JOSEPHINE: A parallel SPH code for free-surface flows

J.M. Cherfils, G. Pinon, E. Rivoalen

Computer Physics Communications

JOSEPHINE: A parallel SPH code for free-surface flows ☆

J.M. Cherfils [a,*], G. Pinon [a], E. Rivoalen [a,b]

[a] Laboratoire Ondes & Milieux Complexes UMR 6294, CNRS-Université du Havre, 25, rue Philippe Lebon, 76058 Le Havre, France
[b] Laboratoire d'Optimisation et Fiabilité en Mécanique des Structures, EA 3828, INSA de Rouen, Avenue de l'Université, BP 08, 76801 St Etienne du Rouvray, France

ARTICLE INFO

Article history:
Received 6 July 2011
Received in revised form 10 January 2012
Accepted 2 February 2012
Available online 9 February 2012

Keywords:
SPH
Weakly compressible
Parallel
Free-surface
Euler equations

ABSTRACT

JOSEPHINE is a parallel Smoothed Particle Hydrodynamics program, designed to solve unsteady free-surface flows. The adopted numerical scheme is efficient and has been validated on a first case, where a liquid drop is stretched over the time. Boundary conditions can also be modelled, as it is demonstrated in a second case: the collapse of a water column. Results show good agreement with both reference numerical solutions and experiments. The use of parallelism allows significant reduction of the computational time, even more with large number of particles. *JOSEPHINE* has been written so that any untrained developers can handle it easily and implement new features.

Program summary

Program title: *JOSEPHINE*
Catalogue identifier: AELV_v1_0
Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AELV_v1_0.html
Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland
Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html
No. of lines in distributed program, including test data, etc.: 5139
No. of bytes in distributed program, including test data, etc.: 22 833
Distribution format: tar.gz
Programming language: Fortran 90 and OpenMPI
Computer: All shared or distributed memory parallel processors, tested on a Xeon W3520, 2.67 GHz.
Operating system: Any system with a Fortran 90 compiler and MPI, tested on Debian Linux.
Has the code been vectorised or parallelised?: The code has been parallelised but has not been explicitly vectorised.
RAM: Dependent upon the number of particles.
Classification: 4.12
Nature of problem: *JOSEPHINE* is designed to solve unsteady incompressible flows with a free-surface and large deformations.
Solution method: *JOSEPHINE* is an implementation of Smoothed Particle Hydrodynamics. SPH is a Lagrangian mesh free particle method, thus, no explicit tracking procedure is required to catch the free surface. Incompressibility is satisfied using a weakly compressible model. Boundary conditions at walls are enforced by means of the ghost particles technique. The free-surface dynamic and kinematic conditions are applied implicitly.
Running time: 15 mn on 4 processors for the dam-break case with 5000 particles, dependent upon the real duration (2 s here).

0. Introduction

Most of the numerical methods (Boundary Element Method, Volume of Fluid) experience some difficulties in modelling large free-surface deformations. Since the last few years, Smoothed Particle Hydrodynamics (SPH) has been a growing interest thanks to its ability to easily model violent free-surface flows with strong interface motion. Since it is a Lagrangian particle method, it is well

suited to simulate several phases with fewer difficulties than with other methods. Indeed, no interface tracking algorithm is required to see the interface evolution.

Some SPH codes are already available to compute free-surface flows. A serial Fortran 77 program is provided with the book by Liu and Liu [16], which does not include some of the latest improvements (described hereinafter). The SPHysics group proposes a set of free codes (serial, parallel, GPU or coupled with a shallow water model) that includes many of the newest features, developed by the SPH community (visit the following link for more information: wiki.manchester.ac.uk/sphysics).

In this paper, the new program *JOSEPHINE* is described, and it is based on an alternate SPH formulation than in SPHysics. The Euler equations are written in discrete form, following the work by Colagrossi and Landrini [6] and Ferrari et al. [9], and in a different manner than in SPHysics, which is based upon the formulation of Monaghan [21], mainly concerning the choice of pressure gradient approximation (see Section 1.3). The complete numerical scheme is described, including a model of wall boundaries. Being the most efficient way to model simple geometries, ghost particles [15,6,25] have been preferred to the dynamic boundary particles technique [8], available in SPHysics, or to the repulsive particles available in the code by Liu and Liu [16]. *JOSEPHINE* is a parallel and optimised Fortran 90 program, designed to deal with free-surface flows in open basin [4]. It is well suited for developers who would like to start from a robust code, written in an understandable way.

First, the SPH model that has been implemented in *JOSEPHINE* is described. Then, two simulation cases are presented for validation. The deformation of an initially circular patch of fluid is simulated to show the ability of *JOSEPHINE* to solve the Euler equations, particularly at a free surface, without gravity force and boundaries. Next, *JOSEPHINE* is validated in a context where wall boundaries have to be modelled: the collapse of a water column followed by its impact on a vertical wall. This case is well known in the SPH community and a reference SPH solution [6] is used for comparison. The third part of this paper is dedicated to the parallel implementation in *JOSEPHINE*. A vertical domain decomposition has been adopted and some details are given about the load balancing strategy. Finally, the program structure is explained and some details about the compilation and the use of *JOSEPHINE* are given.

1. SPH model

The SPH method has been developed for simulations of gravitational systems in Astrophysics simultaneously by Gingold and Monaghan [10] and Lucy [18]. SPH is a Lagrangian mesh-free particle method. It is efficient for computational fluid dynamics in opened domains. The system is modelled by a set of particles which carry their own physical properties (momentum, pressure) and evolve according to chosen conservation laws. A numerical approximation of the system solution is then obtained. Monaghan [21] proposed its extension to free-surface flows, and then introduced the so-called Weakly Compressible SPH approach (WCSPH), which is used to model a nearly incompressible fluid without the need to solve any Poisson equation, as done in the ISPH approach [7]. Since, SPH has been applied to a large variety of non-linear flows: interfacial flows [6], incompressible viscous flows [24], magnetohydrodynamics [27], impacts simulations and explosions [16] (see [22] for a review), thanks to its ability to deal with large deformations of the fluid.

1.1. Integral interpolation

SPH can be seen as a way to discretise any set of equations, and is based on the integral representation of a function [29]. Then,

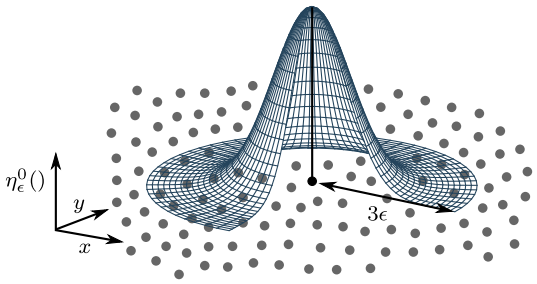

Fig. 1. SPH interpolation.

the value $\langle f(\boldsymbol{x})\rangle$ of a field $f()$ at position $\boldsymbol{x}$ can be reconstructed by convolution of the known values $f(\boldsymbol{y})$ with a smoothing kernel $\eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})$ over the domain Ω.

$$\langle f(\boldsymbol{x})\rangle = \int_\Omega f(\boldsymbol{y})\eta_\epsilon(\boldsymbol{y}-\boldsymbol{x})\,\mathrm{d}\boldsymbol{y}. \tag{1}$$

Particles, which are the computational nodes, are primarily defined by a position $\boldsymbol{x}_i$ and a volume V_i. The previous formulae can be discretised to represent the field value at the particle $\boldsymbol{x}_i$ (see Fig. 1).

$$\langle f(\boldsymbol{x}_i)\rangle = \sum_j f(\boldsymbol{x}_j)\eta_\epsilon(\boldsymbol{x}_i-\boldsymbol{x}_j)V_j. \tag{2}$$

The function $\eta_\epsilon(\boldsymbol{x})$ has to satisfy the following properties:

$$\int_\Omega \eta_\epsilon(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 1, \tag{3}$$

also called the normalisation property, and:

$$\lim_{\epsilon\to 0}\eta_\epsilon(\boldsymbol{x}) = \delta_0(\boldsymbol{x}), \tag{4}$$

which ensures that the interpolation sum (1) converges to the Dirac delta function δ_0 when ϵ goes to zero. The smoothing kernel is commonly a compactly supported function of radius δ_c:

$$\eta_\epsilon(\boldsymbol{x}) = 0 \quad \text{if } |\boldsymbol{x}| > \delta_c, \tag{5}$$

so that the sampling points needed to compute (2) are all located in the vicinity of the interpolation node. This property allows us to introduce some optimised neighbours search algorithm in order to reduce the computational time. In *JOSEPHINE*, the smoothing kernel $\eta_\epsilon(\boldsymbol{x})$ is the modified Gaussian function, as proposed by Colagrossi and Landrini [6]:

$$\eta_\epsilon(\boldsymbol{x}) = \frac{e^{-(\frac{|\boldsymbol{x}|}{\epsilon})^2} - e^{-(\frac{\delta_c}{\epsilon})^2}}{2\pi\int_0^{\delta_c} s(e^{-(\frac{s}{\epsilon})^2} - e^{-(\frac{\delta_c}{\epsilon})^2})\,\mathrm{d}s}. \tag{6}$$

The cut-off distance δ_c has been introduced to satisfy (5), and is set to $\delta_c = 3\epsilon$. The smoothing length ϵ is linked to the initial inter-particle distance h by the following relation:

$$\epsilon = \kappa h. \tag{7}$$

$\kappa = 1.33$ has been used for all the following presented computations. This value of κ leads to 50 particles within the kernel support, which can be seen as the interpolation stencil. Increasing κ is a way to improve the accuracy of interpolations but it also increases the computation time exponentially. A large variety of smoothing kernels are available but the choice of a Gaussian kernel has been motivated by its lower sensibility to particle disorder [23]. This kernel is thus more adapted when considering various cases.

After some algebra (see [31,16]), the interpolation (2) can be extended to the approximation of the gradient of $f()$:

$$\langle \nabla f(\boldsymbol{x}_i) \rangle = \sum_j f(\boldsymbol{x}_j) \nabla \eta_\epsilon (\boldsymbol{x}_i - \boldsymbol{x}_j) V_j. \tag{8}$$

1.2. Governing equations

The 2D Euler equations for mass and momentum conservation of a compressible fluid, expressed in Lagrangian form, are:

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{\nabla P}{\rho} + \boldsymbol{g}, \tag{9}$$

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \boldsymbol{u} + \text{Boundary Conditions}, \tag{10}$$

where $\boldsymbol{u}$, ρ, P are respectively the velocity, the density and the pressure fields. $\boldsymbol{g} = (0, -g)$ stands for the action of gravity. The system (9)–(10) is closed by an equation of state:

$$P = \frac{\rho_0 c_0^2}{\gamma} \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right), \tag{11}$$

where c_0 is the sound speed, ρ_0 the density at the free surface and γ the polytropic constant ($\gamma = 7$ for water). c_0 could be set to the real sound speed in the simulated media, leading to extremely small timesteps to keep the time integration procedure stable. In order to reduce the computational cost, c_0 is set to a lower value than the physical one, but large enough to model a nearly incompressible (weakly compressible) fluid. The variation rate of the density can be approximated using [21]:

$$\frac{|\delta\rho|}{\rho} \approx \frac{|\boldsymbol{u}_{\max}|^2}{c_0^2}, \tag{12}$$

where $|\boldsymbol{u}_{\max}|$ is the maximum expected velocity, which depends on the case studied. In each simulation described after, c_0 is then defined to a value 10 times or more larger than $|\boldsymbol{u}_{\max}|$, to ensure that density oscillations would not exceed 1%. The numerical values of $|\boldsymbol{u}_{\max}|$ and c_0 will be given for each test case in Section 2.

1.3. Numerical scheme

The derivation of Euler equations using the SPH formalism has been widely studied [3,27]. Following [6,9], the system of discrete Euler equations that has been implemented in *JOSEPHINE* is:

$$\frac{D\boldsymbol{x}_i}{Dt} = \boldsymbol{u}_i, \tag{13}$$

$$\frac{D\boldsymbol{u}_i}{Dt} = -\frac{1}{\rho_i} \sum_j (P_i + P_j) \nabla \eta_\epsilon (\boldsymbol{x}_{ij}) \frac{m_j}{\rho_j} + \boldsymbol{g} + \boldsymbol{f}_{iG_i}, \tag{14}$$

$$\frac{D\rho_i}{Dt} = \rho_i \sum_j \left(\boldsymbol{u}_{ij} + \underbrace{\boldsymbol{n}_{ij} \left(\frac{c_{ij}}{\rho_i} (\rho_j - \rho_i - \Delta\rho_{ij}) \right)}_{\text{Rusanov flux}} \right) \cdot \nabla \eta_\epsilon (\boldsymbol{x}_{ij}) \frac{m_j}{\rho_j}, \tag{15}$$

$$P_i = \frac{\rho_0 c_0^2}{\gamma} \left(\left(\frac{\rho_i}{\rho_0} \right)^\gamma - 1 \right) + \text{Boundary Conditions}, \tag{16}$$

with $\boldsymbol{u}_{ij} = \boldsymbol{u}_i - \boldsymbol{u}_j$ and $\boldsymbol{x}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$. Eq. (13) is basically the equation for the displacement of particles. Eq. (14) stands for the momentum equation containing an extra term, $\boldsymbol{f}_{iG_i}$, which will be described and discussed in Section 1.4. This discrete approximation of the pressure gradient has already been employed by Colagrossi and Landrini [6] and is known to be more accurate when simulating various fluids [6]. Eq. (15) is the continuity equation, where a Rusanov flux has been added, following the work by Ferrari et al. [9]. It helps to enhance the scheme stability by reducing density fluctuations, which are often observed with weakly compressible models. The Rusanov flux involves the effective numerical sound velocity $c_{ij} = \max(c_i, c_j)$, which may be computed by:

$$c_i = \sqrt{\frac{\partial P_i}{\partial \rho_i}} = c_0 \left(\frac{\rho_i}{\rho_0} \right)^{\frac{\gamma-1}{2}}, \tag{17}$$

depending on the equation of state (16). $\boldsymbol{n}_{ij}$ is the normal vector between particle i and j:

$$\boldsymbol{n}_{ij} = \frac{\boldsymbol{x}_{ij}}{r_{ij}}, \tag{18}$$

where r_{ij} is the distance between the two particles. The present model differs from the one proposed by Ferrari et al. [9] by adding a corrective *hydrostatic* term, in the density difference, which is computed as follows:

$$\Delta\rho_{ij} = \frac{\delta\rho_{ij} + \delta\rho_{ji}}{2}, \tag{19}$$

$$\delta\rho_{ij} = \left[\rho_i^{\gamma-1} + \frac{\rho_0^{\gamma-1}(\gamma-1)}{c_0^2} g(y_j - y_i) \right]^{\frac{1}{\gamma-1}} - \rho_i. \tag{20}$$

This term helps to prevent excessive mass fluxes in the fluid, especially where a density jump between two interacting particles is mainly due to the gravitational force. This correction term may be inappropriate in the case of liquid drops freely falling in the domain, where the pressure field does not contain any hydrostatic component. Another example could be the case of a plunging jet (wave breaking), when the free surface is not a single value function. But, this term is very small if compared to the density difference and it only acts in the case of very long simulations. A better but more computationally expensive corrective term could be introduced, similarly to [19].

The choice of introducing this flux instead of the Moving Least Square (MLS) density re-initialisation [6] has been motivated by the lower numerical cost it provides and by the fact that no parameter calibration is required (i.e. the number of time steps between two re-initialisations). Recently, the new δ-SPH model has been proposed by Molteni and Colagrossi [20], where the numerical oscillations are removed by the use of a diffusive term in the continuity equation and an artificial viscosity in the momentum equation. Since, the δ-SPH scheme has been improved and applied with success to free-surface flows in different contexts [2,1,19], provided that the various parameters are carefully calibrated. As it will be presented in the following of this paper, the present model gives satisfactory results and seems to be more suited for a first approach of SPH computations.

JOSEPHINE provides two schemes for the time-integration of Eqs. (13)–(16): a predictor–corrector scheme and a fourth order Runge–Kutta scheme, with a unique timestep for all the particles. Both methods require to satisfy a Courant Friedrichs Levy condition on the timestep to remain stable. The condition is

$$\Delta t = \beta \frac{\epsilon}{c_0}, \tag{21}$$

with $\beta \approx 0.1$ in the case of the predictor–corrector scheme, and $\beta \approx 0.3$ with the fourth order Runge–Kutta scheme.

1.4. Boundary conditions

Various approaches are available in the literature to enforce solid boundary conditions [21,30,8,12]. In *JOSEPHINE*, free-slip

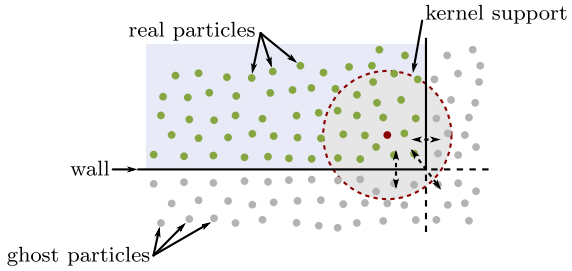

Fig. 2. The ghost particles technique for wall boundaries: axial symmetries are applied near straight lines, and central symmetries near corners.

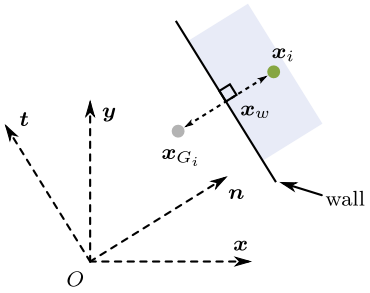

Fig. 3. The ghost particles technique for wall boundaries: local coordinates system used in the definition of ghost particles properties. $\boldsymbol{n}$ and $\boldsymbol{t}$ are respectively the normal and tangent vectors to the wall, $\boldsymbol{x}_i$ is the fluid particle position, $\boldsymbol{x}_{G_i}$ is the position of the ghost particle and $\boldsymbol{x}_w$ is the orthogonal projection of $\boldsymbol{x}_i$ on the wall.

conditions on solid boundaries are modelled using the ghost particles technique [15,32,6]. It consists in mirroring all the fluid particles, whose kernel support intersects the wall (Fig. 2). These ghost particles are updated at each time step and their density, pressure and velocity are deduced from those of the linked real particles. The ghost particles method may be more time-consuming than others but, to our opinion, it seems to be the best compromise in obtaining a physical non-penetrating boundary condition. However, its extension to complicated shaped bodies can be difficult [13]. A fixed ghost particles method has been introduced by Marrone et al. [19] to extend the classical method to arbitrary shaped bodies, but has not been implemented in *JOSEPHINE*. Adopting the notations described in Fig. 3, the position $\boldsymbol{x}_{G_i}$ of a ghost particle is:

$$\boldsymbol{x}_{G_i} = 2\boldsymbol{x}_w - \boldsymbol{x}_i, \tag{22}$$

$\boldsymbol{x}_i$ being the position of the original particle and $\boldsymbol{x}_w$, the position of the orthogonal projection of $\boldsymbol{x}_i$ on the wall. The velocity of the ghost particle depends upon the condition to enforce, the shape and the motion of the boundary. In the case of a free-slip condition on a fixed plane wall, the velocity is computed as:

$$\boldsymbol{u}_{G_i} = -u_{in}\boldsymbol{n} + u_{it}\boldsymbol{t}, \tag{23}$$

where $\boldsymbol{n}$ and $\boldsymbol{t}$ are respectively the normal and tangent vectors to the wall boundary. u_{in} and u_{it} are respectively the orthogonal projections of $\boldsymbol{u}_i$ onto $\boldsymbol{n}$ and $\boldsymbol{t}$. The pressure boundary condition to enforce on a fixed wall is:

$$\frac{\partial P}{\partial \boldsymbol{n}} = \frac{\partial P}{\partial \rho}\frac{\partial \rho}{\partial \boldsymbol{n}} = \rho \boldsymbol{g}\cdot\boldsymbol{n}. \tag{24}$$

Combining the latter form with the equation of state (11) leads to:

$$\frac{\partial \rho}{\partial \boldsymbol{n}} = \rho\frac{\rho_0^{\gamma-1}}{c_0^2\rho^{\gamma-1}}\boldsymbol{g}\cdot\boldsymbol{n}. \tag{25}$$

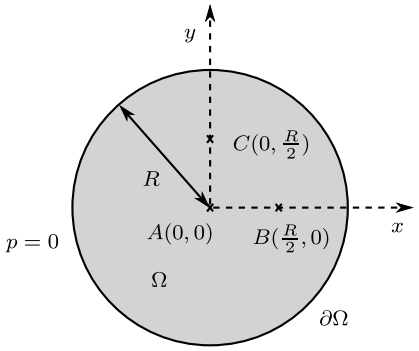

Fig. 4. Evolution of a circular patch of fluid: definition of the initial fluid domain. A, B and C are the locations of analytical and numerical comparisons. R is the radius of the fluid domain Ω. $\partial\Omega$ is the free surface of the fluid domain, where the pressure falls to zero.

Then, integrating the previous equation, one can define the right density for a ghost particle near a fixed horizontal wall (see [13,25]):

$$\rho_{G_i} = \left[\rho_i^{\gamma-1} + \frac{\rho_0^{\gamma-1}(\gamma-1)}{c_0^2}g(y_{G_i} - y_i)\right]^{\frac{1}{\gamma-1}}. \tag{26}$$

In the case of a fixed vertical wall, this relation becomes:

$$\rho_{G_i} = \rho_i. \tag{27}$$

Using the equation of state (16), the pressure associated to this ghost particle is obtained. The mass of a ghost particle being the same as the original particle ($m_{G_i} = m_i$), its volume is computed as $V_{G_i} = m_i\rho_{G_i}$. This approach produces the desired repulsion mechanism at the wall. However, in some cases such as the impact of a jet on a wall, or when the timestep is not sufficiently small to keep all the fluid particles inside the domain, an extra force is added to avoid any wall penetration. This force is inspired by the repulsive particles method [21] but is computed only between a real particle and its ghost:

$$\boldsymbol{f}_{iG_i} = \begin{cases} gH\left[\left(\frac{\epsilon}{2r_{iG_i}}\right)^4 - \left(\frac{\epsilon}{2r_{iG_i}}\right)^2\right]\frac{\boldsymbol{x}_{iG_i}}{r_{iG_i}^2} & \text{if } \left(\frac{\epsilon}{2r_{iG_i}}\right) \leqslant 1, \\ 0 & \text{otherwise,} \end{cases} \tag{28}$$

with r_{iG_i} being the distance between the real and the ghost particles, H is the fluid height in the simulated case, and $\boldsymbol{x}_{iG_i} = \boldsymbol{x}_{G_i} - \boldsymbol{x}_i$. In this way, the exerted force is purely normal to the wall and only acts in the extreme case, where the particle is at a distance lower than $\epsilon/4$ from the boundary.

2. Validation results

2.1. Evolution of a circular patch of fluid

First, *JOSEPHINE* has been used to simulate a case where no wall boundary condition has to be enforced. Thus, in this first step, only the numerical model for Euler equations with a free surface will be validated. Let us consider a circular patch of fluid of radius R, free from any external force (see Fig. 4). In the absence of gravity, the correction term of Eq. (20) is zero. The initial velocity and pressure fields are prescribed as:

$$\begin{cases} u_0(x,y) = A_0 x \\ v_0(x,y) = -A_0 y \\ P_0(x,y) = \dfrac{1}{2}\rho_0 A_0^2\left(R^2 - (x^2+y^2)\right) \end{cases} \quad \forall (x,y),\ x^2+y^2 \leqslant R. \tag{29}$$

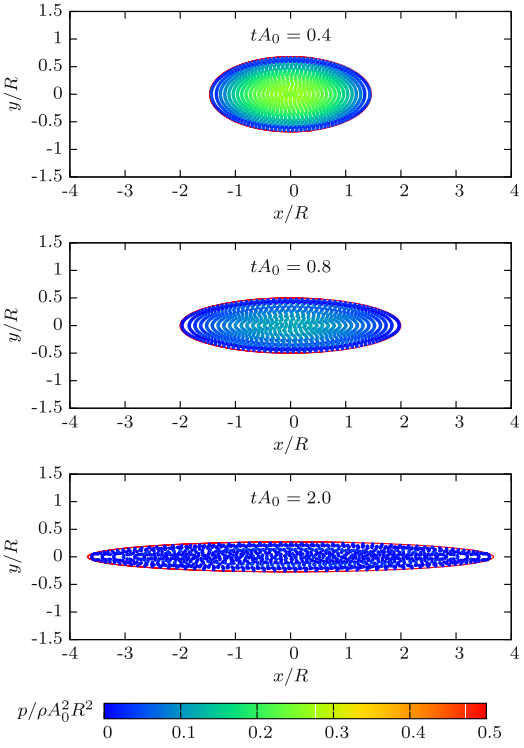

Fig. 5. Evolution of a circular patch of fluid: numerical solution using *JOSEPHINE* at instants $tA_0 = 0.4$, 0.8 and 2.0 (from top to bottom). The particles are coloured according to the non-dimensional pressure $p/\rho_0 A_0^2 R^2$. The red line figures the analytical solution. The computation parameters are: $R/\epsilon = 14.7$, $c_0 = 14RA_0$ and $\Delta tA_0 = 10^{-3}$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Subjected to this initial field, the drop deforms as time evolves. If A_0 is positive, the drop is stretched along the x-axis and compressed along the y-axis. The generated flow is irrotational, and as a consequence of the incompressibility, the drop keeps an elliptical shape. An analytical solution is available [5]. This case has already been investigated [21,3,33,9] in order to validate various corrected SPH models. The main difficulty is to simultaneously obtain a smooth pressure field solution together with the right interface evolution, even up to large time ($tA_0 = 2$).

This case has been computed using *JOSEPHINE* with three different particle spacings: $R/\epsilon = 14.7$, 29.4 and 58.8. Respectively, 1140, 4800 and 19 150 particles have been spread on concentric circles instead of on a Cartesian lattice to avoid any *grid* distortion. For the three computations, the fluid properties have been set as follows: $\rho_0 = 1000$, $c_0 = 14RA_0$. $|\boldsymbol{u}_{\max}|$ is here RA_0, which ensures that $|\delta\rho|/\rho = 14^{-2} < 1\%$, according to Eq. (12). The solution has been integrated in time with a predictor–corrector scheme and the following timesteps $\Delta tA_0 = 10^{-3}$ for the $R/\epsilon = 14.7$ case, $\Delta tA_0 = 5 \times 10^{-4}$ ($R/\epsilon = 29.4$) and $\Delta tA_0 = 2.5 \times 10^{-4}$ ($R/\epsilon = 58.8$). This choice of ΔtA_0 satisfies the CFL condition (Eq. (21)), for all the three presented computations. Fig. 5 depicts the time-evolution of the drop ($R/\epsilon = 14.7$) and shows a good agreement between the computed shape and the analytical solution. In the upper part of Fig. 6, the time evolution of pressure at the centre of the drop is plotted. The solution shows small amplitude oscillations that come from the propagation of sound waves inside the weakly compressible fluid. Filtering these results leads to a more accurate solution: we kept only frequencies $f < 3.5A_0$, which corresponds to the lower frequency of the dominant mode (see [5] for details). The horizontal velocity at B is plotted in the lower part of Fig. 6 and is very close to the analytical solution, even at the lowest resolution (i.e. $R/\epsilon = 14.7$, red-squares line in Fig. 6). The vertical velocity at C is shown on the same figure. The velocity magnitude decreases slowly and jump to zero at $tA_0 \approx 0.8$, this

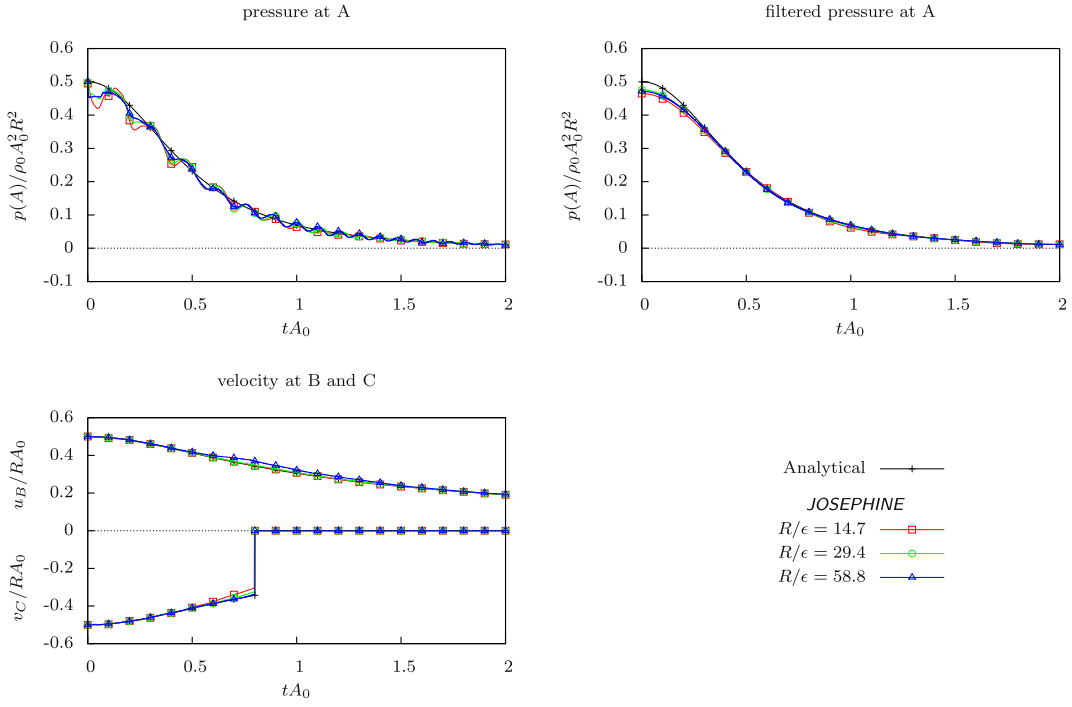

Fig. 6. Evolution of a circular patch of fluid. Three SPH simulations using *JOSEPHINE*, with different discretisations, (colour lines) are compared to the analytical solution (black line). Top left: time evolution of the pressure at point A (see Fig. 4). Top right: time evolution of filtered pressure (frequencies $f \geqslant 3.5A_0$ have been filtered out by a post-processing tool). Bottom left: horizontal velocity at B and vertical velocity at C. Computational parameters are: $\rho_0 = 1000$, $c_0 = 14RA_0$, $\Delta tA_0 = 10^{-3}$ for the $R/\epsilon = 14.7$ case (red-squares line), $\Delta tA_0 = 5 \times 10^{-4}$ for the $R/\epsilon = 29.4$ case (green-circles line) and $\Delta tA_0 = 2.5 \times 10^{-4}$ for the $R/\epsilon = 58.8$ case (blue-triangles line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

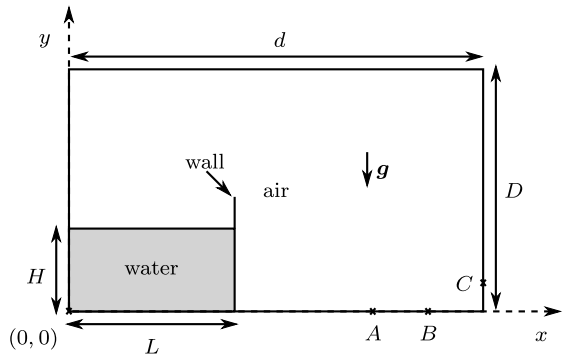

Fig. 7. Definition of the dam-break problem. A rectangular patch of water is initially at rest behind a wall. H is the height of the water column, L is its length, d is the length of the rectangular box that contains the fluid and D is its height. $L/H = 2$ and $d/H = 5.366$. A$(3.721H, 0)$, B$(4.542H, 0)$ and C$(0, 0.192H)$ are the locations of analytical, experimental and numerical comparisons.

jump has been captured using an external post-processing code and considering that point C is no longer in Ω when:

$$\sum_j \eta_\epsilon(\boldsymbol{x}_C - \boldsymbol{x}_j)\frac{m_j}{\rho_j} < 0.5, \tag{30}$$

which corresponds to a half-empty kernel support and comes from Eq. (3). This procedure is accurate regardless of the number of involved particles. However, a finer resolution (i.e. $R/\epsilon = 58.8$, blue-triangle line in Fig. 6) leads to a better velocity computation before the passage of the interface through point C. The results of the three discretisations show a fairly good convergence of the present numerical scheme applied to this academic test case, in terms of pressure, velocity and interface evolution.

2.2. Dam-break problem

The collapse of a water column is often considered in the SPH literature, since this case shows the ability of SPH models to deal with large deformation of the interface. Again, many corrected formulations can be used in order to compute this kind of flow: artificial viscosity and XSPH [21,6], re-normalisation [3,17], diffusive terms [20,19], Riemann solvers [28] or turbulence models [34]. Some efforts by means of a fully incompressible SPH model can be found [30,14] but these approaches require an accurate algorithm for the detection of the free surface.

Fig. 7 shows a sketch of the initial setup for this test case. Computations have been run with the same geometrical setup as [6]: $L/H = 2$, $d/H = 5.366$. Particles are initially positioned on a Cartesian lattice. Three space resolutions have been studied: 25×50, 50×100 and 100×200. The solution has been integrated in time by a fourth order Runge–Kutta scheme with $\Delta t\sqrt{g/H} = 8 \times 10^{-4}$, 4×10^{-4}, 2×10^{-4} respectively and $c_0 = 10\sqrt{gH}$, which satisfy the previous incompressibility and CFL requirements. The flow starts with the removal of the wall, which is not explicitly modelled (see Fig. 8, $\tau_0 = t\sqrt{g/H} = 0^+$). The pressure field is then initialised with an approximate solution of the Laplace problem, where the pressure is zero everywhere at the free surface (on the top and right sides of the fluid domain) [11,5]. Under the action of gravity, the water column collapses and propagates along the floor (from τ_1 to τ_2). Then, the fluid impacts the right-hand side vertical wall (τ_3). A vertical jet grows up, overturns backward and then falls down (from τ_4 to τ_6). Several reconnections of the interface occur (from τ_7 to τ_{10}), that are not accurately computed here, since the air phase is not taken into account (as in [6]).

The total energy can be obtained by:

$$E = \underbrace{\frac{1}{2}\sum_i m_i \|\boldsymbol{u}_i\|^2}_{\text{kinetic energy}} + \underbrace{g\sum_i m_i y_i}_{\text{potential energy}} + \underbrace{\sum_i m_i e_i}_{\text{internal energy}}, \tag{31}$$

where the internal energy is approximated by:

$$\frac{De_i}{Dt} = -\frac{P_i}{\rho_i}(\nabla \cdot \boldsymbol{u}_i) = \frac{P_i}{\rho_i}\sum_j \boldsymbol{u}_{ij} \cdot \nabla \eta_\epsilon(\boldsymbol{x}_{ij})\frac{m_j}{\rho_j}. \tag{32}$$

The time evolution of energy is plotted in Fig. 9. In a dimensionless form, E_0 stands for the initial total (potential and internal) energy before the dam-break and ΔE is the energy difference between the initial state and the hydrostatic final state, where the fluid is again at rest. An advantage of the Rusanov flux used here, see Eq. (15), is that it avoids the use of an artificial viscosity, and the model becomes less dissipative. *JOSEPHINE* only loses 1% of the total energy for the 50×100 case, which is less than the former result of Colagrossi and Landrini [6], who carefully tuned the artificial viscosity, density re-initialisation and XSPH parameters to reach the best conservation rate. Increasing the number of particles even reduces the energy dissipation, which mainly comes from the Rusanov Flux. Thus, the energy dissipation could be further improved by reducing the impact of the Rusanov Flux, this aspect is under investigation. However, for a similar space discretisation (i.e. 49×100, Fig. 9), the use of a Rusanov flux as in *JOSEPHINE* is less dissipative than the use of an artificial viscosity and a density re-initialisation [6].

Fig. 10 depicts the front position evolution computed, using the three space resolutions, by *JOSEPHINE*. Results are very close to the literature [6]. The water heights at points A$(3.721H, 0)$ and B$(4.542H, 0)$ (see Fig. 7) have been extracted and compared to numerical results of Colagrossi and Landrini [6] and experiments by Zhou et al. [35]. Figs. 11 and 12 show a good agreement between both numerical solutions at the earlier stages of flow (before τ_8, the collapse of the entrapped air cavity). The experimental water heights are larger than the numerical results between τ_1 and τ_3, these differences are mainly due to the way the experimental runs start. Considering both the position of the water front and the evolution of water heights, the results of *JOSEPHINE* converge to a single value, which is close to the numerical and the experimental data.

The impact pressure at point C$(0, 0.192H)$ has been interpolated and is plotted in Fig. 13. *JOSEPHINE* slightly underestimates this pressure during the earlier instants following the impact (from τ_3 to τ_4), but this small difference is reduced as time evolves. The first peak (τ_8) comes from the impact of the backward plunging jet while the second one is generated by the collapse of the air cavity (τ_{10}). The numerical results of Colagrossi and Landrini [6] and our results are in good agreement. In [6], the author also shows a two-phase solution, which is closer to the experiments since air-cushion effects are taken into account. Increasing the number of particles in *JOSEPHINE* allows to converge to a single and less oscillating solution (Fig. 13).

3. Parallel implementation

3.1. Domain decomposition

To cope with large number of particles and to compute the previous cases with higher performance, *JOSEPHINE* has been parallelised. The present program has been designed to handle two-dimensional flows in open basins, so we chose to distribute the computational effort by dividing the domain into vertical sub-domains. There are as many vertical sub-domains as the desired

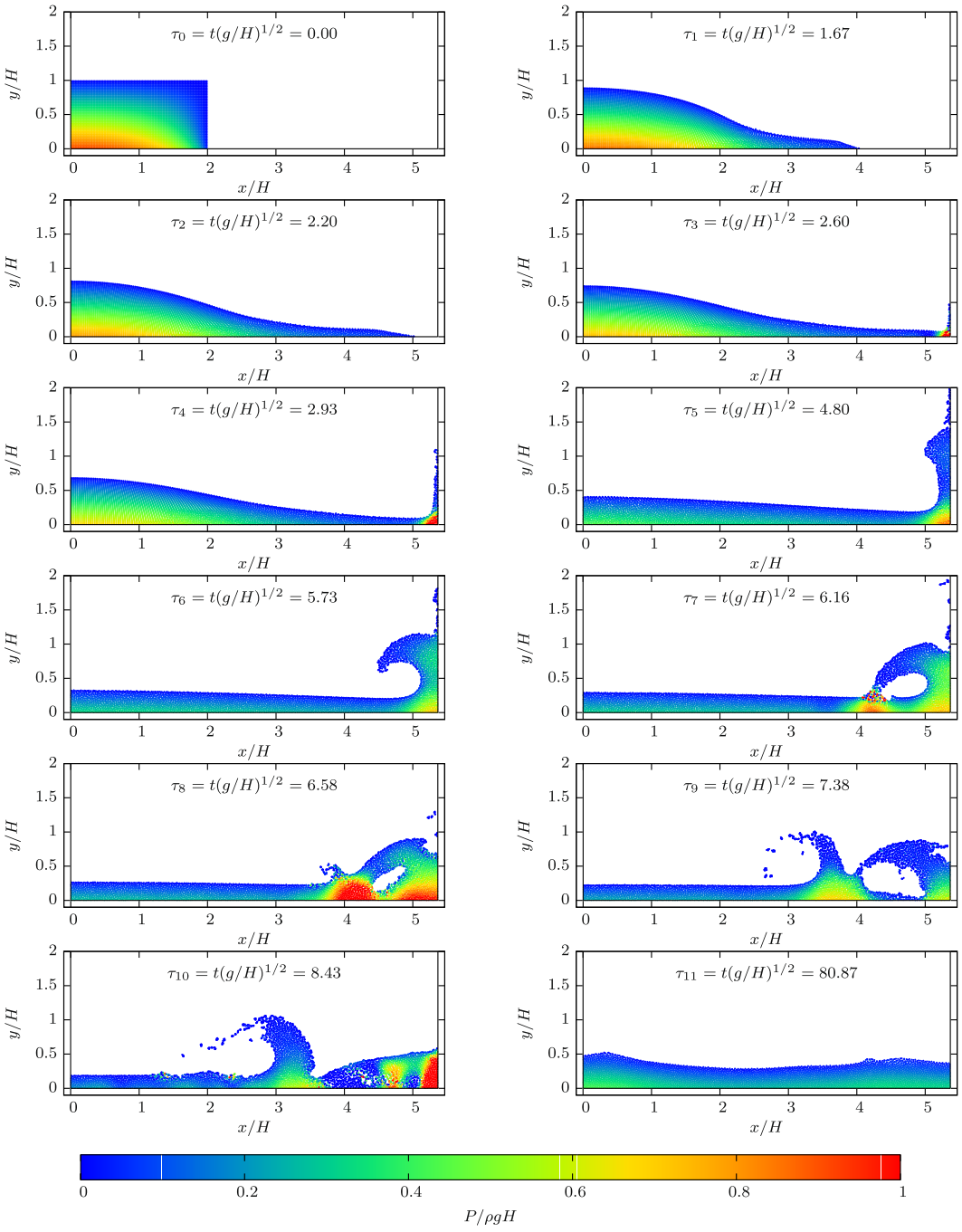

Fig. 8. Time evolution of the water domain after the dam breaking, using *JOSEPHINE*, involving 5000 particles. The solution is shown at twelve dimensionless instants and particles are coloured by the dimensionless pressure $P/\rho_0 g H$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

number of processors. All communications are realised thanks to the use of dedicated MPI libraries, in order to optimise the number and the size of the messages between processors. On both sides of the interface between processors, the data transfer for the evaluation of Eqs. (14)–(15) is only required on a vertical stripe, the length of which is basically the radius of the interpolation kernel (see Fig. 14 for details). The size of these interaction zones is then optimised for applications, where the domain height is weak compared to its length (a wave tank, for example). Moreover, the initial domain division between processors is easily made thanks to the use of the first position coordinate of the particles. For extended computational domains, this allows a significant gain in memory since all particles data is never stored on a single machine. The number of information updates is minimised too, since one processor have a maximum of two neighbouring processors.

3.2. Load balancing

At the end of a timestep, particles may have crossed processors interfaces. These particles and their related information are then transferred towards the processor treating the considered domain, to preserve the vertical shape of each sub-domains. However, these transfers may lead to important load imbalances, which eventually damage the CPU time consumption. To handle with this, the processor interface is updated at the end of each time step in order to keep a constant number of particles in the interaction zones.

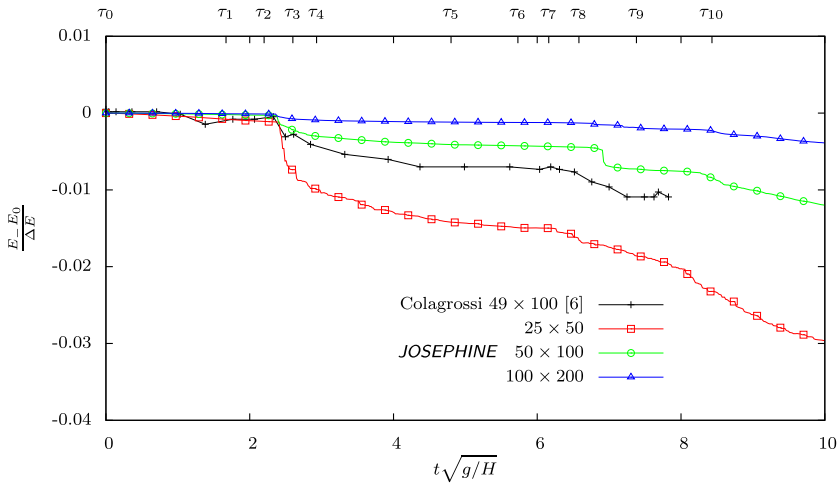

Fig. 9. Total energy evolution. Three SPH simulations using *JOSEPHINE* (colour lines) are compared to numerical results of Colagrossi and Landrini [6] (black line). Computational parameters are: $\rho_0 = 1000$, $c_0 = 10\sqrt{gH}$, $\Delta t\sqrt{g/H} = 8 \times 10^{-4}$ for the 25×50 case (red-squares line), $\Delta t\sqrt{g/H} = 4 \times 10^{-4}$ for the 50×100 case (green-circles line) and $\Delta t\sqrt{g/H} = 2 \times 10^{-4}$ for the 100×200 case (blue-triangles line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

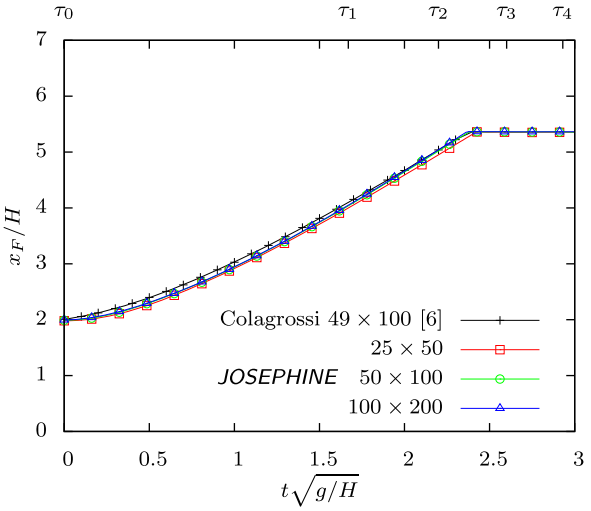

Fig. 10. Water front evolution after the dam-breaking. Three SPH simulations using *JOSEPHINE* (colour lines) are compared to numerical results of Colagrossi and Landrini [6] (black line). Computational parameters are: $\rho_0 = 1000$, $c_0 = 10\sqrt{gH}$, $\Delta t\sqrt{g/H} = 8 \times 10^{-4}$ for the 25×50 case (red-squares line), $\Delta t\sqrt{g/H} = 4 \times 10^{-4}$ for the 50×100 case (green-circles line) and $\Delta t\sqrt{g/H} = 2 \times 10^{-4}$ for the 100×200 case (blue-triangles line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 15 summarises the three steps of the load balancing procedure:

(a) At the beginning of the timestep, particles in the interaction zone are detected by the left processor and right processors.
(b) At the end of the timestep, particles are advected according to their own velocity. The number of particles remaining in the interaction zone is evaluated showing, as scheduled, a difference with the initial number. The first coordinate of these particles is stored in a table and the new interface position is computed thanks to the determination of the abscissa value, that allows to conserve a constant number of particles in the interaction zone. This operation, only realised by one of the two processors, actually uses a fast algorithm, Quickselect [26], whose complexity is of $O(\log(N))$. Quickselect allows to find the kth greatest value in any unordered list. The CPU time dedicated to this task is negligible if compared to the overall CPU time consumption.
(c) Particles are affected to a processor according to their relative position with the new border.

This algorithm reveals to be very efficient. However, we have observed that a disequilibrium of a single particle might occur from time to time. Such small disequilibria may lead, at the end of a complete computation of thousands of timesteps, to important load imbalance and performance losses. These successive disequilibria come from the step (b) of the previous procedure. Trying to maintain a constant number of particles in each interaction region does not imply that the local number of particles of each processor will remain constant. Let N_n^{zi} be the initial number of particles in the interaction zone of interest, before the nth timestep and Quickselect(list, N) be the calling sequence to find the Nth largest number in list. Calling Quickselect(Particles in interaction zone, N_n^{zi}) allows to find the new boundary position for the $(n+1)$th timestep that minimises the number of particles to transfer, but this is not sufficient to conserve the local number of particles. Calling Quickselect(Particles in the processor, N_0), N_0 being the local number of particles of the processor, will strictly conserve the number of particles, but will be more time consuming too, since the search procedure is called with the whole particles list. Introducing a *memory* $\Delta N = N_n - N_0$, which is the difference between the initial local number of particles and the new one, the call of Quickselect(Particles in interaction zone, $N_n^{zi} + \frac{\Delta N}{2}$) allows to keep a constant local number of particles on each processor for a lower number of processor operations. Fig. 16 shows the behaviour of such a parallel algorithm on the dam-break case. This computation case, with rapid dynamics requires an efficient load balancing between processors owing to the important fluid domain deformation and stretching. Particles are coloured according to their depending processor. One can observe the important domain extension associated with the 4th processor (red colour) between τ_0 and τ_3, when the whole fluid domain is stretched. It is then brutally narrowing after the impact occurs, between τ_4 and τ_6.

CPU time performance of the parallel implementation have been realised and compared with the sequential computations. Fig. 17 depicts the speed up of the parallel implementation for the dam break case for several numbers of particles varying from 5000 to 1,280,000. These results show that the parallel implementation allows important CPU time savings for computations with significant number of particles. For low numbers of particles, load balancing operations take significant CPU time, if compared with the overall computation, which makes the use of additional pro-

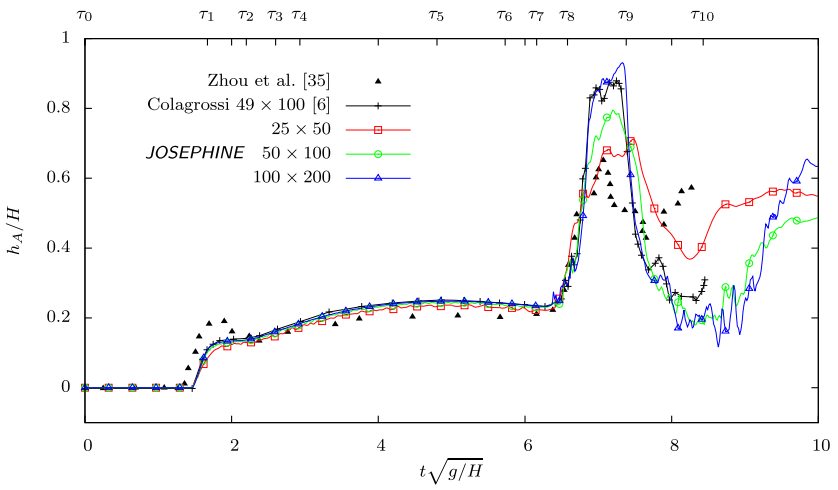

Fig. 11. Water height evolution at point A (see Fig. 7). Three SPH simulations using *JOSEPHINE* (colour lines) are compared to numerical results of Colagrossi and Landrini [6] (black line) and to experimental data by Zhou et al. [35] (black triangles). Computational parameters are: $\rho_0 = 1000$, $c_0 = 10\sqrt{gH}$, $\Delta t\sqrt{g/H} = 8 \times 10^{-4}$ for the 25×50 case (red-squares line), $\Delta t\sqrt{g/H} = 4 \times 10^{-4}$ for the 50×100 case (green-circles line) and $\Delta t\sqrt{g/H} = 2 \times 10^{-4}$ for the 100×200 case (blue-triangles line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

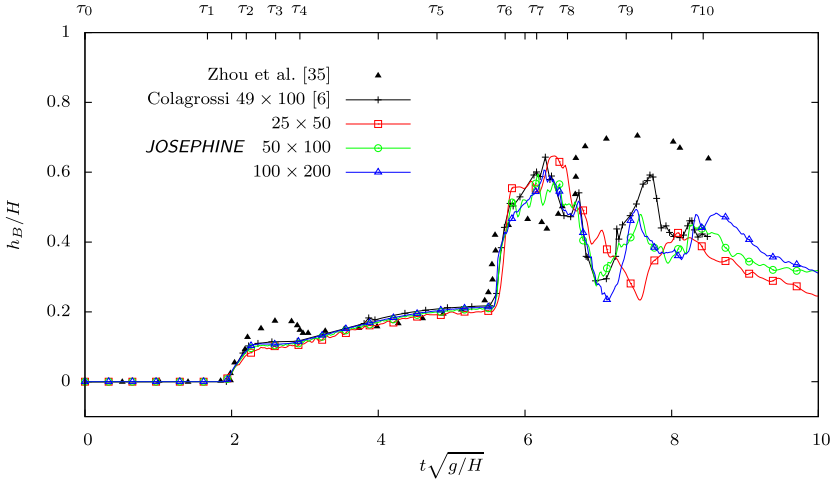

Fig. 12. Water height evolution at point B (see Fig. 7). Three SPH simulations using *JOSEPHINE* (colour lines) are compared to numerical results of Colagrossi and Landrini [6] (black line) and to experimental data by Zhou et al. [35] (black triangles). Computational parameters are: $\rho_0 = 1000$, $c_0 = 10\sqrt{gH}$, $\Delta t\sqrt{g/H} = 8 \times 10^{-4}$ for the 25×50 case (red-squares line), $\Delta t\sqrt{g/H} = 4 \times 10^{-4}$ for the 50×100 case (green-circles line) and $\Delta t\sqrt{g/H} = 2 \times 10^{-4}$ for the 100×200 case (blue-triangles line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

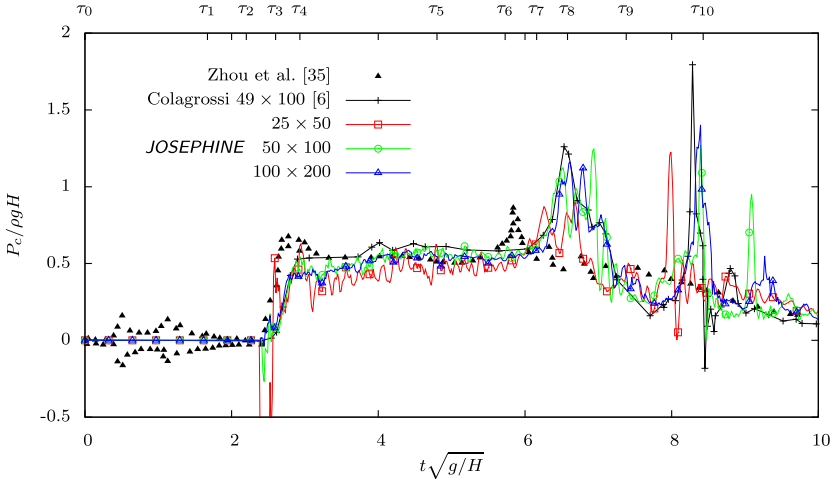

Fig. 13. Pressure evolution at point C (see Fig. 7). Three SPH simulations using *JOSEPHINE* (colour lines) are compared to numerical results of by Colagrossi and Landrini [6] (black line) and to experimental data by Zhou et al. [35] (black triangles). Computational parameters are: $\rho_0 = 1000$, $c_0 = 10\sqrt{gH}$, $\Delta t\sqrt{g/H} = 8 \times 10^{-4}$ for the 25×50 case (red-squares line), $\Delta t\sqrt{g/H} = 4 \times 10^{-4}$ for the 50×100 case (green-circles line) and $\Delta t\sqrt{g/H} = 2 \times 10^{-4}$ for the 100×200 case (blue-triangles line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

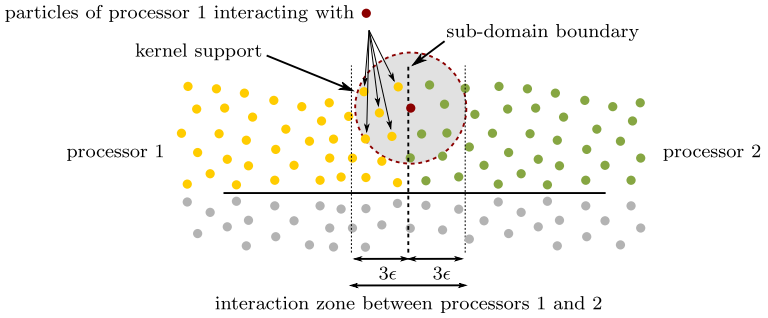

Fig. 14. *JOSEPHINE* parallelisation strategy: the whole domain is divided in several vertical sub-domains. Between two consecutive sub-domains, an interaction zone must be defined to perform the calculations properly.

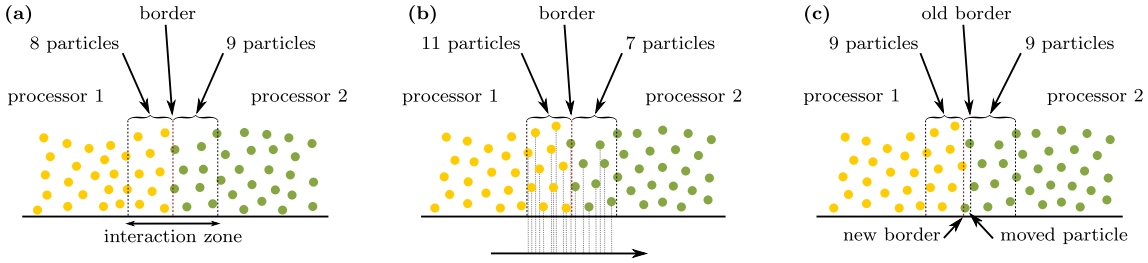

Fig. 15. Load balancing strategy.

cessors inefficient. For the higher numbers of particles (i.e. 320,000 and 1,280,000), the serial CPU time, used in the computation of the speedup, is biased, probably due to the use of swap memory, resulting in a speedup higher than ideal. This phenomenon is obvious for 2, 4 and even 8 processors with 1,280,000 particles. The present parallel implementation is efficient and is intrinsically designed to minimise the number of communications. Synchronous MPI communications have then been preferred, to prevent any error during the simulations. The use of asynchronous calls and its impact on the CPU time consumption is currently under investigation.

4. Program documentation

4.1. Folder structure

The source code is divided in several `.F` files which are stored in the following folder structure. Each Fortran 90 file contains some comments for description of the algorithms.

```
.
|-- DATA              data output folder
|-- make.inc          compilation parameters
|-- param_cir.dat     parameters file for the circular patch
|                     case
|-- param_dam.dat     parameters file for the dam-break case
|-- param_io.dat      input/output parameters
|-- param_sim.dat     simulation parameters
|-- param_sph.dat     SPH model parameters
`-- SRC               main sources folder
    |-- BOUND         boundary conditions
    |-- CORE          equation of state, kernel,
    |                 Navier-Stokes equations
    |-- INIT          initialization routines
    |-- INTERFACES    Fortran 90 interfaces
    |-- IO            input/output routines
    |-- ITER          time integration schemes
    |-- MAKE          compilation parameters templates
    |                 (gfortran, ifort)
    |-- Makefile
    |-- MODULES       Fortran 90 modules
    |-- PAIR          neighbours search routines
    |-- PARA          parallelization routines
    `-- josephine.F   main program
```

4.2. Compilation

The `make.inc` file has to be edited before trying to compile the whole code. The name of the desired compiler can be mentioned. Program name, debugging and optimisation options can also be defined. Templates are provided for `mpif90` compilers based on `gfortran` or `ifort`, in folder `SRC/MAKE/`. To compile the code, the following commands may be used:

```
cd SRC/
(make clean)
make
```

In order to switch to debugging mode, the folder structure has to be cleaned up with `make clean` and the last command is replaced by `make dbg`.

4.3. Running JOSEPHINE

All running options are contained in `.dat` files, at the root of the folder structure. These files can be edited but the number of lines or the order must be kept, otherwise errors may occur running the program. So, a backup is recommended before editing. To run the program, simply type the following command:

```
mpirun -np #NUMBER_OF_PROCESSORS# ./josephine.exe
```

4.4. Data output

In this first version of *JOSEPHINE*, data output is only available under `ASCII` format, and all files are put in the `DATA` folder. A different file, containing every particles data, is generated according to the `param_io.dat` parameters. Fluid particles are saved in `part#######.dat` files (where `#######` stands for the iteration number). Ghost particles (if present) are saved in `virt#######.dat` files. `GnuPLOT` (version 4.4 or newer) configuration and animation commands are also generated in `.gnu` files, for data visualisation. For example, to plot particles coloured by pressure, as in Fig. 8, type the following commands:

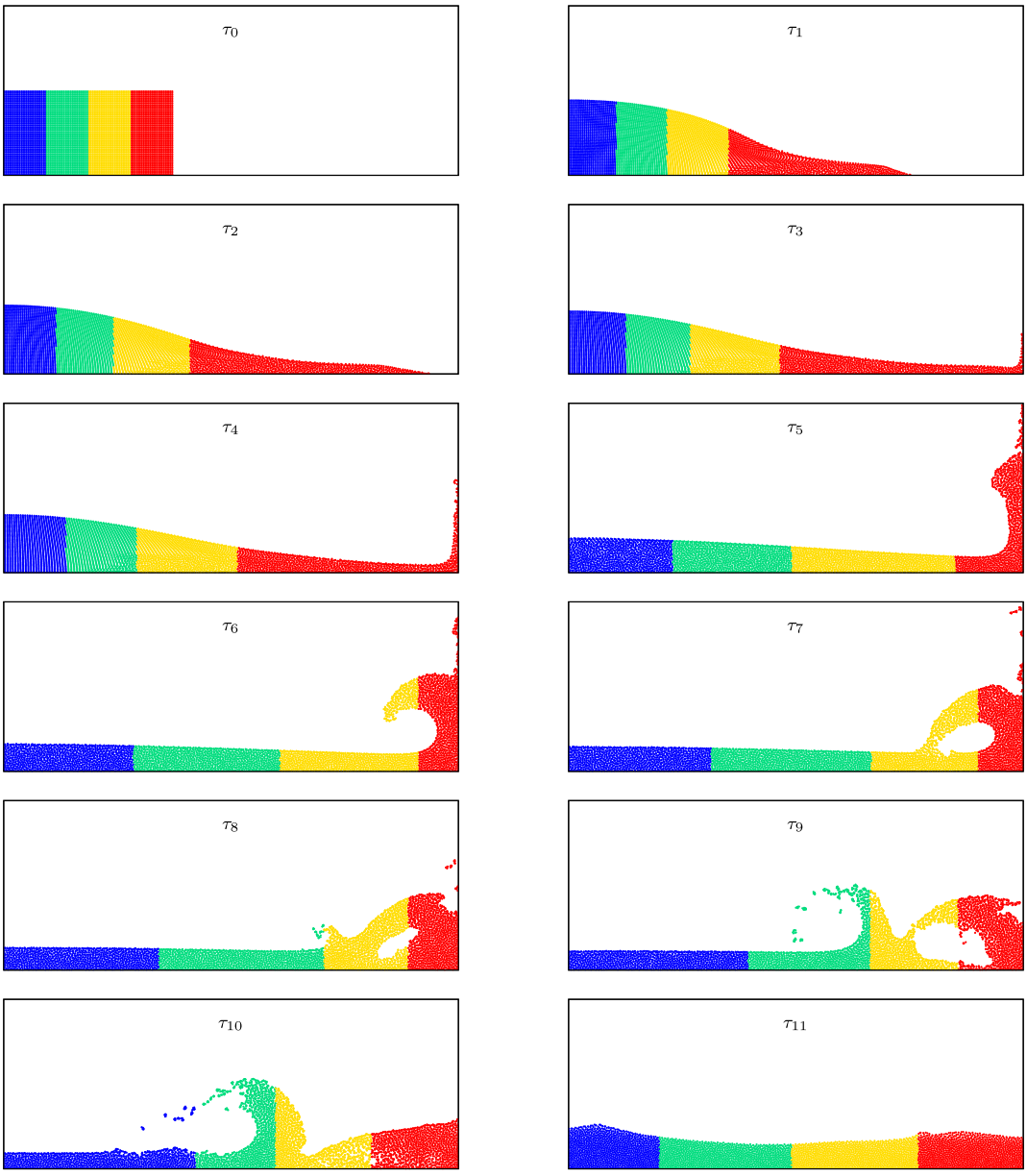

Fig. 16. Evolution of the domain decomposition for the dam-break case. Each colour corresponds to a different processor: one can easily observe that the domain extends horizontally in order to preserve a constant number of particles for each processor. (For interpretation of the references to colour in this figure, the reader is referred to the web version of this article.)

```
cd DATA/
gnuplot
load "dam_pre.gnu" (settings)
load "pre.gnu"     (animation)
```

All `.gnu` files are only scripts that have been provided to handle *JOSEPHINE* quickly, one can modify them by editing the corresponding source code file `SRC/IO/io_write_gnu.F`. Each `part#######.dat` particle file contains the following data formatted in eight columns:

1. x-coordinate
2. y-coordinate
3. x-component of the velocity
4. y-component of the velocity
5. pressure
6. density
7. mass
8. internal energy

Any modification of this format can only be made in the `SRC/IO/io_write_dat.F` source file, and the `SRC/IO/io_write_gnu.F` should be modified accordingly to make the previous visualisation scripts work.

5. Conclusion

In this paper, a new parallel program has been described. *JOSEPHINE* can be employed to compute free-surface non-viscous flows. Validation results have been given, in two different contexts, together with convergence analysis. The first one shows the ability of *JOSEPHINE* to deal with free-surface flows in open domain. The second one demonstrates that simple solid boundaries condition can also be enforced using *JOSEPHINE*.

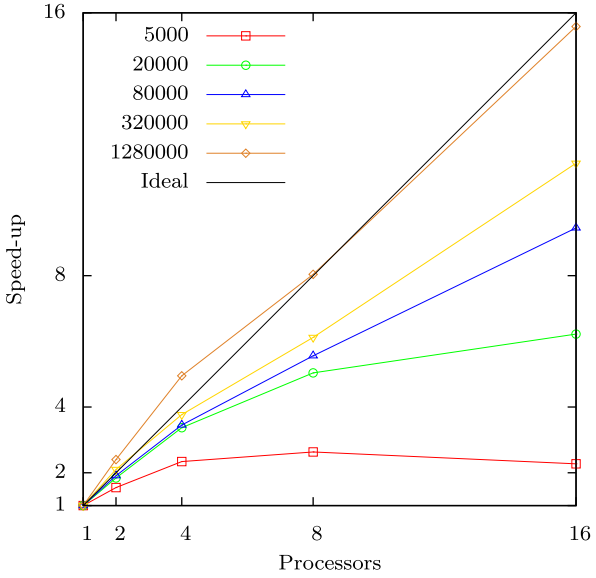

Fig. 17. Parallel speedup of *JOSEPHINE* computed the dam-break case for five numbers of particles.

The source code has been designed to allow any developer to extend its features easily. It might be also helpful for any student who would like to investigate SPH methods without starting from scratch.

JOSEPHINE will be extended in the future. Some new boundary conditions will be integrated and the program should be able to deal with 3D, multi-fluids cases soon. Alternative methods of stabilisation should also be added.

Appendix A

DAM-BREAK RUN OUTPUT

```
      __ _____    _____ _____ ____    __  __ __ _    __ _____
     / // __ \ / ___// ___// __ \ / / / // // | / // ___/
 __ / // / / / \_ \ / _/  / /_/ // /_/ // //  |/ // _/
/ /_/ // /_/ /___/ // /__ / _____// __  // // /|  // /__
\____/ \____//____//____//_/    /_/ /_//_//_/ |_//____/

##########################################################
#                                                        #
#                    DOUBLE PRECISION                    #
#                                                        #
##########################################################
#                                                        #
#                     DAM COLLAPSE                       #
#                                                        #
#      N=    5000     h= 0.16E-01       C= 0.24E+02      #
#      B/(RHO*G*H)= 0.14E+02   C/SQRT(GH)= 0.10E+02      #
#                        CFL= 0.304                      #
#                                                        #
##########################################################

##########################################################
#                                                        #
#            RUNGE-KUTTA 4 TIME-INTEGRATION              #
#                                                        #
#        ITERATION |    REALTIME    | INTERACTIONS      #
#                0 | 15:10:05.415 |            0      #
#             1000 | 15:12:38.501 |       122368      #
#             2000 | 15:15:26.091 |       123968      #
#             3000 | 15:18:26.730 |       122993      #
#             4000 | 15:21:50.795 |       123467      #
#             5000 | 15:25:28.033 |       123230      #
#             6000 | 15:29:19.331 |       122473      #
#             7000 | 15:33:17.864 |       121695      #
#             8000 | 15:37:20.395 |       121490      #
#             9000 | 15:41:27.395 |       118114      #
#            10000 | 15:45:19.366 |       115952      #
#            11000 | 15:49:27.425 |       119315      #
#            12000 | 15:54:04.575 |       119938      #
#            13000 | 15:58:34.801 |       118313      #
#            14000 | 16:02:58.086 |       121027      #
#            15000 | 16:07:17.807 |       121120      #
#            16000 | 16:11:28.228 |       121334      #
#            17000 | 16:15:26.751 |       121649      #
#            18000 | 16:19:18.647 |       121540      #
#            19000 | 16:23:15.296 |       122252      #
#            20000 | 16:27:23.785 |       122127      #
#                                                      #
##########################################################
```

References

[1] M. Antuono, A. Colagrossi, S. Marrone, C. Lugni, Propagation of gravity waves through an sph scheme with numerical diffusive terms, Comput. Phys. Commun. 182 (2011) 866–877.

[2] M. Antuono, A. Colagrossi, S. Marrone, D. Molteni, Free-surface flows solved by means of sph schemes with numerical diffusive terms, Comput. Phys. Commun. 181 (2010) 532–549.

[3] J. Bonet, T.S.L. Lok, Variational and momentum preservation aspects of smooth particle hydrodynamic formulations, Comput. Method. Appl. M. 180 (1999) 97–115.

[4] J.M. Cherfils, Développements et applications de la méthode SPH aux écoulements visqueux surface libre, Ph.D. thesis, Université du Havre, 2011.

[5] A. Colagrossi, A meshless Lagrangian method for free-surface and interface flows with fragmentation, Ph.D. thesis, University of Roma La Sapienza, 2005.

[6] A. Colagrossi, M. Landrini, Numerical simulation of interfacial flows by smoothed particle hydrodynamics, J. Comput. Phys. 191 (2003) 448–475.

[7] S.J. Cummins, M. Rudman, An sph projection method, J. Comput. Phys. 152 (1999) 584–607.

[8] R.A. Dalrymple, O. Knio, Sph modelling of water waves, in: ASCE Conf. Proc., vol. 260, ASCE, Lund, Sweden, 2001, p. 80.

[9] A. Ferrari, M. Dumbser, E.F. Toro, A. Armanini, A new 3d parallel sph scheme for free surface flows, Comput. Fluids 38 (2009) 1203–1217.

[10] R.A. Gingold, J.J. Monaghan, Smoothed particle hydrodynamics: Theory and application to non-spherical stars, Mon. Not. R. Astron. Soc. 181 (1977) 375–389.

[11] M. Greco, A two-dimensional study of green-water loading, Ph.D. thesis, Norwegian University of Science and Technology, 2001.

[12] S. Kulasegaram, J. Bonet, R.W. Lewis, M. Profit, A variational formulation based contact algorithm for rigid boundaries in two-dimensional sph applications, Comput. Mech. 33 (2004) 316–325.

[13] D. Le Touzé, A. Colagrossi, G. Colicchio, Ghost technique for right angles applied to the solution of benchmarks 1 and 2, in: 1st International SPHERIC SPH Workshop, Rome.

[14] E.S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence, P. Stansby, Comparisons of weakly compressible and truly incompressible algorithms for the sph mesh free particle method, J. Comput. Phys. 227 (2008) 8417–8436.

[15] L.D. Libersky, A.G. Petschek, T.C. Carney, J.R. Hipp, F.A. Allahdadi, High strain Lagrangian hydrodynamics: A three-dimensional sph code for dynamic material response, J. Comput. Phys. 109 (1993) 67–75.

[16] G.R. Liu, M.B. Liu, Smoothed Particle Hydrodynamics: A Meshfree Particle Method, World Scientific Publishing, 2003.

[17] M.B. Liu, W.P. Xie, G.R. Liu, Modeling incompressible flows using a finite particle method, Appl. Math. Model. 29 (2005) 1252–1270.

[18] L.B. Lucy, A numerical approach to the testing of the fission hypothesis, Astron. J. 82 (1977) 1013–1024.

[19] S. Marrone, M. Antuono, A. Colagrossi, G. Colicchio, D. Le Touzé, G. Graziani, [delta]-sph model for simulating violent impact flows, Comput. Method. Appl. M. 200 (2011) 1526–1542.

[20] D. Molteni, A. Colagrossi, A simple procedure to improve the pressure evaluation in hydrodynamic context using the sph, Comput. Phys. Commun. 180 (2009) 861–872.

[21] J.J. Monaghan, Simulating free surface flows with sph, J. Comput. Phys. 110 (1994) 399–406.

[22] J.J. Monaghan, Smoothed particle hydrodynamics, Rep. Prog. Phys. 68 (2005) 1703–1759.

[23] J.P. Morris, A study of the stability properties of smooth particle hydrodynamics, Publications Astronomical Society of Australia 13 (1996) 97–102.

[24] J.P. Morris, P.J. Fox, Y. Zhu, Modeling low Reynolds number incompressible flows using sph, J. Comput. Phys. 136 (1997) 214–226.

[25] G. Oger, Aspects théoriques de la méthode SPH et applications l'hydrodynamique surface libre, Ph.D. thesis, Ecole Centrale de Nantes, 2006.

[26] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in Fortran 77 – The Art of Scientific Computing, Cambridge University Press, 1992.

[27] D.J. Price, Smoothed particle hydrodynamics and magnetohydrodynamics, J. Comput. Phys. 231 (2012) 759–794.

[28] V. Roubtsova, R. Kahawita, The sph technique applied to free surface flows, Comput. Fluids 35 (2006) 1359–1371.
[29] I.J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions. Part a – on the problem of smoothing or graduation. A first class of analytic approximation formulae, Q. Appl. Math. 4 (1946) 45–99.
[30] S. Shao, E.Y.M. Lo, Incompressible sph method for simulating Newtonian and non-Newtonian flows with a free surface, Adv. Water. Resour. 26 (2003) 787–800.
[31] J.W. Swegle, S.W. Attaway, On the feasibility of using smoothed particle hydrodynamics for underwater explosion calculations, Comput. Mech. 17 (1995) 151–168.
[32] H. Takeda, S.M. Muyama, M. Sekiya, Numerical simulation of viscous flow by smoothed particle hydrodynamics, Progress of Theoretical Physics 92 (1994) 939–960.
[33] J.P. Vila, On particle weighted methods and smooth particle hydrodynamics, Math. Models Methods Appl. Sci. 9 (1999) 161–209.
[34] D. Violeau, R. Issa, Numerical modelling of complex turbulent free-surface flows with the sph method: An overview, Int. J. Numer. Meth. Fl. 53 (2007) 277–304.
[35] Z.Q. Zhou, J.O. De Kat, B. Buchner, A nonlinear 3-d approach to simulate green water dynamics on deck, in: Proc. of the 7th International Conference on Numerical Ship Hydrodynamics, Nantes, France, pp. 1–15.