timeseries block in that file (i.e., must match with the name);
  - ◇ the quantity in that file must equal friction_coefficient_*<friction type>*, where *<friction type>* is the type as given after frictionType, e.g., quantity = friction_coefficient_Manning.
- ◇ A branch that has functionType = timeSeries can only use one time series. In other words: the friction coefficient is constant along the branch.

Example:

```
[General]
    fileVersion           = 3.01
    fileType              = roughness

[Global]
    frictionId            = Main
    frictionType          = Chezy
    frictionValue         = 45.000

[Branch]
    branchId              = Channel1
    frictionType          = Manning
    functionType          = constant
    numLocations          = 2                       # at two locations
    chainage              = 0.000 100.000
    frictionValues        = 0.20000 0.30000

[Branch]
    branchId              = Channel4
    frictionType          = Chezy
    functionType          = constant
    frictionValues        = 40.00000

[Branch]
    branchId              = Channel7
    frictionType          = Chezy
    functionType          = absDischarge
    numLevels             = 3                       # for three values
    levels                = 100.000 200.000 300.000
    numLocations          = 2                       # at two locations
    chainage              = 0.000 300.000
    frictionValues        = 45.00000 55.00000      # values for |Q|=100
                            41.00000 52.00000      # values for |Q|=200
                            40.00000 50.00000      # values for |Q|=300
```

Example Multiple Constant Roughness Variables:

```
[General]
    fileVersion           = 3.01
    fileType              = roughness

[Global]
    frictionId            = Brick
    frictionType          = WhiteColebrook
    frictionValue         = 0.1

[Global]
    frictionId            = Steel
    frictionType          = WhiteColebrook
    frictionValue         = 0.005
```

Example time-dependent roughness:

```
[General]
    fileVersion           = 3.01
    fileType              = roughness
    frictionValuesFile    = roughness_timeseries.bc

[Global]
    frictionId            = Main
    frictionType          = Chezy
    frictionValue         = 45.000

[Branch]
    branchId              = Channel3
    frictionType          = deBosBijkerk
    functionType           = timeSeries
    timeSeriesId           = rural_secondary

[Branch]
    branchId               = Channel9
    frictionType           = deBosBijkerk
    functionType           = timeSeries
    timeSeriesId           = rural_secondary
```

The accompanying <roughness_timeseries.bc>:

```
[General]
    fileVersion            = 1.01
    fileType               = boundConds

[Forcing]
    name                   = rural_secondary
    function               = timeSeries
    timeInterpolation      = linear
    quantity               = time
    unit                   = days since 2001-01-01
    quantity               = friction_coefficient_deBosBijkerk
    unit                   = s-1
    0  15
    31 21
    59 28
    90 25
    # ...
```

C.16 Cross section files

Cross sections are defined in two files. The first file (CrossDefFile) contains the cross section definitions with the geometrical parameters and the main roughness settings. This file is described in section C.16.1. The second file (CrossLocFile) defines where in the 1D network the cross sections are actually used, and it also specifies the vertical position of the cross sections at those locations. This file is described in section C.16.2. The cross section location file is not used for XYZ cross sections since the XYZ definition already includes a full specification of the location. The keyword CrossDefFile should be used to point to the cross section definition file and the keyword CrossLocFile should be used to point to the cross section location file (see Appendix A).

C.16.1 Cross section definition file

The definition file contains the geometrical parameters for each cross section. It defines the shape and roughness of the cross section. The shape may be defined either by a parametrized shape or a more flexible table (containing either ZW, YZ, or XYZ data). The roughness settings may be specified directly (by means of type and value) or by reference to roughness variables. If no roughness setting is specified, the roughness is in most cases derived from the roughness variable "Main" (see the subsections below for exceptions). If both frictionId references are given, and also frictionType+frictionValue are given in the cross section definition, then the latter two are ignored: roughess variables (roughness sections) take precedence over direct values. The input for roughness variables is described in section C.15.

Table C.22: *Cross section definition file.*

Keyword	Type	Default	Description
[General]			
fileVersion	String	3.00	File version. Do not edit this.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
`fileType`		`crossDef`	File type. Do not edit this.
`[Global]`			
`leveeTransitionHeight`	Double	`0.50`	Levee transition height [m]. Only applicable in case of cross section type `zwRiver`.
`[Definition]`			
`id`	String		Unique cross-section definition id.
`type`	String		Cross section type. Possible values are: ◇ `circle`: circular profile ◇ `rectangle`: rectangular profile ◇ `zwRiver`: tabulated river profile ◇ `zw`: ZW profile ◇ `xyz`: XYZ profile ◇ `yz`: YZ profile For specific profile shapes such as arch, egg, mouth-shape, and trapezium see the description of the tabulated ZW profiles.
`thalweg`	Double		Transverse Y coordinate at which the cross section aligns with the branch **(Keyword used by GUI only)**
`...`			Keywords specific for the cross section type

The type specific keywords and descriptions can be found in the following subsections.

Remark:
- ◇ The `id` may contain spaces, with no quoting needed.

Example:

```
[General]
    fileVersion           = 3.00
    fileType              = crossDef

[Definition]
    id                    = Prof1
    type                  = circle
    thalweg               = 0.0
    diameter              = 2.0
    frictionId            = Brick

[Definition]
    id                    = Prof1A
    type                  = circle
    thalweg               = 0.0
    diameter              = 2.0
    frictionType          = WhiteColebrook
    frictionValue         = 0.1

[Definition]
    id                    = Prof2
    type                  = zw
    thalweg               = 0.0
    template              = steelMouth           # template name and
    height                = 0.950                # values used by GUI
    r                     = 0.600
    r1                    = 0.500
    r2                    = 0.000
    r3                    = 0.000
    a                     = 0.000
    a1                    = 0.000
    numLevels             = 43                   # zw levels used by kernel
    levels                = 0.00000 0.00873 0.01746 0.02619 0.03492 0.04365
    0.05238 0.06111 0.06984 0.07857 0.08730 0.09603 0.10476 0.11349 0.12222
    0.13095 0.13968 0.14841 0.15714 0.16587 0.17460 0.18333 0.19206 0.20079
    0.20952 0.21825 0.22698 0.23571 0.24444 0.25317 0.26190 0.27063 0.27937
    0.28810 0.29683 0.30556 0.31429 0.32302 0.33175 0.34048 0.34921 0.35794
    0.36667
    flowWidths            = 0.00000 0.18605 0.26196 0.31940 0.36716 0.40863
    0.44559 0.47907 0.50976 0.53814 0.56455 0.58927 0.61249 0.63439 0.65508
    0.67470 0.69331 0.71102 0.72787 0.74393 0.75925 0.77388 0.78785 0.80119
    0.80964 0.79011 0.76970 0.74832 0.72589 0.70231 0.67746 0.65120 0.62335
    0.59367 0.56190 0.52763 0.49036 0.44934 0.40341 0.35067 0.28738 0.20396
    0.00000
    frictionId            = Steel

[Definition]
    id                    = xyzProf1
    type                  = xyz
    thalweg               = 283.746
    xyzCount              = 6
    xCoordinates          = 1920.254 1925.508 1925.508 1925.508 1930.761
    1930.761
    yCoordinates          = 1452.223 1168.526 1026.677 953.126 900.589
    884.828
    zCoordinates          = 10.000 0.004 4.994 7.585 9.445 10.000
    conveyance            = segmented
    sectionCount          = 3
    frictionPositions     = 0.000 158.032 344.304 567.706
    frictionIds           = LeftBank; Main; RightBank
```

C.16.1.1 Circle cross section definition

This section describes the additional keywords for the circular shaped cross sections in the **Definition** block compared to the general description in section C.16.1. This cross section type is always closed.

Table C.23: *Circle cross section definition.*

Keyword	Type	Default	Description
[Definition]			
type	String	circle	Cross section type; must read circle.
diameter	double		Internal diameter of the circle [m].
frictionId	String		Name of the roughness variable associated with this cross section. Either this parameter or frictionType should be specified. If neither parameter is specified, the frictionId defaults to "Main".
frictionType	String		Roughness type associated with this cross section (see section C.15 for encoding). Either this parameter or frictionId should be specified.
frictionValue	Double		Roughness value; its meaning depends on the roughness type selected (only used if frictionType specified).

A ground layer will be included in the future; keywords not yet defined.

C.16.1.2 Rectangle cross section definition

This section describes the additional keywords for the rectangular cross sections in the **Definition** block compared to the general description in section C.16.1. A typical rectangular cross section is shown in the figure on the right. The rectangular profile may be used for both open and closed channels. Use the keyword closed to switch between these two cases.

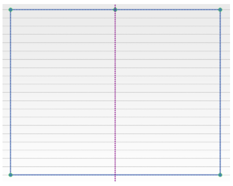


Table C.24: *Rectangular cross section definition.*

Keyword	Type	Default	Description
[Definition]			
type	String	rectangle	Cross section type; must read rectangle.
width	Double		Width of the rectangle [m].
height	Double		Height of the rectangle [m].
closed	Logical	yes	no: Open channel, yes: Closed channel
frictionId	String		Name of the roughness variable associated with this cross section. Either this parameter or frictionType should be specified. If neither parameter is specified, the frictionId defaults to "Main".
frictionType	String		Roughness type associated with this cross section (see section C.15 for encoding). Either this parameter or frictionId should be specified.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
frictionValue	Double		Roughness value; its meaning depends on the roughness type selected (only used if frictionType specified).

A ground layer will be included in the future; keywords not yet defined.

C.16.1.3 Tabulated river cross section definition

This section describes the additional keywords for the tabulated river cross sections in the **Definition** block compared to the general description in section C.16.1. The tabulated river cross section is very similar to the ZW cross section (see section C.16.1.4). Compared to the standard ZW cross section, the tabulated river profile will always be open and the width tables should be monotonically increasing. Furthermore, the roughness over the flow width may be defined separately for three sections: main, floodplain 1 and floodplain 2. This cross section type is always processed using the lumped conveyance type per section.

Table C.25: *Tabulated river cross section definition.*

Keyword	Type	Default	Description
[Definition]			
type	String	zwRiver	Cross section type; must read zwRiver.
numLevels	Int		Number of levels in the table.
levels	Double[]		Space separated list of monotonic increasing heights/levels [m AD].
flowWidths	Double[]		Space separated list of flow widths at the selected heights [m)].
totalWidths	Double[]		Space separated list of total widths at the selected heights [m]. Equal to flowWidths if not specified. If specified, the totalWidths should be larger than flowWidths.
leveeCrestLevel			Relative crest level of levee [m].
leveeBaseLevel			Relative base level of levee [m].
leveeFlowArea			Flow area behind levee [m^2].
leveeTotalArea			Total area behind levee [m^2].
mainWidth	Double	*	Width of main section [m]. Default value: max(flowWidths).
fp1Width	Double	*	Width of floodplain 1 section [m]. Default value: max(flowWidths)-mainWidth
fp2Width	Double	*	Width of floodplain 2 section [m]. Default value: max(flowWidths)-mainWidth-fp1Width
frictionIds	String[]		Semicolon separated list of roughness variable names associated with the roughness sections. Either this parameter or frictionTypes should be specified. If neither parameter is specified, the frictionIds default to "Main", "FloodPlain1" and "FloodPlain2".
			(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
frictionTypes	String[]		Semicolon separated list of roughness types associated with the roughness sections (see section C.15 for encoding). Either this parameter or frictionIds should be specified. Can be specified as a single value if all roughness sections use the same type.
frictionValues	Double[]		Space separated list of roughness values; their meaning depends on the roughness types selected (only used if frictionTypes specified).

Remarks:

- ◇ The optional levee feature was called “summer dike” in SOBEK.
- ◇ The parameters leveeCrestLevel and leveeBaselevel are relative levels with respect to the cross section location.

C.16.1.4 ZW (tabulated) cross section definition

This section describes the additional keywords for the ZW cross sections in the **Definition** block compared to the general description in section C.16.1. The ZW profile may be used for both open and closed channels, so this type may be open or closed. Use a zero total width at the last (highest) level to define a closed cross section. This profile type allows you to distinguish between flow area and storage area; the difference between the total width and the flow width is the storage width. This cross section type is always processed using the lumped conveyance type.

Table C.26: *ZW cross section definition.*

Keyword	Type	Default	Description
[Definition]			
type	String	zw	Cross section type; must read zw.
template	String		(Optional) Name of ZW cross section template; see description below. **(Keyword used by GUI only)** Possible values are: ◇ arch: arch profile ◇ mouth: mouth-shaped profile ◇ egg: egg profile ◇ ellipse: ellipse profile ◇ invEgg: inverted egg profile ◇ steelMouth: steel mouth-shaped profile ◇ trapezium: trapezium profile ◇ uShape: U-shaped profile
...			Keywords specific for the template.
numLevels	Int		Number of levels in the table.
levels	Double[]		Space separated list of monotonic increasing heights/levels [m AD].
flowWidths	Double[]		Space separated list of flow widths at the selected heights [m].
totalWidths	Double[]		Space separated list of total widths at the selected heights [m].
			(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
frictionId	String		Name of the roughness variable associated with this cross section. Either this parameter or frictionType should be specified. If neither parameter is specified, the frictionId defaults to "Main".
frictionType	String		Roughness type associated with this cross section (see section C.15 for encoding). Either this parameter or frictionId should be specified.
frictionValue	Double		Roughness value; its meaning depends on the roughness type selected (only used if frictionType specified).

A ground layer will be included in the future; keywords not yet defined.

Many pipes and channels follow standardized design procedures and as a result these pipes and channels have a standardized shape. For the simulation kernel these shapes are represented by a ZW profile description. To support the user in creating these profiles, the user interface supports templates for common cross section geometries. In these cases, the keyword template is used to specify the templated profile being used and additional keywords are included.

Remarks:

- ◇ If you manually modify this file, please note that the user interface and simulation kernel process this cross section type differently if the template is included. So, do not include that keyword ... or make sure that the information of the template is consistent with the information provided by the numLevels, levels, flowWidths and totalWidths keywords.
- ◇ When importing a cross section definition containing the template keyword, the user interface will not process the keywords numLevels, levels, flowWidths and totalWidths.
- ◇ When the template is empty, the user interface will treat the ZW profile record as a generic ZW cross section and thus it will process the keywords numLevels, levels, flowWidths and totalWidths.
- ◇ The simulation engine will ignore template and associated keywords, and always read the keywords numLevels, levels, flowWidths and totalWidths.

The following tables give an overview of the various templates recognized and the associated keywords **(All used by GUI only)**.

Arch profile template

Additional definition items for the arch profile template. A typical arch profile is shown in the figure on the right. The height is the total height of the profile. The arcHeight is the height of the curved arch. The height of the straight base is consequently height − arcHeight. The arcHeight must thus be less or equal height.

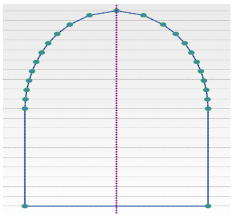


Table C.27: *Arch profile definition.*

Keyword	Type	Default	Description
[Definition]			
template	String	arch	Name of ZW cross section template
width	Double		Width of the cross section [m].
			(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
height	Double		Total height of cross section [m].
arcHeight	Double		Height of arch [m].

Mouth-shaped profile template

Additional definition items for the mouth-shaped profile template. A typical mouth-shaped profile is shown in the figure on the right. This profile was called "Cunette" in SOBEK.

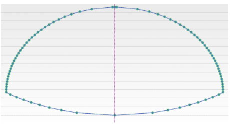


Table C.28: *Mouth-shaped profile definition.*

Keyword	Type	Default	Description
[Definition]			
template	String	mouth	Name of ZW cross section template
width	Double		Width of the mouth profile [m].
height	Double	*	Height of mouth profile [m]. The default height is 0.634 * width

Egg profile template

Additional definition items for the egg profile template. A typical egg profile is shown in the figure on the right. The egg profile has the widest end at the top.

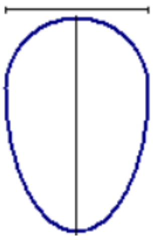


Table C.29: *Egg profile definition.*

Keyword	Type	Default	Description
[Definition]			
template	String	egg	Name of ZW cross section template
width	Double		Width of the egg profile [m].
height	Double	*	Height of the egg profile [m]. The default height is 1.5 * width

Ellipse profile template

Additional items for the ellipse profile template. A typical ellipse profile is shown in the figure on the right.

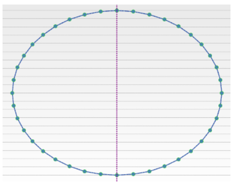


Table C.30: *Ellipse profile definition.*

Keyword	Type	Default	Description
`[Definition]`			
`template`	String	`ellipse`	Name of ZW cross section template
`width`	Double		Width of the ellipse profile [m].
`height`	Double		Height of the ellipse profile [m].

Inverted egg profile template

Additional definition items for the inverted egg profile template. A typical inverted egg profile is shown in the figure on the right. The inverted egg profile has the widest end at the base.

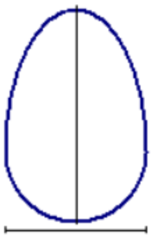


Table C.31: *Inverted egg profile definition.*

Keyword	Type	Default	Description
`[Definition]`			
`template`	String	`invEgg`	Name of ZW cross section template
`width`	Double		Width of the egg profile [m].
`height`	Double	*	Height of the egg profile [m]. The default height is 1.5 * `width`.

Steel mouth-shaped profile template

Additional definition items for the steel mouth-shaped profile template. A typical steel mouth-shaped profile is shown in the figure on the right. This profile was called "SteelCunette" in SOBEK.

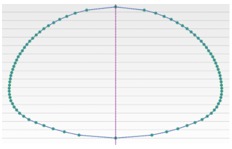


Table C.32: *Steel mouth-shaped profile definition.*

Keyword	Type	Default	Description
`[Definition]`			
`template`	String	`steelMouth`	Name of ZW cross section template
`height`	Double		Height of steel mouth profile [m].
`r`	Double		Radius r [m].
`r1`	Double		Radius r1 [m].
`r2`	Double		Radius r2 [m].
`r3`	Double		Radius r3 [m].
`a`	Double		Angle a [degrees].
`a1`	Double		Angle a1 [degrees].

Trapezium profile template

Additional items for the trapezium profile template. A typical trapezium profile is shown in the figure on the right.

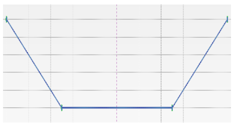


Table C.33: Trapezium profile definition.

Keyword	Type	Default	Description
[Definition]			
template	String	trapezium	Name of ZW cross section template
slope	Double		Slope of trapezium: dZ per 1 m of width [m].
width	Double		Maximum flow width of trapezium [m].
baseWidth	Double	0	Base width of trapezium [m].
closed	Logical	no	no: Open channel, yes: Closed channel

U-shaped profile template

Additional definition items for the U-shaped profile template. A typical U-shaped profile is shown in the figure on the right. The height is the total height of the profile. The arcHeight is the height of the curved arch. The height of the straight top is consequently height − arcHeight. The arcHeight must thus be less or equal height.

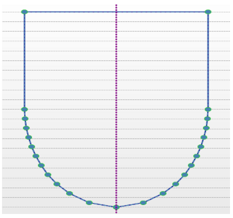


Table C.34: U-shaped profile definition.

Keyword	Type	Default	Description
[Definition]			
template	String	uShape	Name of ZW cross section template
width	Double		Width of the cross section [m].
height	Double		Height of cross section [m].
arcHeight	Double		Height of U-arch [m].

C.16.1.5 XYZ cross section definition

This section describes the additional keywords for the XYZ cross sections in the **Definition** block compared to the general description in section C.16.1. The distance along the XY-line will be converted to a transverse coordinate Y, and subsequently this cross section type will be treated as a YZ-cross section. XYZ cross sections are always open. The cross section may be processed using lumped or vertically segmented conveyance method; please note that this choice influences the available options for the roughness definition.

Table C.35: XYZ cross section definition.

Keyword	Type	Default	Description
[Definition]			
type	String	xyz	Cross section type; must read xyz.
branchId	String		(optional) Branch on which the cross section is located.
xyzCount	Int		Number of XYZ-coordinates

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
xCoordinates	Double[]		X-coordinates [m or degrees East].
yCoordinates	Double[]		Y-coordinates [m or degrees North].
zCoordinates	Double[]		Z-coordinates [m AD].
conveyance	String	lumped	lumped: Lumped, segmented: Vertically segmented. In the case of lumped conveyance, only a single uniform roughness for the whole cross section is allowed, i.e., sectionCount must equal 1.
sectionCount	Int	1	Number of roughness sections. If the lumped conveyance is selected then sectionCount must equal 1.
frictionPositions	Double[]		Locations where the roughness sections start and end. Always one location more than sectionCount. The first value should equal 0 and the last value should equal the cross section length. Keyword may be skipped if sectionCount = 1.
frictionIds	String[]		Semicolon separated list of roughness variable names associated with the roughness sections. Either this parameter or frictionTypes should be specified. If sectionCount = 1 and neither parameter is specified, the frictionIds defaults to "Main".
frictionTypes	String[]		Semicolon separated list of roughness types associated with the roughness sections (see section C.15 for encoding). Either this parameter or frictionIds should be specified. Can be specified as a single value if all roughness sections use the same type.
frictionValues	Double[]		Space separated list of roughness values; their meaning depends on the roughness types selected (only used if frictionTypes specified).

Remark:

◇ The branchId keyword is only needed if there is a possibility for confusion, i.e. an XYZ definition that intersects multiple branches. The XYZ cross section is always positioned at (the chainage given by) the intersection of the XYZ cross section and the polyline of the branch.

C.16.1.6 YZ cross section definition

Additional items for the YZ cross section definition. YZ profiles may be used represent an open channel (Z as a single-valued function of monotonic increasing Y) or a closed pipe or semi-closed channel of arbitrary cross sectional shape (Z is not a single valued function of Y, YZ is not self-intersecting); the keyword singleValuedZ is used to distinguish between these two cases, but only the first option is currently implemented. A cross section of the second type is considered closed if the first and last YZ coordinate pair are equal. YZ cross sections for open channels (singleValuedZ = yes) may be processed using lumped or vertically segmented conveyance method; please note that this choice influences the available options for the roughness definition. If singleValuedZ = no then only the lumped conveyance and a single uniform roughness is allowed.

Table C.36: YZ cross section definition.

Keyword	Type	Default	Description
[Definition]			
type	String	yz	Cross section type; must read yz.
singleValuedZ	Logical	yes	no: Not self-intersecting YZ curve, yes: Monotonic increasing Y, single valued Z(Y) (implemented as any value <>0). **Note:** only the second option is currently available.
yzCount	Int		Number of YZ-coordinates
yCoordinates	Double[]		Y-coordinates [m].
zCoordinates	Double[]		Z-coordinates [m AD].
conveyance	String	segmented	lumped: Lumped, segmented: Vertically segmented. Only the default lumped option is allowed if singleValuedZ = no. In the case of lumped conveyance, only a single uniform roughness for the whole cross section is allowed, i.e., sectionCount must equal 1.
sectionCount	Int	1	Number of roughness sections. If the lumped conveyance is selected then sectionCount must equal 1.
frictionPositions	Double[]		Locations where the roughness sections start and end. Always one location more than sectionCount. The first value should equal 0 and the last value should equal the cross section length. Keyword may be skipped if sectionCount = 1
frictionIds	String[]		Semicolon separated list of roughness variable names associated with the roughness sections. Either this parameter or frictionTypes should be specified. If sectionCount = 1 and neither parameter is specified, the frictionIds defaults to "Main".
frictionTypes	String[]		Semicolon separated list of roughness types associated with the roughness sections (see section C.15 for encoding). Either this parameter or frictionIds should be specified. Can be specified as a single value if all roughness sections use the same type.
frictionValues	Double[]		Space separated list of roughness values; their meaning depends on the roughness types selected (only used if frictionTypes specified).

C.16.2 Cross section location file

The location file contains a listing of all the locations where cross sections are used in the 1D network. For every location included, it points to the cross section definition used at that particular location. The file does not include any records for locations where XYZ cross sections are used; all the information for these cross sections is already included in their definition. Consequently, if a model contains only XYZ cross sections, this file doesn't have to be specified.

Table C.37: *Cross section location.*

Keyword	Type	Default	Description
[General]			
fileVersion	String	1.01	File version. Do not edit this.
fileType		crossLoc	File type. Do not edit this.
[CrossSection]			
id	String		Unique cross-section location id
branchId	String		(optional) Branch on which the cross section is located.
chainage	Double		(optional) Location on the branch [m].
x	Double		(optional) x-coordinate of the location of the cross section.
y	Double		(optional) y-coordinate of the location of the cross section.
shift	Double	0.0	Vertical shift of the cross section definition [m]. Defined positive upwards.
definitionId	String		Id of cross section definition

Remarks:

- The id, branchId and definitionId may contain spaces, with no quoting needed.
- Two types of location specifications are available.
  - When branchId is given, the keywords branchId and chainage will be used for the location specification.
  - When branchId is *not* given, the keywords x and y coordinates will be used.
  - The value for shift is relative in meters with respect to the absolute bed level of the cross section definition (for example as in yz). When the associated cross section definition is of a type that has no absolute bed level (for example circle), then its absolute bed level defaults to 0.0, which means that the resulting actual bed level becomes exactly equal to the shift.

Example:

```
[General]
    fileVersion           = 1.01
    fileType              = crossLoc

[CrossSection]
    id                    = Channel1_50.000
    branchId              = Channel1
    chainage              = 50.000
    shift                 = 1.0
    definitionId          = Prof1

[CrossSection]
    id                    = Channel2_300.000
    branchId              = Channel2
    chainage              = 300.000
    shift                 = 0.0
    definitionId          = Prof2
```

C.17 Storage node file

By default the storage area of a 1D node is equal to half the sum of the surface areas of the connected flow links (i.e. cross sectional width times branch length up to the neighbouring calculation point). The storage node file can be used to assign additional storage area to a calculation point, e.g. the area of a manhole (compartments), a small lake or a pond. The keyword `StorageNodeFile` should be used to point to the storage node file (see Appendix A).

Table C.38: *Definition of Storage Nodes.*

Keyword	Type	Default	Description
`[General]`			
`fileVersion`	String	`2.02`	Major file version. Do not edit this.
`fileType`	String	`storageNodes`	File type. Do not edit this.
`useStreetStorage`	Logical	`true`	Indicates whether above `streetLevel` the `streetStorageArea` must be used (`true`) or the regular storage `area` continues (`false`). This option is only applicable for storage nodes with `useTable = false`.
`[Global]`			*Loss coefficients in the Global section can be overridden by their counterparts in specific StorageNode sections.*
`angleCount`	Int		Number of values in angle lookup table.
`angles`	Double		Angles in lookup table, must be monotonically increasing between 0 and 180 degrees [deg].
`angleLossCoefficient`	Double		Angle loss coefficients corresponding to the angles in the lookup table.
`entranceLossCoefficient`	Double		Entrance loss coefficient.
`exitLossCoefficient`	Double		Exit loss coefficient.
`expansionLossCoefficient`	Double		Expansion loss coefficient.
`[StorageNode]`			
`id`	String		Unique id of the storage node.
`name`	String		Long name in the user interface.
`manholeId`	String		(optional) Unique id of manhole that this (compartment) node is part of.
`nodeType`	String	`unspecified`	(optional) Type of the node. Possible values are: ◇ `inspection`: inspection chamber ◇ `soakawayDrain`: soakaway drain (infiltration) ◇ `compartment`: manhole compartment ◇ `unspecified`: general storage node of unspecified type
`nodeId`	String		(optional) Connection node on which the storage node is located.
`branchId`	String		(optional) Branch on which the storage node is located.
`chainage`	Double		(optional) Location on the branch, distance from start of the branch [m].
`x`	Double		(optional) x-coordinate of the storage node.
`y`	Double		(optional) y-coordinate of the storage node.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
useTable	Logical		Switch to select a simple (false) or tabulated (true) storage area.
If useTable = false then specify:			
bedLevel	Double		Bed level of the storage area [m AD].
area	Double		Storage area from bedLevel up to streetLevel (and beyond if useStreetStorage = false) [m^2].
streetLevel	Double		Street level of the storage area [m AD].
streetStorageArea	Double		Storage area from streetLevel upwards if useStreetStorage = true [m^2].
storageType	String	reservoir	Possible values: ◇ reservoir: Above streetLevel the storage area of this node is also taken into account. ◇ closed: Above streetLevel this storage node has no storage area.
If useTable = true then specify:			
numLevels	Int		Number of levels in storage area table.
levels	Double []		Levels in storage area table [m AD].
storageArea	Double []		Areas in storage area table [m^2].
interpolate	String	linear	Interpolation type for storage area table. Possible values: linear or block.
To override Global loss coefficients for specific StorageNode:			
angleCount	Int		Number of values in angle lookup table.
angles	Double		Angles in lookup table, must be monotonically increasing between 0 and 180 degrees [deg].
angleLossCoefficient	Double		Angle loss coefficients corresponding to the angles in the lookup table.
entranceLossCoefficient	Double		Entrance loss coefficient.
exitLossCoefficient	Double		Exit loss coefficient.
expansionLossCoefficient	Double		Expansion loss coefficient.

Remarks:

- ◇ Three types of location specifications are available.
  - ◇ When nodeId is given, the storage node will be placed on the computational grid point located on the connection node in the network with that nodeId (only for 1D).
  - ◇ When branchId is given, the keywords branchId and chainage will be used for the location specification (only for 1D).
  - ◇ When nodeId and branchId are *not* given and x and y are given, the storage node will be placed on the computational grid point closest to this (x,y) location.
- ◇ Either the nodeId or the combination branchId and chainage can be specified, but not both.

- ◇ The coordinates (x, y) will only be used when nodeId and branchId are omitted.
- ◇ To define a manhole storage node (in urban applications):
  - ◇ Use useTable = false, then the storage area will be constant between bedLevel and streetLevel, and constant above streetLevel. So, the storage area will not linearly vary between area and streetStorageArea.
  - ◇ If useStreetStorage = false, then area is the storage area from bedLevel upwards. Hence the streetStorageArea value will not be used.
  - ◇ Additionally, if useStreetStorage = false, and storageType = closed the storage area above streetLevel is set to a small nonzero slot area.
  - ◇ The streetLevel will be used as the local "ground level" for output in the map file (more details on page 572).
- ◇ To define a lumped water storage node (in rural applications): use useTable = true.
- ◇ The last levels value will be used as the local "ground level" for output in the map file (more details on page 572).
- ◇ The loss coefficients are only considered for storage nodes with nodeType = compartment.
- ◇ Overriding loss coefficients or inheriting global values is handled for each loss parameter separately. For example: one may inherit a global angle loss table, and yet override the bend loss coefficient.

C.18 1D-2D Link file

The 1D-2D links in a model can be configured for geometry and other parameters. Global parameters are available, but additionally, this can also be done for selected links. This configuration is done via the 1D2DLinkFile. This approach is somewhat comparable to placing cross sections on 1D network branches. Note that the 1D-2D links themselves (their location) is stored as mesh contacts in the NetFile (Section B.2.5.2).

Table C.39: *1D-2D link definitions.*

Keyword	Type	Default	Description
[General]			
fileVersion	String	1.00	File version. Do not edit this.
fileType		1D2DLink	File type. Do not edit this.
[MeshContactParams]			*Repeat as needed*
contactType	String	all	(optional) Only affect links of specified type. Only relevant when polygon xCoordinates and yCoordinates are also specified. Possible values are: ◇ streetInlet: links between 2D street and 1D node (gullies). ◇ ...: *Note: later, other link types might be supported here as well.* ◇ all: affect all enclosed 1D-2D links.
contactId	String		(optional) Directly select the link with specified Id.
numCoordinates	int		(optional) number of values in xCoordinates and yCoordinates. This value should be ≥ 3.
xCoordinates	Double[]		(optional) x-coordinates of the link selection polygon. (number of values = numCoordinates).
yCoordinates	Double[]		(optional) y-coordinates of the link selection polygon. (number of values = numCoordinates).
Parameters for contactType = streetInlet *(gullies)*			

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
openingWidth	Double		Opening width of street inlet (also used for pipe cross section definition).
openingHeight	Double		Opening height of street inlet (also used for pipe cross section definition).

Remark:

◇ Two types of location specifications are available. One and just one must be given.

  ◇ When contactId is given, it will be used to configure that identified mesh contact.
  ◇ When instead, the keywords numCoordinates, xCoordinates and yCoordinates are given, these will be used to define a polygon. All 1D-2D links inside that polygon (and matching with the specified contactType, if present) will be configured with the same specified parameters.

C.19 Cache file

A cache file may be used to speed up the initialization of a model significantly. It is automatically filled with the geometrical discretization of some spatial input of a model schematization. The name of this file is always equal to <*mdu_name*.cache> and located in the same directory as the mdu-file. It is a binary file.

The currently supported model features for caching are:

◇ Fixed weirs (see Section 14.1);
◇ Observation points (see Section 5.4.2.2).
◇ Observation cross-sections (see section 5.4.2.3).
◇ Dry points and dry areas (see Section 5.4.2.7)

In principle, caching can also be applied to other features like thin dams, weirs, gates, sources/sinks and embankments. However, in practice we observe that in general the initialization of these features isn't time consuming. Therefore, caching hasn't been implemented for these features. On the other hand, in particular for fixed weirs the initialization can be time consuming.

The behavior of caching is as follows:

1 Upon initialization of the model, when a cache file is present, it is checked on the following requirements:

  ◇ The cache version number in the file should match the one in the current program.
  ◇ The cached MD5 hash of the network file should match the one of the current NetFile.
  ◇ The cached count of observation points and their x- and y-coordinates should match the current set of observation points.
  ◇ The cached count of fixed weirs and their x- and y-polyline coordinates should match the current set of fixed weirs.

2 If any of these checks fail, the cache file is entirely ignored. In this case (and also when no cache file is present at all) the model input is completely initialized onto the model grid. The result is then saved in the same (or new) cache file, for a future model run.
3 If all checks succeeded, the feature discretization is skipped, and instead directly loaded from the cache file.

Caching can be enabled or disabled via the MDU setting UseCaching=1 or 0, respectively. The default or MDU setting for caching can be disabled from the commandline using the option --no-geom-cache.

The cache file is platform-independent and as such can be transferred to other hosts, along with the entire set of model input files.

The cache file should not be edited in any way.

The default value is `UseCaching=1`, which means that caching is applied automatically. Although there is no reason for this, it is possible to switch off caching via keyword `UseCaching=0`.

D Initial conditions and spatially varying input

D.1 Introduction

Spatially varying input is input data that varies in space, not in time, such as initial conditions (water levels, etc.), or spatially varying model parameters (friction coefficients, etc.). D-Flow FM uses input data in *model-independent coordinates*, that is, the input files should contain their own spatial coordinates, and do not correspond with the D-Flow FM grid cell numbering. As a result, the model grid can always be changed, without the need to change all other spatial input. The spatial input will be interpolated onto the active model grid. Spatial input has to be specified in the ext-file (section C.5). All supported quantities are listed in Table C.5 under “Initial fields” and “Spatial physical properties”.
Remark:

- **Spatial input has to be specified in the `IniFieldFile` (section D.2). Formerly, this was specified in the old-style ext-file, which is now only available for backwards compatibility.**

Initial conditions can also be set using a restart-file, which assigns the exact values in the file to the current model grid. No interpolation is performed, and the restart file should correspond exactly with the current model.

D.2 Initial and parameter fields

Initial and parameter fields are time-independent spatial fields that can describe initial conditions and spatially varying model parameters. The name of this file is specified using the keyword `IniFieldFile` in the MDU-file (see Appendix A). This file supersedes the old-style external forcings file for these quantities.

The fields are specified in the ini-file format. The order of blocks is relevant, since multiple data sets maybe combined for the same quantity (see `operand` in Table D.1). The order of keywords within a block is *not* relevant. Keywords and constants are case insensitive. String values (e.g., filenames) are case sensitive.

This file may contain one or more **Initial** blocks or **Parameter** blocks, each followed by a set of keywords that specify the spatial data for that field. See the following table for details.

Table D.1: *Initial and parameter fields.*

Keyword	Type	Default	Description
`[General]`			
`fileVersion`	String	`2.00`	File version. Do not edit this.
`fileType`	String	`iniField`	File type. Do not edit this.
`[Initial]` or `[Parameter]`			*Repeat as needed*
`quantity`	String		Name of the quantity, see Table D.2 hereafter.
`dataFile`	String		Name of file containing the field data values.
`dataFileType`	String		Type of `dataFile`. Supported types: ◇ `arcinfo` ◇ `GeoTIFF` (section D.4.4) ◇ `sample` (section C.3) ◇ `1dField` (section D.4.1), for this type the following keywords are not necessary. ◇ `polygon` (section C.2)

(continued on next page)

(continued from previous page)

Keyword	Type	Default	Description
interpolationMethod	String		Type of (spatial) interpolation. Supported methods: ◇ constant, only with dataFileType=polygon. ◇ triangulation, Delaunay triangulation+linear interpolation. ◇ averaging, grid cell averaging.
operand	String	O	(optional) How this data is combined with previous data for the same quantity (if any). Supported operands: ◇ O, override any previous data. ◇ A, append, sets only where data is still missing. ◇ +, adds the provided values to the existing values. ◇ *, multiplies the existing values by the provided values. ◇ X, takes the maximum of the existing values and the provided values. ◇ N, takes the minimum of the existing values and the provided values.
averagingType	String	mean	(optional) Type of averaging, if interpolationMethod=averaging. Supported types: ◇ mean, simple average ◇ nearestNb, nearest neighbour value ◇ max, highest ◇ min, lowest ◇ invDist, inverse-weighted distance average ◇ minAbs, smallest absolute value ◇ median, median value
averagingRelSize	Double	1.01	(optional) Relative search cell size for averaging.
averagingNumMin	Integer	1	(optional) Minimum number of points in averaging. Must be ≥ 1.
averagingPercentile	Double	0.0	(optional) Percentile value for which data values to include in averaging. 0.0 means off.
extrapolationMethod	Logical	no	Option for (spatial) extrapolation: no or yes.
locationType	String	all	(optional) Target location of interpolation, if quantity is 1d2d distinction possible (see Table D.2). Supported locations: ◇ 1d, interpolate only to 1d nodes/links ◇ 2d, interpolate only to 2d nodes/links ◇ all, interpolate to all nodes/links

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
`value`	Double		(optionally required) Only for `dataFileType=polygon`. The constant value to be set inside for all model points inside the polygon.

Example:

```
# comments
[Initial]
quantity             = waterlevel
dataFile             = iniwaterlevels.asc
dataFileType         = arcinfo
interpolationMethod  = triangulation
locationType         = 2d

[Initial]
quantity             = bedlevel
dataFile             = inibedlevel.ini
dataFileType         = 1dField

[Parameter]
quantity             = frictioncoefficient
dataFile             = manning.xyz
dataFileType         = sample
interpolationMethod  = triangulation

[Parameter]
quantity             = frictioncoefficient
dataFile             = calibration1.pol
dataFileType         = polygon
interpolationMethod  = constant
value                = 0.03
operand              = *
```

D.3 Supported quantities

D.3.1 Accepted quantity names in initial field files

The list of accepted quantity names in initial field files is subdivided into two categories:

- Initial fields
- Spatial model parameters

Table D.2 shows the accepted quantity names, the corresponding names in old external forcing files, the expected units, the possibility of 1d/2d distinction and on which page to find more details. 1d/2d distinction allows to limit a dataset to either the 1D grid points, or 2D, or allow both. This is achieved by setting the `locationType` parameter in the same data block.

Table D.2: *List of accepted field quantity names.*

Quantity	Name in old external forcing file	Unit	1d/2d distinction possible?	Page
Initial fields:				

Quantity	name in old external forcing file	Unit	1d/2d distinction possible?	Page
bedlevel	bedlevel, bedlevel1D, bedlevel2D	[m]	yes	
waterlevel	initialwaterlevel, initialwaterlevel1d, initialwaterlevel2d	[m]	yes	506
waterdepth	N/A	[m]	yes	
InterceptionLayer↩ Thickness	N/A	[m]	yes	285
PotentialEvaporation	N/A	[mm/hr]	yes	286
InfiltrationCapacity[1]	infiltrationcapacity	[mm/day]	yes	285
HortonMaxInfCap[2]	N/A	[mm/hr]	yes	286
HortonMinInfCap	N/A	[mm/hr]	yes	286
HortonDecreaseRate	N/A	[1/hr]	yes	286
HortonRecoveryRate	N/A	[1/hr]	yes	286
Spatial model parameters:				507
frictioncoefficient	frictioncoefficient	depends on its type, see Section 8.4.2.1	Intended for 2D only	

D.3.2 Water levels

Water levels can come from all file types listed below. The restart file sets the water level both at the old and new time step. Non-restart files should be specified using QUANTITY=initialwaterlevel.

D.3.3 Initial velocities

Initial velocities should come from restart files, such that they exactly correspond with the model's waterdepths. They can also come from non-restart files using QUANTITY=initialvelocityx/y, but this is not advised.

D.3.4 Salinity

Salinity concentrations from non-restart files are possible as spatially varying, depth-averaged values, or (for 3D models) as linearly interpolated values between a top layer concentration and a bottom layer concentration. The latter is achieved by specifying two QUANTITY blocks: initialsalinity and initialsalinitytop. . For 3D models, alternatively, spatially constant, but depth-varying salinities can be set using the QUANTITY=initialverticalsalinityprofile, and a vertical profile file (see section D.4.5).

D.3.5 Temperature

Temperature values from non-restart files are possible as depth-averaged values, using QUANTITY=initialtemperature.

[1] Only for constant infiltration (InfiltrationModel = 2).

[2] Only for Horton infiltration (InfiltrationModel = 4).

For 3D models, alternatively, spatially constant, but depth-varying temperatures can be set using the `QUANTITY=initialverticaltemperatureprofile`, and a vertical profile file (see section D.4.5).

D.3.6 Tracers

Initial tracer concentrations from non-restart files are similar to temperature values, now specified using `QUANTITY=initialtracer`<*tracername*>. The set of tracer name(s) is not listed in the MDU-file explicitly, they are inferred from all supplied initial tracer definitions *and* the tracer boundary conditions (see also section E.1.7). The <*tracername*> postfix can uniquely associate an `initialtracer` quantity with a `tracerbnd` quantity.

D.3.7 Sediment

Initial sediment concentrations for the Delft3D-MOR-based sediment transport module (see section 9.4 for an explanation) is via the <∗.sed> file. See the D-Morphology UM (2019) for further details.

D.3.8 Physical coefficients

A second group of spatially varying input are the physical and numerical coefficients listed below. They allow changing model parameters in space or in certain sub-regions of the domain, and as such these values override the uniform values from the MDU file. The input data can come from the in-polygon file type (section D.4.2 or the sample file type (section D.4.3).

quantity name	MDU uniform equivalent	description
`frictioncoefficient`	`UnifFrictCoef`	Friction coefficient (use `IFRCTYP` and `UnifFrictType`, resp. to set the friction type.

`horizontaleddy...`
`...viscositycoefficient`
	`Vicouv`	Horizontal eddy viscosity coefficient (m^2/s).

`horizontaleddy...`
`...diffusivitycoefficient`
	`Dicouv`	Horizontal eddy diffusivity coefficient (m^2/s).

`advectiontype`	`AdvecType`	Type of advection scheme.
`ibotlevtype`	`BedlevType`	Type of bed-level handling.

D.4 Supported file formats

All file formats below, except for restart files, are in *model-independent coordinates* and will be cropped and/or interpolated onto the model grid.

D.4.1 1D field INI file

This section describes the file format that can be used for defining a scalar quantity on a 1D network. This file format is for instance used for the initial conditions defined on a network, in which case it should be referenced as a `dataFile` inside the initial field file (section D.2).

Table D.3: *1D field definition.*

Keyword	Type	Default	Description
`[General]`			
`fileVersion`	String	`2.00`	File version. Do not edit this.
`fileType`	String	`1dField`	File type. Do not edit this.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
[Global]			*One global block per file*
quantity	String		The name of the quantity
unit	String		The unit of the quantity
value	Double		The global default value for this quantity
[Branch]			*Optional: one block per branch*
branchId	String		The name of the branch
numLocations	Int	0	Number of locations on branch. The default 0 value implies branch uniform values.
chainage	Double[]		Space separated list of locations on the branch (m). Locations sorted by increasing chainage. The keyword must be specified if numLocations>0.
values	Double[]		Space separated list of numLocations values; one for each chainage specified. One value required if numLocations=0

Remarks:

- The branchId may contain spaces, with no quoting needed.
- This file does not include an option to specify value locations via x and y.
- The number of values to be specified per line equals the number of locations on that branch. If numLocations is not specified, 0 or 1, one value is expected which will be applied uniformly along the branch. If 2 or more locations have been specified, the values are interpolated *linearly* along the branch in between the locations and assumed constant beyond the first and last location specified.

Example:

```
[General]
    fileVersion          = 2.00
    fileType             = 1dField

[Global]
    quantity             = waterDepth
    unit                 = m
    value                = 10.000                 # default value

[Branch]
    branchId             = Channel1               # overrule for Channel1
    numLocations         = 2                      # at two locations
    chainage             = 0.000 100.000
    values               = 7.000 7.500            # 7 at 0m and 7.5 at 100m

[Branch]
    branchId             = Channel4               # overrule for Channel4
    values               = 8.000                  # uniform value
```

D.4.2 Inside-polygon option

A spatially constant value in a restricted part of the domain can be set by specifying a polygon file (section C.2) in the ext-file as follows:

```
QUANTITY=frictioncoefficient
FILENAME=winterbed.pol
FILETYPE=10
METHOD  =4
VALUE   =55
IFRCTYP =1  # Optional, override uniform [physics] UnifFrictType=..
```

The interpolation option `METHOD=4` simply specifies the no-interpolation is to be performed, only inside-polygon cropping. For `QUANTITY=frictioncoefficient` the default friction type may be overridden with the keyword `IFRCTYP`.

D.4.3 Sample file

Spatial data from sample files (<∗.xyz>, see section C.3) is interpolated by triangle interpolation or sample averaging. The following options control the type of interpolation:

interpolation method	**options**	**description**
`METHOD=5`		Triangle interpolation
	`EXTRAPOLTOL=20`	Allow extrapolation from convex hull of samples up to . . . meters (default: 0).
`METHOD=6`		Sample averaging inside control volumes:
	`AVERAGINGTYPE=1`	Normal (unweighted) averaging
	`AVERAGINGTYPE=2`	Nearest neighbour
	`AVERAGINGTYPE=3`	Maximal value
	`AVERAGINGTYPE=4`	Minimal value
	`AVERAGINGTYPE=5`	Inverse weighted distance averaging
	`AVERAGINGTYPE=6`	Minimal absolute value (Use with care. It takes the absolute value.)
	`RELATIVESEARCHCELLSIZE =...`	Control size of search volume: 1=actual cell, 1.5=50 % larger in all directions.

D.4.4 GeoTIFF file

GeoTIFF[3] is a file format for georeferenced rasterized datasets. When providing a GeoTIFF file as an initial or parameter field (Section D.2), the model coordinate system in that file must match the coordinate system of the D-Flow FM model (e.g., geographic or projected coordinates). Also, only single layer (or band) files are currently supported.

[3] https://trac.osgeo.org/geotiff/

D.4.5 Vertical profile file

Initial values that are constant in 2D space, but vary in depth can be specified using vertical profile files, which are basic polyline files (section C.2). For example for a linearly varying salinity from 30 to 20 ppt from -10 to 0 m :

```
L1
2 2
-10 30
 0 20
```

The METHOD keyword must be specified, but is further ignored: linear interpolation is always used.

D.4.6 Map file

Initial field data from a map file may be used as non-restart input, but this is only supported currently for the legacy MapFormat = 1 (i.e., UGRID map files are not yet supported). Values are read in from the map files as if they were samples, and further treated and interpolated as such. All interpolation options from section D.4.3 can be used here. This option may be used as a non-exact restart state when the model grid has been changed (e.g., locally refined or cropped to a subdomain).

D.4.7 Restart file

Restart files are discussed in section F.3.4. They should be included in your model in the mdu-file as follows:

```
[restart]
RestartFile     = mdu_name_yyyymmdd_HHMMSS_rst.nc
RestartDateTime = # Restart time (YYYYMMDDHHMMSS), only relevant
                 # in case of restart from *_map.nc
```

Concerning parallel runs, each subdomain can generate its own restart/map file. To restart a parallel run, one can use any of the following approaches:

- On each subdomain use its own restart/map file, provided that the partition does not change. (This means that in each partition MDU file specify its own restart/map file.)
- Merge the subdomain restart/map files to one global file (see 6.4.4.2). This file enables to restart a parallel run with a different or the same partition. (This means that specify the merged file for all the partition MDU files.) One can also run a sequential simulation with this merged restart/map file.

E Boundary conditions specification

Boundary conditions are part of the external forcing of a model and and as such declared in the external-forcings file (section C.5).

E.1 Supported boundary types

E.1.1 Open boundaries

E.1.2 Astronomic boundary conditions

Boundary values can be specified for any location in terms of astronomical components in attribute files with extension <cmp> (the BC-format, discussed furtheron, is supported as an alternative). Tidal motion are described as a series of simple harmonic constituent motions, each with its own characteristic period:

$$H(t) = A_0 + \sum_{i=1}^{k} A_i F(c_i) \cos\left(\frac{2\pi}{T(c_i)} t + (V_0 + u)\,(c_i) - G_i\right) \tag{E.1}$$

in which:

c_i	i-th component, specified by label (name)
A_i	amplitude of the i-th component
G_i	phase of the i-th component
$T(c_i)$	period
$H(t)$	boundary value at time t
A_0	constant offset value
k	number of relevant components
$F(c_i)$	nodal amplitude factor
$(V_0 + u)(c_i)$	astronomical argument

The component (by label) c_i, amplitude A_i and phase G_i for each component i are required in the <cmp>-file In addition to the primary constituents, compound and higher harmonic constituents may have to be used. This is the case in shallow water areas for example, where advection, large amplitude to depth ratio, and bottom friction give rise to non-linear interactions.

F, V_0 and u are also *time-dependent*. For a given period, their values are easily calculated or obtained various tidal year books. V_0 is a frequency dependent phase correction factor relating the local time frame of the observations to an internationally agreed celestial time frame. F_i and u_i are slowly varying amplitude and phase corrections, The variations depend on the frequency, often with a cyclic period of 18.6 years. By default, the nodal amplitude factors and astronomical arguments are re-calculated every 6 hours.

E.1.3 Astronomic correction factors

For individual astronomic components, correcting factor α_i for amplitude and offset δ_i may be specified by the user so that the effective amplitude for that component becomes $\alpha_i\, A_i$ and the effective phase becomes $G_i + \delta_i$.

E.1.4 Harmonic flow boundary conditions

Harmonic boundary conditions are specifies in a manner similar their astronomical counterpart, except that the periods (first column of a .cmp file) are declared explicitly, rather than using component names. Equation (E.1) applies with $T(c_i) = T_i$ and furthermore $F = 1$ and $(V_0 + u) = 0$.

E.1.5 QH-relation flow boundary conditions

In this type of boundary condition, a waterlevel is prescribed as a function of the integrated discharge through a crossection defined by the polyline of the boundary.

The relation between waterlevel and discharge is specified in a lookup-table.

Within this table linear interpolation is applied.

Note that only one waterlevel is set for the entire QH-boundary, and that its behaviour is specified by a *single* QH-table for the entire boundary.

E.1.6 Time-series flow boundary conditions

Time-series in the general time-series file format (section C.4) can be used to specify values on boundaries.

Boundary values are provided at discrete points in time, which are then used for interpolation at times in between.

Extrapolation beyond the range of specified times is not supported, Time-series are also supported at multiple vertical levels for quantities that are defined on layers (tracers, salinity, temperature, velocity). These need to be specified in the BC-format (section E.2.3).

E.1.7 Time-series transport boundary conditions

The boundary conditions of tracers, salinity and temperature are specified similar to any other quantity already mentioned. The quantity name should be in the form `tracerbnd`<name>, where name is the user-defined tracer name.

E.1.8 Time-series for the heat model parameters

In order to conduct a heat flux simulation (if in the MDU-file "`Temperature`" under `[physics]` was set to 5), then four time-varying meteorological fields become relevant:

- humidity (or dewpoint)
- airtemperature
- cloudiness
- solar radiation

These are read as a three- or four-column vector (the first three variables of this list, either with or without solar radiation). The corresponding quantity names in the <ext>-file are:

- `humidity_airtemperature_cloudiness`
- `humidity_airtemperature_cloudiness_solarradiation`
- `dewpoint_airtemperature_cloudiness`
- `dewpoint_airtemperature_cloudiness_solarradiation`

As for filetype, currently only the curvilinear format (C.13.2), the multicolumn time series (C.4) and netCDF (C.13.4) are supported.

In the case of netCDF it is also possible to supply `solarradiation` as a separate quantity, and it is possible to supply the term Q_{br} (see Eq.10.19) by the quantity `longwaveradiation`.

E.1.9 Time-series for varying air density

In the computation of the windstress, it is possible to use a spatial and/or time-varying air density. The air density can be specified directly as a time-series or it can be computed from three other time-varying meteorological fields:

- airpressure
- airtemperature
- dewpoint

In order to compute it, option "computedAirdensity" under [wind] in the MDU-file must be set to 1.

This option is only supported for netCDF. The quantity name in the <ext>-file is:

- airdensity

or, for computedAirdensity = 1 the three needed quantity names are:

- airpressure
- airtemperature
- dewpoint

E.2 Boundary signal file formats

E.2.1 The <cmp> format

```
* comments
* ...
c[0]      amplitude[0]    phase[0]
c[1]      amplitude[1]    phase[1]
...
```

where c represents a period in minutes or the name of a valid astronomic component. Amplitudes are assumed to be in the unit of the quantity.

E.2.2 The <qh> format

```
* comments
* ...
discharge[0]      waterlevel[0]
discharge[1]      waterlevel[1]
...
```

E.2.3 The <bc> format

The external forcings file may point to this type of file via the forcingFile keyword (see section C.5.2). The type of forcing is not restricted to boundary conditions, but may also contain meteo time series (global or stations). The <bc> format allows you to combine multiple forcing signals in a single (ASCII) file. Multiple **Boundary** or **Meteo** blocks in the external forcings file may then point to the same file. The file consists of a **General** block followed by one or more **Forcing** blocks. Each forcing block consists of a column-wise table (the data) and a header specifying how this table should be interpreted.

Table E.1: *Description of <bc> format.*

Keyword	Type	Default	Description
`[General]`			
`fileVersion`	String	`1.01`	File version. Do not edit this.
`fileType`	String	`boundConds`	File type. Do not edit this.
`[Forcing]`			*Repeat as needed*
`name`	String		Location name. Possible values are: ◇ `global`: Spatially uniform values, e.g., for meteo forcing. ◇ `<polyline point>`: Boundary polyline (point) reference. ◇ `<stationId>`: Station id, e.g., for meteo stations. More details for each of these options can be found in Section E.2.3.1.
`function`	String		Function type. The supported options are `timeSeries`, `harmonic`, `astronomic`, `harmonic-Correction`, `astronomic-Correction`, `t3D`, `QHTable`, and `constant`. A description of these options follows after this table.
`offset`	Double	0	If `function` equals `timeSeries`, `t3D` or `constant` all values in the table are increased by the offset (after multiplication by factor).
`factor`	Double	1	If `function` equals `timeSeries`, `t3D` or `constant` all values in the table are multiplied with the factor. If `function` equals `harmonic` or `astronomic`) only the amplitude column is multiplied by the given factor.
`verticalPositions`	Double[]		The vertical position is specified as a space- or comma-separated list of floats, the interpretation of which varies according to the vertical position type. The order of the positions in the list becomes relevant when referring to them from the quantity blocks. However, no specific ordering of these positions (ascending or descending) is assumed.
`verticalInterpolation`	String		Possible values are: ◇ `linear`: linear interpolation between vertical positions. ◇ `log`: logarithmic interpolation between vertical positions (e.g. vertical velocity profiles). ◇ `block`: equal to the value at the directly lower specified vertical position.

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
`verticalPositionType`	String		Possible values are: ◇ `percBed`: Percentage w.r.t. water depth from the bed upward. ◇ `ZDatum`: z-coordinate with respect to the reference level of the model. ◇ `ZBed`: Absolute distance from the bed upward. ◇ `ZSurf`: Absolute distance from the free surface downward.
`timeInterpolation`	String		Possible values are: ◇ `linear`: linear interpolation between times. ◇ `Block-From`: equal to that at the start of the time interval (latest specified time value). ◇ `Block-To`: equal to that at the end of the time interval (upcoming specified time value). ◇ `amountToRate`: Input data is interpreted as absolute amount (e.g., millimeters) in past time interval, and returned as an average rate in that time interval (e.g., millimeters/second). Typically used for (but not restricted to) rainfall quantity.
The next three keywords repeated for every column in this data block			
`quantity`	String		Name of quantity
`unit`	String		Unit of quantity
`verticalPositionIndex`	Integer		This is a (one-based) index into the vertical-position-specification, assigning a vertical position to the quantity (`t3D`-blocks only)
The data block			
`...`	This block holds the actual data in columns. The interpretation of the columns is determined by the order of the aforementioned quantity keywords, i.e. the n-th quantity-block refers to the n-th column. The columns should contain valid floating point numbers (scientific notation is allowed). The only exception is the `astronomicComponent` column, which should contain strings representing the component names.		

E.2.3.1 Name in a bc block

The `name` keyword in the above Table E.1 depends on the type of forcing. The following subsections define the details for each of these.

E.2.3.1.1 Global forcings

The location name `global` is reserved for a global, spatially uniform forcing signal, e.g., uniform rainfall.

E.2.3.1.2 Meteo forcings on stations

Future functionality: Time series may be specified for individual locations. The `name` should specify the `<stationId>`, but additionally x- and y- coordinates must be specified. Currently, as the only alternative, station time series from netCDF files must be used. See Section E.2.4 for more details.

E.2.3.1.3 Boundary forcings

Boundary conditions can be prescribed by one or more **Forcing** blocks, each of which is uniquely tied to an individual boundary support point. For a boundary location defined as a polyline, the support points are the ones listed in the <.pli>-file. In that case the name in the bc-file should read `pliname_<point number>`. The point number should be formatted as a 4-digit zero-padded number. Blocks with the function `QHTable` are an exception to this rule, because in qh-boundaries no data is required for individual support points. For this specific type of boundary condition, the name in the <.bc>-file should read `pliname`.

If the boundary location is defined as a node for a 1D network, then the name should equal its `<nodeId>`.

E.2.3.2 Function in a bc block

The `function` keyword specifies how the **Forcing** block should be interpreted. The options are:

`timeSeries`
Value[s] as a function of time. One of the quantities must be called `Time` (see remarks below for unit).

`harmonic`
Period, amplitude, phase. One of the quantities must be called `Harmonic Component` [-]. The other quantities should come in pairs:

◇ *quantity* `amplitude` (the unit depends on the quantity)
◇ *quantity* `phase` (degrees)

`astronomic`
Component-name, amplitude, phase. One of the quantities must be called `Astronomic Component` [-]. The other quantities should come in pairs:

◇ *quantity* `amplitude` (the unit depends on the quantity)
◇ *quantity* `phase` (degrees)

`Harmonic-Correction`
Period, amplitude-factor, phase-offset. One of the quantities must be called `Harmonic Component` [-]. The other quantities should come in pairs:

◇ *quantity* `amplitude` (the unit depends on the quantity)
◇ *quantity* `phase` (degrees)

`Astronomic-Correction`
Component-name, amplitude-factor, phase-offset. One of the quantities must be called `Astronomic Component` [-]. The other quantities should come in pairs:

◇ *quantity* `amplitude` (the unit depends on the quantity)
◇ *quantity* `phase` (degrees)

`t3D`
Values on multiple vertical positions versus time. One of the quantities must be called `Time` (see remarks below for unit).

`QHTable` Discharge as a function of water level. This data block should contain two quantities:

- ◇ *location* `WaterLevel` (m AD)
- ◇ *location* `Discharge` (m^3/s)

`constant` Constant value

Remarks:

- ◇ Although typical files will use UpperCamelCase keywords for chapters and lowerCamelCase for keywords and constants, all strings (including user-defined location and quantity names) will be verified by case insensitive comparison.
- ◇ Valid time units are string indicating the time unit (`days`, `hours`, `minutes`, or `seconds`) `since` a reference date (`YYYY-MM-DD`) and optional time (`HH-MM-SS`) and time zone (`+/-HH`). As in ISO 6801, `T` may be used between date and time; `Z` may be used as time zone indicator.
- ◇ All quantities in a harmonic or astronomic block are forced using the same components. All quantities in a time series block are forced using the same time points.

Example:

```
[General]
    fileVersion          = 1.01
    fileType             = boundConds

[Forcing]
    name                 = left01_0001
    function             = Harmonic
    quantity             = Harmonic Component
    unit                 = -
    quantity             = waterlevelbnd amplitude
    unit                 = m
    quantity             = waterlevelbnd phase
    unit                 = rad/deg/minutes
    0.0  4.114  0.0

[Forcing]
    name                 = left01_0001
    function             = Harmonic
    quantity             = Harmonic Component
    unit                 = -
    quantity             = normalvelocitybnd amplitude
    unit                 = m/s
    quantity             = normalvelocitybnd phase
    unit                 = rad/deg/minutes
    0.0  1.215  0.0

[Forcing]
    name                 = right01_0001
    function             = TimeSeries
    timeInterpolation  = Linear
    quantity             = time
    unit                 = minutes since 2001-01-01
    quantity             = waterlevelbnd
    unit                 = m
       0.000000    2.50
    1440.000000    2.50
```

E.2.4 The netCDF-format

E.2.4.1 Scalar quantity for point-location time-series

The external forcings file may point to this type of file (for boundary condition support points or meteo station time series) via the `forcingFile` keyword (see section C.5.2).

NetCDF files that provide forcing data on point locations should meet the following structure:

- ◇ The location labels have to be stored in a character variable with attribute `cf_role = ’timeseries_id’`. This variable should have two dimensions, the primary being the series dimension and the secondary the string length dimension. The labels should always be unique.
- ◇ The variable containing the timeseries data is identified by its attribute `standard_name = <quantity-name>` and has two dimensions, the series dimension and the time dimension. The latter can be the unlimited dimension.

In analogy with the ASCII BC-format (Section E.2.3), location label and quantity name uniquely identify the timeseries.

Example header of a netCDF file with point data:

```
netcdf timeseries {
dimensions:
        strlen = 7 ;
        node = 3 ;
        time = UNLIMITED ; // (18 currently)
variables:
        double time(time) ;
                time:units = "hours since 2000-01-01 00:00:00" ;
        char location(node, strlen) ;
                location:cf_role = "timeseries_id" ;
        double x(node) ;
                x:axis = "x" ;
                x:units = "km" ;
                x:long_name = "x coordinate of projection" ;
                x:standard_name = "projection_x_coordinate" ;
        double y(node) ;
                y:axis = "y" ;
                y:units = "km" ;
                y:long_name = "y coordinate of projection" ;
                y:standard_name = "projection_y_coordinate" ;
        double rainfall(time, node) ;
                rainfall:long_name = "Rainfall" ;
                rainfall:standard_name = "precipitation" ;
                rainfall:units = "mm" ;
                rainfall:_FillValue = "-999.9f" ;
data:

 location =
  "Station One",
  "Station Two",
  "Station Three" ;
}
```

E.2.4.2 Scalar Quantity for gridded data

The external forcings file may point to this type of file via the `forcingFile` keyword (see section C.5.2).

Gridded data is used to apply per surface external forcing and should therefor spatially be 2D. There are two different ways to supply this data: Either with a timeseries or a harmonic component.

E.2.4.2.1 timeseries

NetCDF files that provide forcing data on a grid with timeseries should meet the following structure:

- The variable containing the timeseries data is identified by its attribute `standard_name = <quantity-name>` and has three dimensions, the time dimension and the grid coordinates.

Example header of a netCDF file with a gridded timeseries:

```
netcdf forcing_timeseries {
dimensions:
        time = 8761 ;
        x = 9 ;
        y = 2 ;
variables:
        double time(time) ;
                time:time_origin = "2013-01-01 00:00:00" ;
                time:long_name = "Time - minutes since 2013-01-01 00:00:00 +00:00" ;
                time:standard_name = "time" ;
                time:calender = "gregorian" ;
                time:units = "minutes since 2013-01-01 00:00:00 +00:00" ;
        double x(x) ;
                x:standard_name = "longitude" ;
                x:long_name = "longitude" ;
                x:units = "degrees_east" ;
        double y(y) ;
                y:standard_name = "latitude" ;
                y:long_name = "latitude" ;
                y:units = "degrees_north" ;
        double air_pressure(time, y, x) ;
                air_pressure:coordinates = "Time y x" ;
                air_pressure:long_name = "Atmospheric Pressure" ;
                air_pressure:standard_name = "air_pressure" ;
                air_pressure:units = "Pa" ;
}
```

E.2.4.2.2 Harmonic component

When the external per surface forcing is periodic in nature, an alternate format can be used in which the data is expressed in the form of a harmonic component. This form generally allows the forcing file to be much smaller while it also provides flexibility in extrapolation.

NetCDF files that provide forcing data on a grid with a harmonic component should meet the following structure:

- The variable containing the amplitude data is identified by its attribute `standard_name = <quantity-name>` and has two dimensions, the grid coordinates.
- The phase offset is identified by the variable name `phase` and should have the same coordinate dimensions as the amplitude data variable and `phase:units = "deg"`.
- The reference time for which the phase offset has been supplied should be defined a global attribute named `reference_time_of_component_phase`.
- The component's period in seconds should be defined a global attribute named `component_period_in_seconds`.

When using this harmonic component format, the timeseries in each grid point i will be given by:

$$H_i(t) = A_i \cos\left(\frac{2\pi}{T}(t - t_{\texttt{ref}}) - \frac{\pi}{180}G_i\right) \tag{E.2}$$

in which:

A_i	amplitude of the component at grid point i
T	period of the component (seconds)
$t_{\texttt{ref}}$	reference time of the component phase (seconds)

G_i phase of the component at grid point i (degrees)

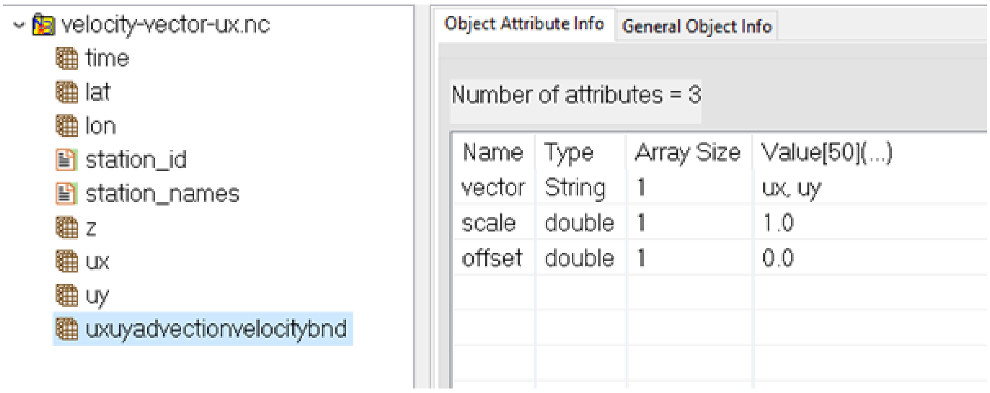
Example header of a netCDF file with a gridded harmonic component:

```
netcdf forcing_component_s1 {
dimensions:
        x = 9 ;
        y = 2 ;
variables:
        double x(x) ;
                x:standard_name = "longitude" ;
                x:long_name = "longitude" ;
                x:units = "degrees_east" ;
        double y(y) ;
                y:standard_name = "latitude" ;
                y:long_name = "latitude" ;
                y:units = "degrees_north" ;
        double amplitude(y, x) ;
                amplitude:coordinates = "y x" ;
                amplitude:long_name = "amplitude_of_airpressure" ;
                amplitude:standard_name = "air_pressure" ;
                amplitude:units = "Pa" ;
        double phase(y, x) ;
                phase:coordinates = "y x" ;
                phase:long_name = "phase_of_component" ;
                phase:standard_name = "phase" ;
                phase:units = "deg" ;

        // global attributes:
                :component = "S1" ;
                :component_period_in_seconds = "86400" ;
                :reference_time_of_component_phase = "2013-01-01 00:00:00 +00:00" ;
}
```

E.2.4.3 Vector quantity

NetCDF file can hold data for a vector quantity. For example, **uxuyadvectionvelocitybnd** boundary condition has X and Y velocity components. To hold this vector quantity, NetCDF file has to have an empty quantity with the name **uxuyadvectionvelocitybnd** in addition to scalar quantities for the velocity components. This empty quantity has to have a string attribute named **vector** that contains the vector element names: X and Y components. These names are separated by comma. The picture below shows an example when the string contains quantities' names **ux** and **uy**. As well the string can contain standard and/or long names.



Number of attributes = 3

Name	Type	Array Size	Value[50](...)
vector	String	1	ux, uy
scale	double	1	1.0
offset	double	1	0.0

Remark:

◇ Be sure that names in the string are not truncated due the string size.

F Output files

D-Flow FM can produce several types of output into several different files. This chapter summarizes the types of output and how to configure them in them model definition. We distinguish five types of output here:

1 The diagnostics file, a log file with live details of a single model run.
2 NetCDF output files for history, map and restart data.
3 Tecplot.
4 Timings file with performance statistics.
5 Shapefiles.

F.1 Diagnostics file

The diagnostics (dia) file is the log file of a single model simulation run. It is an important file to check for warning or error messages, when suspecting a faulty model run. The filename is automatically chosen as <*mdu_name*.dia>. For parallel model runs, each process writes its own file <*mdu_name*_000X.dia>. The dia file has the following global structure:

```
Various INFO/DEBUG messages:
** INFO    : Opened file : unstruc.hlp
[..]
The version of D-Flow FM used for this calculation:
* Deltares, D-Flow FM Version 1.1.145.41271M, Aug 11 2015, 18:05:31
Date and time of model run start:
File creation date: 18:27:44, 06-09-2015
[..]

Check for possible model initialization errors:
** WARNING: readMDUFile: [numerics] cflwavefrac=0.1 was in file, but not used.
[..]

Upon successful initialization, a full printout of the effective model settings is given:
** INFO    : ** Model initialization was successful **
** INFO    : * Active Model definition:
# Generated on 18:27:45, 06-09-2015
# Deltares, D-Flow FM Version 1.1.145.41271M, Aug 11 2015, 18:05:31
[general]
[..]

Next, the time loop is started:
** INFO    : **
** INFO    : Writing initial output to file(s)...
** DEBUG   : Opened netCDF file 'DFM_OUTPUT_035hammen/035hammen_his.nc' as #3.
** DEBUG   : Opened netCDF file 'DFM_OUTPUT_035hammen/035hammen_map.nc' as #4.
** INFO    : Done writing initial output to file(s).
[..]

Finishing summary of model run:
** INFO    : simulation period     (s)  :          6400.0000000000
** INFO    : nr of timesteps       ( )  :           771.0000000000
** INFO    : average timestep      (s)  :             8.3009079118
[..]
Some basic run time statistics
** INFO    : time steps            (s)  :            76.2840000003
** INFO    : Computation started  at: 18:27:45, 06-09-2015
** INFO    : Computation finished at: 18:29:04, 06-09-2015
** INFO    :
Which parallellization options have been active in this run:
** INFO    : MPI    : no.
** INFO    : OpenMP : yes.
        #threads max : 8
```

F.2 Demanding output

The configuration of what output should be produced during a model run is via the MDU file, and several attribute files for history files.

F.2.1 The MDU-file

The MDU-file has an `[output]`-section, with a range of key-value-pair options. See Table A.1 on page 406 for all available output options. Enabling the various output files begins with setting the output interval to a value > 0, see for example `MapInterval` and `HisInterval` in Table A.1.

F.2.2 Observation points

The observation point file defines the locations for time series output of flow "point" variables such as water level and water depth. Observation points in 2D (or 3D) parts of the model domain are used to monitor quantities at grid cell centres. Observation points in 1D parts of the model domain are used to monitor quantities at the grid points. The keyword `ObsFile` should be used to point to the observation point file; optionally a space separated list of files can be specified (see Appendix A).

Two types of observation point files are supported: <∗_obs.ini> files and old <∗.xyn> files. Both file types are described in the subsections below; the first file type is recommended. The following general remarks hold for both types of input:

Remarks:

- ◇ The observation points are stationary by default.
- ◇ Observation points may have non-unique names (also together with moving observation points), but the results can only be distinguished by the column number in the output file, and this practice is not advised.
- ◇ In 2D/3D, the x,y-locations are snapped to the grid cell in which they lie. When an observation point lies on a grid cell edge, snapping results are unpredictable.
- ◇ In 1D, the observation point locations are snapped to the closest computational point; this may not always match the grid cell in which the point is located.
- ◇ Time series are reported for the cell-centered solution data, that is: *no* interpolation to the exact x,y-location takes place.

F.2.2.1 The observation point file with extension <∗_obs.ini>

This file can be used to define observation points in a more detailed way, amongst others in 1D using (`branchId`, `chainage`), but see below for more possibilities.

Table F.1: *Definition of observation points.*

Keyword	Type	Default	Description
`[General]`			
`fileVersion`	String	`2.00`	File version. Do not edit this.
`fileType`		`obsPoints`	File type. Do not edit this.
`[ObservationPoint]`			
`name`	String		Name of the observation point (max. 255 characters).
`locationType`	String	`2d`	(optional) Only when `x` and `y` are also specified. `1d`: snap to closest 1D grid point, `2d`: snap to closest 2D grid cell centre, `all`: snap to closest 1D or 2D point.
`branchId`	String		(optional) Branch on which the observation point is located.
`chainage`	Double		(optional) Location on the branch (m).
`x`	Double		(optional in 1D) x-coordinate of the location of the observation point.
`y`	Double		(optional in 1D) y-coordinate of the location of the observation point.

Remarks:

- ◇ The station id/name may contain spaces, with no quoting needed.
- ◇ If `locationType = all` then the observation point is first checked whether it lies inside a 2D grid cell. When this is not the case, in a second attempt it is snapped to the nearest 1D grid point.
- ◇ Two types of location specifications are available.
  - ◇ When `branchId` is given, the keywords `branchId` and `chainage` will be used for the location specification (only for 1D, the `locationType` is automatically assumed to be `1d`).
  - ◇ When `branchId` is *not* given, the keywords `x` and `y` coordinates will be used. By default this is assumed to be a 2D/3D station location (i.e. `locationType = 2d`). Optionally, this location can be snapped to 1D calculation points using `locationType = 1d`.

Example:

```
[General]
    fileVersion  = 2.00
    fileType     = obsPoints
[ObservationPoint]
    name         = ABDN
    x            = -2.06250000e+00
    y            = 5.71583333e+01
[ObservationPoint]
    name         = AVMH
    x            = -2.73750000e+00
    y            = 5.15083334e+01
[ObservationPoint]
    name         = #St Helier Jersey#
    x            = -2.11250000e+00
    y            = 4.91750000e+01
[ObservationPoint]
    name         = Obs4
    branchId     = Channel1
    chainage     = 275.0
[ObservationPoint]
    name         = Obs5
    locationType = 1d
    x            = -2.31250000e+00
    y            = 5.01750000e+01
```

F.2.2.2 The observation point file with extension <∗.xyn>

The observation point file with extension <∗.xyn> is basically like a sample file (section C.3), but now with station id strings in the third column:

```
-2.06250000e+00 5.71583333e+01 ABDN
-2.73750000e+00 5.15083334e+01 AVMH
-2.11250000e+00 4.91750000e+01 'St Helier Jersey'
```

Remarks:

- ◇ The station id/name may contain spaces, in which case it should be surrounded by single quotes, `'as␣follows'`. Maximum length is 255 characters.
- ◇ Each observation point is first checked whether it lies inside a 2D grid cell. When this is not the case, in a second attempt it is snapped to the nearest 1D grid point. (Equivalent to `locationType = all`.)

F.2.3 Moving observation points

Moving observation points are specified via the <∗.ext>-file:

```
QUANTITY=movingstationtxy
FILENAME=movingstation1.tim
FILETYPE=1
METHOD=1
OPERAND=O
```

The <∗.tim> file (one for each moving observation point) is a standard time series file (section C.4) with three columns, containing the time in minutes since the reference data, the x- and y-position of the moving station at that time.

Remarks:

- Stationary and moving observation points form one large set of observation points in the output his file.
- No space or time interpolation is done: the reported values are instantaneous values for the grid cell in which the moving observation point lies at that each output time.

F.2.4 Observation cross sections

Observation cross sections (or cross sections for short) define the location for time series output of flow "flux" variables such as discharge and velocity. Other typical output quantities are instantaneous as well as cumulative, space-integrated discharges and constituent transport. The keyword `CrsFile` should be used to point to the observation cross section file; optionally a space separated list of files can be specified (see Appendix A).

The location of an observation cross section can be specified using polylines (typically in 2D) or single points (only in 1D). Two types of cross section files are supported: <∗.ini> files and old <∗_crs.pli> files. Both file types are described in the subsections below; the first file type is recommended. The following general remarks hold for both types of input:

Remarks:

- Observation cross sections may have non-unique names, but the results can only be distinguished by the column number in the output file, and this practice is not advised.
- A cross section polyline is snapped to all grid cell edges (i.e., velocity points), whose flow link is intersected by the polyline.
- The flow data on velocity points is integrated along the polyline and across all vertical layers, resulting in a single time series per cross section and flow quantity.
- When a single polyline happens to intersect both 1D and 2D flow links, all of these are included and flow data is integrated for 1D and 2D together.

F.2.4.1 Observation cross section file with extension <∗_crs.ini>

This file can be used to define cross sections in a more detailed way, amongst others in 1D using (`branchId`, `chainage`), but see below for more possibilities.

Table F.2: *Definition of observation cross sections.*

Keyword	Type	Default	Description
`[General]`			
`fileVersion`	String	`2.00`	File version. Do not edit this.
`fileType`		`obsCross`	File type. Do not edit this.
`[ObservationCrossSection]`			

(continued on next page)

Keyword	Type	Default	Description
			(continued from previous page)
name	String		Name of the cross section (max. 255 characters).
branchId	String		(optional) Branch on which the cross section is located
chainage	Double		(optional) Location on the branch (m)
numCoordinates	int		(optional) number of values in xCoordinates and yCoordinates. This value should be greater or equal 2.
xCoordinates	Double []		(optional) x-coordinates of the cross section line. (number of values = numCoordinates)
yCoordinates	Double []		(optional) y-coordinates of the cross section line. (number of values = numCoordinates)

Remarks:

◇ The cross section id/name may contain spaces, with no quoting needed.

◇ Two types of location specifications are available.

- ◇ When branchId is given, the keywords branchId and chainage will be used for the location specification (only for 1D).
- ◇ When branchId is *not* given, the keywords numCoordinates, xCoordinates and yCoordinates will be used to define a polyline. Output will be accumulated over all flow links that the polyline intersects.

Example:

```
[General]
    fileVersion  = 2.00
    fileType     = obsCross
[ObservationCrossSection]
    name           = CrossSec1
    numCoordinates = 4
    xCoordinates   = 132345.0 132165.0 131940.0 131820.0
    yCoordinates   = 549030.0 549285.0 549550.0 549670.0
[ObservationCrossSection]
    name           = CrossSec2
    numCoordinates = 3
    xCoordinates   = 131750.0 131595.0 131415.0
    yCoordinates   = 549865.0 550025.0 550175.0
[ObservationCrossSection]
    name           = CrossSec3
    branchId       = Channel1
    chainage       = 300.000
```

F.2.4.2 Observation cross section file with extension <∗_crs.pli>

The format of the <∗_crs.pli> file is identical to that of a general polyline/polygon file (section C.2).

```
CrossSec1
4  2
            132345.0    549030.0
            132165.0    549285.0
            131940.0    549550.0
            131820.0    549670.0
CrossSec2
```

```
3  2
            131750.0    549865.0
            131595.0    550025.0
            131415.0    550175.0
```

Remarks:

- ◇ The first header line of each polyline block contains the name of each cross section.
- ◇ The cross section id/name may contain spaces, no quotes or other delimiters must be used.

F.3 NetCDF output files

All flow data output produced by D-Flow FM is written in the netCDF format. This cross-platform storage format is widely used in the world. The contents of a file can be documented using variable attributes and standardized meta data.

The following conventions are used in some of our output file types:

- ◇ CF-conventions for both the timeseries (his) files and the spatial (map/class map/Fourier) files. More information on: https://cfconventions.org/.
- ◇ UGRID-conventions for the unstructured grids in the spatial (map/class map/Fourier) files. More information on: http://ugrid-conventions.github.io/ugrid-conventions/.
  Additionally, for 1D and 1D2D grids, the Deltares-extension on top of UGRID is used. More information about this is in Section B.2.
- ◇ ACDD-conventions for the geospatial and time bounds in the spatial (map/class map/Fourier) files. More information on: https://wiki.esipfed.org/Category:Attribute_Conventions_Dataset_Discovery.

Spherical coordinates in NetCDF output files

The coordinate variables in all NetCDF output files are in the same coordinate system as the input grid file (<_net.nc>). If this input file contains metadata describing grid mapping/coordinate transformation, that metadata is also copied over into most output NetCDF files (specifically: into all UGRID-type map/class map/Fourier files, and the history files). For models using a Cartesian/metric coordinate system, an option is available to automatically add the equivalent latitude and longitude coordinate values for all locations in the NetCDF output files. To enable this feature:

- ◇ Verify that the input <_net.nc> file contains a grid mapping variable with coordinate transformation parameters. In order to be found, the grid mapping variable should be called `projected_coordinate_system`, or have the attribute `:grid_mapping_name`. Additionally, it must have an attribute `:proj4_params` containing a PROJ-compatible coordinate transformation string[1].
- ◇ If the input contains the above metadata, switch on the lat-lon conversion in the MDU file using keyword `[output] NcWriteLatLon = 1`.

The fragment below shows an example of NetCDF output for a map file, where the input contained a detailed grid mapping variable for UTM zone 32N:

```
netcdf example_map {
// [..]
variables:
        int projected_coordinate_system ;
                projected_coordinate_system:name = "ETRS89 / UTM zone 32N" ;
                projected_coordinate_system:epsg = 25832 ;
                projected_coordinate_system:grid_mapping_name = "transverse_mercator" ;
                projected_coordinate_system:longitude_of_prime_meridian = 0. ;
                projected_coordinate_system:semi_major_axis = 6378137. ;
                projected_coordinate_system:semi_minor_axis = 6356752.31414036 ;
                projected_coordinate_system:inverse_flattening = 298.257222101 ;
                projected_coordinate_system:EPSG_code = "EPSG:25832" ;
                projected_coordinate_system:value = "value is equal to EPSG code" ;
                projected_coordinate_system:proj4_params =
                        "+proj=utm +zone=32 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs" ;
```

[1] https://proj.org/usage/projections.html

```
                projected_coordinate_system:projection_name = "unknown" ;
                projected_coordinate_system:wkt = "PROJCS[\"ETRS89 / UTM zone 32N\",\n",
                        "    GEOGCS[\"ETRS89\",\n",
                        "        DATUM[\"European_Terrestrial_Reference_System_1989\",\n",
                        "            SPHEROID[\"GRS 1980\",6378137,298.257222101,\n",
                        "                AUTHORITY[\"EPSG\",\"7019\"]],\n",
                        "            TOWGS84[0,0,0,0,0,0,0],\n",
                        "            AUTHORITY[\"EPSG\",\"6258\"]],\n",
                        "        PRIMEM[\"Greenwich\",0,\n",
                        "            AUTHORITY[\"EPSG\",\"8901\"]],\n",
                        "        UNIT[\"degree\",0.0174532925199433,\n",
                        "            AUTHORITY[\"EPSG\",\"9122\"]],\n",
                        "        AUTHORITY[\"EPSG\",\"4258\"]],\n",
                        "    PROJECTION[\"Transverse_Mercator\"],\n",
                        "    PARAMETER[\"latitude_of_origin\",0],\n",
                        "    PARAMETER[\"central_meridian\",9],\n",
                        "    PARAMETER[\"scale_factor\",0.9996],\n",
                        "    PARAMETER[\"false_easting\",500000],\n",
                        "    PARAMETER[\"false_northing\",0],\n",
                        "    UNIT[\"metre\",1,\n",
                        "        AUTHORITY[\"EPSG\",\"9001\"]],\n",
                        "    AXIS[\"Easting\",EAST],\n",
                        "    AXIS[\"Northing\",NORTH],\n",
                        "    AUTHORITY[\"EPSG\",\"25832\"]]" ;
// [..]
        double mesh2d_node_x(mesh2d_nNodes) ;
                mesh2d_node_x:units = "m" ;
                mesh2d_node_x:standard_name = "projection_x_coordinate" ;
                mesh2d_node_x:long_name = "x-coordinate of mesh nodes" ;
        double mesh2d_node_y(mesh2d_nNodes) ;
                mesh2d_node_y:units = "m" ;
                mesh2d_node_y:standard_name = "projection_y_coordinate" ;
                mesh2d_node_y:long_name = "y-coordinate of mesh nodes" ;
        double mesh2d_node_lon(mesh2d_nNodes) ;
                mesh2d_node_lon:units = "degrees_east" ;
                mesh2d_node_lon:standard_name = "longitude" ;
                mesh2d_node_lon:long_name = "longitude coordinate of mesh nodes" ;
        double mesh2d_node_lat(mesh2d_nNodes) ;
                mesh2d_node_lat:units = "degrees_north" ;
                mesh2d_node_lat:standard_name = "latitude" ;
                mesh2d_node_lat:long_name = "latitude coordinate of mesh nodes" ;
```

Detailed per NetCDF output file type

The following sections describe the specific details for each type of output file.

F.3.1 Timeseries as netCDF his-file

The history file <*mdu_name*_his.nc> contains the dimensions and variable definitions in its header, followed by the actual data. For parallel runs, all data is gathered by domain #0 into <*mdu_name*_0000_his.nc> (section 6.4.4).

F.3.1.1 Dimensions

The following dimensions are defined in a his file header:

```
netcdf weirfree_his {
dimensions:
       time = UNLIMITED ; // (64 currently)
       name_len = 64 ;
       stations = 18 ;
       cross_section = 4 ;
       cross_section_name_len = 64 ;
       cross_section_pts = 3 ;
       npumps = 2 ;
       // and possibly nweirgens, nweirs, ngategens, ngates, nsources/sinks
```

F.3.1.2 Location variables

The original location and IDs for stations, cross sections and sources/sinks are also included in the his file:

```
variables:
 double station_x_coordinate(stations) ;
   station_x_coordinate:units = "m" ;
   station_x_coordinate:standard_name = "projection_x_coordinate" ;
   station_x_coordinate:long_name = "x-coordinate" ;
 double station_y_coordinate(stations) ;
   station_y_coordinate:units = "m" ;
   station_y_coordinate:standard_name = "projection_y_coordinate" ;
   station_y_coordinate:long_name = "x-coordinate" ;
 char station_name(stations, name_len) ;
   station_name:cf_role = "timeseries_id" ;
   station_name:long_name = "Observation station name" ;

 double cross_section_x_coordinate(cross_section, cross_section_pts) ;
   cross_section_x_coordinate:units = "m" ;
   cross_section_x_coordinate:standard_name = "projection_x_coordinate" ;
   cross_section_x_coordinate:long_name = "x-coordinate" ;
 double cross_section_y_coordinate(cross_section, cross_section_pts) ;
   cross_section_y_coordinate:units = "m" ;
   cross_section_y_coordinate:standard_name = "projection_y_coordinate" ;
   cross_section_y_coordinate:long_name = "y-coordinate" ;
 char cross_section_name(cross_section, cross_section_name_len) ;

 double source_sink_x_coordinate(source_sink, source_sink_pts) ;
   source_sink_x_coordinate:units = "m" ;
   source_sink_x_coordinate:standard_name = "projection_x_coordinate" ;
   source_sink_x_coordinate:long_name = "x-coordinate" ;
 double source_sink_y_coordinate(source_sink, source_sink_pts) ;
   source_sink_y_coordinate:units = "m" ;
   source_sink_y_coordinate:standard_name = "projection_y_coordinate" ;
   source_sink_y_coordinate:long_name = "y-coordinate" ;
 char source_sink_name(source_sink, source_sink_name_len) ;
```

F.3.1.3 Overview of output quantities in a netCDF his-file

An overview of output quantities in a his-file is given in Table F.3 and the following subsections. These output quantities are present in the his-file only when the output object exists in the model. For example, quantities on observation stations are present only when observation station(s) exists in the model.

Table F.3: *Global overview of output quantities in the history file.*

Quantity name	MDU keywords for output	More information
Basic condition for his output	`HisInterval > 0`	See Section F.2.1.
Quantities on observation stations.		See section F.3.1.4.
Quantities on observation cross sections.		See section F.3.1.5.
Quantities on runup gauges.		
Quantities on sources and sinks.	`Wrihis_sourcesink > 0.`	
Quantities for mass balance.	`Wrihis_balance > 0.`	See section F.3.1.6.
Quantities on hydraulic structures.		See section F.3.1.9.
Quantities on laterals.	`Wrihis_lateral > 0.`	See section F.3.1.7.
Quantities for dredging and dumping.	`[sediment] DredgeFile` is specified.	
Quantities for checkerboard monitor.	`[numerics] checkerboardmonitor = 1.`	
`time`		Instant time.
`timestep`		Latest computational timestep size in each output interval [s].

F.3.1.4 Variables on stations

Variables on observations station are typically defined as follows:

```
double waterlevel(time, stations) ;
  waterlevel:standard_name = "sea_surface_level" ;
  waterlevel:long_name = "Water level" ;
  waterlevel:units = "m" ;
  waterlevel:coordinates = "station_x_coordinate station_y_coordinate station_name" ;
  waterlevel:_FillValue = -999. ;
```

Output for observation stations is available as time series in the history file. The output is shown in history file when one or more observation stations are defined in the model. The available output quantities are listed in Table F.4. These output variables are quantities locating on the observation stations. Such output can be switched on/off by setting value 1/0, respectively, to MDU [output] keywords, as listed in the third column of Table F.4 (More details about these MDU keywords refer to Table A.1.) If for one variable there is no MDU keyword shown in the third column, then the output of this variable is always switched on.

Table F.4: *Output quantities of observation stations in history file.*

Quantity name	Description [Unit]	MDU `[output]` keyword to switch on/off this output
`station_id`	Identification string of the observation stations.	
`station_name`	Names of the observation stations.	
`station_x_coordinate`	Original x-coordinates of stations (non-snapped) [m].	
`station_y_coordinate`	Original y-coordinates of stations (non-snapped) [m].	
# The following variables are only for 3D models:		
`zcoordinate_c`	Vertical coordinate at center of flow element and layer [m].	`Wrihis_zcor` (default=1)
`zcoordinate_w`	Vertical coordinate at centre of flow element and at layer interface [m].	`Wrihis_zcor` (default=1)
`zcoordinate_wu`	Vertical coordinate at edge of flow element and at layer interface [m].	`Wrihis_zcor` (default=1)
# The above variables are only for 3D models.		
`station_geom`	Geometry variables, see Section F.3.1.10.	
# End of time-independent variables.		
`waterlevel`	Water level [m].	`Wrihis_waterlevel_s1` (default=1)
`bedlevel`	Bottom level [m].	`Wrihis_bedlevel` (default=1)
`waterdepth`	Water depth [m].	`Wrihis_waterdepth` (default=0)
`tausx`, `tausy`	x-, y-components of mean bottom shear stress vector [Pa].	`Wrihis_taucurrent` (default=1)
`x_velocity`, `y_velocity`	x-, y-components of flow element center velocity vector [m/s].	`Wrihis_velocity_vector` (default=1)
# Following are only for 3D models:		
`z_velocity`	z-component of flow element center velocity vector [m/s].	`Wrihis_velocity_vector` (default=1)
`depth-averaged_x_velocity`, `depth-averaged_y_velocity`	x-, y-components of flow element depth-averaged center velocity vector [m/s].	`Wrihis_velocity_vector` (default=1)
`tke`	Turbulent kinetic energy [m^2/s^2] when MDU keyword `[numerics] Turbulencemodel >= 3.`	`Wrihis_turbulence` (default=1)
`vicww`	Turbulent vertical eddy viscosity [m^2/s] when MDU keyword `[numerics] Turbulencemodel > 1.`	`Wrihis_turbulence` (default=1)
`eps`	Turbulent energy dissipation [m^2/s^3] when MDU keyword `[numerics] Turbulencemodel = 3.`	`Wrihis_turbulence` (default=1)

(continued on next page)

(continued from previous page)

Quantity name	Description	MDU `[output]` keyword to switch on/off this output
`tau`	Turbulent time scale [1/s] when MDU keyword `[numerics] Turbulencemodel = 4`.	`Wrihis_turbulence` (default=1)
`rich`	Richardson Nr [-].	`Richardsononoutput` (default=0)
# The above are only for 3D models.		
`salinity`	Salinity [ppt] when MDU keyword `[physics] Salinity > 0`.	`Wrihis_salinity` (default=1)
`velocity_magnitude`	Velocity magnitude [m/s]. Or Eulerian velocity magnitude [m/s] when MDU keyword `[output] EulerVelocities` equals 1.	`Wrihis_velocity` (default=1)
`discharge_magnitude`	Average discharge magnitude [m^3/s].	`Wrihis_discharge` (default=1)
`R`	Roller energy per square meter [J/m^2] when MDU keyword `[waves] Wavemodelnr = 4`.	
# The following variables are for models when MDU keyword `[waves] Wavemodelnr > 0:`		
`hwav`	Significant wave height [m]. Or root mean square wave height based on wave energy [m] when MDU keyword `[waves] jahissigwav = 0`.	`Wrihis_waves` (default=1)
`twav`	Wave period [s].	`Wrihis_waves` (default=1)
`phiwav`	Wave from direction [deg from N].	`Wrihis_waves` (default=1)
`rlabda`	Wave length [m].	`Wrihis_waves` (default=1)
`uorb`	Orbital velocity [m/s].	`Wrihis_waves` (default=1)
`ustokes`, `vstokes`	x-, y-components of Stokes drift [m/s].	`Wrihis_waves` (default=1)
# The above variables are for models when MDU keyword `[waves] Wavemodelnr > 0`.		
`wtau`	Mean bed shear stress [Pa].	`Wrihis_taucurrent` (default=1)
# The following variables are for models with temperature:		
`temperature`	Temperature [°C] when MDU keyword `[physics] Temperature > 0`.	`Wrihis_temperature` (default=1)
`wind`	Wind speed [m/s] when MDU keyword `[physics] Temperature > 1`.	`Wrihis_temperature` and `Wrihis_heat_fluxes` (default=1)
`Tair`	Air temperature [°C] when MDU keyword `[physics] Temperature > 1`.	`Wrihis_temperature` and `Wrihis_heat_fluxes` (default=1)
`rhum`	Relative humidity [-] when MDU keyword `[physics] Temperature = 5`.	`Wrihis_temperature` and `Wrihis_heat_fluxes` (default=1)
`clou`	Cloudiness [-] when MDU keyword `[physics] Temperature = 5`.	`Wrihis_temperature` and `Wrihis_heat_fluxes` (default=1)

(continued on next page)

(continued from previous page)

Quantity name	Description	MDU [output] keyword to switch on/off this output
Qsun	Solar influx [W/m^2] when MDU keyword [physics] Temperature = 5.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
Qeva	Evaporative heat flux [W/m^2] when MDU keyword [physics] Temperature = 5.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
Qcon	Sensible heat flux [W/m^2] when MDU keyword [physics] Temperature = 5.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
Qlong	Long wave back radiation [W/m^2] when MDU keyword [physics] Temperature = 5.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
Qfreva	Free convection evaporative heat flux [W/m^2] when MDU keyword [physics] Temperature = 5.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
Qfrcon	Free convection sensible heat flux [W/m^2] when MDU keyword [physics] Temperature = 5.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
Qtot	Total heat flux [W/m^2] when MDU keyword [physics] Temperature > 1.	Wrihis_temperature and Wrihis_heat_fluxes (default=1)
# The above variables are for models with temperature.		
density	Density [kg/m^3] when MDU keyword [physics] Temperature or [physics] Salinity or [sediment] Sedimentmodelnr is larger than 0.	Wrihis_density (default=1)
Names of all constituent	All constituents [unit depends] when there is one or more constituents.	Wrihis_constituents (default=1)
Names of all constituent_3D	All constituents [unit depends] when there is one or more constituents and MDU keyword [processes] Wriwaqbot3Doutput = 1.	Wrihis_constituents (default=1)
All water quality bottom variables	All water quality bottom variables [unit depends] when there is one or more water quality bottom variables.	
water_quality_output_j, j=1,...,noout_user	Water quality output (from 1 to number of user outputs) [unit depends] when water quality process is either substances initiated or processes activated.	
water_quality_stat_j, j=1,...,noout_statt	Water quality statistic variables (from 1 to number of statistic outputs) [unit depends] when water quality process is either substances initiated or processes activated.	
# The following are sediment transport variables:		
# for model when both MDU keywords [sediment] SedFile and MorFile are not empty, and Sedimentmodelnr is 4.		
sedfrac_name	Sediment fraction identifier [-] when the computed first sediment fraction is positive.	Wrihis_sediment (default=1)

(continued on next page)

(continued from previous page)

Quantity name	Description	MDU `[output]` keyword to switch on/off this output
`seddif`	(Only for 3D models) sediment vertical diffusion [m^2/s] when the computed first sediment fraction is positive.	`Wrihis_sediment` (default=1)
`sed`	Sediment concentration [kg/m^3] when the computed first sediment fraction is positive.	`Wrihis_sediment` (default=1)
`ws`	Sediment settling velocity [m/s] when the computed first sediment fraction is positive.	`Wrihis_sediment` (default=1)
`taub`	Bed shear stress for morphology [Pa] when MDU keywords `[sediment] Sedimentmodelnr` is positive, and MorFile keyword `[Output] taub` is true.	`Wrihis_sediment` (default=1)
`sbcx`, `sbcy`	x-, y-components of current related bedload transport, with unit [kg/s/m] if MorFile keyword `[Output] TranspType = 0`, or [m^3/s/m] if MorFile keyword `[Output] TranspType = 1 or 2`. Output of these variables is available when MDU keywords `Sedimentmodelnr` is positive, and MorFile keyword `[Output] BedTranspDueToCurrentsAtZeta` is true.	`Wrihis_sediment` (default=1)
`sbwx`, `sbwy`	x-, y-components of wave related bedload transport, with unit [kg/s/m] if MorFile keyword `[Output] TranspType = 0`, or [m^3/s/m] if MorFile keyword `[Output] TranspType = 1 or 2`. Output of these variables is available when MorFile keyword `[Output] BedTranspDueToWavesAtZeta` is true, and MDU keywords `[waves] Wavemodelnr > 0`, and `[waves] flowWithoutWaves` is false.	`Wrihis_sediment` (default=1)
`sswx`, `sswy`	x-, y-components of wave related suspended transport, with unit [kg/s/m] if MorFile keyword `[Output] TranspType = 0`, or [m^3/s/m] if MorFile keyword `[Output] TranspType = 1 or 2`. Output of these variables is available when MorFile keyword `[Output] SuspTranspDueToWavesAtZeta` is true, and MDU keywords `[waves] Wavemodelnr > 0`, and `[waves] flowWithoutWaves` is false.	`Wrihis_sediment` (default=1)
`sscx`, `sscy`	x-, y-components of current related suspended transport, with unit [kg/s/m] if MorFile keyword `[Output] TranspType = 0`, or [m^3/s/m] if MorFile keyword `[Output] TranspType = 1 or 2`. Output of these variables is available when when MorFile keyword `[Output] SuspTranspDueToCurrentsAtZeta` is true.	`Wrihis_sediment` (default=1)
`sourse`	Source aterm suspended sediment transport [kg/m^3/s], when MorFile keyword `[Output] SourceSinkTerms` is true.	`Wrihis_sediment` (default=1)
`sinkse`	Sink term suspended sediment transport [1/s], when MorFile keyword `[Output] SourceSinkTerms` is true.	`Wrihis_sediment` (default=1)
`bodsed`	Available sediment mass in the bed [kg/m^2], when MorFile keyword `[Underlayer] IUnderLyr = 1`.	`Wrihis_sediment` (default=1)

(continued on next page)

(continued from previous page)

Quantity name	Description	MDU `[output]` keyword to switch on/off this output
`dpsed`	Sediment thickness in the bed [m], when MorFile keyword `[Underlayer] IUnderLyr = 1`.	`Wrihis_sediment` (default=1)
`msed`	Available sediment mass in a layer of the bed [kg/m^2], when MorFile keyword `[Underlayer] IUnderLyr = 2`.	`Wrihis_sediment` (default=1)
`thlyr`	Thickness of a layer of the bed [m], when MorFile keyword `[Underlayer] IUnderLyr = 2`.	`Wrihis_sediment` (default=1)
`poros`	Porosity of a layer of the bed [-], when MorFile keywords `[Underlayer] IUnderLyr = 2` and `[Underlayer] IPorosity > 0`.	`Wrihis_sediment` (default=1)
`lyrfrac`	Volume fraction in a layer of the bed [m], when MorFile keyword `[Underlayer] IUnderLyr = 2`.	`Wrihis_sediment` (default=1)
`frac`	Availability fraction in top layer [-], when MorFile keyword `[output] frac` is true.	`Wrihis_sediment` (default=1)
`mudfrac`	Mud fraction in top layer [-], when MorFile keyword `[output] MudFrac` is true.	`Wrihis_sediment` (default=1)
`sandfrac`	Sand fraction in top layer [-], when MorFile keyword `[output] SandFrac` is true.	`Wrihis_sediment` (default=1)
`fixfac`	Reduction factor due to limited sediment thickness [-], when MorFile keyword `[output] FixFac` is true.	`Wrihis_sediment` (default=1)
`hidexp`	Hiding and exposure factor [-], when MorFile keyword `[output] HidExp` is true.	`Wrihis_sediment` (default=1)
`mfluff`	Sediment mass in fluff layer [-], when MorFile keyword `[FluffLayer] Type > 0` and number of suspended sediment fractions is positive.	`Wrihis_sediment` (default=1)
# The above are sediment transport variables.		
# for model when both MDU keywords `[sediment] SedFile` and `MorFile` are not empty, and `Sedimentmodelnr` is 4.		
`sediment_concentration`	Sediment concentration [kg/m^3], when MDU keywords `[waves] Wavemodelnr > 0` and, either MDU keywords `[sediment] SedFile` or `MorFile` is empty or `Sedimentmodelnr` is not 4.	`Wrihis_sediment` (default=1)
# Variables for wind models		
`patm`	Atmospheric pressure [N/m^2], when atmospheric pressure is provided via the EXT file, for example with quantity `airpressure_windx_windy` (see Table C.5).	`Wrihis_wind` (default=1)
`windx`, `windy`	x-, y-components of the wind vector [m/s], when wind is provided via the EXT file, for example with quantity `airpressure_windx_windy` (see Table C.5), or when `[external forcing] WindExt > 0`.	`Wrihis_wind` (default=1)

(continued on next page)

(continued from previous page)

Quantity name	Description	MDU [output] keyword to switch on/off this output
rain	Precipitation rate [mm/day], when rainfall is provided via the EXT file, for example with quantity rainfall_rate (see Table C.5), or when MDU keyword [external forcing] Rainfall > 0.	Wrihis_rain (default=1)
infiltration_cap, infiltration_actual	Infiltration capacity and actual infiltration rate [mm/hr], respectively. Output of these variables are available when MDU keyword [grw] Infiltrationmodel = 2 or 4.	Wrihis_infiltration (default=1)

F.3.1.5 Variables on cross sections

Variables on cross sections are typically defined as follows:

```
double cross_section_discharge(time, cross_section) ;
cross_section_discharge:units = "m3 s-1" ;
cross_section_discharge:coordinates = "cross_section_name" ;
```

Output on cross sections is available as time series in the history file. The output is only included when one or more cross sections are defined in the model. The available output quantities are listed in Table F.5. These output variables are quantities located on the cross sections. Output of these variables is always switched on.

Table F.5: *Output quantities on cross sections in history file.*

Quantity name	Description [Unit]
`cross_section_name`	Names of cross sections [-].
`cross_section_geom`	Geometry variables, see Section F.3.1.10
# End of time-independent variables.	
`cross_section_discharge`	Discharge through cross sections [m^3/s].
`cross_section_cumulative_discharge`	Cumulative discharge through cross sections [m^3]
`cross_section_area`	Area of cross sections [m^2]
`cross_section_velocity`	Averaged velocity (by area) through cross sections [m/s].
# The following variables are for transport models,	
`cross_section_cumulative_⟨name of constituent⟩`	Cumulative flux (based on upwind flow cell) for each constituent. If the constituent is sediment, then the unit is [kg] when MorFile keyword `[output] TranspType` is 0, and is [m^3] when this keyword is 1 or 2. For non-sediment constituent, the unit is [constituent unit m^3] where [constituent unit] is specified by users, if it is not specified, then the unit is [-].
`cross_section_⟨name of constituent⟩`	Flux (based on upwind flow cell) for each constituent. If the constituent is sediment, then the unit is [kg/s] when MorFile keyword `[output] TranspType` is 0, and is [m^3/s] when this keyword is 1 or 2. For non-sediment constituent, the unit is [constituent unit m^3/s] where [constituent unit] is specified by users, if it is not specified, then the unit is [-].
# The above variables are for transport models.	
# The following variables are for sediment models,	
# when MDU keyword `[sediment] Sedimentmodelnr = 4` and number of sediment tractions is positive.	
`cross_section_bedload_sediment_transport`	Cumulative bed load sediment transport [kg].
`cross_section_suspended_sediment_transport`	Cumulative suspended load sediment transport [kg], when number of suspended sediment fractions is positive.
`cross_section_bedload_sediment_transport_⟨name of sediment fraction⟩`	cumulative bed load sediment transport per fraction [kg]

F.3.1.6 Mass balance output

The history file may also contain model-global, time-integrated mass balance output, for example:

```
double WaterBalance_total_volume(time) ;
  WaterBalance_total_volume:units = "m3" ;
```

The available output quantities are listed in Table F.6. Output of these variables is switched on or off by setting MDU keyword `[output] Wrihis_balance = 1 or 0`, respectively (default is `1`).

Table F.6: *Output quantities for the mass balance in history file.*

Quantity name	Description	[Unit]
`water_balance_total_volume`	Total volume at the end of timestep.	[m^3]
`water_balance_storage`	Net storage since Tstart.	[m^3]
`water_balance_volume_error`	Volume error since Tstart	[m^3]
`water_balance_boundaries_in`	Cumulative inflow through boundaries since Tstart.	[m^3]
`water_balance_boundaries_out`	Cumulative outflow through boundaries since Tstart.	[m^3]
`water_balance_boundaries_total`	Net inflow through boundaries since Tstart.	[m^3]
`water_balance_exchange_with_1D_in`	Cumulative inflow via SOBEK–D-Flow FM 1D–2D boundaries since Tstart.	[m^3]
`water_balance_exchange_with_1D_out`	Cumulative outflow via SOBEK–D-Flow FM 1D–2D boundaries since Tstart.	[m^3]
`water_balance_exchange_with_1D_total`	Net inflow via SOBEK–D-Flow FM 1D–2D boundaries since Tstart.	[m^3]
`water_balance_precipitation_total`	Total precipitation volume since Tstart.	[m^3]
`water_balance_evaporation`	Evaporation.	[m^3]
`water_balance_source_sink`	Net inflow via source–sink elements since Tstart.	[m^3]
`InternalTidesDissipation`	Total internal tides dissipation, when `internaltidesfrictioncoefficient` in EXT file is specified.	[TJ].
`Gravitational_Input`	Total Gravitational Input (incl. SAL), when MDU keyword `[physics] TidalForcing = 1`.	[TJ]
`SAL_Input`	Total input of self attraction and loading, when MDU keyword `[physics] SelfAttractionLoading > 0`.	[TJ]
`SAL_Input_2`	Total input of self attraction and loading with different formulation, when MDU keyword `[physics] SelfAttractionLoading > 0`.	[TJ]
`water_balance_groundwater_in`	Cumulative volume received from groundwater.	[m^3]
`water_balance_groundwater_out`	Cumulative volume out towards groundwater.	[m^3]
`water_balance_groundwater_total`	Total net inflowing volume from groundwater.	[m^3]
`water_balance_laterals_in`	Cumulative inflowing volume through laterals since Tstart.	[m^3]
`water_balance_laterals_out`	Cumulative outflowing volume through laterals since Tstart.	[m^3]
`water_balance_laterals_total`	Total net inflowing volume through laterals since Tstart.	[m^3]
`water_balance_laterals_in_1D`	Cumulative inflowing volume through 1D-laterals since Tstart.	[m^3]

(continued on next page)

(continued from previous page)

Quantity name	Description	[Unit]
`water_balance_laterals_out_1D`	Cumulative outflowing volume through 1D-laterals since Tstart.	[m^3]
`water_balance_laterals_total_1D`	Total net inflowing volume through 1D-laterals since Tstart.	[m^3]
`water_balance_laterals_in_2D`	Cumulative inflowing volume through 2D-laterals since Tstart.	[m^3]
`water_balance_laterals_out_2D`	Cumulative outflowing volume through 2D-laterals since Tstart.	[m^3]
`water_balance_laterals_total_2D`	Total net inflowing volume through 2D-laterals since Tstart.	[m^3]
`water_balance_Qext_in`	Cumulative inflowing volume through external discharge since Tstart.	[m^3]
`water_balance_Qext_out`	Cumulative outflowing volume through external discharge since Tstart.	[m^3]
`water_balance_Qext_total`	Total net inflowing volume through external discharge since Tstart.	[m^3]
`water_balance_Qext_in_1D`	Cumulative inflowing volume through 1D external discharge since Tstart.	[m^3]
`water_balance_Qext_out_1D`	Cumulative outflowing volume through 1D external discharge since Tstart.	[m^3]
`water_balance_Qext_total_1D`	Total net inflowing volume through 1D external discharge since Tstart.	[m^3]
`water_balance_Qext_in_2D`	Cumulative inflowing volume through 2D external discharge since Tstart.	[m^3]
`water_balance_Qext_out_2D`	Cumulative outflowing volume through 2D external discharge since Tstart.	[m^3]
`water_balance_Qext_total_2D`	Total net inflowing volume through 2D external discharge since Tstart.	[m^3]
`water_balance_total_volume_interception`	Total volume of the interception layer at the end of timestep.	[m^3]
`water_balance_evaporation_interception`	Total evaporated volume from interception layer since Tstart.	[m^3]
`water_balance_precipitation_on_ground`	Total volume of rain fallen onto the ground (i.e., not intercepted) since Tstart.	[m^3]

F.3.1.7 Variables on laterals

The history file may also contain variables on laterals. This is controlled by the MDU setting `[output] Wrihis_lateral`. These variables are listed in Table F.7.

Table F.7: *Output quantities for laterals.*

Quantity name	Description	Unit
`lateral_id`	Id of lateral.	
`lateral_prescribed_↩ discharge_instantaneous`	Prescribed discharge through lateral at current time step (instantaneous).	[m^3/s]
`lateral_prescribed_↩ discharge_average`	Prescribed discharge through lateral, average over the last history time interval.	[m^3/s]
`lateral_realized_↩ discharge_instantaneous`	Realized discharge through lateral at current time step (instantaneous).	[m^3/s]
`lateral_realized_↩ discharge_average`	Realized discharge through lateral, average over the last history time interval.	[m^3/s]
`lateral_geom`, `lateral_geom_node_count`, `lateral_geom_node_coordx`, `lateral_geom_node_coordy`	Geometry variables of laterals, see Section F.3.1.10.	

F.3.1.8 Variables on sources/sinks

Variables on sources/sinks are typically defined as follows:

```
double source_sink_prescribed_discharge(time, source_sink) ;
source_sink_prescribed_discharge:units = "m3 s-1" ;
source_sink_prescribed_discharge:coordinates = "source_sink_name" ;
```

Other quantities that can be written are:

- source sink prescribed salinity increment;
- source sink prescribed temperature increment;
- source sink current discharge;
- source sink cumulative volume;
- source sink discharge average;
- geometry variables, see Section F.3.1.10.

These variables are automatically written to his-file if any source/sink exists. One can switch it off by setting in MDU-file that `[output] Wrihis_sourcesink = 0`.

F.3.1.9 Hydraulic structure output

A history file can also contain automatic output for hydraulic structures, without the need to add an explicit cross section at the same location. An example variable is:

```
 double weirgen_discharge(time, weirgens) ;
   weirgen_discharge:long_name = "Discharge through weir" ;
```

Other quantities that can be written are:

- weir crest level at current time;
- Total discharge through gate at current time;

- gate sill level at current time;
- gate sill width at current time;
- gate door lower edge level at current time;
- gate door opening width at current time;
- Total pump discharge at current time;
- Max. pump capacity at current time;
- Geometry variables, see Section F.3.1.10.

Quantities for different hydraulic structures are described in the following subsections.

F.3.1.9.1 Pump

Output for pump structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_pump (cf. Table A.1). The available output quantities are listed in Table F.8.

Table F.8: *Output quantities of pumps in history file.*

Quantity name	Description	Unit
pump_id	Id of the pump.	
pump_xmid, pump_ymid	x- and y-coordinates of representative mid point of the pump location (snapped polyline).	[m]
pump_structure_discharge*	Discharge through the pump.	[m^3/s]
pump_capacity	Prescribed capacity of the pump.	[m^3/s]
pump_discharge_dir*	Discharge w.r.t. pump orientation.	[m^3/s]
pump_s1up	Water level at upstream of the pump.	[m AD]
pump_s1dn	Water level at downstream of the pump.	[m AD]
pump_s1_delivery_side	Water level at delivery side of pump.	[m AD]
pump_s1_suction_side	Water level at suction side of pump.	[m AD]
pump_structure_head	Head difference across the pump structure, this is computed by pump_s1up subtracting pump_s1dn.	[m]
pump_head	Head difference in pumping direction, this is computed by pump_s1_delivery_side subtracting pump_s1_suction_side.	[m]
pump_actual_stage	Actual stage of the pump. It gives valid values only when number of stages is larger than 0.	
pump_reduction_factor	Reduction factor of the pump.	

Remarks:

- Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the pump covers more than one flow link, variable pump_structure_discharge is summed across all wet links.
- pump_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to pump_s1dn considering downstream cells. Moreover, pump_s1_delivery_side/pump_s1_suction_side also follows the similar rule, depending on if its delivery/suction side is wet or not, respectively.
- pump_structure_head and pump_head have a valid value only when both sides are wet. Otherwise it shows a fill value (-999). In the situation when there is more than one upstream/-downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.

- ◇ The pump_capacity does *not* include any reduction factor yet. The actual discharge is available in pump_discharge_dir and only this variable includes a possible reduction factor, as well as volume-based relaxation (cf. Section 14.10.6).

F.3.1.9.2 General structure

Output for general structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_gen (cf. Table A.1). The available output quantities are listed in Table F.9.

Table F.9: *Output quantities of general structures in history file.*

Quantity name	Description	Unit
general_structure_id	Id of the general structure.	
general_structure_discharge*	Total discharge through the general structure.	[m^3/s]
general_structure_crest_level	Actual crest level that is used in the computation.	[m AD]
general_structure_crest_width	Actual crest width that is used in the computation.	[m]
general_structure_gate_lower_edge_level	Actual gate lower edge level that is used in the computation.	[m AD]
general_structure_gate_opening_width	Actual gate opening width that is used in the computation.	[m]
general_structure_s1up	Water level at upstream of the general structure.	[m AD]
general_structure_s1dn	Water level at downstream of the general structure.	[m AD]
general_structure_discharge_through_gate_opening*	Discharge through gate opening.	[m^3/s]
general_structure_discharge_over_gate*	Discharge over gate.	[m^3/s]
general_structure_discharge_under_gate*	Discharge under gate.	[m^3/s]
general_structure_gate_opening_height	Gate opening height.	[m]
general_structure_gate_upper_edge_level	Gate upper edge level.	[m AD]
general_structure_head	Head difference across the general structure.	[m]
general_structure_velocity_through_gate_opening*	Velocity through gate opening.	[m/s]
general_structure_velocity_over_gate*	Velocity over gate.	[m/s]
general_structure_velocity_under_ gate*	Velocity under gate.	[m/s]
general_structure_flow_area*	Total flow area.	[m^2]
general_structure_flow_area_in_gate_opening*	Flow area in gate opening.	[m^2]
general_structure_flow_area_over_gate*	Flow area over gate.	[m^2]
general_structure_flow_area_under_gate*	Flow area under gate.	[m^2]

(continued on next page)

Quantity name	Description	Unit
(continued from previous page)		
general_structure_state*	Flow state through the general structure. Possible values: ◇ 0: No flow ◇ 1: free weir flow ◇ 2: submerged weir flow ◇ 3: free gate flow ◇ 4: submerged gate flow	
general_structure_s1_on_crest*	Water level on crest.	[m AD]
general_structure_velocity*	Average velocity through the general structure.	[m/s]
general_structure_force_difference*	Force difference per unit width at the general structure.	[N/m]

Remarks:

- ◇ Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the general structure covers more than one flow link, the written variables are either summed across all wet links (e.g. discharges and areas), or a weighted average, with the weights being the widths of the wet flow links (e.g. water level on crest and force difference per unit width). Specially, velocity is the summed discharge divided by the summed area on wet links.
- ◇ general_structure_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to general_structure_s1dn considering downstream cells.
- ◇ general_structure_head has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, general_structure_head has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- ◇ general_structure_state shows a fill value (-999) if the flow link is dry.
  If the flow link is wet, on this flow link, three states of the flow are computed: state of flow under gate, state of flow between gate and state of flow over gate. The maximal value of these integers will be used as the flow state of the link. When the general structure covers more than one flow link, then the state may differ per flow link. If not all of the flow links have the same state value, the state is written to file as a fill value (-999).

F.3.1.9.3 Weir

Output for weir structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_weir (cf. Table A.1). The available output quantities are listed in Table F.10.

Table F.10: *Output quantities of weirs in history file.*

Quantity name	Description	Unit
weirgen_id	Id of the weir.	
weirgen_discharge*	Discharge through the weir.	[m^3/s]
weirgen_crest_level	Actual crest level that is used in the computation.	[m AD]
weirgen_crest_width	Actual crest width that is used in the computation.	[m]
(continued on next page)		

Quantity name	Description	Unit
(continued from previous page)		
weirgen_s1up	Water level at upstream of the weir.	[m AD]
weirgen_s1dn	Water level at downstream of the weir.	[m AD]
weirgen_structure_head	Head difference across the weir.	[m]
weirgen_flow_area*	Flow area at the weir.	[m^2]
weirgen_velocity*	Velocity through the weir.	[m/s]
weirgen_state*	Flow state at the weir. Possible values: ◇ 0: No flow ◇ 1: free weir flow ◇ 2: submerged weir flow	
weirgen_force_difference*	Force difference per unit width at the weir.	[N/m]
weirgen_s1_on_crest*	Water level on crest of the weir.	[m AD]

Remarks:

- Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the weir covers more than one flow link, the written variables are either summed across all wet links (e.g. discharge and area), or a weighted average, with the weights being the widths of the wet flow links (e.g. water level on crest and force difference per unit width). Specially, velocity is the summed discharge divided by the summed area on wet links.
- weirgen_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to weirgen_s1dn considering downstream cells.
- weirgen_structure_head has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, weirgen_structure_head has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- weirgen_state shows a fill value (-999) if the flow link is dry. When the weir covers more than one wet flow link, then the state may differ per flow link. If not all of the flow links have the same state value, the state is written to file as a fill value (-999).

F.3.1.9.4 Orifice

Output for orifice structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_orifice (cf. Table A.1). The available output quantities are listed in Table F.11.

***Table F.11:** Output quantities of orifices in history file.*

Quantity name	Description	Unit
orifice_id	Id of the orifice.	
orifice_discharge*	Discharge through the orifice.	[m^3/s]
orifice_crest_level	Actual crest level that is used in the computation.	[m AD]
orifice_crest_width	Actual crest width that is used in the computation.	[m]
orifice_gate_lower_edge_level	Actual gate lower edge level that is used in the computation.	[m AD]
(continued on next page)		

(continued from previous page)

Quantity name	Description	Unit
orifice_s1up	Water level at upstream of the orifice.	[m AD]
orifice_s1dn	Water level at downstream of the orifice.	[m AD]
orifice_gate_opening_height	Gate opening height of the orifice.	[m]
orifice_head	Head difference across the orifice.	[m]
orifice_flow_area*	Flow area at the orifice.	[m^2]
orifice_state*	Flow state at the orifice. Possible values: ◇ 0: No flow ◇ 1: free weir flow ◇ 2: submerged weir flow ◇ 3: free gate flow ◇ 4: submerged gate flow	
orifice_velocity*	Velocity through the orifice.	[m/s]
orifice_s1_on_crest*	Water level on crest of the orifice.	[m AD]
orifice_force_difference*	Force difference per unit width at the orifice.	[N/m]

Remarks:

- Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the orifice covers more than one flow link, the written variables are either summed across all wet links (e.g. discharge and area), or a weighted average, with the weights being the widths of the wet flow links (e.g. water level on crest and force difference per unit width). Specially, velocity is the summed discharge divided by the summed area on wet links.
- orifice_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to orifice_s1dn considering downstream cells.
- orifice_head has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, orifice_head has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- orifice_state shows a fill value (-999) if the flow link is dry. When the orifice covers more than one wet flow link, then the state may differ per flow link. If not all of the flow links have the same state value, the state is written to file as a fill value (-999).

F.3.1.9.5 Bridge

Output for bridge structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_bridge (cf. Table A.1). The available output quantities are listed in Table F.12.

Table F.12: *Output quantities of bridges in history file.*

Quantity name	Description	Unit
bridge_id	Id of the bridge.	
bridge_discharge*	Discharge through the bridge.	[m^3/s]
bridge_s1up	Water level at upstream of the bridge.	[m AD]
bridge_s1dn	Water level at downstream of the bridge.	[m AD]

(continued on next page)

Quantity name	Description	Unit
(continued from previous page)		
bridge_head	Head difference across the bridge.	[m]
bridge_flow_area*	Flow area at the bridge.	[m^2]
bridge_velocity*	Velocity through the bridge.	[m/s]
bridge_blup	Bed level at upstream of the bridge.	[m AD]
bridge_bldn	Bed level at downstream of the bridge.	[m AD]
bridge_bl_actual	Actual bed level of bridge (crest) that is used in the computation.	[m AD]

Remarks:

- Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the bridge covers more than one flow link, the written variables are summed across all wet links (e.g. discharge and area). Specially, velocity is the summed discharge divided by the summed area on wet links.
- bridge_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to bridge_s1dn considering downstream cells.
- bridge_head has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, bridge_head has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- When the bridge covers more than one flow link, variables of bed level (with bl) are weighted averages, with the weights being the widths of the flow links (both dry and wet).
- The actual bed level of the bridge is a crest-like bed level, i.e., at the velocity point.

F.3.1.9.6 Culvert

Output for culvert structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_culvert (cf. Table A.1). The available output quantities are listed in Table F.13.

Table F.13: *Output quantities of culverts in history file.*

Quantity name	Description	Unit
culvert_id	Id of the culvert.	
culvert_discharge*	Discharge through the culvert.	[m^3/s]
culvert_crest_level	Crest level.	[m AD]
culvert_gate_lower_edge_level	Gate lower edge level.	[m AD]
culvert_s1up	Water level at upstream of the culvert.	[m AD]
culvert_s1dn	Water level at downstream of the culvert.	[m AD]
culvert_gate_opening_height	Gate opening height of the culvert.	[m]
culvert_head	Head difference across the culvert.	[m]
culvert_flow_area*	Flow area at the culvert.	[m^2]
(continued on next page)		

Quantity name	Description	Unit
(continued from previous page)		
culvert_state*	Flow state at the culvert. Possible values: ◇ 0: No flow ◇ 1: free culvert flow ◇ 2: submerged culvert flow	
culvert_velocity*	Velocity through the culvert.	[m/s]

Remarks:

- ◇ Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the culvert covers more than one flow link, the written variables are summed across all wet links (e.g. discharge and area). Specially, velocity is the summed discharge divided by the summed area on wet links.
- ◇ culvert_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to culvert_s1dn considering downstream cells.
- ◇ culvert_head has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, culvert_head has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- ◇ culvert_state shows a fill value (-999) if the flow link is dry. When the culvert covers more than one wet flow link, then the state may differ per flow link. If not all of the flow links have the same state value, the state is written to file as a fill value (-999).

F.3.1.9.7 Long culvert

Output for long culvert structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_longculvert (cf. Table A.1). The available output quantities are listed in Table F.14.

Table F.14: *Output quantities of long culverts in history file.*

Quantity name	Description	Unit
longculvert_id	Id of the long culvert.	
longculvert_discharge*	Discharge through the long culvert.	[m^3/s]
longculvert_s1up	Water level at upstream of the long culvert.	[m AD]
longculvert_s1dn	Water level at downstream of the long culvert.	[m AD]
longculvert_head	Head difference across the long culvert.	[m]
longculvert_flow_area*	Flow area at the long culvert.	[m^2]
longculvert_velocity*	Velocity through the long culvert.	[m/s]
longculvert_valve_ relative_opening*	Valve relative opening in long culvert.	[1]

Remarks:

- ◇ Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the long culvert covers more than one flow link, the written variables are summed across all wet links (e.g. discharge and area). Specially, velocity is the summed discharge divided by the summed area on wet links.

- longculvert_s1up shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to longculvert_s1dn considering downstream cells.
- longculvert_head has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, longculvert_head has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- In the case of a long culvert that consists of more than one wet flow link (refer to Figure 14.10):
  - the discharge is used from the first flow link, i.e., the inlet side.
  - the upstream and downstream water levels are taken from the 2D end points of the long culvert structure. No output on any intermediate pressure points is written in the his file. Those can be found in the map file.

F.3.1.9.8 Dam break

Output for dam break structures is available as time series in the history file. The output can be switched on or off using the MDU setting [output] Wrihis_structure_damBreak (cf. Table A.1). The available output quantities are listed in Table F.15.

***Table F.15:** Output quantities of dam breaks in history file.*

Quantity name	Description	Unit
dambreak_id	Id of the dam break.	
dambreak_s1up	Water level at upstream of the dam break.	[m AD]
dambreak_s1dn	Water level at downstream of the dam break.	[m AD]
dambreak_discharge*	Discharge through the dam break.	[m^3/s]
dambreak_cumulative_discharge*	Cumulative discharge through the dam break.	[m^3/s]
dambreak_breach_width_time_derivative	Breach width time derivative of the dam break.	[m/s]
dambreak_water_level_jump	Breach water level jump across the dam break. See remark below.	[m]
dambreak_normal_velocity	Normal velocity through the dam break.	[m/s]
dambreak_structure_head	Head difference across the dam break.	[m]
dambreak_flow_area*	Flow area at the dam break.	[m^2]
dambreak_crest_level	Actual crest level that is used in the computation. When there is no breach yet, the crest level is the bed level ("bob") at the breach starting location.	[m AD]
dambreak_crest_width	Actual crest width that is used in the computation. When there is no breach yet, equal to 0.	[m]

Remarks:

- Contrary to all other hydraulic structures, the dam break output only includes flow through the breach hole. In a 2D model, when multiple flow links are crossed by the dambreak polyline, there may additionally be flow across the fixed weir crest levels of flow links even though the breach growth has not reached them yet. When the total discharge across the entire polyline is desired, it is recommended to include an additional observation cross section (Section F.2.4) in the model.

- ◇ In a similar way as in the previous point, water levels and other quantities are only averaged or summed over the flow links in the breach hole.
- ◇ Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the dambreak covers more than one flow link, the written variables are summed across all wet links (e.g. discharge and area).
- ◇ `dambreak_s1up` shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to `dambreak_s1dn` considering downstream cells.
- ◇ `dambreak_structure_head` has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, `dambreak_structure_head` has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.
- ◇ The quantities `dambreak_structure_head` and `dambreak_water_level_jump` are different. The head is the basic difference between up- and downstream water levels. The water level jump is equal to $\max(0, \zeta_{up} - z_s) - \max(0, \zeta_{dn} - z_s)$ and represents the average water level jump at the breach crest, often used in breach growth formulas.

F.3.1.9.9 Universal weir

Output for universal weir structures is available as time series in the history file. The output can be switched on or off using the MDU setting `[output] Wrihis_structure_uniWeir` (cf. Table A.1). The available output quantities are listed in Table F.16.

Table F.16: *Output quantities of universal weirs in history file.*

Quantity name	Description	Unit
`uniweir_id`	Id of the universal weir.	
`uniweir_discharge`*	Discharge through the universal weir.	[m^3/s]
`uniweir_s1up`	Water level at upstream of the universal weir.	[m AD]
`uniweir_s1dn`	Water level at downstream of the universal weir.	[m AD]
`uniweir_head`	Head difference across the universal weir.	[m]
`uniweir_flow_area`*	Flow area at the universal weir.	[m^2]
`uniweir_velocity`*	Velocity through the universal weir.	[m/s]
`uniweir_crest_level`	Crest level of the universal weir.	[m AD]

Remarks:

- ◇ Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the universal weir covers more than one flow link, the written variables are summed across all wet links (e.g. discharge and area). Specially, velocity is the summed discharge divided by the summed area on wet links.
- ◇ `uniweir_s1up` shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to `uniweir_s1dn` considering downstream cells.
- ◇ `uniweir_head` has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, `uniweir_head` has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.

F.3.1.9.10 Compound structure

Output for compound structures is available as time series in the history file. The output can be switched on or off using the MDU setting `[output] Wrihis_structure_compound` (cf. Table A.1). The available output quantities are listed in Table F.17.

***Table F.17:** Output quantities of compound structures in history file.*

Quantity name	Description	Unit
`cmpstru_id`	Id of the compound structure.	
`cmpstru_discharge`*	Total discharge through the compound structure.	[m^3/s]
`cmpstru_s1up`	Water level at upstream of the compound structure.	[m AD]
`cmpstru_s1dn`	Water level at downstream of the compound structure.	[m AD]
`cmpstru_head`	Head difference across the compound structure.	[m]
`cmpstru_flow_area`*	Total flow area at the compound structure.	[m^2]
`cmpstru_velocity`*	Average velocity through the compound structure.	[m/s]

Remarks:

- Variables with notation $*$ show a fill value (-999) if the flow link is dry. When the compound structure covers more than one flow link, the written variables are summed across all wet links (e.g. discharge and area). Specially, velocity is the summed discharge divided by the summed area on wet links.
- `cmpstru_s1up` shows a fill value (-999) when the upstream cell is dry. If there is more than one upstream cell, then the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream cells are wet. This also applies to `cmpstru_s1dn` considering downstream cells.
- `cmpstru_head` has a valid value only when both upstream and downstream cells are wet. Otherwise it shows a fill value (-999). This means that, when the upstream cell is dry while downstream cell is wet, `cmpstru_head` has a fill value (-999). In the situation when there is more than one upstream/downstream cell, the value is a weighted average, with the weights being the widths of the flow links whose corresponding upstream and downstream cells are both wet.

F.3.1.9.11 Summary

Table F.18 summarizes the above output quantities for different hydraulic structures, where the existence of output is denoted by ×.

***Table F.18:** Summary of output in history files of quantities for different structures.*

Quantity name	Pump	General structure	Weir	Orifice	Bridge	Culvert	Dam break	Universal Weir	Compound structure	Long culvert
Id of structure	×	×	×	×	×	×	×	×	×	×
Total discharge through structure	×	×	×	×	×	×	×	×	×	×
Discharge through gate opening		×								
Discharge over gate upper edge level		×								
Water level at upstream of the structure	×	×	×	×	×	×	×	×	×	×

(continued on next page)

Quantity name	Pump	General structure	Weir	Orifice	Bridge	Culvert	Dam break	Universal Weir	Compound structure	Long culvert
(continued from previous page)										
Water level at downstream of the structure	×	×	×	×	×	×	×	×	×	×
Crest level of structure		×	×	×		×	×	×		
Gate opening height of structure		×		×		×				
Gate lower edge level of structure		×		×		×				
Gate upper edge level of structure		×								
Crest width of structure		×	×	×			×			
Force difference per unit width		×	×	×						
Water level on crest of structure		×	×	×						
Head difference across structure	×	×	×	×	×	×	×	×	×	×
Velocity through structure		×	×	×	×	×		×	×	×
Velocity through gate opening		×								
Velocity over gate upper edge level		×								
Flow area at structure		×	×	×	×	×	×	×	×	×
Flow area in gate opening		×								
Flow area above upper edge level		×								
Flow state at structure		×	×	×		×				
Actual stage of pump	×									
Capacity of pump	×									
Discharge w.r.t. pump orientation	×									
Head difference in pumping direction	×									
Reduction factor of pump	×									
Water level at delivery side of pump	×									
Water level at suction side of pump	×									
Coordinates of representative of pump	×									
Gate opening width		×								
Cumulative discharge through dambreak							×			
Breach width time derivative of dambreak							×			
Breach water level jump of dambreak							×			
Normal velocity through dambreak							×			
Bed level at upstream of bridge					×					
Bed level at downstream of bridge					×					
Actual bed level of bridge					×					
Valve relative opening										×
Geometry variables	×	×	×	×	×	×		×		×

F.3.1.10 Geometry variables

Geometry variables[2], that are also written to the his-files, give the geometry information for those locations for which time series are reported in the his-file. NOTE: the location information of geometry variables does not give the original input location of the geometries, but rather the locations that are snapped to the computational mesh.

[2] For more details about geometry variables please see: https://github.com/cf-convention/cf-conventions/blob/main/ch07.adoc#geometries

For a set of geometries, there are four geometry variables. Take a set of general structures as an example. The geometry variables names contain "_geom" for readability, but this is not required, and they should be identified via CF-compliant attributes. For the set of general structures these are:

- "general_structure_geom",
- "general_structure_geom_node_count",
- "general_structure_geom_node_coordx",
- "general_structure_geom_node_coordy".

The description of the above geometry variables are as follows:

- "general_structure_geom" is a geometry container variable. Such a variable can typically be identified when it is referenced in the `:geometry` attribute of another data variable, and the container variable itself has the `:geometry_type` attribute. This variable contains all the attributes that describe this set of geometries, including the following attributes:
    - "geometry_type": tells the type of the geometry. It can be "point", "line" or "polygon".
    - "node_count": tells the name of a variable that indicates the count of nodes per geometry. The name is "general_structure_geom_node_count" in our example, and it will contain the number of geometry points for each of the general structures in the model input.
    - "node_coordinates": contains the names of variables that contain the node coordinates of the geometry. In our example, it has two names "general_structure_geom_node_coordx" and "general_structure_geom_node_coordy".
- "_node_count": indicates the total number of nodes on this set of geometries. For instance, if variable "general_structure_geom_node_count" contains values "2, 3", it means that there are two general structures, and the first and the second general structures have 2 and 3 nodes, respectively.
- "_node_coordx": gives x-coordinates of all the nodes. Take the above example, then this array will contain 5 points in total; the first 2 values of this variable are the x-coordinates of the 2 nodes of the first general structure, and next 3 values of this variable are the x-coordinates of the 3 nodes of the second general structure.
- "_node_coordy": gives y-coordinates of all the nodes.

The following geometries in his files have geometry variables:

- Observation points.
- Observation cross sections.
- Hydraulic structures, including:
    - Weirs.
    - Orifices.
    - General structures.
    - Universal weirs.
    - Culverts.
    - Gates.
    - Pumps.
    - Bridges.
    - Long culverts.
- Source and sinks.
- Laterals.

F.3.2 Spatial fields as netCDF map-file

The map file <*mdu_name*_map.nc> contains data on the entire model grid in 1D, 2D and 3D. Three formats are currently supported, selected by the `MapFormat` keyword in the mdu-file:

```
[output]
MapFormat = 4  # Map file format, 1: netCDF, 2: Tecplot,
               # 3: netCFD and Tecplot, 4: netCDF-UGRID
```

D-Flow FM is using the more standardized UGRID[3] format. This is the default option. D-Flow FM GUI and Delft3D-QUICKPLOT also support this format. In the examples below the older netCDF format is still shown, but most of the concepts discussed hereafter apply equally well to UGRID output (4), only the variable names will be slightly different.

For parallel runs, each process writes a separate file for each partition as <*mdu_name*_000X_map.nc>.

The map file contains the dimensions and variable definitions in its header, followed by the actual data.

Dimensions

The following dimensions are defined in a map file header:

```
netcdf weirfree_map {
dimensions:
       nNetNode = 310 ;
       nNetLink = 486 ;
       nNetLinkPts = 2 ;
       nBndLink = 252 ;
       nNetElem = 180 ;
       nNetElemMaxNode = 4 ;
       nNetLinkContourPts = 4 ;
       nFlowElem = 180 ;        // Nr.
of grid cells
       nFlowElemMaxNode = 4 ;
       nFlowElemContourPts = 4 ;
       nFlowLink = 246 ;        // Nr.
of flow links (open cell edges)
       nFlowLinkPts = 2 ;
       time = UNLIMITED ; // (64 currently)
```

For plotting solution data, the grid cells (flow nodes) are important, and the flow links (open cell edges). All `Net*` dimensions relate the flow grid tot the original unstructured network. More explanation of these topological naming conventions can be found in section 8.3.1.

Location variables

The location and shape of grid cells is stored in several x,y variables:

```
 double FlowElem_xcc(nFlowElem) ;
   FlowElem_xcc:units = "m" ;
   FlowElem_xcc:standard_name = "projection_x_coordinate" ;
   FlowElem_xcc:long_name = "Flow element circumcenter x" ;
   FlowElem_xcc:bounds = "FlowElemContour_x" ;
 double FlowElem_ycc(nFlowElem) ;
   FlowElem_ycc:units = "m" ;
   FlowElem_ycc:standard_name = "projection_y_coordinate" ;
   FlowElem_ycc:long_name = "Flow element circumcenter y" ;
   FlowElem_ycc:bounds = "FlowElemContour_y" ;
// [..]
```

[3] https://github.com/ugrid-conventions/ugrid-conventions

```
 double FlowElemContour_x(nFlowElem, nFlowElemContourPts) ;
   FlowElemContour_x:units = "m" ;
   FlowElemContour_x:standard_name = "projection_x_coordinate" ;
   FlowElemContour_x:long_name = "List of x-points forming flow element" ;
   FlowElemContour_x:_FillValue = -999.
;
// [..]
 double FlowElem_bl(nFlowElem) ;
   FlowElem_bl:units = "m" ;
   FlowElem_bl:positive = "up" ;
   FlowElem_bl:standard_name = "sea_floor_depth" ;
   FlowElem_bl:long_name = "Bottom level at flow element\'s circumcenter." ;
```

In 3D output the map file also contains the vertical dimensions `laydim` for the layer centers and `wdim` for the layer interfaces. Corresponding to these dimensions, vertical coordinates `LayCoord_cc` and `LayCoord_w`, respectively are added:

```
double LayCoord_cc(laydim) ;
   LayCoord_cc:standard_name = "" ;
   LayCoord_cc:long_name = "z layer coordinate at flow element center" ;
   LayCoord_cc:positive = "up" ;
   LayCoord_cc:units = "m" ;
double LayCoord_w(wdim) ;
   LayCoord_w:standard_name = "" ;
   LayCoord_w:long_name = "z layer coordinate at vertical interface" ;
   LayCoord_w:positive = "up" ;
   LayCoord_w:units = "m" ;
```

These variable contain time- and space-independent layer positions (either in absolute z, or relative sigma coordinates).

In the case the FullGridOutput option is active (`FullGridOutput = 1`), these vertical coordinates `LayCoord_cc` and `LayCoord_w` are excluded from the map-file and replaced by variables `FlowElem_zcc` and `FlowElem_zw`. These variables contain the vertical positions of layer centers and interfaces, respectively, written for each flow cell and each timestep separately. Currently, this is only the case for the flow cell (centre) positions and not for the horizontal flow link positions:

```
double FlowElem_zcc(time, nFlowElem, laydim) ;
   FlowElem_zcc:coordinates = "FlowElem_xcc FlowElem_ycc FlowElem_zcc" ;
   FlowElem_zcc:standard_name = "" ;
   FlowElem_zcc:long_name = "flow element center z" ;
   FlowElem_zcc:units = "m" ;
double FlowElem_zw(time, nFlowElem, wdim) ;
   FlowElem_zw:coordinates = "FlowElem_xcc FlowElem_ycc FlowElem_zw" ;
   FlowElem_zw:standard_name = "" ;
   FlowElem_zw:long_name = "flow element z at vertical interface" ;
   FlowElem_zw:units = "m" ;
```

Variables on grid cells/pressure points

Variables on grid cells (represented by the pressure point/cell circumcenter) are typically defined as follows:

```
 double s1(time, nFlowElem) ;
   s1:coordinates = "FlowElem_xcc FlowElem_ycc" ;
   s1:standard_name = "sea_surface_level" ;
   s1:long_name = "waterlevel" ;
   s1:units = "m" ;
   s1:grid_mapping = "projected_coordinate_system" ;
```

All quantities that are available in the map file are listed in Table F.19 and Table F.20. These quantities are output when the model contains the related object. For example, quantities about waves are output in map file only when the model contains wave computation.

Table F.19: *Output variables in map file of UGRID format. Column 'Location' shows the grid location: CN means cell corners. L means net links. S means pressure points, U means velocity points. S3D and U3D are the corresponding locations in 3D networks. Two more locations in 3D networks are: W, vertical velocity point on all layer interfaces, and WU, vertical viscosity point on all layer interfaces.*

Quantity name	MDU option	Unit	Location
Water level at end of time step.	`Wrimap_waterlevel_s1`	[m AD]	S
Water depth at pressure points.	`Wrimap_waterdepth`	[m]	S
Water level at start of time step.	`Wrimap_waterlevel_s0`	[m AD]	S
Numlimdt: number of times each cell was limiting the time step.	`Wrimap_numlimdt`	[–]	S
Temperature.	`Wrimap_temperature`	[°C]	S or S3D
Tracers and other constituents.	`Wrimap_constituents`	[-]	S or S3D
# The followings are hydrological quantities (Chapter 13):			
Rainfall intensity.	`Wrimap_rain`	[m/s]	S
Interception layer water depth.	`Wrimap_interception`	[m]	S
Potential evaporation.	`Wrimap_evaporation`	[m/s]	S
Actual evaporation.	`Wrimap_evaporation`	[m/s]	S
Prescribed evaporation rate.	`Wrimap_evaporation`	[m/s]	S
# The above are hydrological quantities.			
# The following variables are only for 1D network:			
Freeboard.	`Wrimap_freeboard`	[m]	S
Waterdepth on ground.	`Wrimap_waterdepth_on_ground`	[m]	S
Volume on ground.	`Wrimap_volume_on_groundn`	[m^3]	S
Cumulative time at each node when water is above ground level.	`Wrimap_time_water_on_ground`	[s]	S
Current total 1D2D net inflow.	`Wrimap_total_net_inflow_1d2d`	[m^3/s]	S
Cumulative total 1D2D net inflow.	`Wrimap_total_net_inflow_1d2d`	[m^3]	S
Current total lateral net inflow.	`Wrimap_total_net_inflow_lateral`	[m^3/s]	S
Cumulative total lateral net inflow.	`Wrimap_total_net_inflow_lateral`	[m^3]	S
Water level gradient.	`Wrimap_water_level_gradient`	[1]	U
# The above variables are only for 1D network.			

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
# The following variables are only for 3D models:			
Vertical coordinate of layer centres at pressure points.	`FullGridOutput`	[m]	S3D
Vertical coordinate of layer interfaces at pressure points.	`FullGridOutput`	[m]	W
Bounds of vertical coordinate of layers at pressure points.	`FullGridOutput`	[m]	S3D
Vertical coordinate of layer centres at velocity points.	`FullGridOutput`	[m]	U3D
Bounds of vertical coordinate of layers at velocity points.	`FullGridOutput`	[m]	U3D
Turbulent kinetic energy, if `[numerics] Turbulencemodel >= 3`.	`Wrimap_turbulence`	[m^2/s^2]	WU
Turbulent vertical eddy viscosity, if `[numerics] Turbulencemodel >= 3`.	`Wrimap_turbulence`	[m^2/s]	WU
Turbulent energy dissipation, if `[numerics] Turbulencemodel = 3`.	`Wrimap_turbulence`	[m^2/s^3]	WU
Turbulent time scale, if `[numerics] Turbulencemodel = 4`.	`Wrimap_turbulence`	[1/s]	WU
Flow element center velocity vector, z-component.	`Wrimap_velocity_vector`	[m/s]	S3D
Flow element center GLM depth-averaged velocity if `[output] EulerVelocities = 1` and `[waves] Wavemodelnr > 0`. Otherwise, flow element center depth-averaged velocity. This includes x- and y-components.	`Wrimap_velocity_vector`	[m/s]	S
Flow element center depth-averaged GLM velocity magnitude if `[output] EulerVelocities = 1` and `[waves] Wavemodelnr > 0`. Otherwise, flow element center depth-averaged velocity magnitude.	`Wrimap_velocity_magnitude`	[m/s]	S
Upward velocity on vertical interface, n-component.	`Wrimap_upward_velocity_component`	[m/s]	W
Flow element center mass density.	`Wrimap_density_rho`	[kg/m^3]	S3D
# The above variables are only for 3D models.			
Sum of all water influx.	`Wrimap_Qin`	[m^3/s]	S
Volume of water in grid cell.	`Wrimap_volume1`	[m^3]	S or S3D
Time step per cell based on CFL.	`Wrimap_DTcell`	[s]	S or S3D
Water depth at velocity points.	`Wrimap_waterdepth_hu`	[m]	U
Wet flow area between two neighbouring grid cells.	`Wrimap_flowarea_au`	[m^2]	U or U3D
Velocity at velocity points, n-component.	`Wrimap_velocity_component_u1`	[m/s]	U or U3D

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Velocity at velocity points at previous time step, n-component.	`Wrimap_velocity_component_u0`	[m/s]	U or U3D
Flow element center eulerian velocity vector if `[output] EulerVelocities = 1` and `[waves] Wavemodelnr > 0` and `[waves] flowWithoutWaves` is false. Otherwise, flow element center velocity vector. This includes x- and y-components.	`Wrimap_velocity_vector`	[m/s]	S or S3D
Flow element center eulerian velocity magnitude if `[output] EulerVelocities = 1` and `[waves] Wavemodelnr > 0` and `[waves] flowWithoutWaves` is false. Otherwise, flow element center velocity magnitude.	`Wrimap_velocity_magnitude`	[m/s]	S or S3D
Flow element center eulerian velocity vector based on discharge if `[output] EulerVelocities = 1` and `[waves] Wavemodelnr > 0` and `[waves] flowWithoutWaves` is false. Otherwise, flow element center velocity vector based on discharge. This includes x- and y-components.	`Wrimap_velocity_vectorq`	[m/s]	S or S3D
Discharge through flow link at current time.	`Wrimap_flow_flux_q1`	[m^3/s]	U or U3D
Main channel discharge through flow link at current time.	`Wrimap_flow_flux_q1_main`	[m^3/s]	U or U3D
Horizontal eddy viscosity.	`Wrimap_horizontal_viscosity_viu`	[m^2/s]	U or U3D
Horizontal eddy diffusivity.	`Wrimap_horizontal_diffusivity_diu`	[m^2/s]	U or U3D
Total bed shear stress vector, including x- and y-components.	`Wrimap_taucurrent`	[N/m^2]	S
Total bed shear stress magnitude.	`Wrimap_taucurrent`	[N/m^2]	S
Bed shear stress magnitude for morphology if both MDU keywords `[sediment] SedFile` and `MorFile` are not empty, and `Sedimentmodelnr` is 4.	`Wrimap_taucurrent`	[N/m^2]	S
# The following variables are about roughness:			
Chézy roughness:			
Effective Chézy roughness in flow element center.	`Wrimap_chezy`	[m$^{0.5}$/s]	S
Effective Chézy roughness on flow links.	`Wrimap_chezy_on_flow_links`	[m$^{0.5}$/s]	U
Input roughness on flow links.	`Wrimap_input_roughness`	Depends on roughness type.	U
Input roughness type.	`Wrimap_input_roughness`	[-], see `UnifFrictType` in Table A.1.	U

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Energy loss for fixed weirs in case of Tabellenboek or Villemonte if `[numerics] FixedWeirScheme = 8 or 9`	`Wrimap_fixed_weir_energy_loss`	[m]	U or U3D
Salinity in flow element if `[physics] Salinity > 0.`	`Wrimap_salinity`	[ppt]	S or S3D
# The following variables are about streamline curvature and spiral flow intensity (section 8.6):			
(For non-3D models only) Flow streamline curvature if `[physics] SecondaryFlow` is positive.	`Wrimap_spiral_flow`	[1/m]	S
Spiral flow intensity if `[physics] SecondaryFlow > 0.`	`Wrimap_spiral_flow`	[m/s]	S
# The above variables are about streamline curvature and spiral flow intensity.			
All water quality bottom variables in flow element (3D if `[processes] Wriwaqbot3Doutput = 1`).		[depends]	S or S3D
Extra water quality output.		[depends]	S or S3D
All water quality statistic variables.		[depends]	S or S3D
Water quality mass balance areas.		[-]	S
Atmospheric pressure near surface.	`Wrimap_wind`	[N/m^2]	S
Wind velocity on flow element center, including x- and y-components.	`Wrimap_wind`	[m/s]	S
Wind velocity on flow links, including x- and y-components.	`Wrimap_wind`	[m/s]	U
Wind stress on flow element center, including x- and y-components.	`Wrimap_windstress`	[N/m^2]	S
# The following variables are about heat flux quantities, air temperature, relative humidity, cloudiness and more detailed fluxes (chapter 10):			
Air temperature near surface.	`Wrimap_heat_fluxes`	[°C]	S
Relative humidity near surface.	`Wrimap_heat_fluxes`	[-]	S
Cloudiness.	`Wrimap_heat_fluxes`	[1]	S
Solar influx.	`Wrimap_heat_fluxes`	[W/m^2]	S
Evaporative heat flux.	`Wrimap_heat_fluxes`	[W/m^2]	S
Sensible heat flux.	`Wrimap_heat_fluxes`	[W/m^2]	S
Long wave back radiation.	`Wrimap_heat_fluxes`	[W/m^2]	S
Free convection evaporative heat flux.	`Wrimap_heat_fluxes`	[W/m^2]	S
Free convection sensible heat flux.	`Wrimap_heat_fluxes`	[W/m^2]	S
Total heat flux.	`Wrimap_heat_fluxes`	[W/m^2]	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
# The above variables are about heat flux quantities, air temperature, relative humidity, cloudiness and more detailed fluxes.			
Time-varying bottom level in flow cell center, if `[sediment] Sedimentmodelnr > 0` and `[sediment] SedFile` and `MorFile` are not empty, and `Sedimentmodelnr` is 4, or if the EXT file contains `bedrock_surface_elevation`.	`Wrimap_sediment`	[m]	S
Cumulative subsidence/uplift, if the EXT file contains `bedrock_surface_elevation`.		[m]	S, U or CN
Current related roughness height.	`Wrimap_z0`	[m]	U
Current-wave related roughness height.	`Wrimap_z0`	[m]	U
# The following variables are for models with sediment (D-Morphology UM (2019)),			
# when `[sediment] Sedimentmodelnr > 0` and `[sediment] SedFile` and `MorFile` are not empty, `Sedimentmodelnr` is 4.			
Time interval over which cumulative transports are calculated, if MorFile keyword `[output] AverageAtEachOutputTime` is false.	`Wrimap_sediment`	[s]	-
Average morphological factor over elapsed morphological time.	`Wrimap_sediment`	[-]	-
Current morphological time.	`Wrimap_sediment`	[s]	-
Sediment fraction name.	`Wrimap_sediment`	[-]	-
Suspended sediment fraction name.	`Wrimap_sediment`	[-]	-
(For 3D models only) Bottom layer for sed calculations.	`Wrimap_sediment`	[-]	S
Sediment settling velocity.	`Wrimap_sediment`	[m/s]	W for 3D, otherwise S
(For non-3D models only) Equilibrium sediment concentration.	`Wrimap_sediment`	[kg/m^3]	S
Near-bed reference concentration height, if MorFile keyword `[Output] ReferenceHeight` is true.	`Wrimap_sediment`	[m]	S
Near-bed reference concentration, if MorFile keyword `[Output] ReferenceHeight` is true.	`Wrimap_sediment`	[kg/m^3]	S
Source term suspended sediment fractions, if MorFile keyword `[Output] SourceSinkTerms` is true.	`Wrimap_sediment`	[kg/m^3/s]	S
Sink term suspended sediment fractions, if MorFile keyword `[Output] SourceSinkTerms` is true.	`Wrimap_sediment`	[1/s]	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
(For 3D models only) Near-bed transport correction in face-normal direction, if MorFile keyword `[Output] NearBedTranspCorrAtFlux` is true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	U
Sediment concentration.	`Wrimap_sediment`	[kg/m^3]	S or S3D
Suspended load transport, including n- and t-components.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	U
Bed load transport, including n- and t-components.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	U
Bed slope, including n- and t-components, if MorFile keyword `[Output] Bedslope` is true.	`Wrimap_sediment`	[-]	U
Characteristic velocity in cell centre, including n- and t-components, if MorFile keyword `[Output] VelocAtZeta` is true.	`Wrimap_sediment`	[m/s]	S
Characteristic velocity magnitude, if MorFile keyword `[Output] VelocMagAtZeta` is true.	`Wrimap_sediment`	[m/s]	S
Height above bed for characteristic velocity, if MorFile keyword `[Output] VelocZAtZeta` is true.	`Wrimap_sediment`	[m]	S
Bed shear velocity, if MorFile keyword `[Output] ShearVeloc` is true.	`Wrimap_sediment`	[m/s]	S
Bed load transport due to currents, including x- and y-components, if MorFile keyword `[Output] BedTranspDueToCurrentsAtZeta` and `[Output] RawTransportsAtZeta` are true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Bed load transport due to currents (reconstructed), including x- and y-components, if MorFile keyword `[Output] BedTranspDueToCurrentsAtZeta` is true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Bed load transport due to waves, including x- and y-components, if MorFile keyword `[Output] BedTranspDueToWavesAtZeta` and `[Output] RawTransportsAtZeta` are true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Bed load transport due to waves (reconstructed), including x- and y-components, if MorFile keyword `[Output] BedTranspDueToWavesAtZeta` is true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Suspended load transport due to currents, including x- and y-components, if MorFile keyword `[Output] SuspTranspDueToCurrentsAtZeta` and `[Output] RawTransportsAtZeta` are true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Suspended load transport due to currents (reconstructed), including x- and y-components, if MorFile keyword `[Output] SuspTranspDueToCurrentsAtZeta` is true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Suspended load transport due to waves, including x- and y-components, if MorFile keyword `[Output] SuspTranspDueToWavesAtZeta` and `[Output] RawTransportsAtZeta` are true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Suspended load transport due to waves (reconstructed), including x- and y-components, if MorFile keyword `[Output] SuspTranspDueToWavesAtZeta` is true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Total sediment transport (reconstructed), including x- and y-components.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S
Time-averaged bed load, including x- and y-components, if MorFile keyword `[Output] AverageAtEachOutputTime` is true.	`Wrimap_sediment`	[kg/s/m] if MorFile keyword `[Output] TranspType` is 0, or [m^3/s/m] if MorFile keyword `[Output] TranspType` is 1 or 2.	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Time-averaged suspended load transport, including x- and y-components.	Wrimap_sediment	[kg/s/m] if MorFile keyword [Output] TranspType is 0, or [m^3/s/m] if MorFile keyword [Output] TranspType is 1 or 2.	S
Available sediment mass in the bed, if MorFile keyword [Underlayer] IUnderLyr = 1.	Wrimap_sediment	[kg/m^2]	S
Sediment thickness in the bed, if MorFile keyword [Underlayer] IUnderLyr = 1.	Wrimap_sediment	[m]	S
Available sediment mass in a layer of the bed, if MorFile keyword [Underlayer] IUnderLyr = 2.	Wrimap_sediment	[kg/m^2]	S
Thickness of a layer of the bed, if MorFile keyword [Underlayer] IUnderLyr = 2.	Wrimap_sediment	[m]	S
Porosity of a layer of the bed, if MorFile keyword [Underlayer] IUnderLyr = 2 and [Underlayer] IPorosity > 0.	Wrimap_sediment	[-]	S
Volume fraction in a layer of the bed, if MorFile keyword [Underlayer] IUnderLyr = 2.	Wrimap_sediment	[-]	S
Bed shear stress for morphology, if MorFile keyword [Output] Taub is true.	Wrimap_sediment	[N/m^2]	S
Excess bed shear ratio, if MorFile keyword [Output] taurat is true.	Wrimap_sediment	[-]	S
Arithmetic mean sediment diameter, if MorFile keyword [Output] Dm is true.	Wrimap_sediment	[m]	S
Geometric mean sediment diameter, if MorFile keyword [Output] Dg is true.	Wrimap_sediment	[m]	S
Geometric standard deviation of particle size mix, if MorFile keyword [Output] Dgsd is true.	Wrimap_sediment	[m]	S
All sediment diameter percentile, if MorFile keyword [Output] Percentiles is true.	Wrimap_sediment	[m]	S
Availability fraction in top layer, if MorFile keyword [Output] Frac is true.	Wrimap_sediment	[-]	S
Mud fraction in top layer, if MorFile keyword [Output] MudFrac is true.	Wrimap_sediment	[-]	S
Sand fraction in top layer, if MorFile keyword [Output] SandFrac is true.	Wrimap_sediment	[-]	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Reduction factor due to limited sediment thickness, if MorFile keyword `[Output] FixFac` is true.	`Wrimap_sediment`	[-]	S
Hiding and exposure factor, if MorFile keyword `[Output] HidExp` is true.	`Wrimap_sediment`	[-]	S
Sediment mass in fluff layer, if MorFile keyword `[FluffLayer] Type > 0`.	`Wrimap_sediment`	[kg/m^2]	S
(For 1D only) Time-varying cross-section points level, if MorFile keyword `[Morphology] BedUpd` is true.	`Wrimap_sediment`	[m]	S
(For 1D only) Time-varying cross-section points width, if MorFile keyword `[Morphology] BedUpd` is true.	`Wrimap_sediment`	[m]	S
(For 1D only) Name of cross-section, if MorFile keyword `[Morphology] BedUpd` is true.	`Wrimap_sediment`	[-]	S
(For 1D only) Main channel averaged bed level, if MorFile keyword `[Output] MainChannelAveragedBedLevel` is true.	`Wrimap_sediment`	[m]	S
(For 1D only) Main channel cell area, if MorFile keyword `[Output] MainChannelCellArea` is true.	`Wrimap_sediment`	[m^2]	S
(For 1D only) Main channel cell width, if MorFile keyword `[Output] MainChannelWidthAtFlux` is true.	`Wrimap_sediment`	[m]	U
# The above variables are for models with sediment (D-Morphology UM (2019)),			
# when `[sediment] Sedimentmodelnr > 0` and `[sediment] SedFile` and `MorFile` are not empty, `Sedimentmodelnr` is 4.			
# The following variables are for models with sediment,			
# when `[sediment] Sedimentmodelnr > 0`, and either `[sediment] SedFile` or `MorFile` is empty or `Sedimentmodelnr` is not 4.			
All sediment concentrations.	`Wrimap_sediment`	[kg/m^3]	S
All erodable layer thickness per size fraction in flow element centers,if MDU keyword `[sediment] Jaceneqtr = 1`.	`Wrimap_sediment`	[m]	S
Flow element center bedlevel (bl),if MDU keyword `[sediment] Jaceneqtr = 1`.	`Wrimap_sediment`	[m]	S
All erodable layer thickness per size fraction in flow element corners,if MDU keyword `[sediment] Jaceneqtr` is not 1.	`Wrimap_sediment`	[m]	CN
Flow element corner bedlevel (zk),if MDU keyword `[sediment] Jaceneqtr` is not 1.	`Wrimap_sediment`	[m]	CN

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
# The above variables are for models with sediment,			
# when `[sediment] Sedimentmodelnr > 0`, and either `[sediment] SedFile` or `MorFile` is empty or `Sedimentmodelnr` is not 4.			
# The following variables are for flow without waves, when `[waves] flowWithoutWaves` is true.			
RMS wave height, if EXT file contains `hrms` or `wavesignificantheight`, and MDU keyword `[waves] jamapsigwav = 0`.	`Wrimap_waves`	[m]	S
Significant wave height, if EXT file contains `hrms` or `wavesignificantheight`, and MDU keyword `[waves] jamapsigwav` is non-zero.	`Wrimap_waves`	[m]	S
Peak wave period, if EXT file contains `tp` or `tps` or `rtp` or `waveperiod`.	`Wrimap_waves`	[s]	S
Mean direction of wave propagation relative to ksi-dir. ccw, if EXT file contains `dir` or `wavedirection`.	`Wrimap_waves`	[deg]	S
Mean direction of wave propagation relative to ksi-dir. ccw, if EXT file contains `dir` or `wavedirection`.	`Wrimap_waves`	[deg]	S
Surface layer wave forcing term, x-component, if EXT file contains `fx`.	`Wrimap_waves`	[N/m^2]	S
Surface layer wave forcing term, y-component, if EXT file contains `fy`.	`Wrimap_waves`	[N/m^2]	S
Bottom layer wave forcing term, x-component, if EXT file contains `wsbu`.	`Wrimap_waves`	[N/m^2]	S
Bottom layer wave forcing term, y-component, if EXT file contains `wsbv`.	`Wrimap_waves`	[N/m^2]	S
Wave-induced volume flux in x-direction, if EXT file contains `mx`.	`Wrimap_waves`	[m^3/s/m]	S
Wave-induced volume flux in y-direction, if EXT file contains `my`.	`Wrimap_waves`	[m^3/s/m]	S
Wave energy dissipation rate at the free surface, if EXT file contains `dissurf`.	`Wrimap_waves`	[w/m^2]	S
Wave energy dissipation rate due to white capping, if EXT file contains `diswcap`.	`Wrimap_waves`	[w/m^2]	S
Wave orbital velocity, if EXT file contains `ubot`.	`Wrimap_waves`	[m/s]	S
# The above variables are for flow without waves, when `[waves] flowWithoutWaves` is true.			
# The following variables are for flow with waves, when `[waves] flowWithoutWaves` is false.			
Wave energy per square meter, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[J/m^2]	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Roller energy per square meter, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[J/m^2]	S
Roller energy dissipation per square meter, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[W/m^2]	S
Wave breaking energy dissipation per square meter, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[W/m^2]	S
Wave bottom energy dissipation per square meter, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[W/m^2]	S
Radiation stress, including x- and y-components, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[N/m^2]	S
Radiation stress, xy-component, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[N/m^2]	S
Sea surface wave phase celerity, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[m/s]	S
Sea surface wave group celerity, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[m/s]	S
Sea surface wave mean frequency, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[rad/s]	S
Sea surface wave wavenumber, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[rad/m]	S
Sea surface wave ratio group phase speed, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[-]	S
Sea surface wave refraction celerity, if MDU keyword `[waves] Wavemodelnr = 4`.	`Wrimap_waves`	[rad/s]	S
Sea surface wave wavelength, if MDU keyword `[waves] Wavemodelnr = 4` and `windmodel`, read from the specified file in MDU keyword `[waves] SurfbeatInput`, is 1.	`Wrimap_waves`	[m]	S
Wind source term on wave energy, if MDU keyword `[waves] Wavemodelnr = 4`, and `windmodel` and `windsource` that are read from the specified file in MDU keyword `[waves] SurfbeatInput`, are 1.	`Wrimap_waves`	[J/m^2/s]	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Wind source term on wave period, if MDU keyword `[waves] Wavemodelnr = 4`, and `windmodel` and `windsource`, read from the specified file in MDU keyword `[waves] SurfbeatInput`, are 1.	`Wrimap_waves`	[s/s]	S
(For 3D models only) Surface layer wave forcing term, including x- and y-components, if MDU keyword `[waves] Wavemodelnr` is 3 or 4.	`Wrimap_waves`	[N/m^2]	S
(For 3D models only) Water body wave forcing term, including x- and y-components, if MDU keyword `[waves] Wavemodelnr` is 3 or 4.	`Wrimap_waves`	[N/m^2]	S
RMS wave height, if MDU keyword `[waves] Wavemodelnr > 0` and `[waves] jamapsigwav = 0`.	`Wrimap_waves`	[m]	S
Significant wave height, if MDU keyword `[waves] Wavemodelnr > 0` and `[waves] jamapsigwav` is non-zero.	`Wrimap_waves`	[m]	S
Wave orbital velocity, if MDU keyword `[waves] Wavemodelnr > 0`.	`Wrimap_waves`	[m/s]	S
Stokes drift including x- and y-components, if MDU keyword `[waves] Wavemodelnr > 0`.	`Wrimap_waves`	[m/s]	S or S3D
Stokes drift including n- and t-components, if MDU keyword `[waves] Wavemodelnr > 0`.	`Wrimap_waves`	[m/s]	U or U3D
Wave from direction, if MDU keyword `[waves] Wavemodelnr > 0`.	`Wrimap_waves`	[deg from N]	S
Wave period, if MDU keyword `[waves] Wavemodelnr > 0`.	`Wrimap_waves`	[s]	S
Wave force including x- and y-components, if MDU keyword `[waves] Wavemodelnr` is 3 or 4.	`Wrimap_waves`	[N/m^2]	S or S3D
Wave force at velocity point including n- and t-components, if MDU keyword `[waves] Wavemodelnr` is 3 or 4.	`Wrimap_waves`	[N/m^2]	U or U3D
# The above variables are for flow with waves, when `[waves] flowWithoutWaves` is false.			
Time-varying dune height, if BedformFile keyword `[bedform] BdfOut` and `[bedform] Bdf` are true.		[m]	S
Time-varying dune length, if BedformFile keyword `[bedform] BdfOut` and `[bedform] Bdf` are true.		[m]	S
Ripple roughness height, if BedformFile keyword `[bedform] BdfOut` is true and, either MDU keywords `[waves] Wavemodelnr > 0` and `[waves] Rouwav = VR04`, or Van Rijn 2004 transport formula is used.		[m]	S

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
Megaripple roughness height, if BedformFile keyword `[bedform] BdfOut` is true and, either MDU keywords `[waves] Wavemodelnr > 0` and `[waves] Rouwav = VR04`, or Van Rijn 2004 transport formula is used.		[m]	S
Dune roughness height, if BedformFile keyword `[bedform] BdfOut` is true and, either MDU keywords `[waves] Wavemodelnr > 0` and `[waves] Rouwav = VR04`, or Van Rijn 2004 transport formula is used.		[m]	S
Bedform roughness height, if BedformFile keyword `[bedform] BdfOut` is true and, either MDU keywords `[waves] Wavemodelnr > 0` and `[waves] Rouwav = VR04`, or Van Rijn 2004 transport formula is used.		[m]	S
# The following variables are for trachytope roughness:			
Chezy roughness from trachytopes, if MDU keyword `[physics] UnifFrictType = 0`.	`Wrimap_trachytopes`	[$\mathrm{m}^{0.5}$/s]	L
Manning roughness from trachytopes, if MDU keyword `[physics] UnifFrictType = 1`.	`Wrimap_trachytopes`	[s/$\mathrm{m}^{0.333}$]	L
White-Colebrook roughness from trachytopes, if MDU keyword `[physics] UnifFrictType` is 2 or 3.	`Wrimap_trachytopes`	[m]	L
Roughness from trachytopes, if MDU keyword `[physics] UnifFrictType` is not equal to 0, 1, 2, or 3.	`Wrimap_trachytopes`	[-]	L
Stem density of vegetation.		[1/m^2]	S
Stem diameter of vegetation.		[m]	S
Stem height of vegetation.		[m]	S
Calibration factor for roughness.	`Wrimap_calibration`	[$\mathrm{m}^{0.5}$/s]	L
# The above variables are for trachytope roughness.			
# The following variables are about nudging, when the EXT file contains `nudge_salinity_temperature`.			
Nudging relaxing time.	`Wrimap_nudging`	[s]	S
Nudging temperature.	`Wrimap_nudging`	[°C]	S3D
Nudging salinity.	`Wrimap_nudging`	[1e-3]	S3D
Difference of nudging temperature with temperature.	`Wrimap_nudging`	[°C]	S3D
Difference of nudging salinity with salinity.	`Wrimap_nudging`	[1e-3]	S3D

(continued on next page)

(continued from previous page)

Quantity name	MDU option	Unit	Location
# The above variables are about nudging, when the EXT file contains `nudge_salinity_temperature`.			

Additionally, some variables specific for 1D networks can be written. Most are related to the concept of *ground level* in 1D. The ground level is not to be confused with bed level. The ground level of a 1D node is equal to:

- the `streetLevel`, if a storage node is defined on that node (Section C.17) and if `useTable=false`. Storage on ground is allowed if `storageType≠closed`.
- the highest `levels` value, if a storage node is defined on that node (Section C.17) and if `useTable=true`.
- the highest cross section value (interpolated from the two nearest cross sections) when no storage node is defined on that node. For closed cross sections (see section C.16.1), this level is only used to compute freeboard; no storage above ground level is allowed.

The additional available output variables on 1D nodes are:

- Freeboard at each node (only for 1D nodes):
  - Freeboard is calculated as ground level minus water level. It is the height still available for further water level increase.
  - When no storage node is present nor cross sections are nearby, a fill value is written.
  - Freeboard can be negative when the water level is above the ground level (only relevant for non-closed storage nodes with a street level or open water cross sections).
  - Freeboard has a minimal value of 0 if the storage node or nearest cross sections are closed.
- Waterdepth on ground at each node (only for 1D nodes):
  - This is calculated as water level minus ground level, only when ground level is defined and storage on ground is allowed on that node.
  - When no storage on ground is allowed, a fill value is written.
  - Its minimal value is 0 (for instance when the water level is below ground level).
- Volume on ground at each node (only for 1D nodes):
  - This is calculated as current volume minus the maximum volume below ground level.
  - When no storage on ground is allowed, a fill value is written.
  - Its minimal value is 0 (for instance when current volume is smaller than the maximum volume below ground level, i.e., when the water level is below ground level).
- Cumulative time at each node (only for 1D nodes) when water is above ground level, or more precisely: when waterdepth on ground is larger than `Wrimap_wet_waterdepth_threshold` (this allows to use a different 'wet' threshold for output files than the computational flooding threshold `Epshu`).
- Current total 1D2D net inflow (discharge) and cumulative total 1D2D net inflow (volume), (only for 1D nodes):
  - The current total 1D2D net inflow (discharge) at a 1D node is the summation of instantaneous discharge through all the connected 1D2D links at each output time.
  - The cumulative total 1D2D net inflow (volume) at a 1D node is the cumulative total net volume that has flowed into this node through all the connected 1D2D links since the starting time of the simulation. At each computational time step it is incremented by the summation of the net volume through all the connected 1D2D links.
- Current total lateral net inflow (discharge) and cumulative total lateral net inflow (volume), (only for 1D nodes):
  - The current total lateral net inflow (discharge) at a 1D node is the summation of instantaneous discharge through all the connected laterals at each output time.
  - The cumulative total lateral net inflow (volume) at a 1D node is the cumulative total net volume that has flowed into this node through all the connected laterals since the starting time of the simulation. At each computational time step it is incremented by the summation of the net volume through all the connected laterals.

The additional available output variables on 1D links are:

- water level gradient (only for 1D links). This is the gradient in flow link direction. Depending on the input model grid, flow links may sometimes be oriented opposite to the branch orientation.

Variables on flow links/velocity points

Variables on grid cell edges (represented by the velocity point) are typically defined as follows:

```
double q1(time, nFlowLink) ;
  q1:coordinates = "FlowLink_xu FlowLink_yu" ;
  q1:long_name = "Flow flux" ;
  q1:units = "m3 s-1" ;
```

Other quantities that can be written are:

- Normal velocities at the old and new time level;
- Horizontal viscosity and diffusivity;
- Edge-centered wind speed components;
- Effective roughness values from trachytopes/vegetation (section 15.2)

Variables for flow analysis

By selecting `Wrimap_flow_analysis = 1` under `[Output]` in the MDU file, a number of flow analysis variables will be written to the map file. These variables can be used to find possible bottlenecks in the model.

Table F.20: *Flow analysis variables in the map file. Column 'Location' shows the grid location: S means pressure points, U means velocity points.*

Quantity name	Unit	Location
Number of times negative depth was calculated.	[-]	S
Cumulative number of times negative depth was calculated.	[-]	S
Number of times no nonlinear convergence was caused	[-]	S
Cumulative number of times no nonlinear convergence was caused	[-]	S
Number of times a node was limiting for the computational time step	[-]	S
Cumulative number of times a node was limiting for the computational time step	[-]	S
Courant number	[-]	S

Remarks:

- During the simulation, the non-linear iteration can calculate negative depths. In general an occasional negative depth does not affect the simulation. However frequently calculated negative depths can be the cause of strange results or instabilities.
- The report of 'no convergence in the non linear iteration' (counted in `Number of times no nonlinear convergence is caused`), is a strong indicator that something might be wrong with the model. This count is incremented for the flow node that showed the largest water level difference in the last nonlinear iteration.
- The performance of a model is affected by the maximal timestep that can be used during the simulation. In `Number of times a node was limiting for the computational time step` the locations that are bottlenecks for the maximal timestep can be found. Using this information, the locations that are restricting for the maximal timestep can be investigated. Subsequently an attempt can be made to find a solution to reduce these bottlenecks.
- The Courant number is the limiting factor for the maximal timestep. This variable can be used to investigate whether only a few locations are limiting the timestep or whether it uniformly distributed across the complete model domain.
- The `Courant number` is an instantaneous value, valid for the moment the output is written to the map file.

- ◇ The other output variables come in two forms. The first of each two variables contains the number of occurrences during the past map output interval, whereas the second (denoted with "Cumulative"/_cum) contains the total number of occurrences since the start of the simulation until the current map output time.

F.3.3 Class map files as netCDF clm-file

The class map option gives output in a netCDF file, where the size of the output file is significantly less then the size of a map-file. This is achieved by storing the field of interest (currently only water levels and water depths) not as a double (8 bytes), but as a byte that represents the class it is in. As the class will not change every time, compressing is used to get even smaller files than the factor 8 based on the ratio double/byte.

This format is very suitable for making animations of dryfall and flooding.

The class map file is defined in the mdu-file by:

```
[output]
...
ClassMapFile               = weirfree_clm.nc    # Filename for class map output
WaterlevelClasses          = -3.0 2.0 5.0       # Water level classes (m)
WaterdepthClasses          = 0.5 1.0 5.0 10.0  # Water depth classes (m)
ClassMapInterval           = 5.0                # Class map output interval (s)
```

The class map file is always in UGRID format. When running in parallel, a class map file is generated for each process. Use mapmerge to collect them in a single file.

F.3.4 Restart files as netCDF rst-file

Restart files <*mdu_name*_yyyymmdd_HHMMSS_rst.nc> are almost identical to map files, except that they contain all data for just a single time. The grid and flow geometry information is not present, except for the flow cell/link coordinates. Moreover, restart files resulted from a parallel run contains parallelization information. For this reason, restart files are less suitable for plotting, but efficient for restarting a computation.

F.3.5 Volume tables per- gridpoint output as netCDF file

Dimensions

The per- gridpoint output net file contains all 1D flow geometry dimensions, plus the volume table levels and increment:

```
        netcdf PerGridpoint_TestModel_01 {
dimensions:
        Two = 2 ;
        network1d_nEdges = 4 ;
        network1d_nNodes = 5 ;
        network1d_nGeometryNodes = 141 ;
        strLengthIds = 40 ;
        strLengthLongNames = 80 ;
        mesh1d_nNodes = 28 ;
        mesh1d_nEdges = 27 ;
        nmesh1d_FlowElemContourPts = 7 ;
        levels = 45 ;
        increment = 1 ;
```

Furthermore full network information is included:

```
int network1d ;
        network1d:cf_role = "mesh_topology" ;
        network1d:long_name = "Topology data of 1D network" ;
        network1d:edge_dimension = "network1d_nEdges" ;
        network1d:edge_geometry = "network1d_geometry" ;
        network1d:edge_node_connectivity = "network1d_edge_nodes" ;
        network1d:node_coordinates = "network1d_node_x network1d_node_y" ;
        network1d:node_dimension = "network1d_nNodes" ;
        network1d:topology_dimension = 1 ;
        network1d:node_id = "network1d_node_id" ;
        network1d:node_long_name = "network1d_node_long_name" ;
        network1d:branch_id = "network1d_branch_id" ;
        network1d:branch_long_name = "network1d_branch_long_name" ;
        network1d:edge_length = "network1d_edge_length" ;
int network1d_edge_nodes(network1d_nEdges, Two) ;
        network1d_edge_nodes:cf_role = "edge_node_connectivity" ;
        network1d_edge_nodes:long_name = "Start and end nodes of network edges" ;
char network1d_branch_id(network1d_nEdges, strLengthIds) ;
        network1d_branch_id:long_name = "ID of branch geometries" ;
char network1d_branch_long_name(network1d_nEdges, strLengthLongNames) ;
        network1d_branch_long_name:long_name = "Long name of branch geometries" ;
double network1d_edge_length(network1d_nEdges) ;
        network1d_edge_length:long_name = "Real length of branch geometries" ;
        network1d_edge_length:units = "m" ;
char network1d_node_id(network1d_nNodes, strLengthIds) ;
        network1d_node_id:long_name = "ID of network nodes" ;
char network1d_node_long_name(network1d_nNodes, strLengthLongNames) ;
        network1d_node_long_name:long_name = "Long name of network nodes" ;
double network1d_node_x(network1d_nNodes) ;
        network1d_node_x:units = "m" ;
        network1d_node_x:standard_name = "projection_x_coordinate" ;
        network1d_node_x:long_name = "x-coordinate of network nodes" ;
double network1d_node_y(network1d_nNodes) ;
        network1d_node_y:units = "m" ;
        network1d_node_y:standard_name = "projection_y_coordinate" ;
        network1d_node_y:long_name = "y-coordinate of network nodes" ;
int network1d_geometry ;
        network1d_geometry:geometry_type = "line" ;
        network1d_geometry:long_name = "1D Geometry" ;
        network1d_geometry:node_count = "network1d_geom_node_count" ;
        network1d_geometry:node_coordinates = "network1d_geom_x network1d_geom_y" ;
int network1d_geom_node_count(network1d_nEdges) ;
        network1d_geom_node_count:long_name = "Number of geometry nodes per branch" ;
double network1d_geom_x(network1d_nGeometryNodes) ;
        network1d_geom_x:units = "m" ;
        network1d_geom_x:standard_name = "projection_x_coordinate" ;
        network1d_geom_x:long_name = "x-coordinate of branch geometry nodes" ;
double network1d_geom_y(network1d_nGeometryNodes) ;
        network1d_geom_y:units = "m" ;
        network1d_geom_y:standard_name = "projection_y_coordinate" ;
        network1d_geom_y:long_name = "y-coordinate of branch geometry nodes" ;
int network1d_branch_order(network1d_nEdges) ;
        network1d_branch_order:long_name = "Order of branches for interpolation" ;
        network1d_branch_order:mesh = "network1d" ;
        network1d_branch_order:location = "edge" ;
int network1d_branch_type(network1d_nEdges) ;
        network1d_branch_type:long_name = "Type of branches" ;
        network1d_branch_type:mesh = "network1d" ;
        network1d_branch_type:location = "edge" ;
```

As is the 1D mesh topology information:

```
int mesh1d ;
        mesh1d:cf_role = "mesh_topology" ;
        mesh1d:long_name = "Topology data of 1D mesh" ;
        mesh1d:topology_dimension = 1 ;
        mesh1d:coordinate_space = "network1d" ;
        mesh1d:edge_node_connectivity = "mesh1d_edge_nodes" ;
        mesh1d:node_dimension = "mesh1d_nNodes" ;
        mesh1d:edge_dimension = "mesh1d_nEdges" ;
        mesh1d:node_coordinates = "mesh1d_node_branch mesh1d_node_offset
        mesh1d_node_x mesh1d_node_y" ;
        mesh1d:edge_coordinates = "mesh1d_edge_branch mesh1d_edge_offset
        mesh1d_edge_x mesh1d_edge_y" ;
        mesh1d:node_id = "mesh1d_node_id" ;
        mesh1d:node_long_name = "mesh1d_node_long_name" ;
int mesh1d_node_branch(mesh1d_nNodes) ;
        mesh1d_node_branch:long_name = "Index of branch on which mesh nodes are
        mesh1d_node_branch:start_index = 0 ;
double mesh1d_node_offset(mesh1d_nNodes) ;
        mesh1d_node_offset:long_name = "Offset along branch of mesh nodes" ;
        mesh1d_node_offset:units = "m" ;
double mesh1d_node_x(mesh1d_nNodes) ;
        mesh1d_node_x:units = "m" ;
        mesh1d_node_x:standard_name = "projection_x_coordinate" ;
        mesh1d_node_x:long_name = "x-coordinate of mesh nodes" ;
        mesh1d_node_x:bounds = "mesh1d_FlowElemContour_x" ;
double mesh1d_node_y(mesh1d_nNodes) ;
        mesh1d_node_y:units = "m" ;
        mesh1d_node_y:standard_name = "projection_y_coordinate" ;
        mesh1d_node_y:long_name = "y-coordinate of mesh nodes" ;
        mesh1d_node_y:bounds = "mesh1d_FlowElemContour_y" ;
int mesh1d_edge_branch(mesh1d_nEdges) ;
        mesh1d_edge_branch:long_name = "Index of branch on which mesh edges are
        mesh1d_edge_branch:start_index = 0 ;
double mesh1d_edge_offset(mesh1d_nEdges) ;
        mesh1d_edge_offset:long_name = "Offset along branch of mesh edges" ;
        mesh1d_edge_offset:units = "m" ;
double mesh1d_edge_x(mesh1d_nEdges) ;
        mesh1d_edge_x:units = "m" ;
        mesh1d_edge_x:standard_name = "projection_x_coordinate" ;
        mesh1d_edge_x:long_name = "Characteristic x-coordinate of the mesh edge ;
double mesh1d_edge_y(mesh1d_nEdges) ;
        mesh1d_edge_y:units = "m" ;
        mesh1d_edge_y:standard_name = "projection_y_coordinate" ;
        mesh1d_edge_y:long_name = "Characteristic y-coordinate of the mesh edge ;
char mesh1d_node_id(mesh1d_nNodes, strLengthIds) ;
        mesh1d_node_id:long_name = "ID of mesh nodes" ;
char mesh1d_node_long_name(mesh1d_nNodes, strLengthLongNames) ;
        mesh1d_node_long_name:long_name = "Long name of mesh nodes" ;
int mesh1d_edge_nodes(mesh1d_nEdges, Two) ;
        mesh1d_edge_nodes:cf_role = "edge_node_connectivity" ;
        mesh1d_edge_nodes:long_name = "Start and end nodes of mesh edges" ;
        mesh1d_edge_nodes:start_index = 1 ;
double mesh1d_FlowElemContour_x(mesh1d_nNodes, nmesh1d_FlowElemContourPts) ;
        mesh1d_FlowElemContour_x:units = "m" ;
        mesh1d_FlowElemContour_x:standard_name = "projection_x_coordinate" ;
        mesh1d_FlowElemContour_x:long_name = "list of x-coordinates forming flow
        mesh1d_FlowElemContour_x:_FillValue = -999. ;
double mesh1d_FlowElemContour_y(mesh1d_nNodes, nmesh1d_FlowElemContourPts) ;
        mesh1d_FlowElemContour_y:units = "m" ;
        mesh1d_FlowElemContour_y:standard_name = "projection_y_coordinate" ;
        mesh1d_FlowElemContour_y:long_name = "list of y-coordinates forming flow
        mesh1d_FlowElemContour_y:_FillValue = -999. ;
```

Finally it contains the actual volume table data written. For the per- gridpoint output the volume and surface arrays are two dimensional, containing a value per volume table level for every 1D node.

```
double volume(mesh1d_nNodes, levels) ;
        volume:long_name = "Volume" ;
        volume:units = "m3" ;
double surface(mesh1d_nNodes, levels) ;
        surface:long_name = "Surface" ;
        surface:units = "m2" ;
int numlevels(mesh1d_nNodes) ;
        numlevels:long_name = "Number of levels" ;
        numlevels:units = "" ;
double bedlevel(mesh1d_nNodes) ;
        bedlevel:long_name = "Bedlevel" ;
        bedlevel:units = "m" ;
double topheight(mesh1d_nNodes) ;
        topheight:long_name = "TopHeight" ;
        topheight:units = "m" ;
double increment(increment) ;
```

```
        increment:long_name = "Increment" ;
        increment:units = "m" ;
```

F.3.6 NetCDF versions and file formats

The supported netCDF file formats are netCDF 3 and 4, configurable in the MDU via the `[output] NcFormat` keyword. Format 3 is the default for all netCDF output files, except for the classmap file, which always uses netCDF 4, because it uses compression.

The precision of the data written to the map-file and his-file can be configured in the MDU using the keywords `[output] NcMapDataPrecision` and `[output] NcHisDataPrecision`. Note that the station coordinates are always written in double precision for accuracy.

Compression of the his-file can be enabled by setting `[output] NcCompression=1`. Note that this feature only works when `[output] NcFormat=4`.

The history files, on the other hand, always use netCDF 3, because that was found beneficial for I/O performance.

To improve I/O performance, the following non-default settings may be considered. `NcNoUnlimited` writes the time dimension directly at full length, instead of at an unlimited length. This is only possible for regular model runs with start and stop time set (or at least the map output start and stop time set). The entire file size will be allocated directly upon initialization of the model. This setting is recommended (but not always required) to be combined with `NcFormat=4`, such that the variable size is not limited.

I/O performance can sometimes benefit greatly from disabling flushing every output timestep, using MDU setting `[output] NcNoForcedFlush`. The only consideration here is that during a running simulation, the buffered results may take some time before being visible in the output data on disk.

F.4 Shapefiles

Shapefiles[4] are widely used for visualization in geographic information system (GIS) software. A shapefile stores geometric locations and corresponding attributes information for the spatial features. D-Flow FM can generate shapefiles by setting keyword in the MDU file. Table F.21 shows what features can be written to a shapefile by D-Flow FM, and the corresponding MDU settings. (By default these keywords are zero, which disables writing the shapefile.) Moreover, this function is also valid in parallel simulation.

[4] https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

Table F.21: *Features and MDU settings for generating shapefiles*

Features for shapefiles	**MDU setting**
Cross sections	`[output] Wrishp_crs = 1`
Dambreaks	`[output] Wrishp_dambreak = 1`
Dry area	`[output] Wrishp_dryarea = 1`
Embankments	`[output] Wrishp_ebm = 1`
Fixed weirs	`[output] Wrishp_fxw = 1`
Gates	`[output] Wrishp_gate= 1`
General structures	`[output] Wrishp_genstruc = 1`
Observation stations	`[output] Wrishp_obs = 1`
Pump	`[output] Wrishp_pump = 1`
Source-sinks	`[output] Wrishp_src = 1`
Thin dams	`[output] Wrishp_thd = 1`
Weirs	`[output] Wrishp_weir= 1`

F.5 Post processing on his-file

There are three options for post processing on the history file. Two options are for finding the maximum value and one for the running mean maximum value for each history station or cross section. They are options of the program `dfmoutput`.

F.5.1 Max25 function

The max25 function computes statistics for all stations in the his-file. Basic statistics includes the `first`, `minimum`, `maximum` and `last` value. It computes a running mean over the given filter length in time (by default two filter lengths: 13 and 25). For this running mean the minimum, maximum and last value are computed. The maximum value of the running mean over 25 points (in time) is abbreviated to `max25`.

Basic usage for max25 is:

```
dfmoutput max25 --infile [hisfile.nc]
```

There are three optional arguments:

```
--outfile [name of the outputfile], default max25.out
--varname [variable to find maximum for], default waterlevel
--filterlength [list of numbers of points in time], default two values 13, 25
```

F.5.2 Maximum based on a generic filter

The second option is to find the maximum value after applying a 4th order monotone filter on the time serie for each station. This filter has three parameters: the filter length and two coefficients. These parameters can be given by the user, see below.

Basic usage for maximum based on a generic filter is:

```
dfmoutput genfilter --infile [hisfile.nc]
```

There are five optional arguments:

```
--outfile [name of the outputfile], default max.out
--varname [variable to find maximum for], default waterlevel
--intval [filter period], default 6
--coefimpl [filter coefficient implicit part], default 0.3
--coefexpl [filter coefficient explicit part], default 0.3
```

F.5.3 Maximum based on a running mean filter

The third option is a maximum based on a running mean filter, the same as can be used in the fou-file. See C.14 for a definition of this filter.

Basic usage for maximum based on a running mean is:

```
dfmoutput max_running_mean --infile [hisfile.nc]
```

There are three optional arguments:

```
--outfile [name of the outputfile], default max_running_mean.out
--varname [variable to find maximum for], default waterlevel
--filterlength [filter lengths], default two values 13, 25
```

G Model generation

G.1 Introduction

Parts of the model building process can also be automated, to support model generation. This comes in addition to the full modelling process as described in chapter 5.

G.2 Grid generation

Automated grid generation is limited to a few basic types, described hereafter. More specific grid generation should be done by the modeller in RGFGRID. Automated grid generation is currently done from the commandline using the `dflowfm-cli.exe` program (`dflowfm` on Linux).

G.2.1 Uniform Cartesian grid

A uniform Cartesian (i.e., rectangular) grid can be generated by specifying the bounding box and desired grid resolution:

```
> dflowfm-cli.exe --gridgen:x0=$x_{LL}$:y0=$y_{LL}$:dx=$\Delta x$:dy=$\Delta y$:ncols=$M$:nrows=$N$
```

This produces a file <out_net.nc>. When the coordinate system for the grid should be spherical (WGS84), add the option `:spherical=1`.

G.2.2 Locally refined Cartesian grid

An existing grid can be locally refined based on an additional input file that defines the refinement locations. Here, we will now consider a uniform Cartesian (i.e., rectangular) that needs to be refined based on a raster file (e.g. an ArcInfo file). The grid refinement is currently based on 'Courant' refinement, but this can be applied to different refinement metrics as well. The command for refinement is:

```
> dflowfm-cli.exe --refine:hmin=$\Delta x_{\min}$:dtmax=$\Delta t_{\max}$:connect=1:outsidecell=1
                  unif_net.nc refineC.asc
```

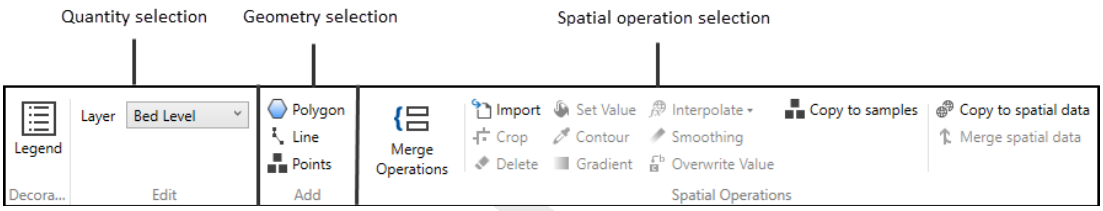
The input file <unif_net.nc> is a file that has been produced by the `--gridgen` command (for this Cartesian case, see section G.2.1), for example with a grid resolution $\Delta x_{\max}$. The input file <refineC.asc> should contain values 1 for locations that need the finest size $\Delta x_{\min}$, 4 for one step coarser, 16 for two steps coarser. In general: 2^{2n}, with n the number of coarsening steps. The value of $\Delta t_{\max}$ is a fictitious value here, and should be set to $\Delta t_{\max} = \Delta x_{\min}/\sqrt{9.81}$. When entering only a few decimal digits for $\Delta t_{\max}$, make sure to round it up.

This again produces a file <out_net.nc>.

H Spatial editor

H.1 Introduction

The spatial editor is a generic feature of the User Interface for editing spatial data (e.g. bed level, roughness, viscosity, initial conditions, sediment availability). The spatial editor supports both point clouds and coverages (e.g. data on a grid or network). Therefore, you can use the spatial editor both to edit spatial data in general and to prepare model input. This Chapter describes the general features of the spatial editor (section H.2), (spatial) quantity selection (section H.3), geometry operations (section H.4), spatial operations (section H.5) and the operation stack (section H.6). The examples given below are typically focusing on use of the spatial editor in the D-Flow FM User Manual.



Layer Bed Level

Figure H.1: *Overview of spatial editor functionality in Map ribbon*

H.2 General

H.2.1 Overview of spatial editor

The spatial editor functionality can be accessed from the “Spatial Operations” ribbon (Figure H.1). Typically, you first select which data set or quantity (either a point cloud or a coverage) to work on (e.g. bed level, roughness, viscosity, initial conditions, sediment availability), then a geometry (e.g. point, line, polygon) and finally which spatial operation to perform (e.g. crop, delete, set value, contour, interpolate, smoothing). Typical workflows are as follows:

Working on a point cloud dataset:

1 Import the dataset as point cloud (section H.2.2)
2 Activate/select the dataset (quantity) in the spatial editor (section H.3)
3 Select a geometry to perform an operation on (section H.4)
4 Select the spatial operation for this geometry (section H.5)
5 Repeat steps 3 and 4 until you are satisfied with the data
6 Export the dataset (section H.6.7)

Working on a coverage (e.g. model input):

1 Activate/select the spatial quantity to work on in the spatial editor (section H.3)
2 Optional: import a dataset as point cloud (section H.5.1)
3 Select a geometry to perform an operation on (section H.4)
4 Select the spatial operation for this geometry (section H.5)
5 Repeat steps 3 and 4 until you are satisfied with the data
6 Interpolate the point cloud to the grid or network (section H.5.10)
7 Optional: export the dataset (section H.6.7)

Upon performing a spatial operation, the ‘Operations’ panel will open (see Figure H.53) with the operations stack. This stack keeps track of the workflow of spatial operations that you performed. This helps you to make transparent how you arrived at your ‘final’ dataset without having to save all the intermediate datasets (steps) separately. Moreover, the stack is reproducible and easily editable without having to start all over again. When working on a coverage (e.g. the second workflow), point clouds can be used to construct the coverage. In this case the coverage (for example ‘Bed level’) is the ‘trunk’ of the workflow and the point clouds (appearing as sets in the stack) are ‘branches’ or subsets of this trunk (see Figure H.53). By selecting the set or coverage in the ‘Operations’ panel you determine on which dataset you are working. The interpolate operation (section H.5.10) allows you to bring data from the point cloud (branch) to the coverage (trunk). For more information on the stack you are referred to section section H.6.

H.2.2 Import/export dataset

To import a (point cloud) dataset use the context menu on “project” in the “Project Tree”, select “import” from the context menu and select the option "Points from XYZ-file" (Figure H.2). There is yet another method to import the point cloud. Click on "Import" icon under the Home ribbon to obtain a drop-down menu with the list of importers. Select the option "Points from XYZ-file" under Spatial data section (Figure H.3) to launch the import wizard.

After the import, the point cloud will be added to the project tree. To activate the point cloud in the spatial editor, either double click the dataset in the project tree (Figure H.4) or select it from the drop down box in the spatial editor ribbon (Figure H.5).

Note: Exporter still to be implemented

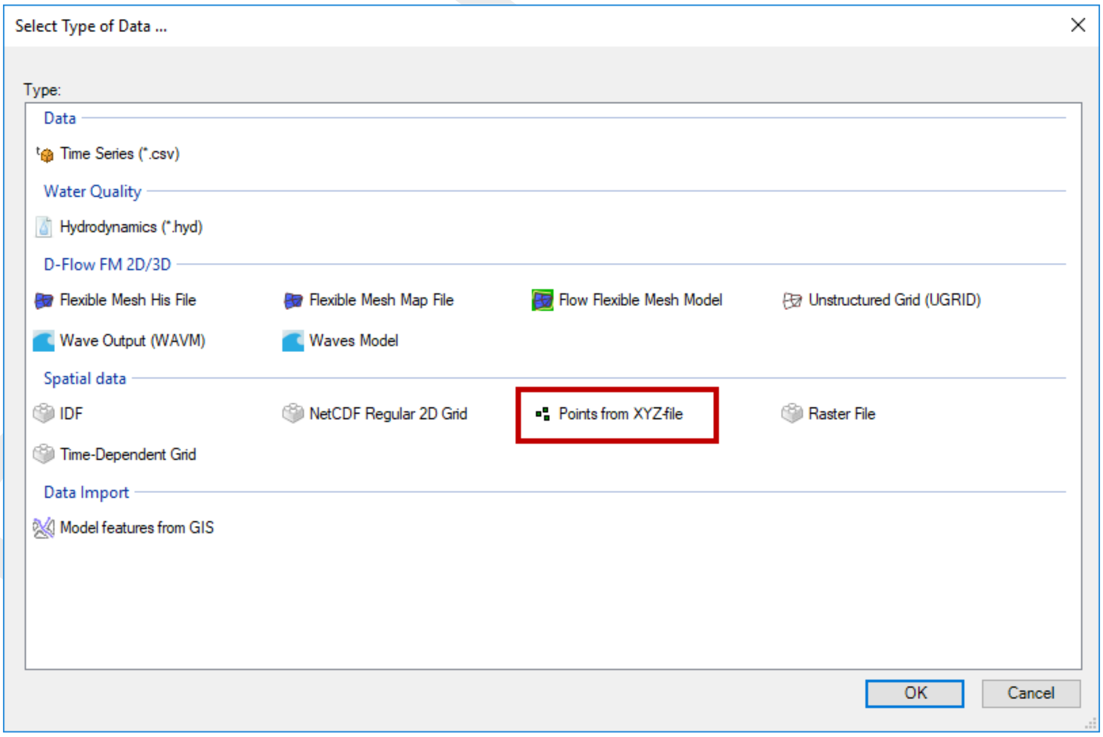


Figure H.2: Importing a point cloud into the project using the context menu on “project” in the project tree

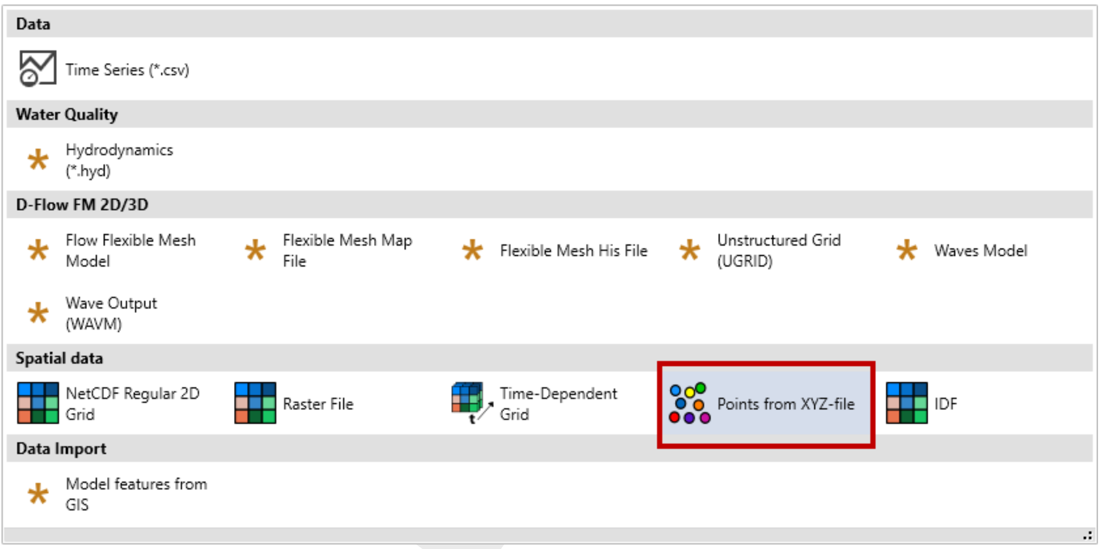


Figure H.3: *Importing a point cloud into the project using the "Import" drop-down menu in the Spatial Operations ribbon*

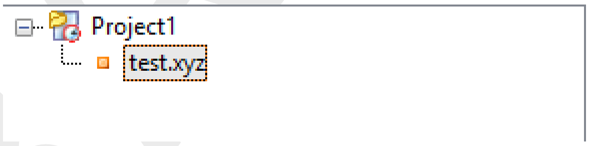


Figure H.4: *Activate the imported point cloud in the spatial editor by double clicking it in the project tree*

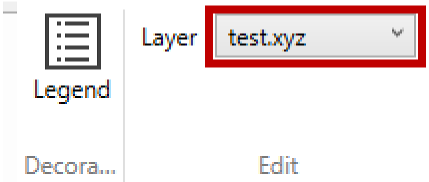


Figure H.5: *Activate the imported point cloud in the spatial editor by selecting it from the dropdown box in the Map ribbon*

H.2.3 Activate (spatial) model quantity

Similar to activating an imported dataset in the spatial editor, you can also activate a (spatial) model quantity (e.g. bed level, initial conditions, roughness, viscosity) in the spatial editor by double clicking the quantity in the project tree or selecting it from the dropdown box in the “Spatial Operations” ribbon.

H.2.4 Colorscale

In the spatial editor the colorscale for a spatial quantity can be made visible in the map window by clicking on the ”Legend" button in the “Spatial Operations” ribbon (Figure H.6). Then, in the Map window a colorscale is activated (Figure H.7). You can (de-)activate the color scale by clicking on the “Legend” button again. By default the colorscale is ranging from the minimum to the maximum value of the active dataset.

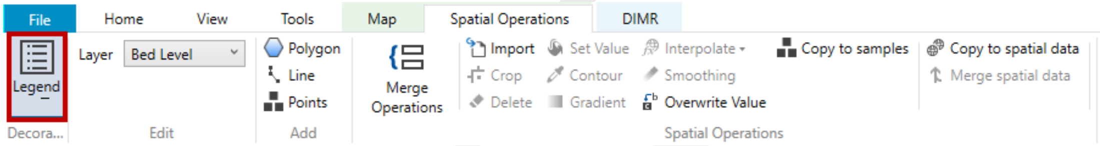


Figure H.6: Legend button in Map ribbon

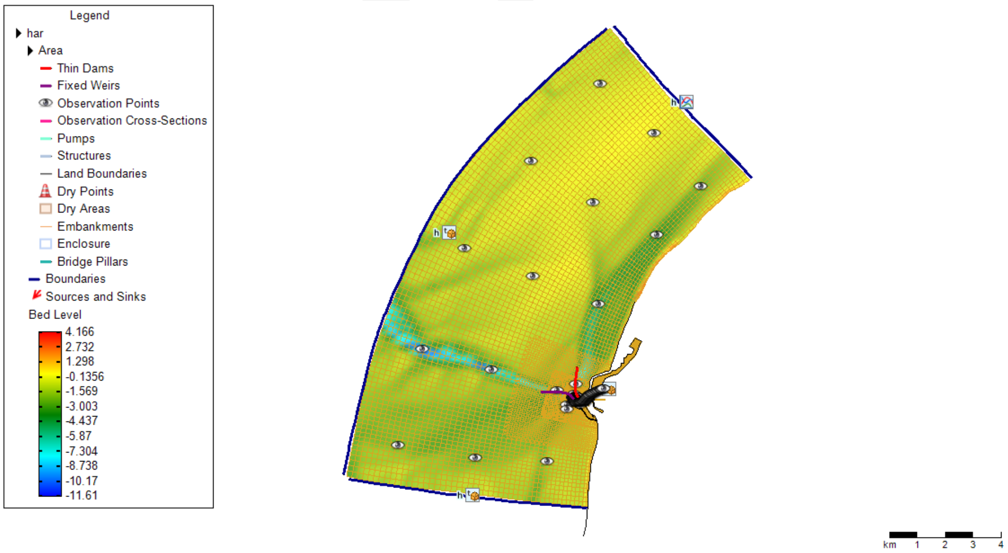
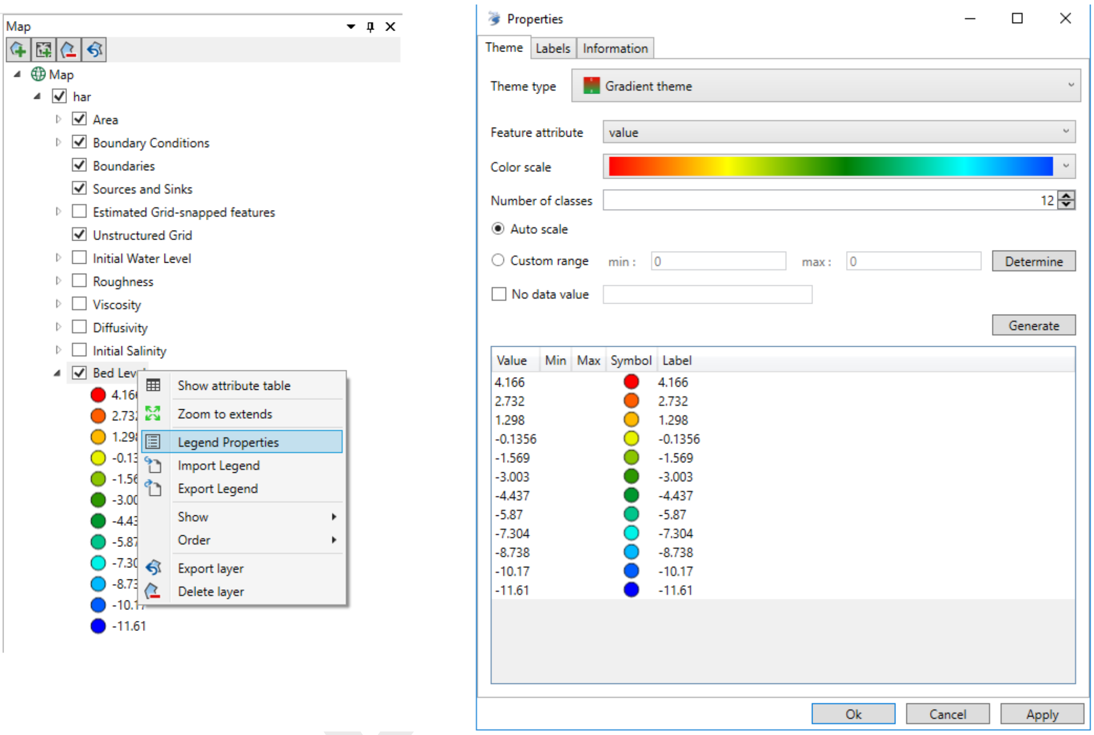


Figure H.7: Activate the colorscale by using the Legend in the map ribbon and the colorscale will become visible in the map window.

You can adjust the colormap and classes of the colorscale using the context menu on the spatial quantity in the “Map tree” and selecting “Legend Properties” (Figure H.8 left panel). A layer properties editor will open in which you can set the colormap to your own preferences (Figure H.8 right panel).



Map tree:

- ☑ har
  - ☑ Area
  - ☑ Boundary Conditions
  - ☑ Boundaries
  - ☑ Sources and Sinks
  - □ Estimated Grid-snapped features
  - ☑ Unstructured Grid
  - □ Initial Water Level
  - □ Roughness
  - □ Viscosity
  - □ Diffusivity
  - □ Initial Salinity
  - ☑ Bed Level

Theme type: Gradient theme

Feature attribute: value

Number of classes: 12

● Auto scale

○ Custom range min : 0 max : 0

□ No data value

Value	Min	Max	Symbol	Label
4.166			●	4.166
2.732			●	2.732
1.298			●	1.298
-0.1356			●	-0.1356
-1.569			●	-1.569
-3.003			●	-3.003
-4.437			●	-4.437
-5.87			●	-5.87
-7.304			●	-7.304
-8.738			●	-8.738
-10.17			●	-10.17
-11.61			●	-11.61

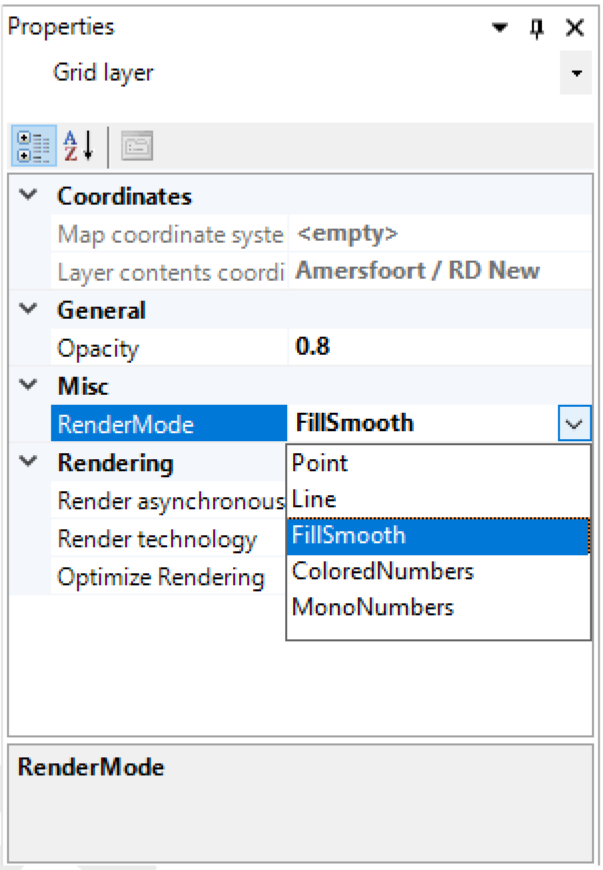
Figure H.8: *Edit the colorscale properties using the context menu on the active layer in the Map Tree*

H.2.5 Render mode

By default point clouds are rendered as (colored) points and coverages as shades (e.g. 'FillSmooth'). The render mode can be edited using the properties of the active layer Figure H.9. The User Interface offers the following render modes:

- Points
- Lines (only for coverages)
- Shades or 'FillSmooth' (only for coverages)
- Colored numbers
- Mono colored numbers

An example of a coverage rendered as colored numbers is given in Figure H.10.



Properties

Grid layer

Coordinates

Map coordinate syste	<empty>
Layer contents coordi	Amersfoort / RD New

General

Opacity	0.8

Misc

RenderMode	FillSmooth

Rendering

Render asynchronous

Render technology

Optimize Rendering

RenderMode

Figure H.9: Select the rendermode for the active layer in the property grid.

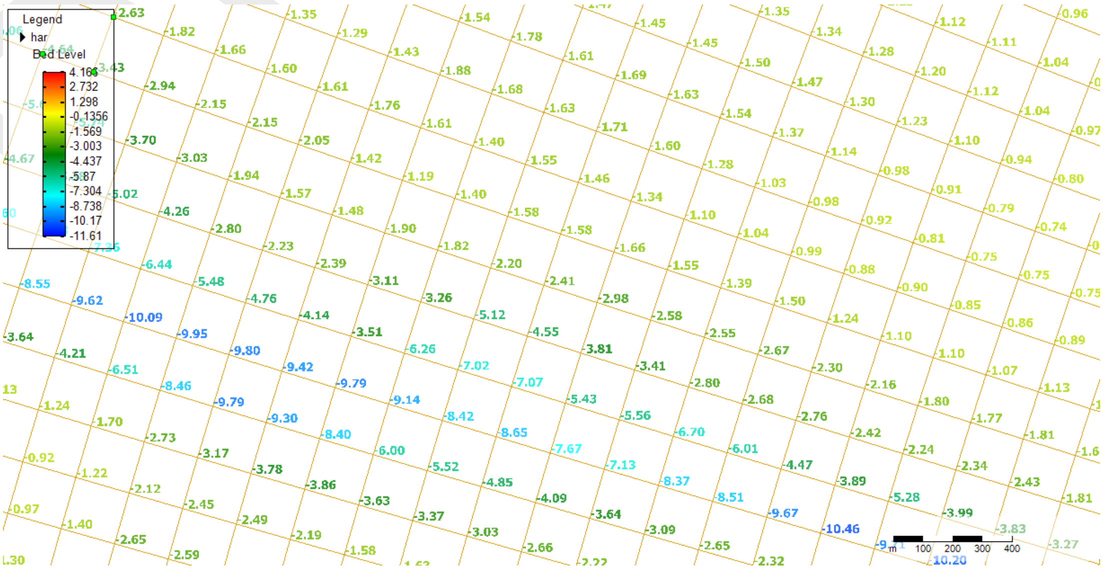


Figure H.10: Example of a coverage rendered as colored numbers.

H.2.6 Context menu

In addition to the selection of spatial operation from the ‘Spatial Operations’ ribbon (see section H.5), you can also select spatial operations using the context menu (e.g. context menu). After drawing a geometry and clicking the context menu all spatial operation available for the geometry will pop-up (see Figure H.11). The spatial operation will become active upon selecting it from the context menu.

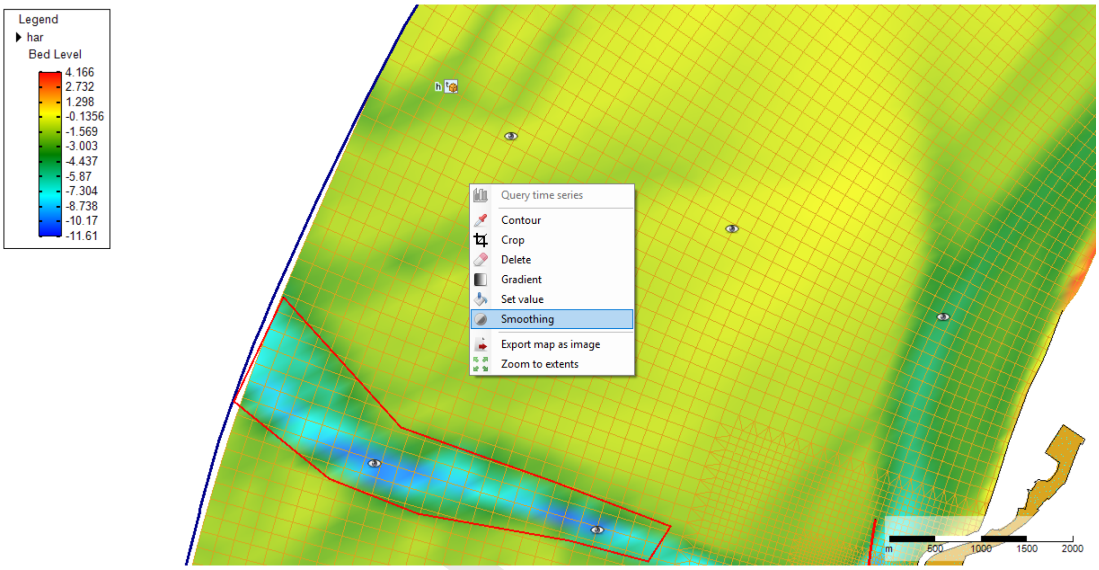


Figure H.11: Selecting a smoothing operation for a polygon geometry from the context menu (using context menu)

H.3 Quantity selection

A spatial quantity can be activated/selected either by double clicking it in the project tree (Figure H.12) or by selecting it from the dropdown box in the “Spatial Operations” ribbon (Figure H.13). Upon selecting the spatial quantity it will be shown as a point cloud (for a dataset) or coverage (for model input) on the central map. Typically, you start from a point cloud (either obtained from import or by generating samples yourself) which will eventually be interpolated to a grid or network (e.g. coverage). The spatial editor will keep track of both the changes made to the point cloud(s) and coverage of the selected spatial quantity. The information will be saved in the User Interface project and available the next time you open the project. **Note:** The operations are not yet saved in a human-readable/editable file

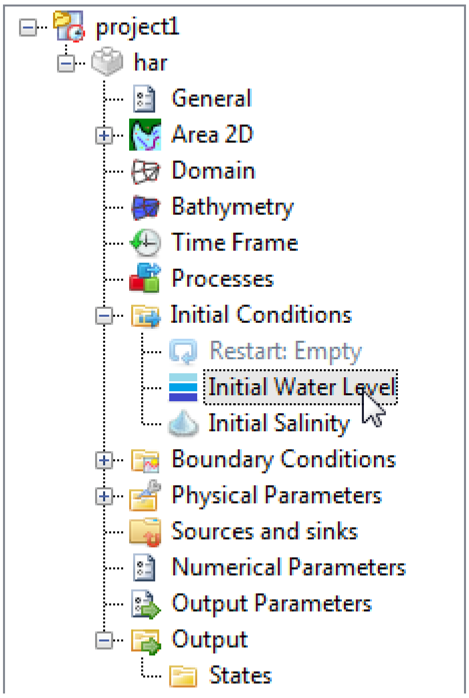


Figure H.12: Activating a spatial quantity by double clicking it in the project tree (in this example ‘Initial Water Level’)

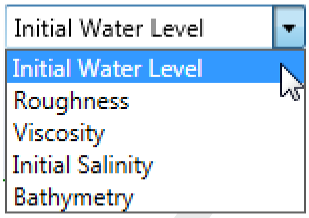


Figure H.13: *Activating a spatial quantity by selecting it from the dropdown box in the 'Spatial Operations' ribbon*

H.4 Geometry operations

The spatial editor supports three types of geometry operations: (1) polygons, (2) lines and (3) points (see also Figure H.14). The following sub-sections subsequently describe how these geometries can be selected and edited. If you do not select any of these three geometry operations, the spatial operation automatically applies to all the data.

Note: Please note that the drawn geometries are not yet persistent, implying that the geometries once drawn cannot be edited yet. Upon pressing the "Esc" button while in editing mode all drawn geometries will disappear.

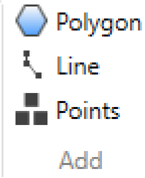


Figure H.14: *Overview of the available geometry operations in the 'Spatial Operations' ribbon*

H.4.1 Polygons

Upon selecting "Polygon" from the "Map" ribbon you can draw one or multiple polygons (Figure H.15). Each polygon point is defined by a single click with the LMB. The polygon is closed by double clicking the LMB. After drawing the (first) polygon, the available spatial operations for polygons are enabled in the "Spatial Operations" ribbon. The following spatial operations are available for polygons:

- Crop (section H.5.2)
- Delete (section H.5.3)
- Set Value (section H.5.4)
- Contour (section H.5.5)
- Gradient (section H.5.9)
- Smoothing (section H.5.11)
- Interpolate - only in case samples and a grid/network are available (section H.5.10)
- Copy to samples (section H.5.6)
- Copy to spatial data (section H.5.7) - only for grid coverages

The selected spatial operation applies to all the drawn polygons.

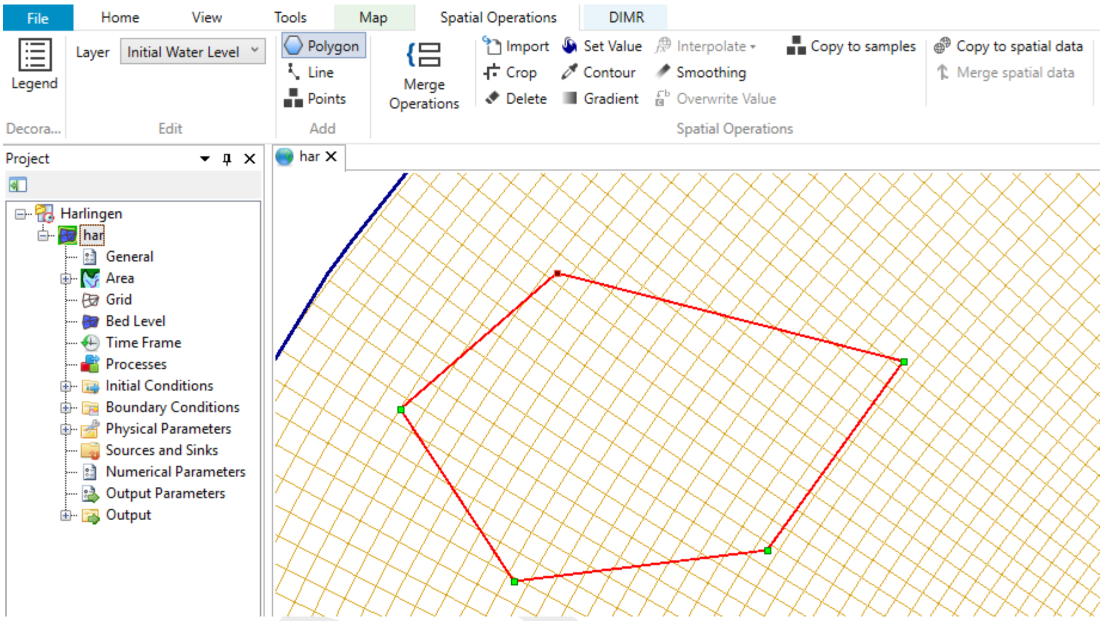


Figure H.15: *Activating the polygon operation and drawing polygons in the central map.*

H.4.2 Lines

Upon selecting “Line” from the “Map” ribbon you can draw one or multiple lines (Figure H.16). Each line point is defined by a single click with the LMB. The line is completed by double clicking the LMB. After drawing the (first) line, the available spatial operations for lines are enabled in the “Map” ribbon. The following spatial operations are available for lines:

- Contour (section H.5.5)
- Copy to samples (section H.5.6)
- Copy to spatial data (section H.5.7) - only for grid coverages

The selected spatial operation applies to all the drawn lines.

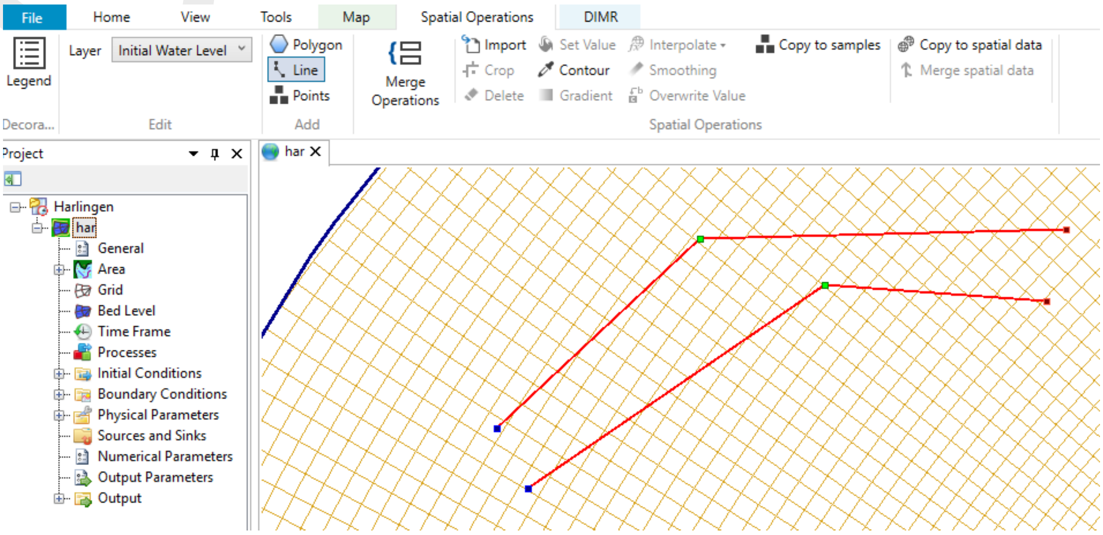
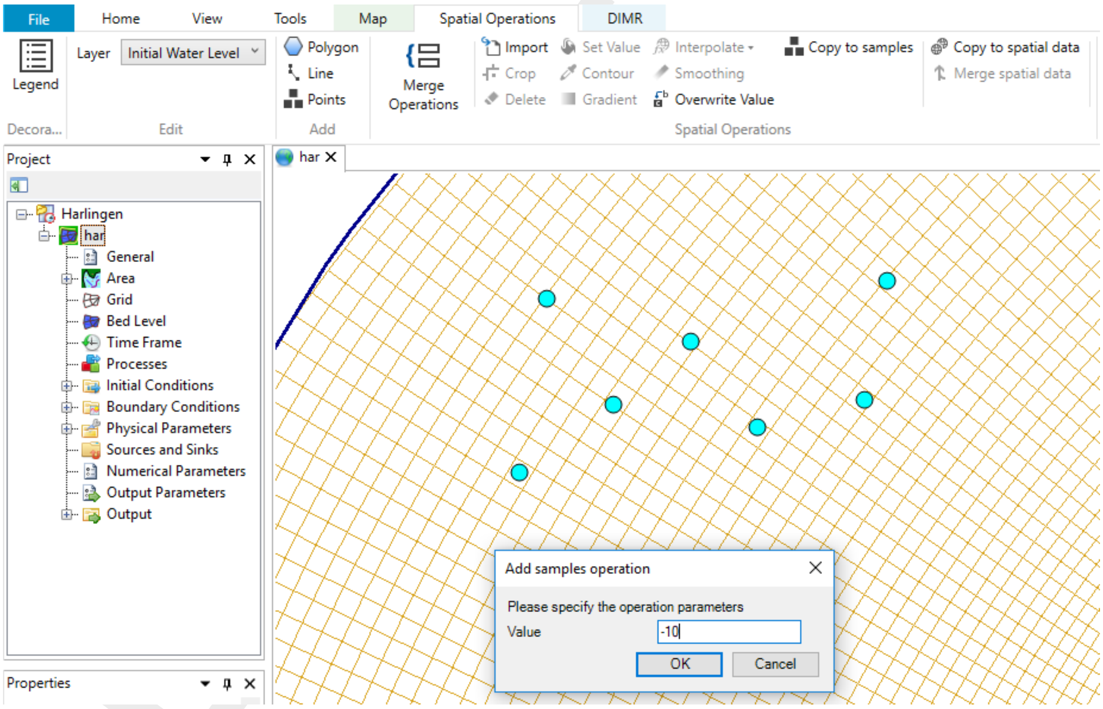


Figure H.16: *Activating the line operation and drawing lines in the central map.*

H.4.3 Points

Upon selecting “Add points” from the “Map” ribbon you can draw one or multiple points and assign a uniform value to them (Figure H.17). Each point is defined by a single click with the LMB. The group of points is completed by double clicking the LMB. Upon double clicking a popup appears in which you can assign a value to the points.



Add samples operation

Please specify the operation parameters

Value -10

Figure H.17: *Activating the ‘Add points’ operation, drawing them in the central map and assigning a value to them.*

H.5 Spatial operations

The spatial editor supports the following spatial operations (see also Figure H.18):

- Import (section H.5.1) - only for point clouds
- Crop (section H.5.2)
- Delete (section H.5.3)
- Set Value (section H.5.4)
- Contour (section H.5.5) - only for point clouds
- Gradient (section H.5.9)
- Interpolate (section H.5.10) - only for point clouds
- Smoothing (section H.5.11)
- Change single value (section H.5.12) - only for grid coverages
- Merge spatial data (section H.5.8) - only for grid coverages
- Copy to samples (section H.5.6) - only for grid coverages
- Copy to spatial data (section H.5.7) - only for grid coverages

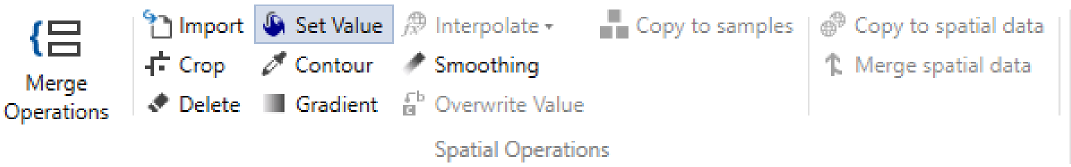


Figure H.18: *Overview of the available spatial operations in the ‘Spatial Operations’ ribbon*

The sections below provide a description of each operation.

H.5.1 Import point cloud

With the import operation you can import a point cloud for the selected spatial quantity. For this operation no geometry is required. The import operation is activated from the 'Spatial Operations' ribbon (Figure H.19). Upon importing a point cloud you are asked whether a coordinate transformation should be applied to the imported dataset (Figure H.20). By default it will be assumed that the imported data is in the same coordinate system as the model. If not, you can indicate from which to which coordinate system the data should be transformed. After import the point cloud is added to the operations stack (Figure H.21). The difference between this importer and importing a point cloud on the project level in the project tree (section H.2.2) is that for this importer the point cloud is directly assigned to the selected spatial quantity (e.g. model input) instead of being treated as a separate dataset.

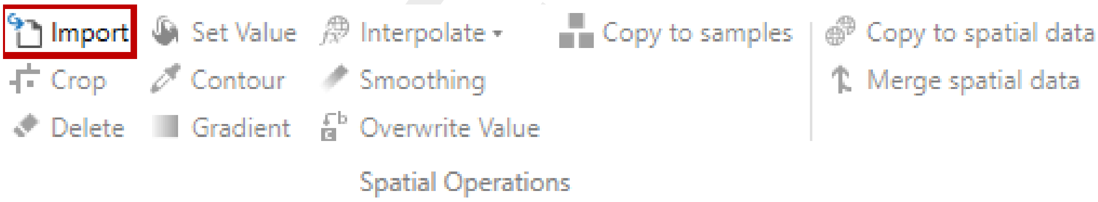
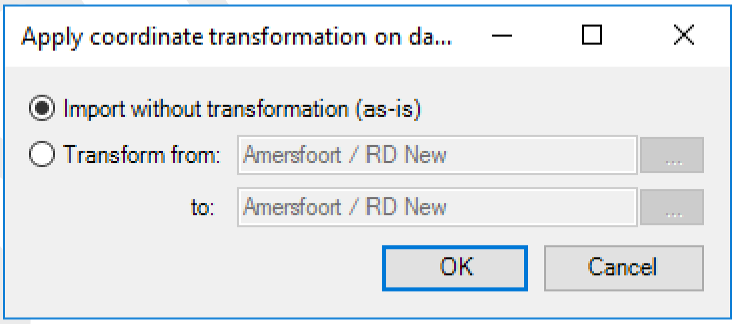


Figure H.19: Importing a point cloud using the 'Import' operation from the 'Spatial Operations' ribbon


Apply coordinate transformation on da...

● Import without transformation (as-is)

○ Transform from: Amersfoort / RD New

to: Amersfoort / RD New

Figure H.20: Option to perform a coordinate transformation on the imported point cloud

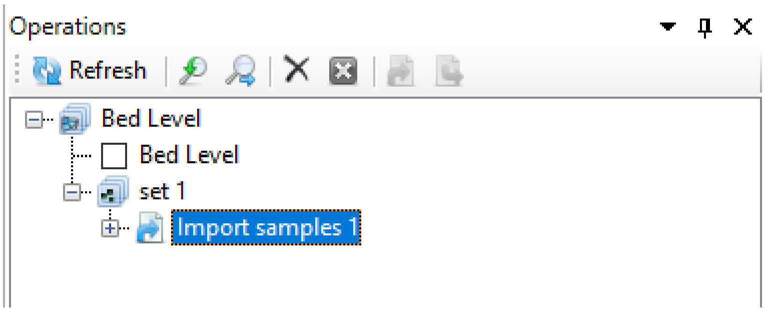


Figure H.21: Appearance of import point cloud operation in the operations stack

H.5.2 Crop

The crop operation crops a point cloud or coverage (depending on which one is active). The crop operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries. You can control which part of the data should be deleted by using polygons. If you provide a polygon outside the point cloud or coverage, all data will be deleted. For an example see Figure H.22.

After cropping (part of) the point cloud of coverage the operation is added to the operations stack (Figure H.23).

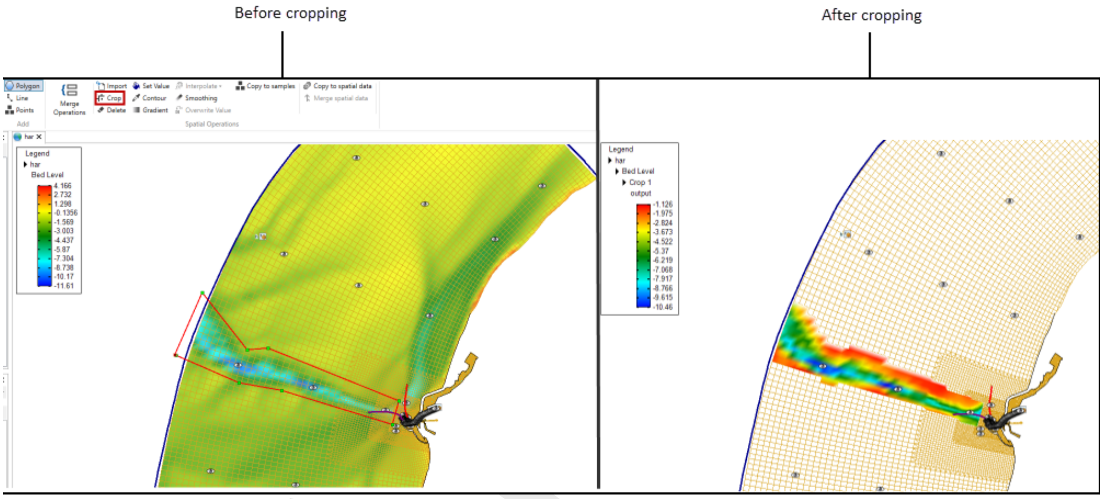


Figure H.22: *Performing a crop operation on a point cloud with a polygon using 'Crop' from the 'Spatial Operations' ribbon*

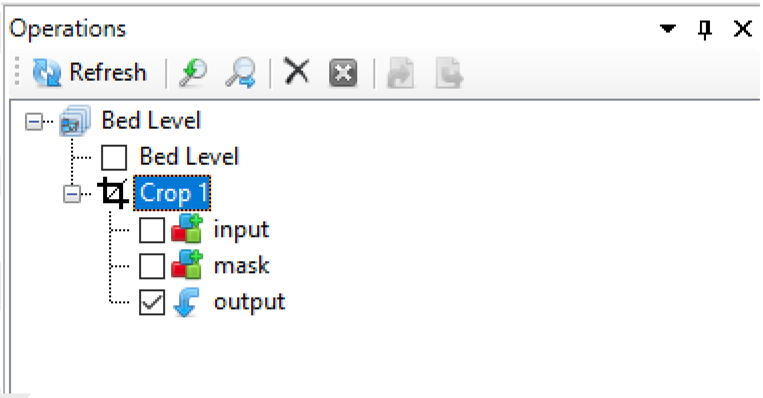


Figure H.23: *Appearance of crop operation in the operations stack*

H.5.3 Delete

The delete operation deletes (part of) a point cloud or coverage (depending on which one is active). The delete operation is activated from the 'Spatial Operations' ribbon. You can control which part of the data should be deleted by using polygons. If no polygons are provided, the total dataset will be deleted. For an example see Figure H.24. After erasing (part of) the point cloud or coverage the operation is added to the operations stack (Figure H.25).

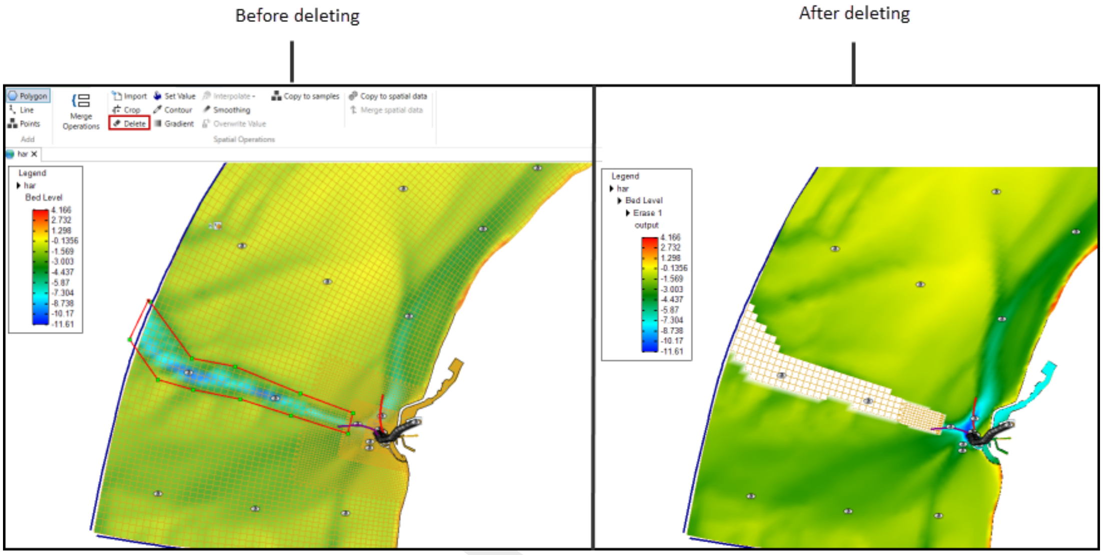


Figure H.24: *Performing an delete operation on a point cloud with a polygon using ‘Delete’ from the ‘Spatial Operations’ ribbon*

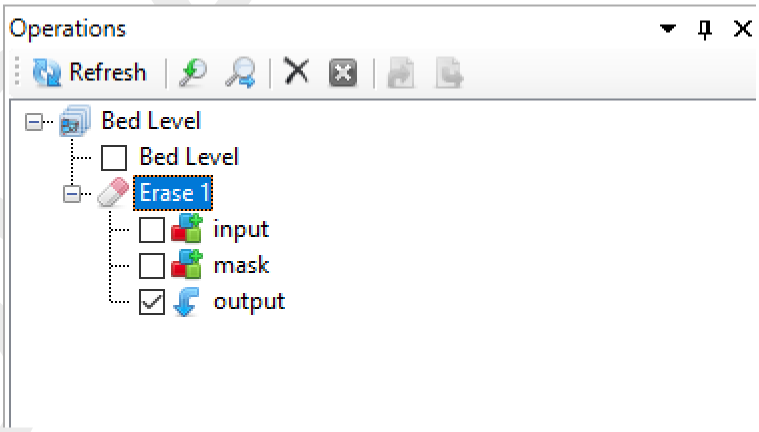


Figure H.25: *Appearance of delete operation in the operations stack*

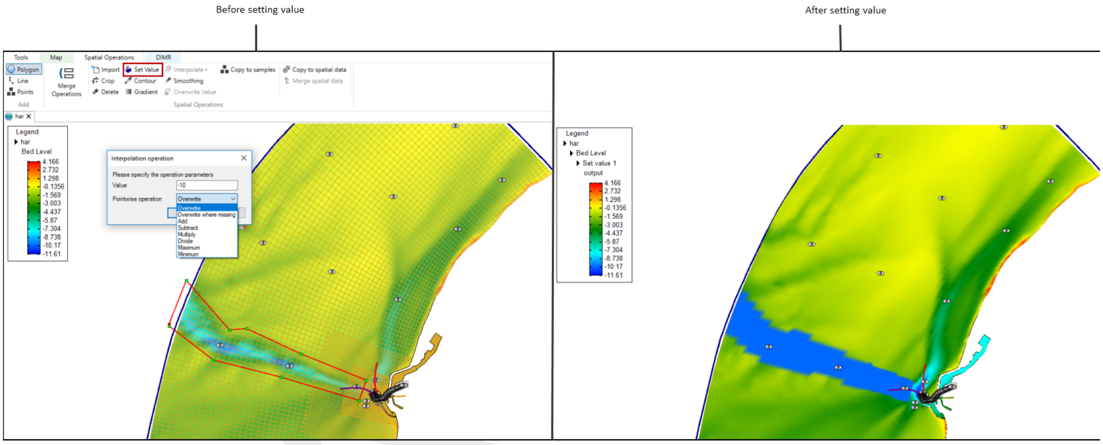
H.5.4 Set value

The set value operation assigns a value to a point cloud or coverage (depending on which one is active). The set value operation is activated from the ‘Spatial Operations’ ribbon and only available for polygon geometries or for the total data set if no polygon is provided. By assigning a value, the user can choose from the following operations:

- **Overwrite** : overwrites all existing points within the polygon (excluding no data values) with the uniform value
- **Overwrite where missing (only for coverages)** : overwrites all missing values within the polygon with the uniform value
- **Add** : Adds the uniform value to all existing points within the polygon (excluding no data values)
- **Subtract** : Subtracts the uniform value from all existing points within the polygon (excluding no data values)
- **Multiply** : Multiplies all existing points within the polygon (excluding no data values) with the uniform value
- **Divide** : Divides all existing points within the polygon (excluding no data values) by the uniform value
- **Maximum** : Sets all existing points within the polygon (excluding no data values) to the maximum of its current value and the uniform value

◇ **Minimum** : Sets all existing points within the polygon (excluding no data values) to the minimum of its current value and the uniform value

For an example see Figure H.26. After performing a set value operation to (part of) the point cloud or coverage the operation is added to the operations stack (Figure H.27).



Interpolation operation

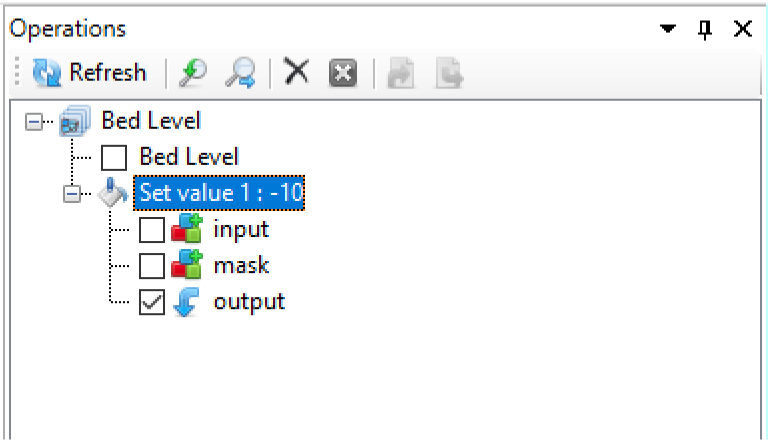
Please specify the operation parameters

Value: -10

Pointwise operation: Overwrite

Overwrite
Overwrite where missing
Add
Subtract
Multiply
Divide
Maximum
Minimum

***Figure H.26:** Performing a set value operation (e.g. overwrite) on a point cloud with a polygon using 'Set Value' from the 'Spatial Operations' ribbon*

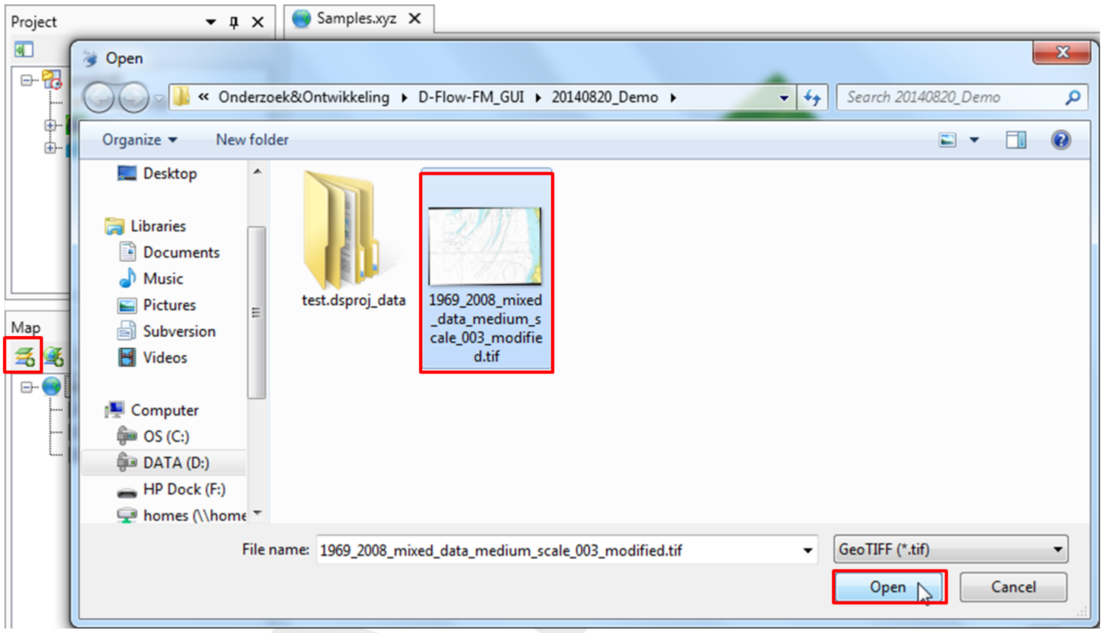


Bed Level
- □ Bed Level
- Set value 1 : -10
  - □ input
  - □ mask
  - ☑ output

***Figure H.27:** Appearance of set value operation in the operations stack*

H.5.5 Contour

The contour operation creates a point cloud with a uniform value along a line or polygon (depending on which one is active). The contour operation is activated from the 'Spatial Operations' ribbon. After drawing the lines or polygons you have to assign the uniform value (argument) and the sampling interval in m. This spatial operation can be useful to digitalize information from nautical charts for example. In this case you first have to import the nautical chart as a geotiff (Figure H.28), set the right map coordinate system (Figure H.29) and then use the contour operation Figure H.30. Sometimes the samples are created behind the geotiff. Then you can use the context menu on the samples layer in the Map tree to bring the samples to the front (Figure H.31). After applying the contour operation it is added to the stack (Figure H.32).



File name: 1969_2008_mixed_data_medium_scale_003_modified.tif

GeoTIFF (*.tif)

Figure H.28: *Import a nautical chart as a georeferenced tiff file*

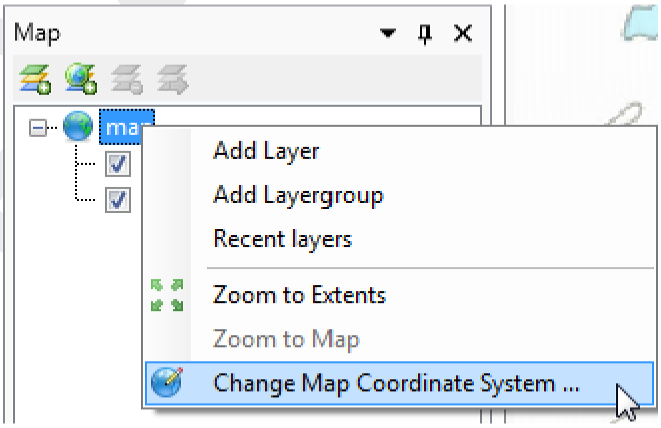
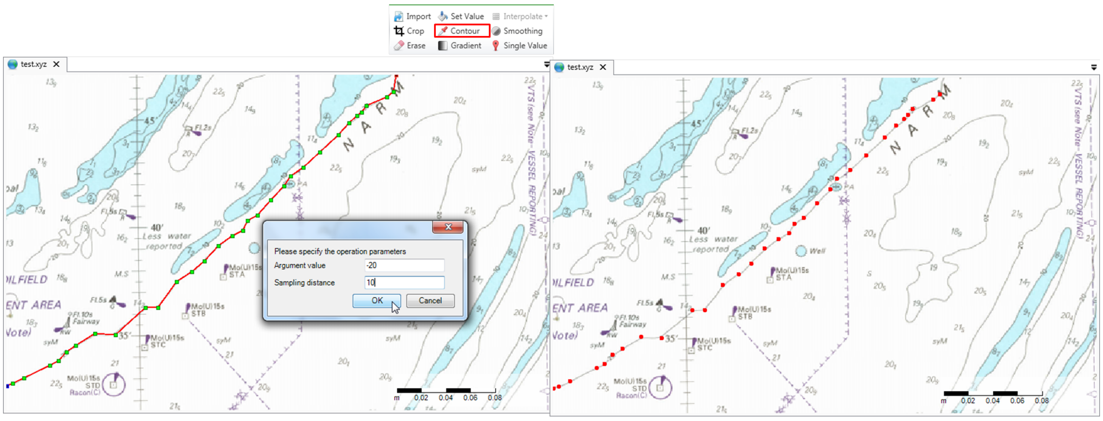


Figure H.29: *Set the right map coordinate system for the geotiff*



Please specify the operation parameters

Argument value -20

Sampling distance 10

Figure H.30: *Performing a contour operation on a nautical chart using lines to define the contours and 'Contour' from the 'Spatial Operations' ribbon*

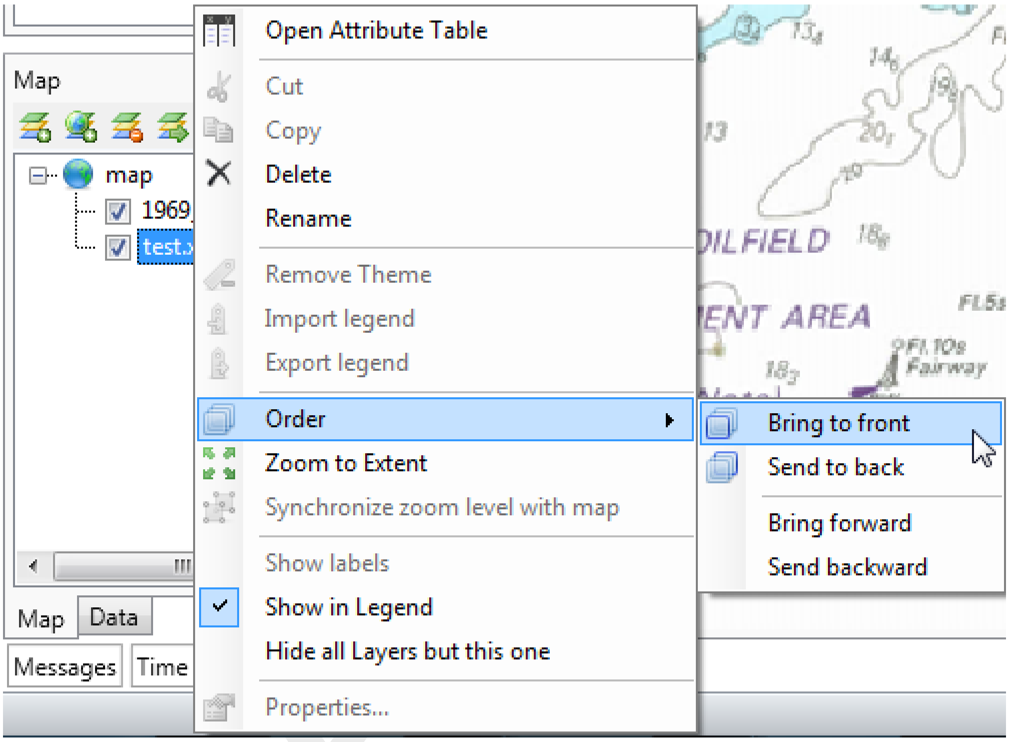


Figure H.31: *Bring the sample set to the front if it appears behind the nautical chart*

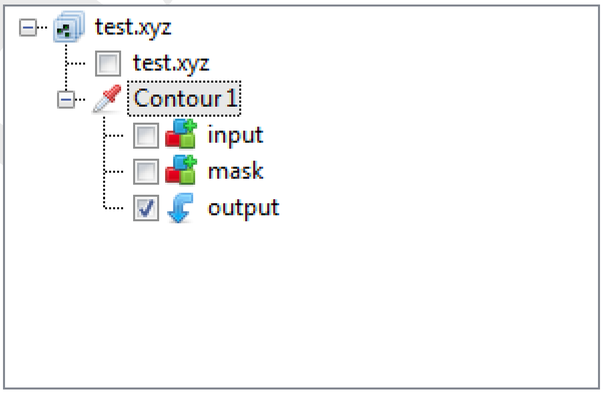
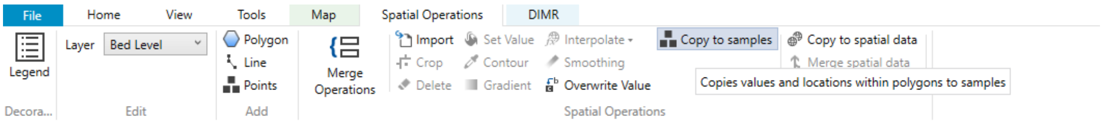


Figure H.32: *Appearance of contour operation in the operations stack*

H.5.6 Copy to samples

This operation converts the currently selected grid coverage to a sample set, which becomes that starting point of a new subset. If polygons have been drawn, the operation will confine the copy to the interiors. The operation will not convert missing values to samples. Note that the operation does keep a reference to the original copied grid coverage; if it is changes by a re-evaluation of the stack, the changes will affect the output point cloud.



Copies values and locations within polygons to samples

Figure H.33: *Applying the copy to samples operation*

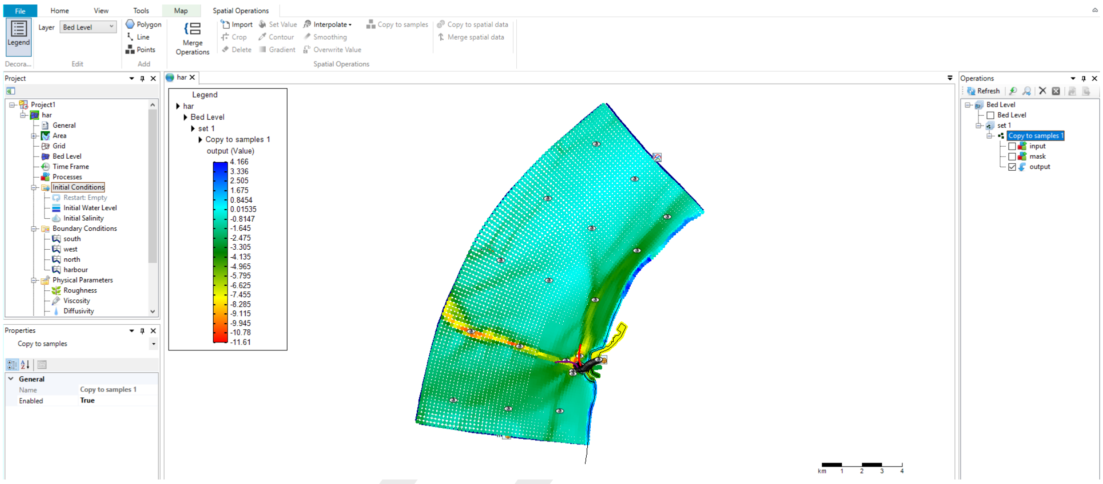
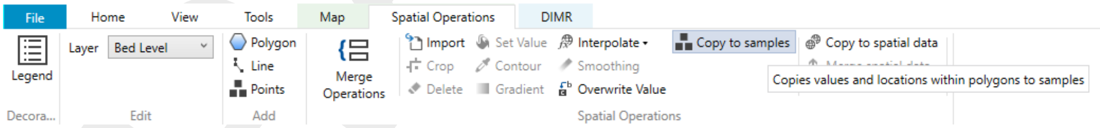


Figure H.34: Copy to samples operation result

H.5.7 Copy to spatial data

This operation simply clones the grid coverage, starting a new subset with a snapshot of the currently selected operation output. Similarly to H.5.6, it keeps a reference to the input data and will perform the clone again upon re-evaluation.



Copies values and locations within polygons to samples

Figure H.35: Applying the copy spatial data operation

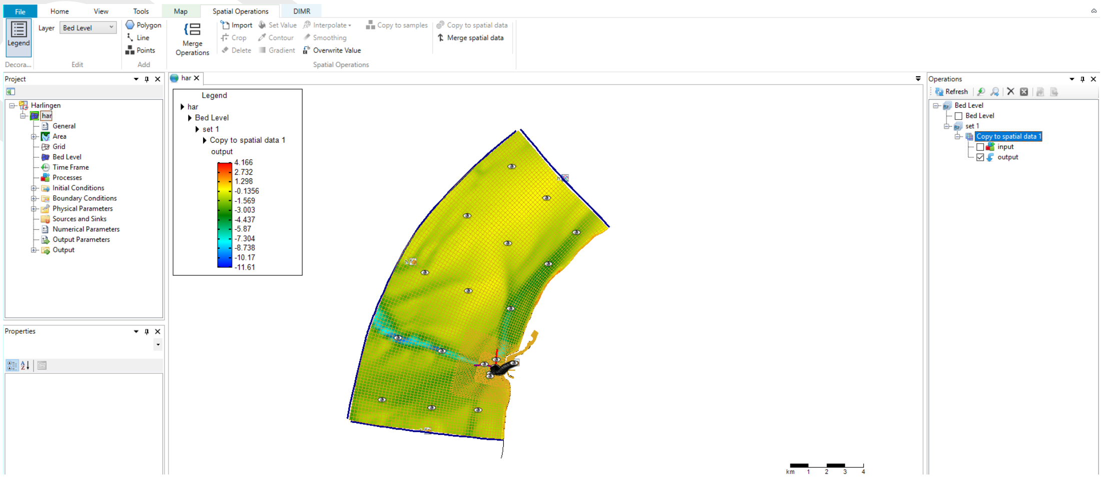


Figure H.36: Copy spatial data operation result

H.5.8 Merge spatial data

Whenever a subset contains a grid coverage as its editing data (after applying e.g. H.5.7) on the same grid as the main operation set, its result can be combined with the main set by applying this operation, similarly to the interpolation operation for sample sets, discussed below. Combining the grid coverages can be achieved with the usual point-wise methods.

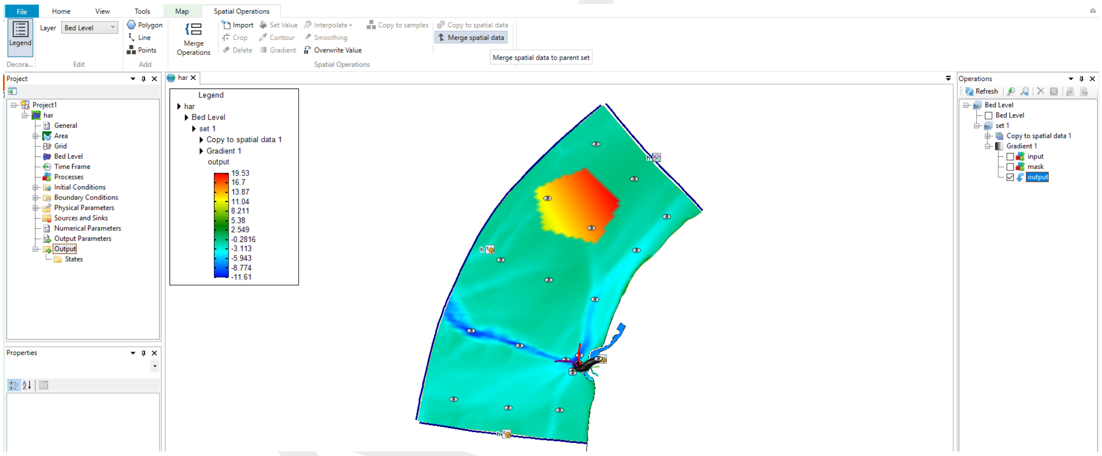
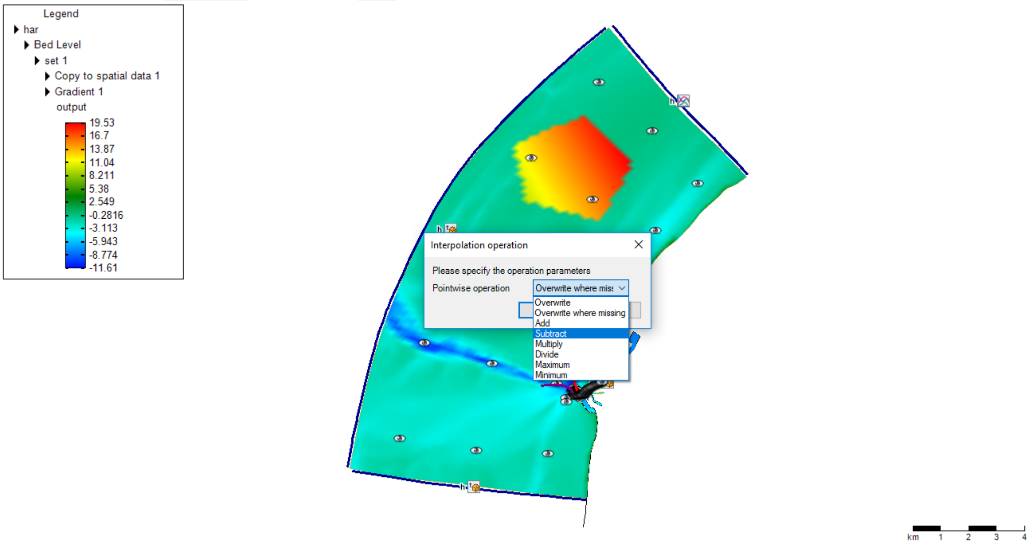


Figure H.37: Activating the merge spatial data tool from the ribbon



Interpolation operation

Please specify the operation parameters

Pointwise operation: Overwrite where miss

Figure H.38: The merge operation requests a pointwise combination method

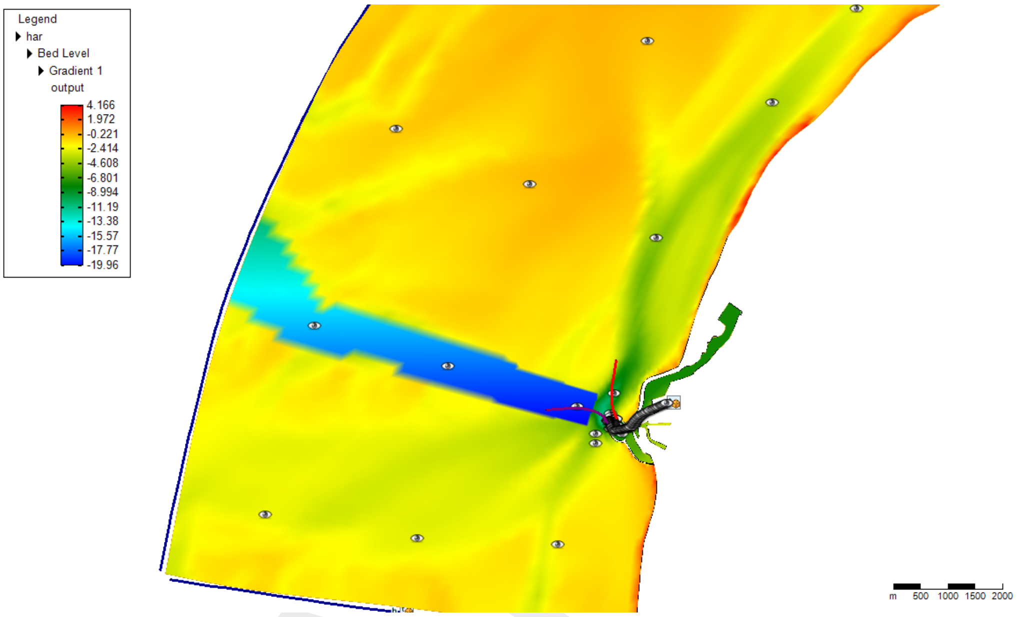
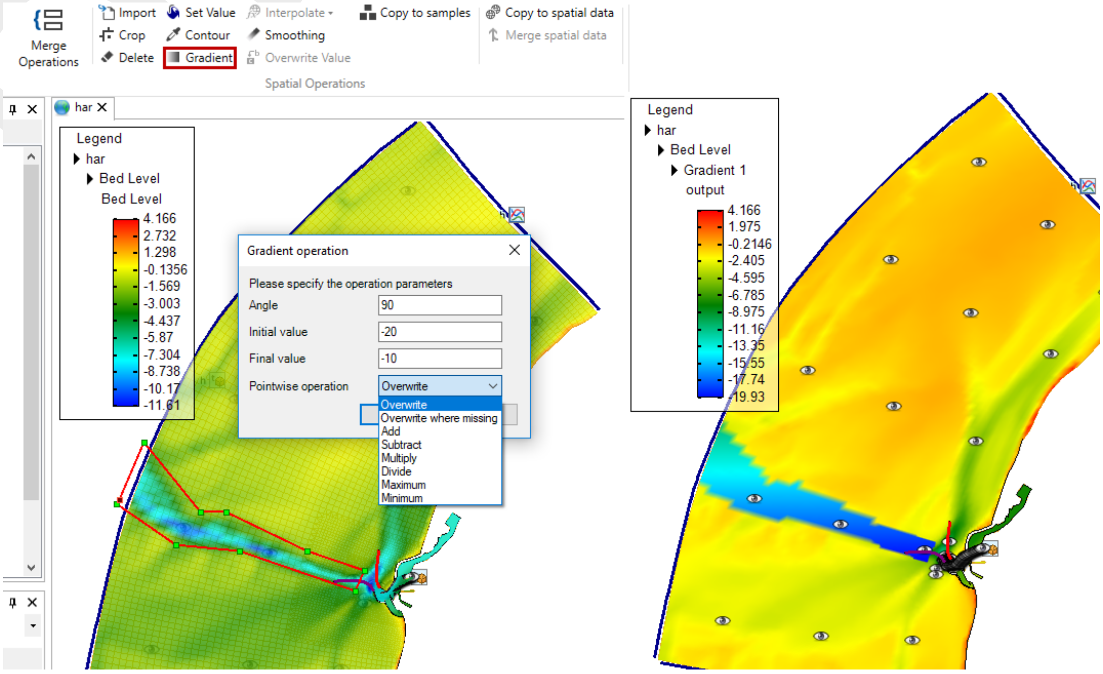


Figure H.39: Resulting grid coverage

H.5.9 Gradient

The gradient operation applies a gradient to a point cloud or grid coverage (depending on which one is active). The gradient operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries or for the total data set if no polygon is provided. You have to assign the initial (start) value, the final (end) value and the going to angle (according to the Cartesian convention with 0 degrees is East, 90 degrees is North, etc **Note:** This is not working properly yet). For an example see Figure H.40. After applying a gradient to (part of) the point cloud or coverage the operation is added to the operations stack (Figure H.41).



Gradient operation

Please specify the operation parameters

Angle: 90

Initial value: -20

Final value: -10

Pointwise operation: Overwrite

Figure H.40: Performing a gradient operation on a point cloud with a polygon using 'Gradient' from the 'Spatial Operations' ribbon

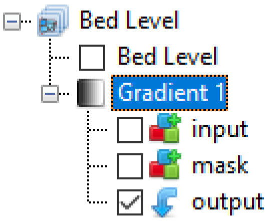
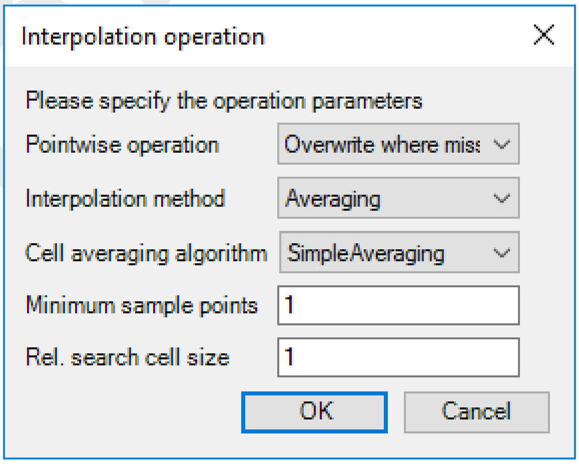


Figure H.41: *Appearance of gradient operation in the operations stack*

H.5.10 Interpolate

The interpolate operation is the way to get sample set(s) to a grid or network (e.g. coverage). In the operation stack this means that we are actively switching from working on a point cloud (helping to construct the coverage) to working on the selected coverage (or spatial quantity). The interpolation is performed on the data within a polygon or polygons (if provided) or all the data (if no polygons are provided). The interpolate operation can be performed on a single (selected) sample set or on multiple sample sets. Both methods are discussed below. The methods for interpolation are either

- Triangulation: performs a Delauney triangulation on the sample point set before projecting onto the grid.
- Averaging: combines sample points within a possible enlarged cell according to an algorithm of choice. The user can set the search cell expansion factor (rel. search cell size) and a threshold for the number of sample points within a cell (minimum sample points), see Figure H.42.


Interpolation operation

Please specify the operation parameters

Pointwise operation: Overwrite where mis:

Interpolation method: Averaging

Cell averaging algorithm: SimpleAveraging

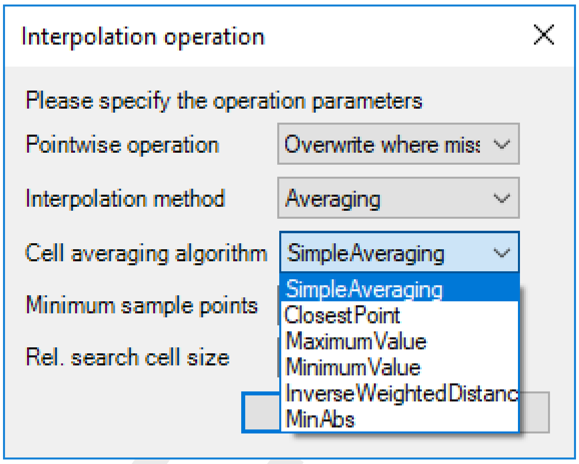
Minimum sample points: 1

Rel. search cell size: 1

Figure H.42: *Interpolation Operation options*

Seven Cell averaging algorithms can be chosen by the user; see Figure H.43. These algorithms are explained below:

- SimpleAveraging: bilinear interpolation is applied, which uses a distance-weighted average of the surrounding samples. The closer the sample point the larger the weighted value.
- ClosestPoint: the value of the closest sample inside the search area is taken.
- MaximumValue: the maximum value of the samples inside the search area is taken.
- MinimumValue: the minimum value of the samples inside the search area is taken.
- InverseWeightedDistance: Instead of a distance-weighted average (w) in case of the inverse of SimpleAveraging, the distance-weighted average (1/w) is taken. The closer the sample point the smaller the weighted value.
- MinAbs: the minimum of the absolute values of the samples inside the search area is taken.
- KdTree: This is an obsolete option, which will be removed in a future release.



Interpolation operation

Please specify the operation parameters

Pointwise operation: Overwrite where mis:

Interpolation method: Averaging

Cell averaging algorithm: SimpleAveraging

Minimum sample points

Rel. search cell size

Figure H.43: *Averaging options*

By default, the interpolation will only overwrite missing values in the gridded data set. However, if the grid coverage already contains values, the user may choose to overwrite or combine the data by a pointwise arithmetic operation.

Interpolate single (selected) set

To perform interpolation on a single sample set, select the sample set (i.e. ‘set1’) in the operation stack and press ‘Interpolate’ in the ‘Map’ ribbon (Figure H.44). Since no polygon is provided in this example, all the samples will be interpolated to the grid. Use polygons if you would like to have more control over the interpolation. After the interpolation the operation is added to the operations stack (Figure H.45). Please note that after performing the interpolation the workflow in the stack is shifting from the sample set (i.e. ‘set1’ - which was a side step to construct the coverage) to the coverage (i.e. ‘bed level’).

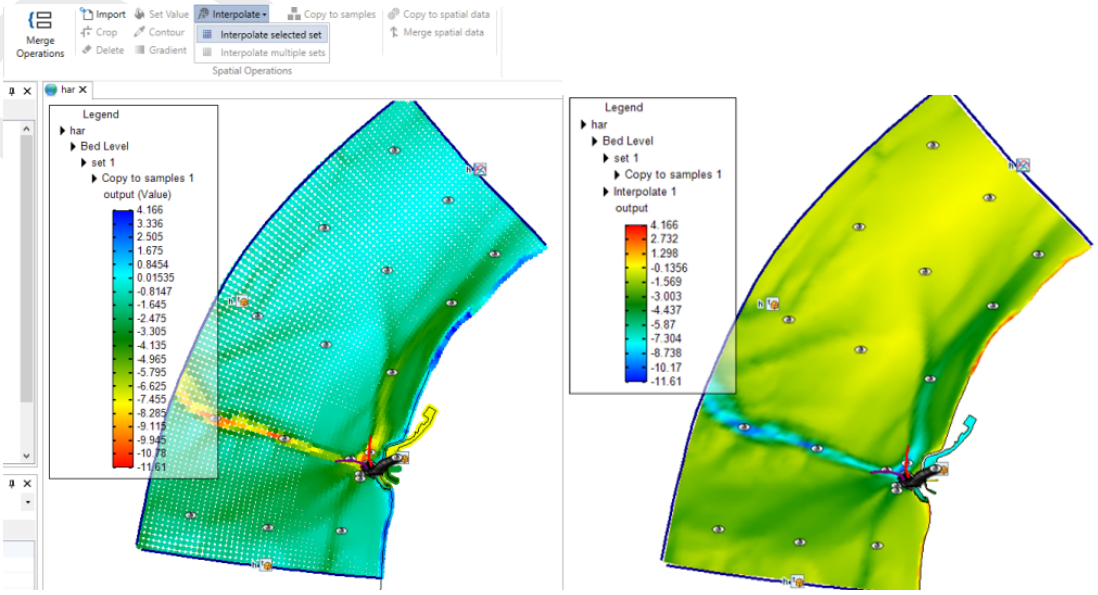
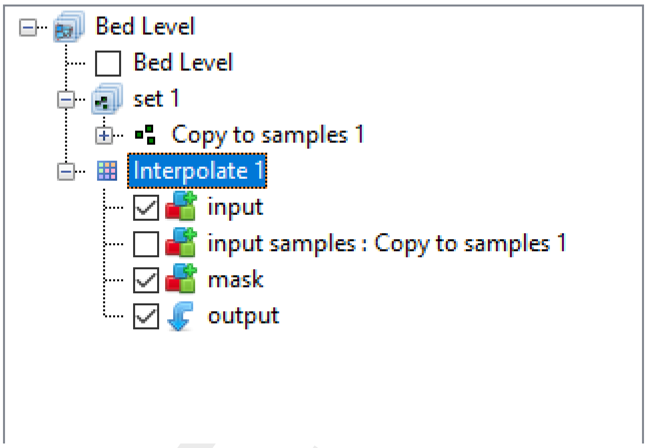


Figure H.44: *Performing an interpolation operation on a single sample set (without using a polygon) using ‘Interpolate’ from the ‘Spatial Operations’ ribbon*



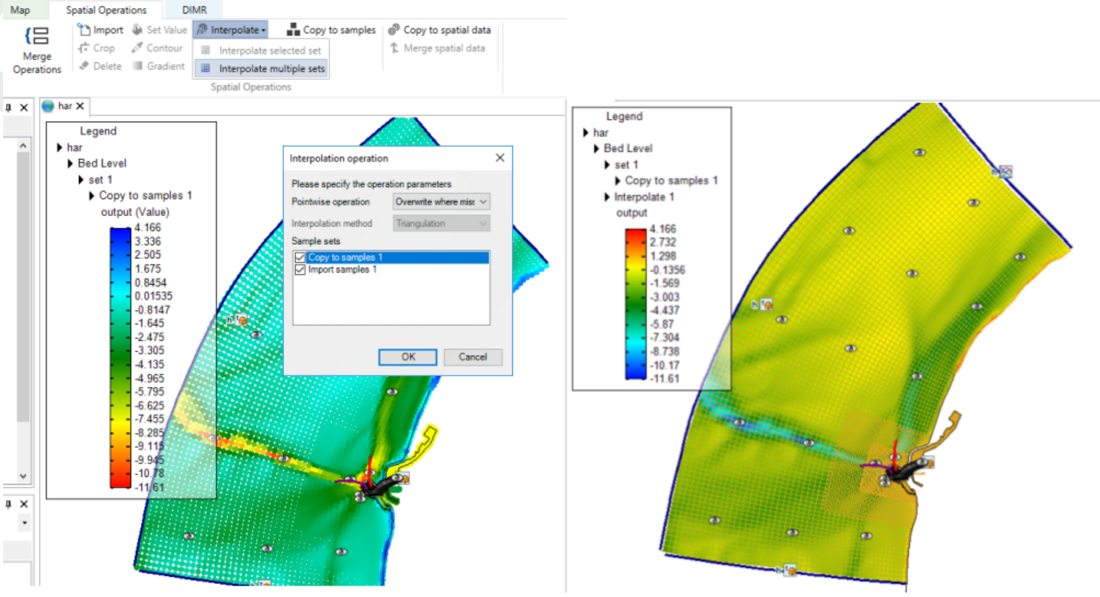
- ⊟ Bed Level
  - □ Bed Level
  - ⊟ set 1
    - ⊞ Copy to samples 1
  - ⊟ Interpolate 1
    - ☑ input
    - □ input samples : Copy to samples 1
    - ☑ mask
    - ☑ output

Figure H.45: *Appearance of interpolation of ‘set1’ to the coverage ’bed level’ in the operations stack*

Interpolate multiple sets

To perform interpolation on multiple sample sets, select the active coverage (i.e. ‘bed level’) in the operation stack and press ‘Interpolate’ in the ‘Map’ ribbon (Figure H.46). In the popup you can select which sample sets to include in the interpolation (in this example both). Since no polygon is provided, all the samples (from the two sets) will be interpolated to the grid. Use polygons if you would like to have more control over the interpolation. After the interpolation the operation is added to the operations stack (Figure H.47). Again note that after performing the interpolation the workflow in the stack is shifting from the sample set (which was a side path to construct the coverage) to the coverage (i.e. bed level).

Note: Please note that interpolation of multiple sample sets can also be achieved by importing/combining different sample sets into the same set in the stack instead of using two separate sets. In this case you can just interpolate the single (selected) set.



Interpolation operation

Please specify the operation parameters

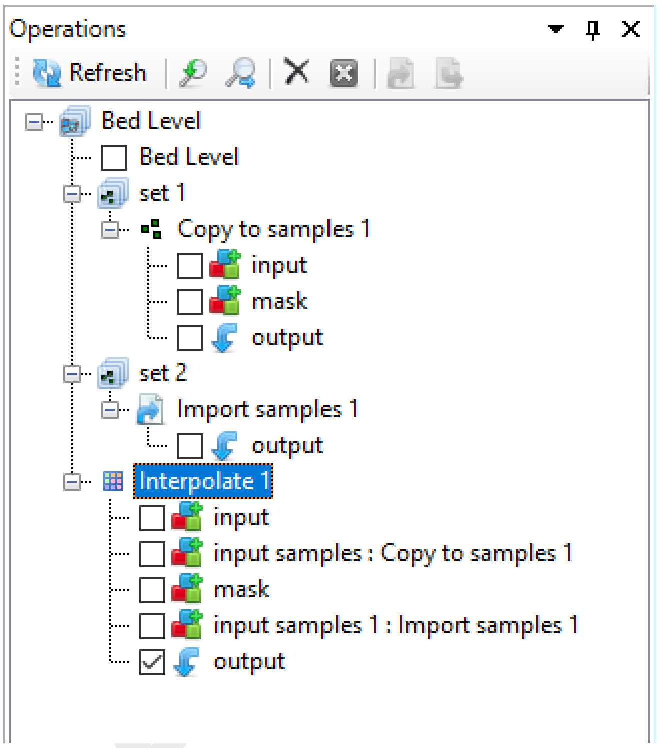
Pointwise operation: Overwrite where mis:

Interpolation method: Triangulation

Sample sets

☑ Copy to samples 1
☑ Import samples 1

Figure H.46: *Performing an interpolation operation on multiple sample sets (without using a polygon) using ‘Interpolate’ from the ‘Spatial Operations’ ribbon*

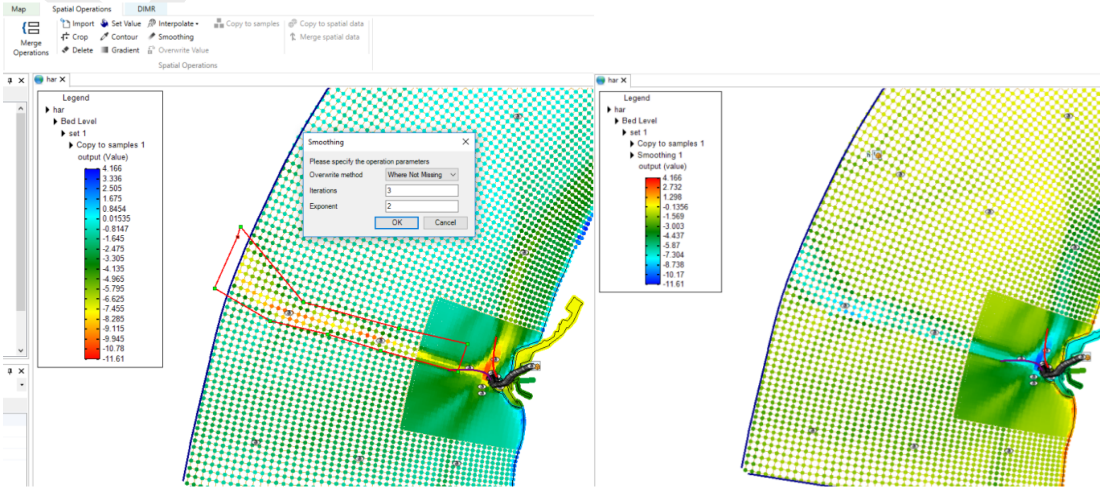


- Bed Level
  - ☐ Bed Level
  - set 1
    - Copy to samples 1
      - ☐ input
      - ☐ mask
      - ☐ output
  - set 2
    - Import samples 1
      - ☐ output
  - Interpolate 1
    - ☐ input
    - ☐ input samples : Copy to samples 1
    - ☐ mask
    - ☐ input samples 1 : Import samples 1
    - ☑ output

Figure H.47: Appearance of interpolation of 'set1' and 'set2' to the coverage 'bed level' in the operations stack

H.5.11 Smoothing

The smoothing operation smooths out (steep) gradients in a point cloud or coverage (depending on which one is active). The smoothing operation is activated from the 'Spatial Operations' ribbon and only available for polygon geometries or for the total data set if no polygon is provided. You have to assign the smoothing exponent and number of smoothing steps. The higher the exponent and the number of smoothing steps, the heavier the smoothing. For an example see Figure H.48. After applying smoothing to (part of) the point cloud or coverage the operation is added to the operations stack (Figure H.49).



Smoothing

Please specify the operation parameters

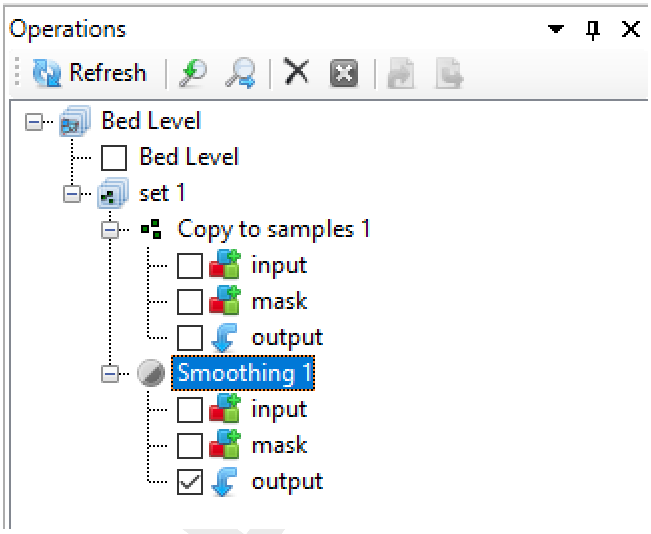
Overwrite method: Where Not Missing

Iterations: 3

Exponent: 2

OK Cancel

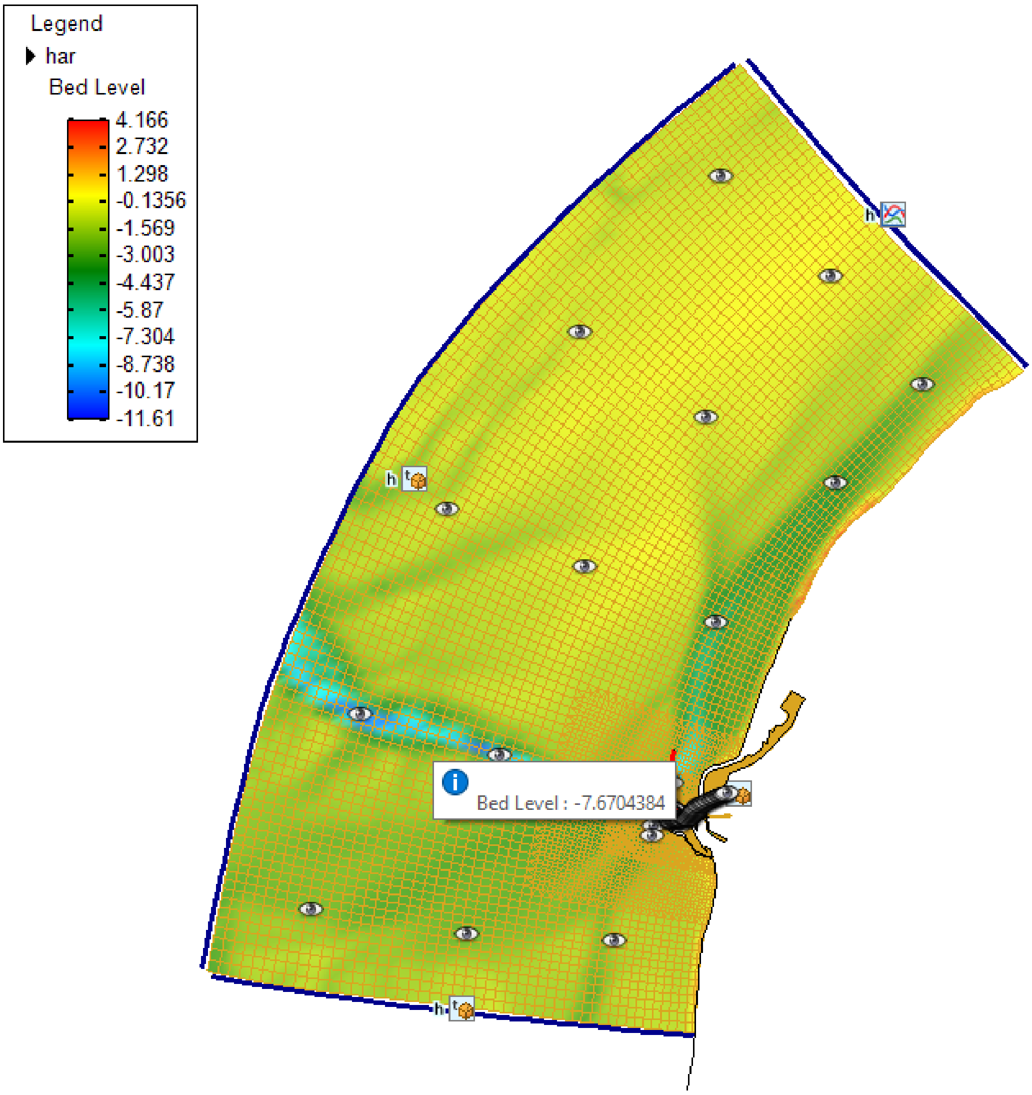
Figure H.48: Performing a smoothing operation on a point cloud with a polygon using 'Smoothing' from the 'Spatial Operations' ribbon



Operations

- Bed Level
  - □ Bed Level
  - set 1
    - Copy to samples 1
      - □ input
      - □ mask
      - □ output
    - Smoothing 1
      - □ input
      - □ mask
      - ☑ output

***Figure H.49:** Appearance of smoothing operation in the operations stack*

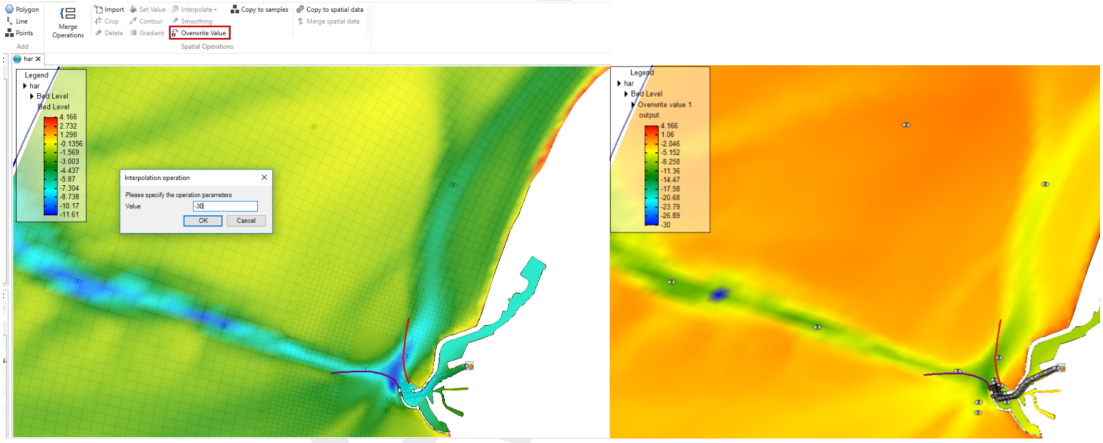


Bed Level : -7.6704384

***Figure H.50:** The cursor for the overwrite operation showing the value of the closest coverage point*

H.5.12 Overwrite (single) value

The ‘overwrite (single) value’ operation allows you to edit single values on the active coverage after the interpolation. The ‘overwrite (single) value’ operation is activated from the ‘Spatial Operations’ ribbon. There is no geometry required for this operation. Upon selecting the operation from the ribbon a cursor will become active showing the coverage value closest to the cursor in a tooltipstring (Figure H.50). Upon clicking LMB a popup appears in which you can overwrite the value of this coverage point Figure H.51. After applying the overwrite operation it is added to the operations stack (Figure H.52).



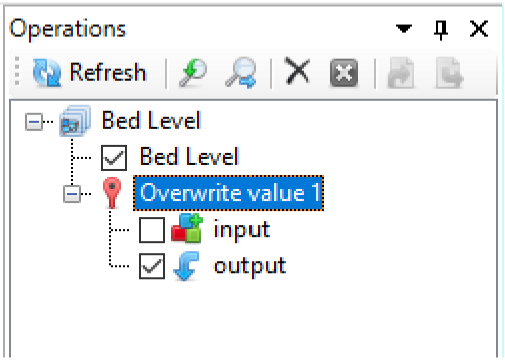
Interpolation operation

Please specify the operation parameters

Value -30

OK Cancel

Figure H.51: Performing an overwrite operation on a coverage point using ‘Single Value’ from the ‘Spatial Operations’ ribbon



☑ Bed Level

Overwrite value 1

□ input

☑ output

Figure H.52: Appearance of overwrite operation in the operations stack

H.6 Operation stack

The operation stack keeps track of the workflow of spatial operations that you performed. This helps you to make transparent how you arrived at your ‘final’ dataset without having to save all the intermediate datasets (steps) separately. Moreover, the stack is reproducible and easily editable without having to start all over again. This section describes the stack workflow (section H.6.1), how to edit operation properties (section H.6.2), how to enable/disable (section H.6.3), delete (section H.6.4), refresh(section H.6.5) operations, quick links (section H.6.6) and import/export functionality (section H.6.7).

Note: Currently, the stack is saved in the User Interface project upon saving the project. The next time you open the project, the stack will reappear. The stack is not (yet) saved in a human readable/editable file.

H.6.1 Stack workflow

Upon performing a spatial operation, the ‘Operations’ panel will open (see Figure H.53) with the operations stack (tree). The stack first shows on which point cloud or coverage you are working (in this example ‘bed level’). Subsequently, all the operations on this dataset are listed. For each operation you can inspect what the input, mask (e.g. the geometry used for the operation) and output are for the operation (Figure H.54). By default the stack continues from the last operation that you performed. If you wish to work on a different dataset or operation within a dataset, you have to select that dataset or operation in the ‘Operations’ panel with the LMB.

When working on a coverage, point clouds (or sets) can be used to construct the coverage. In that case the stack jumps from the ‘trunk’ to a ‘branch’ and the subsequent operations are performed on the point cloud (see Figure H.53). By selecting the set or coverage in the ‘Operations’ panel you determine on which dataset you are working. The interpolate operation (section H.5.10) allows you to bring data from the point cloud (branch) to the coverage (trunk). See also Figure H.53.

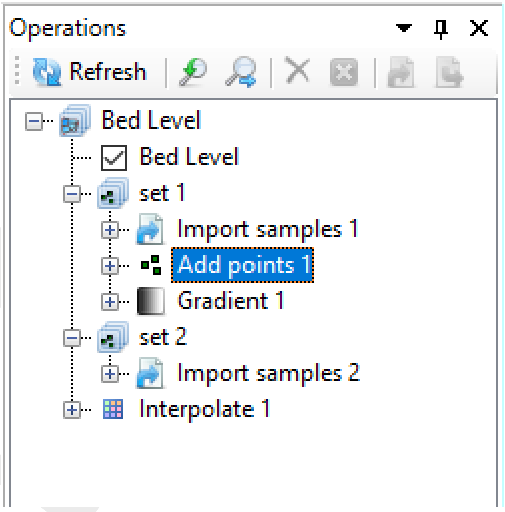


Figure H.53: The ‘Operations’ panel with the operations stack. In this example ‘bed level’ is the coverage (e.g. trunk) that is edited. The point clouds ‘set 1’ and ‘set 2’ (e.g. branches) are used to construct the ‘bed level’ coverage.

H.6.2 Edit operation properties

For each operation that you performed the properties (such as the value or ‘Pointwise operation’ of a ‘Set Value’ operation) can be edited using the ‘Properties’ window (Figure H.55). **Note:** Please note that the mask of an operation cannot (yet) be edited. By editing the operation properties the operation stack becomes ‘out of sync’ and has to be refreshed for the changes to become active (see section H.6.5).

H.6.3 Enable/disable operations

You can (temporarily) enable/disable operations by selecting the operation and pressing boxed cross icon in the stack menu (Figure H.56). Upon disabling an operation the operation will be made grey in the stack and the operation is not taken into account anymore upon evaluation of the overall result. The result of disabling an operation is not directly activated. This is indicated in the stack with the ’out of sync’ exlamation mark (Figure H.56). You need to refresh the stack (see section H.6.5) for the changes to become active.

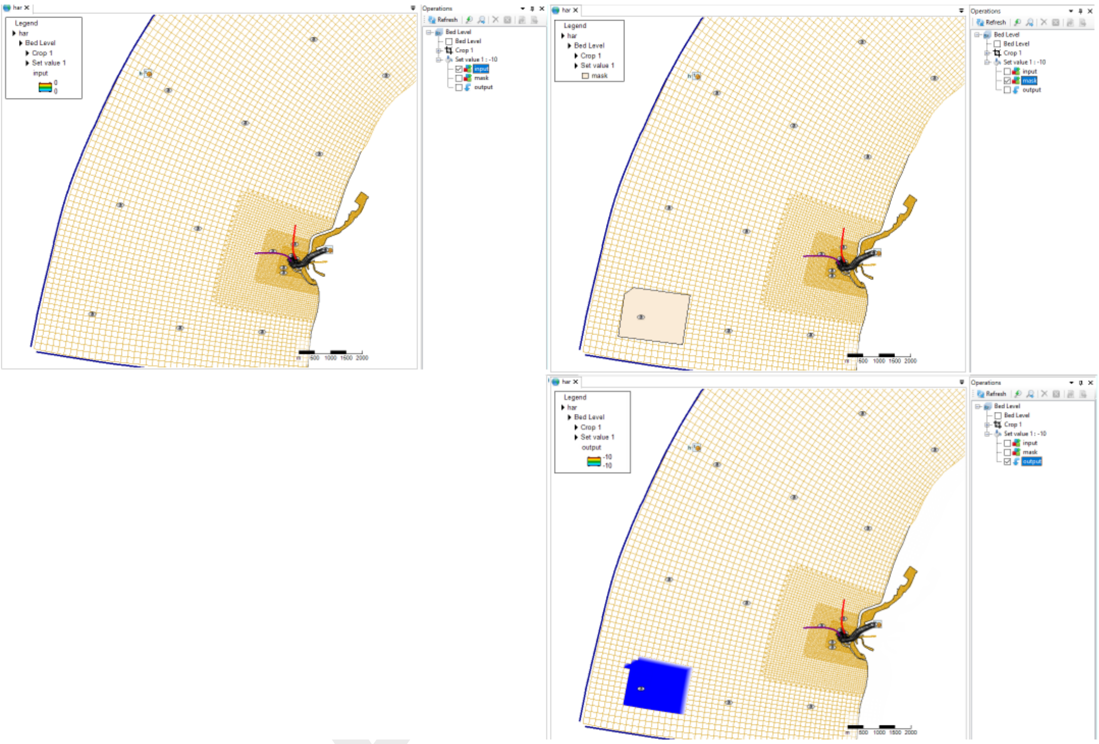


Figure H.54: *Input for the operation (top panel), mask for the operation (middle panel) and output of the operation (bottom panel)*

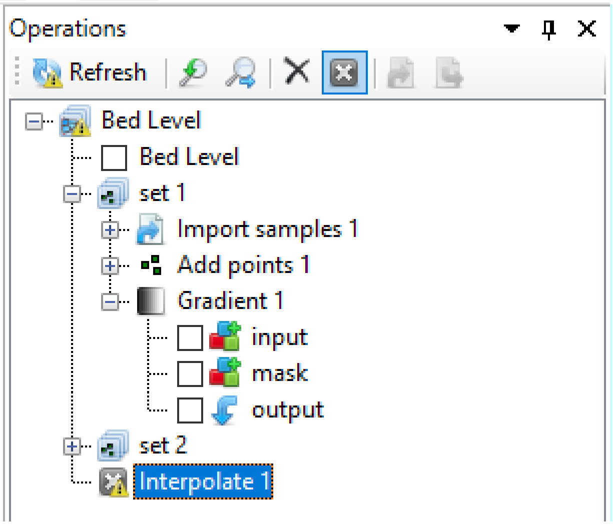
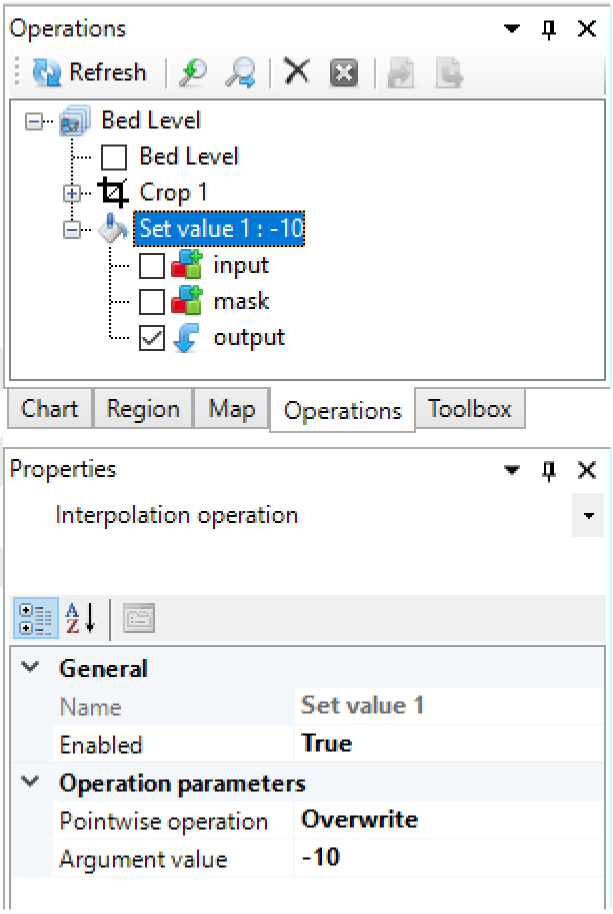


Figure H.56: *Disabling an operation using the boxed cross icon in the stack menu. The operation will become grey. Note the exlamation marks marking the stack ‘out of sync’.*



Operations

Refresh

- Bed Level
  - □ Bed Level
  - Crop 1
  - Set value 1 : -10
    - □ input
    - □ mask
    - ☑ output

Properties

Interpolation operation

General	
Name	Set value 1
Enabled	True
Operation parameters	
Pointwise operation	Overwrite
Argument value	-10

Figure H.55: *Editing the value or ‘Pointwise operation’ of a ‘Set Value’ operation using the properties panel*

H.6.4 Delete operations

To delete an operation permanently you have to select the operation and either press the cross icon (Figure H.57) or use the context menu and select delete (Figure H.58). The operation will be removed from the stack. The result of deleting an operation is not directly activated. This is indicated in the stack with the 'out of sync' exlamation mark. You need to refresh the stack (see section H.6.5) for the changes to become active.

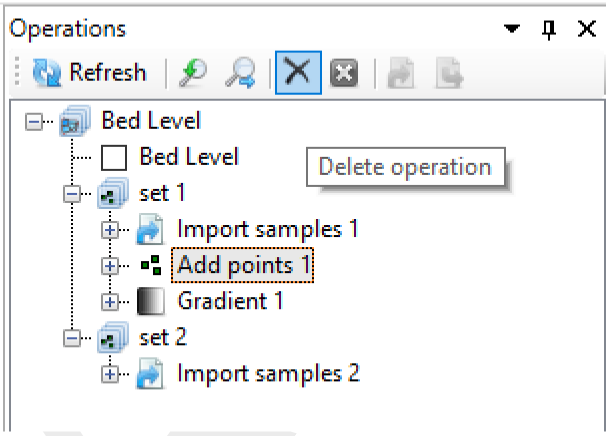


Figure H.57: Removing an operation from the stack using the cross icon in the stack menu

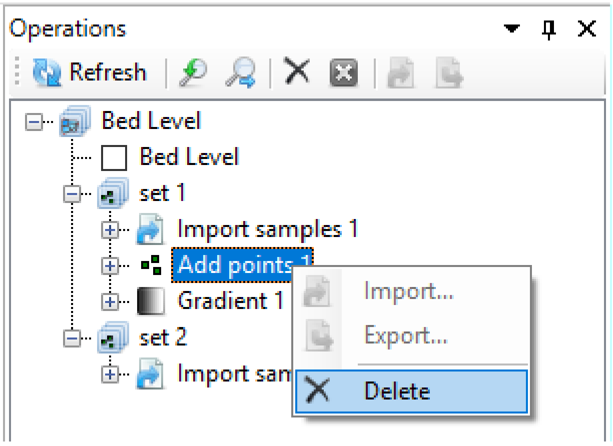


Figure H.58: Removing an operation from the stack using the context menu on the selected operation

H.6.5 Refresh stack

When the stack is marked ‘out of sync’ by exclamation marks, you can refresh the stack by pressing the ‘Refresh’ button for the changes to become active (Figure H.59). Upon refreshing the stack all the (enabled) operations in the stack will be (re-)evaluated. **Note:** Please note that refreshing the stack can take some time when large datasets are (re-)evaluated!

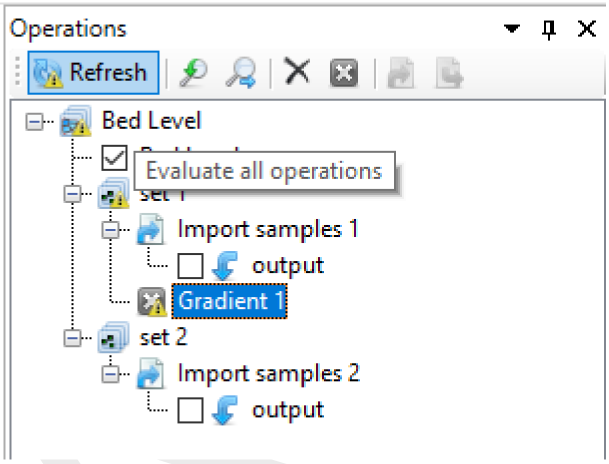


Figure H.59: Refresh the stack using the ‘Refresh’ button so that all operation are (re-) evaluated

H.6.6 Quick links

The stack menu contains two quick links to quickly show the original dataset (e.g. where you started from, Figure H.60) and the end result of the spatial operations (Figure H.61).

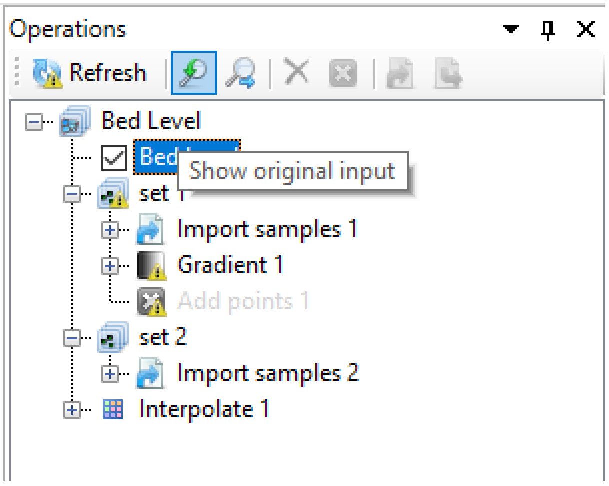


Figure H.60: Quick link to the original dataset before performing any spatial operations

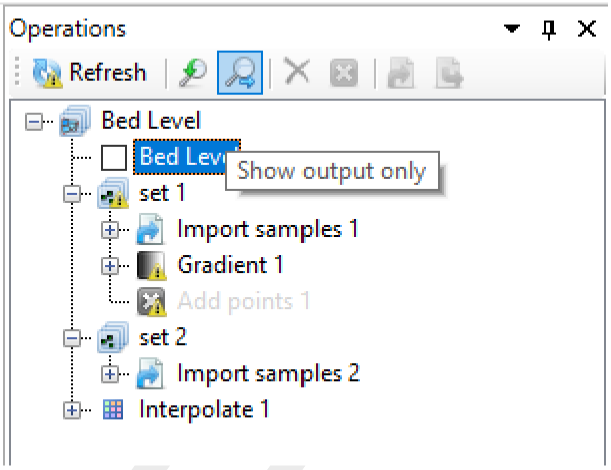


Figure H.61: *Quick link to the output after performing all (enabled) operations*

H.6.7 Import/export

Note: Importing and exporting data into or from the stack is still under construction

Deltares systems

PO Box 177
2600 MH Delft
Boussinesqweg 1
2629 HV Delft
The Netherlands

+31 (0)88 335 81 88
software@deltares.nl
www.deltares.nl/software