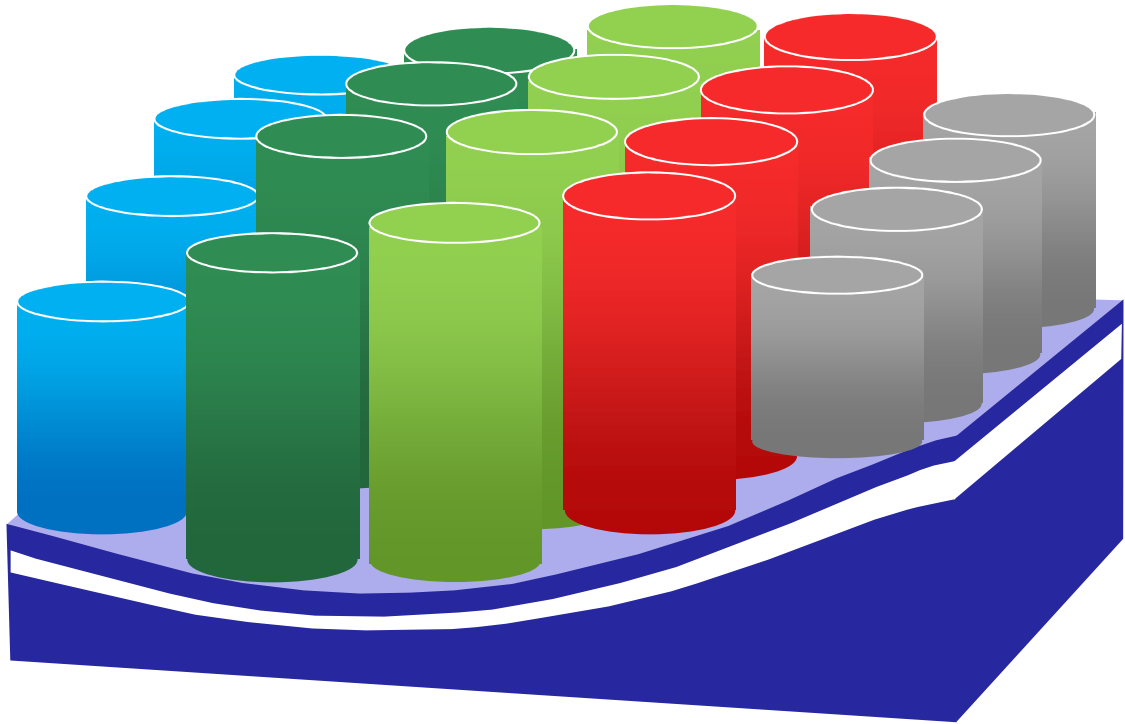


USER GUIDE FOR THE SWE-SPHysics CODE



SWE-SPHysics

March 2013

R. Vacondio (renato.vacondio@unipr.it)
B.D. Rogers (benedict.rogers@manchester.ac.uk)
P.K. Stansby (p.k.stansby@manchester.ac.uk)
P.Mignosa (mignosa@unipr.it)

Acknowledgements

The development and application of *SWE-SPHysics* were partially supported by:

- Flood Risk Management Research Consortium (FRMRC) Phase 2, EPSRC Grant F020511
- Research Councils UK (RCUK) Research Fellowship

Abstract

This guide documents the computer program *SWE-SPHysics* based on Smoothed Particle Hydrodynamics theory. The documentation provides a brief description of the governing equations and the different numerical schemes used to solve them. FORTRAN code is provided for one and two-dimensional versions of the model. Post-processing tools for MATLAB and PARAVIEW visualization are also provided. Finally, several working examples are documented to enable the user to test the program and verify that it is installed correctly.

Contents

1. INTRODUCTION	6
1.1 Introduction to Shallow Water Equations (SWEs) and the need for a meshless SPH solver	6
1.2 Equations of Motion	6
1.3 Solving the Shallow Water Equations using SPH	7
1.4 The SPH method and the weighting function (smoothing kernel)	8
1.5 Future Developments	8
2. IMPLEMENTATION	9
2.1 Overall Implementation	9
2.2 Computational efficiency: linked list	9
2.2.1 Changes to accommodate variable smoothing length	9
2.3 Restart runs & checkpointing (repetitive restarts)	9
3. USER'S MANUAL	11
3.1. Installation	12
3.2. Program outline	12
3.2.1. SWE-SPHysicsgen	12
3.2.1.1. Creating compiling options	13
3.2.1.2. Input files	13
3.2.1.3. Output files	13
3.2.1.4. Subroutines	15
3.2.2. SWE-SPHysics	16
3.2.2.1. Input files	16
3.2.2.2. Output files	16
3.2.2.3. Subroutines	17
4. TEST CASES	21
4.1. Running the model	21
4.1.1. Compiling and executing on Linux and Mac OS	21
4.1.2. Compiling and executing on Windows	23
4.2 1-D Test Cases	
4.2.1 Test case 1: 1-D Wet-bed Dam break	25
4.2.2 Test case 2: 1-D Dry-bed Dam break	26
4.2.3 Test case 3: 1-D Flow over hump with Inflow-Outflow Boundaries Conditions	28
4.3 2-D Test Cases	
4.3.1 Test case 1: 2-D Thacker Basin – rotating surface	30
4.3.2 Test case 2: Tsunami Wave with Inflow-Outflow Boundary Conditions	33
4.3.3 Test case 3: 2-D Dry-Bed Dam Break with <i>Particle Splitting</i>	35
4.3.4 Test case 4: 2-D CADAM Case with 45° Channel	39

5. HOW TO CHANGE <i>SWE-SPHysics</i> FOR YOUR APPLICATION	43
5.1 Introduction	43
5.2 Code Structure	44
5.3 Main variables	45
6. VISUALIZATION	46
6.1 Using Matlab	46
6.2 Using Paraview	47
7. REFERENCES	48
8. PUBLICATIONS USING <i>SWE-SPHysics</i> CODE	50

1. INTRODUCTION

1.1 Introduction to Shallow Water Equations (SWEs) and the need for a meshless SPH solver

The two-dimensional shallow-water equations (SWEs) are widely used to approximate flows for a wide range of rapidly (and slowly) varying free-surface flows, such as dam breaks, river flooding, and tidal flows including storm surge and wave overtopping causing inundation in estuaries and coastal regions. Grid-based solvers are now widely available. Although accurate and robust wetting and drying routines have been developed, grid-based solvers are limited in simulating multi-phase effects, most importantly flows with rapid distortion in flood modelling. Particle methods are quite flexible in this respect and are also naturally adaptive for modelling complex domains. Here, the SPHysics numerical scheme, originally developed to solve Navier-Stokes Equations has been extended to shallow water equations.

1.2 Equations of Motion

The shallow water equations (SWEs) represent the depth-integrated equations of mass and momentum. In order to be solved using SPH, the conservation equations need to be written in Lagrangian form:

<p>Eulerian Form</p> $\frac{\partial d}{\partial t} = \nabla \cdot (\mathbf{v}d)$	<p>Langrangian Form</p> $\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v} \quad (1.1)$
$\frac{\partial(\mathbf{v}d)}{\partial t} + \nabla \cdot (\mathbf{v}d) = -gd\nabla(d+b) + gd\mathbf{S}_f$	$\frac{d\mathbf{v}}{dt} = -\frac{\rho}{\rho_w} \nabla \rho + g(\nabla b + \mathbf{S}_f) \quad (1.2)$

where d is depth as plotted in Figure 1.1, \mathbf{v} is depth-averaged velocity, t is time, b is the bottom elevation, g is the acceleration due to gravity and \mathbf{S}_f is the bed friction source term. The SWEs are formally identical to the Euler equations if we re-define the density ρ as the mass of fluid per unit of area in a 2-D domain; with this definition of ρ we have $\rho = \rho_w d$, where ρ_w is the constant (conventional) density. The density ρ_i of a particle i can vary considerably during a simulation; therefore an SPH scheme with variable smoothing length h in time and space is used to keep the number of neighbour particles roughly constant during the processes of water inundation and retreat.

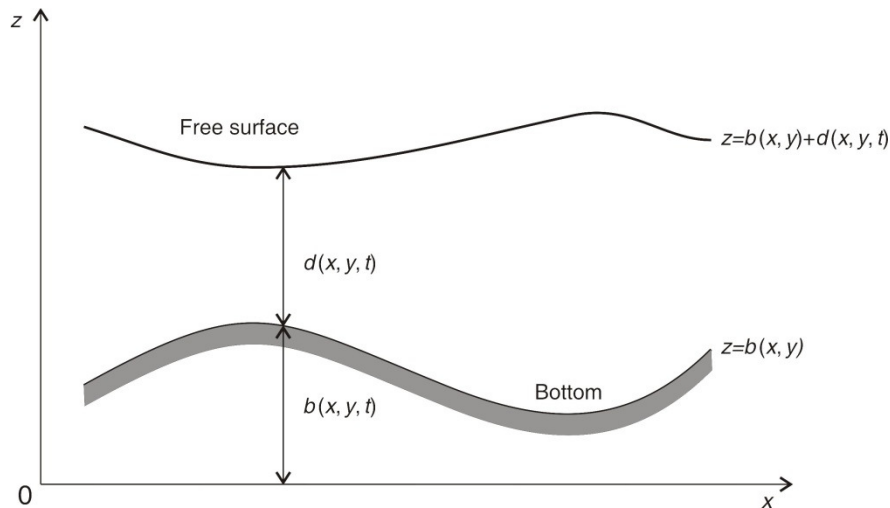


Figure 1.1: Flow with a free surface under the effect of gravity.

1.3 Solving the Shallow Water Equations using SPH

The methodology to solve the SWEs using SPH is described in the following papers, and we encourage you to read and refer to these publications:

Feature	Publication
Inlet-outlet (inflow-outflow) boundaries	Vacondio R, Rogers B D, Stansby P K, Mignosa P. 2012. "SPH modeling of shallow flow with open boundaries for practical flood simulation". <i>Journal of Hydraulic Engineering</i> . DOI: 10.1061/(ASCE)HY.1943-7900.0000543
Solid (no-flow) boundaries	Vacondio R, Rogers B D, Stansby P K. 2011. "Smoothed Particle Hydrodynamics: approximate zero-consistent 2-D boundary conditions and still shallow water tests". <i>Int. Journal for Numerical Methods in Fluids</i> . DOI: 10.1002/fld.2559
Particle splitting and variable h	Vacondio R, Rogers B D, Stansby P K. 2011. "Accurate particle splitting for SPH in shallow water with shock capturing". <i>Int. Journal for Numerical Methods in Fluids</i> . DOI: 10.1002/fld.2646
Water depth (or density evaluation)	Vacondio R, Rogers B D, Stansby P K. 2011. "Accurate particle splitting for SPH in shallow water with shock capturing". <i>Int. Journal for Numerical Methods in Fluids</i> . DOI: 10.1002/fld.2646
Bed Topography Representation	Vacondio R, Rogers B D, Stansby P K, Mignosa P. 2012. "SPH modeling of shallow flow with open boundaries for practical flood simulation". <i>Journal of Hydraulic Engineering</i> . DOI: 10.1061/(ASCE)HY.1943-7900.0000543
Viscosity & Stabilisation terms	Vacondio R, Rogers B D, Stansby P K. 2011. "Accurate particle splitting for SPH in shallow water with shock capturing". <i>Int. Journal for Numerical Methods in Fluids</i> . DOI: 10.1002/fld.2646
Time stepping	Vacondio R, Rogers B D, Stansby P K. 2011. "Smoothed Particle Hydrodynamics: approximate zero-consistent 2-D boundary conditions and still shallow water tests". <i>Int. Journal for Numerical Methods in Fluids</i> . DOI: 10.1002/fld.2559

1.4 The SPH method and the weighting function (smoothing kernel)

The main features of the SPH method, which is based on integral interpolants, are described in detail in the following papers (Monaghan, 1982; Monaghan, 1992; Benz,

1990; Liu, 2003; Monaghan, 2005). Herein we will only refer to the representation of the constitutive equations in SPH notation. In SPH, the fundamental principle is to approximate any function $A(\mathbf{r})$ by

$$\langle A(\mathbf{r}) \rangle = \int A(\mathbf{r}') W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' \quad (1.3)$$

where h is called the smoothing length, $W(\mathbf{r} - \mathbf{r}', h)$ is the weighting function or kernel, and $\langle \rangle$ denotes approximation. This approximation, in discrete notation, leads to the following approximation of the function at a particle (interpolation point) i ,

$$\langle A(\mathbf{r}) \rangle = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h) \quad (1.4)$$

where the summation is over all the particles within the region of compact support of the kernel function. The mass and density are denoted by m_j and ρ_j respectively and $W_{ij} = W(\mathbf{r}_i - \mathbf{r}_j, h)$ is the weight function or kernel.

For a more recent review of SPH applied to the Navier-Stokes equations, please see Gomez-Gesteira *et al.* (2010).

1.5 Future Developments

As with all developments in the *SPHysics* project, official code updates are only released after validation and journal papers are published.

Future developments for *SWE-SPHysics* includes:

1. Particle coalescing (merging) for efficient simulations (Vacondio et al. 2011b)
2. Correction for step changes in the bed (Vacondio et al. 2013)

2. IMPLEMENTATION

2.1 Overall Implementation

SWE-SPHysics was built from the original FORTRAN *SPHysics* Navier-Stokes solver, and hence is very similar in structure and operation. For information on the main solver, users are encouraged to find information in two articles explaining the theory and implementation, Gomez-Gesteira *et al.* (2012a,b).

2.2 Computational efficiency: link list.

Similar to the fortran *SPHysics* code for the Navier-Stokes equations, the computational domain is divided in square cells of side $2h$ (see Figure 1 of Gomez-Gesteira *et al.* 2012b). Here the only difference is that the smoothing length, h , can vary. Hence, changes are required to the code.

2.2.1 Changes to accommodate variable smoothing length

To compute both the water depth and the smoothing length for each i -th particle, a Newton Rapson iterative procedure is implemented in the subroutine `ac_dw`. The grid used in the linked-list procedure is equal to $2h_{max}$, where h_{max} is the maximum smoothing length in the domain. When h_{max} is updated the grid needs to be re-defined.

Two link lists are considered in *SWE-SPHysics*. The first one tracks the open and closed boundary particles and it is partially upgraded every time step. This is due to the fact that the only boundary particles that change their position in time are the ones that describe moving objects. The second link list corresponds to fluid particles and is completely updated every time step.

An additional link list is built at the beginning of the simulation for the bottom particles, and this is not updated during the simulation.

2.3. Restart runs & checkpointing (repetitive restarts)

Restarting previous (unfinished) runs is controlled using the `RESTART` parameter. If the code is being run on computer clusters, there are sometimes limits as to how long a particular job can run, e.g. 24 hours. This can be specified when first launching the *SPHysics* code by setting the `i_restartRun` parameter in the Case files.

`i_restartRun > 1` is used for Checkpointing = repetitive restarting of code (for clusters)

so that:

`i_restartRun = 0` : Start new run, once only

`i_restartRun = 1` : reStart old run, once only

3. USER'S MANUAL

3.1. Installation

Two versions of *SWE-SPHysics* are available in this release:

- *SWE-SPHysics_1D*. The computational domain is 1-D, where x corresponds to the horizontal direction and z to the vertical direction.
- *SWE-SPHysics_2D*. The computational domain is 2-D, where x and y are the horizontal directions and z the vertical direction.

SWE-SPHysics is distributed in a compressed file (gz or zip). The directory tree shown in Figure 3.1 can be observed after uncompressing the package

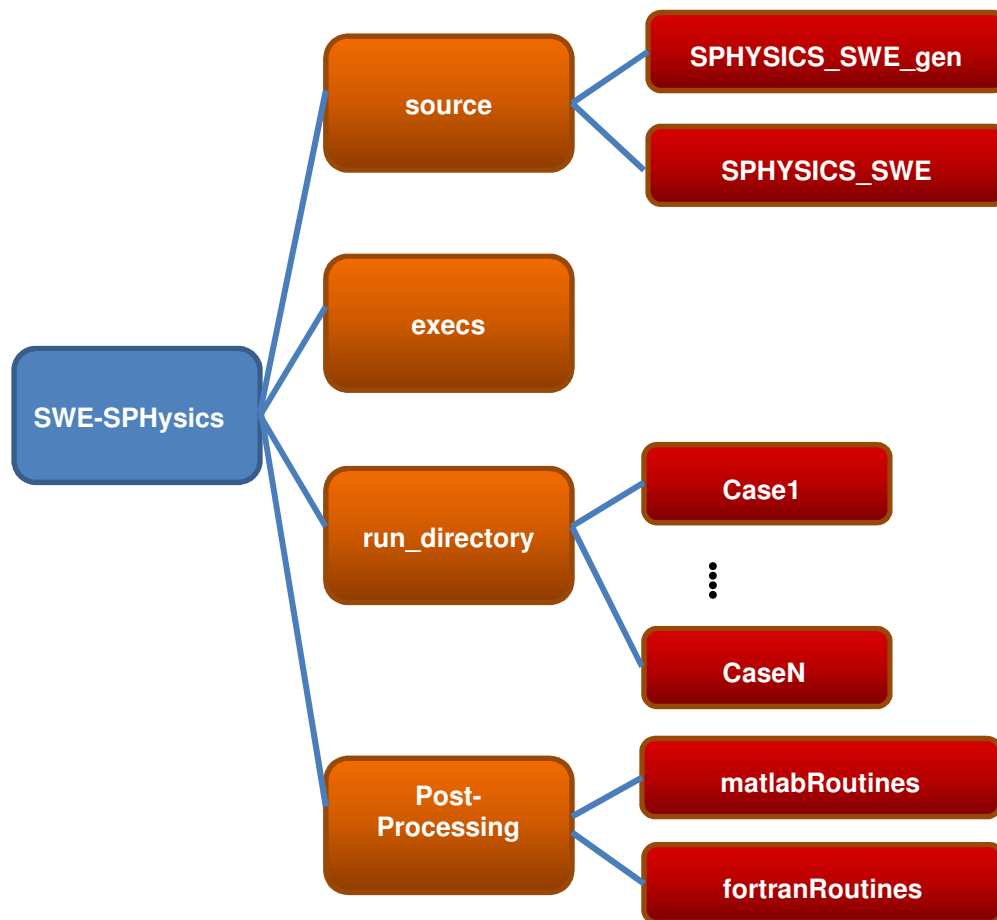


Figure 3.1 *SWE-SPHysics* directory (folder) structure

The following directories can be observed both in 1-D and in 2-D.

source contains the FORTRAN codes. This directory contains two subdirectories:

SPHysics_SWE_gen: contains the FORTRAN codes to create the initial conditions of the run.

SPHysics_SWE: contains the FORTRAN source codes of SPH.

execs contains all executable codes.

run_directory is the directory created to run the model. The different subdirectories *Case1*, ..., *CaseN* placed in this directory correspond to the different working cases to be created by the user. Input and output files are written in these directories
Post-Processing this directory contains codes to visualize results.

3.2. Program Outline

Both the 1-D and 2-D version consist of two general programs:

- (i) Geometry Generation using SPHYSICS_SWE_gen
- (ii) SPH simulation using SPHYSICS_SWE

These are run separately and in the following order.

1-D Code:

SPHYSICS_SWE_gen_1D: Creates the initial conditions and files for a given case.

SPHYSICS_SWE_1D: Runs the selected case with the initial conditions created by SPHYSICS_SWE_gen_1D code.

2-D Code:

SPHYSICS_SWE_gen_2D: Creates the initial conditions and files for a given case.

SPHYSICS_SWE_2D: Runs the selected case with the initial conditions created by SPHYSICS_SWE_gen_2D code.

In general, 1-D or 2-D appended to the source file name means that the source is suited for 1-D or 2-D calculations.

In the remainder of this document, SPHYSICS_SWE_gen and SPHYSICS_SWE, when used, refer to both the aforementioned 1-D and 2-D programs for convenience. For example, SPHYSICS_SWE_gen will refer to both SPHYSICS_SWE_gen_1D and SPHYSICS_SWE_gen_2D.

3.2.1. SPHYSICS_SWE_gen

All subroutines are included in two source files (SPHYSICS_SWE_gen_1D.f or SPHYSICS_SWE_gen_2D.f), depending on the nature one or two- dimensional of the calculation. Each source uses global variables where most of the variables are stored: global_1D.f/global_2D.f. Both versions (1-D and 2-D) can be compiled by the user with any FORTRAN compiler and the resulting executable file is placed in subdirectory \execs.

`SPHYSICS_SWE_gen` plays a dual role:

- (i) Creating the `MAKEFILE` to compile `SPHysics`; and
- (ii) Creating the output files that will form the input files to be read by `SWE-SPHysics`. These files contain information about the geometry of the domain, the distribution of particles and the different running options.

In Windows for example, `SPHYSICS_SWE_gen.exe` can be executed using one of the following two commands,

1. `SPHYSICS_SWE_gen.exe <input_file >output_file`

input_file is the general name (any name can be used) of the file containing the running options. Different examples of *input_file* will be shown in next section.

output_file is the general name (any name can be used) of the file containing general information about the run. This file is never read by the rest of the code and only serves to provide information to the user.

2. `SPHYSICS_SWE_gen.exe`

In this case, data about the run must then be provided by the user by means of the keyboard and the information about the run appears on the screen. This option can be used by beginners to get familiarized with the different options.

3.2.1.1. Creating compiling options

The compilation of `SWE-SPHysics` code depends on the nature of the problem under consideration and on the particular features of the run. Thus, the user can chose the options that are better suited to any particular problem and only those options will be included in the executable versions of `SWE-SPHysics`. This protocol speeds up calculations since the model is not forced to make time consuming logical decisions.

In both 1-D and 2-D the following compiling options can be considered:

- i) stabilization term 1 - artificial viscosity, 2 - Lax Friedrichs flux, 3 - two shocks Riemann solver
- ii) 0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
- iii) Choice of compilers: (1=gfortran; 2=ifort; 3=win_ifort; 4=Silverfrost FTN95).

3.2.1.2. Input files

There are potentially three different types of input file:

- (i) Case input files (mandatory)
- (ii) Bed/Bottom profiles
- (iii) Open Boundary Specifications

Different examples of input files(referred to herein as Case files, e.g. `Case1.txt`) will be shown in Section 4, where several test cases will be described.

3.2.1.3. Output files

As mentioned above, different output files are created by SPHYICS_SWE_gen. These files can be used either by the SPHYICS_SWE *executable* as input files or by MATLAB codes to visualize results (different MATLAB codes are provided in */Post-processing* subdirectory).

SPHysics.mak

Compiling file created by the executable SPHYICS_SWE_gen. It depends on the running options defined by *input_file* and can be used for Intel Fortran, Silverfrost FTN95, ifort and gfortran although it can be adapted to other compilers.

INDAT

Created by SPHYICS_SWE_gen :

Read by SPHYICS_SWE code at GETDATA (see Subsection 3.2.2.3).

UNIT=11

The file contains the following variables:

rho0	
viscos_val	
dw_min_fric	
coef	
vlx	medium extent in x direction
vly	medium extent in y direction
np	Number of particles at start of simulation
np_b	Number of bed particles
npv	number of virtual particles
i_openbc	
distmin	
tol	
ivar_dt	Activate (10yes,0=no) variable timesteps
dt	Initial timestep
CFL	Courant number (0.1-0.5)
tmax	Length of simulation
out	Output interval
trec_ini	
i_restartRun	
hsm_b_max	smoothing length for the bottom particles
n0	max number of particles in one cell
idebug	
iMUSCL	Activate (10yes,0=no) MUSCL reconstruction
i_dw_iter	

i_max_iter	
!refining parameters	
area_lim*ref_lim	limits of area to refine
ref_p	position of refined particles (coeff)
ref_h	smoothing length of refined particles
dw_min_ref	minimum water depth to activate refining
xmin_ref	
ymin_ref	
dxx_ref	
dyy_ref	
ncx_ref	
ncy_ref	
!parameters to export on regular grid	
dx_grd	
dy_grd	

IPART

Created by SPHYSICS_SWE_gen.

Read by SPHYSICS_SWEcode at GETDATA (see subsection 3.2.2.3).

UNIT=13

The file contains the following variables recorded at *time=0*:

In 1-D version

```

xp(1) up(1) rhop(1) dw(1) areap(1)
xp(2) up(2) dw(2) areap (2)
.....
xp(np) up(np) dw(np) areap (np)

```

In 2-D version

```

xp(1) yp(1) up(1) vp(1) dw(1) areap(1) h_var(1) iflag(1)
xp(2) yp(2) up(2) vp(2) dw(2) areap(2) h_var(2) iflag(2)
.....
xp(np) yp(np) up(np) vp(np) dw(np) areap(np) h_var(np) iflag(np)

```

Description:

xp(i) Position in *x* direction of particle *i*.
yp(i) Position in *y* direction of particle *i*.
up(i) Velocity in *x* direction of particle *i*.
vp(i) Velocity in *y* direction of particle *i*.
dw(i) Free-surface elevation of particle *i*.
areap(i) Area of particle *i*.
h_var(i) Smoothing Length (*h*) of particle *i*.
iflag(i) Flag of particle *i*.

3.2.1.4. Subroutines

All subroutines in SPHYSICS_SWE_gen are inside a single source file SPHysics_SWE_gen_1D.f or SPHysics_SWE_gen_2D.f

SPHysicsgen Main program.

Subroutine Name	Purpose
surface_define	Defining initial condition
slopingBed	Defining bottom elevation
parabolicBed	Defining bottom elevation
read_bedProfile	Defining bottom elevation
tocompile_unixStyle	Create make file
tocompile_cvf	Create make file

3.2.2. SPHYSICS_SWE

The SPHYSICS_SWE executable depends on the compiling option determined by SPHYSICS_SWE_gen

3.2.2.1. Input files

The input files correspond to the output files generated by SPHYSICS_SWE_gen and are described in Section 3.2.1.3.

3.2.2.2. Output files

PART_klmn

Created by SPHYSICS_SWE at POUTE_1D.f or POUTE_2D.f with a periodicity in seconds fixed by the *input_file* used to run SPHYSICS_SWE_gen.

UNIT=23

The structure of PART_klmn is the same as that of IPART previously described. The indices k , m , n and l can take any integer value from 0 to 9, in such a way that the maximum number of images is 9999.

Each PART_klmn file is opened, recorded and closed in each call to POUTE_2D.f or POUTE_2D.f subroutines, so, a single UNIT=23 is assigned to all PART_klmn files.

GRD_dklmn, GRD_uklmn, GRD_vklmn

These files contain values for water depth, x -direction velocity, y -direction velocity interpolated to a regular grid of spacing (dx_{grd} , dy_{grd}) with $nplot_x$ and $nplot_y$ points in the x - and y -directions, respectively.

Created by SPHYSICS_SWE subroutines POUTE_GRD_1D.f or POUTE_GRD_2D.f with the same periodicity as PART_klmn.

UNIT=24

For example, in GRD_dklmn the following variables are recorded:

DSAA
 nplot_x, nplot_y
 0.0, vlx
 0.0, vly
 0.0, m_dw_grd
 dw(1,1) dw (2,1) ... dw(nplot_x,1)
 dw(1,2) dw (2,2) ... dw(nplot_x,2)
 ...
 dw(1,nplot_y) dw (2,nplot_y) ... dw(nplot_x,nplot_y)

DT

Created by SPHYSICS_SWE in POUTE_1D.f or POUTE_2D.f

UNIT=19

The following variables are recorded:

time dtnew

Description:

time: Time instant (in seconds)

dtnew: Time step corresponding to next step.

RESTART

Created by SPHYSICS_SWE in SPHYSICS_SWE_1D/2D.f

UNIT=44

The following variables are recorded:

itime time ngrab dt

Description:

itime: Number of iterations since the beginning of the run.

time: Time instant (in seconds).

ngrab: Recording instant.

dt: Time step

3.2.2.3. Subroutines

All subroutines in SPHYSICS_SWE_gen are placed in the same source file, however SPHYSICS_SWE ones are placed in different source files. A short description of each possible subroutine follows.

filename	subroutine name	subroutine called	called by	purpose
ac	ac	celij, celij_vir, celij_ob, self	step	sweeps over link list grid to compute accelerations
ac_alpha	ac_alpha	celij_alpha, celij_alpha_vir, celij_alpha_ob,	ac_dw	sweeps over link list grid to compute

		self_alpha		alpha
ac_b	ac_b	celij_b_c, celij_b	source_slope	sweeps over link list grid to compute bottom gradient
ac_corr	ac_corr	celij_corr, self_corr	ac_dw	sweeps over link list grid to compute kernel gradient correction
ac_dw_var	ac_dw	celij_dw, self_dw, celij_dw_vir, celij_dw_ob, ac_alpha, ac_corr, grid_h_var	step	sweeps over link list grid to compute d and h
ac_dw_var_hj	ac_dw_hj	celij_dw, self_dw, celij_dw_vir, celij_dw_ob, ac_alpha, grid_h_var, loop	step	sweeps over link list grid to compute d and h when splitting is activated
bottom	bottom	celij_hb	step	sweeps over link list grid to compute bottom elevation
celij_alpha	celij_alpha	kernel	ac_alpha	
celij_alpha_ob	celij_alpha_ob	kernel	ac_alpha	
celij_alpha_vir	celij_alpha_vir	kernel	ac_alpha	
celij_b	celij_b	kernel	ac_b	
celij_balsara	celij	limiter, balsara, kernel	ac	
celij_balsara_ob	celij_ob	limiter, balsara, kernel	ac	
celij_b_c	celij_b_c	kernel	ac_b	
celij_corr	celij_corr	kernel	ac_corr	
celij_dw	celij_dw	kernel	ac_dw	
celij_dw_hj	celij_dw	kernel	ac_dw	
celij_dw_ob	celij_dw_ob	kernel	ac_dw	
celij_dw_vir	celij_dw_vir	kernel	ac_dw	
celij_hb	celij_hb	kernel	bottom	
celij_ob	celij_ob	kernel, viscosity	ac	
celij_vir	celij_vir	kernel, viscosity	ac	
celij_vir_balsara	celij_vir	limiter, balsara, kernel	ac	
celij_visc	celij	limiter, kernel, viscosity	ac	
check_limits	check_limits	new_ob_flp, new_fl_ob, new_bcp	step, sph	
divide	divide		step	Link – list

				creation for fluid particles
divide_b	divide_b		sph	Link – list creation for bottom particles
divide_ob	divide_ob		sph	Link – list creation for open boundary particles
divide_vir	divide_vir		sph	Link – list creation for virtual particles
getdata	getdata		sph	Reads the initial data
grid_h_var	grid_h_var	ini_divide, divide, divide_vir, divide_ob	ac_dw	Creation of the 2h grid with size hmax
ini_divide	ini_divide		grid_h_var	
interp_openbc	interp_openbc		step	Interpolates open boundary condition
kernel_cubic	kernel_cubic		Any self*.f and celij*.f subroutines	Cubic kernel
limiter_minmod			celij_visc, celij_balsara, self_visc, self_balsara	Minmod limiter
limiter_noMUSCL			celij_visc, celij_balsara, self_visc, self_balsara	No limiter
loop	loop		ac_dw	Optimization of the d and h calculation when splitting is activated
open_bc	open_bc		sph, step	Updates physical quantities in the buffer zones
open_bc_pos	open_bc_pos		step	Updates open boundary particle velocities and positions
poute	poute		sph	Output PART*.f files
poute_grid	poute_grid		sph	Output in *.grd files

recount	recount		step	
refinement	refinement		sph	Split particles
refinement_v	refinement_v		sph	Compute the velocity of split particles
riemann_balsara	balsara		celij_balsara, celij_balsara_ob, celij_vir_balsara, self_balsara	Balsara stabilization method
self_alpha	self_alpha	kernel	ac_alpha	
self_alpha_hj	self_alpha	kernel	ac_alpha	
self_balsara	self_balsara	limiter, balsara, kernel	ac	
self_corr	self_corr	kernel	ac_corr	
self_dw	celij_dw	kernel	ac_dw	
self_dw_hj	celij_dw	kernel	ac_dw	
self_visc	celij	limiter, kernel, viscosity	ac	
source_slope	source_slope	ac_b	step	Compute gradient of the bed elevation, and fiction source term
SPHYSICS_SWE_2D	sph	ini_divide, divide_b, divide_ob, divide_vir, getdata, check_limits		Main
step_leap_frog	step	variable_time_step, ac, source_slope, interp_openbc, open_bc_pos, refinement, ini_divide, divide, divide_vir, divide_ob, check_limits, recount, bottom, ac_dw, open_bc, refinement_v		
variable_time_step	variable_time_step		step	CFL condition to compute timestep
viscosity_artificial	viscosity		celij_ob, celij_vir, celij, self	Artificial viscosity
viscosity_LF	viscosity		celij_ob, celij_vir, celij, self	Lax-Friedrichs flux

4. TEST CASES

4.1. Running the model

Creating and running executable files can be done step by step by the user (compiling the different source files, putting them in a certain directory and executing the codes while typing the values of the different variables and options when prompted). Nevertheless, this process can become tedious, especially when running different realizations of the same case with small differences in a small number of parameters. The entire process can be automatically done, although with some differences on different computer systems. Here we will show two examples for WINDOWS and LINUX.

NOTE: the default Compiler chosen is INTEL IFORT for WINDOWS, which is option 3 near the end of each Case file.

4.1.1. Compiling and executing on Linux

SPHYSICS_SWE also currently supports the following fortran compilers that have been tested on Linux platforms,

1. gfortran, a free Fortran 95/2003 compiler that can be downloaded from <http://gcc.gnu.org/wiki/GFortran>.
2. The non-commercial Intel ® Fortran Compiler can be downloaded from <http://www.intel.com> website.

In order to run SPHYSICS_SWE on Linux, gfortran, ifort and the GNU make utility need to be installed and available in the default search path (typically /usr/bin or /usr/local/bin). The following paragraphs explain the procedure to compile and run the 2D version of SPHYSICS_SWE. The procedure is exactly the same for the 1-D version.

Compiling SPHYSICS_SWE_gen

In the SWE-SPHysics_2D/source/SPHYSICS_SWE_gen_2D directory there are two Makefiles named SPHysicsgen_gfortran.mak and SPHysicsgen_ifort.mak. As their names suggest, they are used to compile SPHYSICS_SWE_gen using the gfortran and ifort compilers respectively. The gfortran Makefile can be executed using the command 'make -f Makefile_gfortran.mak'. The Makefile,

1. compiles SPHYSICS_SWE_gen
2. checks for existence of SPHysics_2D/execs and SWE-SPHysics_2D/execs.bak directories. If non-existent these directories are created.
3. moves the previous version of the SPHYSICS_SWE_gen executable, if available, from the execs directory to execs.bak directory

4. moves the latest compiled version of SPHYSICS_SWE_gen to the execs directory.

Running SPHYSICS_SWE_gen_2D and SPHYSICS_SWE_2D

As mentioned before, SPHYSICS_SWE_gen_2D, based on the options chose by the user, generates the Makefile, SPHysics.mak, to compile the main program SPHysics. The subroutines tocompile_gfortran and tocompile_ifort, in SPHYSICS_SWE_gen_2D, write out SPHysics.mak for gfortran and ifort compilers respectively.

There are linux batch files located in the four 2D example directories, run_directory/CaseN, where N=1,2,3,4. These batch files are named CaseN_unix_gfortran.bat (N=1,2,3,4) . Similar linux batch files are located in the 2D example directories.

The following table gives a detailed description of the commands used in the script file *Case1_unix_gfortran.bat* which is located in SWE-SPHysics_2D/run_directory/Case1. This batch file can be executed, while in the Case1 directory, by typing *Case1_unix_gfortran.bat* at the command prompt.

COMMAND	COMMENTS
<code>cd ../../source/ SPHYSICS_SWE_gen_2D/</code>	Change to source directory in order to compile SPHysicsgen using SPHysicsgen.mak
<code>make -f SPHYSICS_SWE_gen_gfortran.mak clean</code>	Remove any preexisting object files
<code>make -f SPHYSICS_SWE_gen_gfortran.mak</code>	Compile and generate SPHYSICS_SWE_gen_2D using SPHysicsgen.mak. This Makefile compiles and places the SPHysicsgen_2D executable in the execs directory and moves the older executable to the execs.bak directory.
<code>cd ../../run_directory/Case1</code>	Change to the Case1 example directory.
<code>../../execs/ SPHYSICS_SWE_gen_2D< Case1.txt > Case1.out</code>	Run SPHYSICS_SWE_gen_2D with Case1.txt as the input file instead of command line input. The output from the execution is redirected in Case1.out

<code>cp SPHysics.mak</code>	Copy the generated Makefile to the SPHysics2D source directory.
<code>.././source/SPHYSICS_SWE_2D</code>	
<code>cd .././source/SPHYSICS_SWE_2D</code>	Change to source directory in order to compile <i>SWE-SPHysics</i> using SPHysics.mak
<code>make -f SPHysics.mak clean</code>	Remove any preexisting object files
<code>make -f SPHysics.mak</code>	Compile and generate <i>SPHYSICS_SWE_2D</i> using SPHysics.mak. Similar to the Makefiles for <i>SPHYSICS_SWE_gen_2D</i> , this Makefile compiles and places the <i>SPHYSICS_SWE_2D</i> executable in the execs directory and moves the older executable to the execs.bak directory
<code>rm SPHysics.mak</code>	Remove the Makefile from the <code>source/SPHYSICS_SWE_2D</code> directory.
<code>cd .././run_directory/Case1</code>	Change to the Case1 example directory.
<code>.././execs/SPHYSICS_SWE_gen_2D</code>	Execute SPHysics_2D and direct the output from the run to sph.out

4.1.2. Compiling and executing on Windows.

In the `SWE-SPHysics_2D/source/SPHYSICS_SWE_gen_2D` directory there are two Makefiles named `SPHysicsgen_win_ifort.mak` and `SPHysicsgen_ftn95.mak`. They are used to compile *SPHYSICS_SWE_gen_2D* using the INTEL IFORT compiler and Silverfrost FTN95 compiler (previously Salford Fortran).

As mentioned before, *SPHYSICS_SWE_gen_2D*, based on the options chosen by the user, generates the Makefile, *SPHysics.mak*, to compile the main program *SPHysics*. The subroutine `tocompile_windows` and `tocompile_ftn95`, in *SPHYSICS_SWE_gen_2D*, write out *SPHysics.mak* for ifort and silverfrost ft95 compilers respectively.

There are windows batch files located in the example directories. The batch file *Case1_windows_ifort.bat* located in `SWE-SPHysics_2D\run_directory\Case1` (see Fig. 3.1) is used. Similar batch files correspond to other 2D examples. Examples corresponding to 1D calculations can be found in `SWE-SPHysics_1D\run_directory\Case1`

The user should, while in the Case1 directory, write *Case1_windows_ifort.bat* on a command window. The content of this file is briefly describe in next table.

COMMAND	COMMENTS
del *.exe	Remove previous executable files.
cd ../../source\SPHYSICS_SWE_gen_2D	Change to the directory containing the SPHYSICS_SWE_gen_2Dsource files
NMAKE /f "SPHYSICS_SWE_gen_win_ifort.mak"	NMAKE /f "SPHysicsgen.mak" is used to compile SPHYSICS_SWE_gen_2D.exe.
cd ../../run_directory\Case1	Change directory
copy ../../execs\ SPHYSICS_SWE_gen_2D.exe SPHYSICS_SWE_gen_2D.exe	Copy SPHYSICS_SWE_gen_2D.exe file to the working directory.
SPHYSICS_SWE_gen_2D.exe <Case1.txt > Case1.out	Run SPHYSICS_SWE_gen_2D.exe. This program creates the initial conditions and select the options of the run. In addition, it also creates a file SPHysics.mak that can be used to compile the SPHYSICS_SWE_2Dcode with the right options. Any name can be used for the input and output files
copy SPHysics.mak ../../source\SPHYSICS_SWE_2D\SPHysics.mak	Copy the SPHysics.mak file to the place where the SPHYSICS_SWE_2Dsource files are located.
cd ../../execs\	Change to the directory where the executable file will be created.
del *.obj	Remove previous object files
del SPHYSICS_SWE_2D.exe	Remove previous executable versions of SPHYSICS_SWE_2D.exe
cd..source\ SPHYSICS_SWE_2D	Change to the directory containing the SPHYSICS_SWE_2Dsource files
NMAKE /f "SPHysics.mak"	NMAKE /f "SPHysics.mak" is used to compile SPHysics_2D.exe. There are multiple options to compile SPHYSICS_SWE_2D.exe. They are automatically selected depending on the initial conditions provided by the

	input file (Case1.txt in this example). The file SPHysics.mak, which is automatically created by SPHysics_SWE_gen_2D.exe, contains information about those options.
cd ../../run_directory/Case1	Change directory
copy ../../execs\SPHysics_SWE_2D.exe SPHysics_SWE_2D.exe	Copy SPHysics_SWE_2D.exe file to the working directory
SPHysics_SWE_2D.exe >sph.out	Run the case. Any name can be used for the output file sph.out

4.2. 1-D Test Cases

4.2.1 Test case 1: 1D Wet-bed Dam break

The case can be run using *Case1.bat*: (Case1_windows_ifort.bat, Case1_windows_ftn95.bat, Case1_unix_gfortran.bat or Case1_unix_ifort.bat) whose output directory is *Case1*. The input file *Case1.txt* is located in the output directory. The information contained in that file can be summarized as follows:

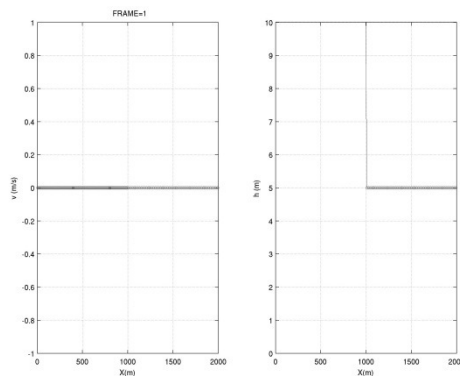


Figure 4.1: Initial configuration of Case1.

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (detailed output at every timestep) 0 for no
1000.	fluid density (1000 kg/m ³ for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrich's flux, 3 - two shocks Riemann solver
2.	valule of alpha (useful just for artificial viscosity)
0	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
1	number of time steps for Newton - Raphson iterative procedure (suggested 10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no

10.	time step, or maximum time step if variable time step activated
0.4	Courant number
10.	output interval (seconds)
50.1	End of the simulation (seconds)
0.	time step for initial output (seconds)
1.2	smoothing length coefficient: $h=dx*coef$ (suggested 1.2)
2000.	length of the domain (m)
0	open boundaries: 0 =no, 2 =downstream, 1 =upstream, 3=both
1	Choose Bed Profile: 1=Plane Slope
10.	Distance between bottom particles (m)
0.	from 0 to xini the bottom is flat, xini (m)
0.	initial beach elevation between 0 and xini (m)
0.	beach slope (degree)
0.	x-coordinate for the beginning of the block
1000.	x-coordinate for the end of the block
10.	water surface elevation at the beginning of the block
10.	water surface elevation at the end of the block
0.	velocity at the beginning of the block
0.	velocity at the end of the block
10.	particle spacing for this block
1	adding another block
1000.	x-coordinate for the beginning of the block
2000.	x-coordinate for the end of the block
5.	water surface elevation at the beginning of the block
5.	water surface elevation at the end of the block
0.	velocity at the beginning of the block
0.	velocity at the end of the block
20.	particle spacing for this block
0	adding another block
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=CVF, 4=Silv
1	1 for single precision, 2 for double precision

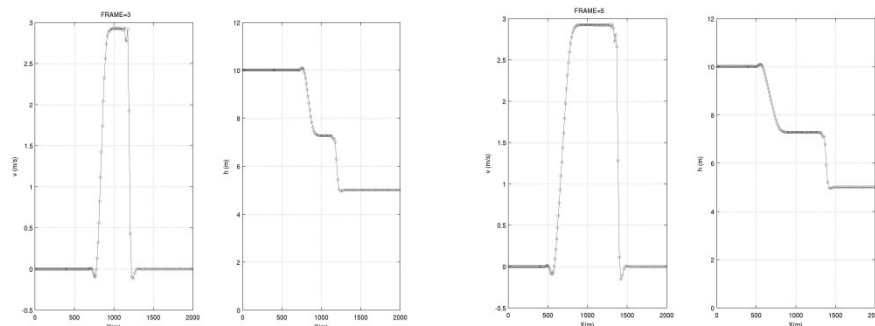


Figure 4.2: Velocity and Depth plots for Case1.

4.2.2. Test case 2: 1D Dry-bed Dam break

The case can be run using *Case2.bat* whose output directory is *Case2*. The input file *Case2_DryBed_DamBreak.txt* is located in the output directory. The information contained in that file can be summarized as follows:

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (detailed output at every timestep) 0 for no
1000.	fluid density (1000 kg/m ³ for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrich, 3 - two shocks Riemann solver
2.	valule of alpha (useful just for artificial viscosity)
1	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
1	number of time steps for Newton - Raphson iterative procedure (suggested 10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no
10.	time step, or maximum time step if variable time step activated
0.4	Courant number
10.	output interval (seconds)
50.1	End of the simulation (seconds)
0.	time step for intitial output (seconds)
1.2	smoothing length coefficient: $h=dx*coef$ (suggested 1.2)
2000.	length of the domain (m)
0	open boundaries: 0 =no, 2 =downstream, 1 =upstream, 3=both
1	Choose Bed Profile: 1=Plane Slope
10.	Distance between bottom particles (m)
0.	from 0 to xini the bottom is flat, xini (m)
0.	initial beach elevation between 0 and xini (m)
0.	beach slope (degree)
0.	x-coordinate for the beginning of the block
1000.	x-coordinate for the end of the block
10.	water surface elevation at the beginning of the block
10.	water surface elevation at the end of the block
0.	velocity at the beginning of the block
0.	velocity at the end of the block
10.	particle spacing for this block
0	adding another block
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=win_ifort, 4=Silverfost)
1	1 for single precision, 2 for double precision

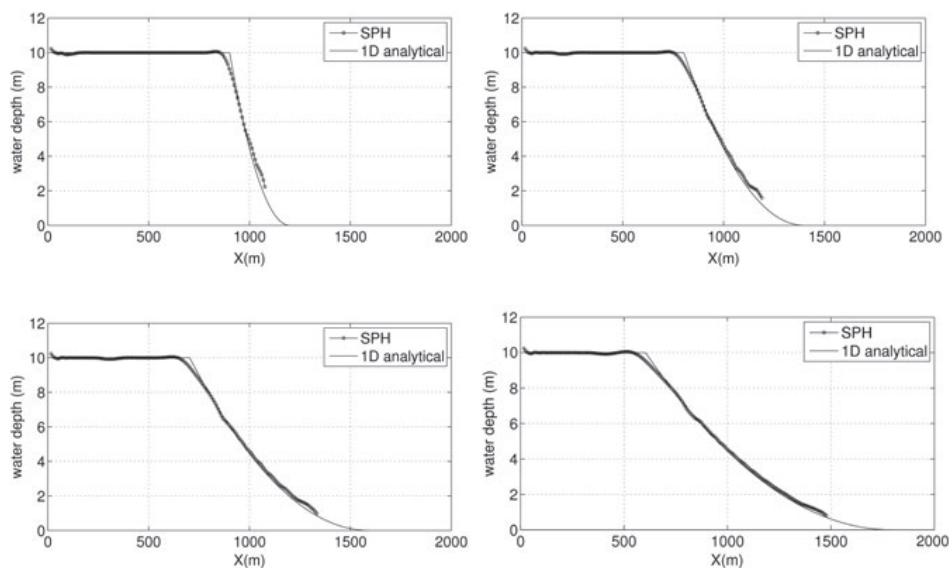


Figure 4.3: Case2 water depths using Lax-Friedrich stabilization terms (Vacondio *et al.* 2011b) $t = 10, 20, 30, 50$ s.

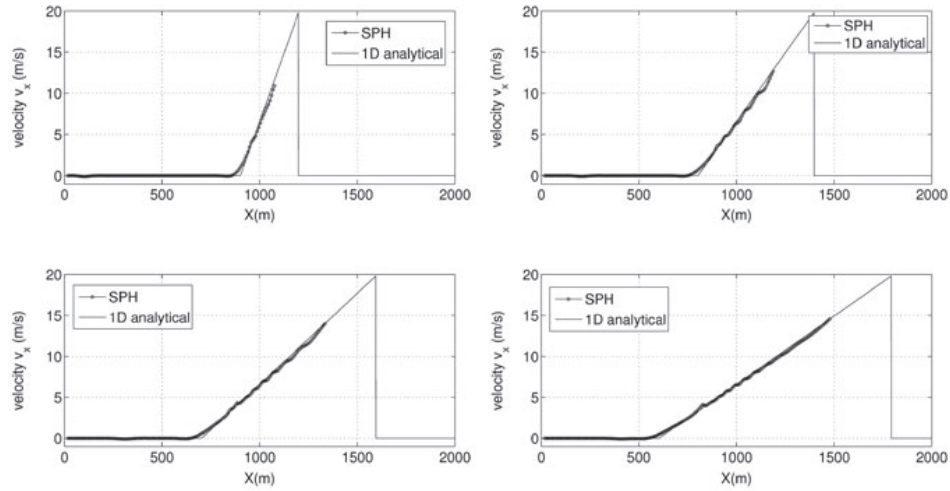


Figure 4.4:Case2 depth-averaged velocities using Lax-Friedrich stabilization terms (Vacondio et al. 2012a) $t = 10, 20, 30, 50$ s.

4.2.3 Test case 3: 1-D Flow over a hump with Inflow-Outflow Boundary Conditions

The case can be run using *Case3.bat* whose output directory is *Case3*. The input file *Case3.txt* is located in the output directory.

Note: There are THREE test cases here demonstrating the effects of changing the outflow boundary conditions (see README file).

1. Case3a - Transcritical Flow

Sub-critical inflow & Sub-critical outflow

2. Case3b - Transcritical Flow

Sub-critical inflow & Super-critical outflow

3. Case3c - Super-critical Flow

Super-critical inflow & Super-critical outflow

DEFAULT input file is Case3a.txt

ALL .bat files load Case3.txt, so to run

- (i) Case3a COPY Case3a.txt to Case3.txt
- (ii) Case3b COPY Case3b.txt to Case3.txt
- (iii) Case3c COPY Case3c.txt to Case3.txt

The information contained in that file can be summarized as follows:

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (detailed output at every timestep) 0 for no
1000.	fluid density (1000 kg/m ³ for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrich's flux, 3 - two shocks Riemann solver
2.	valule of alpha (useful just for artificial viscosity)
1	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
1	number of time steps for Newton - Raphson iterative procedure (suggested 10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no
10.	time step, or maximum time step if variable time step activated
0.4	Courant number
1.	output interval (seconds)
50.1	End of the simulation (seconds)
0.	time step for intitial output (seconds)
1.2	smoothing length coefficient: $h=dx*coef$ (suggested 1.2)
10.	length of the domain (m)
3	open boundaries: 0 =no, 2 =downstream, 1 =upstream, 3=both
2	kind of open boundary upstream: 2 for inflow or 3 - for outflow
0.05	dx for upstream bc (m)
0.1	water surface elevation for upstream bc (m)
4.	velocity for upstream bc (m/s)
5	kind of open boundary downstream: 4 for inflow or 5 - for outflow
0.05	dx for downstream bc (m)
0.1	water surface elevation for downstream bc(m)
4.	velocity for downstream bc (m/s)
2	Choose Bed Profile 2 - Parabolic
0.05	distance between bottom particles (m)
5	parabola equation $y-y_0=h_0*[(x-x_0)/a]**2$, x_0 parameter (m)
0.2	parabola equation $y-y_0=h_0*[(x-x_0)/a]**2$, y_0 parameter (m)
2.	parabola equation $y-y_0=h_0*[(x-x_0)/a]**2$, a parameter (m)
-0.2	parabola equation $y-y_0=h_0*[(x-x_0)/a]**2$, h_0 parameter (m)
0	x-coordinate for the beginning of the block (m)
10	x-coordinate for the end of the block (m)
0.3	water surface elevation at the beginning of the block (m)
0.3	water surface elevation at the end of the block (m)
4.	velocity at the beginning of the block (m/s)
4.	velocity at the end of the block (m/s)
0.05	particle spacing for this block (m)
0	Add another block (1=yes)
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=win_ifort, 4=Silverfrost)

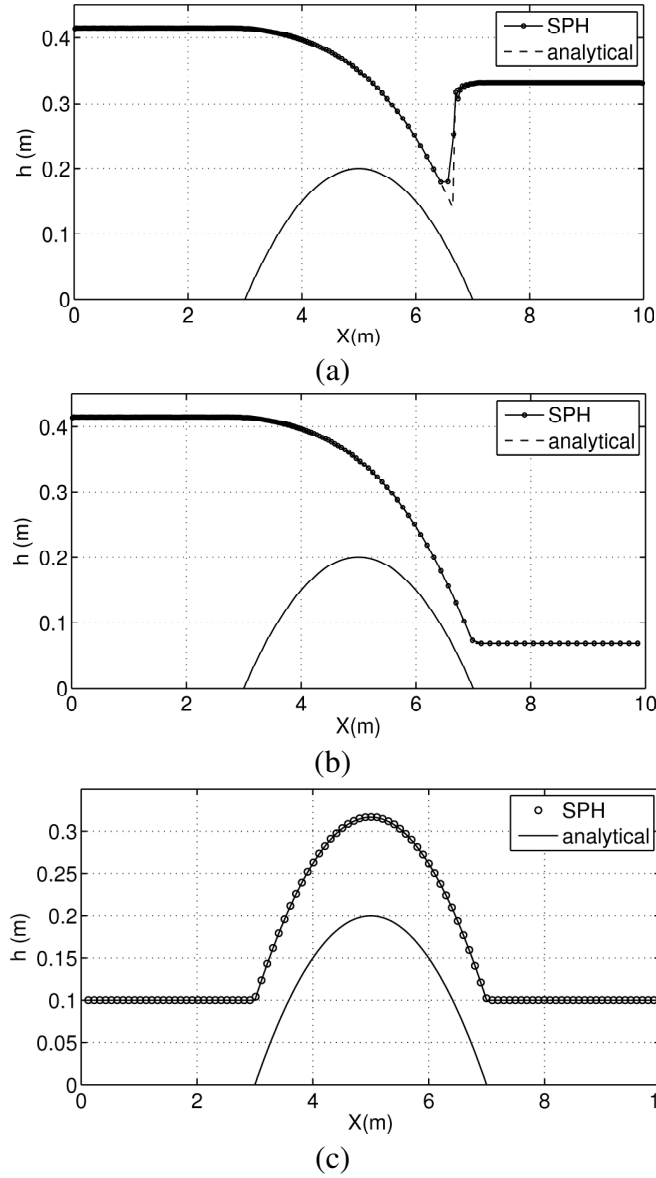


Figure 4.5 Inflow-Outflow over a hump (a) transcritical flow, (b) subcritical inflow & supercritical outflow, (c) supercritical inflow & outflow (Vacondio *et al.* 2012b)

4.3. 2-D Test Cases

4.3.1 Test case 1: 2-D Thacker Basin – rotating surface

Thacker (1981) provides analytical solutions for shallow water flow in basins. This test case simulated by Vacondio *et al.* (2012a) shows good agreement with the analytical solution for an initially sloping water surface rotating around a parabolic basin.

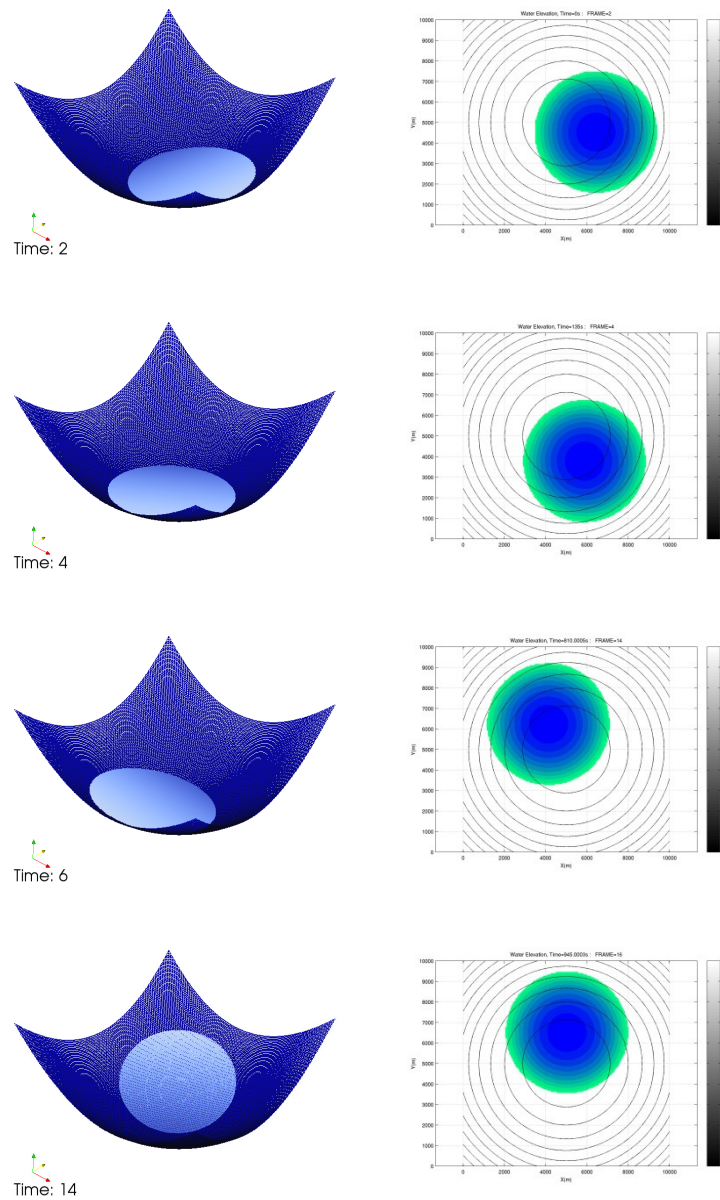


Figure 4.6: Case4 2-D Thacker Basin (Vacondio *et al.*, 2012a), water surface and contours

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (output at every timestep) 0 for no
1000.	fluid density (1000 kg/m3 for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrichs flux, 3 - two shocks Riemann solver
2.	value of alpha (useful just for artificial viscosity)
0.001	minimum depth for the friction source term
1	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
10	number of time steps for Newton - Raphson iterative procedure (suggested 10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no
10.	time step, or maximum time step if variable time step activated
0.4	Courant number
67.5	output interval (seconds)
1350.1	End of the simulation (seconds)
0.	time step for initial output (seconds)
1.2	smoothing length coefficient: $h=dx*coef$ (suggested 1.2)
10000.	x length of the domain (m)

10000.	y - length of the domain (m)
60.	x size of the regular grid for output
60.	y size of the regular grid for output
0	open boundaries: 1 for yes, 0 for no
0	Closed boundaries: 1 for yes, 0 for no
2	Choose Bed Profile 2 = Parabolic basin
3000.	Eq. is $y=h0*(x**2+y**2)/a**2$, parameter a:
10.	Eq. is $y=h0*(x**2+y**2)/a**2$, parameter h0:
10000.	length of the domain
10000.	height of the domain
50.	bottom particle spacing
1.2	coefficient for the smoothing length (sugg. 1.2)
0.	manning coefficient
1	Choose fluid particle distribution 1 square blocks, 2 circular distribution
3000.	initial position of the block, x - coord
0.	initial position of the block, y - coord
7000.	length of the block
10000.	height of the block
0.83333	water surface elevation at South - West (m)
2.416.667	water surface elevation at South - East (m)
0.83333	water surface elevation at North - West (m)
2.416.667	water surface elevation at North - East (m)
0.	x - velocity component at South - West (m)
0.	x - velocity component at South - East (m)
0.	x - velocity component at North - West (m)
0.	x - velocity component at North - East (m)
-700.357	y - velocity component at South - West (m)
-700.357	y - velocity component at South - East (m)
-700.357	y - velocity component at North - West (m)
-700.357	y - velocity component at North - East (m)
50.	particle spacing for this block (m)
0	Add another block (1=yes)
0	refinement procedure 1 for yes 0 for no
120	size of the grid for refinement along x direction (m)
120	size of the grid for refinement along y direction (m)
0.00	minimum water depth for splitting (m)
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=win_ifort, 4=Silverfrost FTN95)
1	1 for single precision, 2 for double precision

4.3.2 Test case 2: 2-D Tsunami Wave with Inflow-Outflow Boundary Conditions

1993 Okushiri tsunami

The Okushiri tsunami in 1993 produced flooding of the coast near Monai, Japan, and was later investigate by a physical model built to 1:400 scale with dimensions 5.448 × 3.402 m. The incoming wave was induced in the wave tank by a mechanical paddle placed at x=0 and the water elevation was measured by three gauges at locations: (4.521, 1.196), (4.521, 1.696) and (4.521, 2.196). The experimental data are available until the reflected wave reaches the paddle (22.5 s after the beginning of the experiment). The registered datasets are available at the Third International Workshop on Long Wave Run-up Models (2004) website.

This test case has been simulated by Vacondio *et al.* (2012b) and includes a number of difficult aspects: open and closed boundaries, irregular bathymetry, wetting and drying fronts and complex shape of the reflected waves.

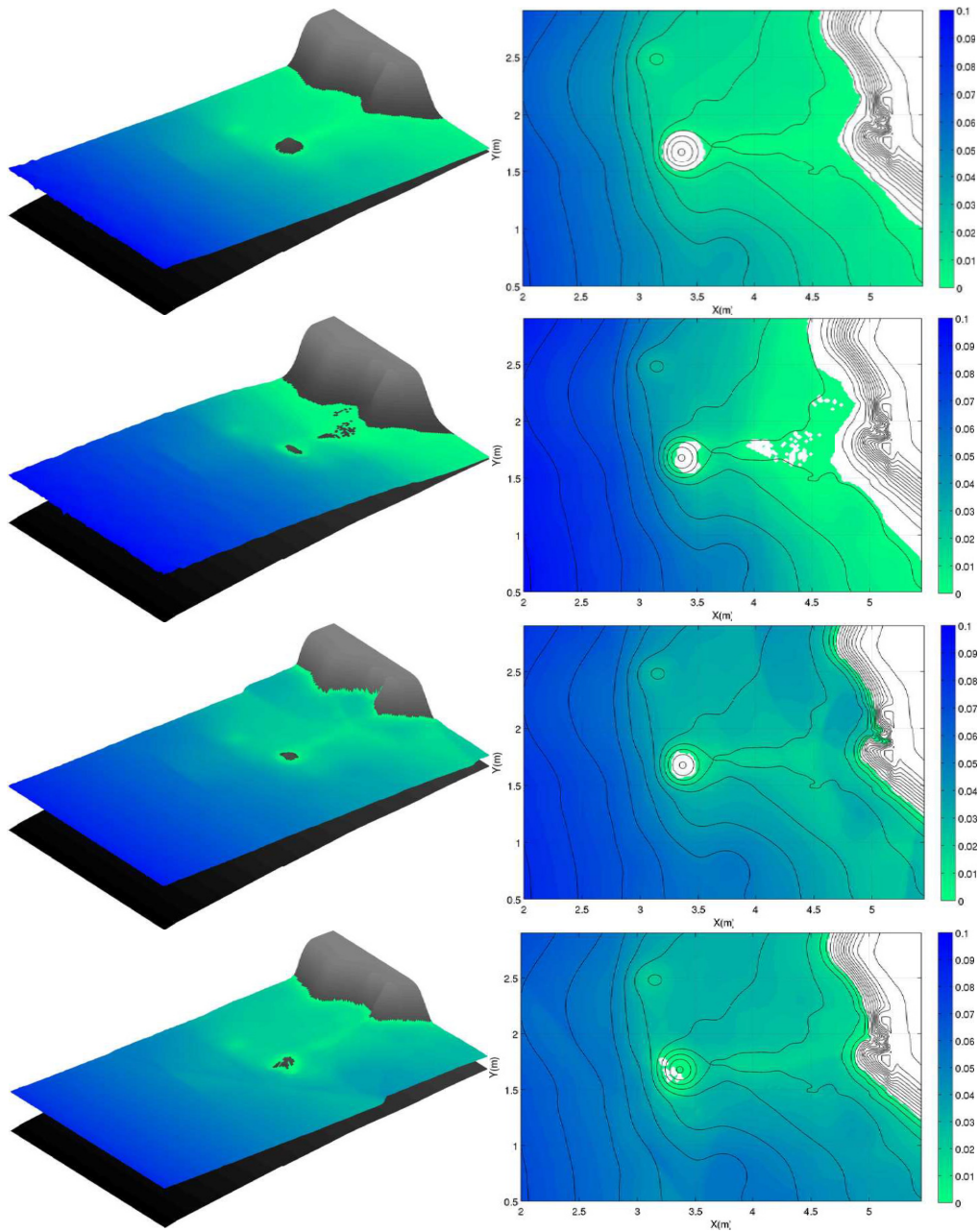


Figure 4.7: Case4 2-D Tsunami with Inflow-Outflow open boundaries (Vacondio *et al.* 2012b) Water surface and contour snapshots

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (output at every timestep) 0 for no
1000.	fluid density (1000 kg/m3 for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrichs flux, 3 - two shocks Riemann solver
2.	value of alpha (useful just for artificial viscosity)
0.001	minimum depth for the friction source term
1	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
1	number of time steps for Newton - Raphson iterative procedure (suggested 10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no
10.	time step, or maximum time step if variable time step activated
0.4	Courant number
0.2	output interval (seconds)
22.51	End of the simulation (seconds)
0.	time step for initial output (seconds)
1.2	smoothing length coefficient: $h=dx*coef$ (suggested 1.2)

5.448	x length of the domain (m)
3.402	y - length of the domain (m)
0.01	x size of the regular grid for output
0.01	y size of the regular grid for output
1	open boundaries: 1 for yes, 0 for no
0.009375	minimum distance between particles in the buffer zone (m)
1	number of boundary conditions
3	kind of open boundary condition: 2 for inflow, 3 for outflow
0.	x-coordinate for the beginning of the openbc (m)
0.	y-coordinate for the beginning of the openbc (m)
3.402	length of the bc (m)
1.	x - coord of the unit vector normal to the bc
0.	y - coord of the unit vector normal to the bc
1.	1 if z is exiting from the plane x-y, -1 if it is entering in the plane x-y
0.01875	dx for open boundary particles (m)
1	steady bc 0 for yes, 1 for no
tsunami_obc	filename with open boundaries
1	closed boundaries: 1 for yes, 0 for no
1	do you want to add a straight closed boundary? 1 for yes
0.	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
5.448	endpoint x - coord (m)
0.	endpoint y - coord (m)
1	marker endpoint point
0.009375	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
1.	unit vector pointing inside the domain, y coord
1	closed boundaries: 1 for yes, 0 for no
0.	starting point x - coord (m)
3.402	starting point y - coord (m)
1	marker starting point
5.448	endpoint x - coord (m)
3.402	endpoint y - coord (m)
1	marker endpoint point
0.009375	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
-1.	unit vector pointing inside the domain, y coord
1	closed boundaries: 1 for yes, 0 for no
5.448	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
5.448	endpoint x - coord (m)
3.402	endpoint y - coord (m)
1	marker endpoint point
0.009375	distance between virtual particles (sugg. 0.5 dx)
-1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
0	do you want to add another closed boundary? 1 for yes 0 for no
0	do you want to add a circular closed boundary 0 for no
3	Choose Bed Profile 3 = Load Bed Profile from File
tsunami_bed	Enter filename of Bed Profile
1	Choose fluid particle distribution 1 = square blocks
0.	initial position of the block, x - coord
0.	initial position of the block, y - coord
5.448	length of the block
3.402	height of the block
.13535	water surface elevation at South - West (m)
.13535	water surface elevation at South - East (m)
.13535	water surface elevation at North - West (m)
.13535	water surface elevation at North - East (m)

0.	x - velocity component at South - West (m)
0.	x - velocity component at South - East (m)
0.	x - velocity component at North - West (m)
0.	x - velocity component at North - East (m)
0.	y - velocity component at South - West (m)
0.	y - velocity component at South - East (m)
0.	y - velocity component at North - West (m)
0.	y - velocity component at North - East (m)
0.01875	particle spacing for this block (m)
0	Add another block (1=yes)
0	refinement procedure 1 for yes 0 for no
0.02	size of the grid for refinement along x direction (m)
0.02	size of the grid for refinement along y direction (m)
0.001	minimum water depth for splitting (m)
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=CVF, 4=Silverfrost FTN95)
1	1 for single precision, 2 for double precision

4.3.3. Test case 3: 2-D Dry-Bed Dam Break with *Particle Splitting*

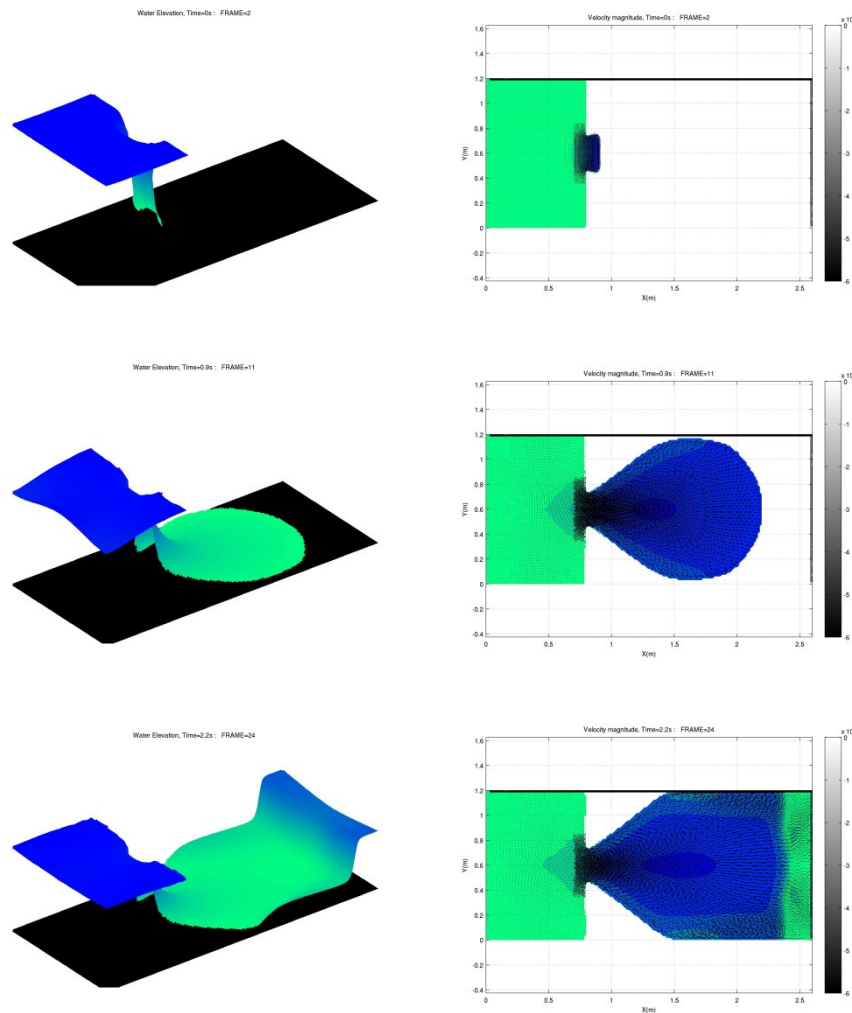


Figure 4.8: Case4 2-D Dam Break from University of Parma with *particle splitting* at *dam exit*: Water surface and velocity vector snapshots

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (output at every timestep) 0 for no

1000.	fluid density (1000 kg/m3 for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrichs flux, 3 - two shocks Riemann solver
2.	value of alpha (useful just for artificial viscosity)
0.001	minimum depth for the friction source term
1	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
1	number of time steps for Newton - Raphson iterative procedure (suggested 10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no
10.	time step, or maximum time step if variable time step activated
0.4	Courant number
0.1	output interval (seconds)
5.01	End of the simulation (seconds)
0.	time step for initial output (seconds)
1.2	smoothing length coefficient: $h=dx*coef$ (suggested 1.2)
2.60	x length of the domain (m)
1.20	y - length of the domain (m)
0.015	x size of the regular grid for output
0.015	y size of the regular grid for output
0	open boundaries: 1 for yes, 0 for no
1	closed boundaries: 1 for yes, 0 for no
1	do you want to add a straight closed boundary? 1 for yes
0.	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
0.8	endpoint x - coord (m)
0.	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.825	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
2.60	endpoint x - coord (m)
0.	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
2.60	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
2.60	endpoint x - coord (m)
1.20	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
-1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
2.60	starting point x - coord (m)
1.20	starting point y - coord (m)
1	marker starting point
0.825	endpoint x - coord (m)
1.20	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
-1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.8	starting point x - coord (m)

1.20	starting point y - coord (m)
1	marker starting point
0.	endpoint x - coord (m)
1.20	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
-1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.	starting point x - coord (m)
1.20	starting point y - coord (m)
1	marker starting point
0.	endpoint x - coord (m)
0.	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.8	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
0.8	endpoint x - coord (m)
0.45	endpoint y - coord (m)
0	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
-1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.8	starting point x - coord (m)
0.75	starting point y - coord (m)
0	marker starting point
0.8	endpoint x - coord (m)
1.2	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
-1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.825	starting point x - coord (m)
0.75	starting point y - coord (m)
0	marker starting point
0.825	endpoint x - coord (m)
1.2	endpoint y - coord (m)
1	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
+1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.825	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
0.825	endpoint x - coord (m)
0.45	endpoint y - coord (m)
0	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.8	starting point x - coord (m)
0.45	starting point y - coord (m)
0	marker starting point

0.825	endpoint x - coord (m)
0.45	endpoint y - coord (m)
0	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.8	starting point x - coord (m)
0.75	starting point y - coord (m)
0	marker starting point
0.825	endpoint x - coord (m)
0.75	endpoint y - coord (m)
0	marker endpoint point
0.0075	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
-1.	unit vector pointing inside the domain, y coord
0	do you want to add another closed boundary? 1 for yes 0 for no
0	do you want to add a circular closed boundary 0 for no
1	Choose Bed Profile 1 = flat bed in blocks
0.	initial position of the block, x - coord
0.	initial position of the block, y - coord
2.60	length of the block
1.20	height of the block
0.015	bottom particle spacing
1.2	coefficient for the smoothing length (sugg. 1.2)
0.007	manning coefficient of the block
0.	constant elevation of the block
0	do you want to add another block 1 for yes
1	Choose fluid particle distribution 1 = square blocks
0.	initial position of the block, x - coord
0.	initial position of the block, y - coord
0.8	length of the block
1.2	height of the block
0.15	water surface elevation at South - West (m)
0.15	water surface elevation at South - East (m)
0.15	water surface elevation at North - West (m)
0.15	water surface elevation at North - East (m)
0.	x - velocity component at South - West (m)
0.	x - velocity component at South - East (m)
0.	x - velocity component at North - West (m)
0.	x - velocity component at North - East (m)
0.	y - velocity component at South - West (m)0.7975
0.	y - velocity component at South - East (m)
0.	y - velocity component at North - West (m)
0.	y - velocity component at North - East (m)
0.015	particle spacing for this block (m)
0	Add another block (1=yes)
1	refinement procedure 1 for yes 0 for no
0.4	eta coefficient for refined particle, distance (suggested 0.4)
0.9	alpha coefficient for refined particle, smoothing length (suggested 0.9)
0.005	size of the grid for refinement along x direction (m)
0.005	size of the grid for refinement along y direction (m)
0.0001	minimum water depth for splitting (m)
0.7	x_min_ref_lim (m)
0.4535	y_min_ref_lim (m)
0.8	x_max_ref_lim (m)
0.7465	y_max_ref_lim (m)
0.9	Ratio of initial (max Area before splitting)/(Area) before splitting
1	Add another block for refinement(1=yes)
0.7	x_min_ref_lim (m)
0.35	y_min_ref_lim (m)
0.79	x_max_ref_lim (m)

0.4530	y_max_ref_lim (m)
0.9	Ratio of initial (max Area before splitting)/(Area) before splitting
1	Add another block for refinement(1=yes)
0.7	x_min_ref_lim (m)
0.7430	y_min_ref_lim (m)
0.79	x_max_ref_lim (m)
0.85	y_max_ref_lim (m)
0.9	Ratio of initial (max Area before splitting)/(Area) before splitting
0	Add another block for refinement(1=yes)
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=CVF, 4=Silverfrost FTN95)
1	1 for single precision, 2 for double precision

4.3.4. Test case 4: 2-D CADAM Case with 45° Channel

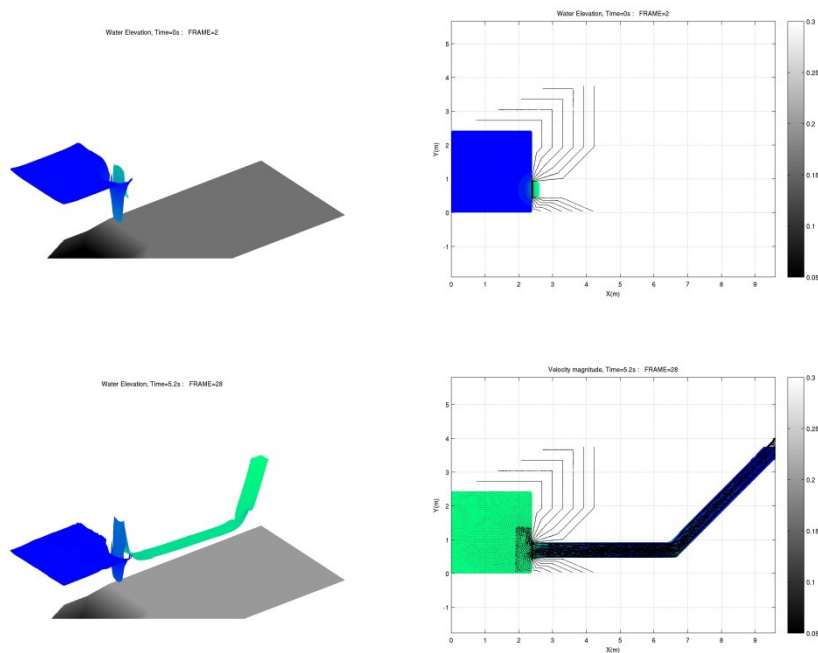


Figure 4.9: Case4 2-D Dam Break from University of Parma with *particle splitting* at dam exit: Water surface and velocity vector snapshots (Vacondio *et al.* 2012a)

0	Choose Starting options: 0=new, 1=restart
0	debug activated: 1 - for yes (output at every timestep) 0 for no
1000.	fluid density (1000 kg/m3 for water)
2	stabilization term 1 - artificial viscosity, 2 - Lax Friedrichs flux, 3 - two shocks Riemann solver
2.	valule of alpha (useful just for artificial viscosity)
0.001	minimum depth for the friction source term
1	0 - No MUSCL reconstruction, 1 - MUSCL reconstruction
1.E-3	tolerance in the Newton - Raphson algorithm (suggested 1.E-3)
1	number of time steps for Newton - Raphson iterative procedure (suggested 1-10)
50	maximum number of iterations in the Newton - Raphson iterative procedure
1	variable time step: 1 for yes, 2 for no
10.	time step, or maximum time step if variable time step activated
0.2	Courant number
0.2	output interval (seconds)
40.01	End of the simulation (seconds)
0.	time step for intital output (seconds)
1.2	smoothing length coefficient: h=dx*coef (suggested 1.2)
9.59	x length of the domain (m)
3.75	y - length of the domain (m)
0.02	x size of the regular grid for output

0.02	y size of the regular grid for output
0	open boundaries: 1 for yes, 0 for no
1	closed boundaries: 1 for yes, 0 for no
1	do you want to add a straight closed boundary? 1 for yes
0.	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
2.39	endpoint x - coord (m)
0.	endpoint y - coord (m)
1	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
2.39	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
2.39	endpoint x - coord (m)
0.445	endpoint y - coord (m)
0	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
-1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
2.39	starting point x - coord (m)
0.445	starting point y - coord (m)
0	marker starting point
6.64	endpoint x - coord (m)
0.445	endpoint y - coord (m)
1	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
6.64	starting point x - coord (m)
0.445	starting point y - coord (m)
1	marker starting point
957.449	endpoint x - coord (m)
337.949	endpoint y - coord (m)
0	marker endpoint point
0.02475	d distance between virtual particles (sugg. 0.5 dx)
-.707107	unit vector pointing inside the domain, x coord
.707107	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.	starting point x - coord (m)
0.	starting point y - coord (m)
1	marker starting point
0.	endpoint x - coord (m)
2.44	endpoint y - coord (m)
1	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
0.	starting point x - coord (m)
2.44	starting point y - coord (m)
1	marker starting point
2.39	endpoint x - coord (m)
2.44	endpoint y - coord (m)
1	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord

-1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
2.39	starting point x - coord (m)
2.44	starting point y - coord (m)
1	marker starting point
2.39	endpoint x - coord (m)
0.94	endpoint y - coord (m)
0	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
-1.	unit vector pointing inside the domain, x coord
0.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
2.39	starting point x - coord (m)
0.94	starting point y - coord (m)
0	marker starting point
64.335	endpoint x - coord (m)
0.94	endpoint y - coord (m)
0	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
0.	unit vector pointing inside the domain, x coord
-1.	unit vector pointing inside the domain, y coord
1	do you want to add another closed boundary? 1 for yes 0 for no
64.335	starting point x - coord (m)
0.94	starting point y - coord (m)
0	marker starting point
9.22	endpoint x - coord (m)
372.954	endpoint y - coord (m)
0	marker endpoint point
0.02475	distance between virtual particles (sugg. 0.5 dx)
+ .707107	unit vector pointing inside the domain, x coord
- .707107	unit vector pointing inside the domain, y coord
0	do you want to add another closed boundary? 1 for yes 0 for no
0	do you want to add a circular closed boundary 0 for no
1	Choose Bed Profile 1 = flat bed in blocks
0.	initial position of the block, x - coord
0.	initial position of the block, y - coord
2.39	length of the block
2.44	height of the block
0.0495	bottom particle spacing
4.8	coefficient for the smoothing length (sugg. 1.2)
0.01	manning coefficient of the block
0.	constant elevation of the block
1	do you want to add another block 1 for yes
2.39	initial position of the block, x - coord
0.445	initial position of the block, y - coord
2.	length of the block
0.495	height of the block
0.0495	bottom particle spacing
4.8	coefficient for the smoothing length (sugg. 1.2)
0.01	manning coefficient of the block
0.33	constant elevation of the block
1	do you want to add another block 1 for yes
4.39	initial position of the block, x - coord
0.	initial position of the block, y - coord
5.61	length of the block
4.5	height of the block
0.0495	bottom particle spacing
4.8	coefficient for the smoothing length (sugg. 1.2)
0.01	manning coefficient of the block
0.33	constant elevation of the block
0	do you want to add another block 1 for yes

1	Choose fluid particle distribution 1 = square blocks
0.	initial position of the block, x - coord
0.	initial position of the block, y - coord
2.39	length of the block
2.44	height of the block
0.58	water surface elevation at South - West (m)
0.58	water surface elevation at South - East (m)
0.58	water surface elevation at North - West (m)
0.58	water surface elevation at North - East (m)
0.	x - velocity component at South - West (m)
0.	x - velocity component at South - East (m)
0.	x - velocity component at North - West (m)
0.	x - velocity component at North - East (m)
0.	y - velocity component at South - West (m)
0.	y - velocity component at South - East (m)
0.	y - velocity component at North - West (m)
0.	y - velocity component at North - East (m)
0.0495	particle spacing for this block (m)
0	Add another block (1=yes)
1	refinement procedure 1 for yes 0 for no
0.4	eta coefficient for refined particle, distance (suggested 0.4)
0.9	alpha coefficient for refined particle, smoothing length (suggested 0.9)
0.02	size of the grid for refinement along x direction (m)
0.02	size of the grid for refinement along y direction (m)
0.001	minimum water depth for splitting (m)
1.90	x_min_ref_lim (m)
0.455	y_min_ref_lim (m)
2.50	x_max_ref_lim (m)
0.90	y_max_ref_lim (m)
0.9	Ratio of initial (max Area before splitting)/(Area) before splitting
1	Add another block for refinement(1=yes)
1.90	x_min_ref_lim (m)
0.04	y_min_ref_lim (m)
23.653	x_max_ref_lim (m)
0.455	y_max_ref_lim (m)
0.9	Ratio of initial (max Area before splitting)/(Area) before splitting
1	Add another block for refinement(1=yes)
1.90	x_min_ref_lim (m)
0.90	y_min_ref_lim (m)
23.653	x_max_ref_lim (m)
1.385	y_max_ref_lim (m)
0.9	Ratio of initial (max Area before splitting)/(Area) before splitting
0	Add another block for refinement(1=yes)
1	Which compiler is desired? (1=gfortran, 2=ifort, 3=CVF, 4=Silverfrost FTN95)
1	1 for single precision, 2 for double precision

5. HOW TO CHANGE *SWE-SPHysics* FOR YOUR APPLICATION

5.1 Introduction

When people start using the SPHysics codes, we often get asked if the code can do a particular function that is not included in the demonstration cases. The answer we give is normally yes, but the particular functionality required may require some re-coding. We do not normally propose to do this re-coding ourselves unless the application area coincides closely with our own area and current projects, or there is a bug. The reason behind this is that SPHysics softwares primarily research codes and we have released what we have found useful for our own research. As the code is research oriented, it is up to the user to adapt the code and the subroutines to their satisfaction.

This short section is aimed at helping those people who want to change the code for their own purposes. Figure 5.1 displays the main structure of the code. Here, we list which subroutines in the code you should examine for possible modification.

Important Note: if you create any new subroutines for the main source code, you must include the names of these new files in the “make files” used for compiling the code which are written in subroutines `tocompile_win_ifort`, `tocompile_ftn95`, `tocompile_gfortran`, `tocompile_ifort` in `SPHYSICS_SWE_gen_1D/2D.f`. Read Section 3.2.2.3 to see where each of the subroutines is compiled.

Changing the boundary conditions.

Boundary conditions are treated in each `celij` & `self` subroutines. Any modification to the boundary conditions should be done in these subroutines.

Changing the timestepping algorithm

The timestepping is performed in all of the `step` subroutines: `step_leap_frog_1D/2D.f`. These subroutines then call subroutines `ac` which control the sweep across the particles (or $2h$ grid) for each (part of the) timestep.

Changing the kernel calculation

The smoothing kernel and its derivatives are calculated in the kernel subroutines: `kernel_cubic_1D/2D.f`.

Changing the viscous formulation

The viscous terms are all calculated in the viscosity subroutines which are called from `celij` & `self`:

`viscosity_artificial_1D/2D.f` & `viscosity_LF_1D/2D.f`.

Loading in data files and setting useful parameters

If you wish to examine and modify what data *SWE-SPHysics* loads initially, all the useful data is imported in subroutine `getdata_1D/2D.f`. Furthermore, all the useful parameters that remain the same throughout the simulation are calculated here such as the kernel normalization factors, etc. All global variables are defined in `global_1D/2D.f`.

Zeroing variables

Many variables that are evaluated throughout the timestep, such as the accelerations, a_x , a_y , a_z are zeroed initially in the different ac subroutines: `ac_1D/2D.f`.

Changing the input bathymetry

SWE-SPHysicscan use both (i) special shapes available in `SPHYSICS_SWE_gen_1D/2D.f` and (ii) arbitrary bed/bottom geometries (bathymetries) which are loaded via input files in the format of CSV files with XYZ data + smoothing length & roughness coefficients. **Case4_Tsunami** uses a file called “`tsunami_bed`”.

Particle refinement

SWE-SPHysicscan split particles according to pre-defined criteria (see Vacondio *et al.* 2011b). In the code, this is controlled in subroutine `refinement_2D.f` and `refinement_v_2D.f`.

5.2 Code Structure

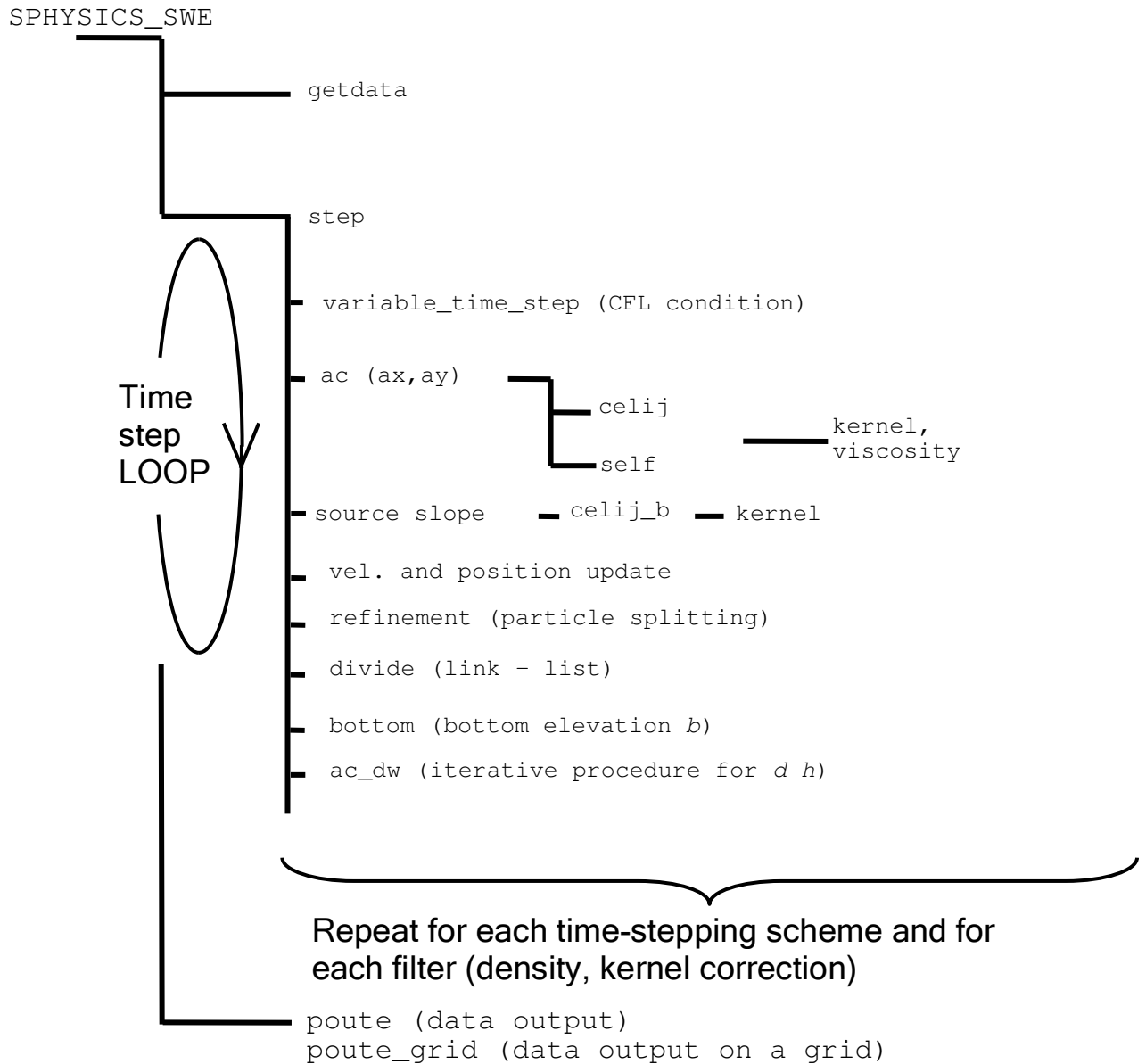


Figure 5.1 Outline of code structure

5.3 Main Variables

Here we present a table of the main variables (or those with less than obvious names) and the counterpart in equations:

SPHysics variable	SPHquantity
udot(i), vdot(i)	$\frac{d\mathbf{v}_a}{dt} = \frac{d\vec{v}_a}{dt}$
dH_SPH_x(i), dH_SPH_y(i)	$\frac{\partial b}{\partial x}, \frac{\partial b}{\partial y}$

ar(i)	$\frac{d\rho_a}{dt}$
cs(i)	c_a
drx, dry	$x_{ab} = (x_a - x_b), y_{ab} = (y_a - y_b)$
dux, duy	$u_{ab} = (u_a - u_b), v_{ab} = (v_a - v_b)$
frxi, frxj, fryi, fryj	$\frac{\partial W_{ab}}{\partial x_a}, \frac{\partial W_{ba}}{\partial x_b}, \frac{\partial W_{ab}}{\partial y_a}, \frac{\partial W_{ba}}{\partial y_b}$
pm(j)	m_b
pVol(j)	$V_b = \frac{m_b}{\rho_b}$
h_t(i)	b
rhop(i)	ρ_a
dw(i)	d
h_var(i)	h
rhop_sum	$\sum_b \rho_b W_{ab} \frac{m_b}{\rho_b} = \sum_b m_b W_{ab}$
rr2	r_{ij}^2
sum_wab	$\sum_b W_{ab} \frac{m_b}{\rho_b}$
up(i), vp(i)	$\mathbf{v}_a = \vec{v}_a = (u_a, v_a)$
xp(i), yp(i)	$\mathbf{r}_a = \vec{r}_a = (x_a, y_a)$
Wab	$W_{ab} = W(\vec{r}_a - \vec{r}_b)$

6. VISUALIZATION

To visualize the results obtained from *SWE-SPHysics* simulations, some basic post-processing programs have been provided in the *SWE-SPHysics_1D/Post-Processing* and *SWE-SPHysics_2D/Post-Processing* directories.

Detailed README files, explaining the procedure to view the results using Matlab and Paraview, are available in those directories. The user is encouraged to read these README files prior to using the visualization programs.

6.1 Using Matlab

To view the results using Matlab, start Matlab and navigate to a *run_directory/CaseN*.

To visualise 1-D results at the command prompt type (and then hit enter):

```
SPHYSICS_SWE_1D_Plot
```

To visualise 2-D results at the command prompt type (and then hit enter):

```
SPHYSICS_SWE_2D_Plot
```

In each case, enter the required information and the plotting routine will cycle through the frames producing images such as those for the 2-D Thacker Basin:

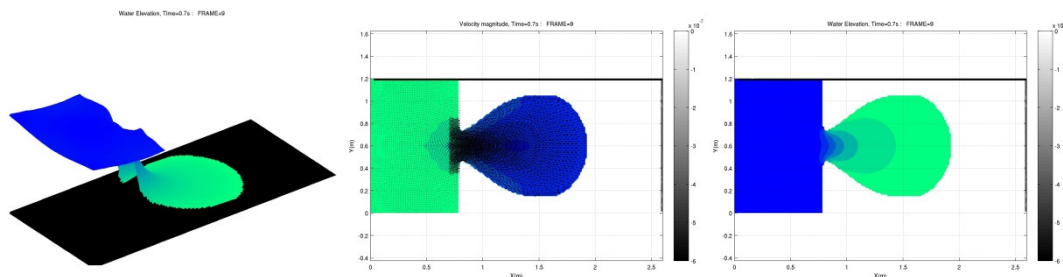


Figure 6.1 Example output images using Matlab



6.2 Using Paraview (open-source) 2-D only

To view the results using the open-source Paraview software (www.paraview.org), the *SWE-SPHysics* output files, *PART_0001*, etc., must be converted to VTK format.

1. For windows or linux, there are different commands:
 - (i) Linux with **gfortran**: Enter
`./PART2VTU_SWE_unix_gfortran.bat`
 - (ii) Linux with **ifort**: Enter
`./PART2VTU_SWE_unix_ifort.bat`
 - (iii) Windows with **ifort**: Enter
`./PART2VTU_SWE_windows_ifort.bat`

(iv) Windows with **ftn95**: Enter

`./PART2VTU_SWE_windows_ftn95.bat`

2. Start Paraview and navigate to `run_directory/CaseN/ParaviewFiles/VTU/`
3. Open `VTUinp.pvd` and click 
4. Open `ibottom.vtu` and click 
5. Use the “Warp by Scalar” filter (Menu: Filters→Alphabetical→Warp by Scalar) and Select “Elevation” to scale the water surface and bed profiles as desired:

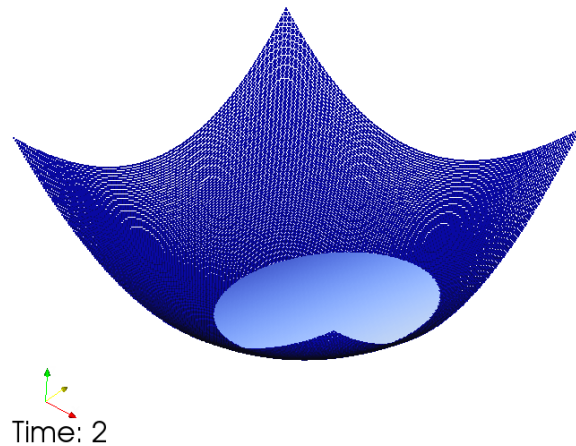
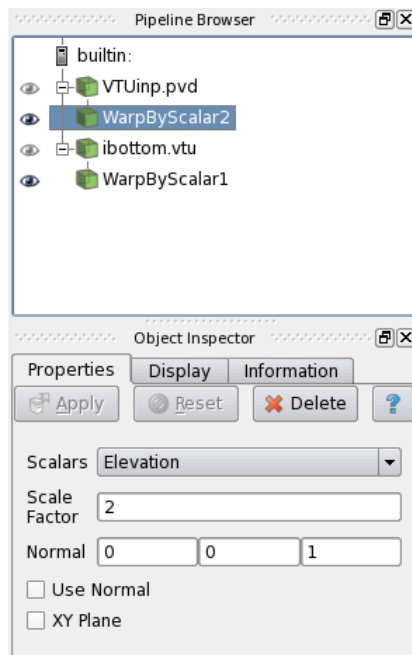


Figure 6.2 Using Paraview and “Warp by Scalar” to display in Paraview

No paraview routines are provided for 1-D.

7. REFERENCES

- Benz W. 1990. Smoothed Particle Hydrodynamics: A review in *The numerical Modelling of Nonlinear Stellar Pulsations: Problems and Prospects*, J.R. Butcher ed., Kluwer Acad. Publ. 269-288
- Gomez-Gesteira M, Rogers B D, Dalrymple R A, Crespo AJC. 2010. "State-of-the-art of classical SPH for free-surface flows". *Journal of Hydraulic Research*. Vol. 48. Issue Extra Issue. pp 6-27. DOI: 10.1080/00221686.2010.9641242
- Gomez-Gesteira M, Rogers B D, Crespo AJC, Dalrymple RA, Narayanaswamy M. In Press. "SPHysics - development of a free-surface fluid solver- Part 1: Theory and Formulations". *Computers & Geosciences*. Vol. 48, November 2012, Pages 289–299 DOI: 10.1016/j.cageo.2012.02.029.

- Gomez-Gesteira M, Crespo AJC, Rogers B D, Dalrymple RA, Dominguez JM. 2012. "SPHysics - development of a free-surface fluid solver- Part 2: Efficiency and test cases". *Computers & Geosciences.*, Vol. 48. pp 300-307. DOI: [10.1016/j.cageo.2012.02.028](https://doi.org/10.1016/j.cageo.2012.02.028).
- Liu, G.R. 2003. Mesh Free methods: Moving beyond the finite element method. CRC Press, pp. 692.
- Monaghan, J. J. 1982 Why particle methods work. *Siam J. Sci. Stat. Comput.* 3: 422-433.
- Monaghan, J. J. 1992. Smoothed particle hydrodynamics. *Annual Rev. Astron. Appl.*,30: 543- 574.
- Monaghan, J. J. 2005. Smoothed Particle Hydrodynamics. *Rep. Prog. Phys.* 68: 1703-1759.
- Thacker WC. 1981. Some exact solutions to the nonlinear shallow-water wave equations. *Journal of Fluid Mechanics*, 107:499–508.
- Vacondio, R., Rogers, B.D., Stansby, P.K. and Mignosa, P. (2013), A correction for balancing discontinuous bed slopes in two-dimensional smoothed particle hydrodynamics shallow water modeling. *Int. J. Numer. Meth. Fluids*. Vol. 71: 850–872. DOI: [10.1002/fld.3687](https://doi.org/10.1002/fld.3687)
- Vacondio R, Rogers B D, Stansby P K. (2011a), "Smoothed Particle Hydrodynamics: approximate zero-consistent 2-D boundary conditions and still shallow water tests". *Int. J. Numer. Meth. Fluids*. Vol. 69. Issue 1. pp 226-253, DOI: [10.1002/fld.2559](https://doi.org/10.1002/fld.2559)
- Vacondio R, Rogers B D, Stansby P K. (2012a), "Accurate particle splitting for SPH in shallow water with shock capturing". *Int. Journal for Numerical Methods in Fluids*. July, Vol. 69. Issue 8. pp 1377-1410, DOI: [10.1002/fld.2646](https://doi.org/10.1002/fld.2646)
- Vacondio R, Rogers B D, Stansby P K, Mignosa P. (2012b), "SPH modeling of shallow flow with open boundaries for practical flood simulation". *Journal of Hydraulic Engineering*. Vol. 138. No. 6. pp. 530-541, DOI: [10.1061/\(ASCE\)HY.1943-7900.0000543](https://doi.org/10.1061/(ASCE)HY.1943-7900.0000543)

8. PUBLICATIONS USING THE *SWE-SPHysics* CODE

Journal Papers

- Vacondio, R., Rogers, B.D., Stansby, P.K. and Mignosa, P. (2013), A correction for balancing discontinuous bed slopes in two-dimensional smoothed particle hydrodynamics shallow water modeling. *Int. J. Numer. Meth. Fluids*. Vol. 71: 850–872. [DOI: 10.1002/fld.3687](https://doi.org/10.1002/fld.3687)
- Vacondio R, Rogers B D, Stansby P K. (2011a), "Smoothed Particle Hydrodynamics: approximate zero-consistent 2-D boundary conditions and still shallow water tests". *Int. J. Numer. Meth. Fluids*. Vol. 69. Issue 1. pp 226-253, [DOI: 10.1002/fld.2559](https://doi.org/10.1002/fld.2559)
- Vacondio R, Rogers B D, Stansby P K. (2012a), "Accurate particle splitting for SPH in shallow water with shock capturing". *Int. Journal for Numerical Methods in Fluids*. July, Vol. 69. Issue 8. pp 1377-1410, [DOI: 10.1002/fld.2646](https://doi.org/10.1002/fld.2646)
- Vacondio R, Rogers B D, Stansby P K, Mignosa P. (2012b), "SPH modeling of shallow flow with open boundaries for practical flood simulation". *Journal of Hydraulic Engineering*. Vol. 138. No. 6. pp. 530-541, [DOI: 10.1061/\(ASCE\)HY.1943-7900.0000543](https://doi.org/10.1061/(ASCE)HY.1943-7900.0000543)

Conference Proceedings

- Vacondio R, Rogers B D, Stansby P K, Mignosa P, Feldman J. 2011b "A dynamic particle coalescing and splitting scheme for SPH". *Proc. 6th International SPHERIC Workshop*. Editor T. Rung & C. Ulrich. pp 93-100. 8-10 June 2011.
- Vacondio R, Mignosa P, Rogers B D, Stansby P K. "SPH Shallow Water Equation Solver for real flooding simulation". *Proc. 5th International SPHERIC Workshop*. Editor B.D. Rogers. pp 106-113. June 2010.
- Vacondio R, Mignosa P, Rogers B D, Stansby P K. "2-D numerical modeling of rapidly varying shallow water flows by Smoothed Particle Hydrodynamics technique". *Proc. of Riverflow 2010*. pp 1621. 8-10 September 2010.

PhD Thesis

- Vacondio, R., Shallow Water and Navier-Stokes SPH-like numerical modelling of rapidly varying free-surface flows, Università degli Studi di Parma Facoltà di Ingegneria, 2010.