 CMES, vol.25, no.3, pp.133-147, 2008

Parallel 3-D SPH Simulations

C. Moulinec[1], R. Issa[2], J.-C. Marongiu[3] and D. Violeau[4]

Abstract: The gridless Smoothed Particle Hydrodynamics (SPH) numerical method is preferably used in Computational Fluid Dynamics (CFD) to simulate complex flows with one or several convoluted free surfaces. This type of flows requires distorted meshes with classical Eulerian mesh-based methods or very fine meshes with Volume of Fluid method. Few 3-D SPH simulations have been carried out to our knowlegde so far, mainly due to prohibitive computational investment since the number of particles required in 3-D is usually too large to be handled by a single processor. In this paper, a parallel 3-D SPH code is presented. Parallelisation validation is discussed as well as promising results regarding speed up of the code. The results of quantitative validations for 2 academic test cases, *i.e.* a 3-D closed channel flow and a 3-D periodic hill flow are also shown. The code is then applied to a more industrial case, namely a 3-D dam breaking flow impinging on an obstacle. SPH results are satisfactorily compared to experimental data.

Keyword: SPH, parallelism, speed-up, 3-D closed channel flow, 3-D hill flow, 3-D dam breaking dam.

1 Introduction

The Smoothed Particle Hydrodynamics (SPH) numerical method is successfully applied in fluid mechanics to simulate strongly distorted free surface flows (dam breaking, wave flumes, etc.) in situation where classical Eulerian approaches would fail because of grid steepness. SPH is very efficient for rapid, convection dominated flows. Few 3-D SPH simulations have been performed to our knowledge so far, mainly due to their high computational cost, because a large number of particles is necessary to simulate such flows and because of the nature of classical SPH which requires homogeneous initial particle distribution for incompressible flows. However, most realistic environmental or/and industrial problems involve 3-D phenomena. In order to handle those problems while reducing computational time, a parallel 3-D SPH code, namely Spartacus-3D, has been developped and is presented here. After describing the methodology, some SPH principles and the numerics used in Spartacus-3D, the parallelisation technique is explored. The code is then applied to two academic test cases, a 3-D closed channel flow for which an analytical solution exists and a 3-D periodic hill where results are compared to data from a widely validated Eulerian code (*Code_Saturne* [Archambeau, Méchitoua, and Sakiz (2004)]), and finally to a 3-D dam breaking flow impinging on an obstacle, where results are compared to experimental data ([Kleefsman, Fekken, Veldman, Iwanowski, and Buchner (2005)]).

2 Methodology

The pseudo-compressible Navier-Stokes equations written in Lagrangian form for Newtonian fluids and incompressible and laminar flows read:

$$\begin{cases} \frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = -\frac{1}{\rho}\nabla p + \nabla\cdot(\nu\nabla\mathbf{u}) + \mathbf{F^e} \\ \frac{\mathrm{d}\rho}{\mathrm{d}t} = \rho\nabla\cdot\mathbf{u} \end{cases} \tag{1}$$

where $\mathbf{u}$ is the velocity vector, t the time, ρ the density, p the pressure, ν the molecular viscosity and $\mathbf{F^e}$ an external force, such as gravity, or any other force driving the flow. '∇' and '$\nabla\cdot$' are respectively the gradient and divergence operators.

[1] INCKA, France.
[2] EDF R&D, France. Corresponding author. Email: reza.issa@edf.fr
[3] ECL, France.
[4] EDF R&D, France.

The system made of 4 equations in 3-D (see Eq. 1) is not closed, because $\mathbf{u}$, p and ρ are all unknowns. To close it, the pressure is expressed in function of the density following a state equation [Monaghan (1992)], as:

$$p = \frac{\rho_0 c_0^2}{\gamma}\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{2}$$

where ρ_0 is a reference density (1,000 $kg.m^{-3}$ for water for instance), c_0 a numerical speed of sound and γ a constant coefficient equal to 7 for water. In order to simulate an incompressible flow, c_0 must be at least ten times larger than the maximal velocity of the flow. The nearly-incompressible assumption used here implies that the Mach number M has to be less than 0.1. Consequently, the relative variation of density, which scales as M^2, is less than 1%. Through Eq. 2, it comes that the pressure automatically goes to zero when density equals the reference density. This ensures the zero pressure condition at a free surface. Fully incompressible algorithms would be more suitable to ensure a better pressure field accuracy [Cummins and Rudman (1999), Lee, Moulinec, Xui, Violeau, Laurence, and Stansby (2008)], but are not considered here.

Particle position is updated at each temporal iteration by the following integration:

$$\frac{\mathrm{d}\mathbf{r}}{\mathrm{d}t} = \mathbf{u} \tag{3}$$

Eqs. 1, 2 and 3 are discretised explicitly in time and SPH approach is used to perform spatial discretisation.

3 The SPH method

The key idea of SPH is that any flow property A can be expressed in any point of the fluid domain, localised by $\mathbf{r}$, by a convolution product with the Dirac distribution δ, following the interpolation formula (4), as:

$$A(\mathbf{r}) = \int_{\Omega} A(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}')d\mathbf{r}' \tag{4}$$

where Ω is the whole fluid domain.

For numerical reasons, the Dirac distribution δ is approximated by a smoothing function w_h called kernel, as:

$$A(\mathbf{r}) = \int_{\Omega} A(\mathbf{r}')w_h(\mathbf{r}-\mathbf{r}')d\mathbf{r}' + O\left(h^2\right) \tag{5}$$

The interpolating function w_h plays an important role in SPH, as it is used for the transition from the continuous form of the Navier-Stokes equations to the discrete form.

In SPH formalism, the fluid is discretized by a finite number of macroscopic fluid volumes called 'particles'. Each particle a is always characterised by a mass m_a, a density ρ_a, a pressure p_a, a velocity vector $\mathbf{u}_a$ and a position vector $\mathbf{r}_a$. Other quantities might play a role for turbulent flows, for instance. Note that w_h depends on the distance between two particles (for a spherical kernel) and of a parameter h called the smoothing length, which is function of the initial particle distribution.

The move towards a discrete set of equations is achieved by approximating the integral of Eq. (5) by a Riemann summation, as follows:

$$A(\mathbf{r}) = \sum_b \frac{m_b}{\rho_b} A_b w_h(|\mathbf{r}-\mathbf{r}_b|) + O(h^2) \tag{6}$$

where A_b denotes the value of A for particle b. The summation is now discrete and the elemental volume $d\mathbf{r}'$ (see Eq. 5) is calculated as the particle volume defined from the mass and the density as m_b/ρ_b. The sum is in theory performed over all the particles b of the domain. However, the use of kernel compact supports of radius h_t, proportional to the smoothing length h allows to reduce the number of particles b which contribute to the sum in Eq. 6 and, thus to reduce the computational time. Consequently, only particles b located in the sphere (a circle in 2-D) of radius h_t and centered in a contribute to the evaluation of the function A relative to particle a (see Fig. 1 (left)). Kernel general expressions are given in [Monaghan (1992), Morris, Fox, and Zhu (1997), Vignjevic, Vuyst, and Gourma (2001)] as well as kernel improvement regarding consistency in [Vignjevic, Reveles, and Campbell (2006)]. In most SPH codes, spline kernels are used. A fourth order spline kernel is used in this work, represented in Fig. 1 (right) in 2-D for simplicity.

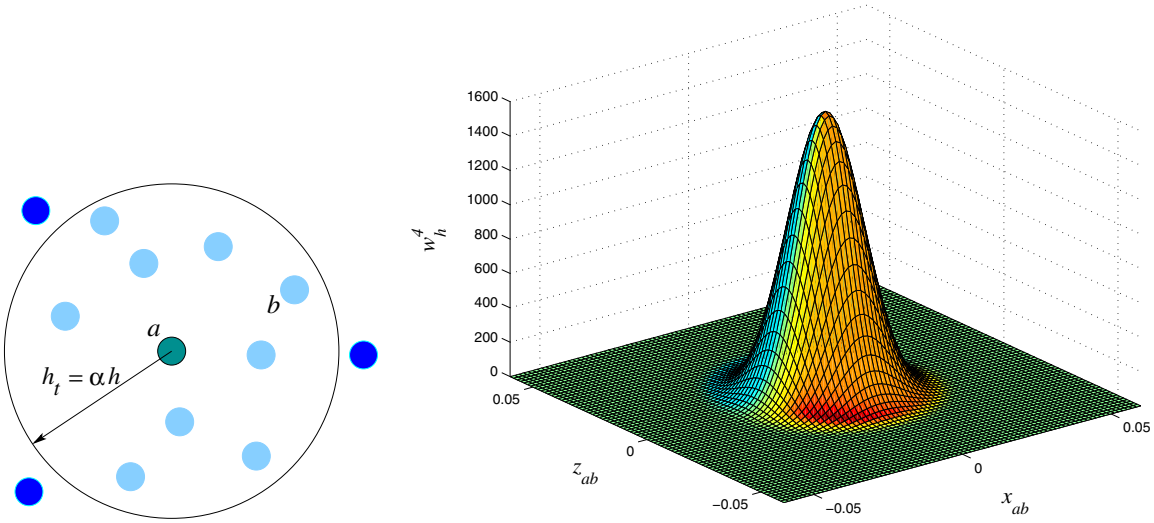

Figure 1: Left: 2-D neighbours of particle a contained in a kernel compact support. Right: 2-D plot of the fourth order spline kernel.

The spatial discretisation of the momentum and continuity equations is presented hereafter.

An SPH form of the continuity equation reads:

$$\frac{\mathrm{d}\rho_a}{\mathrm{d}t} = \sum_b m_b \mathbf{u}_{ab} \cdot \nabla_a w_{ab} \tag{7}$$

where $\mathbf{u}_{ab} = \mathbf{u}_a - \mathbf{u}_b$, $r_{ab} = |\mathbf{r}_{ab}| = |\mathbf{r}_a - \mathbf{r}_b|$ and $w_{ab} = w_h(r_{ab})$. $\nabla_a w_{ab}$ is the gradient of the kernel with respect to a. $\mathrm{d}/\mathrm{d}t$ is a Lagrangian derivative obtained by following particle motion. The SPH momentum equation reads

$$\frac{\mathrm{d}\mathbf{u}_a}{\mathrm{d}t} = -\sum_b m_b \left(\frac{p_a + p_b}{\rho_a \rho_b} - 16 \frac{\nu}{\rho_a + \rho_b} \frac{\mathbf{u}_{ab} \cdot \mathbf{r}_{ab}}{r_{ab}^2} \right) \nabla_a w_{ab} + \mathbf{F}_a^e \tag{8}$$

where $\mathbf{F}_a^e$ is the external force applied to particle a. Considering the expression of a Gaussian kernel, it can easily be shown that the pressure gradient term written in SPH formalism corresponds to a central repulsive force between particle pairs, which behaves the same way intermolecular forces act between small fluid volumes. Note that the pressure gradient expression selected here is anti-symmetric with respect to a and b subscripts as well as the viscous term. This ensures momentum conservation and allows to reduce computational cost, because the contribution of particle b can be always be deduced from the contribution of particle a, when calculating the interaction between particles a and b.

4 Numerics

4.1 Brief presentation of Spartacus-3D code

The Spartacus-2D code has been developed since 1999 at EDF R&D mainly for coastal and environmental applications such as spillways, dam breaking, breaking waves [Issa and Violeau (2007)]. Based on this 2-D version, a 3-D one has been developped [Marongiu (2007)]. Eqs 1-3 are solved, based on Eqs 7-8 and 3.

4.2 Conditions on the time step

Equations are integrated in time by the classical first order Euler explicit scheme with a variable time step δt calculated as:

$$\delta t = \min\left(\delta t_{CFL}, \delta t_{forces}, \delta t_{visc}\right). \tag{9}$$

The CFL condition δt_{CFL} reads:

$$\delta t_{CFL} = 0.4 \frac{h}{c_0} \tag{10}$$

and imposes the time step δt_{CFL} to be less than or equal to the convection time on the length h relative to the spatial discretisation.

The condition on the forces δt_{forces} reads:

$$\delta t_{forces} = 0.25 \min_a \sqrt{\frac{h}{|\mathbf{f}_a|}} \tag{11}$$

and ensures that particles do not get too close to each other during the integration of their movement [Morris, Fox, and Zhu (1997)]. $\mathbf{f_a}$ denotes the internal and external forces associated to particle a (i.e. the magnitude of the r.h.s. of Eq. 8).

The condition on the viscous forces δt_{visc} reads:

$$\delta t_{visc} = 0.125 \frac{h^2}{\nu} \tag{12}$$

This viscous criterion must be taken into account to make the time step inferior to the viscous phenomenon time scale [Morris, Fox, and Zhu (1997)].

4.3 Link list

Eqs 7 and 8 show that SPH operators are expressed as differences or sums of contribution of particles a and b, with a sum on b. Due to the particle motion, the search for particles b has to be performed at each temporal iteration. The CPU time would scale as $NPART^{5/2}$ (in 3-D) where $NPART$ denotes the particle total number, if the search would be carried out over the whole set of particles. Since spline kernels have a compact support, each particle a is only linked to its closest neighbours b for which $r_{ab} < h_t$. It is thus important to optimise the construction of the link list relative to particle connections at each temporal iteration: for each particle a, all its closest neighbours b are detected according to the following algorithm:

1. The whole fluid domain is embedded in a coarse homogeneous Cartesian grid which cell size is h_t, where h_t is the kernel compact support size. This grid is used to reduce the computational cost while looking for the particles interacting with particle a (see Fig. 2).

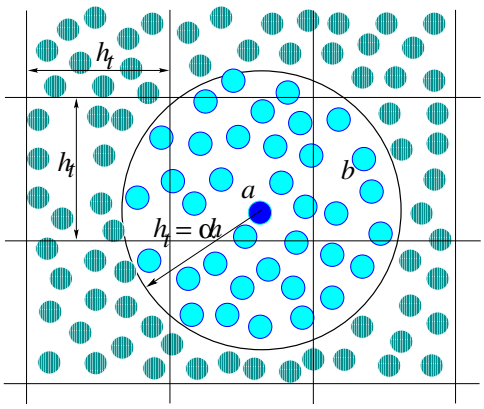

Figure 2: Localisation of particle a neighbours involved in the calculation of SPH operators (2-D view).

2. The index of the cube a_{cube} where particle a is located is stored.

3. The coarse grid is built in such a way that for each particle a (cube size equal to h_t), neighbouring particles referenced by b are located in one of the $3^3 - 1 = 26$ neighbouring cubes of a_{cube}, referenced by b_{cube}. The search for particle a links is then restricted to a search within 27 cubes and only particles which satisfy $r_{ab} < h_t$ are considered as neighbours of particle a and their index is stored.

4.4 Periodic conditions

Special treatment is required when periodic boundary conditions apply. For simplicity of the presentation, they are set in the streamwise direction x, the domain going from x_{min} to x_{max} (see Fig. 3). Those conditions apply in order to avoid truncated compact supports (*truncated spheres*) compared to compact supports for particles located in the inner domain (*entire spheres*), those particles being located at a distance larger than h_t from boundaries. It is assumed that particles a for which $x_{min} \leq x_a \leq x_{min} + h_t$ (respectively $x_{max} - h_t \leq x_a \leq x_{max}$) which naturaly interact with particles b such that $x_{min} + h_t \leq x_b \leq x_{min} + 2h_t$ (respectively $x_{max} - 2h_t \leq x_b \leq x_{max} + h_t$), also interact with particles b such that $x_{max} - h_t \leq x_b \leq x_{max}$ (respectively $x_{min} \leq x_b \leq x_{min} + h_t$), as if they would be shifted by $x_b + (x_{max} - x_{min})$ (respec-

tively $x_b - (x_{max} - x_{min})$), all their characteritics but their abscissae being conserved. The cubes adjacent to particle a are therefore completed by those located in the vicinity of the opposite boundary. Consequently, the code considers that particles b located in the vicinity of x_{min} (see Fig. 3) are copied and translated to $x_b - (x_{max} - x_{min})$ and particles b then become a neighbour of particle a. The same process applies for particles located in the vicinity of the boundary defined by x_{max}.

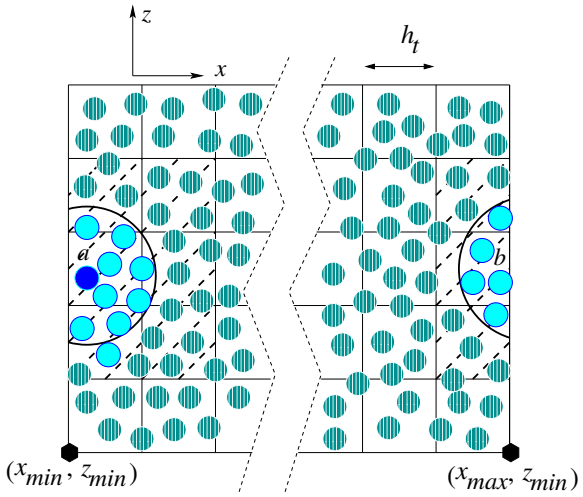

Figure 3: Adjacent cells relative to a periodic flow with respect to x-direction in 2-D. They are highlighted by the black dashed strips.

4.5 Wall modelling

Walls are modelled by solid particles called 'wall particles'. Moreover, in order to ensure the impermeability of the wall, three layers of so-called 'fictitious particles' are added under each wall. They are built by extrusion of the wall particles. The density of wall and fictitious particles is calculated in function of the contribution of the fluid particles through the continuity equation (7). When the fluid particle a is linked to the fictitious particle b, the contribution of particle b to the evolution of the density of particle a is equal to (see Eq. 7)

$$m_b \mathbf{u}_{ab} \cdot \nabla_a w_{ab} \tag{13}$$

As particle b is also linked to fluid particle a, the contribution of particle a to the evolution of density of particle b is also given by Eq. 13, since the continuity equation (Eq. 7) is symmetric with respect to a and b subscripts. The pressure of wall and fictitious particles are then computed by the state equation and these particles are involved in the pressure gradient relative to fluid particles in the equation of motion. These wall conditions also enable a perfect impermeability of the wall in rapid dynamic conditions such as dam breaking. Contrary to the repulsive forces commonly used in SPH to represent walls [Monaghan (1995)], the present formulation does not introduce any *ad hoc* coefficient. Moreover, in contrast to traditional mirror particles used in most SPH codes, the method proposed here considers fixed particles and can easily be implemented even for non flat walls. It is also suitable for modelling moving walls, following what is done in 2-D in [Issa and Violeau (2007)].

5 Parallelisation

Parallelisation is fundamental to enable SPH codes to compute 3-D flows efficiently in terms of CPU time. A profiling of the serial version of Spartacus-3D shows that the search for particle a's links (see Paragraph 4.3) is the most expensive part of a temporal iteration, consumming up to 50% of the CPU time. While running parallel instead of serial, another part could take a lot of time, which consists in rebuilding the list of particles by re-numbering them at each temporal iteration, in order to minimise the communications between processors while searching the links. This re-numbering is compulsory because fluid particles move at each temporal iteration.

In the parallel version of Spartacus-3D, three main steps are then carried out per temporal iteration. At the $(n+1)^{th}$ iteration, for instance it yields (see Fig. 4):

- Step 1: Generation of the new particle list,
- Step 2: Search for particle links,
- Step 3: Resolution of the equations.

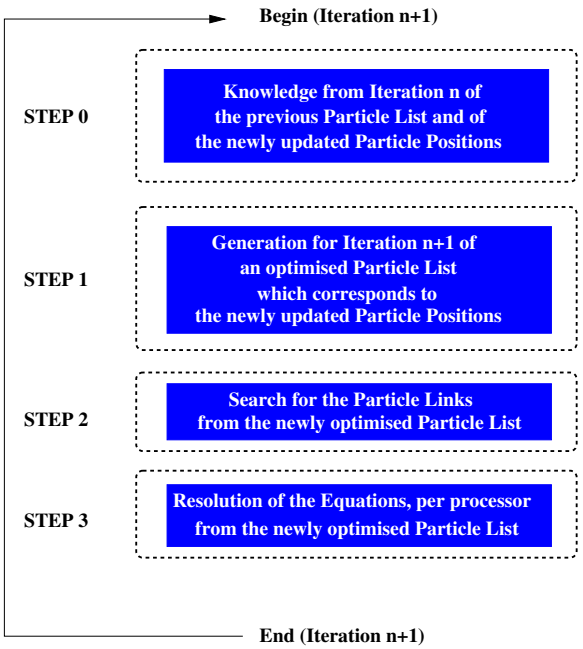

Figure 4: Simplified parallel algorithm.

Step 1, 2 and 3 are detailed in the following. Note that Spartacus-3D communications are based on MPI library and are mainly global communications.

5.1 Step 1. List of particles per processor

Assuming that the total number of processors is $NPROCS$ and that the total number of particles is $NPART$, the general idea is to allocate $NPARTPROC \approx NPART/NPROCS$ particles a per processor to ensure node balancing, while reducing the number of processors containing the neighbours b of the $NPARTPROC$ particles a.

- The Cartesian grid is generated as explained in Paragraph 4.3 (see Fig. 5) and the cubes denoted by C_i ($i = 1, NCUBES$) are linerarly ordered in an array of dimension $NCUBES$, where $NCUBES$ is the whole number of cubes. The loop is carried out over $NCUBES$ in this operation.

- Denoting the number of particles per cube C_i as NPC_i, a temporary array $DISPTB_i$ is built from the NPC_i's as the accumulated number of particles contained in the cubes already listed until cube of index 'i', *i.e.* $DISPTB_i = \sum_{j=1}^{i} NPC_j$ (see Fig. 6). The loop is carried out over $NCUBES$.

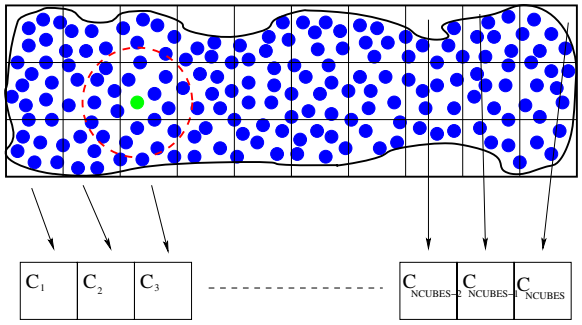

Figure 5: Coarse Cartesian grid embedding a fluid domain in 2-D. The cubes are linearly ordered.

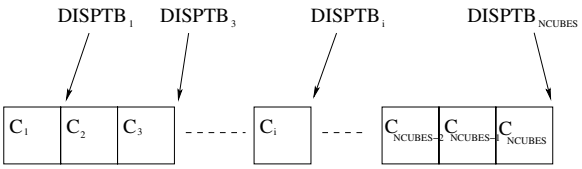

Figure 6: Construction of the temporary array $DISPTB_i$.

- $NPROCS$ blocks BLO_l are built from $DISPTB_i$, with the condition of having approximately $NPARTPROC \approx NPART/NPROCS$ particles per block (and not per processor, as the notion of distribution per processor does not exist at iteration $n+1$ yet). The loop is over $NCUBES$.

- This step consists of detecting which particles (which index is defined in the list corresponding to iteration n) located on a given processor $PROC_k$ belong to a given block BLO_l as defined in the previous item. At the end of this operation, each processor has access to the number of its particles located on block BLO_l, and to the local list of its own particles per block. Both arrays are not complete as the loop is performed over the particles present on each processor $PROC_k$ at iteration n, and not over the total number of particles $NPART$.

- Communications are now required to build up the complete local list per processors. This loop is over all the blocks, *i.e NPROCS*.
- The complete local list assembled on the master node is then broadcast to the other processors.
- The global list is built up from the local list. The loop is over the new number of particles per processor, *i.e* the number of particles per processor at iteration $n+1$. The number of particles per processor is also known at iteration $n+1$ (see Fig. 7).

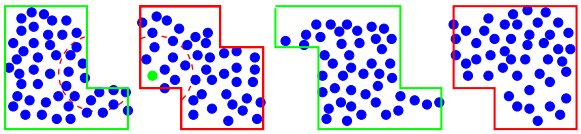

Figure 7: Particle distribution on 4 processors.

5.2 Step 2. Link generation

Links between particles a and their neighbouring particles b used in the calculation of the operators are established by processors, taking into account the cube structure. The first loop goes over particle a present on processor $PROC_k$ and the second one on the neighbouring cubes of the cube containing particles a.

5.3 Step 3. Resolution

As in the serial version of the code, the equations are solved explicitly. However, the operators are built per processors in the parallel version, with loops going over the list of particles on $PROC_k$ at iteration $n+1$, and not over the total number of particles.

5.4 Parallelisation validation

The test case of a 3-D dam breaking described in 2-D in [Violeau and Issa (2007)] and extruded in the third direction is simulated in order to validate the parallelisation procedure. 6 particles used as probes are tracked in time during $1,000$ temporal iterations. Spartacus-3D is first ran serial and then parallel. Axial velocity, density and number of neighbours relative to each tracked particle are compared in both configuration. The evolution in time of the difference between serial and parallel on those quantities is of the order of machine precision and is not plotted here. Parallel and scalar versions of Spartacus-3D hence give the same results. Figure 8 shows particle distribution on 4 processors in a (x,z) section of the dam breaking case. The repartition is done by layers.

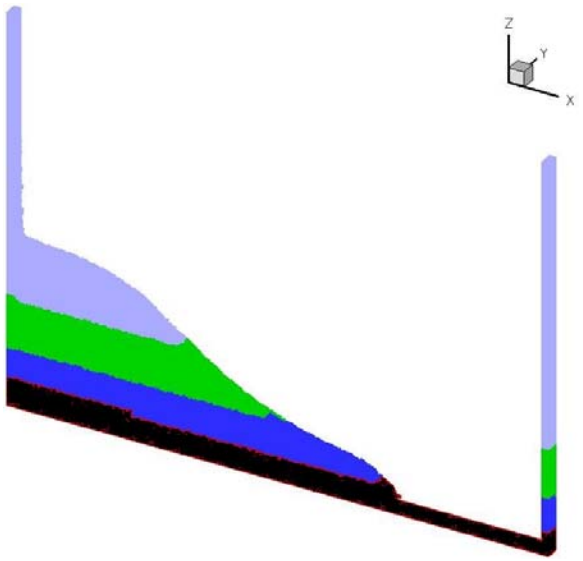

Figure 8: Distribution of the particles per processors (4 processors in total).

The speed up (defined as the ratio between serial and parallel CPU times) is plotted after running several computations with various number of processors, going from 1 to 16, on one of EDF R&D clusters (IBM, 64 nodes of 2 processors each. 4 Gb and 2.6 GHz Mono-Core processors). Table 1 shows CPU time and speed-up in function of the number of processors used.

The speed-up of the code is satisfactory up to 8 processors with this amount of particles (see Fig. 9). With 16 processors, the speed-up is not optimal anymore, probably due to the low particle number of particles allocated per processor. Further tests will be carried out with $2,000,000$ particles for the same geometry.

Table 1: CPU time with respect to the number of processors used.

Number of processors	CPU time (s)	Speed-Up
1	4,332	1.00
2	2,160	2.01
4	1,116	3.88
8	576	7.52
16	322	13.45

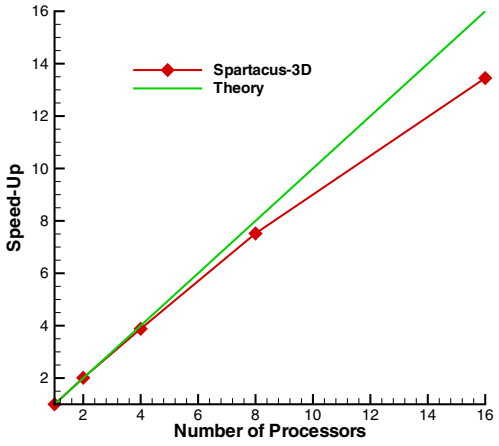

Figure 9: Spartacus-3D speed up.

6 Laminar flow in a 3-D straight closed channel

This Poiseuille flow case is computed to investigate the accuracy of Spartacus-3D in a case where an analytical solution is available.

6.1 Geometry of the system

The geometry consists of a 3-D channel. The system is represented in Fig. 10.

6.2 System modelling

A laminar flow characterised by a Reynolds number of 200 based on the mean bulk velocity of 1.0 $m.s^{-1}$ and on the channel height $H = 2m$ is computed. The system discretization is described in Table 2. The initial distance between particles is $\delta r = 20.83\ mm$.

Particles are initially distributed on a regular lattice with zero velocity. The fictitious particle velocities do not evolve during the calculation but they contribute to the calculation of the fluid particle viscous term. Periodic conditions are applied with respect to $x-$ and $y-$direction. The gravity is neglected and the fluid is driven by an external horizontal force $\mathbf{F^e}$ updated at each temporal iteration in order to impose the correct mass flow rate [Issa, Lee, Violeau, and Laurence (2005)]. The smoothing length value $h/\delta r = 1.2$ has been determined through numerical tests. 100,000 iterations are carried out, which means that a particle moving with a velocity equal to the bulk velocity has been transported on a distance equal to 48 times the geometry length.

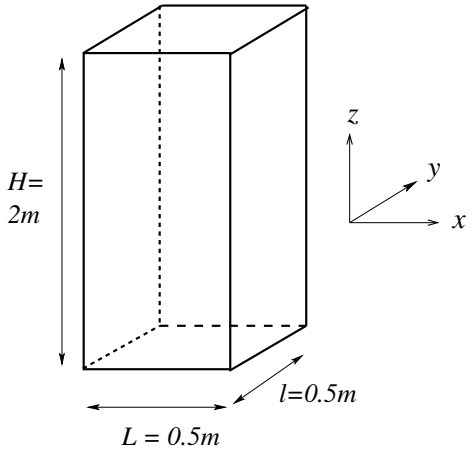

Figure 10: Geometry of the 3-D channel.

Table 2: Fluid, wall and fictitious particle discretisations for the 3-D channel flow.

Fluid particles	59,375
Wall particles	1,250
Fictitious particles	5,000
Total particle number	65,625
Particle initial spacing (mm)	20.83

6.3 Results

The simulation has been carried out in 28 CPU hrs on 8 processors. Streamwise velocity profiles are plotted against y-direction. The agreement with the analytical solution is very good.

Spartacus-3D hence shows its ability to produce accurate results for this academic flow.

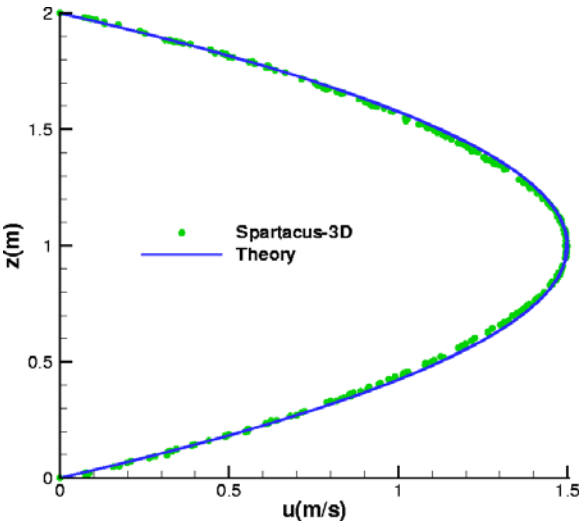

Figure 11: Comparison of Spartacus-3D velocity profile with the analytical Poiseuille parabolic profile.

7 Laminar flow in a 3-D hill channel

This test case is simulated to show the ability of Spartacus-3D to simulate recirculating flows. The validation is carried out comparing SPH results with results obtained by an Eulerian finite volume based software, *Code_Saturne* [Archambeau, Méchitoua, and Sakiz (2004)].

7.1 Geometry of the system

The geometry consists of a 2-D hill channel extruded in the third direction (y). It has been defined in ERCOFTAC workshops [Mellen, Froelich, and Rodi (2000), Uribe and Laurence (2000)] to run Large-Eddy Simulations. The system is represented in Fig. 12 where H, which is the hill height is equal to 28 mm.

7.2 System modelling

A laminar flow characterised by a Reynolds number of 50 based on a mean bulk velocity of $1,785.10^{-3}\ m.s^{-1}$ for water is computed. The system discretization is given in Table 3.

Particles are initially distributed on a regular lattice with zero velocity. Periodic conditions are once again applied with respect to $x-$ and $y-$direction and an external force drives the flow.

Table 3: Fluid, wall and fictitious particle discretisations for the 3-D hill flow.

Fluid particles	315,008
Wall particles	8,096
Fictitious particles	12,384
Total particle number	355,488
Particle initial spacing (mm)	1

The gravity is neglected. A smoothing length value $h/\delta r = 1.2$ has been determined through numerical tests. 50,000 iterations are carried out, which means that a particle moving with a velocity equal to the bulk velocity has been transported on a distance equal to 5 times the channel length.

7.3 Results

The simulation has been carried out in 48 CPU hrs on 16 processors. Some of the probes choosen in an ERCOFTAC workshop [Uribe and Laurence (2000)] are used for comparison and represented in Fig. 13. Since no analytical solution exists for this problem, the results obtained with Spartacus-3D are compared to those obtained by the Eulerian software *Code_Saturne* [Archambeau, Méchitoua, and Sakiz (2004)]. Figure 13 shows that axial velocity profiles obtained by Spartacus-3D are in excellent agreeement with those of *Code_Saturne*. Moreover, no spanwise averaging is performed to smooth SPH data at the probe locations, which shows that SPH solution is homogeneous enough in the spanwise y-direction.

Spartacus-3D is then able to compute recirculating flows.

8 3-D dam breaking flow impinging on an obstacle

The dam breaking case is very popular for validation, because it shows strong free surface deformation although its set-up is easy: no special in- or outflow conditions are needed, for instance. It is chosen as well to demonstrate the ability of Spartacus-3D to compute 3-D free surface flows. Several experiments have been performed for dam breaking flows by the MAritime Research Institute Netherlands (MARIN). One of them, used in

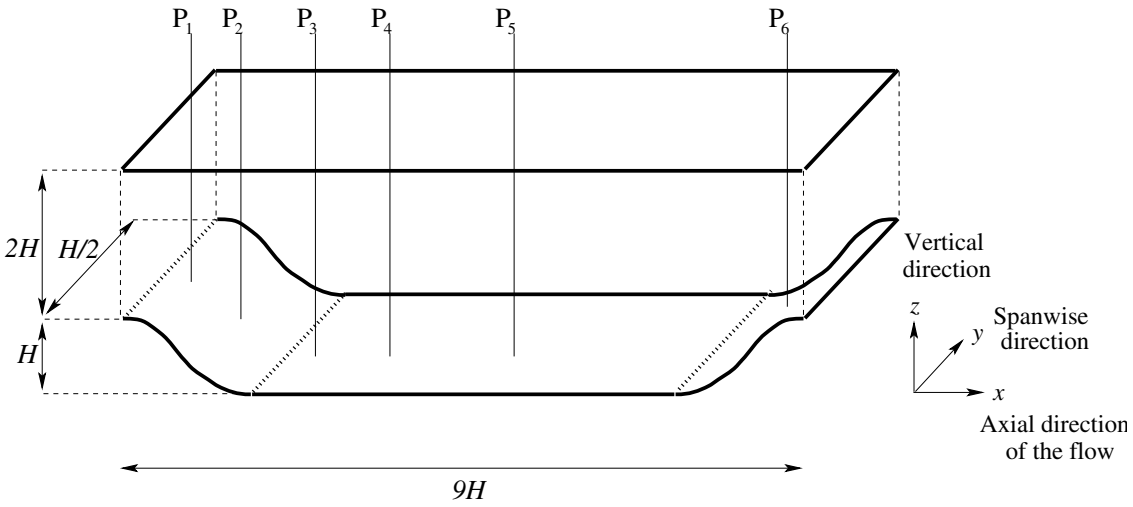

Figure 12: Geometry of the 3-D hill channel.

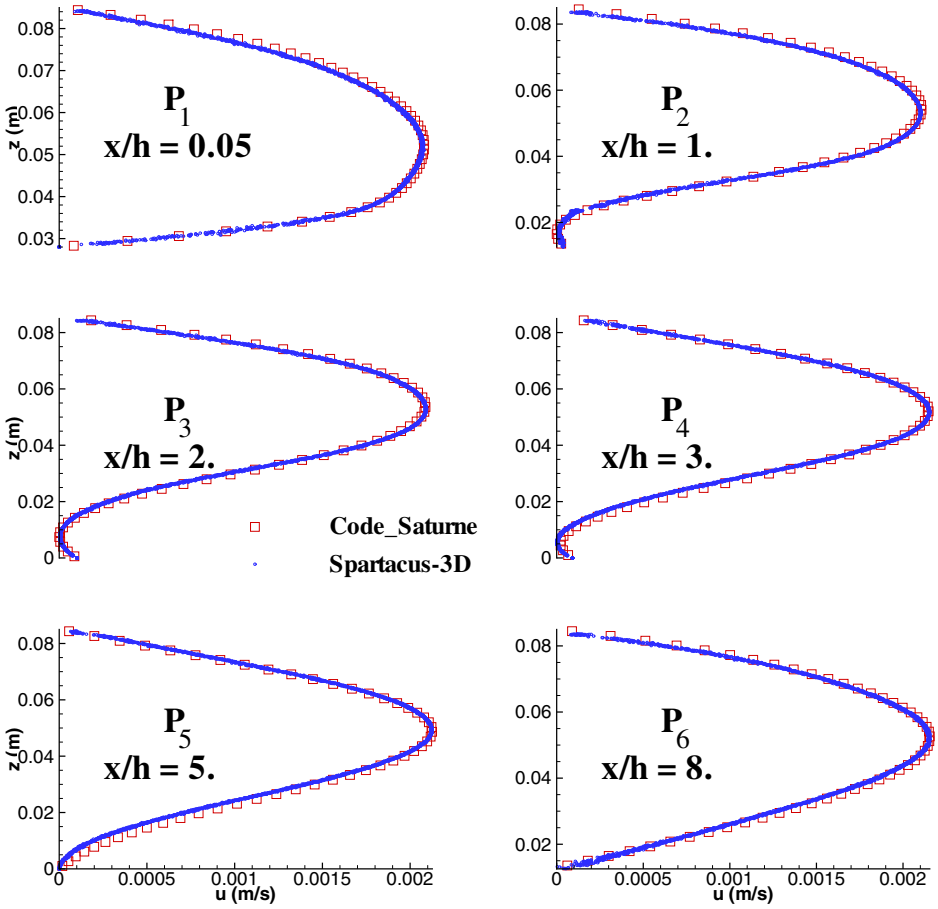

Figure 13: Comparison of Spartacus-3D results with data obtained from the Eulerian code, Code_Saturne.

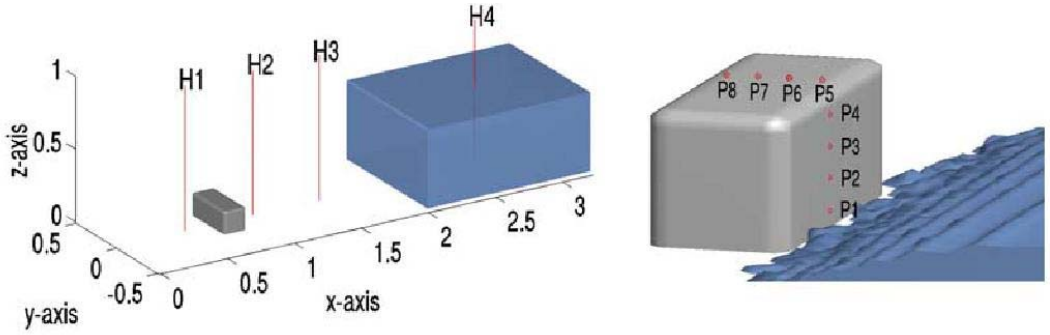

Figure 14: Geometry of the dambreak. Left: measurement positions for water heights in the dambreak experiment [Kleefsman, Fekken, Veldman, Iwanowski, and Buchner (2005)]. Right: zoom of the box.

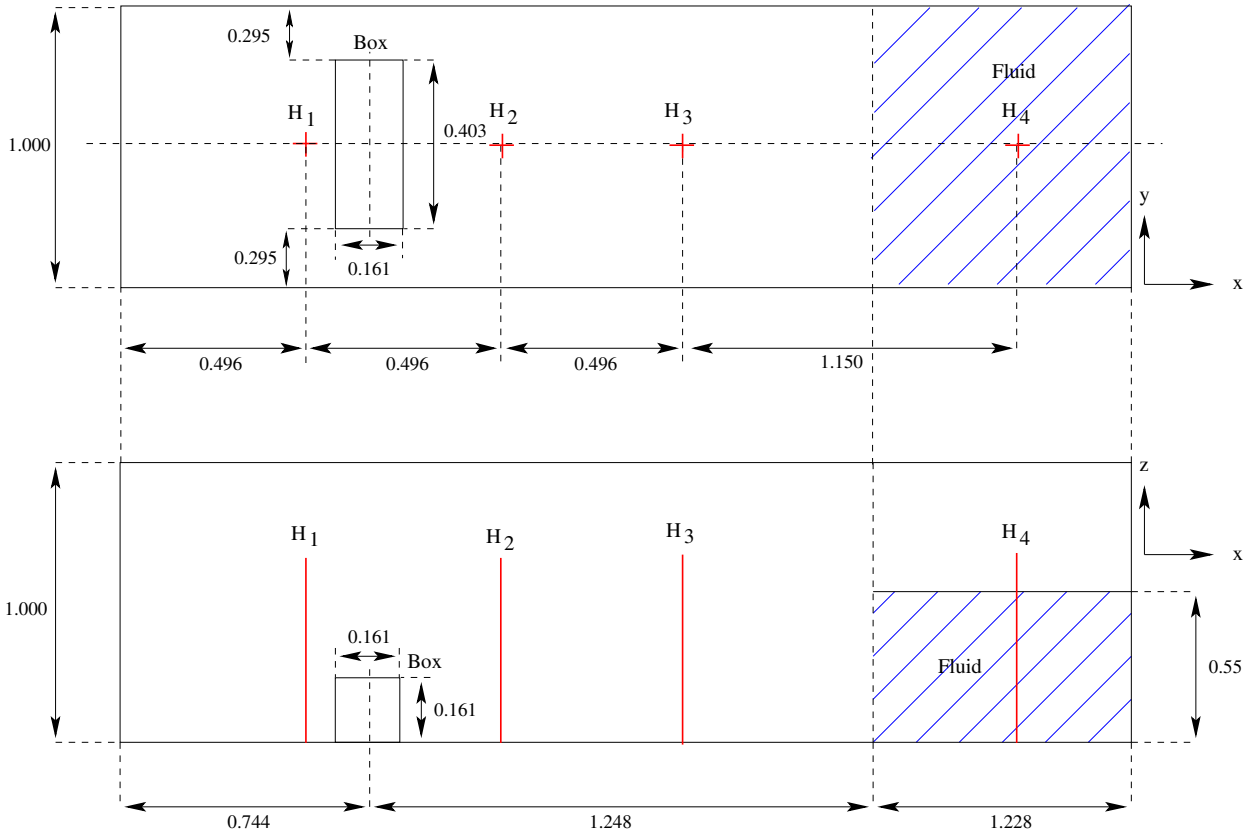

Figure 15: General description of the system. Top: top view. Bottom: side view.

[Kleefsman, Fekken, Veldman, Iwanowski, and Buchner (2005)], consists of a large tank of $3.22 \times 1 \times 1$ m with an open roof. The right part (see Fig. 14 for $2\ m \leq x \leq 3\ m$) of the tank is locked by a door. 0.55 m of water is waiting behind the door to flow into the tank when the door is opened. This is done by releasing a weight, which almost instantaneously pulls the door up. A box has been put in the tank. During the experiment, measurements on water heights have been performed. As shown in Fig. 14, four vertical height probes have been used: one in the reservoir and the three others in the tank.

8.1 Geometry of the system

The geometry of the system is described in Fig. 15 where H_i correspond to the vertical water height probes used in the experiment. All dimensions are in meters.

8.2 System modelling

Table 4 recalls the system discretisation.

Table 4: System discretization (fluid and walls) for the 3-D dam breaking simulation.

Fluid particles (x,y,z)	$67 \times 54 \times 30$
Bottom wall particles (x,y)	183×62
Side wall particles (x,z)	183×55
Side wall particles (y,z)	54×55
Particle initial spacing: δr (cm)	55/30

The box is also discretized by wall particles and filled by fictitious particles (see Table 5).

Table 5: Box discretisation for the dambreaking simulation.

Top wall particles (x,y)	8×21
Side wall particles (x,z)	10×9
Side wall particles (y,z)	21×9

The numbers of particles used for this test case are gathered in Table 6.

Table 6: General particle discretization.

Fluid particles	108,540
Wall particles	38,142
Fictitous particles	113,592
Total	260,274

A 3-D view of the discretized system is given in Fig. 16 (in order to facilitate visualization, some wall and fictitious particles are not plotted on Fig. 16).

8.3 Results

In order to simulate a physical time of 6 s to compare with the experiments, 30,000 iterations have been run, which requires 32 hours of CPU time on 16 processors of the previously mentionned cluster. Figs. 17-19 qualitatively compare Spartacus-3D results to experimental snapshots at three physical times t, *i.e* $t = 0.32$ s, $t = 0.64$ s

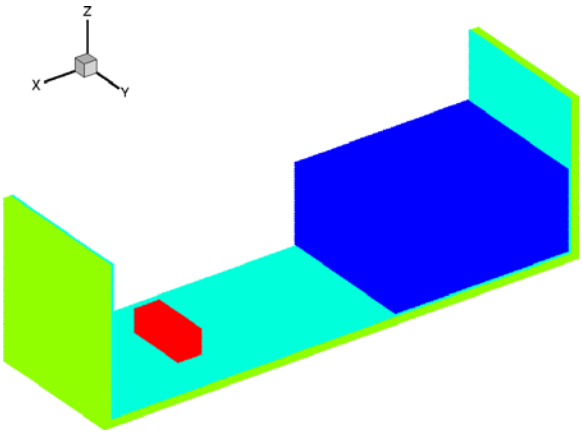

Figure 16: 3-D view of discretized system.

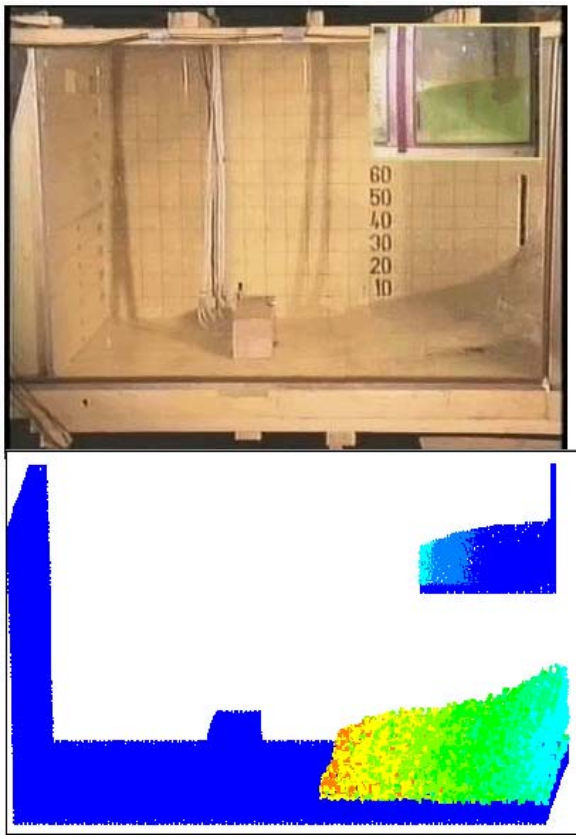

Figure 17: Comparison between experiment (top) and Spartacus-3D (bottom) at 0.32 s.

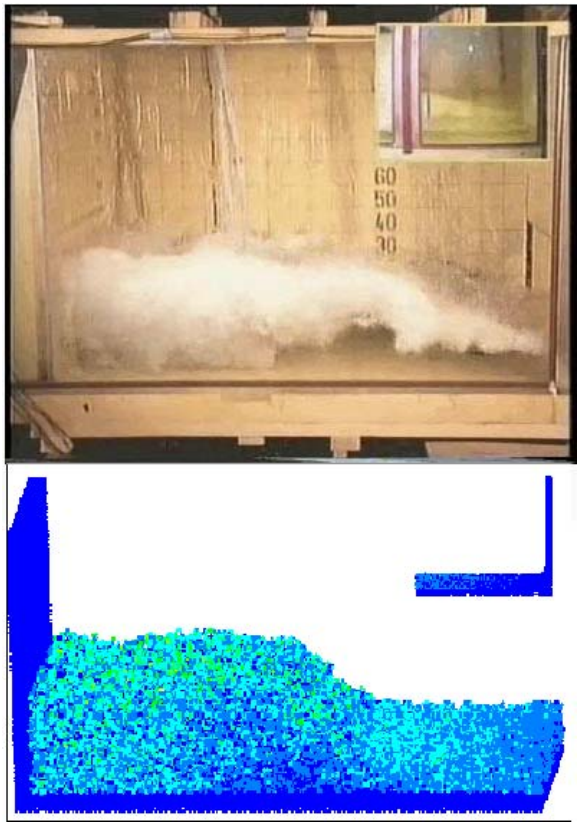

Figure 18: Comparison between experiment (top) and Spartacus-3D (bottom) at 0.64 s.

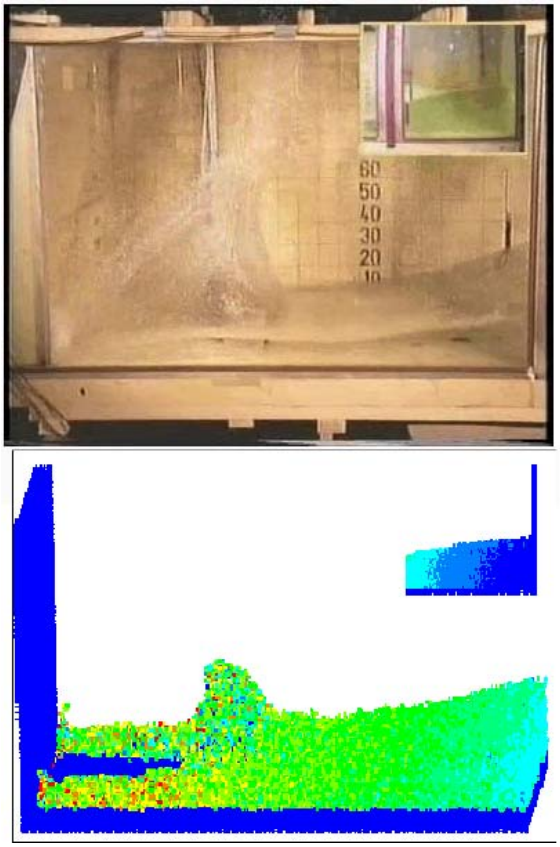

Figure 19: Comparison between experiment (top) and Spartacus-3D (bottom) at 2.0 s.

and $t = 2$ s. Spartacus-3D reproduces quite well the global motion of the fluid.

As shown in Fig. 20, the water elevation simulated by Spartacus-3D is very close to experimental results until 2.5 s. This part corresponds to the collapse of the water column. Between 2.5 s and 6 s, Spartacus-3D results are similar to experimental ones but are shifted of approximately 0.5 s in time. The water level increase between 2.5 and 3 s is directly linked to the return of the flow after impingement on the front wall. It hence seems that the friction at the wall is overestimated by Spartacus-3D. This could be linked to the spatial discretization and will be investigate more carefully in the future.

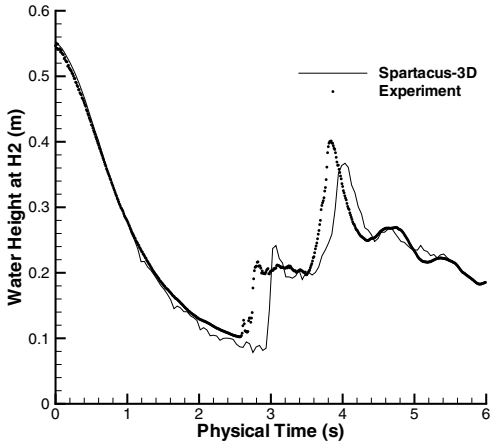

Figure 20: Comparison of water elevation at H_4 versus time.

9 Final Remarks-Perspectives

Parallel 3-D SPH software has been developed and has shown its ability to produce accurate results compared to analytical solution (Poiseuille flow), to compute recirculating flows (3-D peri-

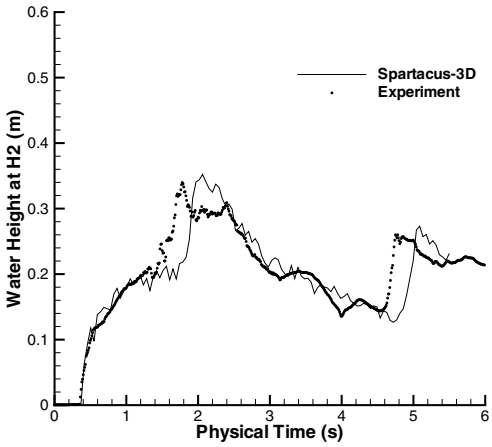

Figure 21: Comparison of water elevation at H_2 versus time.

odic hill) and to simulate a complex 3-D free surface flow (dam breaking impacting an obstacle). The performance in terms of parallelisation is currently good up to 16 processors.

To improve parallelisation performance and to be able to get a good scaling while running on more than 16 processors, more local communications are required. This is now under investigation. This work done in collaboration with IBM should lead to the development of an efficient massively parallelised code able to perform simulations on thousands of processors.

Acknowledgement: The authors would like to thank Theresa Helmolt-Kleefsman from MARIN institute (NL) for providing them with dam breaking data.

References

Archambeau, F.; Méchitoua, N.; Sakiz, M. (2004): Code_Saturne: a finite volume code for the computation of turbulent incompressible flows - Industrial applications. *International Journal on Finite Volumes*, vol. 1, pp. 1–62.

Cummins, S. J.; Rudman, M. (1999): An SPH projection method. *J. Comp. Phys.*, vol. 152, pp. 584–607.

Issa, R.; Lee, E. S.; Violeau, D.; Laurence, D. (2005): Incompressible separated flows simulations with the Smoothed Particle Hydrodynamics gridless method. *International Journal for Numerical Methods in Fluids*, vol. 47, pp. 1101–1106.

Issa, R.; Violeau, D. (2007): Modelling a plunging breaking solitary wave with eddy-viscosity turbulent SPH midels. *CMC: Computers, Materials and Continua*, pg. To be published.

Kleefsman, K. M. T.; Fekken, G.; Veldman, A. E. P.; Iwanowski, B.; Buchner, B. (2005): A volume-of-fluid based simulation method for wave impact problems. *J. Comp. Phys.*, vol. 206, pp. 363–393.

Lee, E.; Moulinec, C.; Xui, R.; Violeau, D.; Laurence, D.; Stansby, P. (2008): Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. *JCP*, vol. Second Review.

Marongiu, J. (2007): *Parallel 3-D developments involving the Smoothed Particle hydrodynamics method.* PhD thesis, PhD Thesis of ECL, 2007.

Mellen, C.; Froelich, J.; Rodi, W. (2000): Large Eddy Simulation of the flow over periodic hills. In *16th IMACS World Congress, Switzerland*.

Monaghan, J. J. (1992): Smoothed Particle Hydrodynamics. *Annu. Rev. Astron. Astrophy.*, vol. 30, pp. 543–574.

Monaghan, J. J. (1995): Simulating gravity currents with SPH III Boundary forces. *Mathematics Reports and Preprints*, vol. 95/5.

Morris, J. P.; Fox, P. J.; Zhu, Y. (1997): Modelling low Reynolds Number Incompressible Flows Using SPH. *J. Comp. Phys.*, vol. 136, pp. 214–226.

Uribe, J.; Laurence, D. (2000): 10th Ercoftac/IAHR Workshop on Refined Turbulence Modelling. In *10th joint ERCOFTAC (SIG-15)/IAHR/QNET-CFD Workshop on Refined Turbulence Modelling*.

Vignjevic, R.; Reveles, J. R.; Campbell, J. (2006): SPH in a Total Lagrangian Formalism. *CMES: Computer Modeling in Engineering and Sciences*, vol. 14, pp. 181–198.

Vignjevic, R.; Vuyst, T. D.; Gourma, M. (2001): On interpolation in SPH. *CMES: Computer Modeling in Engineering and Sciences*, vol. 2, pp. 319–336.

Violeau, D.; Issa, R. (2007): Numerical modelling of complex turbulent free-surface flows with the sph method: an overview. *International Journal For Numerical Methods in Fluids*, vol. 53, pp. 277–304.