



DASSFLOW v3.0 : User and Developer Guide

Contents

1 Models and Data Assimilation capabilities	3
1.1 Licence	3
1.2 System requirements	4
1.3 Directories organization	4
2 Makefile(s): Compilation and Execution	6
2.1 Makefile(s)	6
2.1.1 Makefile.inc (and Makefile)	6
2.1.2 The user parameters in Makefile.inc	6
2.2 Compilation & Execution commands	7
3 Inputs files	8
3.1 The input.txt file	8
3.1.1 Mesh parameters	8
3.1.2 Simulation parameters	8
3.1.3 Numerical parameters	8
3.1.4 Physical parameters	9
3.1.5 Output Results Files Switches	9
3.1.6 Variational Data Assimilation Parameters	10
3.2 The m_user_data.f90 file	10
3.2.1 Bed elevation and Manning-Strickler coefficient	10
3.2.2 Initial condition	11
3.2.3 Boundary conditions	11
3.2.4 Exact solution	11
3.3 Unstructured mesh files	11
3.3.1 Inhouse format	11
3.3.2 Gmsh format	12
3.4 Boundary conditions	14
3.4.1 Wall	15
3.4.2 Inflow	15
3.4.3 Outflow	15
3.5 Observations	16
4 Outputs	18
4.1 Model unknowns	18
4.2 Post-processed variables	18
4.3 Observations	18
4.4 Cost function	19
4.5 Restart file and initial condition generation	19
5 Adjoint Model	20
5.1 Adjoint Generation	20
5.2 Sensibility analysis	20
5.3 4D-Var data assimilation	20
6 Simple Practice Case	22
6.1 Forward solver	22
6.2 Adjoint solver	22

7	Numerical schemes	24
7.1	Preliminaries	24
7.2	Schemes for conservative fluxes	24
7.2.1	First order scheme	24
7.2.2	Second order scheme	25
7.3	Well-balanced schemes with wet/dry front treatment	26
7.3.1	E-well-balanced scheme	26
7.3.2	A-well-balanced scheme	27
7.4	Time stepping and friction source term implicitation	28
7.5	Boundary conditions	29
8	Validation	31
8.1	Water at rest	31
8.2	Parabolic bowl with linear friction	31
8.3	Dam break(s) with slope and non-linear friction	33
8.3.1	“Failling Gaussian”	33
8.3.2	Flat slope and wet/dry front	33
8.4	simple channel	35
8.4.1	Mac Donald’s benchmark	35
8.4.2	perturbed topography	36
8.5	Solitary wave on a simple beach	37
9	Cost functions	39
10	Python scripts	40
11	Annexes	42
11.1	Example of the <code>input.txt</code> file	42
11.2	Example of the <code>m_user_data.f90</code> file	43
11.3	Example of <code>bc.txt</code> file	45
11.4	Example of <code>hydrograph.txt</code> file	45
11.5	Example of <code>ratcurve.txt</code> file	46
11.6	Example of <code>land_use.txt</code> file	46
11.7	Example of <code>obs.txt</code> file	46

WRAPPED VERSION: TO BE DOCUMENTED IN FALL 2021...

1 Models and Data Assimilation capabilities

DASSFLOW is an open computational software designed for Variational Data Assimilation (4D-VAR) applied to Shallow Water (SW) type flows and hydrology (rainfall-runoff model). Source files are written in Fortran 2003. In the latest version, the following physical models are available: the 2D Shallow Water (SW) equations in variables (h, q_x, q_y) , its version for Herschel-Bulkley rheology (non Newtonian fluids, power-law rheology with threshold) and the GR4H rainfall-runoff model.

Note that a 1D version of SW system in variables (S, Q) (Saint-Venant's equations) is available but in another DassFlow version (please consult DassFlow webpage for details).

The 2D SW equations for Newtonian fluids Let us denote by h the water depth (m), by \mathbf{q} the discharge (m^3/s), $\mathbf{q} = h\mathbf{u}$, $\mathbf{u} = (u, v)^T$ the depth-averaged velocity (m/s).

On a computational domain $\Omega \in \mathbb{R}^2$ and for a time interval $[0, T]$, the equations numerically solved are:

$$\begin{cases} \partial_t h + \text{div}(\mathbf{q}) & = 0 & \text{in } \Omega \times]0, T] \\ \partial_t \mathbf{q} + \text{div} \left(\frac{\mathbf{q} \otimes \mathbf{q}}{h} + g \frac{h^2}{2} \right) & = -gh \nabla z_b - g \frac{n^2 \|\mathbf{q}\|}{h^{7/3}} \mathbf{q} & \text{in } \Omega \times]0, T] \end{cases} \quad (1)$$

where g is the magnitude of the gravity, z_b the bed elevation and n the Manning-Strickler roughness coefficient.

It is accompanied by initial and all usual boundary conditions necessary for real-world applications.

The 2D SW equations for Non-Newtonian fluids (Herschel-Bulkley rheology)

TO BE WRITTEN

The bucket-type rainfall-runoff model

TO BE WRITTEN

Data Assimilation& inversion features Cost functions (direct model output) can be defined by the user, in particular cost functions measuring the discrepancy between the simulated state and data. Data can be provided by external datasets (in-situ or satellites datasets etc). For analysis or experiment purposes, they can be produced by the code itself (twin experiments mode).

The inversions are minimization process requiring the cost function gradient. The latter is computed by the adjoint method.

The adjoint code is generated by performing the Automatic Differentiation tool Tapenade ¹. The source code written in Fortran and Makefile(s) have been designed to do so.

The cost gradient computation enables to analyse spatially-distributed sensitivities; this the 'local sensitivity mode'.

A complete Variational Data Assimilation process is available. This provides a calibrated model from datasets and/or the identification of some input parameters values (eg. the bathymetry, some inflows or physical parametrizations) . The VDA process is based on local minimization algorithms (e.g. BFGS).

The code can be runned in parallel (MPI).

1.1 Licence

The DASSFLOW software is distributed under the CeCILL ² license in version 2 compatible with the GNU GPL license.

List of the external librairies that are eventually used by the DASSFLOW software:

- **Scotch** ³ : a software package for graph and mesh/hypergraph partitioning, graph clustering, and sparse matrix ordering available as free software under the CeCILL-C license.
- **Mpich** ⁴ : a high performance and widely portable implementation of the Message Passing Interface (MPI) standard available as free software.

¹<http://www.sop.inria.fr/tropics/>

²Ce(A)C(nrs)I(NRIA)L(ogiciel)L(ibre), <http://www.cecill.info/>

³<https://gforge.inria.fr/projects/scotch/>

⁴<http://www.mpich.org/>

- **Tapenade** ⁵ : External executable Java program that is called to generate the discrete model adjoint. Free of use in an academic usage.
- **m1qn3** ⁶ : large-scale unconstrained minimization routine distributed under the GNU General Public License and listed in the Free Software Directory.
- **L-BFGSb** ⁷ : TO DO
- **Mumps** ⁸ : a parallel sparse direct solver available in public domain, based on public domain software developed during the Esprit IV European project PARASOL (1996-1999).

1.2 System requirements

One particularity of DASSFLOW software is that you need to compile it if you want to use it. The main reason is that some inputs are defined by some *"user-functions"* which are part of the Fortran code, so we need to recompile the code every time we change one of them.

In this document, we will assume that you have a Linux or Unix operating system even if compilation remains possible on Windows (using Cygwin ⁹ for example) and MacOS (native Linux) operating systems. One of these two compilers must be installed,

- the GNU Fortran compiler ¹⁰ (free software under the GPL free license).
- the INTEL Fortran compiler ¹¹ (commercial software under a proprietary license).

Result output files are written in the following formats,

- text format (`.txt` extension) : a basic format file that can generally be plotted with your favorite graphical XY-plot software ¹², like the Gnuplot ¹³ free software. Python scripts are available in the directory `trunk/tools`
- VTK format (`.vtk` extension) : the Kitware free format file ¹⁴ that can be read by visualization free softwares like Paraview ¹⁵ or Mayavi ¹⁶ for example.
- Tecplot format (`.plt` extension) : a proprietary format file used by the visualization commercial software Tecplot ¹⁷.

and gives the choice of the visualization software(s) to be used.

1.3 Directories organization

At the root of the DASSFLOW distribution or archive that is in the directory `/trunk` of the svn, one finds the following directories Fig.(1).

In the root directory (`/trunk`)

- `/doc` contains the documentation(s) you need.
- `/code` : see below for a detailed description.
- `/tools` contains Python and Bash scripts, in particular ones for plotting model outputs, observations, cost function evolution etc. These scripts are useful or not depending on the user goal(s). TO BE CLARIFIED.
- contains the input files necessary to perform a "case". These files enable to perform DassFlow to simple cases (benchmarks) or a particular case e.g. the Garonne river 2D SW flow.
The files present in a case directory (e.g. `sw_mono/channel`) are of same type than the ones the user must provide in the `/bin` directory to build up his own case. See below for details on such a `/bin` directory.

⁵<http://www-sop.inria.fr/tropics/>

⁶<https://who.rocq.inria.fr/Jean-Charles.Gilbert/modulopt/optimization-routines/m1qn3/m1qn3.html>

⁷<https://\{}>

⁸<http://graa.ens-lyon.fr/MUMPS/>

⁹<http://www.cygwin.com/>

¹⁰<http://gcc.gnu.org/fortran/>

¹¹<http://software.intel.com/en-us/intel-compilers>

¹²http://en.wikipedia.org/wiki/List_of_graphing_software

¹³<http://www.gnuplot.info/>

¹⁴<http://www.vtk.org/>

¹⁵<http://www.paraview.org/>

¹⁶<http://code.enthought.com/projects/mayavi/#Mayavi>

¹⁷<http://www.tecplot.com/>

bin/	bin/
cpp/	cpp/
doc/	doc/
libs/	libs/
Licence_CeCILL_V2-en.html	Licence_CeCILL_V2-en.html
Licence_CeCILL_V2-en.txt	Licence_CeCILL_V2-en.txt
Makefile	Makefile
obj/	obj/
simu/	simu/
souk/	souk/
src/	src/
tap/	tap/
test/	test/

Figure 1: Directories in: (L) /trunk (R) /trunk/code

In the directory /code

- **/bin** : the main directory to make run DASSFLOW software. Necessary input files must be put in this directory by the user.

The complete name of this directory is freely defined by the user in Makefile.inc (e.g. **/bin_mytest**), see below for details.

Multiple **/bin** directories can be created; therefore enabling simultaneous computations.

By default, the code contains an unique directory entitled (e.g. **/bin_A**).

- ▷ the **input.txt** file must be provided by the user. It defines all variables concerning the mesh, numerical and physical parameters and input/output options.
- ▷ the **m_user_data.f90** file must be provided by the user. It contains user-subroutines defining initial conditions, boundary conditions etc.
- ▷ the **bc.txt** file enables the user to assign boundary conditions to labelled boundaries of the mesh.

Moreover, depending on the model nature, some additionnal files are required.

For the Shallow Water model (src/sw_mono source files), the code requires the following files too:

- ▷ the **obs.txt** file enables the user to define observations (in view of data assimilation).
- ▷ the **hydrograph.txt** file contains tabulated hydrographs to be imposed at specified boundaries.
- ▷ the **ratcurve.txt** file which contains all tabulated rat curves to imposed at specified boundaries.

For the hydrological model (src/hydro source files), the code requires the following files too:

- ▷ TO BE WRITTEN
- ▷ The linked executable **exe** is located in this same directory; all simulation result files too (in **/bin{/res** directory).

- **/libs** contains external libraries. Some are compiled, others are not, depending on the user paramters provided in the Makefile.inc.

- **/src** : contains all Fortran source files. The main program is entitled **main.f90**. The sub-directories are the following.

common contains modules and the necessary files for all model types (SW, hydrological model, non-linear rheology etc).

sw_mono contains the files necessary to solve the classical Shallow Water model (Newtonian fluid). ('\\char'__mono' since it is a mono layer fluid model).

2 Makefile(s): Compilation and Execution

2.1 Makefile(s)

2.1.1 Makefile.inc (and Makefile)

The Makefile is driven by the user parameter defined in the Makefile.inc. *The user has to set up his computation in the Makefile.inc.*

The user does not have to open the Makefile, and must not change it of course.

The compilation process is automated using a **Makefile**¹⁸. Some user-defined variables at the header of the file control the software parts that have to be compiled as the linked external librairies. These options are described in the next paragraph. The **Makefile** can also control the execution of the DASSFLOW software even if it is not mandatory. This will be explained in the second paragraph.

2.1.2 The user parameters in Makefile.inc

The user must set up the following variables in the Makefile.inc.

These variables are next used in the Makefile to compile as expected.

- **<CASEDIR>** : String equal to the directory name of the **/bin** directory. It can be freely defined by the user. By default, its value is **/bin_A**.
Note that in view to perform simultaneously different cases or computations, multiple **/bin** directories can be created.
- **<COMPILO>** :
 - ▷ '0' the GNU Fortran compiler is called ('**gfortran**' command).
 - ▷ '1' the INTEL Fortran compiler is called ('**ifort**' command).
- **<OPTIM>** :
 - ▷ '0' set the compilation options in debug mode (for developers to check bugs).
 - ▷ '1' set the compilation options in optimization mode implying a faster execution (that users should use).
- **<MODEL>** :
 - ▷ '0' a developer mode using the **m_user_test.f90** file placed in the **/bin** directory that can be used to test the Fortran MODULE structures implemented. Some examples can be found in the **directory**.
 - ▷ '1' the standard mono-layer Shallow Water model in variables (h,qx,qy) is compiled.
 - ▷ '2' the Herschel-Bulkley version of the Shallow Water model (power-law rheology with threshold containing the Bingham case too) is compiled.
 - ▷ '3' the multi-layer Shallow Water model is compiled. Check if it is available in your distribution or not.
- **<HYDRO>** :
 - ▷ '1' the Hydrological module is compiled.
 - ▷ '0' by default.
- **<ADJOINT>** :
 - ▷ '1' the adjoint files are compiled (for adjoint method). Before, the adjoint .f90 files had to be generated by performing Tapenade.
 - ▷ '0' by default.
- **<MPI>** :
 - ▷ '1' the MPI library is used to run the program in parallel. The compiler command (**gfortran** or **ifort**) is replaced by the **mpif90** wrapper command. The **Scotch** library must be compiled.
 - ▷ '0' by default.

¹⁸<https://www.gnu.org/software/make>, GNU Make.

- `<NB_PROC>` : number of processes called in order to run (in the case 'MPI'=1).
- `<SOLVER>` :
 - ▷ '1' the **Mumps** library (efficient direct solver) is compiled to be linked to the executable. Recommended option.
 - ▷ '2' the **AGMG** library (iterative multi-grid solver) is compiled to be linked to the executable.
 - ▷ '0' by default: no library has to be (re)compiled.
- `<VALID>` :
 - ▷ '1' the user must provide the exact solution in the `m_user_data.f90` file in order to produce relative error norms and exact output result files.
 - ▷ '0' by default.

2.2 Compilation & Execution commands

Once the desired functionalities have been chosen in `Makefile.inc`, the user has to run at the root directory some of the following commands.

- `'make lib'` command : compile the external libraries. Not necessary if `MPI=0` and `SOLVER=0`, values set by default.
- `'make tap_files'` command : generate the adjoint routines by performing Tapenade software. Not necessary if `ADJOINT=0`, value set by default.
- `'make'` command : compile all and link the executable `exe`. The latter is generated in the `/bin` directory. Remember that the user had to first provide all necessary files in the `/bin` directory with at least `input.txt` and `m_user_data.f90` files, see details in the previous section.

If the compilation has successfully been achieved, DASSFLOW can be runned by the following **Makefile** commands:

- `'make runexe'` : run the direct model (using `<NB_PROC>` threads in parallel mode) equivalent to the `'./exe'` command (or `'mpirun -np <NB_PROC> ./exe'` in parallel mode) in the `/bin` directory.
- `'make rungrad'` : compute the direct + the adjoint model to obtain the cost gradient that is a 'local sensitivity analysis'. This command is equivalent to the `'./exe grad'` command in the `/bin` directory (or `'mpirun -np <NB_PROC> ./exe grad'` in parallel mode).
- `'make runmin'` : run a complete VDA process / minimization of the user-defined cost function. This is equivalent to the `'./exe min'` command in the `/bin` directory (with `'mpirun -np <NB_PROC> ./exe min'` in parallel mode).
- `'make runtestadj'` : run a test of the adjoint model. This is equivalent to the `'./exe testadj 0'` command in the `/bin` directory (with `'mpirun -np <NB_PROC> ./exe testadj 0'` in parallel mode).

3 Inputs files

Two main input files must be provided in the **/bin** directory in order to compute a simulation in direct or adjoint modes,

1. the **input.txt** file
2. and the **m_user_data.f90** file ¹⁹

According to the parameters entered in the **input.txt** file, another files must be provided in this same **/bin** directory.

3.1 The input.txt file

The **input.txt** file contains all physical, numerical and simulation control parameters.

3.1.1 Mesh parameters

- **<mesh_type>** : type of the mesh desired,
 - ▷ **'basic'** : structured cartesian mesh. The parameters **<lx>**, **<ly>**, **<nx>** and **<ny>** must be provided.
 - ▷ **'dassflow'** : unstructured mesh with an inhouse format (3.3.1).
 - ▷ **'gmsh'** : unstructured mesh in the Gmsh²⁰ format (3.3.2), a free software distributed under the terms of the GPLv2 license.
- **<mesh_name>** : name of the mesh file that must be placed in the **/bin** directory.
- **<lx>** and **<ly>** in [m] : lengths of the computational domain in horizontal x direction and vertical y direction. By default, the origin (0,0) is taken at the bottom left. It is important for the user-functions defined in the **m_user_data.f90** file.
- **<nx>** and **<ny>** : number of nodes (and not cells) of the computational domain in respectively horizontal x direction and vertical y direction.
- **<bc_N>**, **<bc_S>**, **<bc_W>** and **<bc_E>** : type of the boundary condition at respectively North, South, West and East of the computational domain.

3.1.2 Simulation parameters

- **<ts>** in [s] : total physical simulation time.
- **<dtw>** in [s] : time step to output a result file of all model unknowns as some inputs according to the switches described in the next section 3.1.5.
- **<ntp>** in [s] : time step to output some post-treatment variables.
- **<dta>** in [s] : time step to generate boundary inflow/outflow files in order to use the adjoint mode.
- **<verbose>** : level of verbosity at screen.

3.1.3 Numerical parameters

- **<temp_scheme>** : choice of the numerical scheme to integrate in time,
 - ▷ **'euler'** : the classical first order Euler explicit time stepping with a splitted implicit discretization of the friction source term.
 - ▷ **'imex'** : a Implicit-Explicit Runge-Kutta time stepping scheme giving a global second order method with an implicit discretization of the friction source term.

all schemes are stabilized with a small enough time step Δt .

- **<spatial_scheme>** : choice of the numerical scheme to integrate in space,

¹⁹alternatively **m_user_test.f90**, as it will be explained later.

²⁰<http://geuz.org/gmsh/>, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities

- ▷ **'first_b1'**: the Finite Volume Godonov scheme with the Audusse and al. well-balanced property (recommended with the **'euler'** temporal scheme).
- ▷ **'muscl_b1'**: the Finite Volume Godonov scheme with linear reconstructions at edge with the Audusse and al. well-balanced property.

all with the same HLLC approximate Riemann solver with an L^∞ stability property.

- **<adapt_dt>** :
 - ▷ '0' the time step is fixed at the must provided **<dt>** value.
 - ▷ '1' the time step is calculated and the **<cfl>** number is applied.
- **<dt>** in [s] : fixed time step.
- **<cfl>** : CFL-like number with a value $\in [0, 1]$.
- **<heps>** in [m] : Cut-off water depth that can be usefull to stabilize schemes.
- **<friction>** : '1' active the inclusion of the Manning-Strickler source term in the Shallow Water model.
- **<feedback_inflow>** : '1' active a feedback control on ghost bathymetries to obtain the proper discharg desired at a subset Γ of the computational domain boundary $\partial\Omega$.
- **<coef_feedback>** : coefficient applied to the feedback control (⚠ be carefull to not enter a too strong value that will exhibit severe oscillations on the discharg).
- **<max_nt_for_direct>** : maximum number of time iterations to perform. It can be usefull to cut a very long simulation.

Additional space or time schemes are available ! See Monnier-Couderc et al. AWR 2016 for the numerical analysis and comparisons.

This additionnal schemes are the following, with the .f90 files still available in the directory `src/sw_mono/other_finite_vol`

- **'first'** : the Finite Volume Godonov scheme without well-balanced property.
 - ▷ **'first_b2e'** : the Finite Volume Godonov scheme with the Fernandez-Nieto and al. well-balanced property and a explicit equivalent bed elevation.
 - ▷ **'first_b2i'** : the Finite Volume Godonov scheme with the Fernandez-Nieto and al. well-balanced property and a implicit equivalent bed elevation.
 - ▷ **'muscl'** : the Finite Volume Godonov scheme with linear reconstructions at edges without well-balanced property.
 - ▷ **'muscl_b'** : the Finite Volume Godonov scheme with linear reconstructions at edges without well-balanced property and the Barth limiter.
 - ▷ **'muscl_b1_b'** : the Finite Volume Godonov scheme with linear reconstructions at edge with the Audusse and al. well-balanced property and the Barth limiter (recommended with the **'imex'** temporal scheme).
 - ▷ **'rk2'** : a hybrid Runge-Kutta type time stepping scheme, a simple convex combination of the previous Euler time stepping, more precise and stable but only at first order.

3.1.4 Physical parameters

- **<g>** in $[m.s^{-2}]$: gravitational acceleration.

3.1.5 Output Results Files Switches

- **<w_tecplot>** :
 - ▷ '0' by default.
 - ▷ '1' output of result files in ASCII Tecplot format.
- **<w_vtk>** :
 - ▷ '0' by default.

- ▷ '1' output of result files in ASCII VTK format.
- ▷ '2' output of result files in binary VTK format.
- **<w_exact>** :
 - ▷ '0' by default.
 - ▷ '1' output of exact solution result files and relative error norms. The exact solution must be provided in the **m_user_data.f90** file.
- **<w_norm>** :
 - ▷ '1' relative L^1 , L^2 and L^∞ norms in time.
 - ▷ '2' relative L^1 , L^2 and L^∞ norms integrated in time.
 - ▷ '3' relative L^1 , L^2 and L^∞ norms at the end of the simulation.
 - ▷ '4' absolute L^1 , L^2 and L^∞ norms in time.
 - ▷ '5' absolute L^1 , L^2 and L^∞ norms integrated in time.
 - ▷ '6' absolute L^1 , L^2 and L^∞ norms at the end of the simulation.

The norms are outputted in 'norm_rel_x_yyy.zzz' or 'norm_abs_x_yyy.zzz' files. The **x** characters corresponds to the physical variable (h , $\|u\|$ and $\|q\|$), the **y** characters to the norm type (L^1 , L^2 and L^∞) and the **z** characters to the file format.

- **<w_obs>** :
 - ▷ '0' by default.
 - ▷ '1' output of observations result files and related cost function. The **obs.txt** file need to be prescribed.
- **<use_obs>** :
 - ▷ '0' by default.
 - ▷ '1' active the cost function calculation with the innovation vector.
- **<use_Qobs>** :
 - ▷ '0' by default.
 - ▷ '1' active the cost function calculation with the innovQ vector.

3.1.6 Variational Data Assimilation Parameters

- **<c_manning>**, **<c_bathy>**, **<c_ic>**, **<c_hydrograph>** and **<c_ratcurve>** : a value of '1' active respectively the Manning-Strickler coefficient land use, the bed elevation, the initial condition, the hydrograph(s) and the rat curve(s) in the vector control to minimize, '0' by default.
- **<eps_manning>**, **<eps_bathy>**, **<eps_ic>**, **<eps_hydrograph>** and **<eps_ratcurve>** : coefficient to generate the perturbation control vector.
- **<max_nt_for_adjoint>** : maximum number of adjoint time iterations to perform. It can be usefull to manage the memory consumption.
- **<restart_min>** : maximum number of minimization iterations to perform. It can be usefull to cut a very long minimization process.
- **<eps_min>** : mlqn3 parameter.

3.2 The m_user_data.f90 file

The **m_user_data.f90** file contains Fortran user-functions to define model constants variables, initial condition, boundary conditions and eventually the exact solution.

3.2.1 Bed elevation and Manning-Strickler coefficient

- **<bathy_user(x , y)>** : returns the bed elevation at point $(x, y) \in \Omega$.
- **<manning_user(x , y)>** : returns the Manning-Strickler roughness coefficient at point $(x, y) \in \Omega$.

3.2.2 Initial condition

- `<zs0_user(x , y)>` : returns the initial free surface elevation ($\eta = h + z_b$) at point $(x, y) \in \Omega$.
- `<u0_user(x , y)>` : returns the initial velocity in the horizontal x-direction at point $(x, y) \in \Omega$.
- `<v0_user(x , y)>` : returns the initial velocity in the vertical y-direction at point $(x, y) \in \Omega$.

3.2.3 Boundary conditions

- `<inflow_user(t , x , y)>` : returns a eventual complementary value to finish to define the inflow boundary condition at point $(x, y) \in \Omega$ and time t .
- `<outflow_user(t , x , y)>` : returns a eventual complementary value to finish to define the inflow boundary condition at point $(x, y) \in \Omega$ and time t .

3.2.4 Exact solution

- `<zs_exact(x , y)>` : returns the exact free surface elevation ($\eta = h + z_b$) at point $(x, y) \in \Omega$.
- `<u_exact(x , y)>` : returns the exact velocity in the horizontal x-direction at point $(x, y) \in \Omega$.
- `<v_exact(x , y)>` : returns the exact velocity in the vertical y-direction at point $(x, y) \in \Omega$.

3.3 Unstructured mesh files

The mesh file must be created outside DASSFLOW by a mesh generator software ²¹. This file contains :

- a first list of the mesh nodes with (x, y) coordinates and eventually a z -coordinate defining the bed elevation.
- a second list of the mesh elements (cells and eventually edges) defined by a suited list of the previous defined nodes.

By default, a wall type boundary is applied at the boundary $\partial\Omega$ of the computational domain Ω . Boundary edges defining a subset Γ of the computational boundary $\partial\Omega$ where is considered an inflow or outflow boundary condition must be provided in the mesh file. Each subset Γ is also defined by a group number and the boundary type to apply is controlled by the `bc.txt` file.

3.3.1 Inhouse format

In the previous DASSFLOW software, an inhouse mesh format described in the Tab.(1) has been created in order to consider special issues in the Shallow Water model : the bed elevation z_b and the Manning-Strickler roughness coefficient n .

The bed elevation can be defined by two ways :

- at the nodes corresponding to the z -coordinate in the node list.
- at the cells gravity center (piecewise constant approximation over mesh cells) ²².

The Manning-Strickler roughness coefficients are defined as a land use (zoning) if the `land_use.txt` file is found in the `/bin` directory. The `land_use.txt` file format is given in Tab.(2). According to the prescribed land code of a cell in the inhouse mesh format file Tab.(1), the Manning-Strickler coefficient is defined with the associated value to land code in the `land_use.txt` file.

Otherwise, if the `land_use.txt` file is not found in the `/bin` directory, the Manning-Strickler coefficients are defined by the `<manning_user(x , y)>` Fortran function in the `m_user_data.f90` file (3.2). The function is called for each cell gravity center.

In the last part of the inhouse mesh format file Tab.(1), two subsets Γ_{in} and Γ_{out} of the computational boundary $\partial\Omega$ are defined to consider inflow(s) and/or outflow(s) boundary conditions. The inflow(s) and outflow(s) boundary conditions definition are divided into two successive parts. Beginning by the inflow(s) boundary condition(s), the number of cells just after the INLET label represents the total number of interior cells at boundary of the computational domain made up the subset Γ_{in} . The number of inlet represents the number of inflows to consider in the subset Γ_{in} . If this number is not zero, then the group number of the inflow

²¹<http://www.robertschneiders.de/meshgeneration/software.html>

²²the only way yet in DASSFLOW v3.0

```

$ comment line
number-of-nodes  number-of-cells  scaling

```

```

$ comment line
node-index  x-coordinate  y-coordinate  bed-elevation
...

```

```

$ comment line
cell-index  node1  node2  node3  node4  land-code  bed-elevation
...

```

```

$ comment line
INLET  number-of-cells  number-of-inlet
cell-index  edge-index  boundary-type  ghost-cell-bed-elevation[, group-number]
...
OUTLET number-of-cells number-of-outlet
cell-index edge-index boundary-type ghost-cell-bed-elevation[, group-number]
...

```

Table 1: Inhouse mesh format file ([.] denotes an optional parameter).

```

$ 3 comment lines
number-of-lands

```

```

$ 3 comment lines ...
land-code-1  Manning-Strickler-coefficient-1
land-code-2  Manning-Strickler-coefficient-2
...

```

Table 2: The `land_use.txt` file format. An example of file is given in appendix (11.6).

must be provided for each interior cell. This is the same construction after the OUTLET label. ⚠ Be carefull that if one of the numbers of inlet and outlet is zero, other one must be zero. In this case, no group number has to be provided and the group 1 is automatically assigned to the inflow and the group 2 to the outflow.

Let's consider an exemple with the piece of mesh Fig.(2). We will find the following lines in the mesh file,

```

22      6      5      3      6
31      5      4     10      5
53      1      2      3      1
65     21     11     15     21
69     17     20     22     17
70     21     19     17     21
100    10     12     11     10

```

and in the part defining the inflow(s) or outflow(s) boundary condition, we will find,

```

22      2      1  ghost-cell-bed-elevation
31      3      1  ghost-cell-bed-elevation
53      3      1  ghost-cell-bed-elevation
65      1      1  ghost-cell-bed-elevation
69      3      1  ghost-cell-bed-elevation
70      3      1  ghost-cell-bed-elevation
100     3      1  ghost-cell-bed-elevation

```

The ghost cells bed elevation definition is an important issue because it controls in a certain proportion the flow at inflow(s) and outflow(s).

3.3.2 Gmsh format

“Gmsh is a 3D finite element grid generator with a build-in CAD engine and post-processor. Its design goal is to provide a fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities.

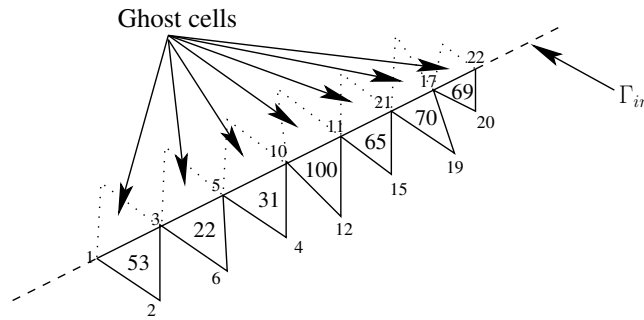


Figure 2: Example of boundary condition definition.

*Gmsh is built around four modules: geometry, mesh, solver and post-processing. The specification of any input to these modules is done either interactively using the graphical user interface or in ASCII text files using Gmsh's own scripting language.*²³

The Gmsh free software can be used to create a unstructured mesh with triangles, quadrangles or both and then interface with DASSFLOW.

The computational domain boundary has to be defined in a first time using the Gmsh graphical user interface or creating directly a .geo file like the given example in the Lst.(1). The “*physical entities*” have then been to be defined in order to create the mesh (the “*physical surface*” in the Lst.(1)) and mainly to define the inflow/outflow boundary conditions (the “*physical line*” in the Lst.(1)). An important issue is that the chronological order of definition of the “*physical lines*” will assign in ascending order the label of the inflow/outflow boundary conditions.

Listing 1: the multibc.geo file in Gmsh format building the geometry and the mesh in Fig.(3).

```
lc = 10;
Point(1) = {0, 0, 0, lc};
Point(2) = {0, 100, 0, lc};
Point(3) = {1000, 0, 0, lc};
Point(4) = {1000, 100, 0, lc};
Point(5) = {0, 200, 0, lc};
Point(6) = {0, 300, 0, lc};
Point(7) = {1000, -100, 0, lc};
Point(8) = {1000, -200, 0, lc};
Point(9) = {350, 100, 0, lc};
Point(10) = {450, 100, 0, lc};
Point(11) = {650, 0, 0, lc};
Point(12) = {550, 0, 0, lc};
Point(13) = {200, 180, 0, lc};
Point(14) = {200, 280, 0, lc};
Point(15) = {800, -80, 0, lc};
Point(16) = {800, -180, 0, lc};
Line(1) = {1, 2};
Line(2) = {3, 4};
Line(3) = {5, 6};
Line(4) = {7, 8};
Line(5) = {1, 12};
Line(6) = {2, 9};
Line(7) = {3, 11};
Line(8) = {4, 10};
BSpline(9) = {5, 13, 9};
BSpline(10) = {6, 14, 10};
BSpline(11) = {8, 16, 12};
BSpline(12) = {7, 15, 11};
Line Loop(14) = {5, -11, -4, 12, -7, 2, 8, -10, -3, 9, -6, -1};
Plane Surface(14) = {14};
Physical Line("in1") = {1};
Physical Line("in2") = {3};
Physical Line("out1") = {2};
Physical Line("out2") = {4};
Physical Surface("surf") = {14};
```

After typing the command 'gmsh -2 multibc.geo', the resulting generated mesh is showed in the Fig.(3). But one can also use the graphical user interface to generate the mesh and so the numerous Gmsh tools.

²³<http://geuz.org/gmsh/>

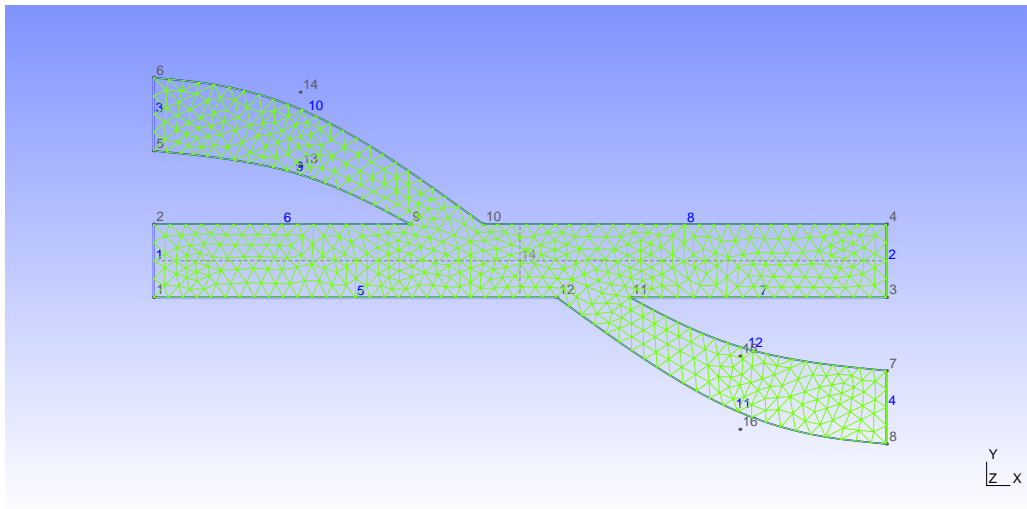


Figure 3: Example of mesh generation with Gmsh. Node and line numbers correspond to the listing 1.

3.4 Boundary conditions

The available boundary conditions are listed in the Tab.(3). There are three main types of boundary conditions : the wall, the inflow and the outflow ²⁴ with subtype for the inflow and the outflow.

Type	Name	Description	Data
Wall	'wall'	wall/slip condition	-
Inflow	'discharg1'	discharg is imposed (approximate backwater curve)	the hydrograph.txt file if exists in the /bin directory, the <code><inflow_user(t , x , y)></code> Fortran function in the m_user_data.f90 file otherwise.
	'discharg2'	discharg is imposed (fixed water elevation)	the hydrograph.txt file if exists in the /bin directory, the <code><inflow_user(t , x , y)></code> Fortran function in the m_user_data.f90 file otherwise.
Outflow	'transm'	homogeneous Neumann	-
	'zspresc'	water elevation η is prescribed	the <code><outflow_user(t , x , y)></code> Fortran function in the m_user_data.f90 file.
	'hpresc'	water depth h is prescribed	the <code><outflow_user(t , x , y)></code> Fortran function in the m_user_data.f90 file.
	'ratcurve'	rating curve based	the ratcurve.txt file.

Table 3: Available boundary conditions.

For a cartesian mesh, the boundary condition type is prescribed directly at one face of the computational domain boundary setting the `<bc_N>`, `<bc_S>`, `<bc_W>` and `<bc_E>` variables in the **input.txt** file (3.1) to the desired associated name in the Tab.(3).

For a unstructured mesh, the boundary condition type is prescribed in the **bc.txt** file in the format described in the Tab.(4). A group number is assigned at each subset of the computational domain boundary $\partial\Omega$ as explained in the previous section(3.3). Thus, the associated type of boundary condition is read in this **bc.txt** file that provides a relative flexibility. If no **data-type** file is prescribed, then the **m_user_data.f90** file is used to define the inflow and the outflow.

²⁴the periodic boundary condition is not yet available.

\$ 3 comment lines		
number-of-tagged-bc		
<hr/>		
\$ 3 comment lines		
<hr/>		
group-1	bc-type-1	[data-type-1]
group-2	bc-type-2	[data-type-2]
...		

Table 4: The **bc.txt** file format.

3.4.1 Wall

The 'wall' boundary condition type is a slip condition since there is no viscous term in the model. It can be also used as a symmetry boundary condition. It is the default boundary condition assigned at a boundary edge not in a defined subset of the computational domain boundary $\partial\Omega$.

3.4.2 Inflow

The inflow boundary condition consists of applying a discharge $Q(t)$ to a subset Γ of the computational domain boundary $\partial\Omega$. The discharge relation $Q(t)$ is prescribed either by the `<inflow_user(t , x , y)>` user-function in the **m_user_data.f90** file or by the **hydrograph.txt** file Tab.(5). The **data-type** option must be setted to 'file' in the **bc.txt** file in order to use the tabulated values of the relation $Q(t)$ law in the **hydrograph.txt** file.

The type 'discharg1' gives a more robust method to prescribed the discharge relation $Q(t)$. It should be preferred to the 'discharg2' type which can be more precise but can generate a calculation divergence.

When the considered wet surface corresponding to Γ is non trivial, practice shows that one can observe a wrong prescribed discharge $Q(t)$. In order to overcome this drawback, a solution is to apply a feedback process on the associated ghost bed elevations setting the variable `<feedback_inflow>` to '1' in the **input.txt** file (by default) and eventually changing the associated feedback process coefficient `<coef_feedback>` (to '0.1' by default).

\$ 3 comment lines	
number-of-hydrographs	
<hr/>	
\$ 3 comment lines	
<hr/>	
number-of-tabulated-times-hydrograph-1	
<hr/>	
t11	Q11
t12	Q12
...	
<hr/>	
\$ 3 comment lines	
<hr/>	
number-of-tabulated-times-hydrograph-2	
<hr/>	
t21	Q21
t22	Q22
...	

Table 5: The **hydrograph.txt** file format.

3.4.3 Outflow

The outflow boundary condition can be prescribed in different ways :

- the 'transm' type for which simple homogeneous Neumann conditions are applied to water depth and normal velocity. It can be used in simple cases where the ghost bed elevations are well defined (a constant slope channel for example). Otherwise, it can generates an improper outflow boundary condition.
- the 'zspresc' and 'hpresc' types where respectively a water surface relation $z_b(t)$ and a water depth relation $h(t)$ are imposed at the boundary.

- the 'ratcurve' type for which a tabulated rating curve in the **ratcurve.txt** file described in the Tab.(6) is used to impose a relation $h(Q)$ at the boundary.

\$ 3 comment lines	
number-of-rating-curves	
\$ 3 comment lines	
number-of-tabulated-water-elevation-1	water-elevation-reference
h11	Q11
h12	Q12
...	
\$ 3 comment lines	
number-of-tabulated-water-elevation-2	water-elevation-reference
h21	Q21
h22	Q22
...	

Table 6: The **ratcurve.txt** file format.

3.5 Observations

There is the possibility to generate output result files containing user defined observations. This can be realized creating a **obs.txt** file at the root of the **/bin** directory according to the format Tab.(7). The control variable **<w_obs>** must be setted to '1' in the **input.txt** file.

In context of the shallow water model, two types of observations are possible,

- The stations : the evolution in time of the water depth $h(t)$ and the velocity components $u(t)$ and $v(t)$ are outputted with a prescribed time step and a weight (usefull for the cost function definition 4.4),
 - ▷ a user-defined point representing the constant value of the cell containing the point (stations keyword).
 - ▷ a list of user-defined points representing the mean value over the constant values of the cells containing the list of points (stations_with_grp keyword).
- The sections : the evolution in time and along a line of the water depth $h(t, x, y)$ and the velocity components $u(t, x, y)$ and $v(t, x, y)$ are outputted with a prescribed number of points and time step. It is written the constant values of the cells containing each line point.

In case of stations defined with a group of cells (stations_with_grp), a file per station 'station_grp_xxxx.txt' must be provided with a two column list of points or cell indexes to finish to define the observation (the x characters is corresponding to the station label).

In a SWOT context, it is necessary to add an other file to the **/bin** directory, the **param_obs.txt** file. This file is used to provide informations to DASSFLOW about SWOT parameters. Those informations, for each observation stations, are: the lenght of river which is observed by the SWOT band, time observation, coordinates of intersections points between river and SWOT band. The format of **param_obs.txt** file is detailed in Tab.(9).

Remarks : In a SWOT case, parameter obs-dt-X define in **param_obs.txt** correponds to the offset of SWOT passage of SWOT band X and station-X-dt of **obs.txt** corresponds to the SWOT satellite period.

\$ free lines					
<hr/>					
stations number-of-stations					
<i>x</i> -coord	<i>y</i> -coord	station-1-dt	station-1-weight		
<i>x</i> -coord	<i>y</i> -coord	station-2-dt	station-2-weight		
...					
<hr/>					
\$ free lines					
<hr/>					
stations_with_grp number-of-stations					
station-1-label	station-1-dt	station-1-weight			
station-2-label	station-2-dt	station-2-weight			
...					
<hr/>					
\$ free lines					
<hr/>					
sections number-of-sections					
<i>x</i> 1-coord	<i>y</i> 1-coord	<i>x</i> 2-coord	<i>y</i> 2-coord	section-1-nb-of-points	section-1-dt
<i>x</i> 1-coord	<i>y</i> 1-coord	<i>x</i> 2-coord	<i>y</i> 2-coord	section-2-nb-of-points	section-2-dt
...					
<hr/>					

Table 7: The **obs.txt** file format.

\$ free lines		\$ free lines	
<hr/>		<hr/>	
points		indexes	
<i>x</i> -coord	<i>y</i> -coord	index-1	
<i>x</i> -coord	<i>y</i> -coord	index-2	
...		...	
<hr/>		<hr/>	

Table 8: The 'station_grp_xxxx.txt' file format to define a group of cells representing a station observation.

number-of-stations					
river-length-1	obs-dt-1	<i>x</i> -coord-1	<i>y</i> -coord-1	<i>x</i> -coord-2	<i>y</i> -coord-2
river-length-2	obs-dt-2	<i>x</i> -coord-1	<i>y</i> -coord-1	<i>x</i> -coord-2	<i>y</i> -coord-2
river-length-3	obs-dt-3	<i>x</i> -coord-1	<i>y</i> -coord-1	<i>x</i> -coord-2	<i>y</i> -coord-2
...					
<hr/>					

Table 9: The **param_obs.txt** file format.

4 Outputs

All output result files are written in the **/bin/res** directory. The frequency and the format(s) are controlled in the user-defined **input.txt** file respectively by the `<dtw>`, `<ntp>` and `<nta>` time steps and the output switches described in the section (3.1.5).

4.1 Model unknowns

The Shallow Water model primitive unknowns h , u , v as some model constants z_b , n can be written in output result files with the `<dtw>` time step in some available formats,

- the VTK format (**.vtk** extension) in ASCII encoding setting the `<w_vtk>` to '1' and in binary encoding setting the `<w_vtk>` to '2'.
- the Tecplot format (**.plt** extension) in ASCII encoding setting the `<w_tecplot>` to '1'.

The file names are **'result_XXXXXXE+xx.yyy'** where the **x** characters are representing the simulation time, except for the initial and final output result files for which the file names are respectively **'result_initial.yyy'** and **'result_final.yyy'**, and the **y** characters the above file format.

4.2 Post-processed variables

Several post-processed variables are also written in output result files with the `<ntp>` time step and in the format defined in the **input.txt** file by the switches described in the section (3.1.5),

- the time step dt in [s] in the **'time_step.yyy'** file.
- the volume of water in the computational domain in $[m^3]$ in the **'water_vol.yyy'** file.
- the volume of water in the computational domain in $[m^3]$ eventually added numerically in the **'water_vol_num_add.yyy'** file (Effect of cutting negative water depth in mass conservation).
- the discharges in $[m^3.s^{-1}]$ at inflow and outflow in the **'sum_mass_flux_zzz_xxx.yyy'** and **'sum_q_zzz_xxx.yyy'** files. The **z** characters is **inflow** or **outflow**, the **x** characters the label of the boundary.

4.3 Observations

An observation output result file is generated according to the **obs.txt** file for each defined station and two files for each defined section,

- the **'obs_station_yyyy.xxx'** files with the format described in Tab.10.
- the **'obs_section_yyyy.xxx'** files with the format described in Tab.11.
- the **'obs_q_h_section_yyyy.xxx'** files with the format described in Tab.12.

where the **y** characters are the label of the stations and the sections. These labels are defined in ascending order.

\$ header according to the format choosen

time-1	water-depth-1	x-velocity-component-1	y-velocity-component-1
time-2	water-depth-2	x-velocity-component-2	y-velocity-component-2
...			

Table 10: the **'obs_station_yy.xxx'** files format.

\$ header according to the format choosen						
\$ time-1						
<i>x1-coord</i>	<i>y1-coord</i>	<i>bathy-1</i>	<i>depth-1-1</i>	<i>elevation-1-1</i>	<i>x-velocity-1-1</i>	<i>y-velocity-1-1</i>
<i>x2-coord</i>	<i>y2-coord</i>	<i>bathy-2</i>	<i>depth-2-1</i>	<i>elevation-2-1</i>	<i>x-velocity-2-1</i>	<i>y-velocity-2-1</i>
...						
\$ time-2						
<i>x1-coord</i>	<i>y1-coord</i>	<i>bathy-1</i>	<i>depth-1-2</i>	<i>elevation-1-2</i>	<i>x-velocity-1-2</i>	<i>y-velocity-1-2</i>
<i>x2-coord</i>	<i>y2-coord</i>	<i>bathy-2</i>	<i>depth-2-2</i>	<i>elevation-2-2</i>	<i>x-velocity-2-2</i>	<i>y-velocity-2-2</i>
...						

Table 11: the 'obs_section_yy.xxx' files format.

\$ header according to the format choosen	
water-depth-max-1	discharge-1
water-depth-max-2	discharge-2
...	

Table 12: the 'obs_q_h_section_yy.xxx' files format.

4.4 Cost function

A cost function is calculated and printed to screen at the end of the simulation,

$$J = \sum_{i=1, \text{Nobs}} \omega_i \sum_{j=1, \text{Ntimes}} \left((\bar{h}_{i,j} - \bar{h}_{i,j}^{obs}) + (\bar{w}_{i,j} - \bar{w}_{i,j}^{obs}) \right)^2 \quad (2)$$

where Nobs is the number of stations and Ntimes the time serie number, both controlled by the user-defined **obs.txt** file (3.5).

The observations files can be generated by a previous simulation or created in the same format described in the Tab.(10). These files have to be placed in the **/bin/obs** directory (must be created) and the control variable **<use_obs>** has to be setted to '1'. Otherwise, the h^{obs} are not taken into account in the Eq.(2).

For more detail see 9.

4.5 Restart file and initial condition generation

A file named **'restart.bin'** is written in addition to the model output result files. In particular, it can be used to perform a long simulation in several times setting the variable **<max_nt_for_direct>** to a maximum number of iterations to perform for each part of the long simulation.

This **'restart.bin'** file can be renamed in **'ic.bin'** file and used as an initial condition.

5 Adjoint Model

In order to calculate the gradient of the cost function ∇J in a relative minimum of time computation, and so to perform sensibility analysis (5.2) or data assimilation (5.3), the DASSFLOW software has been originally designed, in sense of source code structuration, to generate automatically the discrete model adjoint using the differentiation tool **Tapenade**. Some final tricks calling a Perl program²⁵ have also been implemented to override some **Tapenade** manual operations (dynamic array sizes, adjoint variables management and the adjoint of the MPI standard communication operations) in order to really achieve to a complete automatic generation. User can for example define a new cost function and so generate the associated discrete model adjoint.

5.1 Adjoint Generation

- The automatic generation is called typing the command `'make run tap_files'` at the DASSFLOW root directory. All discrete adjoint model source files are placed in the `/tap` directory.
- In order to compile/link these source files, the `ADJOINT` variable must be set to '1' in the **Makefile**.
- Then, typing the commane `'make'` generates the executable **exe** in the `/bin` directory.

5.2 Sensibility analysis

The generation of the discrete model adjoint provides a way to perform sensibility analysis and estimate the influence of the different parameters. If \mathbf{k} denotes the control vector containing all the parameters, then the absolute model sensibility to a given parameter k_i ,

$$s_{k_i} = \frac{\partial J}{\partial k_i} \quad (3)$$

are written in output result files providing a measure of the model change response due to a change of a given parameter.

This mode is called typing `'make rungrad'` at the DASSFLOW root directory (or typing `'./exe grad'` in the `/bin` directory in sequential mode and `'mpirun -np <NB_PROC> './exe grad'` in parallel mode).

In context of the Shallow Water model, the control vector \mathbf{k} and the corresponding output result files are,

- the hydrograph time series and its gradient in the `'hydrographxxx_grad'` files. The `x` characters denotes the label of the hydrograph.
- the bed elevation scalar field z_b and its gradient in the `'bathy_grad.yyy'` file. The `y` characters correspond to the file format.
- the Manning-Strickler roughness coefficient n and its gradient in the `'manning_grad.yyy'` file. The `y` characters correspond to the file format.

All these output result files are placed in the `/bin/grad` directory at the end of the calculation of the cost function gradient ∇J .

5.3 4D-Var data assimilation

In this mode, a local minimum of the cost function is sought using the discrete model adjoint to calculate the cost function gradient ∇J and a local descent algorithm, i.e. **m1qn3**. One can identify the model input control variables “matching at best” with observation data.

The identification process principle is sketched in Fig(4). Given a first guess \mathbf{k}_0 (user is free to configure his first guess as he would have configured a direct simulation), we seek the iterates \mathbf{k}_i which make decrease the cost function using a descent algorithm. To do so, at each iterate,

1. the cost function $J(\mathbf{k}_i)$ and its gradient $\nabla J(\mathbf{k}_i)$ are computed calling the discrete forward model (from 0 to T) and its adjoint (from T to 0, reverse in time).
2. given \mathbf{k}_i , $J(\mathbf{k}_i)$ and $\nabla J(\mathbf{k}_i)$, the **m1qn3** library is invoked in order to find a new iterate such that $J(\mathbf{k}_{i+1}) < J(\mathbf{k}_i)$.
3. the convergence criteria is tested, i.e., the `<eps_min>` parameter in the `input.txt` file.

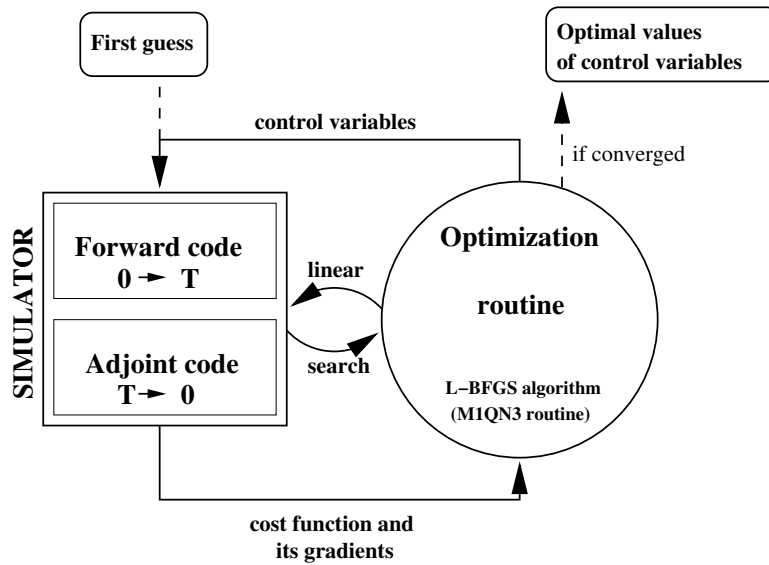


Figure 4: 4D-Var identification process principle.

This mode is called typing 'make runmin' at the DASSFLOW root directory (or typing './**exe min**' in the **/bin** directory in sequential mode and 'mpirun -np <NB_PROC> './**exe min**' in parallel mode). The iterate informations are written on screen and in the '**min_cost.txt**' file in the **/bin/min** directory.

The control variables building the vectors \mathbf{k}_i are activated setting to '1' the following parameters in the **input.txt** file,

- <c_manning> for the Manning-Strickler roughness coefficient and written in the 'manning.xxx' output result files.
- <c_bathy> for the bed elevation and written in the 'bathy.xxx' output result files.
- <c_ic> for the initial condition and written in the 'ic.xxx' output result files.
- <c_hydrograph> for the hydrograph(s) and written in the 'hydrograph_yyy.xxx' output result files.
- <c_ratcurve> for the rating curve(s) and written in the 'ratcurve_yyy.xxx' output result files.

The identification process could be very long in time and the <restart_min> parameter in the **input.txt** file can be setted to the maximum number of iterations to perform. A file '**restart_min.bin**' is generated at each iteration and is read if it exists to restart the identification process.

²⁵ ./src/adjoint/finish_to_gen_adjoint.pl

6 Simple Practice Case

It is presented here a simple tutorial test case showing how to use the DASSFLOW software step by step. The case of simulation is saved in the directory `simu/sw_mono/channel_adj`. It is a simple rectangular channel with a constant slope perturbed by sinuous waves.

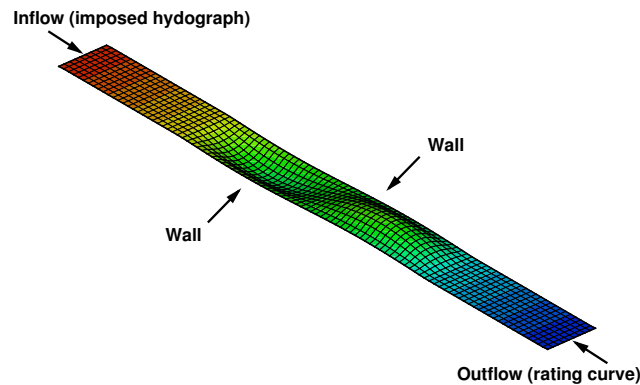


Figure 5: Rectangular channel bed elevation.

6.1 Forward solver

In order to use the direct solver, one must follow these steps :

1. Copy the `input.txt`, `m_user_data.f90`, `hydrograph.txt`, `ratcurve.txt` and `obs.txt` files from the `simu/sw_mono/channel_adj` directory to the `/bin` directory.
2. Choose compilation options in the `Makefile` and compile/link the executable called `exe` and placed in the `/bin` directory typing the command `'make'`.
3. Run the direct solver using the command `'make runexe'`, or directly `./'exe'` in the `/bin` directory.
4. Check the result files in the `bin/res` directory.

6.2 Adjoint solver

In order to use the adjoint solver, one must follow these steps :

1. Set to '1' the `ADJOINT` parameter in the `Makefile`.
2. Generate the adjoint Fortran program files running the command `'make tap_files'` (Tapenade must be installed with appropriate Java).
3. Compile the executable by running the command `'make'`.
4. Copy the observation files from the `/bin/res` directory into a new directory `/bin/obs` in order to use it setting to '1' the option `<use_obs>` in the `input.txt` file.
5. Change the `hydrograph.txt` file by a new file (`hydrograph_first_guess.txt`).
6. We have the choice to compute the cost function using the command `'make runexe'`, to calculate its gradient using the command `'make rungrad'` or to minimize it using the command `'make runmin'`.

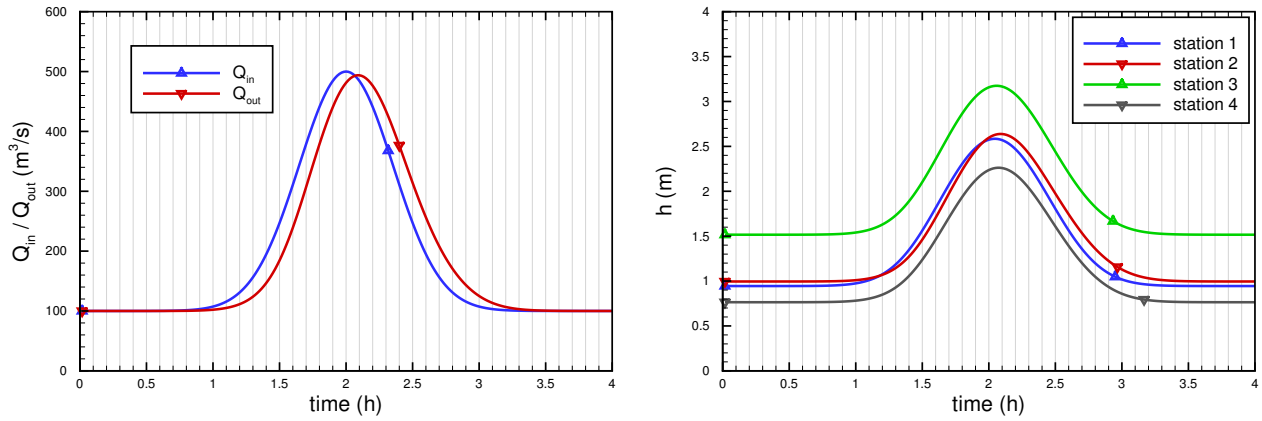


Figure 6: (left) Imposed hydrograph ; (right) Resulting water depth observed in time at stations 1 & 2.

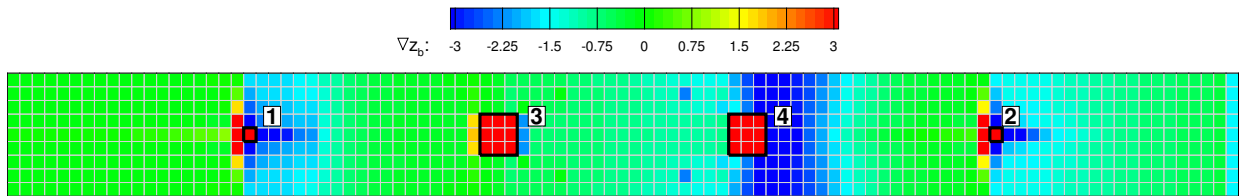


Figure 7: Topography gradient.

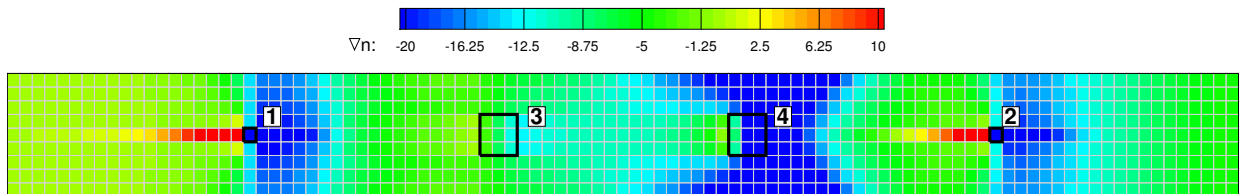


Figure 8: Manning's roughness coefficient gradient.

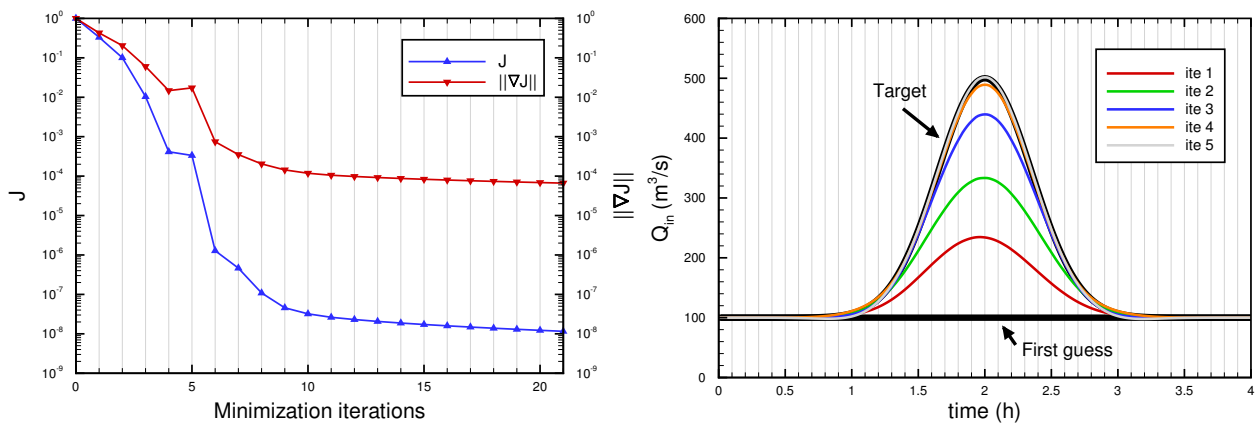


Figure 9: Identification of Q_{in} : (left) File min_cost.txt; (right) Evolution of Q_{in} with iterations of minimization.

7 Numerical schemes

In this section, we describe the numerical methods based on a Finite Volume approach solving the Shallow water model implemented in the DASSFLOW software.

7.1 Preliminaries

Following the formalism of the Finite Volume method, we introduce a discretization \mathcal{T}_h of a computational domain Ω with N cells K_i (cells can be triangles or quadrilaterals that can be also mixed in practice). Let us to introduce some notations and conventions (see Fig.10), considering a given cell K (omitting above i index),

- m_K is the area of the cell K , $m_{\partial K}$ its perimeter and x_K its barycenter.
- e is one of the δK boundary edges, m_e its length and x_e its center.
- K_e is the neighboring cell to K across e .
- $\mathbf{n}_{e,K}$ is the unit normal to e oriented from K to K_e .

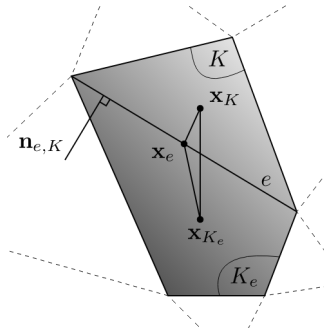


Figure 10: Notations and conventions for mesh discretization and numerical schemes associated.

The Shallow Water equations are rewritten in conservative form as follows,

$$\begin{aligned} \partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) + \partial_y \mathbf{G}(\mathbf{U}) &= \mathbf{S}_g(\mathbf{U}) + \mathbf{S}_f(\mathbf{U}) \\ \mathbf{U} &= \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} hu \\ hu^2 + \frac{gh^2}{2} \\ huv \end{bmatrix}, \quad \mathbf{G}(\mathbf{U}) = \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{gh^2}{2} \end{bmatrix}, \\ \mathbf{S}_g(\mathbf{U}) &= \begin{bmatrix} 0 \\ -gh\nabla z_b \end{bmatrix}, \quad \mathbf{S}_f(\mathbf{U}) = \begin{bmatrix} 0 \\ -g \frac{n^2 \|\mathbf{u}\|}{h^{1/3}} \mathbf{u} \end{bmatrix} \end{aligned} \quad (4)$$

We introduce the piecewise constant approximation over mesh cells,

$$\mathbf{U}_K = \frac{1}{m_K} \int_K \mathbf{U} dK \quad (5)$$

7.2 Schemes for conservative fluxes

7.2.1 First order scheme

Integrating over the cell K the hyperbolic system of partial differential equations (4) without the source terms \mathbf{S}_g and \mathbf{S}_f and noting \mathbf{U}_K^n and \mathbf{U}_K^{n+1} the piecewise constant approximation of \mathbf{U} at time t^n and t^{n+1} (with $t^{n+1} = t^n + \Delta t^n$), and applying the Godunov method [13], the semi-discrete scheme for the homogeneous system can be written as follows,

$$\mathbf{U}_K^{n+1} = \mathbf{U}_K^n - \frac{\Delta t^n}{m_K} \sum_{e \in \partial K} m_e \mathbf{F}_e(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K}) \quad (6)$$

The rotational invariance property of the shallow water equations (4) allows to reduce this sum of 2D problem to a sum of 1D local Riemann problems such that $\mathbf{F}_e(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K}) = \mathbf{R}_{e,K}^{-1} \hat{\mathbf{F}}_e(\hat{\mathbf{U}}_{e,K}^n, \hat{\mathbf{U}}_{e,K_e}^n)$ with $\hat{\mathbf{U}}_{e,K}^n = \mathbf{R}_{e,K} \mathbf{U}_K^n$ where $\mathbf{R}_{e,K}$ is the rotation matrix, see e.g. [29]. The HLLC approximate Riemann solver is used,

$$\left\{ \begin{array}{l} [\hat{\mathbf{F}}_e^{HLLC}]_{1,2} = \frac{s_{K_e} [\mathbf{F}(\hat{\mathbf{U}}_K)]_{1,2} - s_K [\mathbf{F}(\hat{\mathbf{U}}_{K_e})]_{1,2} + s_K s_{K_e} ([\hat{\mathbf{U}}_{K_e}]_{1,2} - [\hat{\mathbf{U}}_K]_{1,2})}{s_{K_e} - s_K} \\ [\hat{\mathbf{F}}_e^{HLLC}]_3 = [\hat{\mathbf{F}}_e^{HLLC}]_1 \hat{v} * \quad \text{with} \quad \hat{v} * = \begin{cases} \hat{v}_K & \text{if } s^* \geq 0 \\ \hat{v}_{K_e} & \text{if } s^* < 0 \end{cases} \end{array} \right. \quad (7)$$

with the wave speed estimates due to J.-P. Vila [30],

$$\begin{aligned} s_K &= \min(0, \hat{u}_K - \sqrt{gh_K}, \hat{u}_{K_e} - 2\sqrt{gh_{K_e}} + \sqrt{gh_K}) \\ s_{K_e} &= \max(0, \hat{u}_{K_e} + \sqrt{gh_{K_e}}, \hat{u}_K + 2\sqrt{gh_K} - \sqrt{gh_{K_e}}) \end{aligned} \quad (8)$$

as it has been demonstrated that it insures L^∞ stability, positivity and consistency with entropy condition under a CFL condition. This choice is made for the intermediate wave speed estimate, see e.g. [29],

$$s^* = \frac{s_K h_{K_e} \hat{u}_{K_e} - s_{K_e} h_K \hat{u}_K - s_K s_{K_e} (h_{K_e} - h_K)}{h_{K_e} (\hat{u}_{K_e} - s_{K_e}) - h_K (\hat{u}_K - s_K)} \quad (9)$$

The CFL-like condition for the time step Δt^n is, see e.g. [?],

$$\Delta t^n = c \min_{K \in \Omega} \left(\frac{2 m_K}{m_{\partial K} (\|\mathbf{u}_K^n\| + \sqrt{gh_K^n})} \right) \quad (10)$$

with a constant $c \in [0.5, 0.8]$ in practice. For numerical purpose, the eventual round-off error generating a negative water depth is eliminated at the end of the time step $h_K^{n+1} = \max(0, h_K^n)$. A small cut-off water depth h_ϵ is also eventually introduced below which the velocity components are set to zero in order to stabilize the numerical model. This cut-off has been found however seldom required with used HLLC flux and is generally fixed to machine epsilon.

7.2.2 Second order scheme

A monoslope second order MUSCL scheme, see e.g. [9, 5], consists in local linear reconstructions by calculating a vectorial slope $[\nabla \mathbf{U}_K^n]_i$ in each cell K for each variable i such that the two reconstructed conservative variables vectors $\mathbf{U}_{e,K}^n$ and \mathbf{U}_{e,K_e}^n at each side of edge e ,

$$\begin{aligned} \mathbf{U}_{e,K}^n &= \mathbf{U}_K^n + \nabla \mathbf{U}_K^n \cdot \mathbf{x}_K \mathbf{x}_e \\ \mathbf{U}_{e,K_e}^n &= \mathbf{U}_{K_e}^n + \nabla \mathbf{U}_{K_e}^n \cdot \mathbf{x}_{K_e} \mathbf{x}_e \end{aligned} \quad (11)$$

replace \mathbf{U}_K^n and $\mathbf{U}_{K_e}^n$ in the original first order semi-discrete scheme (6) to evaluate the numerical flux \mathbf{F}_e ,

$$\mathbf{U}_K^{n+1} = \mathbf{U}_K^n - \frac{\Delta t^n}{m_K} \sum_{e \in \partial K} m_e \mathbf{F}_e(\mathbf{U}_{e,K}^n, \mathbf{U}_{e,K_e}^n, \mathbf{n}_{e,K}) \quad (12)$$

Predicted slope With a such linear reconstruction, one can expect a scheme with a second-order accuracy in space (for sufficient regular solutions). To this end, a least square method is first employed to predict the vectorial slopes for each primitive variable ($\mathbf{W}_K^n = [h_K^n \ u_K^n \ v_K^n]^T$). These sums of squares,

$$E_i \left(\left[\widetilde{\nabla \mathbf{W}_K^n} \right]_i \right) = \sum_{e \in \partial K} ([\mathbf{W}_{K_e}^n]_i - ([\mathbf{W}_K^n]_i + [\nabla \mathbf{W}_K^n]_i \cdot \mathbf{x}_K \mathbf{x}_{K_e}))^2 \quad (13)$$

are minimized by setting the gradients to zero solution of simple 2×2 linear systems. This method represents a good alternative among others to find the hyperplane [9, 15] because of its accuracy and robustness independently to the number of neighbours (an important property as it will be shown later with our wet/dry front treatment for a well-balanced scheme).

Limitating procedure In order to prevent from large numerical dispersive instabilities, the predicted vectorial slopes need to be limited. A first and efficient method consists to simply apply the maximum principle to the two edge unlimited reconstructed primitive variable calculated from (11) and (13) such that the two employed limited reconstructed variables $\mathbf{W}_{e,K}^n$ and \mathbf{W}_{e,K_e}^n at edge e verify,

$$\min(\mathbf{W}_K^n, \mathbf{W}_{K_e}^n) \leq \mathbf{W}_{e,K}^n, \mathbf{W}_{e,K_e}^n \leq \max(\mathbf{W}_K^n, \mathbf{W}_{K_e}^n) \quad (14)$$

what we call the MP limiter (for Maximum Principle). We observe in practice that it generates very moderate oscillations at solution singularities whereas diffusion is minimized comparatively to other classic limiters like Minmod or Van Albada, see also e.g. [3, 21, 12, 11]. Nevertheless, generation of new extremas and the presence of *wet/dry fronts* can break the Finite Volume mass conservation property in unacceptable proportion (the scheme is no longer positive). A first solution is to manage “by hand” the mass conservation at the end of each time step. The mass artificially added cutting-off to zero an eventual negative water depth is removed proportionally to neighboring wet cells. A second solution is to use the Barth limiter [4] for the water depth h ,

$$[\nabla \mathbf{W}_K^n]_i = \min_{e \in \partial K} (1, [\phi_{K,e}]_i) [\widetilde{\nabla \mathbf{W}_K^n}]_i$$

$$[\phi_{K,e}]_i = \begin{cases} 0 & \text{if } ([\mathbf{W}_{K_e}^n]_i - [\mathbf{W}_K^n]_i) [\nabla \mathbf{W}_K^n]_i \cdot \mathbf{x}_K \mathbf{x}_e < 0 \\ \frac{[\mathbf{W}_{K_e}^n]_i - [\mathbf{W}_K^n]_i}{[\nabla \mathbf{W}_K^n]_i \cdot \mathbf{x}_K \mathbf{x}_e} & \text{if not} \end{cases} \quad (15)$$

The reconstructed conservative variables vectors \mathbf{U}_K^n and $\mathbf{U}_{K_e}^n$ are then obtained by a simple multiplication.

7.3 Well-balanced schemes with wet/dry front treatment

If previous schemes are used with a naive topography gradient ∇z_b discretization in the gravity source term $\mathbf{S}_g(\mathbf{U})$ to perform real data simulations with large topography gradients (typically perpendicular to the streamlines), it is well known that it produces unacceptable numerical spurious velocities (citations). The reason is that there is no discrete balance between the hydrostatic pressure and the gravity source term ($\nabla (gh^2/2) \neq -gh\nabla z_b$). We present here two methods to enforce this balance property. A first one part of the class of well-balanced schemes of Chacón Rebollo and al. [8, 7]. And as this method has been found stable only at first order, we present also the second order well-balanced scheme due to Audusse and al. [2, 3] with a wet/dry front treatment (the first order version is obvious, no special wet/dry treatment is needed).

7.3.1 E-well-balanced scheme

In order to build a well-balanced scheme, these equalities must be verified when $z_K + h_K = z_{K_e} + h_{K_e}$ and $\mathbf{u}_K = \mathbf{u}_{K_e} = 0$,

$$\begin{aligned} \mathbf{F}_e^h(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K}) &= 0 \\ \sum_{e \in \partial K} m_e \mathbf{F}_e^{hu}(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K}) &= -g \sum_{e \in \partial K} m_e h_e (z_e - z_K) \mathbf{n}_{e,K} \end{aligned} \quad (16)$$

using the property $h\nabla z_b = \nabla(hz_b) - z_b\nabla h$ and the Green’s identity to find the piecewise constant approximation of $\mathbf{S}_g(\mathbf{U})$ incorporated in semi-discrete scheme (6). One has now to find h_e and z_e , reconstructions at edge e of the water depth and the topography respectively, to derive a well-balanced scheme with a consistent discretization of the gravitational source term $\mathbf{S}_g(\mathbf{U})$. Because of the geometrical property $\sum_{e \in \partial K} m_e \mathbf{n}_{e,K} = 0$, the

term $\mathbf{F}_e^{hu}(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K})$ can be subtracted in the first sum. After some calculations and by identification, one can easily verify using the HLLC approximate Riemann solver (7) to evaluate \mathbf{F}_e that these two reconstructions give the second equality in (16),

$$z_e = \frac{1}{2} (z_K + z_{K_e}) \quad \text{and} \quad h_e = \frac{s_K}{s_{K_e} - s_K} (h_K + h_{K_e}) \quad (17)$$

To find a zero mass flux, the water depths h_K and h_{K_e} in the diffuse upwinded part of the HLLC approximate Riemann solver are replaced by $z_K + h_K$ and $z_{K_e} + h_{K_e}$. At wet/dry fronts, the bottom topography is changed in a dry cell K_e such that $z_{K_e} = z_K + h_K$. This numerical method can be extended to second-order taking into account the reconstructed water depths at edge e but it has not been found stable in all cases. More generally, changing the diffuse upwinded part of the HLLC approximate Riemann solver cannot ensure a stable numerical model.

7.3.2 A-well-balanced scheme

Following the method of Audusse and al. [2, 3], a new vectorial slope for the variable $\eta = h + z$ is first calculated in addition to primitive variables ones (11). Then, the so-called hydrostatic reconstructed water depth $h_{e,K}^*$ is defined with the help of a reconstructed topography z_e at edge e ,

$$h_{e,K}^* = \max(0, h_{e,K} + z_{e,K} - z_e) \quad \text{with} \quad \begin{cases} z_{e,K} = \eta_{e,K} - h_{e,K} \\ z_e = \max(z_{e,K}, z_{e,K_e}) \end{cases} \quad (18)$$

The standard MUSCL reconstructed conservative variable vector $\mathbf{U}_{e,K}^n$ in semi-discrete scheme (12) is replaced by,

$$\mathbf{U}_{e,K}^* = \begin{bmatrix} h_{e,K}^* \\ h_{e,K}^* \mathbf{u} \end{bmatrix} \quad (19)$$

Including a constant discretization of the gravitational source term \mathbf{S}_g with the continuous formulation, we finally obtain a well-balanced scheme at second order,

$$\begin{aligned} \mathbf{U}_K^{n+1} &= \mathbf{U}_K^n - \frac{\Delta t^n}{m_K} \sum_{e \in \partial K} m_e \left(\mathbf{F}_e(\mathbf{U}_{e,K}^n, \mathbf{U}_{e,K_e}^n, \mathbf{n}_{e,K}) + \right. \\ &\quad \left. \mathbf{S}_g(\mathbf{U}_{e,K}^n, \mathbf{U}_{e,K}^n, \mathbf{n}_{e,K}) + \mathbf{S}_g^c(\mathbf{U}_K^n, \mathbf{U}_{e,K}^n, z_K, z_{e,K}, \mathbf{n}_{e,K}) \right) \\ \mathbf{S}_g(\mathbf{U}_{e,K}^n, \mathbf{U}_{e,K}^n, \mathbf{n}_{e,K}) &= \begin{bmatrix} 0 \\ \frac{g}{2} \left((h_{e,K}^*)^2 - (h_{e,K}^n)^2 \right) \mathbf{n}_{e,K} \end{bmatrix} \\ \mathbf{S}_g^c(\mathbf{U}_K^n, \mathbf{U}_{e,K}^n, z_K, z_{e,K}, \mathbf{n}_{e,K}) &= \begin{bmatrix} 0 \\ -\frac{g}{2} (h_{e,K}^n + h_K^n) (z_{e,K} - z_K) \mathbf{n}_{e,K} \end{bmatrix} \end{aligned} \quad (20)$$

Our numerical experiments showed that this scheme is not stable at wet/dry fronts perpendicular to the flow streamlines. Let us specify here some additional requirements in order to stabilize in all cases and for all limiters this second order well-balanced scheme. Firstly, all vectorial slopes are enforced to zero in dry cells (what can be viewed as a mask in practice). Secondly, these same dry cells are not taken into account in the least square method (13) to predict the vectorial slope $\nabla \eta^n$,

$$\begin{aligned} [\nabla \mathbf{U}_K^n]_i &= \mathbf{0} \quad \text{if} \quad h_K = 0 \\ E(\nabla \eta_K^n) &= \sum_{e \in (\partial K \setminus h_{K_e}=0)} (\eta_{K_e}^n - (\eta_K^n + \nabla \eta_K^n \cdot \mathbf{x}_K \mathbf{x}_{K_e}))^2 \end{aligned} \quad (21)$$

These conditions are related to the typical situation in Fig.(11) where the well-balanced property is desired with presence of a wet/dry front because the water surface level η is constant around the cell K . The dry cell K_e has a bottom topography z_{K_e} upper than the neighboring wet cell K water surface level η_K , the fluxes must be enforced to zero at the edge e between the two cells (water cannot “climb” in this situation) and the well-balanced property must be verified in the cell K .

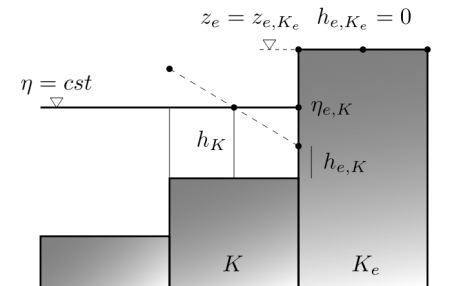


Figure 11: Typical situation of desired well-balanced property in presence of a wet/dry front.

Making the hypothesis that the normal velocity $\hat{u}_{e,K}$ is not necessary zero, a simple way to enforce $\mathbf{F}_e(\mathbf{U}_{e,K}^{*n}, \mathbf{U}_{e,K_e}^{*n}, \mathbf{n}_{e,K})$ in (20) to be zero is for the reconstructed hydrostatic water depths $h_{e,K}^*$ and h_{e,K_e}^* to be always zero (we remark that it is effectively always true for the first order version of the scheme since $z_e = \max(z_K, z_{K_e}) = z_{K_e} > h_K + z_K$, $h_{e,K}^* = \max(0, h_K + z_K - z_e) = 0$ and $h_{e,K_e}^* = \max(0, h_{K_e} + z_{K_e} - z_e) = 0$). On other side, a key-point in the second order well-balanced method of Audusse and al. [2, 3] is that the vectorial slope for the variable $\eta = h + z$ must be zero to find the well-balanced property in a cell K . The variable η must be constant in all neighboring cells of a wet cell which is not true in our typical case Fig.(11). So, in order to find a zero vectorial slope for the variable η in the cell K , the dry cell K_e is not taken into account in the least square method (13). Now, a consequence is that $\eta_{e,K} = h_K + z_K < \eta_{e,K_e} = z_{e,K_e}$ implying that $z_e = \max(z_{e,K}, z_{e,K_e}) = z_{e,K_e}$ and concluded that it is always find that $h_{e,K}^* = \max(0, \eta_{e,K} - z_e) = 0$. Since all vectorial slopes are enforced to zero in dry cells, it is obvious that we have also $h_{e,K_e}^* = 0$. The sum of the non negative components of the two sources terms $\mathbf{S}_g(\mathbf{U}_K^n, \mathbf{U}_{e,K}^{*n}, \mathbf{n}_{e,K})$ and $\mathbf{S}_g^c(\mathbf{U}_K^n, \mathbf{U}_{e,K}^{*n}, z_K, z_{e,K}, \mathbf{n}_{e,K})$ is reduce to $-\frac{g}{2}h_K^2\mathbf{n}_{e,K}$ because $z_{e,K} - z_K = h_K - h_{e,K}$ and it do not change the demonstration in [3] to find the well-balanced property in the cell K (that use the geometrical property $\sum_{e \in \partial K} m_e \mathbf{n}_{e,K} = 0$).

So, we have demonstrated that our additional requirements (21) to the original scheme (20) are sufficient to find the well-balanced property in presence of wet/dry interfaces without any consideration on limitation such it is done before. And it do not change the global scheme at wet/dry fronts because we only manipulate the predicted vectorial slopes. The scheme “falls at worst” at the first order in the cells with a neighboring dry cell.

7.4 Time stepping and friction source term implicitation

The friction source term \mathbf{S}_f can be discretized in time using a splitting approach, see e.g. [23, 29]. Noting $\bar{\mathbf{U}}_K^{n+1}$ the previous solution at time t^{n+1} relative to the well-balanced schemes at first and second order including a consistant discretization of the gravitationnal source term \mathbf{S}_g (20), the semi-implicit scheme including the friction source term \mathbf{S}_f in the discretization of the model (4) can be written in a general form as,

$$\mathbf{U}_K^{n+1} = \bar{\mathbf{U}}_K^{n+1} + \Delta t^n \mathbf{S}_f(\bar{\mathbf{U}}_K^{n+1}, \mathbf{U}_K^{n+1}) \quad (22)$$

Since the friction source term \mathbf{S}_f is zero in the mass conservation equation, we remark that $h^{n+1} = \bar{h}^{n+1}$ simplifying the development of a semi-implicit scheme. Now, in order to avoid the possible numerical instabilities at wet/dry fronts induced by the presence of $h^{1/3}$ at denominator in \mathbf{S}_f expression, one should introduce the “highest level” of implicitation in (22) and after some developments, this final expression can be derived,

$$\mathbf{U}_K^{n+1} = \begin{bmatrix} \bar{h}^{n+1} \\ h^{n+1} \bar{\mathbf{u}}^{n+1} \left(\frac{(h^{n+1})^{4/3}}{(h^{n+1})^{4/3} + \Delta t^n g n^2 \|\bar{\mathbf{u}}^{n+1}\|} \right) \end{bmatrix} \quad (23)$$

Liang and Marche [19] have derived a very similar expression by linearization of the full implicit method,

$$\mathbf{U}_K^{n+1} = \begin{bmatrix} \bar{h}^{n+1} \\ h^{n+1} \bar{\mathbf{u}}^{n+1} \left(\frac{(h^{n+1})^{4/3} + \Delta t^n g n^2 \|\bar{\mathbf{u}}^{n+1}\|}{(h^{n+1})^{4/3} + 2\Delta t^n g n^2 \|\bar{\mathbf{u}}^{n+1}\|} \right) \end{bmatrix} \quad (24)$$

But the **full implicit problem** that can be written in this form,

$$\|\mathbf{u}^{n+1}\| \mathbf{u}^{n+1} + c(\mathbf{u}^{n+1} - \bar{\mathbf{u}}^{n+1}) = 0 \quad \text{with} \quad c = \frac{2(h^{n+1})^{4/3}}{g n^2 \Delta t^n} \quad (25)$$

has an exact solution. Remarking that $\mathbf{u}^{n+1} = \alpha \bar{\mathbf{u}}^{n+1}$ with $\alpha \in]0, 1[$ reduce this system of non-linear equations to the resolution of a quadratic equation in α . The final analytical solution is,

$$\mathbf{U}_K^{n+1} = \begin{bmatrix} \bar{h}^{n+1} \\ h^{n+1} \bar{\mathbf{u}}^{n+1} \left(\frac{2(h^{n+1})^{2/3}}{(h^{n+1})^{2/3} + \sqrt{(h^{n+1})^{4/3} + 2\Delta t^n g n^2 \|\bar{\mathbf{u}}^{n+1}\|}} \right) \end{bmatrix} \quad (26)$$

We note that (23) and (24) can be obtained by Taylor series at first order of this exact solution of the full implicit problem. These three expressions (23,24,26) give the property $q^n q^{n+1} \geq 0$ and no limiting value is needed like it is done by Burguete and al. [6] to stabilize the method at wet-dry front. Now, in order to obtain a global second order solver as to stabilize in time the spatial second order well-balanced method, a time stepping strategy is needed. A first version can be obtained simply replacing the explicit splitted step for \mathbf{S}_f in

the classical second order SSP-RK2 method, see e.g. [14], by one of the three semi-implicit schemes presented above,

$$\begin{aligned}
\mathbf{U}_K^{(1)} &= \mathbf{U}_K^n + \Delta t^n \mathbf{L}(\mathbf{U}_K^n) \\
\mathbf{U}_K^{(2)} &= \mathbf{U}_K^{(1)} + \Delta t^n \mathbf{S}_f(\mathbf{U}_K^{(1)}, \mathbf{U}_K^{(2)}) \\
\mathbf{U}_K^{(3)} &= \mathbf{U}_K^{(2)} + \Delta t^n \mathbf{L}(\mathbf{U}_K^{(2)}) \\
\mathbf{U}_K^{(4)} &= \mathbf{U}_K^{(3)} + \Delta t^n \mathbf{S}_f(\mathbf{U}_K^{(3)}, \mathbf{U}_K^{(4)}) \\
\mathbf{U}_K^{n+1} &= \frac{1}{2} (\mathbf{U}_K^n + \mathbf{U}_K^{(4)})
\end{aligned} \tag{27}$$

If this time splitting strategy is stable and preserve positivity, it is only first order accurate. In order to derive a second order time splitting with an full implicit discretization of the source term, Pareschi and Russo in [22] derive an implicit-explicit (IMEX) Runge-Kutta method for hyperbolic conservation laws with stiff relaxation terms. The original scheme named IMEX-SSP(3,2,2) have been choosen and transformed for numerical implementation purpose writes as follow,

$$\begin{aligned}
\mathbf{U}_K^{(1)} &= \mathbf{U}_K^n + \frac{\Delta t^n}{2} \mathbf{S}_f(\mathbf{U}_K^{(1)}) \\
\mathbf{U}_K^{(2)} &= 2\mathbf{U}_K^n - \mathbf{U}_K^{(1)} + \frac{\Delta t^n}{2} \mathbf{S}_f(\mathbf{U}_K^{(2)}) \\
\mathbf{U}_K^{(3)} &= \mathbf{U}_K^n + \Delta t^n \mathbf{L}(\mathbf{U}_K^{(2)}) \\
\mathbf{U}_K^{(4)} &= \mathbf{U}_K^{(1)} + \mathbf{U}_K^{(2)} + \mathbf{U}_K^{(3)} - 2\mathbf{U}_K^n + \frac{\Delta t^n}{2} \mathbf{S}_f(\mathbf{U}_K^{(4)}) \\
\mathbf{U}_K^{(5)} &= \mathbf{U}_K^n + \Delta t^n \mathbf{L}(\mathbf{U}_K^{(4)}) \\
\mathbf{U}_K^{n+1} &= \frac{1}{2} (\mathbf{U}_K^{(5)} - \mathbf{U}_K^{(3)}) + \mathbf{U}_K^{(4)}
\end{aligned} \tag{28}$$

The three stages implicit part of this time splitting is a DIRK scheme [22] respecting A-stability, L-stability and is stiffly accurate ($R(\infty) = 0$) like it is demonstrated in [16]. Such stability properties allow to deal with wet/dry front where the friction source term \mathbf{S}_f can “as stiff as possible” since the water depth is vanishing. The explicit part is the same SSP-RK2 scheme presented above. This version of the DIRK scheme has been found important to treat non trivial boundaries conditions without any special correction.

7.5 Boundary conditions

Main classic idea to manage the boundary conditions is to define ghost conservative vectors, $\mathbf{U}_{K_e}^G$ for the first order scheme and \mathbf{U}_{e,K_e}^G for the second order one, that are directly used in the well-balanced scheme (20) when the edge e is at the boundary of the computational domain Ω (the well-balanced schemes have a consequence that is not easy to directly impose the numerical flux). The notation $\mathbf{U}_{K_e}^G$ denotes a piecewise constant approximation of \mathbf{U} in a virtual cell K_e that is not in the computational domain but exactly used as other ones in the first order scheme. To simulate the presence of a wall, these primitive variables ghost values are specified,

$$\begin{cases} h_{K_e}^G = h_K \\ \hat{u}_{K_e}^G = -\hat{u}_K \\ \hat{v}_{K_e}^G = \hat{v}_K \end{cases} \tag{29}$$

Now, if we want to impose a discharge Q_{in} to a part Γ_{in} of the computational domain boundary ∂K , one way to achieve this goal is to specify these primitive variables ghost values,

$$\begin{cases} h_{K_e}^G = h_K \\ \hat{u}_{K_e}^G = \frac{Q_{\text{in}} h_K^{2/3}}{\sum_{e \in \Gamma_{\text{in}}} m_e h_K^{5/3}} \\ \hat{v}_{K_e}^G = \hat{v}_K \end{cases} \tag{30}$$

One can easily verify that $\sum_{e \in \Gamma_{\text{in}}} m_e \hat{q}_{K_e}^G = Q_{\text{in}}$ and $\hat{q}_{K_e}^G = c h_K^{5/3}$. This last condition corresponds to the equilibrium between the gravitationnal and friction source terms. This boundary condition can be viewed as an approximation of the backwater curve since a Neumann boundary condition is applied for the water depth. Nevertheless, when the considered wet surface corresponding to Γ_{in} is non trivial, practice show that one can observe the wrong imposed discharge Q_{in} (as it will be show later). In order to overcome this drawback, a solution is to apply a feedback process on the associated ghost bathymetry,

$$z_b^{\text{new}} = z_b^{\text{old}} + c (\mathbf{F}_e^h(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K}) - \hat{q}_{K_e}^G) \quad (31)$$

This process correct the ghost bathymetry z_b^G at each time step to finally converge to a value implying that the mass flux $\mathbf{F}_e^h(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K})$ is stricly equal to the imposed lineic discharge $\hat{q}_{K_e}^G$.

In order to control the water flow at a part Γ_{out} of the computational domain boundary ∂K , a water depth or a rating curve $Q_{out}(h)$ is imposed such that,

$$\begin{cases} h_{K_e}^G &= h_{\text{user}} \text{ or } f_{\text{user}}(Q_{out}) \\ \hat{u}_{K_e}^G &= \hat{u}_K + 2 \left(\sqrt{g(h_K - h_K^G)} \right) \\ \hat{v}_{K_e}^G &= \hat{v}_K \end{cases} \quad (32)$$

Imposing the water depth in the ghost cell directly or linearly interpolated from a tabulated rating curve $\eta(Q_{out})$ with $Q_{out} = \sum_{e \in \Gamma_{out}} m_e \mathbf{F}_e^h(\mathbf{U}_K^n, \mathbf{U}_{K_e}^n, \mathbf{n}_{e,K})$ to be consistant with the manner to impose the discharge Q_{in} (at the condition to relax the relation $\eta(Q_{out})$ in time).

In the case of the second-order scheme that needs to reconstruct the conservative variable vector \mathbf{U}_{e,K_e}^G at ghost side of edge e , $\mathbf{U}_{K_e}^G$ is first used in the least square method (13) to calculate the vectorial slope in the interior cell K . The conditions (29), (30) and (32) are then used a second time replacing \mathbf{U}_K by $\mathbf{U}_{e,K}$ to obtain \mathbf{U}_{e,K_e}^G . Finally, The hydrostatic reconstructed water depth \mathbf{U}_K^{G*} and $\mathbf{U}_{e,K}^{G*}$ are calculated in the same way through equations (18) and (19) with $z_{e,K_e}^G = z_{e,K_e}$.

8 Validation

This section is dedicated to the validation of the numerical schemes implemented in the DASSFLOW software presenting several challenging and relevant test cases in context of hydraulic and environmental studies : dam break, channel and run-up problems with dynamic wet/dry fronts and considering non-linear friction. We will show the formally global second-order convergence in space and time of our shallow water numerical model with the Manning-Strickler friction source term as the robust treatment of wet/dry fronts.

In order to quantify the numerical errors, we define these relative error norms,

$$e_p(x) = \frac{\|x^{\text{num}} - x^{\text{exact}}\|_p}{\|x^{\text{exact}}\|_p} \quad , \quad e_p^T(x) = \frac{1}{T} \int_0^T e_p(x) dt \quad (33)$$

$$\|x\|_p = \left(\sum_{K \in \Omega} m_K |x_K|^p \right)^{1/p}$$

where x^{num} is the numerical solution with n cells of discretization and x^{exact} is the analytical solution of the test case or the converge reference solution when the analytical solution is not known (x_i^{exact} is properly numerically integrated with desired accuracy in sense of used Finite Volume schemes).

8.1 Water at rest

We consider in a box 1000 meter long a zero bed elevation, randomly perturbed cell by cell with a maximum amplitude of 1 m ,

$$z_b = r \quad \text{with} \quad r \in [-1, 1] \quad (34)$$

If the simulation is initialized with a water elevation of zero, i.e. $z_b + h = 0$, the resulting “lake” should stay at rest. After a simulation time of 3600 s , the results with different schemes are showed in the Fig.(12). The norms of the final velocities are given in the Tab.13.

	1°order	1°order / WB-A	2°order	2°order / MP limiter / WB-A / CFL 0.5	2°order/ MP limiter / WB-A / CFL 0.25	2°order/ Barth limiter / WB-A
$\ \mathbf{u}\ _\infty$	2.30	$1.04 \cdot 10^{-13}$	2.23	$1.45 \cdot 10^{-2}$	$9.66 \cdot 10^{-14}$	$7.66 \cdot 10^{-14}$
$\ \mathbf{u}\ _1$	1.14	$6.92 \cdot 10^{-16}$	0.95	$1.95 \cdot 10^{-5}$	$9.40 \cdot 10^{-16}$	$6.97 \cdot 10^{-16}$

Table 13: Water at rest test case, absolute infinity and L_1 norms of the final velocities.

8.2 Parabolic bowl with linear friction

Analytical solutions of the nonlinear shallow water equations considering a linear friction and a parabolic bottom topography were derived by Sampson [24], following the work of Thacker [28], by considering a linear friction in replacement of Coriolis force. These solutions involve a flat surface oscillating motion with wet/dry fronts, decaying over time because of friction. It gives a very popular test for a shallow water numerical solver [3, 19, 25, 18, 17]. The solution is,

$$\begin{aligned} \eta(x, t) &= h_0 + \frac{a^2 B^2 e^{-\tau t}}{8g^2 h_0} \left(-s\tau \sin 2st + \left(\frac{\tau^2}{4} - s^2 \right) \cos 2st \right) \\ &\quad - \frac{B^2 e^{-\tau t}}{4g} - \frac{e^{-\tau t/2}}{g} \left(Bs \cos st + \frac{\tau B}{2} \sin st \right) x \\ u(x, t) &= B e^{-\tau t/2} \sin st \end{aligned} \quad (35)$$

with $p = \sqrt{8gh_0}/a$ and $s = \sqrt{p^2 - \tau^2}/2$ (and then valid if $p > \tau$). We choose two set of parameters,

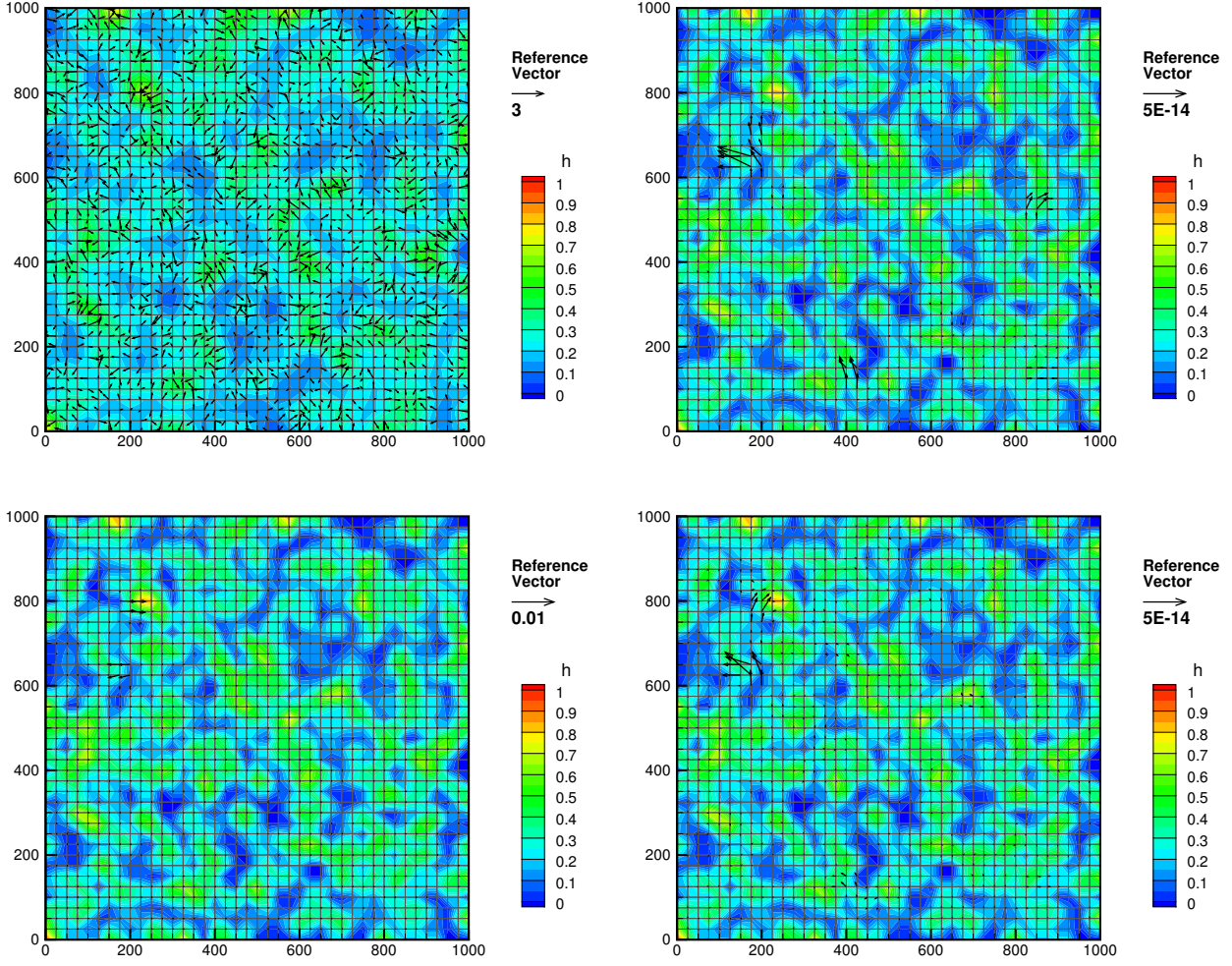


Figure 12: Water at rest test case, final water depth and velocities; (up-left) first order scheme; (up-right) well-balanced first order scheme; (down-left) well-balanced second order scheme with the MP limiter and a CFL number of 0.5; (down-right) well-balanced second order scheme with the Barth limiter.

$$\text{set 1} \quad \begin{cases} h_0 = 10 \text{ m} \\ a = 3000 \text{ m} \\ B = 5 \text{ m.s}^{-1} \\ \tau = 0.001 \text{ s}^{-1} \end{cases} \quad \text{set 2} \quad \begin{cases} h_0 = 1 \text{ m} \\ a = 30 \text{ m} \\ B = 1 \text{ m.s}^{-1} \\ \tau = 0.2 \text{ s}^{-1} \end{cases} \quad (36)$$

The first one corresponding to the classic parameters chosen into litterature as in [19, 18]. The Fig.(13) shows the calculated relative error norms $e_1^T(h)$ with this first set of parameters using the MP limiter for the second order well-balanced scheme. One can observe the correct convergence behavior of all well-balance schemes and demonstrate that the moving wet-dry fronts are robustly captured (a cut-off water depth $h_\epsilon \in [10^{-4}, 10^{-8}]$ have been used for these simulations in order to find a linear convergence behavior in log-log scale). A rate of convergence of approximatively 1.5 is found for the second-order well-balance scheme with the hybrid SSP-RK2 time splitting as with the IMEX time splitting (It is found a relative error norm $e_1^T(h) \simeq 2.10^{-3}$ for $n = 100$ which is approximatively the same in [19]). This loss of second order accuracy is most probably due to the presence of the two moving wet/dry fronts introducing singularities in the solution. But a most interesting observation is that there is no major difference between the two time splitting strategy despite the fact that the hybrid RK2 is first order accurate and the IMEX second order accurate, except maybe for very fine meshes. This demonstrates that the spatial consistency error is greater than the time one despite the accuracy slopes difference. If simulations are performed now with the second set of paramaters (36), then it is obtained a significant difference between the two time splitting methods Fig.(13). After an asymptotic first order behavior for coarse meshes, the hybrid RK2 time splitting gives a rate of convergence of 1 and the IMEX time splitting maintains the same 1.5 rate of convergence obtained with the first classic set of parameters (36).

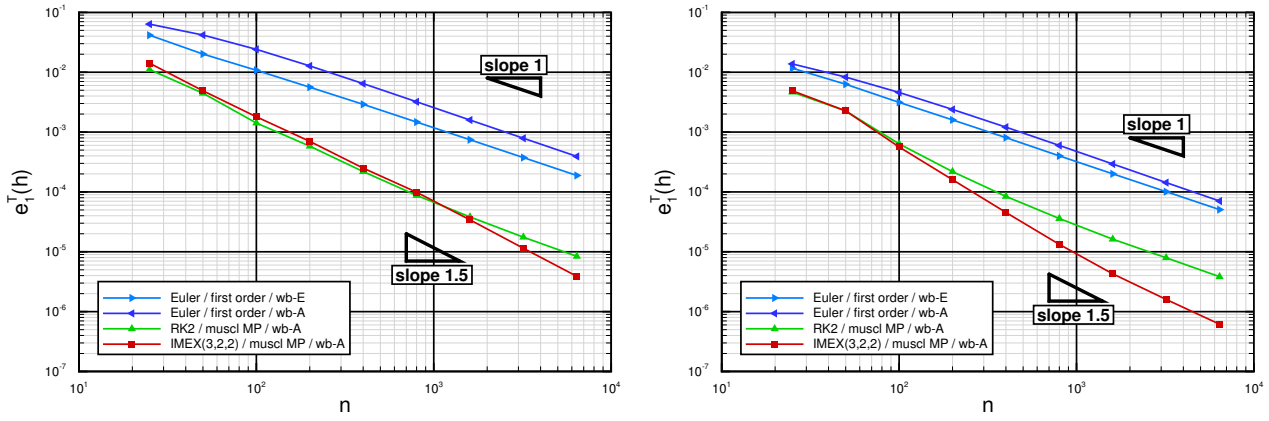


Figure 13: (left) Rate of convergence for the norm $e_1^T(h)$ with set 1 of physical parameters, n is the cells number (right) Same with set 2.

Nevertheless, the friction source term is linear for these simulations and it does not correspond to our original model where the Manning-Strickler friction source term is non-linear. The singularities at the moving wet/dry fronts cannot guarantee to measure precisely the right accuracy of the numerical methods of regular solutions. In conclusion, this test case is interesting to check robustness to deal with wet/dry fronts, but largely insufficient to be relevant to test the behavior of our numerical solver in context of hydraulic and environmental studies.

8.3 Dam break(s) with slope and non-linear friction

Numerical model behavior is investigated for dam break problems with a non zero slope and with the non-linear Manning-Strickler friction source term vanishing with the water depth at wet/dry front.

8.3.1 “Failling Gaussian”

A “regularised” dam break problem is first performed in sense that all eventual singularities in numerical solution are eliminated by construction. Considering the following initial condition,

$$\begin{cases} z_b(x) = 0.5 e^{-\frac{(x-l_x/2)^2}{2\sigma^2}} \\ h(x, t=0) = 0.1 + 0.5 e^{-\frac{(x-l_x/2)^2}{2\sigma^2}} \end{cases} \quad (37)$$

with $\sigma = 100 \text{ m}$, a computational domain length $l_x = 1000 \text{ m}$, a Manning-Strickler roughness coefficient $n = 0.05$ and a gravitational acceleration $g = 10 \text{ m.s}^{-1}$, a converge simulation is performed with a 12800 cells mesh at time $t = 100 \text{ s}$. Initial condition and converge result are plotted in Fig.14. One can note that these parameters are suitable in context of hydraulic. The topography gradient is chosen not constant in order to avoid an exact calculation of the topography gradient. The rates of convergence for the norms $e_1(h)$ and $e_1(q)$ showed on Fig.14 give a very clear accuracy result. As the hybrid RK2 time stepping method is only first order accurate, the same overall first order convergence rate is found than for the spatial first order well-balanced scheme. The relative error norms are still smaller in significant proportion due to the enhanced spatial accuracy at second order. Concerning the IMEX time stepping method, a second-order rate of convergence is found. As a consequence, the relative error norms become very small even for very coarse meshes. This demonstrates the global second-order convergence rate of presented numerical model with MUSCL linear reconstruction with appropriate limitation and IMEX time stepping. Nevertheless, one may be careful that this rate of convergence will be smaller with singularities in numerical solution as with non trivial boundary conditions.

8.3.2 Flat slope and wet/dry front

The idealized dam break problem involving a dynamic wet/dry front is now investigated. The non-linear Manning-Strickler friction is taken into account and a non zero flat slope is considered as it is sketched in Fig.15. The parameters used are a computational domain length $l_x = 1000 \text{ m}$ with wall boundaries at each

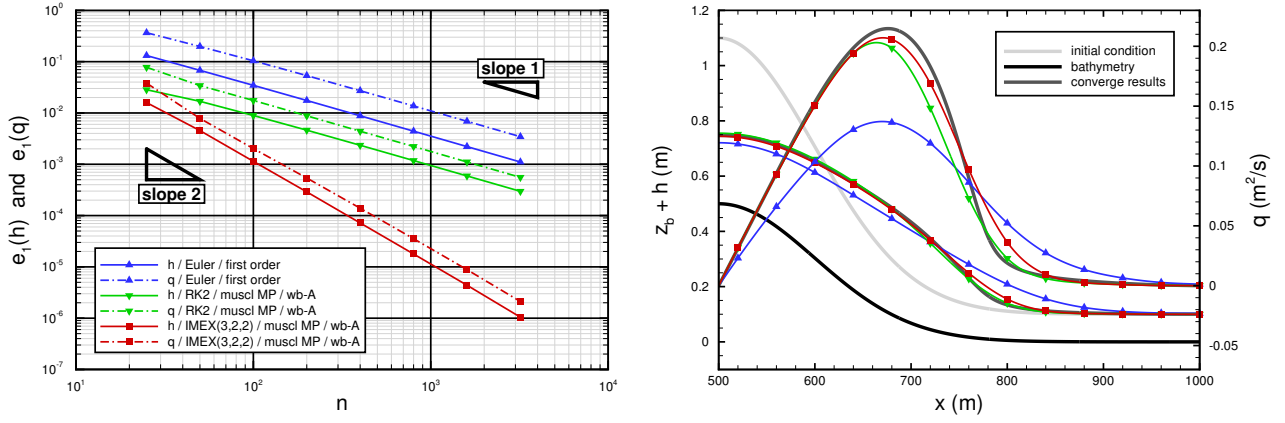


Figure 14: (left) Rates of convergence for the norms $e_1(h)$ and $e_1(q)$, n is the cells number (right) Half-domain initial condition, converge simulation with 12800 cells and relative simulations with 25 cells.

side, a slope $s = 0.5 \%$, a Manning-Strickler roughness coefficient $n = 0.05$ and a gravitational acceleration $g = 10 \text{ m.s}^{-1}$. The water column has a length $L = 50 \text{ m}$ and two height are considered, $H = 1$ or 5 m .

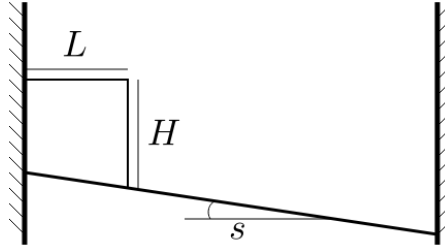


Figure 15: dam-break problem sketch.

As no analytical solution can be derived, two converge simulations are again performed with 12800 cells for $H = 5 \text{ m}$ at time $t = 500 \text{ s}$ and for $H = 1 \text{ m}$ at time $t = 2000 \text{ s}$. After the initial simulation time, a rarefaction wave goes upstream and interacts with the left wall boundary while a shock wave goes downstream, both modifying the initial water column shape. After a sufficient simulation time, the water column completely disappears and the new water shape exhibits more mass and stronger gradients downstream. The dynamic wet/dry front is robustly captured by well-balanced first- and second-order schemes without any cut-off water depth h_ϵ , despite that the well-balanced property is needless for this problem. We have verify that the schemes are stricly positive and do not generate any negative water depth.

Observing the relative error norms $e_1(h)$ in Fig.16 and Fig.17, on can first note that the previous obtained second-order convergence rate is loosed due to the singularities in the numerical solution and especially at wet/dry front. Nevertheless, the relative error norm $e_1(h)$ is smaller for a given grid size using the IMEX time stepping rather than the hybrid RK2 one and even more comparatively to the first order schemes. If there is no major difference between the schemes with or without the well-balanced property when initially $H = 5 \text{ m}$, this difference is obvious when $H = 1 \text{ m}$. The well-balanced second-order scheme with the hybrid

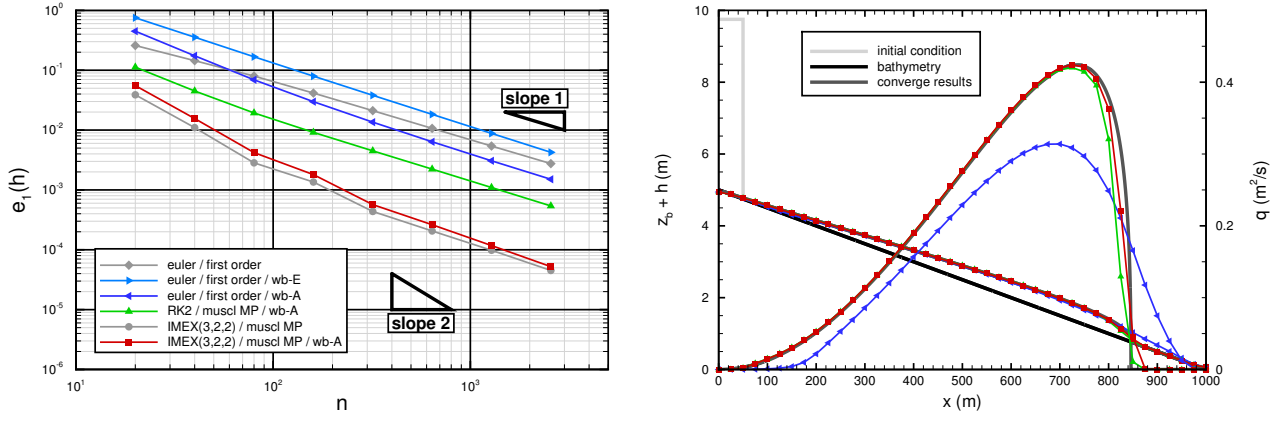


Figure 16: Idealized dam break problem with $H = 5$ m. (left) rates of convergence for the norm $e_1(h)$ for different schemes; (right) corresponding initial condition, converge simulation with 12800 cells and results with 100 cells.

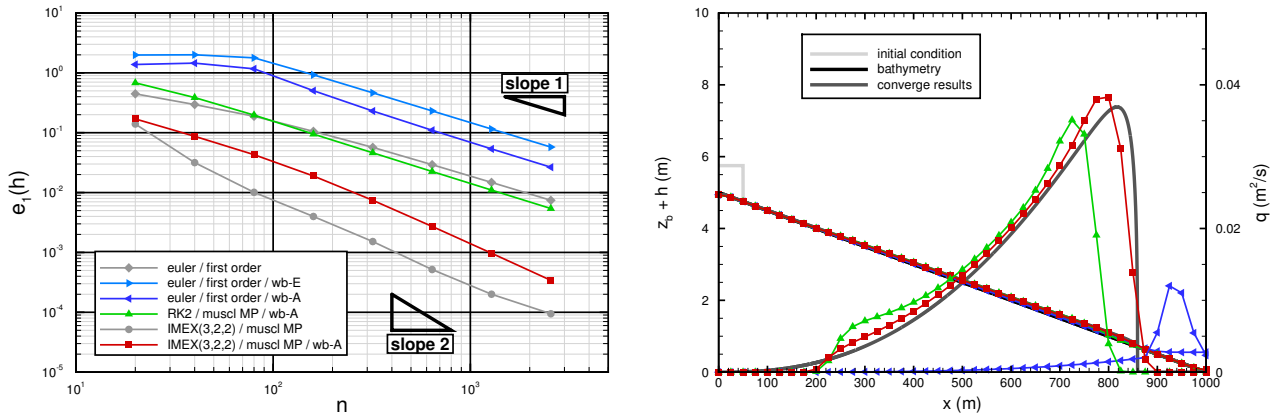


Figure 17: Idealized dam break problem with $H = 1$ m. (left) rates of convergence for the norm $e_1(h)$ for different schemes; (right) corresponding initial condition, converge simulation with 12800 cells and results with 100 cells.

RK2 time stepping gives a result very similar to the first order scheme without the well-balanced property. This indicates a drawback in both well-balanced methods when the water depth is close to the bathymetry difference. Sufficiently important to give very wrong numerical solutions with first order schemes. In conclusion, in order to use these methods to simulate hydraulic flows, one may be careful to verify this criteria to produce meshes.

8.4 simple channel

In order to perform river flood simulations with real topography, it is interesting now to check the numerical model behavior to simulate an open-channel flow with a non constant topography gradient.

8.4.1 Mac Donald's benchmark

As it would be too restrictive to consider a channel with a constant topography gradient, the Mac Donald's analytic steady solution for open-channel flow is preferred [20]. Making the hypothesis of a constant discharge in space, it can be easily verify that the steady states verify,

$$\begin{aligned} \partial_t h &= \partial_t q = \partial_x q = 0 \\ \partial_x z_b &= \left(\frac{q^2}{gh^3} - 1 \right) \partial_x h - \frac{n^2 q^2}{h^{10/3}} \end{aligned} \quad (38)$$

that gives analytical steady solutions with a non necessary constant topography gradient. We consider like in [10] a computational domain length $l_x = 1000$ m, a lineic discharge $q = 2$ m²/s and a Manning-Strickler roughness coefficient $n = 0.033$. Considering this exact water depth,

$$h^{\text{exact}}(x) = \left(\frac{4}{g}\right)^{1/3} \left(1 + \frac{1}{2} \exp\left(-16 \left(\frac{x}{1000} - \frac{1}{2}\right)^2\right)\right) \quad (39)$$

insuring that the flow is subcritical in all the channel, the bottom topography is numerically integrated with appropriate accuracy that gives the xyplot Fig.18.

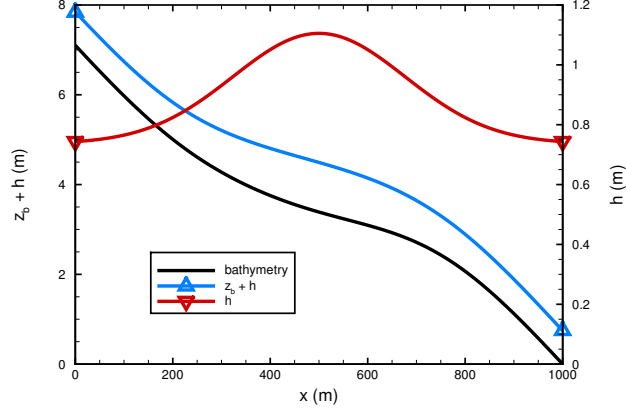


Figure 18: Open-channel steady state solution (38) considering the exact water depth (39).

The discharge is imposed upstream of the flow and the exact water depth is prescribed downstream. Initialising the simulations with dry cells except at the first cell at inflow, the steady states are properly achieved in time. The relative error norms $e_1(h)$ and $e_1(q)$ showed in Fig.19 give the same result than for the previous “regularised” dam break problem. The well-balanced second order scheme with the hybrid RK2 time stepping is only first order accurate with a smaller error than for the first order whereas with the IMEX time stepping the numerical model is formally at second order with very small errors even for very coarse meshes. A surprising result is that this rate of convergence is obtained with non trivial inflow/outflow boundary conditions even if this benchmark problem is only onedimensional.

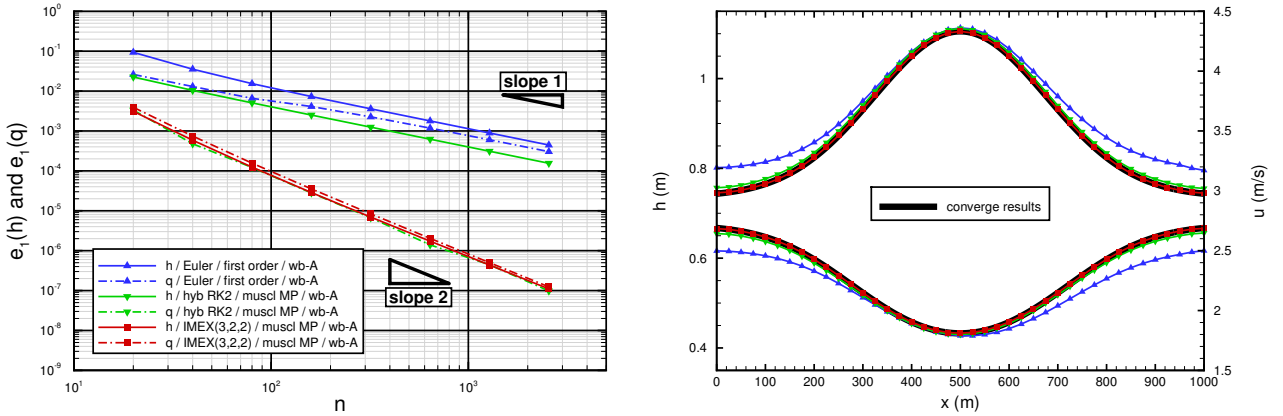


Figure 19: Open-channel benchmark results : (left) relative error norms $e_1(h)$ and $e_1(q)$, n is the cells number; (right) exact solutions for h and u and simulations with 25 cells.

8.4.2 perturbed topography

The topography is not so smooth in real-life than in the previous test case where a channel 1000 m long was considered. It is tested here the numerical model introducing gradually in frequency some perturbations to an initial channel 10000 m long with a constant slope of 0,25 % such that,

$$z_b = z_b + \sum_{i=1}^{n_p} 0.25 \sin \left(2\pi \left(\frac{p_i}{l_x} x + r_i \right) \right) \quad \text{with } r_i \in [0, 1] \quad (40)$$

with $n_p = 1, 2, 3, 4, 5$ and 6 and $p_i = 7, 24, 57, 108, 205$ and 402 . In order to control the flow at inflow and outflow, and especially the backwater curve at the inflow, the topography gradient is kept constant close to taking care to maintain perturbations spatially continuous. The other parameters are a Manning-Strickler roughness coefficient $n = 0.05$, a lineic discharge $q = 1 \text{ m}^2/\text{s}$ and a gravitational acceleration $g = 10 \text{ m.s}^{-1}$. Since the exact solution for the initial constant slope channel is $h = 1 \text{ m}$ (and $u = 1 \text{ m/s}$), the topography perturbations are of the same order of magnitude. The opposite case would be obviously irrelevant. Main goal is to find a criteria that insures a good accuracy to apply the numerical model with real topography. Simulations are performed with the well-balanced second-order scheme and the IMEX time stepping until the steady state is properly found. For each of these six cases, a converge simulation is performed with 12800 cells of discretization showed at the left of the Fig.20. One can note that the flow respects the shallow water model hypothesis since the smaller wavelength has a value of $25 h$.

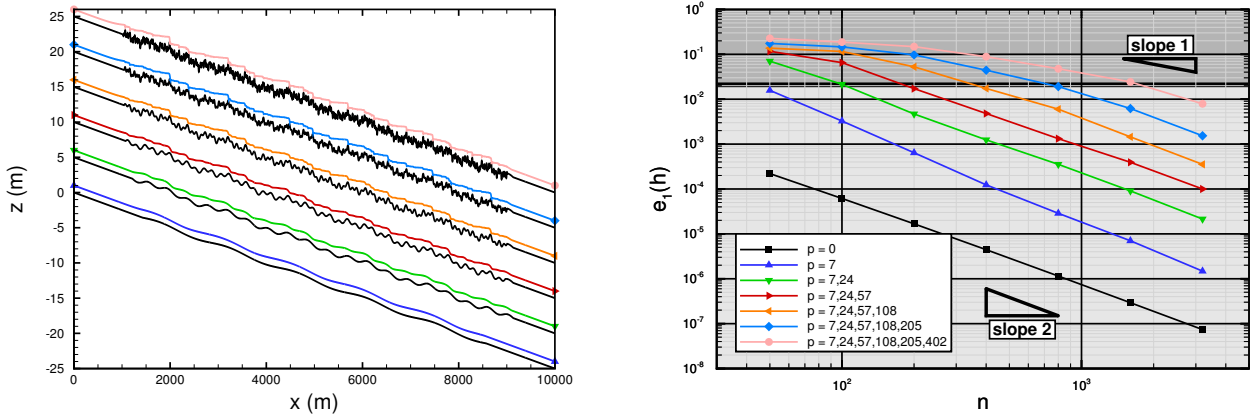


Figure 20: Channel 10000 m long with a constant slope of 0,25 % introducing gradually in frequency some perturbations in topography according to (40) : (left) six cases with channel bottom topography and associated converge simulations; (right) associated relative error norm $e_1(h)$ with n cells of discretization (the greyer zone corresponds to less than 4 cells of discretization for the highest frequency for each case).

The calculation of the relative error norms $e_1(h)$ evolving the mesh step size n shows in Fig.20 that introducing gradually in frequency some perturbations in the topography damages the simulations accuracy. This phenomenon is in a first time likely proportional to the highest perturbation frequency before saturating when the mesh is too coarse comparatively to the highest frequency introduced. The criteria to maintain a good accuracy as the right correct second-order convergence rate is to discretize the channel topography with a minimum of four points for the highest perturbation frequency.

8.5 Solitary wave on a simple beach

It is well know that the shallow water equations can be a suitable model to study numerically storm surges as the complete life-cycle of a tsunami (generation, propagation and run-up on the coast). The classic benchmark problem was introduced by Synolakis and al. [27] deriving an adimensionnal analytical solution for the run-up of a solitary wave on a simple sloping beach. Following the parameters retained in the NOAA technical report for evaluation of tsunami numerical models [26], the initial condition with dimensional parameters sketched in Fig.21 is,

$$\eta(x, 0) = H \operatorname{sech}^2 \left(\gamma \left(\frac{x - x_0 - L}{d} \right) \right) \quad \text{and} \quad u(x, 0) = -\sqrt{\frac{g}{d}} \eta(x, 0) \quad (41)$$

with $\cot \beta = 19.85$, $\gamma = \sqrt{3H/4d}$ and $L = \operatorname{arccosh}(\sqrt{20})/\gamma$. The wave height satisfies the adimensionnal parameter $H/d = 0.0185$. The computational domain is $100 d$ long, the mesh step size respects $\Delta x = d/10$ and wall boundaries conditions are prescribed. Using the well-balanced second-order numerical scheme with the MP limiter and IMEX time stepping, resulting water levels profiles and two time series are plotted in Fig.22 as superposed the nonlinear analytical solution.

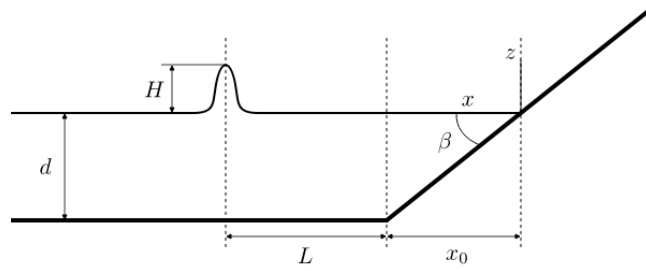


Figure 21: Dimensional sketch of the solitary wave run-up benchmark problem sketch.

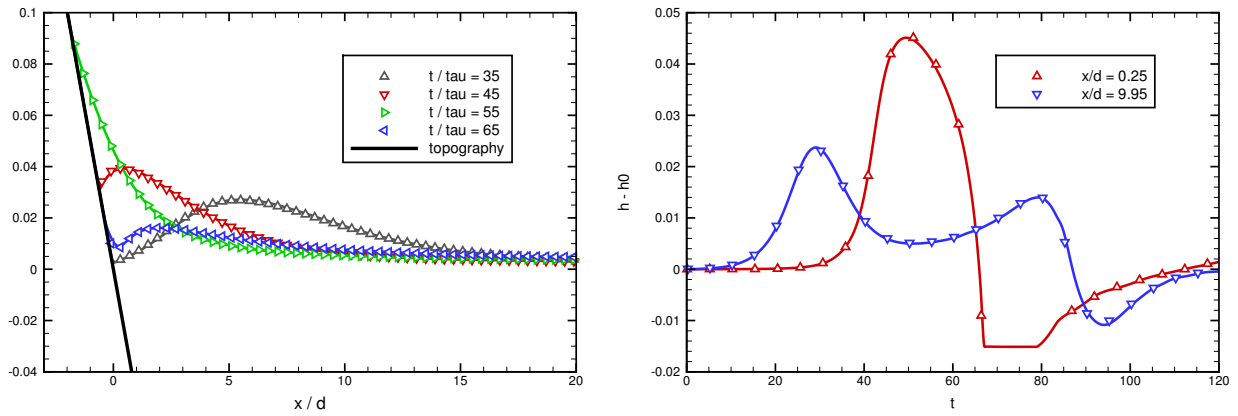


Figure 22: Run-up of a solitary wave on a simple beach : comparison of analytical solution (scatters) versus numerical solution (solid lines).

According to Synolakis and al. in [26] page 5, “any well benchmarked code should produce results within 5% of the calculated value from the analytical solution” which is the case for our simulation which is spatially everywhere very close to the analytical solution. There is no particular “distortion” in the numerical solution at adimensional time $t/\tau = 45$ and 65 in proximity to the wet/dry front whereas other numerical models can produce some like it is showed in the NTHMP workshop report [1]. This demonstrates that our well-balanced treatment at wet/dry front do not affect a wave run-up computation accuracy.

9 Cost functions

Computation of the cost function:

Computation of the cost function is done by the subroutine *calc_cost_function(cost,mesh)* of the file *src/common/m_obs.f90*.

Result *cost* is the sum of 3 *cost_part* (*cost=cost_part(1)+cost_part(2)+cost_part(3)*):

- *cost_part(1)*:

$$\sum_{i=1,Nobs} \omega_i \sum_{j=1,Ntimes} (\bar{h}_{i,j} - \bar{h}_{i,j}^{obs})^2 + \sum_{i=1,Nobs} \omega_i \sum_{j=1,Ntimes} (\bar{w}_{i,j} - \bar{w}_{i,j}^{obs})^2$$

$$\sum_{i=1,Nobs} \omega_i \sum_{j=1,Ntimes} (innovation_{i,j}) + \sum_{i=1,Nobs} \omega_i \sum_{j=1,Ntimes} (innovW_{i,j})$$

- *cost_part(2)*: Bathy regularization
- *cost_part(3)*: Hydrograph regularisation

Computation of the innovation and innovW vector:

Computation of the innovation vector is done by the subroutine *calc_innovation(dof,mesh)* of the file *src/sw_mono/read_write_obs.f90* and *innovation_{i,j}* is given by:

$$innovation_{i,j} = \bar{h}_{i,j} - \bar{h}_{i,j}^{obs}$$

with $\bar{h}_{i,j}$ compute as :

$$\frac{1}{S_{wet}} \int_{S_{wet}} h dx, \frac{1}{S_{obs}} \int_{S_{obs}} h dx \text{ or } \frac{1}{S_{river}} \int_{S_{obs}} h dx \text{ (see code)}$$

Computation of the innovW vector is done by the subroutine *calc_innovW(dof,mesh)* of the file *src/sw_mono/read_write_obs.f90* and *innovW_{i,j}* is given by:

$$innovW_{i,j} = \bar{w}_{i,j} - \bar{w}_{i,j}^{obs}$$

with $\bar{w}_{i,j}$ compute as:

$$\frac{S_{wet}}{L_i}$$

Computation of observation ($\bar{h}_{i,j}^{obs}$ and $\bar{w}_{i,j}^{obs}$):

The computation of $\bar{h}_{i,j}^{obs}$ and $\bar{w}_{i,j}^{obs}$ are done in the subroutine *sub_write(file_type)* of the file *src/sw_mono/read_write_obs.f90*. It is the same computation as $\bar{h}_{i,j}$ and $\bar{w}_{i,j}$.

10 Python scripts

- **createDassFlowObs.py**: Create a group of cells (stations_with_grp) file from an area (define with 4 parameters : x_{min} , x_{max} , y_{min} , y_{max}) and a result file.
 - ▷ nameFile: Name of output file. Ex: 'station_grp_0001.txt'
 - ▷ resultFile: Name of input result file. Ex: './bin/res/result_initial.dat'
 - ▷ xlimInf: Lower bound in x . Ex: 450
 - ▷ xlimSup: Upper bound in x . Ex: 550
 - ▷ ylimInf: Lower bound in y . Ex: -300
 - ▷ ylimSup: Lower bound in y . Ex: 300
 - ▷ outputDirectory: Output directory. Ex : './bin/'
- **meshToShapefile.py**: Create from a DassFlow mesh a shapefile file of the mesh (node and edge)
 - ▷ meshFile: Name of the input mesh file. Ex: './bin/test_case_deb.msh'
 - ▷ outputDirectory: Name of the output directory file. Ex: './bin/mesh'
 - ▷ TraceNode: Boolean if true, nodes are printed, else no. Ex: True
 - ▷ TraceEdge: Boolean if true, edges are printed, else no. Ex: True
 - ▷ EPGScode: EPGS code of the output
- **gmeshMeshToShapefile.py**: Same than DassFlowMeshToShapefile.py but with gmesh file.
- **resultToShapefile.py**: Create shapefile file from DassFlow result.
 - ▷ resultFile: Name of the result file. Ex: './bin/res/result_final.dat'
 - ▷ outputDirectory: Name of the output directory. Ex: './bin/res/shapefile_1'
 - ▷ EPGScode: EPGS code of the output
- **resultToShapefileAll.py**: Create shapefile file from all DassFlow result.
- **stationObservationToShapefile.py**: Create shapefile file of observation. Each points of the shapefile represent a observations stations or a points of a station with group.
- **gmeshMeshChangeBathy.py**: Change the bathy of a gmesh file from a bathy function.
 - ▷ bathy(x,y): Bathymetry function
 - ▷ meshFile: Path of the original Gmesh file. Ex: './bin/test_1.msh'
 - ▷ meshFileOut: Path of the output Gmesh file with new bathy. Ex: './bin/test_2.msh'
- **modifobs.py**: Change observation file (in order to add noise for example)
 - ▷ numberofobs: Number of observation file.
- **grad.py**: Plot compute gradient for hydrograph.
- **hydrograph.py**: Plot result of the minimization (for hydrograph).
 - ▷ pathOutFile: Path of the output directory. Ex: './bin/plotHydrograph/' .
 - ▷ pathTargetHydrograph: Path of the target hydrograph. Ex: './bin/hydrograph_target.txt'
 - ▷ pathFirstGuess: Path of the first guess hydrograph. Ex: './bin/hydrograph.txt'
 - ▷ legend: Boolean if true plot legend, else no. Ex: True.
 - ▷ legendTime: Time used ('seconds', 'hours', 'days').
 - ▷ plotMin: Boolean if true plot the result of the minimization else no.
 - ▷ plotObs: Boolean if true plot time observation for each observation station.
 - ▷ plotIntermediaire: Boolean if true plot transitional computation of the minimization.
- **manning.py**: Same than plotHydrograph.py for manning minimization.

- **obsT.py**: For t fixed, plot the evolution of h^{obs} with x .
 - ▷ **cheminDossierSortie**: Path of the ouput directory
- **obsxh.py**: For each station, plot the evolution of h^{obs} with time t
- **obsxw.py**: For each station, plot the evolution of w^{obs} with time t
- **obsx.py**: For each station, plot the evolution of h^{obs} and w^{obs} with time t
- **minCost.py** plot J and $|\nabla J|$ vs iterations.

11 Annexes

11.1 Example of the `input.txt` file

Listing 2: The complete `input.txt` file with default values.

```

=====
!   Input File for Shallow-Water Model
!   (configure your text editor with a 4 tabs space for better reading)
=====
&list_input

! =====
!   Mesh Type
! =====

    mesh_type      =   'basic',                ! 'basic' , 'dassflow' , 'gmsh'
    mesh_name       =   'mesh_name.geo'         ! Mesh File

! =====
!   Structured mesh parameters (in basic case)
! =====

    lx              =   1000.,                  ! Domain length in x horizontal direction
    ly              =   100.,                  ! Domain length in y vertical direction

    nx              =   100,                   ! Number of nodes in x horizontal direction
    ny              =   10,                   ! Number of nodes in y vertical direction

! =====
!   Boundary conditions
! =====

    bc_N            =   'wall',                ! Type of Boundary condition at Mesh North
    bc_S            =   'wall',                ! Type of Boundary condition at Mesh South
    bc_W            =   'discharg1',           ! Type of Boundary condition at Mesh West
    bc_E            =   'transm',              ! Type of Boundary condition at Mesh East

! =====
!   Simulation parameters
! =====

    ts              =   14400.,                ! Simulation Time
    dtw             =   14400.,                ! Time-Step to output Result Files
    dtp             =   60.,                  ! Time-Step to output Post Variables
    dta             =   60.,                  ! Data Assimilation Boundary Arrays Time Step
    verbose         =   0,                    ! Verbosity Level

! =====
!   Numerical parameters
! =====

    temp_scheme     =   'euler',              ! Choice of Temporal Scheme ( euler , rk2 )
    spatial_scheme  =   'first_b1',           ! Choice of Spatial Scheme ( first , muscl )

    adapt_dt        =   1,                    ! Choice of an Adaptative Time-Step (1) or not (0)
    dt              =   0.05,                 ! Fixed Simulation Time-Step if adapt_dt = 0
    cfl             =   0.8,                  ! CFL number in case of Adaptative Time Step
    heps            =   0.,                  ! Cut-off water depth to stabilize SW Numerical Schemes
    friction        =   1,                    ! Manning Source Term

    feedback_inflow =   1,
    coef_feedback   =   0.1,

    max_nt_for_direct = 10000000,

! =====
!   Physical parameters
! =====

    g               =   10.,                  ! Gravity constant

! =====
!   Output Results Files Switches
! =====

    w_tecplot       =   1,                    ! in Tecplot format
    w_vtk           =   1,                    ! in VTK format

```

```

w_exact      = 0,          ! Output Exact Solution (if provided in m_user_data.f90)
w_norm       = 0,          ! Output Linf, L1 and L2 relative error norms

w_obs        = 0,          ! Writing obs data relating to obs.txt file
use_obs       = 0,

! =====!
! Variational Data Assimilation Parameters
! =====!

c_manning     = 0,
c_bathy       = 0,
c_ic          = 0,
c_hydrograph  = 0,
c_ratcurve    = 0,

eps_manning   = 0.1,
eps_bathy     = 0.1,
eps_ic        = 0.1,
eps_hydrograph = 0.1,
eps_ratcurve  = 0.1,

max_nt_for_adjoint = 2500,

restart_min   = 0,
eps_min       = 1.d-4

/

```

11.2 Example of the m_user_data.f90 file

Listing 3: The `m_user_data.f90` file.

```

MODULE m_user_data

USE m_common
USE m_mesh
USE m_sw

implicit none

real(rp), parameter :: h0 = 10._rp
real(rp), parameter :: a = 3000._rp
real(rp), parameter :: b = 5._rp

CONTAINS

! =====!
! Bed elevation
! =====!

real(rp) FUNCTION bathy_user( x , y )

implicit none

real(rp), intent(in) :: x , y

real(rp) :: xx

xx = x - demi * lx

bathy_user = h0 * ( xx / a )**2

END FUNCTION bathy_user

! =====!
! Free surface elevation ( zs = h + bathy )
! =====!

real(rp) FUNCTION zs0_user( x , y )

implicit none

real(rp), intent(in) :: x , y

zs0_user = zs_exact( x , y , zero )

END FUNCTION zs0_user

! =====!
! Manning coefficient
! =====!

real(rp) FUNCTION manning_user( x , y )

implicit none

```

```

        real(rp), intent(in) :: x , y

        manning_user = 0.001_rp

    END FUNCTION manning_user

!=====
! Initial x velocity
!=====

    real(rp) FUNCTION u0_user( x , y )

        implicit none

        real(rp), intent(in) :: x , y

        u0_user = 0._rp

    END FUNCTION u0_user

!=====
! Initial y velocity
!=====

    real(rp) FUNCTION v0_user( x , y )

        implicit none

        real(rp), intent(in) :: x , y

        v0_user = 0._rp

    END FUNCTION v0_user

!=====
! Inflow boundary condition
!=====

    real(rp) FUNCTION inflow_user( t , x , y )

        implicit none

        real(rp), intent(in) :: t , x , y

        inflow_user = 0.0_rp

    END FUNCTION inflow_user

!=====
! Outflow boundary condition
!=====

    real(rp) FUNCTION outflow_user( t , x , y )

        implicit none

        real(rp), intent(in) :: t , x , y

        outflow_user = 0.0_rp

    END FUNCTION outflow_user

!=====
! Test Constants
!=====

    real(rp) FUNCTION om( a , h0 )

        implicit none

        real(rp), intent(in) :: a , h0

        om = sqrt( 8._rp * g * h0 / a**2 )

    END FUNCTION om

    real(rp) FUNCTION s( om , tau )

        implicit none

        real(rp), intent(in) :: om , tau

        s = demi * sqrt( om**2 - tau**2 )

    END FUNCTION s

!=====
! Exact water elevation
!=====

    real(rp) FUNCTION zs_exact( x , y , t )

```

```

implicit none

real(rp), intent(in) :: x , y , t

real(rp) :: tau , ss , xx

xx = x - demi * lx

tau = manning_user( xx , y )

ss = s( om( a , h0 ) , tau )

zs_exact = ( a**2 * b**2 * exp( - tau * t ) ) / ( 8._rp * g**2 * h0 )

zs_exact = zs_exact * ( - ss * tau * sin( two * ss * t ) + &
( d1p4 * tau**2 - ss**2 ) * cos( two * ss * t ) )

zs_exact = zs_exact - ( b**2 * exp( - tau * t ) ) / ( 4._rp * g )

zs_exact = zs_exact - exp( - demi * tau * t ) * &
b * ( ss * cos( ss * t ) + tau * demi * sin( ss * t ) ) * xx / g

zs_exact = zs_exact + h0

END FUNCTION zs_exact

!=====
! Exact x velocity
!=====

real(rp) FUNCTION u_exact( x , y , t )

implicit none

real(rp), intent(in) :: x , y , t

real(rp) :: tau , ss

tau = manning_user( x , y )

ss = s( om( a , h0 ) , tau )

u_exact = b * exp( - demi * tau * t ) * sin( ss * t )

END FUNCTION u_exact

!=====
! Exact y velocity
!=====

real(rp) FUNCTION v_exact( x , y , t )

implicit none

real(rp), intent(in) :: x , y , t

v_exact = 0._rp

END FUNCTION v_exact

END MODULE m_user_data

```

11.3 Example of bc.txt file

Listing 4: The **bc.txt** file.

```

!=====
! Number of boundary conditions
!=====
4
!=====
! List of boundary conditions
!=====
1  discharg1  file
2  discharg1  file
3  transm
4  transm

```

11.4 Example of hydrograph.txt file

Listing 5: The **hydrograph.txt** file.

```

!=====

```

```

! Number of hydrograph
! =====!
2
! =====!
! First hydrograph
! =====!
3
    0          10
    50000      12
    100000     20
! =====!
! Second hydrograph
! =====!
3
    0          50
    50000      80
    100000     100

```

11.5 Example of ratcurve.txt file

Listing 6: The **ratcurve.txt** file.

```

! =====!
! Number of ratcurve
! =====!
1
! =====!
! First rating curve
! =====!
2  0.0
0.  0.0
7.  10.0
! =====!
! Second rating curve
! =====!
2  0.0
0.  0.0
7.  10.0

```

11.6 Example of land_use.txt file

Listing 7: The **land_use.txt** file.

```

! =====!
! Number of Land Uses
! =====!
5
! =====!
! List of Land Uses
! =====!
1  0.1
2  0.3
3  0.2
4  0.1
5  0.1

```

11.7 Example of obs.txt file

Listing 8: The **obs.txt** file.

```

! =====!
! File Defining Stations and/or Sections to output Result Files at a prescribed temporal frequency
! =====!

stations      2

400.   50.   60.
600.   50.   60.

sections      2

0.     50.   1000.  50.   100   60.
500.   0.    500.   100.  100   60.

```

References

- [1] *Proceedings and results of the 2011 NTHMP model benchmarking workshop*, <http://nthmp.tsunami.gov/index.html>, 2012.
- [2] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM J. Sci. Comput.*, 25(6):2050–2065, June 2004.
- [3] E. Audusse and M.-O. Bristeau. A well-balanced positivity preserving "second-order" scheme for shallow water flows on unstructured meshes. *J. Comput. Phys.*, 206(1):311–333, June 2005.
- [4] T. Barth. Numerical methods for conservative laws on structured and unstructured meshes. Technical report, VKI Lecture Series, 2003.
- [5] T. Buffard and S. Clain. Monoslope and multislope muscl methods for unstructured meshes. *Journal of Computational Physics*, 229:3745–3776, 2010.
- [6] J. Burguete, P. Garc  a-Navarro, and J. Murillo. Friction term discretization and limitation to preserve stability and conservation in the 1d shallow-water model: Application to unsteady irrigation and river flow. *International Journal for Numerical Methods in Fluids*, 58(4):403–425, 2008.
- [7] M.J. Castro D  az, T. Chac  n Rebollo, E. D. Fern  ndez-Nieto, J. M. Gonz  lez-Vida, and C. Par  s. Well-balanced finite volume schemes for 2d non-homogeneous hyperbolic systems. application to the dam break of aznalc  llar. *Comput. Methods Appl. Mech. Engrg.*, 197:3932–3950, 2008.
- [8] T. Chac  n Rebollo, A. Dom  nguez Delgado, and E. D. Fern  ndez Nieto. Asymptotically balanced schemes for non-homogeneous hyperbolic systems - application to the shallow water equations. *Comptes Rendus Mathematique*, 338(1):85 – 90, 2004.
- [9] P. Ch  vrier and H. Galley. A van leer finite volume scheme for the euler equations on unstructured meshes. *ESAIM: Mathematical Modelling and Numerical Analysis*, 27(2):183–201, 1993.
- [10] C. Delestre, O. Lucas, F. Ksinant, P.A. Darboux, C. Laguerre, T.N.T. Vo, F. James, and S. Cordier. Swashes: a compilation of shallow water analytic solutions for hydraulic and environmental studies. 40 pages, April 2012.
- [11] A. I. Delis and I. K. Nikolos. A novel multidimensional solution reconstruction and edge-based limiting procedure for unstructured cell-centered finite volumes with application to shallow water dynamics. *International Journal for Numerical Methods in Fluids*, 71(5):584–633, 2013.
- [12] A.I. Delis, I.K. Nikolos, and M. Kazolea. Performance and comparison of cell-centered and node-centered unstructured finite volume discretizations for shallow water free surface flows. *Archives of Computational Methods in Engineering*, 18:57–118, 2011.
- [13] S. K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Math. Sbornik*, 47:271–306, 1959.
- [14] S. Gottlieb, C.-W. Shu, and E. Tadmor. Strong stability-preserving high-order time discretization methods. *SIAM Rev.*, 43:89–112, 2001.
- [15] F. Haider, J.-P. Croisille, and B. Courbet. Stability analysis of the cell centered finite-volume muscl method on unstructured grids. *Numer. Math.*, 113(4):555–600, September 2009.
- [16] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer, 2004.
- [17] J. Hou, Q. Liang, F. Simons, and R. Hinkelmann. A 2d well-balanced shallow flow model for unstructured grids with novel slope source term treatment. *Advances in Water Resources*, 52(0):107 – 131, 2013.
- [18] Q. Kesserwani, G. Liang. Locally limited and fully conserved rkdg2 shallow water solutions with wetting and drying. *Journal of Scientific Computing*, 50:120–144, 2012.
- [19] Q. Liang and F. Marche. Numerical resolution of well-balanced shallow water equations with complex source terms. *Advances in Water Resources*, 32(6):873–884, June 2009.
- [20] I. MacDonald, M.J. Baines, N.K. Nichols, and P.G. Samuels. Analytic benchmark solutions for open-channel flows. *Journal of Hydraulic Engineering*, 123(11):1041–1045, 1997.

- [21] I.K. Nikolos and A.I. Delis. An unstructured node-centered finite volume scheme for shallow water flows with wet/dry fronts over complex topography. *Computer Methods in Applied Mechanics and Engineering*, 198(47-48):3723–3750, 2009.
- [22] L. Pareschi and G. Russo. Implicit-explicit runge-kutta schemes and applications to hyperbolic systems with relaxation. *Journal of Scientific Computing*, 25:129–155, 2005.
- [23] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer Series in Computational Mathematics. Springer, 2008.
- [24] J. Sampson, A. Easton, and M. Singh. Moving boundary shallow water flow above parabolic bottom topography. *ANZIAM Journal*, 47, 2006.
- [25] L. Song, J. Zhou, J. Guo, Q. Zou, and Y. Liu. A robust well-balanced finite volume model for shallow water flows with wetting and drying over irregular terrain. *Advances in Water Resources*, 34(7):915–932, 2011.
- [26] C. Synolakis. *Standards, criteria, and procedures for NOAA evaluation of tsunami numerical models*. NOAA evaluation of tsunami numerical models, 2007.
- [27] C.E. Synolakis. The runup of solitary waves. *Journal of Fluid Mechanics*, 185:523–545, 1987.
- [28] W.C. Thacker. Some exact solutions to the nonlinear shallow-water wave equations. *J. Fluid Mech.*, 107:499–508, 1981.
- [29] E.F. Toro. *Shock-capturing methods for free-surface shallow flows*. John Wiley, 2001.
- [30] J.-P. Vila. Simplified godunov schemes for 2 x 2 systems of conservation laws. *SIAM J. Numer. Anal.*, 23(6):1173–1192, December 1986.