Deltares Integrated Model Runner

DIMR

Dutch Delta Systems

Technical Reference Manual

DIMR

Technical Reference Manual

Version: 1.0
Revision: 48961

16 July 2024

DIMR, Technical Reference Manual

Published and printed by:

Deltares	telephone: +31 88 335 82 73
Boussinesqweg 1	e-mail: Information
2629 HV Delft	www: Deltares
P.O. 177	
2600 MH Delft	
The Netherlands	

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

Contents

List of Tables

List of Figures

1 A guide to this manual

1.1 Introduction

This Technical Reference Manual concerns the module Deltares Integrated Model Runner (DIMR).

DIMR is a small programme to run combinations of simulation cores, components, ensuring that data is exchanged at the right timing. In a DIMR configuration file, the user can describe how the components should be combined and what data has to be exchanged.

1.2 Overview

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

Chapter 2: Basic functionality describes the basics on DIMR.

Chapter 4: Iterative coupling describes a more sophisticated way to couple modules: the implementation of the iterative coupling between D-Flow 1D and D-Flow FM.

1.3 Manual version and revisions

This manual applies to:

◇ SOBEK 3 (version 3.5 and higher)
◇ D-HYDRO Suite (version 2016.2 and higher)
◇ Delft3D Flexible Mesh Suite (version 2017 and higher)

1.4 Changes with respect to previous versions

This is the first edition.

2 Basic functionality

2.1 Introduction

DIMR stands for Deltares Integrated Model Runner. The module is dedicated to deploy modules that simulate physical phenomena, such as hydraulics or waves, and the exchange of data between them.

In GUI framework all so-called *Integrated model* runs are executed via DIMR. The configuration is given in the <DIMR_config.xml>. An example is given in Appendix A: DIMR configuration file (XML-format).

DIMR can also be executed from the command-line. To do this, an export is needed, see **?**. In short:

- ◇ RightMouseClick in the **Project** window on the appropriate <Integrated model> and choose *Export. . . .*
- ◇ Select *DIMR configuration*
- ◇ Press the *OK* button; and
- ◇ specify the output directory and file name

Now DIMR can be started with the exported <DIMR_config.xml>, see section 2.2: Running a computation via a console/command box.

2.2 Running a computation via a console/command box

The following recipe can be used for both Windows and Linux computations (modifications for Linux are in notes below the recipe):

- ◇ Build your model using the GUI framework.
- ◇ Export your model following the description in section 3.1.
- ◇ Locate the file <run_dimr.bat> on your system. It is in the installation folder and then the path <\plugins\DeltaShell.Dimr\kernels\x64\scripts>. (One directory up is a set of folders, one for each component.)
- ◇ Open a console/command box in the directory where you placed your exported model. The DIMR configuration file should be in this directory, see figure Figure 2.1
- ◇ Execute <run_dimr.bat> in this location

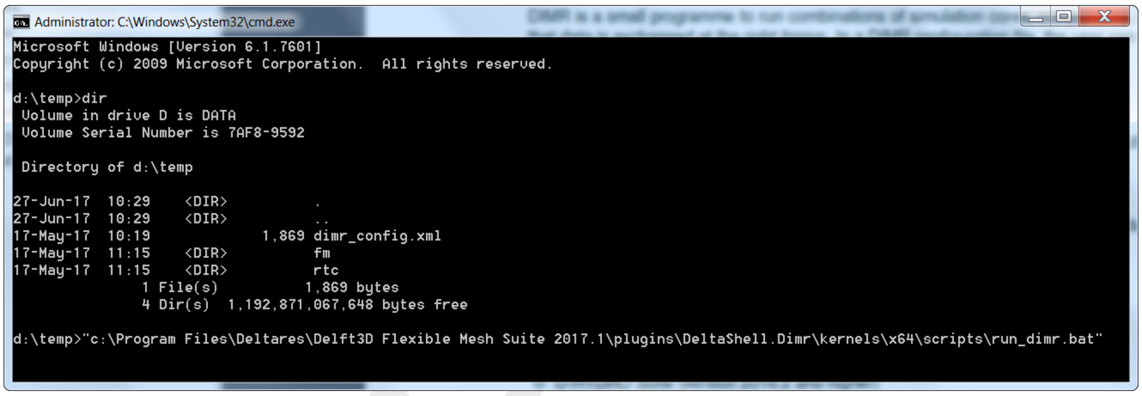
Note: Note the quotes around <run_dimr.bat> and its path; this is necessary when the path contains white spaces.

Note: When the DIMR configuration file has another name than <dimr_config.xml>, then this name must be given as argument of <run_dimr.bat>. A "usage" text is shown on errors and when adding argument "--help" to <run_dimr.bat>.

Note: When a kernel runs in parallel using MPI, the script <run_dimr_parallel.bat> should be used instead of <run_dimr.bat>; it is located in the same directory.

Note: On Linux, the recipe is identical. The name of the Linux run script is <run_dimr.sh>, located in directory <lnx64\bin>. A submit-script is needed when using a computation cluster with a queueing tool and when doing parallel computations. A script, named <submit_dimr.sh> is available for the Deltares cluster; this script can be used, after modification, for clusters outside Deltares.

Note: On Linux, when copying the binaries manually to another location, you have to execute the script <bin\libtool_install.sh> after the copy action.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

d:\temp>dir
 Volume in drive D is DATA
 Volume Serial Number is 7AF8-9592

 Directory of d:\temp

27-Jun-17  10:29    <DIR>          .
27-Jun-17  10:29    <DIR>          ..
17-May-17  10:19             1,869 dimr_config.xml
17-May-17  11:15    <DIR>          fm
17-May-17  11:15    <DIR>          rtc
               1 File(s)          1,869 bytes
               4 Dir(s)  1,192,871,067,648 bytes free

d:\temp>"c:\Program Files\Deltares\Delft3D Flexible Mesh Suite 2017.1\plugins\DeltaShell.Dimr\kernels\x64\scripts\run_dimr.bat"
```

Figure 2.1: *Console example for running DIMR*

2.3 Time management

Time is essential and each component is free to handle time in its own way. Some components have *Boundary Conditions* which are specified in a particular calendar, for example the *Julian* calendar. Other components do not use a specific point of time.

Currently DIMR supports the following *Integrated models*:

- ◇ Stand alone execution of each individual component D-Flow 1D, D-Flow FM, D-Water Quality and D-Waves
- ◇ D-Flow 1D and D-RR: Both components must have the same start-date. Both models will start at time equals zero.
- ◇ D-Flow 1D and D-RTC: The reference-dates of both components must be equal. The D-RTC timestep must match the timestep as specified in the DIMR configuration file, since D-RTC will perform exactly one (privately defined) timestep on each call.
- ◇ D-Flow 1D and D-Flow FM, using 1d2dCoupledModel: See Chapter 4. This is on flooding events, where no astronomical *Boundary Conditions* and tide generated forces are supported. Reference-dates are equal to start-dates. Both components start at time equals zero.
- ◇ D-Flow 1D, D-Flow FM (using 1d2dCoupledModel) and D-RTC: See items above.
- ◇ D-Flow 1D and D-Water Quality: First execute D-Flow 1D, then D-Water Quality. This is possible using one DIMR configuration file.
- ◇ D-Flow FM and D-RTC: See the combination of D-Flow 1D and D-RTC above.
- ◇ D-Flow FM and D-Waves: The reference-date of both components must be equal. Both models start at time equals zero.
- ◇ D-Flow FM, D-Waves and D-RTC: See items above.
- ◇ D-Flow FM and D-Water Quality: First execute D-Flow FM, then D-Water Quality. This is possible using one DIMR configuration file.

3 Technical Information

3.1 Introduction

section 3.2 describes the BMI interface. section 3.3 to section 3.5 describe the DIMR configuration file.

3.2 BMI interface

The communcation between DIMR and the components is according to the BMI-interface (https://csdms.colorado.edu/wiki/BMI_Description). All components in D-HYDRO Suite (D-Flow 1D, D-Flow FM, D-Waves, etc.) are BMI-compliant. The basic BMI-interface implements:

- initialize
- update (perform one timestep, or more - upto the simulation timespan)
- finalize
- get (data)
- set (data)

Note: DIMR itself is also BMI-compliant; The GUI framework uses the BMI-interface in the communication with DIMR.

3.3 The DIMR configuration file

The DIMR input file is in XML format. Appendix A contains an example.

3.4 Sequential and parallel simulations

DIMR enables sequential and parallel simulations in three ways:

- **Sequential simulations**: Component 1 is executed for its full simulation period, output is produced, then component 2 is executed, optionally using the output produced by simulation component 1. Component 2 can not influence component 1. This is normally the configuration when doing a D-Flow FM calculation followed by a D-Water Quality calculation.
- **Parallel simulation**: Components 1 and 2 are both started, component 1 simulates a time period, exchanges data with component 2, component 2 is executed and exchanges data with component 1. This is repeated until the full simulation period is handled. This is normally the configuration when doing a D-Flow FM simulation containing a structure being controlled based on hydrodynamic results, for example the water level at a certain location. Also the parallel simulation of D-Flow FM with D-WAVE uses this configuration. A combination of D-Flow FM, D-RTC and D-WAVE, with different communication frequencies is possible. In Appendix A: DIMR configuration file (XML-format), an example is given for the parallel simulation of D-Flow FM with D-RTC. Mark the following lines:

  - *<parallel>* Parallel simulation
  - *<start name="dflowfm" />* Start the "master" component
  - *<startGroup>* Start a "slave" component
  - *<start name="Real-Time Control" />*

  Note: Inside a parallel simulation, exactly one component must act as the "master" simulation. All other components must be defined using the *<startGroup>* block. Normally, the flow component is the "master" simulation.
  Note: The order of appearance in the *<control>* block defines the order of execution during an "update" event.

◇ **Parallel simulation using a parallel component**: In the examples above, D-Flow FM itself can run in parallel, using several partitions to do the flow simulation, exchanging data directly via MPI. In the example in Appendix A: DIMR configuration file (XML-format), the line *<process>0 1 2</process>* indicates that in a multi processor computation, this component should run in the first three processes. Also the following "magic" line must be added: *<mpiCommunicator>DFM_COMM_DFMWORLD</mpiCommunicator>*

3.5 Data exchange

When DIMR has to take care of data-exchange, it must be specified in the config file using a *<coupler>* block. Each coupler has a unique name and can optionally be used more than once in the *<control>* block. Currently, only scalar quantities can be exchanged by DIMR. In the example in Appendix A: DIMR configuration file (XML-format), two couplers are used; their definitions (source/target components, source/target itemNames) are at the end of the example.

4 Iterative coupling

4.1 Iterative 1D2D coupling

The iterative 1D2D coupling is currently implemented in the C# class *Iterative1D2DCoupler*. The activity diagram for the computation is shown in Figure 4.1.

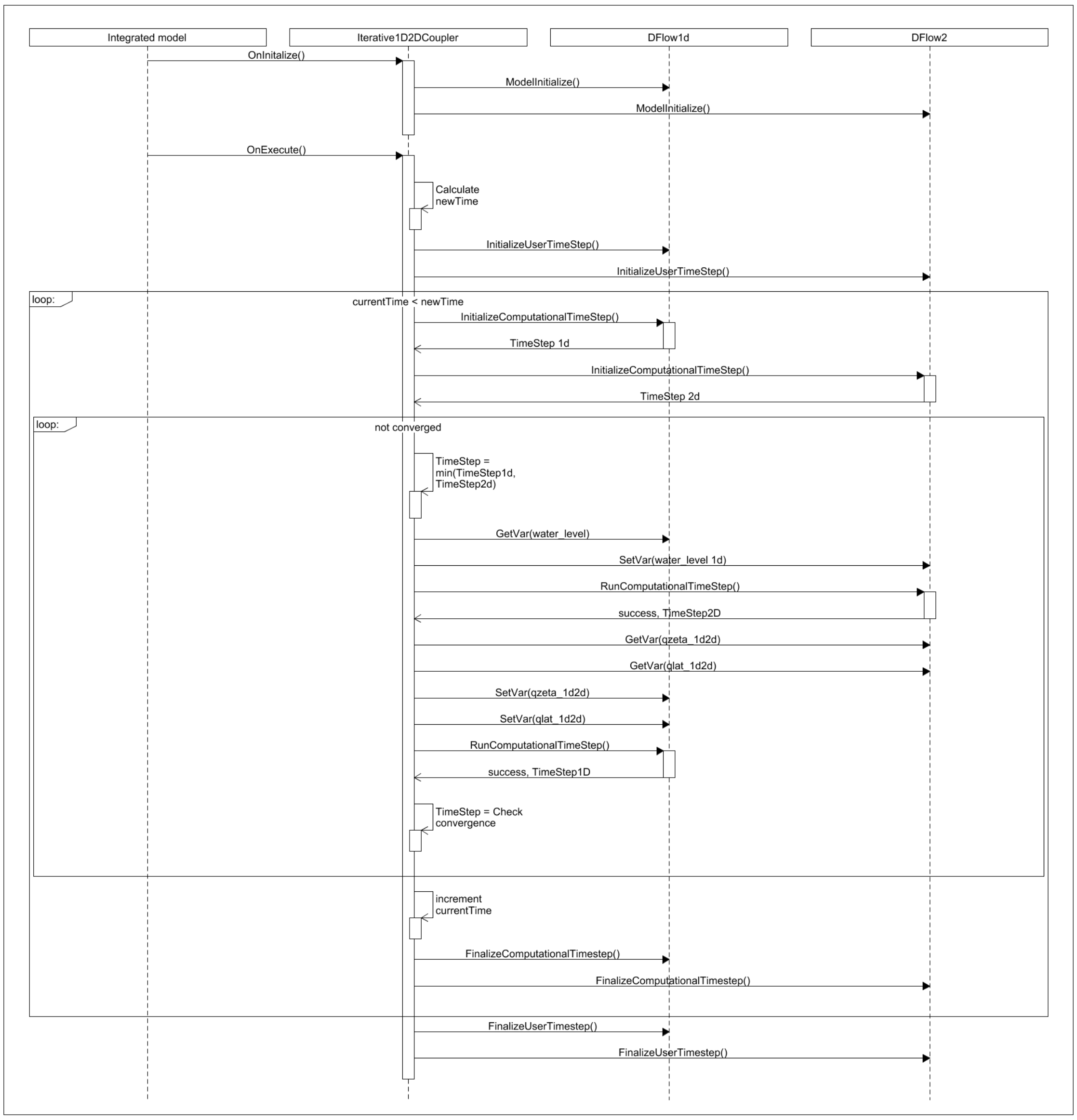


Figure 4.1: *workflow of coupled time step computation.*

In the *Iterative1D2DCoupler* class the mapping of the boundary interfaces of D-Flow FM and the D-Flow 1D water level points is made. The user interface shows the mapping by orange double sided arrows.

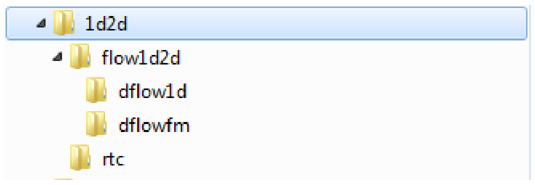


Figure 4.2: *Sample directory structure of a 1d2d coupled model and RTC*

4.2 General

In DIMR the model 1d2dCoupledModel is implemented with the following properties:

- ◇ The BMI interface is implemented.
- ◇ The OnExecute (Run) command lets the model proceed a number of **user** time steps.
- ◇ The var_name in get_var and set_var will have an extra part, which will contain the name of the submodel. (eg. dflow1d, rmm or dflowFM). Either:
  var_name=«model_name»/«variable»
  or
  var_name=«model_name»/«location_type»/«location_id»/«quantity_id».
  The 1d2dCoupledModel will dispatch the calls to the correct submodel.

The mapping of the 1d2d boundary in D-Flow FM to the water level points in D-Flow 1D is performed in the GUI framework and will be written to a file for use in 1d2dCoupledModel. 1d2dCoupledModel contains the two models D-Flow FM and D-Flow 1D. A sample directory structure is shown in Figure 4.2

The 1d2dCoupledModel model is defined in an 'ini'-type input file, located in the directory flow1d2d. The description for this file can be found in section 4.3.

4.3 1d2dCoupledModel model definition file

The model definition for the 1d2dCoupledModel is given in this file. In this file the name, model definition file and the location is given for the stand alone D-Flow 1D model and the D-Flow FM model. The mapping file defines the location of the 1D-2D interfaces. (See section 4.4)

Table 4.1: *1d-2d model definition file.*

Keyword	Type	Default	Description
`[General]`			
`majorVersion`	Int	`1`	Major file version. Do not edit this.
`minorVersion`	Int	`0`	Minor file version. Do not edit this.
`fileType`		`1D2D`	File type. Do not edit this.
`[Model]`			
`type`	string	–	Possible values are "Flow1d" and "FlowFM".
`name`	string	–	model name.
`directory`	string	–	subdirectory for modeldefinition file.
`modelDefinitionFile`	string	–	model definition file.

(continued on next page)

Keyword	Type	Default	Description
		(continued from previous page)	
[Files]			
mappingFile			Name of the mapping file for 1D-2D connections
logFile			Name of the log file for 1D-2D computation
[Parameters]			
maximumIterations	Int	3	maximum number of iterations
maximumError	Double	0.0001	maximum allowable difference between two iterations

4.4 Mapping file

In this file the individual interfaces between a 2D grid cell and a 1D cell is given in terms of (x,y) coordinates.

Table 4.2: *1d-2d model definition file.*

Keyword	Type	Default	Description
[General]			
majorVersion	Int	1	Major file version. Do not edit this.
minorVersion	Int	0	Minor file version. Do not edit this.
fileType		1D2Dmapping	File type. Do not edit this.
[1d2dLink]			
XY_2D	double*2	–	(x,y)-coordinate of the 2D grid cell.
XY_1D	double*2	–	(x,y)-coordinate of the 1D grid cell.

5 Architecture

5.1 System

The operating system employed to develop the software has been Windows 7 Enterprise 64 bits. At the same time, the software has been developed using a version control through SVN.

5.2 Applications

This tool has been fully developed in C#. To develop it the following applications and plugins have been used:

- Visual Studio (VCS 2015 x86 compiler) with the following plugins and NuGet packages
  - Windows Presentation Forms (WPF).
  - Resharper.
  - NUnit.
  - Ghostdoc.
- TortoiseSVN. Necessary for the version control.
- TeamCity. Web application for build management and continuous integration.

5.3 Hardware requirements

The minimum requirements are as follows:

- Intel core i5, x86 (32 bit) processor.
- 8GB RAM.
- 10GB free memory disk.

A DIMR configuration file (XML-format)

A.1 Protected file extensions

In Table A.1 a listing of the protected file extension is given, these file extensions are used by the Delft3D Flexible Mesh Suite and D-HYDRO Suite to detect which importer should be used for importing the model.

Table A.1: *Protected file name and extensions*

File or Extension	Importer
∗.mdu	D-Flow FM
∗.md1d	D-Flow 1D
settings.json	D-RTC

Remark:
◇ If the name of the input file for D-RTC is not given, the assumptions is made that the input xml-files are located at the sub-directory: <./rtc/∗.xml>

A.2 Example DIMR configuration file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<dimrConfig xsi:schemaLocation="http://schemas.deltares.nl/dimr
      http://content.oss.deltares.nl/schemas/dimr-1.0.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://schemas.deltares.nl/dimr">
   <documentation>
      <fileVersion>1.00</fileVersion>
      <createdBy>Deltares, Coupling team</createdBy>
      <creationDate>2015-05-20T07:56:32+01</creationDate>
   </documentation>
   <control>
      <parallel>
         <startGroup>
            <time>0.0 60.0 99999999.0</time>
            <coupler name="flow2rtc"/>
            <start name="myNameRTC"/>
            <coupler name="rtc2flow"/>
            </startGroup>
         <start name="myNameDFlowFM"/>
      </parallel>
   </control>
   <component name="myNameDFlowFM">
      <library>dflowfm</library>
      <process>0 1 2</process>
      <mpiCommunicator>DFM_COMM_DFMWORLD</mpiCommunicator>
      <workingDir>fm</workingDir>
      <inputFile>weirtimeseries.mdu</inputFile>
   </component>
   <component name="myNameRTC">
      <library>RTCTools_BMI</library>
      <process>0</process>
      <workingDir>rtc</workingDir>
         <!-- component specific -->
      <inputFile>settings.json</inputFile>
   </component>
   <coupler name="flow2rtc">
      <sourceComponent>myNameDFlowFM</sourceComponent>
      <targetComponent>myNameRTC</targetComponent>
      <item>
```

```
         <sourceName>observations/Upstream/water_level</sourceName>
         <targetName>input_ObservationPoint01_water_level</targetName>
      </item>
   </coupler>
   <coupler name="rtc2flow">
      <sourceComponent>myNameRTC</sourceComponent>
      <targetComponent>myNameDFlowFM</targetComponent>
      <item>
         <sourceName>output_weir_crest_level</sourceName>
         <targetName>weirs/weir01/crest_level</targetName>
      </item>
   </coupler>
</dimrConfig>
```

Rijkswaterstaat
Ministry of Infrastructure and Water Management

Deltares
Enabling Delta Life