IMEX_SfloW2D, a Shallow Water numerical solver

Mattia de' Michieli Vitturi *, Giacomo Lari °

* Researcher, Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Pisa
° PhD student, Dipartimento di Matematica, Università di Pisa

Contents

1 Introduction

IMEX_SfloW2D is a FORTRAN90 code designed to model shallow water granular flows over digital elevation models (DEMs) of natural terrain, with the friction forces described by the Voellmy-Salm rheology. The system is described by an hyperbolic system of partial differential equations with relaxation and source terms.

The model is discretized in time with an explicit-implicit Runge-Kutta method where the hyperbolic part is solved explicitly and the other terms (friction) are treated implicitly to allow for larger time steps and to enforce the stopping condition. The finite volume solver for the hyperbolic part of the system is based on the Kurganov and Petrova 2007 semi-discrete central scheme and it is not tied on the specific eigenstructure of the model. The implicit part is solved with a Newton-Raphson method where the elements of the Jacobian of the nonlinear system are evaluated numerically with a complex step derivative technique. This automatic procedure allows for easy changes of the friction term.

The code can deal with different scenarios, but its first aim is to treat gravitational flows over topographies described as digital elevation models (DEMs) in the ESRI ascii format. The output files can be handled very well with gnuplot, in particular with the package it is provided a small script that create a video from the output data saved at different times. Moreover it is possible to save the solution as as ESRI ascii files, suitable for GIS softwares.

One of the reasons we wrote such a code is the possibility to have an open source code that can be freely downloaded by the users, which can simply use it without any intervention on the code or they can modify and widen their own version. IMEX_SfloW2D is written in FORTRAN90, which has the advantage to have a simple syntax and to be well known into the scientific community.

In this document we will give a short description of the mathematical model we assumed and the numerical approach we adopted. This work is mainly inspired by the paper of Kurganov and Petrova of the 2007 (reference [5] in bibliography). In the last part we will show some examples we tested with the code and that can be replied by the users.

2 Domain and Topography

In order to perform both ideal and realistic simulations, we need to include a terrain where the fluid can move over. For simulating a wide number of tests it is useful to obtain the topography from different sources. In particular the code allows to upload a terrain, that we indicate with B, directly from a DEM or defined by few points at the end of the input file. Moreover with small modification of the code it is possible to define the topography also through a function.

2.1 Creation of a 3D terrain

We consider a rectangular Cartesian coordinates system with vertical axis along the gravity force. As we will see, the program creates a rectangular and uniform grid in the domain. It is defined by the initial coordinates in x and y and by the number and the size of cells in which we want to split the domain (respectively `comp_cells_x`, `comp_cells_y` and `cell_size`). In the formulas we indicate the last one with Δx; note that from the Courant-Friedrichs-Lewy (CFL) condition the integration time step depends on the space resolution.

In the case of a terrain defined by a function it is simple to know the value of the topography in each point of the grid. In the case of a DEM we need to handle input topographies defined with a resolution finer or wider than that we chose. This is done in file *geometry_2d.f90*, where a linear interpolation of the height values is performed.

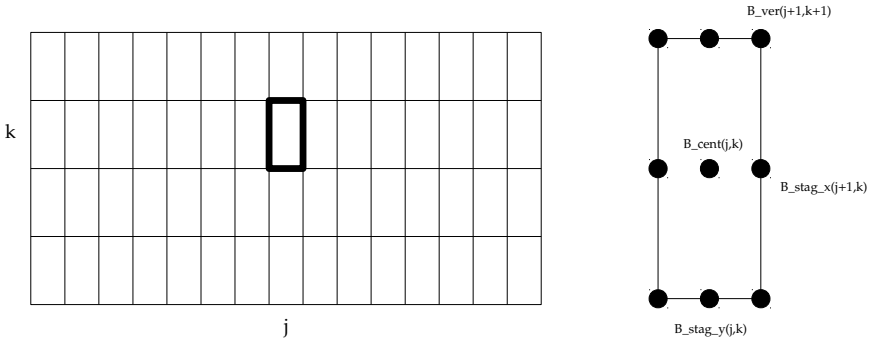

Figure 1: *Example of a 2D grid and a view of a single cell.*

Because we need to calculate fluxes, we have to know the height of the terrain not only in the centre of the cells, but also in the centre of the interfaces (see Figure 1). They are the sides of the cells and their number is `comp_interfaces_x`×`comp_cells_y` (vertical sides) plus `comp_cells_x`×`comp_interfaces_y` (horizontal sides).

Following [5], we define a piecewise linear approximation of the terrain. We assign the uploaded/calculated values of the height in the matrix `B_ver` ($B_{j,k}$ in the formulas, whose indexes (j,k) correspond to the vertexes of the cells). Departing from them, we build linearly the values in the interfaces centres (`B_stag_x(j+1,k)` and `B_stag_y(j,k+1)`) and in the centre of the cells (`B_cent(j,k)`, that is also the mean value on the cell), which are respectively

$$B_{j+\frac{1}{2},k} = \frac{1}{2}(B_{j,k} + B_{j+1,k}), \qquad B_{j,k+\frac{1}{2}} = \frac{1}{2}(B_{j,k} + B_{j,k+1}),$$
$$B_{j+\frac{1}{2},k+\frac{1}{2}} = \frac{1}{4}(B_{j,k} + B_{j+1,k} + B_{j,k+1} + B_{j+1,k+1})$$

Because we need also informations about the slope of the terrain, we have to calculate the derivatives of the topography both in x and y directions. We define `B_prime_x` and `B_prime_y`

$$B^x_{j,k} = \frac{B_{j+\frac{1}{2},k} - B_{j-\frac{1}{2},k}}{\Delta x}, \qquad B^y_{j,k} = \frac{B_{j,k+\frac{1}{2}} - B_{j,k-\frac{1}{2}}}{\Delta x}.$$

Moreover for the friction forces we will need to calculate the component of the gravity acceleration $\mathbf{g} = (0, 0, -g)$ along the the surface normal vector $\mathbf{N}$ in each cell (see Figure 2). The module of this component is:

$$g_n = g\frac{1}{\sqrt{1 + (B^x_{j,k})^2 + (B^y_{j,k})^2}}$$

3 Mathematical model

In this section we describe the equations that govern the motion of the fluid. We omit their derivation, that comes from the manipulation of the mass conservation law and the Newton's second dynamical equation.

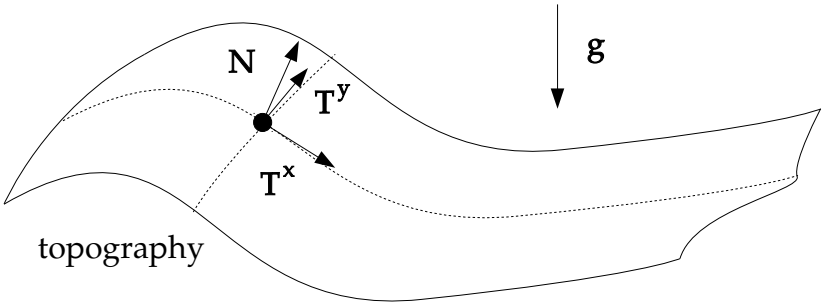

Figure 2: *The normal vector in a surface point.*

3.1 Shallow Water equations

The model we use for the fluid evolution is described by the Shallow Water equations. Such partial differential equations assume that the water can move just in 2D directions and that for each point of the plane we have a value (and only one) for its thickness.

The variables of the problem are (h, u, v), which are the height, the velocity along the x axis and the velocity along the y axis. In order to take into account the topography we consider as first variable $w = h + B$ and we denote (w, u, v) as first set of physical variables. If we want that the numerical scheme verifies the conservation laws, we have to express the equations in the conservative variables set $\mathbf{U} = (w, hu, hv)$.

For these variables the Shallow Water equations are:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = \mathbf{S}(\mathbf{U}) \tag{1}$$

where in this case the subscript means the derivation with respect to the correspondent variable, and the functions are defined

$$\begin{aligned} \mathbf{F}(\mathbf{U}) &= (hu, hu^2 + \frac{1}{2}gh^2, huv)^T \\ \mathbf{G}(\mathbf{U}) &= (hv, huv, hv^2 + \frac{1}{2}gh^2)^T \\ \mathbf{S}(\mathbf{U}) &= (0, gB^x h, gB^y h)^T \end{aligned} \tag{2}$$

The left hand side of the equation is the hyperbolic part, while $\mathbf{S}$ is the source term. It is important to note that in the numerical integration the first part will be solved explicitly, while the source term will be treated implicitly.

Here we neglect the vertical acceleration due to the motion of the fluid along the z axis, that would result in a quantity dw/dt to add to the gravity g in the function $\mathbf{F}$ and $\mathbf{G}$ (see [1]).

Yet there is a lack in the model if we want to deal with realistic flows: we need to add a rheology.

3.2 Voellmy-Salm rheology

If we want to obtain realistic simulations, we have to modify the classic Shallow Water equations, in order to include friction forces. In the code we implemented the Voellmy-Salm rheology, whose formulas are shown below. This model is commonly used for avalanches and debris flows, but it fits also for volcanic gravitational flows.

The terms we consider comprehend different effects and they appear only in the

momentum equations (see for example [2] and Titan2d user guide [3])

$$S_{VS}^{(2)}(\mathbf{U}) = \frac{u}{\sqrt{u^2+v^2}}\left(h\mu g_n + \frac{g}{\xi}(u^2+v^2)\right)$$
$$S_{VS}^{(3)}(\mathbf{U}) = \frac{v}{\sqrt{u^2+v^2}}\left(h\mu g_n + \frac{g}{\xi}(u^2+v^2)\right) \quad (3)$$

The parameters μ and ξ depend on the fluid and the terrain, then their values are different from one scenario to another. Generally they are empirical parameters that must be calibrated from the observations (for example the runout of the avalanche). The parameter μ describes the dry Coulomb friction, while ξ is the viscous resistance turbulent friction parameter. They can be assigned in the input file.

3.3 Other rheologies

In the code it is possible to select the rheology model we want to use, through the option `rheology_model`. So far in the code we have implemented the Voellmy-Salm rheology (set option to 1) and a simple plastic model with parameter τ (set option to 2). In this document we focus only on the first one; in the next future it is possible to add other rheologies, in order to have a wider set of models to choose.

4 Numerical model

The equations (1) are continuous in time t and in space (x, y). If we want to solve them numerically we need to perform a discretization of the domain. We use a finite volume method in order to treat the Shallow Water equations only in a finite number of cells and for a finite number of temporal instants. In this way we pass from a continuous differential system to an algebraic one, transforming the temporal derivatives, the divergences and the source part in a finite difference of terms.

The numerical approach is taken from [5]. Here we report briefly the main points of the scheme. For details we invite the reader to look at the article.

It is worthy to note that in the program all the variables are complex. The reason of this choice is the stability of the numerical derivative calculation when the increment is very small. This is a very important point when we want to deal with implicit terms in the differential equations and we have to calculate numerically the Jacobian in order to perform the Newton-Raphson method. For more details the reader can look at paper [6].

4.1 Central-Upwind scheme

The method used in [5] is a semi-discrete central-upwind scheme. We give a short explanation of all these characteristics: semi-discrete because it considers the mean value of the variables in each cell; central because the value in one cell depends on the value of the adjacent cells (in 2D we have 4 adjacent cells); upwind because the scheme follows the flux, calculating the local speeds of propagation in the interfaces. The details of this scheme can be found in [4].

In the article it is shown that this method brings to an ordinary differential equations system for the average of the variables $\overline{\mathbf{U}}_{j,k}$ in each cell:

$$\frac{d}{dt}\overline{\mathbf{U}}_{j,k}(t) = -\frac{\mathbf{H}^x_{j+\frac{1}{2},k}(t) - \mathbf{H}^x_{j-\frac{1}{2},k}(t)}{\Delta x} - \frac{\mathbf{H}^y_{j,k+\frac{1}{2}}(t) - \mathbf{H}^y_{j,k-\frac{1}{2}}(t)}{\Delta y} + \overline{\mathbf{S}}_{j,k}(t)$$

where $\mathbf{H}^x$ and $\mathbf{H}^y$ are the numerical fluxes, calculated from the value of the variables in the interfaces of the cells.

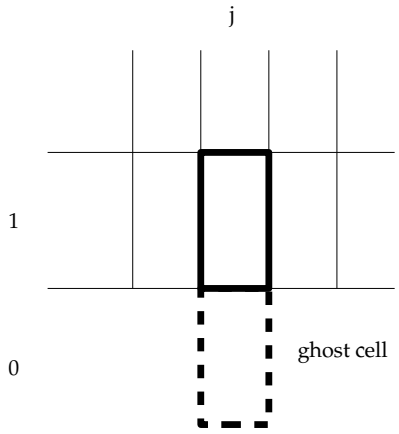

Figure 3: *Example of a ghost cell under the first row.*

From the average value of the variables we can operate a linear reconstruction in order to obtain the variables in each point of the cell. In particular in the centres of the interfaces (in the program they are indicated with `q_interfaceE` and so on, where E means East, W means West, N means North and S means South)

$$\mathbf{U}^E_{j,k} = \overline{\mathbf{U}}_{j,k} + \frac{\Delta x}{2}(\mathbf{U}_x)_{j,k}, \qquad \mathbf{U}^W_{j,k} = \overline{\mathbf{U}}_{j,k} - \frac{\Delta x}{2}(\mathbf{U}_x)_{j,k},$$
$$\mathbf{U}^N_{j,k} = \overline{\mathbf{U}}_{j,k} + \frac{\Delta x}{2}(\mathbf{U}_y)_{j,k}, \qquad \mathbf{U}^S_{j,k} = \overline{\mathbf{U}}_{j,k} - \frac{\Delta x}{2}(\mathbf{U}_y)_{j,k},$$

where $\mathbf{U}_x$ is the derivative of the variables with respect to x, and $\mathbf{U}_y$ with respect to y. These derivatives are calculated using an opportune limiter. In the input file it is possible to choose it: 0 means we use 0 as value of the derivative, 1 stands for the *MinMod* limiter, 2 for *Superbee* and 3 for *Van Leer*.

In order to calculate the numerical fluxes (`H_interfaceR` and so on in the code), we need to know the values of the variables at the two sides of the interfaces. For example for a right vertical side

$$\mathbf{H}^x_{j+\frac{1}{2},k} = \frac{a^+_{j+\frac{1}{2},k}\mathbf{F}(\mathbf{U}^E_{j,k}, B_{j+\frac{1}{2},k}) - a^-_{j+\frac{1}{2},k}\mathbf{F}(\mathbf{U}^W_{j,k}, B_{j+\frac{1}{2},k})}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}} + \frac{a^+_{j+\frac{1}{2},k}a^-_{j+\frac{1}{2},k}}{a^+_{j+\frac{1}{2},k} - a^-_{j+\frac{1}{2},k}}(\mathbf{U}^W_{j+1,k} - \mathbf{U}^E_{j,k}) \tag{4}$$

we need the value of the variables and the local speeds at the right and at the left of the interface. This calculation is performed in the subroutine `eval_flux_KT` in *solver_2d.f90*.

The local speeds of flux propagation $a^+_{j+\frac{1}{2},k}$ and $a^-_{j+\frac{1}{2},k}$ are

$$a^+_{j+\frac{1}{2},k} = \max\left(u^E_{j,k} + \sqrt{gh^E_{j,k}}, u^W_{j+1,k} + \sqrt{gh^W_{j+1,k}}, 0\right),$$
$$a^-_{j+\frac{1}{2},k} = \min\left(u^E_{j,k} - \sqrt{gh^E_{j,k}}, u^W_{j+1,k} - \sqrt{gh^W_{j+1,k}}, 0\right)$$

In the case of boundary cells we consider special cells, that we name ghost cells (see Figure 3). They are adjacent to the boundary in order to provide values at the empty side of the interfaces. These values can be given by boundary conditions or by adjacent cells.

Moreover in the code it is performed a correction for low waters (see formula (3.24) of [5]) and for preserving the positivity of h. This is done by formulas (3.20)-(3.23) taken from the same article and it can be found in the `reconstruction` subroutine in *solver_2d.f90*. It is important to note that in order to verify the theorem of positivity (see formula (3.32)) the coefficient of the CFL condition must be below 0.25.

4.2 Runge-Kutta method

In order to solve numerically the differential equations we use a Runge-Kutta method. Briefly this integrator solve part of the equations implicitly and the other part explicitly (*IMEX*). In particular the hyperbolic part is solved explicitly, while the relaxation and source term implicitly. The reader can find details in [7].

The user can choose the order of the method in the input file (option `n_RK`), paying attention that the implicit solver occurs when the parameter is greater than 1 (this is important when we consider the rheology (3.2)).

5 Examples

In the input file *IMEX_SfloW2D.inp* there are different options to be set in order to perform tests. An example is reported in Appendix B.

First of all we have to set the domain and the values of the variables on the boundary: `x_0`, `x_N`, `comp_cells_x`, `comp_cells_y` and `cell_size` define the rectangle and its resolution, while `h_bcW`, `u_bcW` and `v_bcW` (and similar) fix the boundary conditions.

If we want just to test the program with the classic Riemann problem we can assign the value T (TRUE) to `riemann_flag`. In this way the program reads the values of the variables at the right and left sides of the Riemann interface and it defines the initial conditions.

If we want to perform a more generic example, then we have to set the same option F (FALSE); in this way the program will read the initial conditions of the problem from the input file (options of `release`) or from other appropriate files (`restart` option must be TRUE), which are already present for the examples provided with the code.

In the input file we can set `topography_demfile`: if we assign F then the program will read the topography's height from the input file. In the other case the program will upload the terrain directly from a DEM file.

For the output the user can set a value `dt_output` after which all the times the code creates a file: its kind depends on the preferences of the user. If the user selects the flags `output_phys_flag` or `output_cons_flag` (supported by gnuplot), the program will create a file with different columns: the first two are the coordinates of the centres of the cell, while the others are the values of the chosen variables and topography in that point.

If `output_esri_flag` (supported by GIS) is selected, the program will provide a DEM file with the thickness of the fluid in the domain.

Here we show three examples: in the first two, a simple Riemann problem and an exercise taken from [5], we switch off the rheology from the code. In the third and last test we perform a simulation of a gravitational flow on a realistic topography.

5.1 Test 1: Riemann problem

In the Riemann problem we have a discontinuity of the water thickness between left and right side of the domain. In our example the terrain is flat, then it is easier to define it directly at the end of the input file, simply adding

'TOPOGRAPHY_PROFILE'

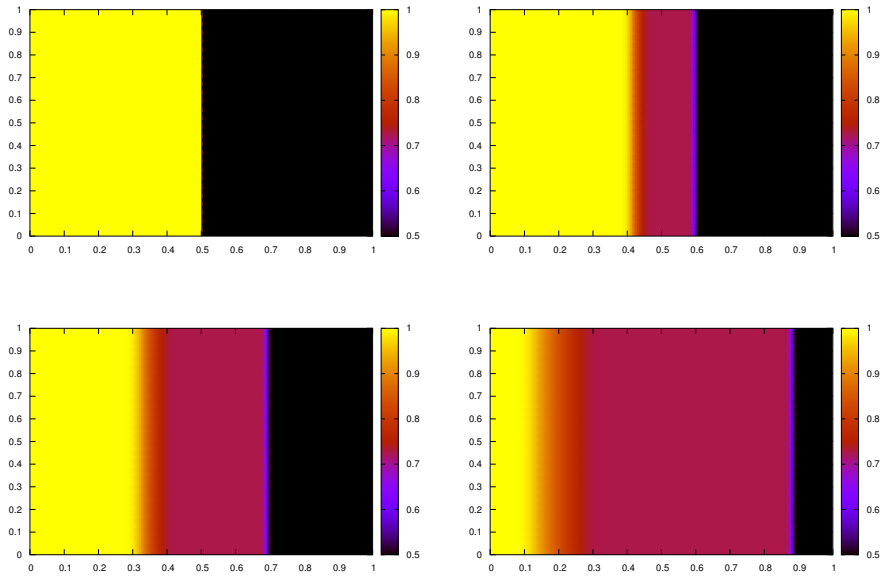

Figure 4: *Test 1: thickness h of the water at the initial time (top left), after 0.1 second (top right), after 0.2 seconds (bottom left) and after 0.4 seconds (bottom right).*

2
2
0.d0 0.d0 0.d0
0.d0 1.d0 0.d0
1.d0 0.d0 0.d0
1.d0 1.d0 0.d0

We set initial velocities to 0 and at the left of a certain coordinate `riemann_interface` of the domain we fix an height `hB_W` of 1.0 meters, and at the right `hB_E` of 0.5 meters.

In this case the flow will move in the West-East direction; the variable v does not have any influence to the result. The same test can be performed in the direction South-North and the result has the same qualitative behaviour. In this second case the variable u is 0 everywhere (apart from numerical fluctuations).

If the user wants to perform this test, he must set `rheology_flag` F and the parameter `grav` to 1.0.

In the output we can see the classic solution, where a wave front (at the right) and a rarefaction wave (at the left) come out from leaving the water fall (see Figure 4).

5.2 Test 2: Riverbed

We try to test the code with the last example of [5]. In this case it is simulated a river on a realistic discontinuous bottom, defined by the function B (formula (4.10) of the article):

$$B(x,y) = \frac{7}{32}e^{-8(x-0.3)^2-60(y-0.1)^2} - \frac{1}{8}e^{-30(x+0.1)^2-90(y+0.2)^2} + b(x,y),$$

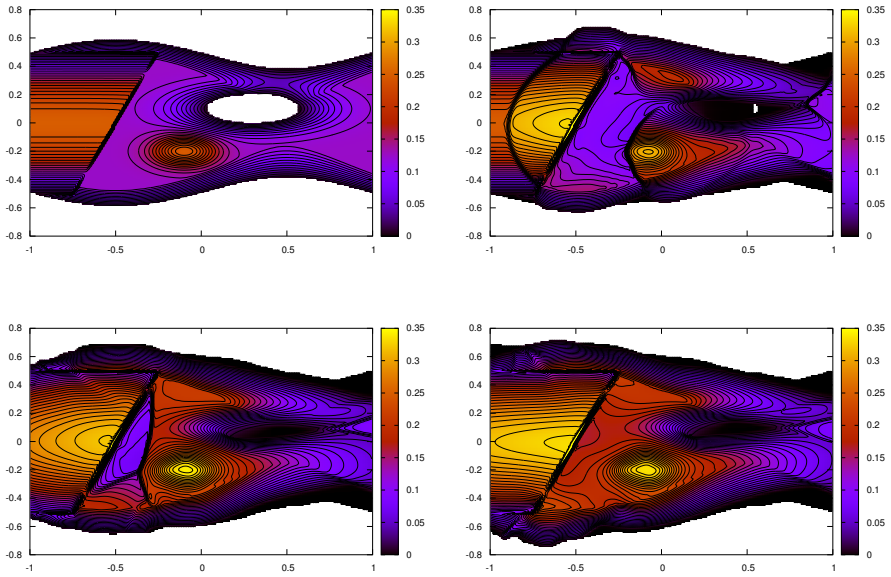

Figure 5: *Test 2: thickness h of the water at the initial time (top left), after 1 second (top right), after 2 seconds (bottom left) and after 4 seconds (bottom right).*

where

$$b(x,y) = \begin{cases} y^2, & |y| \le \frac{1}{2}, x \le \frac{y-1}{2} \\ y^2 + \frac{1}{10}\sin \pi x, & |y| > \frac{1}{2}, x \le \frac{y-1}{2} \\ \max\left(\frac{1}{8}, y^2 + \frac{1}{10}\sin \pi x\right), & x > \frac{y-1}{2} \end{cases}$$

The initial conditions are given instead in (4.11):

$$w_0(x,y) = \max\left(\frac{1}{4}, B(x,y)\right)$$

$$u_0(x,y) = \begin{cases} \frac{1}{2}, & |y| \le \frac{1}{2} \\ 0, & |y| > \frac{1}{2} \end{cases}$$

$$v_0(x,y) = 0$$

If the user wants to perform this test, he must set `rheology_flag` F and the parameter `grav` to 2.0.

We do not give here all the details about this example; however the reader can look at the paper for a description more accurate. We show just the output for the water thickness (see Figure 5) and we invite the reader to compare our plots with those of Kurganov and Petrova. The correspondence is very good and it is an important first step for the validation of the code. In fact in this test the program has to deal with a discontinuous topography and also with dry regions of the domain (white in the pictures).

5.3 Test 3: Volcanic avalanche

The third test has been performed taking a real topography and releasing a column of fluid with zero initial velocity (Figure 6 top left). With this example we want to

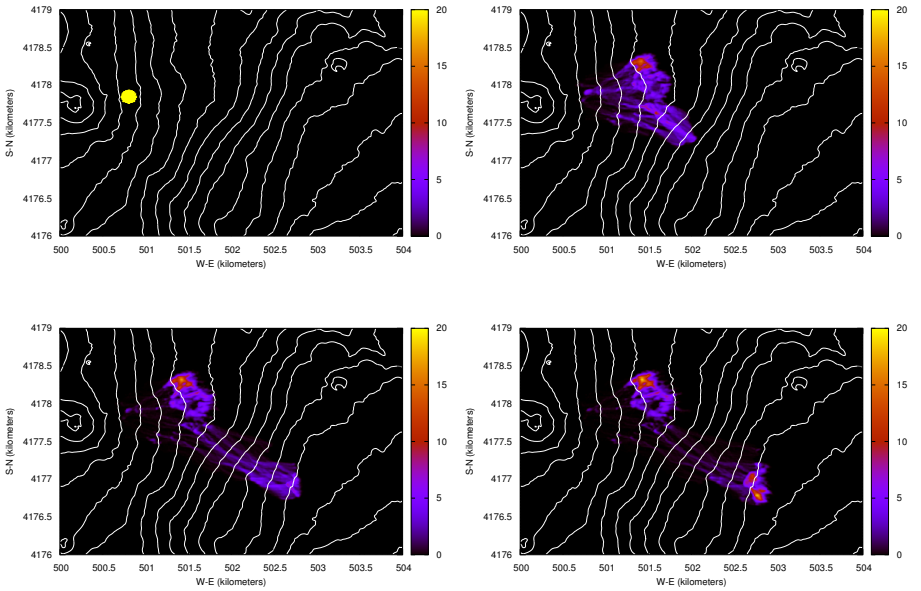

Figure 6: *Test 3: thickness h of the water at the initial time (top left), after 60 second (top right), after 120 seconds (bottom left) and after 180 seconds (bottom right). The white lines are the level curves of the topography.*

test the code in a realistic applicative scenario, where we are interested in simulating a volcanic fluid that moves down from the crater where it comes out.

Because we want to treat implicitly the rheology terms (3.2), we need to set `n_RK` equal to 2.

We took a DEM of the Etna Mountain and we placed a fluid column of volume $2\times10^6\ m^3$ (parameter `released_volume`) with centre in $(500800, 4177850)$ (parameters `x_release` and `y_release`). By default we consider that the pile has height equal to the radius. Note that the domain we define must be internal to the downloaded DEM; in this case we chose $[500000 : 504000] \times [4176000 : 4179000]$.

If the user wants to perform this test, he must set `rheology_flag` T and `rheology_model` to 1 (Voellmy-Salm model). We chose the parameters `grav`, `mu` and `xi` respectively 9.81, 0.3 and 500. In the appendix we reported the input file for this example. Moreover it is necessary to download a DEM of the volcano.

The output of the simulation show the evolution of the water thickness. Because of the high friction and because of the greater slope in the South-East direction, the fluid moves in that direction, while it stops quite early in the North-East direction. From the plots we reported we can see the areas where the avalanche cumulates (red and yellow in Figure 6).

The test can be considered very satisfying: the program manages to treat quick gravitational flows, dry areas, low waters and a real topography. It is worthy to note that actually the avalanche does not stop finally, but it continues to move downward very slowly. This is a problem reported for the most part of this kind of software, however the effect is very small as we can note comparing the bottom pictures of the same Figure.

Moreover the loss of mass during the simulation is nearly null apart from numerical fluctuations of the order of the computer precision. This can be considered another

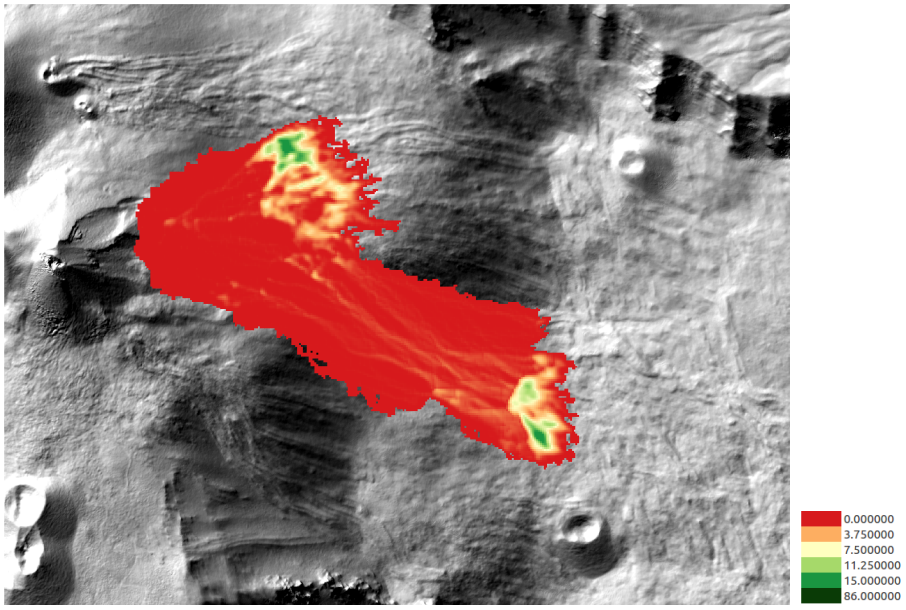

Figure 7: *Test 3: thickness h of the water after 180 seconds from the volume release.*

proof of the stability of the code.

In Figure 7 it is presented a view of the solution with GIS software.

The advantage of gnuplot is the possibility to plot easily mathematical informations about the terrain: in this case from Figure 6 we can see how the fluid stops where the distance of two level curves increases, in fact they are the areas where the slope decreases.

In the case of GIS, instead, the details of the terrain are very enhanced, showing the fluid between a realistic 3D representation of mountains and valleys.

Acknowledgements: This project has been performed within the scope of the contract about "Development and application of Shallow Water numerical model for the propagation of gravitational flows on 3D topographies" stipulated on 12/09/2016 in the context of INGV-DPC B2, Objective 4, Subtask D1.

A Download and compilation instructions

In this first appendix we describe step by step the procedure to download and compile IMEX_SfloW2D. The commands below are tested with Ubuntu OS.

The code can be found at the url *https://github.com/demichie/IMEX_SfloW2D* and it can be downloaded in .zip format and then extracted in the user's computer. Instead of this two actions it is possible to use the *git* command in the terminal:

```
git clone --recursive https://github.com/demichie/IMEX_SfloW2D.git
```

In this way a folder named *IMEX_SfloW2D* and containing the program is created in the user's *home*. Now it is very easy to compile the code: from the terminal enter in the new folder and then give these three commands:

```
./configure
```

```
make
```

```
make install
```

This will create the executable in the bin folder. The user can test the executable copying it in the *examples* folder and running it. As described in the document, the output can be easily plotted with gnuplot or GIS software. Pay attention that for the more interesting tests it is necessary to download DEM files in the *examples* folder.

B Input file

```
&RUN_PARAMETERS
 RUN_NAME="EtnaTest",
 RESTART=F,
 TOPOGRAPHY_DEMFILE=T,
 T_START=  0.0000000000000000      ,
 T_END=  200.00000000000000001E-000,
 DT_OUTPUT=  10.000000000000000E-000,
 OUTPUT_ESRI_FLAG = .TRUE. ,
 OUTPUT_CONS_FLAG = .FALSE. ,
 OUTPUT_PHYS_FLAG = .TRUE. ,
 VERBOSE_LEVEL=           0,
 /
&NEWRUN_PARAMETERS
 X0=  500000.0000000000000000      ,
 Y0=  4176000.0000000000000000      ,
 COMP_CELLS_X=         400,
 COMP_CELLS_Y=         300,
 CELL_SIZE = 10.0     ,
 RHEOLOGY_FLAG = T ,
 RHEOLOGY_MODEL = 1 ,
 RIEMANN_FLAG= F ,
 /
&INITIAL_CONDITIONS
 RELEASED_VOLUME = 2.0D+6 ,
 X_RELEASE = 500800.0 ,
 Y_RELEASE = 4177850.0 ,
 VELOCITY_MOD_RELEASE = 0.0 ,
 VELOCITY_ANG_RELEASE = 0.0 ,
 T_INIT = 1.0 ,
 T_AMBIENT = 1.0 ,
 /
&WEST_BOUNDARY_CONDITIONS
 HB_BCW%FLAG=           1,
 HB_BCW%VALUE=  0.0000000000000000      ,
 U_BCW%FLAG=           1,
 U_BCW%VALUE=  0.0000000000000000      ,
 V_BCW%FLAG=           1,
 V_BCW%VALUE=  0.0000000000000000      ,
 /
&EAST_BOUNDARY_CONDITIONS
 HB_BCE%FLAG=           1,
 HB_BCE%VALUE=  0.0000000000000000      ,
 U_BCE%FLAG=           1,
 U_BCE%VALUE=  0.0000000000000000      ,
 V_BCE%FLAG=           1,
 V_BCE%VALUE=  0.0000000000000000      ,
 /
&SOUTH_BOUNDARY_CONDITIONS
 HB_BCS%FLAG=           1,
 HB_BCS%VALUE=  0.0000000000000000      ,
 U_BCS%FLAG=           1,
 U_BCS%VALUE=  0.0000000000000000      ,
 V_BCS%FLAG=           1,
 V_BCS%VALUE=  0.0000000000000000      ,
 /
&NORTH_BOUNDARY_CONDITIONS
 HB_BCN%FLAG=           1,
 HB_BCN%VALUE=  0.0000000000000000      ,
 U_BCN%FLAG=           1,
 U_BCN%VALUE=  0.0000000000000000      ,
```

```
 V_BCN%FLAG=              1,
 V_BCN%VALUE=   0.0000000000000000      ,
 /
&NUMERIC_PARAMETERS
 MAX_DT=   1.0000000000000000E-000,
 SOLVER_SCHEME="KT",
 CFL= 0.2400000000000000000      ,
 LIMITER= 3*1              ,
 THETA=   1.300000000000000      ,
 RECONSTR_COEFF=   1.0000000000000000      ,
 N_RK=              2,
 /
&SOURCE_PARAMETERS
 GRAV= 9.81       ,
 MU= 0.3      ,
 XI= 500.0       ,
 /
```

References

[1] Denlinger R.P., Iverson R.M. (2004) *Granular avalanches across irregular three-dimensional terrain: 1. Theory and computation.* Journal of Geophysical Research 109, F01014

[2] Fischer J.T., Kowalski J., Pudasaini S.P. (2012) *Topographic curvature effects in applied avalanche modeling.* Cold Regions Science and Technology 74-75, 21-30

[3] Geophysical Mass Flow Group (2016) *Titan2D - Geophysical Mass-Flow Simulation Software. User Guide, version 4.0.0.* http://www.gmfg.buffalo.edu/software.php

[4] Kurganov A., Noelle S., Petrova G. (2001) *Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton-Jacobi equations.* SIAM J. Sci. Comput., Vol. 23, No. 3, 707–740

[5] Kurganov A., Petrova G. (2007) *A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant System.* Commun. Math. Sci. vol. 5, no,1, 133-160

[6] Martins J.R.R.A., Sturdza P., Alonso J.J. (2003) *The complex-step derivative approximation.* ACM Transactions on Mathematical Softwar, Vol. 29, Issue 3, 245-262

[7] Pareschi L., Russo G. (2005) *Implicit–Explicit Runge–Kutta Schemes and Applications to Hyperbolic Systems with Relaxation.* Journal of Scientific Computing, Vol. 25, Nos. 1/2, 129-155