INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2004; **60**:1475–1512 (DOI: 10.1002/nme.1011)

On the Galerkin formulation of the smoothed particle hydrodynamics method

L. Cueto-Felgueroso, I. Colominas[*,†], G. Mosqueira, F. Navarrina and M. Casteleiro

Group of Numerical Methods in Engineering, GMNI, Department of Applied Mathematics, Civil Engineering School, Universidad de La Coruña, Campus de Elviña, La Coruña 15192, Spain

SUMMARY

In this paper, we propose a Galerkin-based smoothed particle hydrodynamics (SPH) formulation with moving least-squares meshless approximation, applied to free surface flows. The Galerkin scheme provides a clear framework to analyse several procedures widely used in the classical SPH literature, suggesting that some of them should be reformulated in order to develop consistent algorithms. The performance of the methodology proposed is tested through various dynamic simulations, demonstrating the attractive ability of particle methods to handle severe distortions and complex phenomena. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: computational fluid dynamics; particle methods; SPH; free surface flows; Galerkin method

1. INTRODUCTION

Meshless methods have experimented an intense development in the last decade. Their potential seems to be such that many researchers feel to be living a period of pre-revolutionary activity in computational mechanics (if Kuhn's conception is applicable), even though it is not clear where the definitive fracture may come from.

Particle methods are (too) frequently regarded as *numerical models* rather than *numerical methods*, well suited to give physically reasonable *qualitative* solutions in complex problems, but not *quantitatively* accurate solutions, even in simple situations. Much work has been devoted in recent years to banish such a prejudice.

[*]Correspondence to: I. Colominas, Escuela Tecnica Superior de Ingenieros de Caminos, Canales y Puertos, Universidad de La Coruña, Campus de Elviña, La Coruña 15192, Spain.
[†]E-mail: colominas@iccp.udc.es

Contract/grant sponsor: Ministerio de Ciencia y Tecnología; contract/grant number: DPI 2001-0556
Contract/grant sponsor: Xunta de Galicia; contract/grant number: PGDIT01PXI11802PR
Contract/grant sponsor: Xunta de Galicia; contract/grant number: PGIDIT03PXIC118002PN

Received 25 July 2003
Revised 10 October 2003
Accepted 29 October 2003

In the basis of meshless formulations, we find specific interpolation techniques, such as kernel estimates [1] or moving least-squares approximations [2]. However, meshless methods in computational mechanics are not simply different interpolation schemes but constitute, indeed, a powerful and ambitious attempt to solve the equations of continuum mechanics without the computational workload associated to the explicit partition of the domain into certain non-overlapping cells. This essentially distinctive nature of meshless methods is, in turn, the origin of many well-known shortcomings. Unlike finite elements, the absence of a spatial framework complicates the application of certain numerical methodologies such as the weighted residuals (Galerkin) method. Nevertheless, we must note that these techniques, nowadays the strongest basis to develop practical implementations of meshless methods, may be not as adequate for particle methods as they are for finite element methods. Certainly, the development of the mathematical knowledge would provide better techniques to obtain the discrete equations in computational mechanics, taking out the whole power of the particle approach.

The smoothed particle hydrodynamics (SPH) method was developed in the late 1970s to simulate fluid dynamics in astrophysics [3, 4]. The extension to solid mechanics was introduced by Libersky *et al.* [5] and Randles [6]. Johnson and Beissel proposed a normalized smoothing function (NSF) algorithm [7] and other corrected SPH methods have been developed by Bonet *et al.* [8, 9] and Chen *et al.* [10]. More recently, Dilts has introduced moving least-squares (MLS) shape functions into SPH computations [11].

Early SPH formulations included both a new approximation scheme and certain characteristic discrete equations (the so-called SPH equations), which may look quite 'esoteric' for those researchers with some experience in methods with a higher degree of formalism such as finite elements. The formulation described in this paper follows a different approach, and the discrete equations are obtained using a Galerkin weighted residuals scheme. This derivation may result somewhat disconcerting for those accustomed to the classical SPH equations. However, we believe that Galerkin formulations provide a strong framework to develop consistent algorithms. Note that we use moving least-squares shape functions and a Galerkin formulation: is this SPH or EFG? It is not our purpose to carry out a 'taxonomic' study of meshless methods. Perhaps the correct question is: how many (really different) meshless methods there exist? Whatever the answer, we feel that our endeavour follows the spirit of Lucy, Gingold and Monaghan's smoothed particle hydrodynamics.

The outline of the paper is as follows. We begin with a brief review of standard SPH and moving least-squares approximations. After introducing the model equations, their discrete counterpart is obtained using a Galerkin formulation, and various important computational issues, with special emphasis on numerical integration, are addressed. Finally, the methodology is applied to the simulation of fluid dynamics and free surface flows.

2. MESHLESS APPROXIMANTS

2.1. Standard SPH shape functions

The simplest way to construct meshless test and trial functions corresponds to the standard SPH approximants, given by

$$N_j(\mathbf{x}) = V_j W_j(\mathbf{x}) = V_j W(\mathbf{x} - \mathbf{x}_j, h) \tag{1}$$

In the above expression, $W(\mathbf{x}-\mathbf{x}_j, h)$ is a kernel (smoothing) function with compact support centred at particle j and V_j is the tributary or statistical 'volume' associated to particle j. The parameter h, usually called *smoothing length* in the SPH literature or *dilation parameter* in the RKPM literature [12], is a certain characteristic measure of the size of the support of W_j (e.g. the radius in circular supports). Exponential and spline functions are most frequent kernels. The SPH approximation $\hat{u}(\mathbf{x})$ of a given function $u(\mathbf{x})$ can be posed in terms of the shape functions (1) and certain particle or *nodal parameters* $\{u_j\}$ as

$$\hat{u}(\mathbf{x}) = \sum_{j=1}^{n} N_j(\mathbf{x}) u_j = \sum_{j=1}^{n} V_j W_j(\mathbf{x}) u_j \tag{2}$$

Using standard kernels, the approximation given by (2) is poor near boundaries, and lacks even zeroth-order completeness, i.e.

$$\sum_{j=1}^{n} N_j(\mathbf{x}) \neq 1 \tag{3}$$

Thus, the set $\{N_j(\mathbf{x}),\ j=1,\dots,n\}$, where n is the total number of particles, does not constitute a signed partition of unity in the sense of Duarte [13]. The gradient of $\hat{u}(\mathbf{x})$ is evaluated as

$$\nabla_{\mathbf{x}}\hat{u}(\mathbf{x}) = \sum_{j=1}^{n} \nabla_{\mathbf{x}} N_j(\mathbf{x}) u_j = \sum_{j=1}^{n} V_j \nabla_{\mathbf{x}} W_j(\mathbf{x}) u_j \tag{4}$$

Alternative expressions are frequent in the SPH literature (see, for example, References [14, 15]) to enforce conservation properties in the discrete equations, which are not assured by the approximation scheme.

2.2. Moving least-squares-based shape functions

Other meshless interpolation schemes have been proposed. Although different in their formulation, kernel-based approximants (Moving least squares, reproducing kernel particle method) can be seen as corrected SPH methods, and in practice they are very similar.

Within this approach both standard moving least squares (MLS) [2] and moving least-squares reproducing kernel (MLSRKPM) [12] shape functions are analysed. Let us consider a function $u(\mathbf{x})$ defined in a bounded, or unbounded, domain Ω. The basic idea of the MLS approach is to approximate $u(\mathbf{x})$, at a given point $\mathbf{x}$, through a polynomial least-squares fitting of $u(\mathbf{x})$ in a neighbourhood of $\mathbf{x}$ as

$$u(\mathbf{x}) \approx \hat{u}(\mathbf{x}) = \sum_{i=1}^{m} p_i(\mathbf{x}) \alpha_i(\mathbf{z})|_{\mathbf{z}=\mathbf{x}} = \mathbf{p}^{\mathrm{T}}(\mathbf{x}) \boldsymbol{\alpha}(\mathbf{z})|_{\mathbf{z}=\mathbf{x}} \tag{5}$$

where $\mathbf{p}^{\mathrm{T}}(\mathbf{x})$ is an m-dimensional polynomial basis and $\boldsymbol{\alpha}(\mathbf{z})|_{\mathbf{z}=\mathbf{x}}$ is a set of parameters to be determined, such that they minimize the following error functional:

$$J(\boldsymbol{\alpha}(\mathbf{z})|_{\mathbf{z}=\mathbf{x}}) = \int_{\mathbf{y}\in\Omega_{\mathbf{x}}} W(\mathbf{z}-\mathbf{y}, h)|_{\mathbf{z}=\mathbf{x}} [u(\mathbf{y}) - \mathbf{p}^{\mathrm{T}}(\mathbf{y}) \boldsymbol{\alpha}(\mathbf{z})|_{\mathbf{z}=\mathbf{x}}]^2 \, \mathrm{d}\Omega_{\mathbf{x}} \tag{6}$$

being $W(\mathbf{z}-\mathbf{y}, h)|_{\mathbf{z}=\mathbf{x}}$ a symmetric kernel with compact support (denoted by $\Omega_{\mathbf{x}}$), frequently chosen among the kernels used in standard SPH. As mentioned before, h is the smoothing length, which measures the size of $\Omega_{\mathbf{x}}$. The stationary conditions of J with respect to $\boldsymbol{\alpha}$ lead to

$$\int_{\mathbf{y}\in\Omega_{\mathbf{x}}} \mathbf{p}(\mathbf{y}) W(\mathbf{z}-\mathbf{y}, h)|_{\mathbf{z}=\mathbf{x}} u(\mathbf{y})\, \mathrm{d}\Omega_{\mathbf{x}} = \mathbf{M}(\mathbf{x})\boldsymbol{\alpha}(\mathbf{z})|_{\mathbf{z}=\mathbf{x}} \tag{7}$$

where the moment matrix $\mathbf{M}(\mathbf{x})$ is

$$\mathbf{M}(\mathbf{x}) = \int_{\mathbf{y}\in\Omega_{\mathbf{x}}} \mathbf{p}(\mathbf{y}) W(\mathbf{z}-\mathbf{y}, h)|_{\mathbf{z}=\mathbf{x}} \mathbf{p}^{\mathrm{T}}(\mathbf{y})\, \mathrm{d}\Omega_{\mathbf{x}} \tag{8}$$

In numerical computations, the global domain Ω is discretized by a set of n particles. We can then evaluate the integrals in (7) and (8) using those particles inside $\Omega_{\mathbf{x}}$ as quadrature points (nodal integration) to obtain, after rearranging,

$$\boldsymbol{\alpha}(\mathbf{z})|_{\mathbf{z}=\mathbf{x}} = \mathbf{M}^{-1}(\mathbf{x})\mathbf{P}_{\Omega_{\mathbf{x}}}\mathbf{W}_V(\mathbf{x})\mathbf{u}_{\Omega_{\mathbf{x}}} \tag{9}$$

where the vector $\mathbf{u}_{\Omega_{\mathbf{x}}}$ contains certain nodal parameters of those particles in $\Omega_{\mathbf{x}}$, the discrete version of $\mathbf{M}$ is $\mathbf{M}(\mathbf{x}) = \mathbf{P}_{\Omega_{\mathbf{x}}}\mathbf{W}_V(\mathbf{x})\mathbf{P}^{\mathrm{T}}_{\Omega_{\mathbf{x}}}$, and matrices $\mathbf{P}_{\Omega_{\mathbf{x}}}$ and $\mathbf{W}_V(\mathbf{x})$ can be obtained as

$$\mathbf{P}_{\Omega_{\mathbf{x}}} = (\mathbf{p}(\mathbf{x}_1) \;\; \mathbf{p}(\mathbf{x}_2) \;\; \cdots \;\; \mathbf{p}(\mathbf{x}_{n_{\mathbf{x}}})) \tag{10}$$

$$\mathbf{W}_V(\mathbf{x}) = \mathrm{diag}\{W_i(\mathbf{x}-\mathbf{x}_i)V_i\}, \quad i = 1, \ldots, n_{\mathbf{x}} \tag{11}$$

Complete details can be found in Reference [12]. In the above equations, $n_{\mathbf{x}}$ denotes the total number of particles within the neighbourhood of point $\mathbf{x}$ and V_i and $\mathbf{x}_i$ are, respectively, the tributary volume (used as quadrature weight) and co-ordinates associated to particle i. Note that the tributary volumes of neighbouring particles are included in matrix $\mathbf{W}_V$, obtaining an MLS version of the reproducing kernel particle method (the so-called MLSRKPM) [16]. Otherwise, we can use $\mathbf{W}$ instead of $\mathbf{W}_V$,

$$\mathbf{W}(\mathbf{x}) = \mathrm{diag}\,\{W_i(\mathbf{x}-\mathbf{x}_i)\}, \quad i = 1, \ldots, n_{\mathbf{x}} \tag{12}$$

which corresponds to the classical MLS approximation (in the nodal integration of the functional (6), the same quadrature weight is associated to all particles). Introducing (9) in (5) the interpolation structure can be identified as

$$\hat{u}(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}(\mathbf{x})\mathbf{M}^{-1}(\mathbf{x})\mathbf{P}_{\Omega_{\mathbf{x}}}\mathbf{W}_V(\mathbf{x})\mathbf{u}_{\Omega_{\mathbf{x}}} = \mathbf{N}^{\mathrm{T}}(\mathbf{x})\mathbf{u}_{\Omega_{\mathbf{x}}} \tag{13}$$

And, therefore, the MLS shape functions can be written as

$$\mathbf{N}^{\mathrm{T}}(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}(\mathbf{x})\mathbf{M}^{-1}(\mathbf{x})\mathbf{P}_{\Omega_{\mathbf{x}}}\mathbf{W}_V(\mathbf{x}) \tag{14}$$

It is most frequent to use a scaled and locally defined polynomial basis, instead of the globally defined $\mathbf{p}(\mathbf{y})$. Thus, if a function is to be evaluated at point $\mathbf{x}$, the basis would be of the form $\mathbf{p}((\mathbf{y}-\mathbf{x})/h)$. The shape functions are, therefore, of the form

$$\mathbf{N}^{\mathrm{T}}(\mathbf{x}) = \mathbf{p}^{\mathrm{T}}(\mathbf{0})\mathbf{M}^{-1}(\mathbf{x})\mathbf{P}_{\Omega_{\mathbf{x}}}\mathbf{W}_V(\mathbf{x}) \tag{15}$$

In the 2D examples shown in this work, a linear polynomial basis $\mathbf{p}((\mathbf{y}-\mathbf{x})/h) = (1, (y_1-x_1)/h, (y_2-x_2)/h)$ was used, where (x_1, x_2) and (y_1, y_2) are, respectively, the Cartesian co-ordinates of **x** and **y**. This basis provides linear completeness, i.e.

$$\sum_{j=1}^{n} N_j(\mathbf{x}) = 1, \quad \sum_{j=1}^{n} \nabla_{\mathbf{x}} N_j(\mathbf{x}) = \mathbf{0} \tag{16}$$

$$\sum_{j=1}^{n} \mathbf{x}_j N_j(\mathbf{x}) = \mathbf{x}, \quad \sum_{j=1}^{n} \mathbf{x}_j \otimes \nabla_{\mathbf{x}} N_j(\mathbf{x}) = \mathbf{I} \tag{17}$$

2.3. The choice of kernel

A wide variety of kernel functions appear in the literature, most of them being spline or exponential functions. We have not found a general criterion for an optimal choice. The following cubic spline has been extensively used [14]:

$$W_j(\mathbf{x}) = W(\mathbf{x} - \mathbf{x}_j, h) = \frac{\alpha}{h^v} \begin{cases} 1 - \frac{3}{2}s^2 + \frac{3}{4}s^3, & s \leqslant 1 \\ \frac{1}{4}(2-s)^3, & 1 < s \leqslant 2 \\ 0, & s > 2 \end{cases} \tag{18}$$

where $s = \|\mathbf{x} - \mathbf{x}_j\|/h$, v is the number of dimensions and α takes the value $2/3, 10/7\pi$ or $1/\pi$ in one, two or three dimensions, respectively. The coefficient α/h^v is a scale factor necessary only if non-corrected SPH interpolation is being used, to assure the normality property $\int W \, \mathrm{d}V = 1$. We do not use it in our MLS computations.

This approach corresponds to radial weights (i.e. the support of the kernel in two/three dimensions is a circle/sphere with radius $2h$). However, 2D an 3D kernels can be constructed as tensor-product weights, where the weighting function in higher dimensions is computed as the product of one-dimensional kernels as

$$W_j(\mathbf{x} - \mathbf{x}_j, h) = \prod_{n=1}^{v} W_j^n(x^n - x_j^n, h^n) \tag{19}$$

where x^n is the nth co-ordinate of particle **x**. In the above expression, we let W_j^n and h^n (the one-dimensional kernel function and its characteristic smoothing length) be different for each dimension. If the same weighting scheme is employed in all dimensions, then the support is a square/cube in 2D/3D.

3. CONTINUUM EQUATIONS

In finite deformation analysis two possible co-ordinate systems can be chosen to describe the continuum under consideration [17, 18]:

- a certain reference configuration (usually an 'initial configuration'). This is called a Lagrangian or material description, and all relevant quantities are referred to a initial problem domain, Ω^0;
- the current continuum configuration. This is called a Eulerian or spatial description; relevant quantities are referred to the current problem domain, Ω.

The former is most frequent in solid mechanics, whereas the latter is typical in fluid mechanics. These two descriptions will lead, in general, to non-equivalent discretizations in particle methods [19].

Let us assume that the behaviour of a continuum could be analysed as if it was governed by the following equations:

(a) *Continuity equation*: Conservation of mass can be written in a material form as an algebraic equation:

$$\rho J = \rho^0 \tag{20}$$

where ρ^0 and ρ are, respectively, the initial and current densities and J is the determinant of the deformation gradient, $J = \det(\mathbf{F})$, $\mathbf{F} = \mathrm{d}\mathbf{x}/\mathrm{d}\mathbf{X}$. In the following, $\mathbf{X}$, $\mathbf{x} = \mathbf{x}(\mathbf{X})$, $\nabla_{\mathbf{X}}$ and $\nabla_{\mathbf{x}}$ denote co-ordinates and gradient operators in the reference and current configurations, respectively. Most SPH codes use a Eulerian rate form for mass conservation

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho\,\mathrm{div}(\mathbf{v}) \tag{21}$$

where $\mathrm{d}\cdot/\mathrm{d}t$ denotes the material time derivative and $\mathrm{div}(\mathbf{v})$ is computed in the current configuration in terms of the velocity gradient tensor $\mathbf{l}$ as [17]

$$\mathrm{div}(\mathbf{v}) = \mathrm{tr}(\mathbf{l}), \quad \mathbf{l} = \frac{\partial \mathbf{v}(\mathbf{x}, t)}{\partial \mathbf{x}} = \nabla_{\mathbf{x}}\mathbf{v} \tag{22}$$

(b) *Momentum equation*: In a Lagrangian description, conservation of linear momentum can be written as

$$\rho^0 \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \nabla_{\mathbf{X}} \cdot \mathbf{P} + \mathbf{b} \tag{23}$$

where $\mathbf{b}$ is the body force per unit volume and $\mathbf{P}$ is the first Piola–Kirchhoff stress tensor. Its Eulerian counterpart is

$$\rho \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma} + \mathbf{b} \tag{24}$$

where stresses are now related to the Cauchy stress tensor $\boldsymbol{\sigma}$, and ρ is the current density. Note that, in fluid dynamics, the governing equations include a momentum equation which can be written in an arbitrary Lagrangian–Eulerian (ALE) form as [20]

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}^*\nabla_{\mathbf{x}}\mathbf{v}\right) = \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma} + \mathbf{b} \tag{25}$$

where $\mathbf{v}^*$ is the convective velocity. In finite element analysis, $\mathbf{v}^*$ is defined as the difference between the fluid velocity and the mesh velocity. Further details can be found in the excellent book by Donea and Huerta [21] on finite element methods for flow problems. SPH-like particle methods follow the movement of a set of particles, so $\mathbf{v}^* = \mathbf{0}$ and the convective term in (25) vanishes. This is considered a Lagrangian description of the movement. When we say that (24) is posed in Eulerian form we mean that relevant quantities are referred to the current configuration, although the description is Lagrangian in the aforementioned sense.

(c) *Angular momentum conservation*: We consider neither mass distributions of polar momenta nor magnetizable media.

(d) *Energy equation*: Conservation of energy may also be considered in processes involving heat transfer or other related phenomena:

$$\rho \frac{\mathrm{d}U}{\mathrm{d}t} = \boldsymbol{\sigma} : \mathbf{d} - \operatorname{div}(\mathbf{q}) + \rho Q \tag{26}$$

where U is the internal energy per unit mass, $\mathbf{q}$ the energy flux, Q the thermal source (energy per unit time and mass) and $\mathbf{d}$ the deformation gradient tensor, defined as

$$\mathbf{d} = \tfrac{1}{2}(\nabla_{\mathbf{x}}\mathbf{v} + \nabla_{\mathbf{x}}\mathbf{v}^{\mathrm{T}}) \tag{27}$$

We confine our study to problems governed by Equations (20)–(24). The extension of the methodology is straightforward.

4. DISCRETE EQUATIONS

4.1. Weighted residuals. Test and trial functions

The meshless discrete equations can be derived using a weighted residuals formulation. The discrete counterpart of the Galerkin weak form is almost equivalent to that obtained from kernel estimates [22] such as classical SPH formulations. Furthermore, such an equivalence indicates that SPH can be studied in the context of Galerkin methods. The global weak (integral) form of the spatial momentum equation can be written as

$$\int_{\Omega} \rho \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} \cdot \delta\mathbf{v}\,\mathrm{d}\Omega = -\int_{\Omega} \boldsymbol{\sigma} : \delta\mathbf{l}\,\mathrm{d}\Omega + \int_{\Omega} \mathbf{b} \cdot \delta\mathbf{v}\,\mathrm{d}\Omega + \int_{\Gamma} \boldsymbol{\sigma}\mathbf{n} \cdot \delta\mathbf{v}\,\mathrm{d}\Gamma \tag{28}$$

being Ω the problem domain, Γ its boundary and $\mathbf{n}$ the outward unit normal to the boundary. If $\delta\mathbf{v}$ and $\mathbf{v}$ are approximated by certain test and trial functions $\delta\hat{\mathbf{v}}$ and $\hat{\mathbf{v}}$,

$$\int_{\Omega} \rho \frac{\mathrm{d}\hat{\mathbf{v}}}{\mathrm{d}t} \cdot \delta\hat{\mathbf{v}}\,\mathrm{d}\Omega = -\int_{\Omega} \hat{\boldsymbol{\sigma}} : \delta\hat{\mathbf{l}}\,\mathrm{d}\Omega + \int_{\Omega} \mathbf{b} \cdot \delta\hat{\mathbf{v}}\,\mathrm{d}\Omega + \int_{\Gamma} \hat{\boldsymbol{\sigma}}\mathbf{n} \cdot \delta\hat{\mathbf{v}}\,\mathrm{d}\Gamma \tag{29}$$

The spatially discretized equations are obtained after introducing meshless test and trial functions and their gradients in (29) as

$$\delta\hat{\mathbf{v}}(\mathbf{x}) = \sum_{i=1}^{n} \delta\mathbf{v}_i N_i^*(\mathbf{x}), \quad \nabla\delta\hat{\mathbf{v}}(\mathbf{x}) = \sum_{i=1}^{n} \delta\mathbf{v}_i \otimes \nabla_{\mathbf{x}} N_i^*(\mathbf{x}) \tag{30}$$

$$\hat{\mathbf{v}}(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{v}_j N_j(\mathbf{x}), \quad \nabla\hat{\mathbf{v}}(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{v}_j \otimes \nabla_{\mathbf{x}} N_j(\mathbf{x}) \tag{31}$$

to yield

$$\sum_{i=1}^{n} \delta\mathbf{v}_i \cdot \left\{ \sum_{j=1}^{n} \int_{\Omega} \rho N_i^*(\mathbf{x}) N_j(\mathbf{x}) \frac{\mathrm{d}\mathbf{v}_j}{\mathrm{d}t}\,\mathrm{d}\Omega + \int_{\Omega} \hat{\boldsymbol{\sigma}} \nabla_{\mathbf{x}} N_i^*(\mathbf{x})\,\mathrm{d}\Omega \right.$$
$$\left. - \int_{\Omega} N_i^*(\mathbf{x}) \mathbf{b}\,\mathrm{d}\Omega - \int_{\Gamma} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}}\mathbf{n}\,\mathrm{d}\Gamma \right\} = 0 \tag{32}$$

Thus, for each particle i the following identity must hold:

$$\sum_{j=1}^{n} \int_{\Omega} \rho N_i^*(\mathbf{x}) N_j(\mathbf{x}) \frac{\mathrm{d}\mathbf{v}_j}{\mathrm{d}t} \, \mathrm{d}\Omega = - \int_{\Omega} \hat{\boldsymbol{\sigma}} \nabla_{\mathbf{x}} N_i^*(\mathbf{x}) \, \mathrm{d}\Omega + \int_{\Omega} N_i^*(\mathbf{x}) \mathbf{b} \, \mathrm{d}\Omega + \int_{\Gamma} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}} \mathbf{n} \, \mathrm{d}\Gamma \tag{33}$$

In this paper we follow a Bubnov Galerkin approach and, therefore, $N_j^* = N_j$. The Lagrangian counterpart of (33) (i.e. with quantities referred to the initial configuration Ω^0 and Lagrangian shape functions) can be written as

$$\sum_{j=1}^{n} \int_{\Omega^0} \rho^0 N_i^*(\mathbf{X}) N_j(\mathbf{X}) \frac{\mathrm{d}\mathbf{v}_j}{\mathrm{d}t} \, \mathrm{d}\Omega^0 = - \int_{\Omega^0} \hat{\mathbf{P}} \nabla_{\mathbf{X}} N_i^*(\mathbf{X}) \, \mathrm{d}\Omega^0 + \int_{\Omega^0} N_i^*(\mathbf{X}) \mathbf{b} \, \mathrm{d}\Omega^0 + \int_{\Gamma^0} N_i^*(\mathbf{X}) \hat{\mathbf{P}} \mathbf{n} \, \mathrm{d}\Gamma^0 \tag{34}$$

where $\mathbf{X}$ denote particle co-ordinates in the reference configuration. For convenience, we can write (33) in a compact form

$$\mathbf{M}\mathbf{a} = \mathbf{F}^{\text{int}} + \mathbf{F}^{\text{ext}} \tag{35}$$

where the mass matrix $\mathbf{M} = \{m_{ij}\}$, internal forces $\mathbf{F}^{\text{int}} = \{\mathbf{f}_i^{\text{int}}\}$ and external forces $\mathbf{F}^{\text{ext}} = \{\mathbf{f}_i^{\text{ext}}\}$ are respectively defined by

$$m_{ij} = \int_{\Omega} \rho N_i^*(\mathbf{x}) N_j(\mathbf{x}) \, \mathrm{d}\Omega \tag{36}$$

$$\mathbf{f}_i^{\text{int}} = - \int_{\Omega} \hat{\boldsymbol{\sigma}} \nabla_{\mathbf{x}} N_i^*(\mathbf{x}) \, \mathrm{d}\Omega \tag{37}$$

$$\mathbf{f}_i^{\text{ext}} = \int_{\Omega} N_i^*(\mathbf{x}) \mathbf{b} \, \mathrm{d}\Omega + \int_{\Gamma} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}} \mathbf{n} \, \mathrm{d}\Gamma \tag{38}$$

A completely analogous expression could be derived for the Lagrangian version (34). The internal forces will be related to the field variables through the nominal stress tensor, $\boldsymbol{\sigma}$ or $\mathbf{P}$, and the corresponding constitutive equations.

The MLS shape functions do not vanish on essential boundaries and, therefore, the boundary integral in (38) can be decomposed as

$$\int_{\Gamma} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}} \mathbf{n} \, \mathrm{d}\Gamma = \int_{\Gamma_u} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}} \mathbf{n} \, \mathrm{d}\Gamma_u + \int_{\Gamma_n} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}} \mathbf{n} \, \mathrm{d}\Gamma_n \tag{39}$$

where Γ_u and Γ_n are, respectively, the parts of the boundary where essential and natural boundary conditions are prescribed and $\Gamma = \Gamma_u \cup \Gamma_n$. This feature will be revisited later in Section 5.3.2.

If expression (21) is used for mass conservation, its Galerkin weak form is equivalent to a point collocation scheme and, thus, the continuity equation must be enforced at each particle i,

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = -\rho_i \operatorname{div}(\mathbf{v})_i = -\rho_i \sum_{j=1}^{n} \mathbf{v}_j \cdot \nabla_{\mathbf{x}} N_j(\mathbf{x}_i) \tag{40}$$

where expression (31) for $\nabla \hat{\mathbf{v}}_i$ has been used.

4.2. Numerical integration

4.2.1. Introduction. The final step to obtain a set of discrete equations corresponds to the numerical integration of the weak form. This is a most important issue in meshless methods and is the source of well-known inaccuracies and instabilities [9, 22–24]. Some aspects must be considered when choosing (or designing) a numerical quadrature for particle methods.

- The method should provide reasonable accuracy.
- In Lagrangian hydrodynamics applications (at least), the numerical quadrature should retain the meshless character of the method.
- It should be computationally efficient.

The matter of numerical integration concerns the nature itself of meshless methods, and has received much attention in Galerkin-based meshless formulations such as the element-free Galerkin (EFG) method [2]. However, numerical integration has not been explicitly studied in SPH, probably because nodal integration lies in the basis of its early formulations and the method was considered a collocation method. In the context of SPH, the use of alternative numerical quadratures appeared implicitly within the concept of 'stress-points'. More recently, Belytschko *et al.* [22] have reinterpreted SPH as a nodally integrated Galerkin method. Following a similar approach, we believe that, in the context of Galerkin methods, the question about SPH and numerical integration gains full sense, providing a clear framework to analyse the use of the aforementioned 'stress-points'.

4.2.2. Particle methods and continuum equations. There are important differences between meshless methods and mesh-based methods such as the finite element method, concerning numerical integration, when a global Galerkin weak form is defined over the entire problem domain in a continuum mechanics problem:

- The complexity of the shape functions.
- The absence of a spatial framework to define the integration points and their corresponding weights.

Moving least-squares shape functions and their derivatives are complex functions that, in general, cannot be integrated exactly using numerical quadratures [24]. Moreover, the actual integration domains in the globally defined Galerkin weak form correspond to the intersection between nodal supports. Given an arbitrary set of particles, the definition of quadrature subdomains on the basis of support intersections may constitute a formidable task in dynamic problems, requiring the generation of a complex integration mesh each time step.

However, the second difference is much more important and recalls the question about the 'nature' of meshless methods. Mesh-based methods perform a partition of the domain into certain non-overlapping 'elements', which are *explicitly* representative of a piece of the domain. In domains undergoing very high deformations, these individual elements may suffer from severe distortions, with a dramatic loss of accuracy in the computations. In turn, the mesh of elements provides a natural spatial framework to integrate the globally defined Galerkin weak form, which is split into assembled 'elemental contributions' (calculated element-by-element). Moreover, the discrete equations are established in terms of equilibrium between 'regions' (elements) of the domain, which is consistent with the derivation of the continuum mechanics model.

Unlike elements, particles bear certain masses and volumes of which they are *implicitly* representative (the volumes are *concentrated* at particles regardless of their actual shape). This is a very powerful computational advantage of particle methods, but the spatial framework is no longer preserved. The resulting discrete equations can be seen as force equilibrium between interacting particles, which does not correspond to the fundamentals of continuum mechanics: in this sense, the particle philosophy is not conceptually consistent with the continuum mechanics model. This contradiction is partially eliminated by considering the particles as representative of a certain region; note that, however, Cauchy's fundamental axiom of continuum mechanics establishes that the interactions between regions occur in the form of boundary force densities, but not forces between centroids. Nevertheless, the spatial 'uncertainty' introduced by particles suffices to turn the simple integration of the Galerkin weak form into a challenging numerical problem.

In the following, we present a brief review of various integration techniques widely used by researchers and some comments about our own experience in their practical implementation. We must note that this question is far from being closed and is likely to produce fundamental modifications in particle formulations.

4.2.3. Nodal integration. Nodal integration has been used, at least implicitly, in all SPH formulations, and lies, indeed, in the basis of its early formulation. Obviously, this is the cheapest option and the resulting scheme is truly meshless (no background mesh is needed). The particles are used as quadrature points and the corresponding integration weights are their tributary volumes. Recalling the weak form derived in the previous section, the discrete Eulerian momentum equation can be written as

$$\mathbf{Ma} = \mathbf{F}^{\text{int}} + \mathbf{F}^{\text{ext}} \tag{41}$$

where

$$m_{ij} = \sum_{k=1}^{n} \rho_k N_i^*(\mathbf{x}_k) N_j(\mathbf{x}_k) V_k \tag{42}$$

$$\mathbf{f}_i^{\text{int}} = -\sum_{k=1}^{n} \hat{\boldsymbol{\sigma}}_k \nabla_{\mathbf{x}} N_i^*(\mathbf{x}_k) V_k \tag{43}$$

$$\mathbf{f}_i^{\text{ext}} = \sum_{k=1}^{n} N_i^*(\mathbf{x}_k) \mathbf{b}_k V_k + \sum_{k=1}^{n} N_i^*(\mathbf{x}_k) \hat{\boldsymbol{\sigma}}_k \mathbf{n} A_k \tag{44}$$

In the above, V_k represents the tributary volume associated to particle k. Usual techniques to determine such volumes vary from simple domain partitions to Voronoi diagrams. In the most frequent approach in SPH simulations, the particles are set up with certain initial densities, volumes and, therefore, masses. These *physical* masses $\{M_k\}$ remain constant during the simulation and densities are field variables updated using the continuity equation. Thus, particle volumes are obtained each time step as $V_k = M_k/\rho_k$. Note that, in our formulation, the *real* or *physical* particle masses M_k are different, in general, from the *numerical* masses m_{ij} given by (42), and derived in the Galerkin scheme.

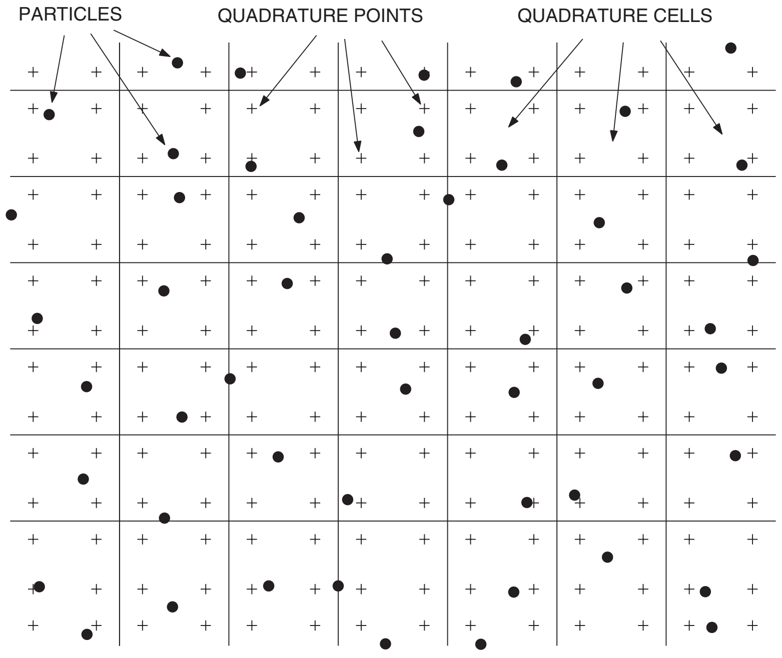

Figure 1. Background integration mesh.

To enforce natural boundary conditions, according to (44), we must first *locate the boundary*, in fact an approximated boundary formed by certain particles, and then determine the boundary weights A_k, associated to each boundary particle and necessary to compute (44).

Nodal integration is the origin of well-known instabilities in meshless methods. Beissel and Belytschko [23], and Bonet and Kulasegaram [9], have analysed the performance of this quadrature in the context of EFG and corrected SPH (CSPH) methods, and proposed modified variational principles based on least-squares stabilizations, requiring second derivatives of the shape functions. Chen and coworkers developed a stabilized conforming nodal integration [25], based on a strain smoothing, which requires the Voronoi diagram of the cloud of particles, at a high computational cost.

4.2.4. Background integration mesh. The most frequent approach in EFG and RKPM is the definition of a background integration mesh, composed by non-overlapping cells covering the whole domain, where high-order Gauss quadratures are defined (Figure 1) [26]. In general, these cells do not match integration domains; however, the *spatial framework* required by the Galerkin method is recovered (at the cost of the generation of an integration mesh). The mass matrix and force vectors are obtained as

$$m_{ij} = \sum_{k=1}^{\text{ninte}} \rho_k N_i^*(\mathbf{x}_k) N_j(\mathbf{x}_k) \mathscr{W}_k \tag{45}$$

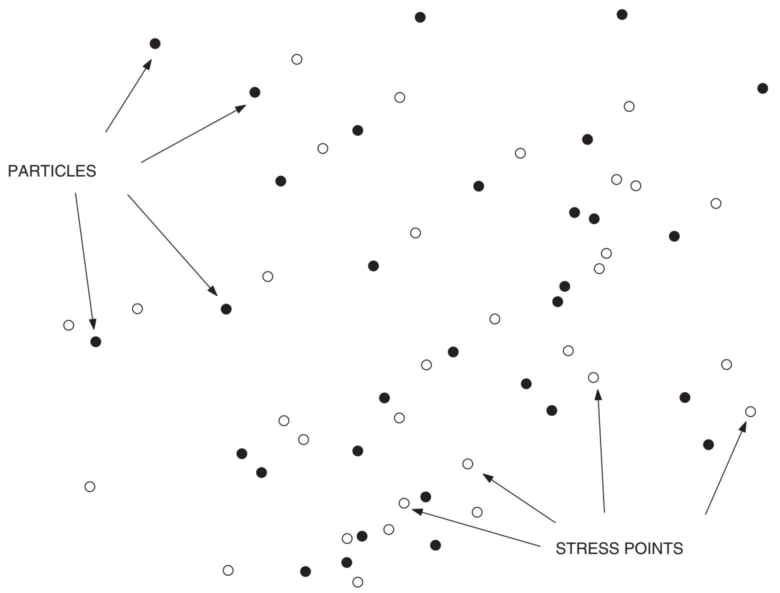

Figure 2. Particles and stress points (double grid).

$$\mathbf{f}_i^{\text{int}} = -\sum_{k=1}^{\text{ninte}} \hat{\boldsymbol{\sigma}}_k \nabla_{\mathbf{x}} N_i^*(\mathbf{x}_k) \mathscr{W}_k \tag{46}$$

$$\mathbf{f}_i^{\text{ext}} = \sum_{k=1}^{\text{ninte}} N_i^*(\mathbf{x}_k)\mathbf{b}_k \mathscr{W}_k + \sum_{k=1}^{\text{ninte}^B} N_i^*(\mathbf{x}_k)\hat{\boldsymbol{\sigma}}_k \mathbf{n} \mathscr{W}_k^B \tag{47}$$

where ninte is the number of integration points (in actual computations only a few points would be considered, according to nodal supports), and $\mathscr{W}_k$ is the quadrature weight of point k. The number of boundary integration points is ninte^B, and $\mathscr{W}_k^B$ is the boundary weight of boundary particle k. This technique, which has been successfully applied to a wide variety of problems in computational mechanics [26], is not appropriate for Lagrangian SPH simulations, where the domain is continuously changing. Instead, a similar but somewhat relaxed scheme has been used in the SPH literature: the so-called 'stress-point' approach.

4.2.5. Stress points. The concept of 'stress points' was introduced by Dyka *et al.* [27], as an attempt to eliminate tensile instabilities [28] in SPH. The basic idea, which still remains in the stress-point SPH literature, is to 'calculate stresses away from the centroids (particles)'. This is equivalent to '*use a quadrature other than nodal integration*' in the Galerkin weak form. Therefore, the discrete equations are completely analogous to (45)–(47), but now the integration points are called *stress points*, which are *moving* integration points spread among the cloud of particles, with no reference to any background mesh (Figure 2). Stress points are set up in certain positions and their movement is completely determined by the movement of the particles, as

$$\hat{\mathbf{v}}_i^{\text{s}} = \sum_{j=1}^{n} \mathbf{v}_j N_j(\mathbf{x}_i^{\text{s}}), \quad \frac{\text{d}\hat{\mathbf{x}}_i^{\text{s}}}{\text{d}t} = \hat{\mathbf{v}}_i^{\text{s}} \tag{48}$$

where the superscript s was used in reference to the stress points. In this context, some authors [22] refer to stress points and particles as 'slave' nodes and 'master' nodes, respectively.

In our implementation, and for consistency with the definition of the test functions, densities are computed at particles through the continuity equation and then interpolated at stress points, as

$$\rho_i^{\mathrm{s}} = \sum_{j=1}^{n} \rho_j N_j(\mathbf{x}_i^{\mathrm{s}}) \tag{49}$$

Note that in the above we assume $\rho_j = \hat{\rho}_j$; that is, we use as *density nodal parameters* ρ_j the *real* nodal values $\hat{\rho}_j$, obtained from the continuity equation. We expect the errors introduced by this assumption to be negligible, particularly in the case of nearly incompressible fluids. Vignjevic *et al.* [29] have proposed a different implementation, where the continuity equation is enforced at stress points. Considering the resulting discrete equations we can readily understand that displacement, velocity and acceleration are tracked at particles, whereas other field variables such as stress are required only at quadrature (stress) points [6, 29].

With this technique, quadrature points move continuously in time: this allows the integration 'mesh' to adapt the moving domain, avoiding the rigid background mesh. We have the positions of quadrature points but, which are their weights? The answer is that the two sets of points are uncoupled. Initially, both represent the total computational domain and particles and integration points are set up with masses, densities and volumes such that

$$\sum_{i\mathrm{p}=1}^{n} V_{i\mathrm{p}} = V, \quad \sum_{i\mathrm{s}=1}^{\mathrm{ns}} V_{i\mathrm{s}} = V \tag{50}$$

$$\sum_{i\mathrm{p}=1}^{n} M_{i\mathrm{p}} = M, \quad \sum_{i\mathrm{s}=1}^{\mathrm{ns}} M_{i\mathrm{s}} = M \tag{51}$$

where M, V, n, ns are the initial mass and volume of the body under study, number of particles and number of quadrature (stress) points, respectively. With time, both sets result in two different computational domains and volumes: particles represent *positions* and *movements* of the body, whereas the set of quadrature points constitutes the *actual* computational domain. Obviously, this domain duplicity must be kept under control to preserve the accuracy of the method.

Belytschko *et al.* [22] have proposed an alternative implementation, where both particles and stress points are used as quadrature points (the stress points are, therefore, *additional* quadrature points) (Figure 3). The internal forces, without boundary tractions, result

$$\mathbf{f}_i^{\mathrm{int}} = -\sum_{i\mathrm{p}=1}^{n} \hat{\boldsymbol{\sigma}}_{i\mathrm{p}} \nabla_{\mathbf{x}} N_i^*(\mathbf{x}_{i\mathrm{p}}) V_{i\mathrm{p}}^{\mathrm{p}} - \sum_{i\mathrm{s}=1}^{\mathrm{ns}} \hat{\boldsymbol{\sigma}}_{i\mathrm{s}} \nabla_{\mathbf{x}} N_i^*(\mathbf{x}_{i\mathrm{s}}) V_{i\mathrm{s}}^{\mathrm{s}} \tag{52}$$

The weights $V_{i\mathrm{p}}^{\mathrm{p}}$ and $V_{i\mathrm{s}}^{\mathrm{s}}$ (different from 'real' volumes) are such that

$$\sum_{i\mathrm{p}=1}^{n} V_{i\mathrm{p}}^{\mathrm{p}} + \sum_{i\mathrm{s}=1}^{\mathrm{ns}} V_{i\mathrm{s}}^{\mathrm{s}} = V \tag{53}$$

The determination of such weights is the most important drawback of this scheme for Eulerian kernels. The Voronoi diagram of the cloud of particles and stress points must be computed

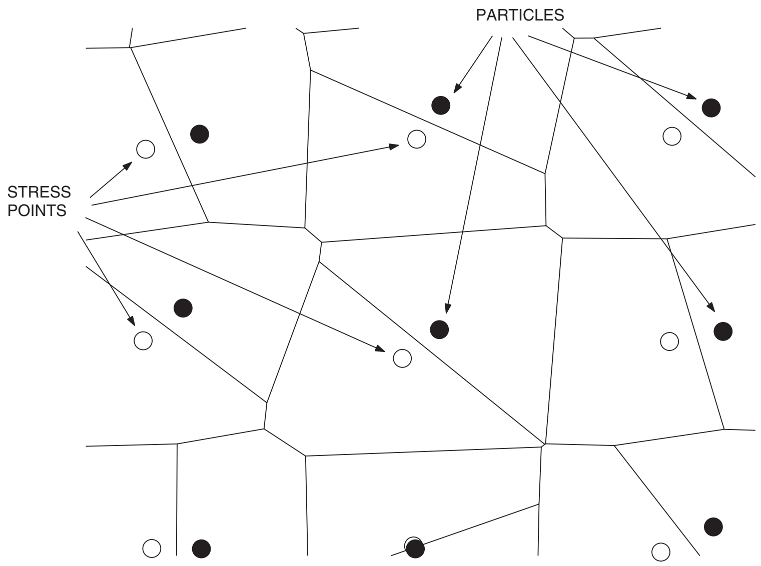

Figure 3. Particles, stress points and Voronoi cells.

at each time step, with an important computational cost. However, if Lagrangian kernels are employed the diagram is computed only in the initial configuration, and the method results quite efficient.

Finally, we would like to propose another implementation of stress points, which is in some sense a 2D extension of the 1D algorithm by Dyka *et al.* [27], where an 'element' was associated to each SPH particle and stresses computed using two integration (stress) points inside each 'element'. In a 2D version of this approach, certain region is associated to each particle and several *representative* points within that region are used as quadrature points. Unlike the 'double grid approach', stress points are now *associated* to particles and represent certain portion of the nodal volume. We would also like to avoid the *explicit* determination of such nodal associated volumes (computing the Voronoi diagram), so an *assumed* nodal region is used instead. Stress points are defined in such regions (Figure 4), and given quadrature weights V_{ik}^{s}, such that

$$\sum_{k=1}^{\mathrm{ns}_i} V_{ik}^{\mathrm{s}} = V_i \tag{54}$$

where ns_i is the number of stress points associated to particle i, V_{ik}^{s} is the weight of stress point k associated to particle i and V_i is the volume of particle i. We assume that the relation V_{ik}^{s}/V_i remains constant throughout the simulation. Particles are usually set up in regular lattices, so the set-up of stress points (the determination of initial assumed nodal associated regions) should be easy. If stress points are moved with the same velocity as their associated particle, the shape of the assumed nodal regions remains constant. This approach may not be adequate under high distortions, so it is better to move the stress points using (48). With this implementation, the movement of stress points provides certain measure of the distortion of the regions associated to particles.

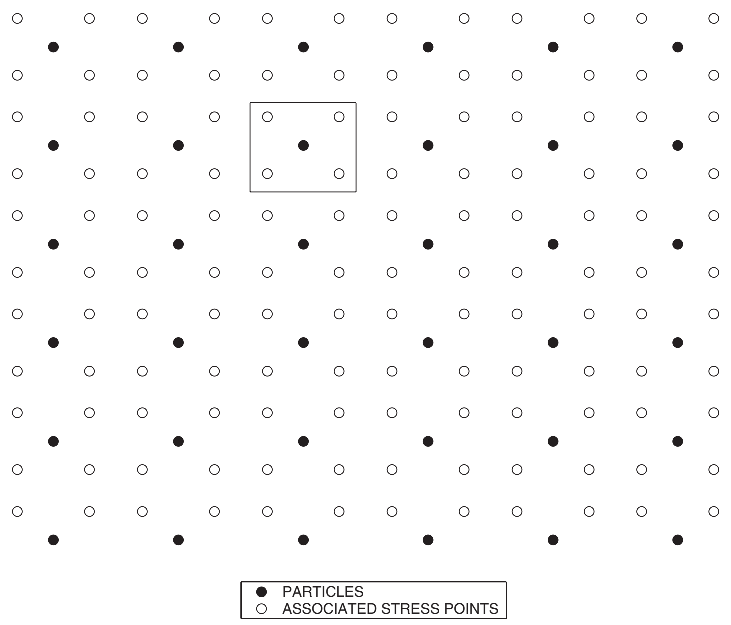

Figure 4. Particles and associated stress points.

4.3. Mass lumping

As it is known in FEM analysis, it is not efficient to use the complete mass matrix in practical applications, and lumped (diagonal) mass matrices are most frequently used. A simple lumping technique corresponds to a row-sum mass matrix. Thus, the lumped mass $\mathscr{M}_i$ associated to particle i is

$$\mathscr{M}_i = \sum_{j=1}^{n} m_{ij} = \sum_{j=1}^{n} \int_\Omega \rho N_i^*(\mathbf{x}) N_j(\mathbf{x})\, \mathrm{d}\Omega = \int_\Omega \rho N_i^*(\mathbf{x}) \left(\sum_{j=1}^{n} N_j(\mathbf{x}) \right) \mathrm{d}\Omega = \int_\Omega \rho N_i^*(\mathbf{x})\, \mathrm{d}\Omega \tag{55}$$

provided that trial functions are, at least, zeroth-order complete. The discrete counterpart of (55), regardless the particular integration method

$$\mathscr{M}_i = \sum_{k=1}^{\text{ninte}} \rho_k N_i^*(\mathbf{x}_k) \mathscr{W}_k \tag{56}$$

where $\mathscr{W}_k$ is the weight of quadrature point k. Note that, if test functions are also zeroth-order complete, this lumping verifies

$$\sum_{i=1}^{n} \mathscr{M}_i = \sum_{i=1}^{n} \sum_{k=1}^{\text{ninte}} \rho_k N_i^*(\mathbf{x}_k) \mathscr{W}_k = \sum_{k=1}^{\text{ninte}} \rho_k \left(\sum_{i=1}^{n} N_i^*(\mathbf{x}_k) \right) \mathscr{W}_k = \sum_{k=1}^{\text{ninte}} \rho_k \mathscr{W}_k = M \tag{57}$$

where M is the total mass of the body. Note the importance of a correct choice of quadrature weights and densities for the integration points in this scheme.

However, (56) is not a common expression in the SPH literature. The most extended practice corresponds to using the *real* particle masses as *numerical* lumped masses, as

$$\mathscr{M}_i = M_i \tag{58}$$

verifying, obviously,

$$\sum_{i=1}^{n} M_i = M \tag{59}$$

5. APPLICATIONS. FLUID DYNAMICS

5.1. Stress tensor

We assume a compressible Newtonian fluid and Eulerian kernels (derivatives and quantities referred to the current configuration). Thus, the internal forces are related to the Cauchy stress tensor, given by

$$\boldsymbol{\sigma} = -p\mathbf{I} + 2\mu\mathbf{d}' \tag{60}$$

where p is the pressure, μ the viscosity and $\mathbf{d}'$ the deviatoric part of the rate of deformation tensor $\mathbf{d}$, given by

$$\mathbf{d}' = \mathbf{d} - \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d})\mathbf{I}, \quad \mathbf{d} = \tfrac{1}{2}\,(\nabla\mathbf{v} + \nabla\mathbf{v}^{\mathrm{T}}) \tag{61}$$

The discrete velocity gradient $\nabla\hat{\mathbf{v}}(\mathbf{x})$ and rate of deformation tensor $\hat{\mathbf{d}}$ are obtained as

$$\nabla\hat{\mathbf{v}}(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{v}_j \otimes \nabla N_j(\mathbf{x}), \quad \hat{\mathbf{d}} = \tfrac{1}{2}(\nabla\hat{\mathbf{v}} + \nabla\hat{\mathbf{v}}^{\mathrm{T}}) \tag{62}$$

Thus, the velocity gradient at a given point $\mathbf{x}$ (in actual computations a particle or quadrature point) is computed in terms of particle velocity parameters $\{\mathbf{v}_j\}$. The velocity divergence is given by

$$\nabla \cdot \hat{\mathbf{v}}(\mathbf{x}) = \mathrm{tr}(\nabla\hat{\mathbf{v}}(\mathbf{x})) = \sum_{j=1}^{n} \mathbf{v}_j \cdot \nabla N_j(\mathbf{x}) \tag{63}$$

We use an equation of state of the form [30]

$$p = \kappa\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{64}$$

where typically $\gamma = 7$ and κ is chosen such that the fluid is nearly incompressible.

In gravity flows the initial particle densities are adjusted to obtain the correct hydrostatic pressure computed as (64) [30]:

$$\rho = \rho_0\left(1 + \frac{\rho_0 g(H - z)}{\kappa}\right)^{1/\gamma} \tag{65}$$

where H is the total depth and $g = 9.81\,\mathrm{m/s^2}$.

5.2. Discrete equations

We can now write the complete spatially discretized set of equations. We assume a Bubnov–Galerkin scheme, where both test and trial functions are chosen from the same space.

- Momentum equation:

$$\mathscr{M}_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{f}_i^{\text{int}} + \mathbf{f}_i^{\text{ext}} \tag{66}$$

  where $\mathscr{M}_i$ is the lumped mass of particle i and $\mathbf{f}_i^{\text{int}}$ and $\mathbf{f}_i^{\text{ext}}$ are, respectively, the internal and external forces, given by

$$\mathbf{f}_i^{\text{int}} = -\sum_{k=1}^{\text{ninte}} \hat{\boldsymbol{\sigma}}_k \nabla N_i(\mathbf{x}_k) \mathscr{W}_k \tag{67}$$

$$\mathbf{f}_i^{\text{ext}} = \sum_{k=1}^{\text{ninte}} N_i(\mathbf{x}_k)\mathbf{b}_k \mathscr{W}_k + \sum_{k=1}^{\text{ninte}^B} N_i(\mathbf{x}_k)\hat{\boldsymbol{\sigma}}_k \mathbf{n} \mathscr{W}_k^B \tag{68}$$

  We prefer this general approach, where ninte is the total number of quadrature points, regardless the particular integration technique chosen. Note that appropriate weights, $\mathscr{W}_k$ and $\mathscr{W}_k^B$, must be defined for interior and boundary quadrature points. The stress tensor must be computed at each quadrature point

$$\hat{\boldsymbol{\sigma}}_k = -\hat{p}_k \mathbf{I} + 2\mu_k \hat{\mathbf{d}}'_k \tag{69}$$

  and $\hat{\mathbf{d}}'_k$ is related to the velocity gradient tensor as expressed in (62).
- Continuity equation:

$$\frac{\mathrm{d}\rho_i}{\mathrm{d}t} = -\rho_i \operatorname{div}(\mathbf{v})_i = -\rho_i \sum_{j=1}^{n} \mathbf{v}_j \cdot \nabla N_j(\mathbf{x}_i) \tag{70}$$

5.3. Consequences of the lack of interpolation property

5.3.1. Moving the particles. Recall the meshless approximation $\hat{u}(\mathbf{x})$ of a function $u(\mathbf{x})$, computed in terms of the shape functions as

$$\hat{u}(\mathbf{x}) = \sum_{j=1}^{n} u_j N_j(\mathbf{x}) \tag{71}$$

where n is the total number of particles and $\{u_j\}$ is a set of nodal parameters. In this context, the interpolated velocity at a given point $\mathbf{x}$ can be obtained from neighbour particles as

$$\hat{\mathbf{v}}(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{v}_j N_j(\mathbf{x}) \tag{72}$$

Note that, unlike finite element interpolants, standard SPH and moving least-squares shape functions do not verify the interpolation property, i.e.

$$N_j(\mathbf{x}_i) \neq \delta_{ij} \tag{73}$$

Thus, in general,

$$\hat{\mathbf{v}}_i = \hat{\mathbf{v}}(\mathbf{x}_i) = \sum_{j=1}^{n} \mathbf{v}_j N_j(\mathbf{x}_i) \neq \mathbf{v}_i \tag{74}$$

and nodal parameters do not necessarily coincide with interpolated values. We suspect this property has introduced some confusion in many SPH formulations where the fact is disregarded. In order to derive a consistent algorithm, the discrete equations must be treated carefully, as they are not as directly meaningful as those obtained with shape functions that bear the interpolation property. The discrete momentum and continuity equations are written in terms of velocities, velocity gradients and accelerations, which are computed using the nodal velocity parameters $\{\mathbf{v}_j\}$, *not the 'real' (interpolated) nodal velocities* $\{\hat{\mathbf{v}}_j\}$. However, before moving the particles, we must compute such *interpolated* velocities

$$\hat{\mathbf{v}}(\mathbf{x}_i) = \sum_{j=1}^{n} \mathbf{v}_j N_j(\mathbf{x}_i) \tag{75}$$

and use $\{\hat{\mathbf{v}}_j\}$ to move the particles according to

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \hat{\mathbf{v}} \tag{76}$$

We must note that moving the particles with $\{\hat{\mathbf{v}}_j\}$ is not a correction, but the *correct* scheme, consistent with the meshless approximation method used. Many SPH practitioners have been using a somewhat modified version of this approach. We refer to the so-called XSPH correction, proposed by Monaghan [30]. To illustrate this point, let us consider MLS shape functions $\{N_j(\mathbf{x})\}$. After computing the nodal parameters $\{\mathbf{v}_j\}$, we move particle i with the velocity

$$\hat{\mathbf{v}}_i = \sum_{j=1}^{n} \mathbf{v}_j N_j(\mathbf{x}_i) = \mathbf{v}_i N_i(\mathbf{x}_i) + \sum_{j\neq i} \mathbf{v}_j N_j(\mathbf{x}_i) \tag{77}$$

Even MLS shape functions with constant basis form a partition of unity and, therefore, the above expression can be written as

$$\hat{\mathbf{v}}_i = \mathbf{v}_i \left(1 - \sum_{j\neq i} N_j(\mathbf{x}_i)\right) + \sum_{j\neq i} \mathbf{v}_j N_j(\mathbf{x}_i) \tag{78}$$

which can be rearranged, to yield

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \sum_{j\neq i} (\mathbf{v}_j - \mathbf{v}_i) N_j(\mathbf{x}_i) = \mathbf{v}_i + \sum_{j=1}^{n} (\mathbf{v}_j - \mathbf{v}_i) N_j(\mathbf{x}_i) \tag{79}$$

Now consider standard SPH shape functions, given by $N_j(\mathbf{x}) = (m_j/\rho_j) W_j(\mathbf{x})$. The above derivation is not correct for such functions, since they do not form a partition of unity,

$$\sum_{j=1}^{n} \frac{m_j}{\rho_j} W_j(\mathbf{x}) \neq 1 \tag{80}$$

Let us write $N_i(\mathbf{x})$ as

$$N_i(\mathbf{x}) = \frac{m_i}{\rho_i} W_i(\mathbf{x}) = 1 - \alpha \sum_{j \neq i} \frac{m_j}{\rho_j} W_j(\mathbf{x}) \tag{81}$$

where α is a certain parameter which may depend on the nodal arrangement, kernel function, etc. Introducing this expression in (77) and rearranging

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \alpha \sum_{j=1}^{n} \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) W_j(\mathbf{x}_i) + (1-\alpha) \sum_{j \neq i} \frac{m_j}{\rho_j} \mathbf{v}_j W_j(\mathbf{x}_i) \tag{82}$$

Using a simplified version of (82) we can write

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \varepsilon \sum_{j=1}^{n} \frac{m_j}{\rho_j} (\mathbf{v}_j - \mathbf{v}_i) W_j(\mathbf{x}_i) \tag{83}$$

where ε is another (unknown) parameter. This expression is completely analogous to the XSPH correction, defined as [30]

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \varepsilon \sum_{j=1}^{n} \frac{m_j}{\rho_{ij}} (\mathbf{v}_j - \mathbf{v}_i) W_j(\mathbf{x}_i) \tag{84}$$

where $\rho_{ij} = 0.5(\rho_i + \rho_j)$. We believe this 'correction' is indeed a *heuristic* version of the consistent form (75). In the XSPH correction, the continuity equation is solved using the corrected velocities given by (84). Note that this procedure is not consistent with our formulation, since the continuity equation is posed in terms of $\{\mathbf{v}_j\}$.

In practice, nodal velocity parameters are used to solve the discrete equations and interpolated nodal velocities are used to move the particles. We would like to emphasize that this approach is *neither a smoothing nor a correction* of the velocities, but the procedure consistent with an approximation scheme which does not bear the interpolation property. Note that the continuity equation is posed in terms of 'real' densities and, therefore, the so obtained densities can be introduced directly in the equation of state to compute pressures.

5.3.2. Enforcement of the essential boundary conditions. The imposition of boundary conditions is a most important and problematic issue in SPH, since they cannot be enforced as directly as in finite elements. Natural boundary conditions are included in the weak form through the boundary traction term. The most important difficulty at this point 'reduces' to the determination of the boundary particles and their weights. In this study, boundary tractions correspond to free surface boundaries and thus, the natural boundary integrals vanish (surface tension is neglected). Unfortunately, the test functions do not vanish on essential boundaries, and a term including tractions on essential boundaries remains in the problem weak form. The external forces in the momentum equation of a fluid particle i are, thus:

$$\mathbf{f}_i^{\text{ext}} = \int_{\Omega} N_i^*(\mathbf{x}) \mathbf{b} \, \mathrm{d}\Omega + \int_{\Gamma_u} N_i^*(\mathbf{x}) \hat{\boldsymbol{\sigma}} \mathbf{n} \, \mathrm{d}\Gamma_u \tag{85}$$

The discrete external forces, in the case of nodal integration, result

$$\mathbf{f}_i^{\text{ext}} = \sum_{j=1}^{n} N_i^*(\mathbf{x}_j) \mathbf{b}_j V_j + \sum_{j \in \mathscr{B}} N_i^*(\mathbf{x}_j) \hat{\boldsymbol{\sigma}}_j \mathbf{n} A_j \tag{86}$$

where $\mathscr{B}$ is the set of particles from the fluid located on the essential boundaries (located 'sufficiently' close to the boundary). Note that the second sum in (86) involves those neighbours of particle i belonging to $\mathscr{B}$.

In addition, given that the MLS shape functions do not bear the interpolation property, velocities cannot be imposed directly. In the SPH literature, essential boundary conditions have been rather treated as fluid–structure contact and specific techniques such as mirror particles and boundary forces have been extensively used [14]. In the classical boundary force approach, certain forces are applied to those particles that approach the boundary (those particles in $\mathscr{B}$). The contact with solid boundaries is detected by checking the distance between fluid particles and certain *boundary particles*, which are fixed at the solid boundary and are not included in the general computations. Consider, for instance, a fluid particle $i \in \mathscr{B}$. The external force vector for particle i:

$$\mathbf{f}_i^{\text{ext}} = \sum_{j=1}^{n} N_i^*(\mathbf{x}_j)\mathbf{b}_j V_j + \sum_{j\in\mathscr{B}} N_i^*(\mathbf{x}_j)\hat{\boldsymbol{\sigma}}_j \mathbf{n} A_j + \sum_{j\in\mathscr{S}} \mathscr{F}_i^j \tag{87}$$

where $\mathscr{S}$ is the set of solid boundary particles and $\mathscr{F}_i^j$ is the boundary force exerted by boundary particle j on i (which may depend on the distance between i and j, their relative velocity, etc.). In most SPH formulations the second term on the right-hand side of (87) (tractions on essential boundaries) is simply dropped, to yield

$$\mathbf{f}_i^{\text{ext}} = \sum_{j=1}^{n} N_i^*(\mathbf{x}_j)\mathbf{b}_j V_j + \sum_{j\in\mathscr{S}} \mathscr{F}_i^j \tag{88}$$

whereas for interior particles ($i \notin \mathscr{B}$),

$$\mathbf{f}_i^{\text{ext}} = \sum_{j=1}^{n} N_i^*(\mathbf{x}_j)\mathbf{b}_j V_j \tag{89}$$

This technique is reformulated here by means of a penalization technique similar to that used in implicit meshless formulations [9]. Thus, the essential boundary conditions are enforced using a penalty boundary potential Π^{bp} given by

$$\Pi^{\text{bp}} = \frac{\eta}{2}\int_{\Gamma_u} (\hat{\mathbf{v}} - \mathbf{v}^{\text{B}})^2 \, \mathrm{d}\Gamma_u \tag{90}$$

where η is a high penalty value and $\mathbf{v}^{\text{B}}$ is the prescribed velocity at boundary Γ_u. The first variation of (90) is

$$\delta\Pi^{\text{bp}} = \eta \int_{\Gamma_u} \delta\hat{\mathbf{v}} \cdot (\hat{\mathbf{v}} - \mathbf{v}^{\text{B}}) \, \mathrm{d}\Gamma_u \tag{91}$$

and this expression should be added to the Galerkin weak form (28). Using nodal integration, the following expression is obtained for the force $\mathbf{f}_i^{\text{bp}}$ over particle i due to the boundary potential (90):

$$\mathbf{f}_i^{\text{bp}} = \eta \sum_{j\in\mathscr{B}} (\hat{\mathbf{v}}_j - \mathbf{v}_j^{\text{B}}) N_i^*(\mathbf{x}_j) A_j \tag{92}$$

The expression

$$\mathscr{F}_j = \eta(\hat{\mathbf{v}}_j - \mathbf{v}_j^{\mathrm{B}})A_j \tag{93}$$

can be interpreted as a *force due to the fact that* j *is close to the boundary* (i.e. that $j \in \mathscr{B}$), and (92) rewritten as

$$\mathbf{f}_i^{\mathrm{bp}} = \sum_{j\in\mathscr{B}} \mathscr{F}_j N_i^*(\mathbf{x}_j) \tag{94}$$

The above expression suggests a new implementation of the boundary force approach for solid boundaries. Thus, the 'forces' $\mathscr{F}_j$ are computed in a fashion similar to that exposed above for the classical boundary force scheme, as

$$\mathscr{F}_j = \sum_{k\in\mathscr{S}} \mathscr{F}_j^k \tag{95}$$

where a general dependence for $\mathscr{F}_j^k$ could be of the form

$$\mathscr{F}_j^k = \mathscr{F}_j^k(\Delta\mathbf{v}_{jk}, \Delta\mathbf{x}_{jk}) \tag{96}$$

being $\Delta\mathbf{v}_{jk} = \hat{\mathbf{v}}_j - \mathbf{v}_k^{\mathrm{B}}$ and $\Delta\mathbf{x}_{jk} = \mathbf{x}_j - \mathbf{x}_k$, where $\mathbf{v}_k^B$ is the velocity prescribed at boundary particle k. The particular expression for $\mathscr{F}_j^k$ used in this study will be presented in Section 6. Using (94) and (95), the external force vector, for a given particle i, is

$$\begin{aligned}\mathbf{f}_i^{\mathrm{ext}} &= \sum_{j=1}^{n} N_i^*(\mathbf{x}_j)\mathbf{b}_j V_j + \sum_{j\in\mathscr{B}} N_i^*(\mathbf{x}_j)\hat{\boldsymbol{\sigma}}_j\mathbf{n}A_j + \sum_{j\in\mathscr{B}} N_i^*(\mathbf{x}_j)\mathscr{F}_j \\ &= \sum_{j=1}^{n} N_i^*(\mathbf{x}_j)\mathbf{b}_j V_j + \sum_{j\in\mathscr{B}} (\mathscr{F}_j + \hat{\boldsymbol{\sigma}}_j\mathbf{n}A_j)N_i^*(\mathbf{x}_j)\end{aligned} \tag{97}$$

Note that, following this approach, boundary forces are not only 'felt' by those particles located immediately close to the boundary, but also by their neighbours. Furthermore, note that the term corresponding to the integral over essential boundaries has been incorporated in the formulation. Given that the penalty forces $\mathscr{F}_j$ are much greater than the terms $\hat{\boldsymbol{\sigma}}_j\mathbf{n}A_j$ we can assume that

$$\mathbf{f}_i^{\mathrm{ext}} = \sum_{j=1}^{n} N_i^*(\mathbf{x}_j)\mathbf{b}_j V_j + \sum_{j\in\mathscr{B}} \mathscr{F}_j N_i^*(\mathbf{x}_j) \tag{98}$$

5.3.3. Initialization of the field variables. Finally, special attention must be paid to the initialization of the field variables, in particular the initial velocity field, known in terms of nodal velocities $\{\hat{\mathbf{v}}_j^0\}$. Once more, in our computations we need the nodal velocity *parameters* $\{\mathbf{v}_j^0\}$, such that

$$\hat{\mathbf{v}}_i^0 = \sum_{j=1}^{n} \mathbf{v}_j^0 N_j(\mathbf{x}_i) \tag{99}$$

and a linear system of equations should be solved for $\{\mathbf{v}_j^0\}$. Fortunately, the linear completeness of the MLS shape functions chosen simplifies certain initial conditions. Consider, for instance, a constant initial velocity field, $\mathbf{v}^0(\mathbf{x}) = \mathbf{v}^0$. Then, we can choose $\mathbf{v}_j^0 = \mathbf{v}^0, \ \forall j = 1, \ldots, n$ because

$$\hat{\mathbf{v}}^0(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{v}_j^0 N_j(\mathbf{x}_i) = \mathbf{v}^0 \sum_{j=1}^{n} N_j(\mathbf{x}) = \mathbf{v}^0 \tag{100}$$

Similarly, given a linear field $\mathbf{v}^0(\mathbf{x}) = \mathbf{C}\mathbf{x}$, where $\mathbf{C}$ is a constant 2×2 matrix,

$$\hat{\mathbf{v}}^0(\mathbf{x}) = \sum_{j=1}^{n} \mathbf{C}\mathbf{x}_j N_j(\mathbf{x}_i) = \mathbf{C} \sum_{j=1}^{n} \mathbf{x}_j N_j(\mathbf{x}) = \mathbf{C}\mathbf{x} \tag{101}$$

Therefore, $\mathbf{v}_j^0 = \mathbf{C}\mathbf{x}_j, \ \forall j = 1, \ldots, n$. The above is true provided that linearly complete shape functions are used. However, even with standard SPH shape functions (which lack zeroth-order completeness), it is a common practice to take $\mathbf{v}_j^0 = \hat{\mathbf{v}}_j^0$. We suspect that, indeed, most SPH formulations assume that $\mathbf{v}_j \equiv \hat{\mathbf{v}}_j$ throughout the computations. This practice, as we would like to demonstrate in this paper, is not correct and causes important errors in the computations.

5.4. Alternative discrete equations

It is frequent in the SPH literature to calculate the velocity gradient as

$$\nabla \hat{\mathbf{v}}_i = \sum_{j=1}^{n} (\mathbf{v}_j - \mathbf{v}_i) \otimes \nabla N_j(\mathbf{x}_i) \tag{102}$$

With linearly complete MLS trial functions

$$\sum_{j=1}^{n} (\mathbf{v}_j - \mathbf{v}_i) \otimes \nabla N_j(\mathbf{x}_i) = \sum_{j=1}^{n} \mathbf{v}_j \otimes \nabla N_j(\mathbf{x}_i) - \mathbf{v}_i \otimes \sum_{j=1}^{n} \nabla N_j(\mathbf{x}_i) = \sum_{j=1}^{n} \mathbf{v}_j \otimes \nabla N_j(\mathbf{x}_i) \tag{103}$$

and (102) is equivalent to (31) (except for round-off errors). Even though in the case of constant velocity fields (102) may eliminate such round-off errors [11], we have not found significant advantages in general problems. If standard SPH interpolation is used, (103) is not true, and both formulations are not equivalent. This symmetrization has been widely used to assure correct velocity gradients for constant velocity fields. However, according to our analysis, this corresponds to a different election of the trial functions, which should be considered, for consistency, wherever trial functions may appear in the formulation.

A similar reasoning could be applied to the case of the internal forces. An expression widely used in the SPH literature is (in the case of nodal integration):

$$\mathbf{f}_i^{\text{int}} = -\sum_{k=1}^{n} (\hat{\boldsymbol{\sigma}}_k \pm \hat{\boldsymbol{\sigma}}_i) \nabla N_i^*(\mathbf{x}_k) V_k \tag{104}$$

most frequently

$$\mathbf{f}_i^{\text{int}} = -\sum_{k=1}^{n} (\hat{\boldsymbol{\sigma}}_k + \hat{\boldsymbol{\sigma}}_i) \nabla N_i^*(\mathbf{x}_k) V_k \tag{105}$$

Standard SPH test functions verify $\nabla N_i^*(\mathbf{x}_k) V_k = -\nabla N_k^*(\mathbf{x}_i) V_i$ and, therefore, the use of (105) assures local conservation of linear momentum. Angular momentum will be also preserved in the absence of shear stresses [8]. Note that, in general, using MLS approximation

$$\sum_{k=1}^{n} \nabla N_i^*(\mathbf{x}_k) \neq \mathbf{0} \tag{106}$$

and

$$\nabla N_i^*(\mathbf{x}_k) V_k \neq -\nabla N_k^*(\mathbf{x}_i) V_i \tag{107}$$

Therefore, (105) is neither skew-symmetric nor consistent with our definition of the test functions (30) and, consequently, we do not use this expression for internal forces.

5.5. Time integration

We use explicit time integration to update the field variables. One of the most widely used algorithms is the leap-frog scheme, involving the following sequence of updates:

- Compute velocities at step $k + \frac{1}{2}$:

$$\mathbf{v}_i^{k+1/2} = \mathbf{v}_i^{k-1/2} + 0.5(\Delta t^k + \Delta t^{k+1})\mathbf{a}_i^k \tag{108}$$

- Update densities and positions:

$$\rho_i^{k+1} = \rho_i^k + \Delta t^{k+1} D_i(\mathbf{v}^{k+1/2}) \tag{109}$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \Delta t^{k+1} \hat{\mathbf{v}}_i^{k+1/2} \tag{110}$$

In the above expressions, $\mathbf{a}_i^k = \mathrm{d}\mathbf{v}_i^k/\mathrm{d}t$ is the acceleration nodal parameter of particle i (computed using the momentum equation with variables at step k) and $D_i(\mathbf{v}^{k+1/2})$ is the density rate $\mathrm{d}\rho_i/\mathrm{d}t$ computed with positions at step k and intermediate velocities $\mathbf{v}_i^{k+1/2}$. With $\hat{\mathbf{v}}_i$ we denote the interpolated nodal velocities, computed as (72).

We have also used the following second-order predictor–corrector scheme, as exposed in Reference [14]. The sequence of updates to obtain the field variables at step $k+1$ is

- Predictions:

$$\mathbf{v}_i^{\mathrm{P}} = \mathbf{v}_i^k + \frac{\Delta t}{2}\mathbf{a}_i^k \tag{111}$$

$$\rho_i^{\mathrm{P}} = \rho_i^k + \frac{\Delta t}{2} D(\mathbf{x}^k, \mathbf{v}^k) \tag{112}$$

$$\mathbf{x}_i^{\mathrm{P}} = \mathbf{x}_i^k + \frac{\Delta t}{2}\hat{\mathbf{v}}_i^k \tag{113}$$

- Corrections:

$$\mathbf{v}_i^{\mathrm{C}} = \mathbf{v}_i^k + \frac{\Delta t}{2}\mathbf{a}_i^{1k}(\mathbf{v}^{\mathrm{P}}, \boldsymbol{\rho}^{\mathrm{P}}, \mathbf{x}^{\mathrm{P}}) \tag{114}$$

$$\rho_i^{\mathrm{C}} = \rho_i^k + \frac{\Delta t}{2} D(\mathbf{x}^{\mathrm{P}}, \mathbf{v}^{\mathrm{C}}) \tag{115}$$

$$\mathbf{x}_i^{\mathrm{C}} = \mathbf{x}_i^k + \frac{\Delta t}{2}\hat{\mathbf{v}}_i^{\mathrm{C}} \tag{116}$$

- Finally, the variables at step $k+1$ result

$$\mathbf{v}_i^{k+1} = 2\mathbf{v}_i^{\mathrm{C}} - \mathbf{v}_i^k \tag{117}$$

$$\rho_i^{k+1} = 2\rho_i^{\mathrm{C}} - \rho_i^k \tag{118}$$

$$\mathbf{x}_i^{k+1} = 2\mathbf{x}_i^{\mathrm{C}} - \mathbf{x}_i^k \tag{119}$$

where the superscripts P and C have been used in reference to the prediction and correction phase, respectively. In practice, we take $\mathbf{a}^k \approx \mathbf{a}^{2(k-1)}$, with a lower cost without changing the order of the method [14]. In both schemes the time step is limited by the Courant–Friedrichs–Lewy (CFL) stability condition. Following Bonet and Lok [8],

$$\Delta t = C_{\mathrm{FL}} \frac{h_{\min}}{\max(c_i + \|\mathbf{v}_i\|)} \tag{120}$$

where C_{FL} is the Courant number ($0 \leqslant C_{\mathrm{FL}} \leqslant 1$) and c_i is the wave celerity at point i,

$$c_i = \sqrt{\gamma\kappa/\rho_i} \tag{121}$$

being γ and κ the same material properties as in (64).

6. IMPLEMENTATION

In this section, we present a schematic flow chart and several remarks about the practical implementation of the methodology exposed above.

Searching for particle neighbours is a key issue in SPH and it is very important to choose an appropriately efficient algorithm. In this work we use a classical cell-partition algorithm, similar to that exposed in Reference [14].

In the examples shown below constant smoothing length is used throughout the computations; a value of $h = 1.5d$, where d is the typical initial distance between particles, seems to be adequate, although we have not found a rigorous criterion for the choice of h. Future versions of the code should allow variable smoothing lengths, but much care must be taken to the consequences of such modification, in order to retain the consistency of the formulation.

The construction of MLS shape functions with linear basis requires the inversion of the moment matrix in (13). This matrix will be singular if there is not enough neighbours or they are aligned (in 2D). So, eventually, the computation of linear MLS shape functions at certain particles may be impossible. In such cases, we use MLS shape functions with constant basis (Shepard functions), which can always be constructed. A similar variable-rank procedure is used in Reference [11]. Note that, even with enough neigbours available, the moment matrix can become highly ill-conditioned and, in that case, the algorithm may become unstable. This latter effect can be alleviated by using adaptable time stepping. In practice, we compute for each particle the determinant or the condition number of the moment matrix and, if it is lower than a certain tolerance, we use constant basis. We have not explored the errors introduced by this implementation, but, given that the number of particles involved (if any) is very small, the effects are expected to be negligible.

6.1. Main algorithm

Let us consider the leap-frog time-integration scheme. The outline of the main steps is as follows:

- Initialize geometry, cell connectivity and field variables: $\mu, \kappa, \mathbf{v}, V, \rho, M, p, c$, etc.
- At each time step:

  I. Find neighbours.
  II. Compute shape functions and their derivatives at quadrature points, according to (14).
  III. Compute stresses at quadrature points, according to (69) and (62).
  IV. Compute internal and external forces, according to (67) and (68). Compute nodal acceleration parameters, $\mathbf{a}$.
  V. Update nodal velocity parameters, as $\mathbf{v}^* = \mathbf{v}^{-1/2} + \mathbf{a}\Delta t$.
  VI. Compute boundary forces, $\mathbf{f}^{\mathrm{b}}$.
  VII. Update nodal velocity parameters, as $\mathbf{v}^{1/2} = \mathbf{v}^* + \mathbf{f}^{\mathrm{b}}\Delta t$.
  VIII. Compute density rates, according to (70).
  IX. Compute the interpolated nodal velocities $\hat{\mathbf{v}}$, used to move the particles, as (72).
  X. Update positions and other field variables.

- End loop.

6.2. Boundary conditions

Throughout the examples shown in this paper, the essential boundary conditions were enforced using the boundary force approach exposed above. Solid boundaries are formed by particles fixed at the boundary, separated one-third of the initial distance between domain particles. At each time step the boundary forces are computed as follows:

- Initialize boundary forces $\mathbf{f}_i^{\mathrm{b}} = \mathbf{0}, \ \forall i = 1, \dots, n$.
- Determine the set $\mathscr{B}$ of particles located 'sufficiently close' to the boundary. In practice, particle j belongs to $\mathscr{B}$ if, and only if, there exists a boundary particle k such that the distance between j and k is less than a certain specified value r_0; in this paper the value $r_0 = 0.7h$ was used. The set of boundary particles k associated in the aforementioned sense to particle j is denoted by $\mathscr{S}_j$.
- For each particle $j \in \mathscr{B}$:
  - For each particle $k \in \mathscr{S}_j$:
    - Compute the boundary force $\mathscr{F}_j^k$ exerted by boundary particle k on j. In the examples shown below, zero normal velocities were imposed. The expression used for boundary forces:

$$\mathscr{F}_j^k = f_1(\Delta \mathbf{v}_{jk})\, f_2(\Delta \mathbf{x}_{jk}) \tag{122}$$

where $\Delta \mathbf{v}_{jk}$ and $\Delta \mathbf{x}_{jk}$ are defined as in (96) and

$$f_1(\Delta \mathbf{v}_{jk}) = \begin{cases} 1, & \Delta \mathbf{v}_{jk} \cdot \mathbf{n}_k < 0 \\ 0, & \Delta \mathbf{v}_{jk} \cdot \mathbf{n}_k \geqslant 0 \end{cases} \tag{123}$$

$$f_2(\Delta \mathbf{x}_{jk}) = \frac{A}{r}\left(\left(\frac{r_0}{r}\right)^4 - \left(\frac{r_0}{r}\right)^2\right)\mathbf{n}_k \tag{124}$$

where $r = \|\Delta \mathbf{x}_{jk}\| = \|\mathbf{x}_j - \mathbf{x}_k\|$, $\mathbf{n}_k$ is the unit normal to the boundary associated to boundary particle k and A is a certain constant. Note that (124) is a modified version of the Lennard-Jones forces [30] (modified in the sense that we consider *normal* and not *radial* forces).

- Add $\mathscr{F}_j^k N_i^*(\mathbf{x}_j)$ to the boundary force $\mathbf{f}_i^{\mathrm{b}}$ applied to each neighbour i of particle j.
- End loop.
- End loop.

Although the proposed implementation fairly outperforms our previous work with classical boundary force approaches in the case of simple non-penetrating boundaries [16, 31–33], we must note that the correct enforcement of more general boundary conditions is still an open problem in SPH.

7. EXAMPLES

In this section several fluid flow simulations are presented. In all the examples shown, nodal integration and the leap-frog time integration have been used, in order to demonstrate that good results can be obtained even with the simplest implementation. The proposed methodology to enforce non-penetrating boundaries was employed with $A = 10$ in all cases except example 7.5, where a lower value of $A = 4$ was used.

7.1. Breaking dam problems

The first example corresponds to a classical breaking dam simulation, with fluid (initially confined between two walls) descending a ramp of slope 0.5 (Figure 5). The total number of particles involved in the simulation is 2500, representing a square water column of side 0.1 m. Particles are set up with densities computed according to (65) and $\rho_0 = 1000\,\mathrm{kg/m^3}$. The fluid

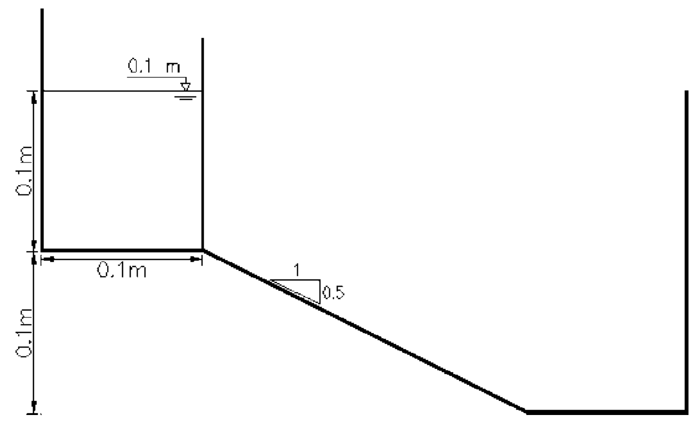

Figure 5. Breaking dam flow: scheme of the initial configuration.

viscosity is $\mu = 0.5\,\text{kg/m/s}$. This a slow flow and, therefore, the use of the 'real' water bulk modulus would lead to very long computations, due to the CFL stability condition. Instead, an artificial bulk modulus is usually employed, computed as [8, 14, 30]:

$$\kappa = \frac{100\rho(2gH)}{\gamma} \tag{125}$$

where H is the height of the dam. Unfortunately, in long simulations this artificial (artificially low) bulk modulus results in an excessively compressible flow, demonstrating the convenience of the development of a fully incompressible algorithm. The evolution of the simulation is shown in Figure 6 and the particle distribution at $t = 0.152\,\text{s}$ is detailed in Figure 7.

7.2. Solitary waves

The propagation of long gravity waves has been widely studied in the last years, given its great interest in hydraulic and coastal engineering. Tsunamis are examples of such waves and shallow water models are often used to perform numerical simulations. No consideration of the vertical dimension is a simplification that reduces the complexity of the problem under study, but that is no longer valid in certain situations [34]. A scheme of the initial configuration of the example presented in this section is shown in Figure 8. The tank contains water at rest in two different levels, and the bore evolves rapidly after the beginning of the simulation (Figure 9). The density of water is $\rho_0 = 1000\,\text{kg/m}^3$ and the viscosity employed is again $\mu = 0.5\,\text{kg/m/s}$. The fluid was modelled using 3980 particles.

7.3. Gravity currents

A gravity current is the flow of a fluid into another fluid of different density [35]. In this example (Figures 10 and 11), we consider a tank with fluid of density $\rho_0^1 = 1000\,\text{kg/m}^3$ (left) separated by a lock gate from another fluid of density $\rho_0^2 = 1500\,\text{kg/m}^3$ (right). The lock is rapidly ('instantaneously') removed and fluid 2 flows under fluid 1. An experimental measurement of the velocity of the head of the current was proposed by Rottman and Simpson (1983, cited by Monaghan *et al.* [35]):

$$v_h \sim 0.4\left(gD\,\frac{\Delta\rho}{\rho_0}\right)^{1/2} \tag{126}$$

where D is the depth of the tank. In this example, $D = 0.06\,\text{m}$, $g = 9.81\,\text{m/s}^2$, $\rho_0 = 1000\,\text{kg/m}^3$ and $\Delta\rho = 500\,\text{kg/m}^3$. Thus, (126) predicts a head velocity of $v_h = 0.217\,\text{m/s}$. In our numerical experiments we have obtained $v_h \sim 0.2\,\text{m/s}$, which is in good agreement with the experimental value. The simulation involved 5100 particles.

7.4. Impacts

In the first example, a circular drop of water falls vertically at $2\,\text{m/s}$ on a mass of water initially at rest (Figures 12 and 13). This example demonstrates the exceptional performance of the method in the absence of boundary distortions (Figure 14). The fluid density and viscosity are $\rho_0 = 1000\,\text{kg/m}^3$ and $\mu = 0.5\,\text{kg/m/s}$, respectively. The total number of particles is 5539.

An example of fluid–structure interaction is shown in Figures 15–17. The total number of fluid particles is 4470.

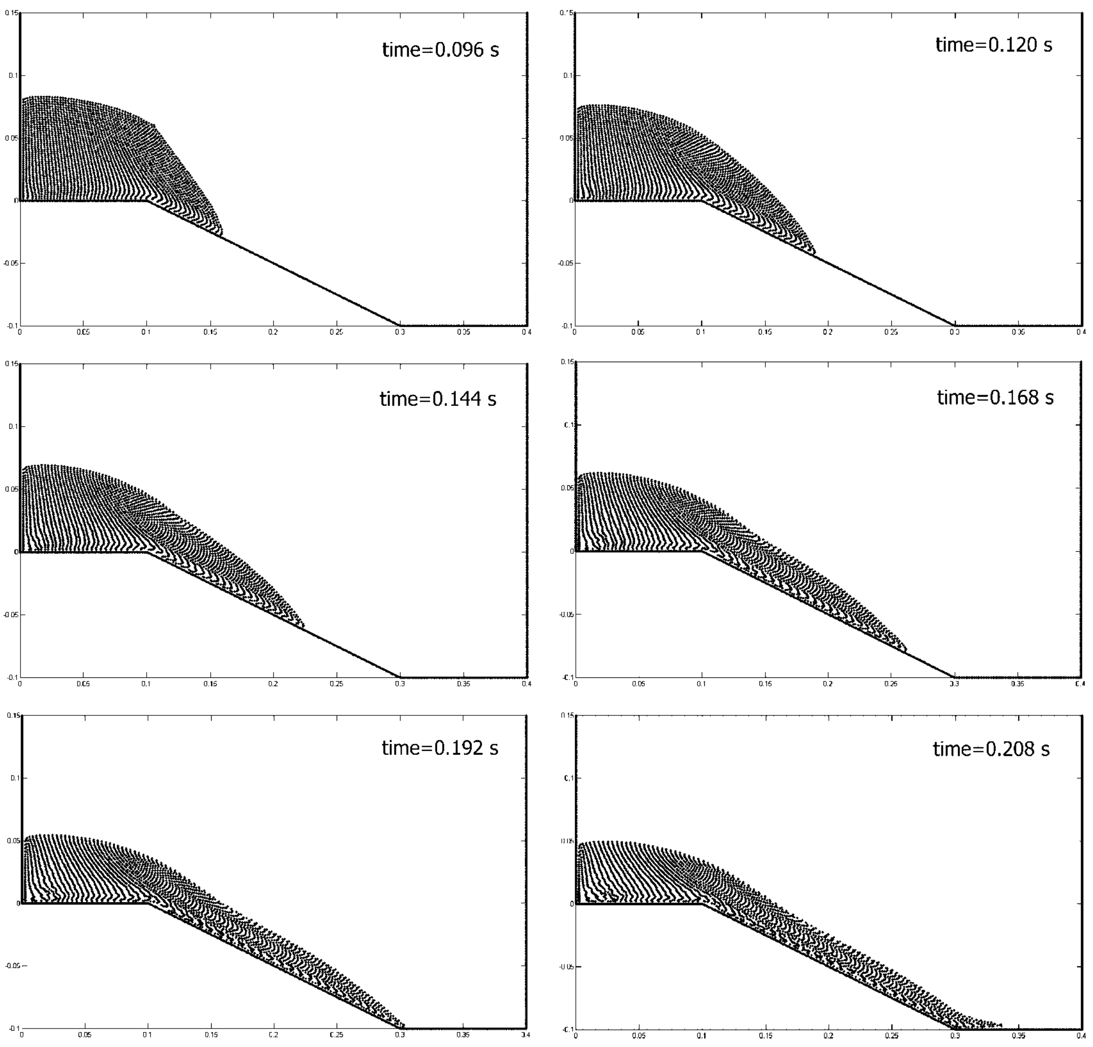

Figure 6. Breaking dam flow: simulation at various stages.

7.5. Mould filling

If previous examples showed the good performance away from the boundaries, the following demonstrates that the formulation proposed is particularly sensitive to the correct (better, to the *incorrect*) enforcement of boundary conditions, which is still a challenging and open problem.

The simulation corresponds to the filling of a circular mould with core (Figure 18). The velocity of the jet at the gate is 18m/s and the viscosity is $\mu = 0.01$ kg/m/s. The bulk modulus κ was chosen such that the wave celerity is 1000m/s and the total number of particles is 14 314. Several instants of the simulation are shown in Figure 19, with times referred to the impact between the jet and the core. The overall shape of the two jets passing the core looks quite

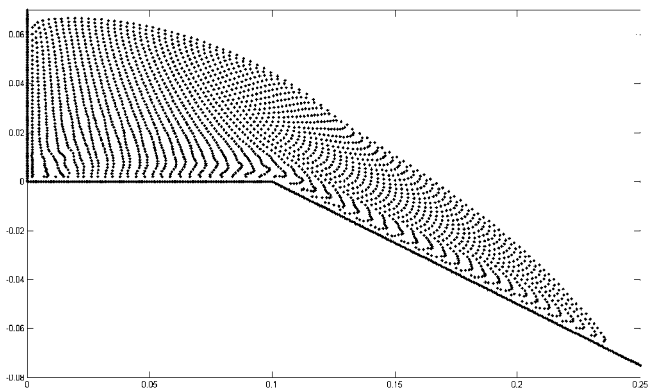

Figure 7. Breaking dam flow: simulation at $t = 0.152$ s with $\mu = 0.5$.

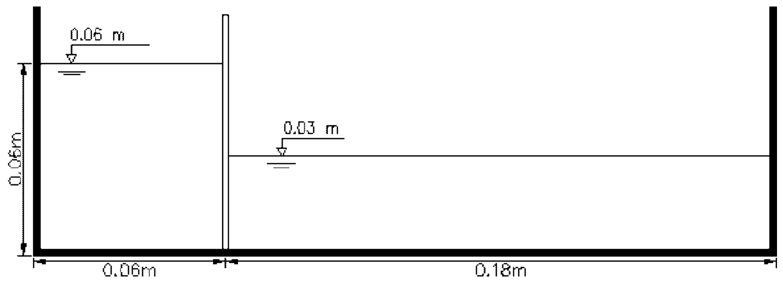

Figure 8. Bore: scheme of the initial configuration.

satisfactory and agree with previous results [36]. In spite of using the consistent formulation of *boundary forces*, we have found excessive distortion near the boundaries, compared with the flow away from their influence (see Figure 20). This effect could be caused by the 'particle based' boundary approach. We expect to develop better algorithms in the future. Figure 21 shows a comparison between the solution computed and the experiments carried out by Schmid and Klein [37].

8. CONCLUSION AND FUTURE DEVELOPMENTS

In this paper, we explored the application of a Galerkin based SPH formulation to the simulation of viscous free surface flows. This approach provides a clear framework to interpret various techniques traditionally used by SPH practitioners such as 'stress-points', boundary forces, special gradient computations, etc.

The start point of our research included an overwhelming amount of questions about the nature and application of meshless methods. And now we feel immensely happy to leave them perfectly unresolved. Much work is still to be done until a complete knowledge of meshless methods could take out their whole power. The set of unresolved questions include: development

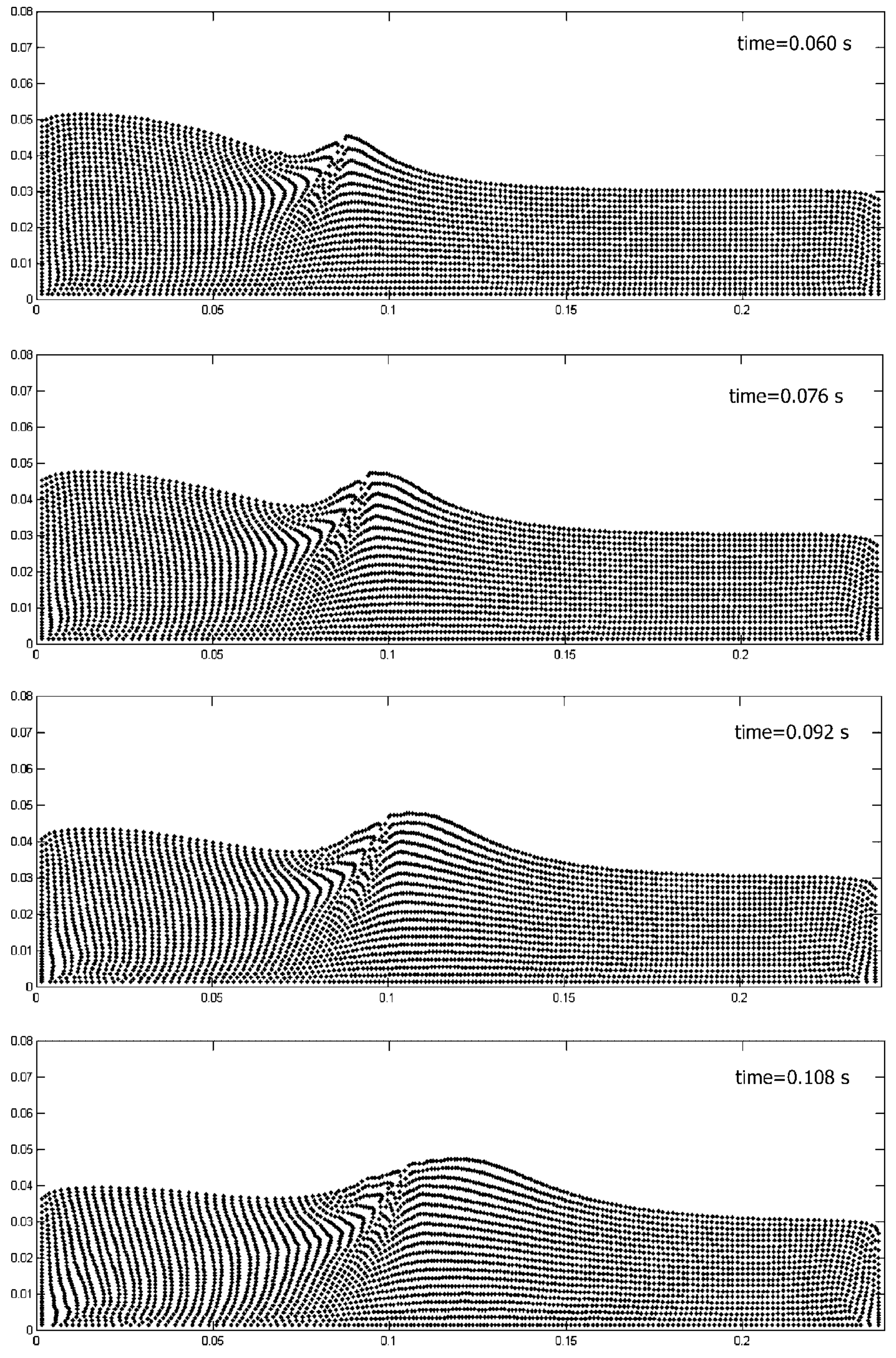

Figure 9. Bore: simulation at various stages.

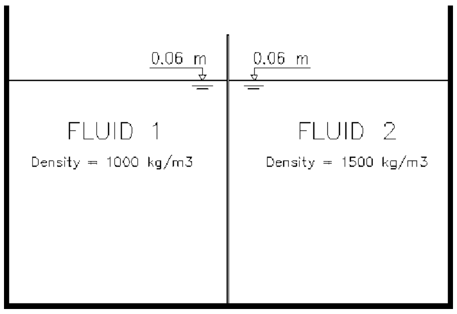

Figure 10. Horizontal gravity current: scheme of the initial configuration.

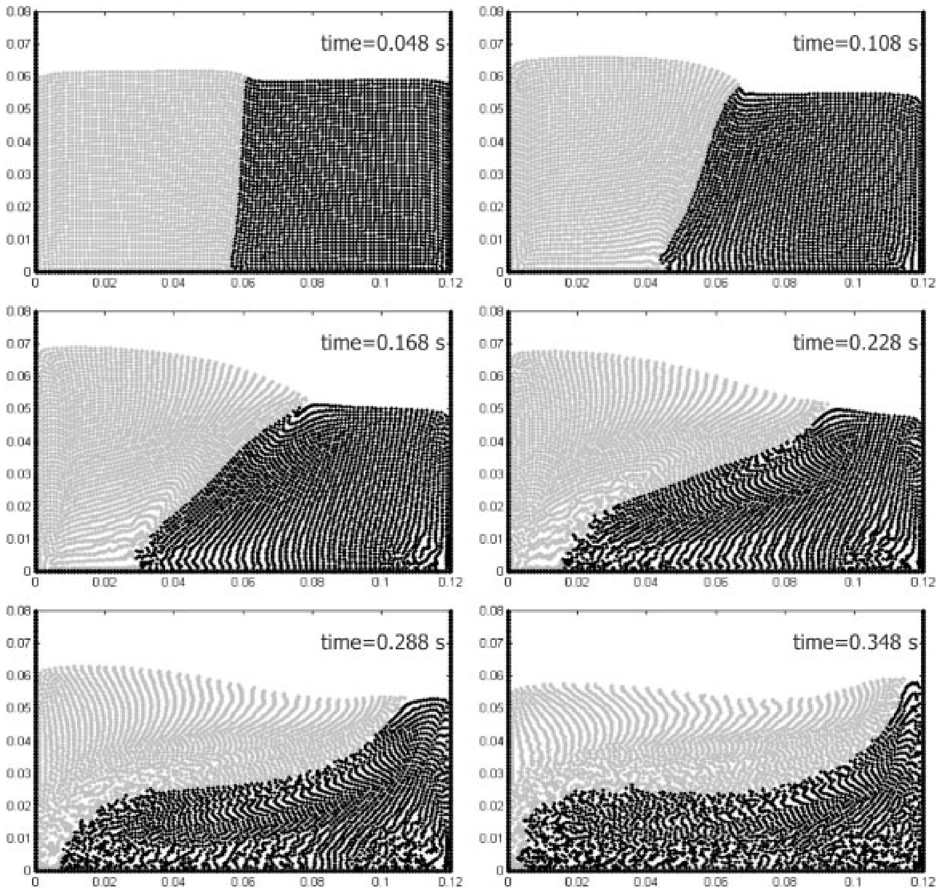

Figure 11. Horizontal gravity current: simulation at various stages.

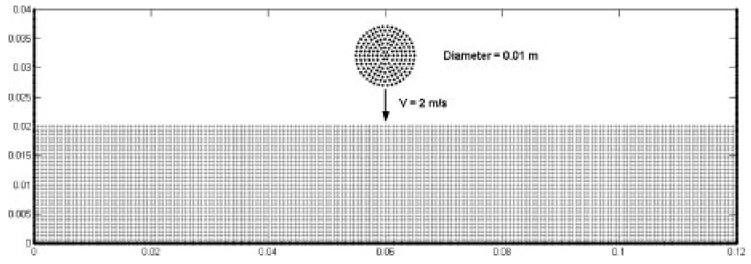

Figure 12. Fluid–fluid impact: scheme of the initial configuration.

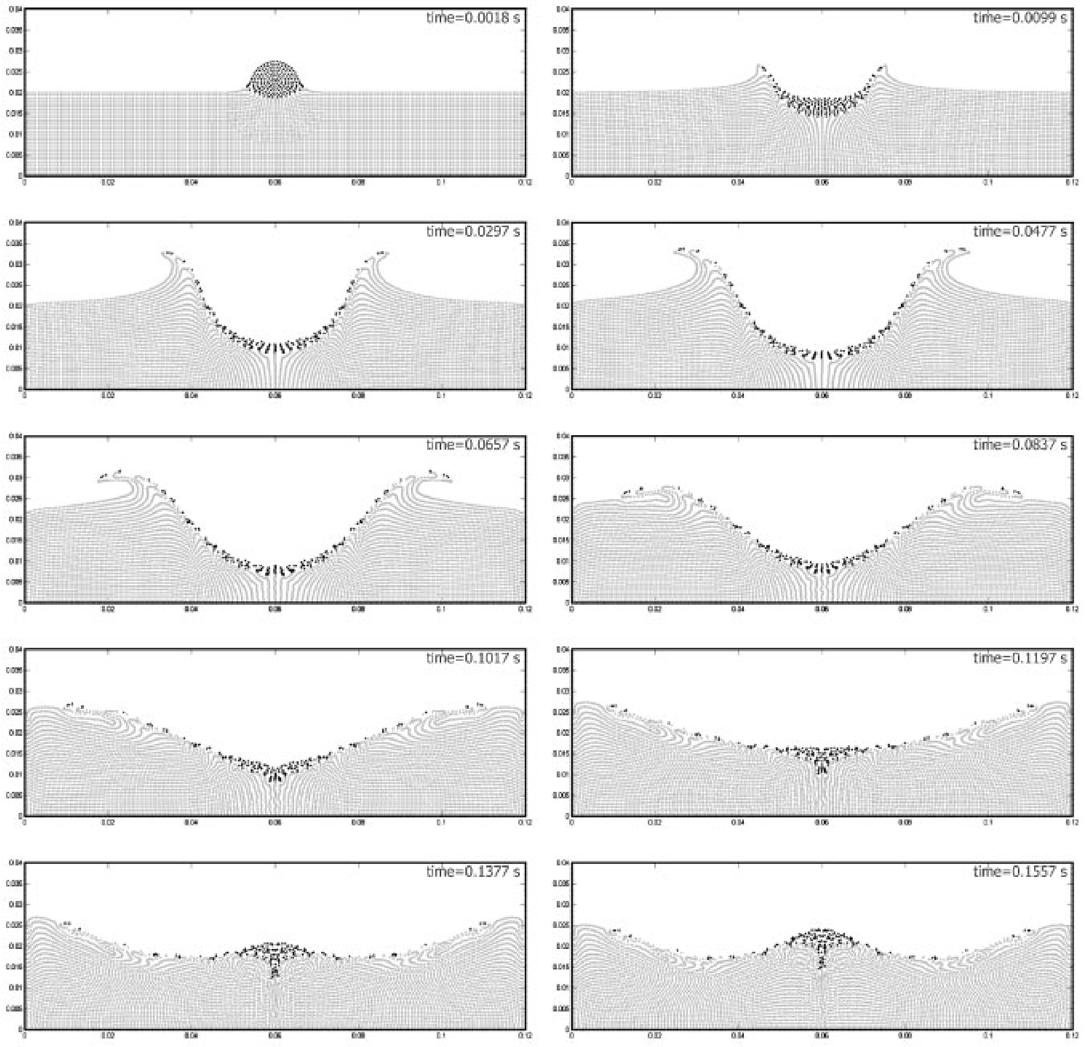

Figure 13. Fluid–fluid impact: simulation at various stages.

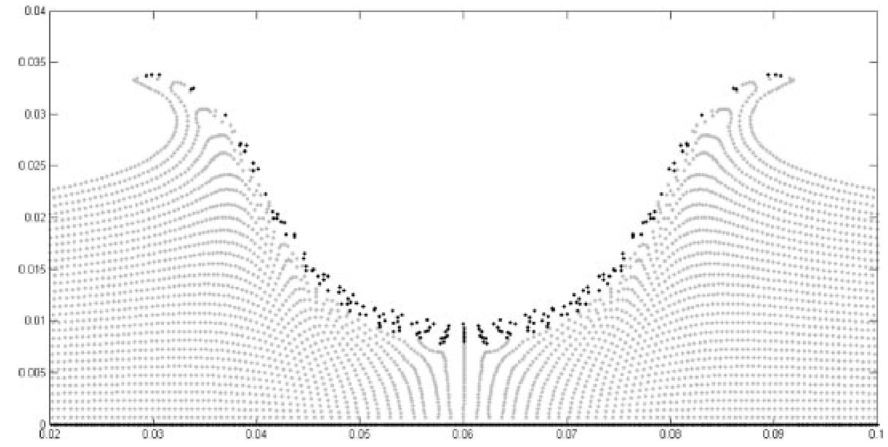

Figure 14. Fluid–fluid impact: simulation at $t = 0.0396$ s (detail).

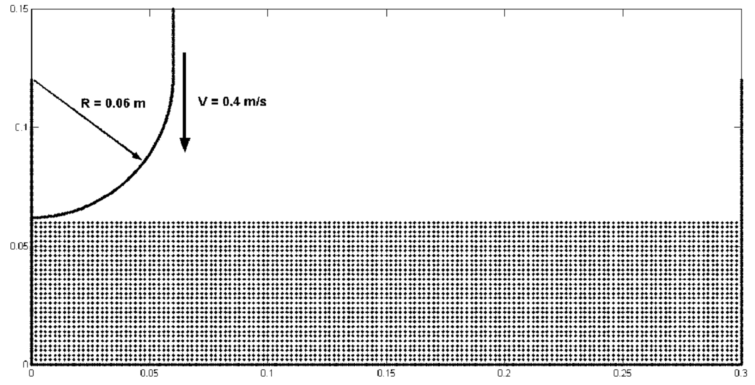

Figure 15. Fluid–structure impact: scheme of the initial configuration.

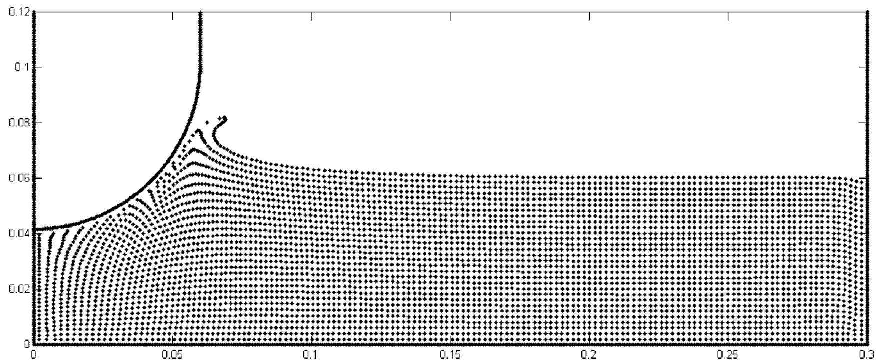

Figure 16. Fluid–structure impact: simulation at $t = 0.051$ s.

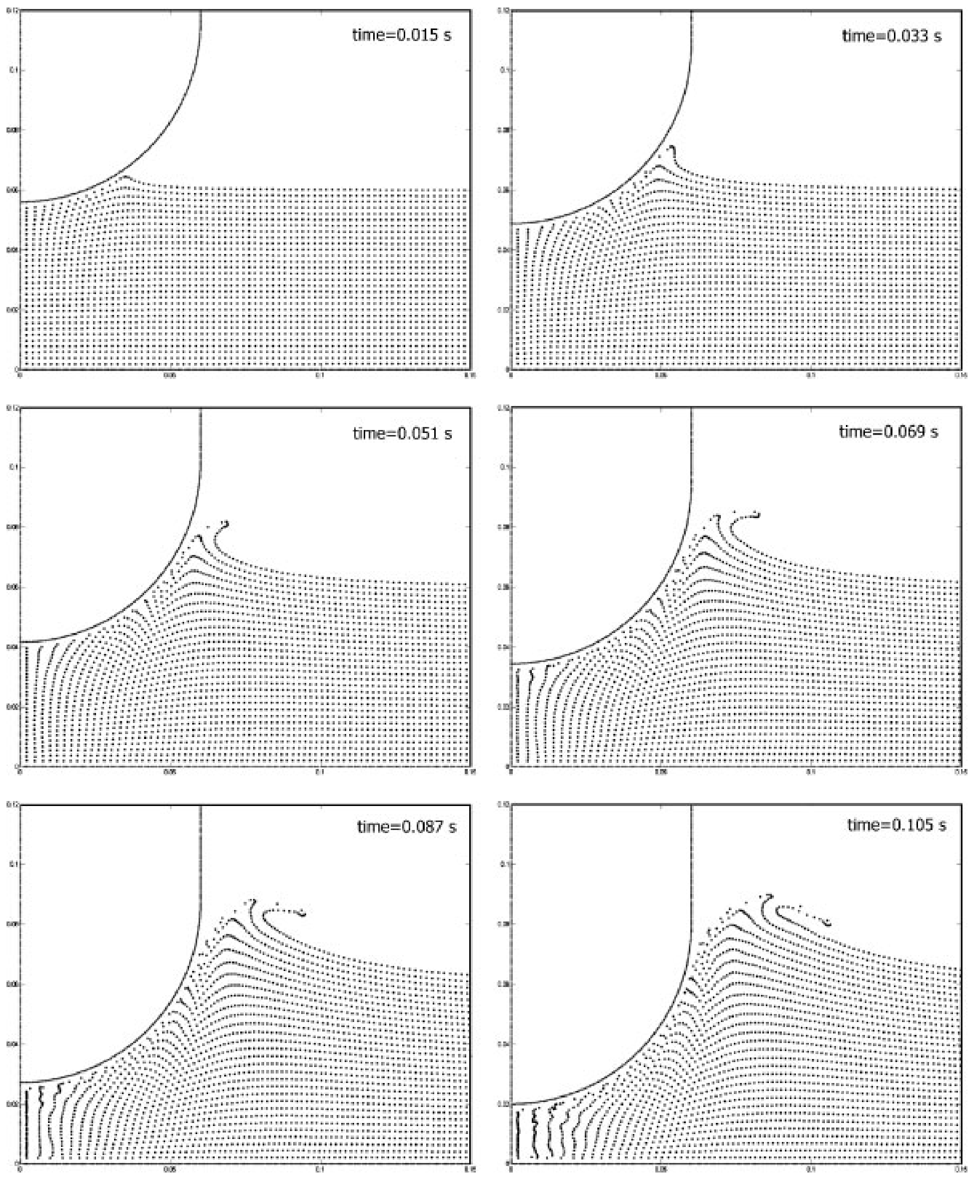

Figure 17. Fluid–structure impact: simulation at various stages (detail).

of an efficient fully incompressible algorithm, adequate enforcement of boundary conditions, error estimations, efficient algorithms with variable smoothing length, numerical integration of the weak form, development of efficient local formulations, etc.

The results obtained so far are not less encouraging in view of all these 'questions' for we keep a 'logical' hope as, if Wittgenstein was right, if a question can at least be formulated, then it can be answered.

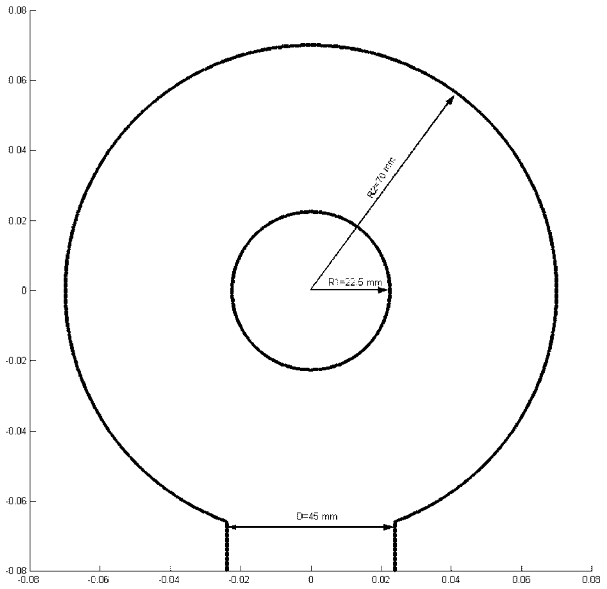

Figure 18. Mould filling: dimensions of the mould.

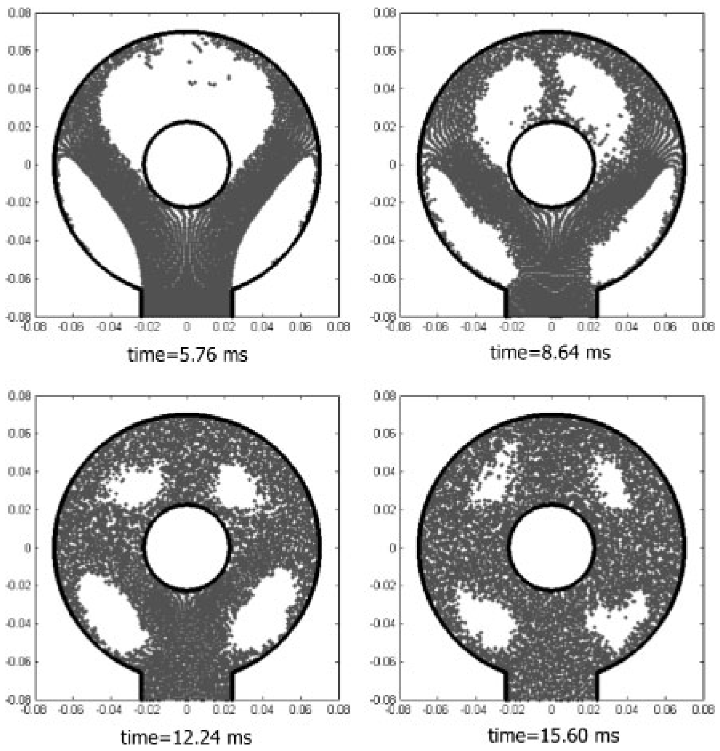

Figure 19. Mould filling: simulation at various stages.

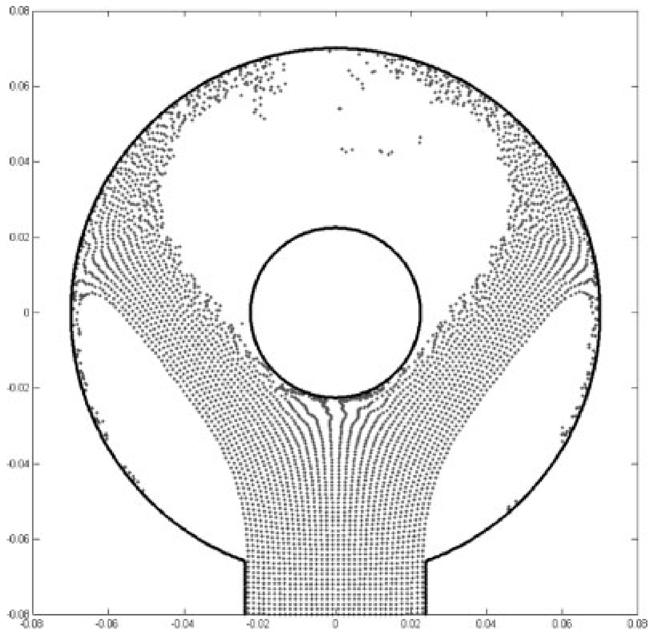

Figure 20. Mould filling: simulation at $t = 5.76\,\text{ms}$.

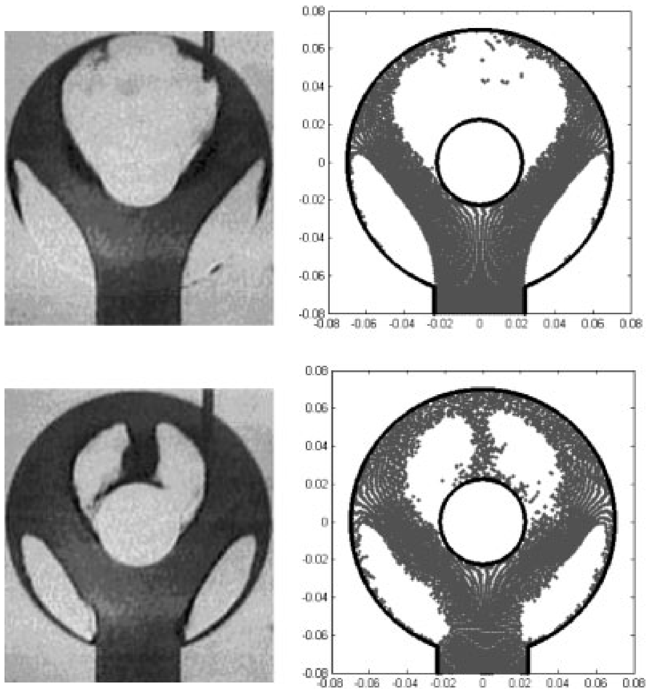

Figure 21. Mould filling: experimental (left) and numerical (right) results.

ACKNOWLEDGEMENTS

This work has been partially supported by the SGPICT of the 'Ministerio de Ciencia y Tecnología' of the Spanish Government (Grant DPI# 2001-0556), the 'Xunta de Galicia' (Grants # PGDIT01PXI11802PR and PGIDIT03PXIC118002PN) and the University of La Coruña.

Mr Cueto-Felgueroso gratefully acknowledges the support received from 'Fundación de la Ingeniería Civil de Galicia' and 'Colegio de Ingenieros de Caminos, Canales y Puertos'.

REFERENCES

1. Monaghan JJ. An introduction to SPH. *Computer Physics Communications* 1988; **48**:89–96.
2. Belytschko T, Lu YY, Gu L. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering* 1994; **37**:229–256.
3. Lucy LB. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal* 1977; **82**:1013.
4. Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 1977; **181**:378.
5. Libersky LD, Petschek AG, Carney TC, Hipp JR, Allahdadi FA. High strain Lagrangian hydrodynamics. *Journal of Computational Physics* 1993; **109**:67–75.
6. Randles PW, Libersky LD. Smoothed particle hydrodynamics: some recent improvements and applications. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:375–408.
7. Johnson GR, Beissel SR. Normalized smoothing functions for SPH impact computations. *International Journal for Numerical Methods in Engineering* 1996; **39**:2725–2741.
8. Bonet J, Lok T-S.L. Variational and momentum preserving aspects of smooth particle hydrodynamics (SPH) formulations. *Computer Methods in Applied Mechanics and Engineering* 1999; **180**:97–115.
9. Bonet J, Kulasegaram S. Correction and stabilization of smoothed particle hydrodynamics methods with applications in metal forming simulations. *International Journal for Numerical Methods in Engineering* 2000; **47**:1189–1214.
10. Chen JK, Beraun JE. A generalized smoothed particle hydrodynamics method for nonlinear dynamic problems. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:225–239.
11. Dilts GA. Moving-least-squares-particle hydrodynamics. *International Journal for Numerical Methods in Engineering* Part I 1999; **44**:1115–1155; Part II 2000; **48**:1503.
12. Liu WK, Li S, Belytschko T. Moving least-square reproducing kernel methods: (I) Methodology and Convergence. *Computer Methods in Applied Mechanics and Engineering* 1997; **143**:113–154.
13. Duarte CA. The hp cloud method. *Ph.D. Thesis*, University of Texas at Austin, 1996.
14. Morris JP. An overview of the method of smoothed particle hydrodynamics. *Internal Report*, Universitat Kaiserslautern, 1995.
15. Belytschko T, Krongauz Y, Dolbow J, Gerlach C. On the completeness of meshfree particle methods. *International Journal for Numerical Methods in Engineering* 1998; **43**:785–819.
16. Cueto-Felgueroso L. A unified analysis of meshless methods: formulation and applications. *Technical Report*, Universidad de La Coruña, 2002, (in Spanish).
17. Bonet J, Wood RD. *Non-linear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press: Cambridge, UK, 1997.
18. Gurtin ME. *An Introduction to Continuum Mechanics*. Academic Press, Inc.: NY, 1981.
19. Bonet J, Kulasegaram S. Remarks on tension instability of Eulerian and Lagrangian corrected smooth particle hydrodynamics (CSPH) methods. *International Journal for Numerical Methods in Engineering* 2001; **52**:1203–1220.
20. Sung J, Choi HG, Yoo JY. Time-accurate computation of unsteady free surface flows using an ALE-segregated equal-order FEM. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:1425–1440.
21. Donea J, Huerta A. *Finite Element Methods for Flow Problems*. Wiley: New York, 2003.
22. Belytschko T, Guo Y, Liu WK, Xiao SP. A unified stability analysis of meshless particle methods. *International Journal for Numerical Methods in Engineering* 2000; **48**:1359–1400.
23. Beissel S, Belytschko T. Nodal integration of the element-free Galerkin method. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:49–74.
24. Dolbow J, Belytschko T. Numerical integration of the Galerkin weak form in meshfree methods. *Computational Mechanics* 1999; **23**:219–230.

25. Chen J-S, Wu C-T, Yoon S, You Y. A stabilized conforming nodal integration for Galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering* 2001; **50**:435–466.
26. Belytschko T, Krongauz Y, Organ D, Fleming M, Krysl P. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:3–47.
27. Dyka CT, Randles PW, Ingel RP. An approach for tension instability in Smoothed Particle Hydrodynamics (SPH). *Computers and Structures* 1995; **57**:573–580.
28. Swegle JW, Hicks DL, Attaway SW. Smoothed particle hydrodynamics stability analysis. *Journal of Computational Physics* 1995; **116**:123–134.
29. Vignjevic R, Campbell J, Libersky LD. A treatment of zero-energy modes in the smoothed particle hydrodynamics method. *Computer Methods in Applied Mechanics and Engineering* 2000; **184**:67–85.
30. Monaghan JJ. Simulating free surface flows with SPH. *Journal of Computational Physics* 1994; **110**: 399–406.
31. Cueto-Felgueroso L, Mosqueira G, Colominas I, Navarrina F, Casteleiro M. Análisis de formulaciones numéricas SPH para la resolución de problemas de flujo en superficie libre. Published In *Métodos Numéricos en Ingeniería V*, Goicolea JM, Mota Soares C, Pastor M, Bugeda G (eds). Sociedad Española de Métodos Numéricos en Ingeniería SEMNI: Barcelona, 2002.
32. Mosqueira G, Cueto-Felgueroso L, Colominas I, Navarrina F, Casteleiro M. SPH approaches for free surface flows in engineering applications. Published In *Fifth World Congress on Computational Mechanics, WCCM V*, Mang HA, Rammerstorfer FG, Eberhardsteiner J (eds). Vienna University of Technology: Austria, http://wccm.tuwien.ac.at, 2002.
33. Cueto-Felgueroso L, Colominas I, Mosqueira G, Navarrina F, Casteleiro M. An MLSPH algorithm for free surface flows in engineering applications. Published In *Computational Fluid and Solid Mechanics 2003*, 873–876, Bathe KJ (ed.). Elsevier: Oxford, UK, 2003.
34. Cea L, Ferreiro A, Vazquez-Cendon ME, Puertas J. An experimental and numerical analysis of solitary waves generated by bed and boundary movements. *Internal Report*, Universidad de La Coruña, 2002.
35. Monaghan JJ, Cas RAF, Kos AM, Hallworth M. Gravity currents descending a ramp in a stratified tank. *Journal of Fluid Mechanics* 1999; **379**:39–69
36. Cleary P, Ha J, Alguine V, Nguyen T. Flow modelling in casting processes. *Applied Mathematics and Modelling* 2002; **26**:171–190
37. Schmid M, Klein F. Fluid flow in die cavities—experimental and numerical simulation, NADCA 18. *International Die Casting Congress and Exposition*, Indianapolis, 1995.