INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2013; **71**:687–716
Published online 15 May 2012 in Wiley Online Library (wileyonlinelibrary.com/journal/nmf). DOI: 10.1002/fld.3680

An efficient edge-based level set finite element method for free surface flow problems

R. Rossi[1,2,*,†], A. Larese[1,2], P. Dadvand[1,2] and E. Oñate[1,2]

[1]*Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE), Barcelona, Spain*
[2]*UPC, BarcelonaTech, Campus Norte UPC, 08034 Barcelona, Spain*

SUMMARY

We present an efficient technique for the solution of free surface flow problems using level set and a parallel edge-based finite element method. An unstructured semi-explicit solution scheme is proposed. A custom data structure, obtained by blending node-based and edge-based approaches is presented so to allow a good parallel performance. In addition to standard velocity extrapolation (for the convection of the level set function), an explicit extrapolation of the pressure field is performed in order to impose both the pressure boundary condition and the volume conservation. The latter is also improved with a modification of the divergence free constrain. The method is shown to allow an efficient solution of both simple benchmark cases and complex industrial examples. Copyright © 2012 John Wiley & Sons, Ltd.

Received 1 August 2011; Revised 17 January 2012; Accepted 30 March 2012

KEY WORDS: edge-based formulation; free surface flows; CFD; level set; fractional step; OpenMP parallel

1. INTRODUCTION

The aim of the current work is to define a method that allows the predictive simulation of free surface flows on complex domains. Free surface flows are relevant in many different areas of engineering, and they are particularly crucial in civil engineering where the correct prediction of the free surface is mandatory for the design of hydraulic structures. Over the years, the importance of the topic motivated a large amount of effort in the development of novel techniques to allow the simulation of the problem. Two basic trends can be identified in the literature: one related to the development of Lagrangian particle-based techniques [1, 2] and a second focusing on the use of Eulerian approaches complemented by methods to allow an accurate tracking of the free surface [3, 4]. Some recent works attempt blending the two approaches [5]. The particle finite element method (PFEM) for example was proved satisfactory for the simulation of a wide class of problems; some of which are very difficult to deal with using any other approach, see for example [6–12]. SPH technique on the other hand enjoyed a widespread success because of its simplicity and high computational efficiency [13, 14]. Eulerian approaches are winners in terms of raw computational efficiency and of robustness, although they are still limited in the range of application compared with their Lagrangian counterparts.

Despite the maturity of the subject, the development of a robust computational model, which allows dealing with complex geometries still represents an open challenge with many different proposals being presented but still no definitive conclusion being reached.

*Correspondence to: R. Rossi, CIMNE, Campus Norte UPC, Edifici C1, c/ Gran Capitá, s/n, 08034 Barcelona, Spain.
†E-mail: rrossi@cimne.upc.edu

Level set and to a minor extent the volume of fluid techniques are known to suffer volume conservation problems, which typically manifest in the total volume of a given fluid not being maintained during long term simulations.

A large number of authors identify the re-initialization of the level set function as a source of this error. Interesting proposals were made to alleviate such problem by enriching the re-initialization step with information on the volume to be conserved either at a global or local level [15, 16].

Other works [17] suggest that improving the formulation in order to respect better the discrete divergence constraint is sufficient to solve the problem.

Following the latter, we also use a level set approach, and we do not make any special effort to improve our convection and re-initialization steps, which are performed using standard techniques. Rather, we strive to represent as accurately as possible the divergence constraint at the node level, taking particular care to the treatment of the free-surface condition, which we consider as the source of our conservation problems. The results we report in the paper seem to indicate the correctness of such assumption.

In this work, a single fluid formulation is used. This implies, in principle, imposing Neumann boundary conditions on the free surface and wetting and drying portions of the control domain. The standard approach in this context is to extrapolate only the velocity field allowing the convection of the level set function. In the present work, we also extrapolate the pressure field, and we strive to use this information to impose both the pressure boundary condition and the volume conservation.

We begin with a presentation of the mathematical model and its discrete counterpart, discussing its modification so to improve volume conservation properties. Then, we introduce and justify the use of a semi-explicit time integrator and discuss an efficient edge-based implementation so to allow the efficient computation of the terms involved. The level set tracking procedure is described next, providing the necessary details for using it in combination with the proposed method.

This is followed by the description of the parallelization procedure. Finally, we demonstrate the efficiency of the approach by presenting a number of simple benchmarks followed by some examples of industrial-grade applications.

Through the whole paper, the use of simplicial meshes is assumed.

The code is implemented inside Kratos multiphysics open source C++ software [18, 19] and is available on-line at *http://kratos.cimne.upc.edu/*.

2. DISCRETIZATION OF THE NAVIER-STOKES EQUATIONS

The first step towards the solution of free-surface problem is the discretization of the Navier-Stokes equations. Because the flow is expected to be incompressible, the non-conservative form of the Navier-Stokes equations is chosen and discretized using the standard Galerkin approach. Defining $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ as the fluid domain in a time interval $(0, T)$, the continuous form of the Navier-Stokes equations is

$$\begin{aligned} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u} + \nabla p - 2\nabla \cdot \nu \nabla^s \mathbf{u} - \mathbf{b} = \mathbf{0} \text{ in } \Omega,\ t \in (0, T); \\ \nabla \cdot \mathbf{u} = 0 \text{ in } \Omega,\ t \in (0, T); \end{aligned} \tag{1}$$

where $\mathbf{u}$ is the velocity, p the pressure, ν the viscosity.

In dealing with free surface problems, it is often important to consider the effect of the surface tension force [20]. The level set function could be used to recover the local curvatures so to evaluate the effect of such force, nevertheless in the current work, this is not taken into account.

The problem is completed by the definition of the following initial and boundary conditions:

$$\begin{aligned} \mathbf{u}(\mathbf{x}, 0) &= \mathbf{u}_0(\mathbf{x}) \quad \text{in} \quad \Omega; \\ \mathbf{u}(\mathbf{x}, t) &= \mathbf{g}(\mathbf{x}, t) \quad \text{on} \quad \partial\Omega_D, \quad t \in (0, T); \\ \mathbf{n} \cdot \boldsymbol{\sigma}(\mathbf{x}, t) &= \mathbf{t}(\mathbf{x}, t) \quad \text{on} \quad \partial\Omega_N, \quad t \in (0, T); \end{aligned} \tag{2}$$

DOI: 10.1002/fld

where $\mathbf{n}$ is the unit normal vector and $\boldsymbol{\sigma} = -p\mathbf{I} + 2\nu\nabla^s\mathbf{u}$. Ω_D and Ω_N are the Dirichlet and Neumann boundary, respectively. The difficulty in modeling the free-surface problem is obviously related to the changing shape of the fluid domain, which makes difficult to impose the Neumann conditions on the free surface.

Applying the Galerkin approach to Equation (1) and applying integration by parts of the stress term, we obtain the following (well-known) variational problem.

$$\begin{aligned}\int_\Omega \mathbf{w}\partial_t\mathbf{u}\mathrm{d}\Omega + \int_\Omega \mathbf{w}\mathbf{u}\cdot\nabla\mathbf{u}\mathrm{d}\Omega - \int_\Omega p\nabla\cdot\mathbf{w}\mathrm{d}\Omega \\ +2\int_\Omega \nabla\mathbf{w} : \nu\nabla^s\mathbf{u}\mathrm{d}\Omega - \int_\Omega \mathbf{w}\mathbf{b}\mathrm{d}\Omega - \int_{\partial\Omega_N} \mathbf{w}\mathbf{t}\mathrm{d}\Gamma = 0 \quad \forall\mathbf{w}\in\mathcal{V}; \\ \int_\Omega q\nabla\cdot\mathbf{u}\mathrm{d}\Omega = 0 \quad \forall q\in\mathcal{Q};\end{aligned} \tag{3}$$

For a fixed $t \in (0,T)$, $\mathbf{u}$ is assumed to belong to the velocity space $\mathcal{V} \in \left[\mathcal{H}_0^1(\Omega)\right]^d$ of vector functions whose components and derivatives are square-integrable and vanish on $\partial\Omega$, whereas the pressure p belongs to the pressure space $\mathcal{Q} \in \mathcal{L}_2$ of square-integrable functions. If we let $\mathcal{V}_h$ be a finite element space to approximate $\mathcal{V}$ and $\mathcal{Q}_h$ a finite element approximation to $\mathcal{Q}$, the problem becomes finding $\mathbf{u}_h \in \mathcal{V}_h$ and $p_h \in \mathcal{Q}_h$ such that

$$\begin{aligned}\int_\Omega \mathbf{w}_h\partial_t\mathbf{u}_h\mathrm{d}\Omega + \int_\Omega \mathbf{w}_h\mathbf{u}_h\cdot\nabla\mathbf{u}_h\mathrm{d}\Omega - \int_\Omega p_h\nabla\cdot\mathbf{w}_h\mathrm{d}\Omega \\ +2\int_\Omega \nabla\mathbf{w}_h : \nu\nabla^s\mathbf{u}_h\mathrm{d}\Omega - \int_\Omega \mathbf{w}_h\mathbf{b}\mathrm{d}\Omega - \int_{\partial\Omega_N} \mathbf{w}_h\mathbf{t}_h\mathrm{d}\Gamma = 0 \quad \forall\mathbf{w}_h\in\mathcal{V}_h; \\ \int_\Omega q_h\nabla\cdot\mathbf{u}_h\mathrm{d}\Omega = 0 \quad \forall q_h\in\mathcal{Q}_h.\end{aligned} \tag{4}$$

The matricial counterpart of such residuum equations can be defined once the following definitions are provided:

$$\begin{aligned}\mathbf{G}_{ij} &:= \int_\Omega \nabla N_i N_j\mathrm{d}\Omega; \\ \boldsymbol{\nabla}_{ij} &:= \int_\Omega N_i\nabla N_j\mathrm{d}\Omega; \\ \mathbf{D}_{ij} &:= \int_\Omega N_i\nabla N_j^T\mathrm{d}\Omega; \longrightarrow \mathbf{D}_{ij} := \mathbf{G}_{ji}^T \\ L_{ij} &:= \int_\Omega \nabla N_i\cdot\nabla N_j\mathrm{d}\Omega; \\ C_{ij}(\mathbf{u}) &:= \int_\Omega N_i(\mathbf{u}_g\cdot\nabla N_j)\mathrm{d}\Omega; \\ M_{ij} &:= \delta_{ij}\sum_j \int_\Omega N_i N_j\mathrm{d}\Omega;\end{aligned} \tag{5}$$

where node j is one of the nodes, which are connected by at least one edge of the finite element mesh to node i and $\mathbf{u}_g$ appearing in the convective operator is a velocity evaluated at the integration point. Note that the Kronecker delta (δ_{ij}) is used in the definition of the mass operator (M) so to obtain its lumped form.

On the basis of such operators, our Navier-Stokes problem becomes finding $\mathbf{u}$ and p such that

$$\mathbf{M}\frac{\partial\mathbf{u}}{\partial t} = \mathbf{r}(\mathbf{u}, p); \tag{6}$$

and

$$\mathbf{D}\mathbf{u} = 0; \tag{7}$$

DOI: 10.1002/fld

with

$$\mathbf{r}(\mathbf{u}, p) := \mathbf{F} - \nu\mathbf{L}\mathbf{u} - \mathbf{C}(\mathbf{u})\mathbf{u} + \mathbf{G}p; \tag{8}$$

We shall observe that in writing Equation (8), the Laplacian form of the viscous operator is taken, which implies a further approximation with respect to Equation (4). This has implications, which become relevant for highly viscous flows [21] but that are completely acceptable for the range of problems of interest and was chosen here because it slightly reduces the computational cost.

Equations (6) and (7) define a 'monolithic' problem in which velocity and pressure are tightly coupled. Furthermore, the problem of interest is likely to be convection dominated, which implies a non-linear dependence upon the velocity. Because our interest is in using low-order simplicial elements, stabilization is needed for both convection and for the incompressibility constraint. All the examples presented in the current work were run using Orthogonal Sub-grid Scale stabilization [22], nevertheless the finite calculus approach [23] was also considered as an alternative. Our choice was to avoid providing any detail of the particular choice made for the stabilization technique, as this is unrelated to the algorithm we propose. The different alternatives considered were verified to provide equivalent results.

The solution of the monolithic problem represented by Equations (6) and (7), represents a computational challenge given the difficulty (and cost) of resolving the resulting linear system. Several techniques exist to simplify it. We will focus on the use of a fractional step (FS) approach, which we will modify so to obtain a semi-explicit version which appears to be convenient for the problem of interest. More specifically we will define an approximate technique for the solution of the free-surface problem, which is well integrated within the solution framework provided by the FS approach. In this sense, we will discuss in detail the treatment of the pressure constraint and the divergence freeness condition of the problem of interest.

2.1. Fractional step solver

Pressure-splitting approaches of the FS type are very convenient because of their high computational efficiency for flows at high Re and have enjoyed widespread popularity since the original works of Chorin [24] and Temam [25]. The fundamental idea is to solve the momentum equation keeping the pressure fixed and later correcting the pressure so to guarantee reproducing correctly the divergence constraints. An algebraic presentation of the method can be found in [26]. The FS approach is traditionally presented in an implicit context, typically using a first or second order backward differentiation formula (BDF1 or BDF2, respectively) algorithm for the time integration of the momentum equation [27]. In dealing with free-surface problems unfortunately the shape of the domain, and consequently the boundary conditions, are subjected to frequent and radical changes which need to be accurately tracked in time in order to allow a satisfactory representation of the solution. In practice, it is typically observed that even fully implicit schemes are practically limited to time steps for which the free surface approximately moves one element length per time step. Such heuristic constraint is equivalent in essence to a restriction on the practical Courant-Friedrichs-Lewy number to values in the order of the unity. This observation effectively implies that explicit scheme will be competitive provided that $CFL \approx 1$ can be used and that meshes of sufficiently good quality can be generated. This motivates us to use an explicit form of the FS scheme (see, e.g., [28]) based on the use of a 4th order Runge Kutta (RK4) method in dealing with the momentum equation.

Before proceeding to the description of our method, we shall observe that the algebraic splitting proposed by Codina [26] leads naturally to the definition of a 'Discrete Laplacian' $\mathrm{DM}^{-1}\mathrm{G}$ which in principle does not introduce any additional error in the imposition of the divergence freeness with respect to the original monolithic scheme.

In practice, however, the use of the 'Discrete Laplacian' implies a large computational burden as the matrix is rather densely populated. In order to overcome such problem, the 'Discrete Laplacian' is typically substituted by a continuum Laplacian, which has a much smaller (around six times smaller in $3D$) stencil than the discrete one. This fact has important consequences both on the

DOI: 10.1002/fld

efficiency and on the stability of the numerical scheme (see, e.g., [26]) but in particular has an important impact on the volume conservation properties of the method [29]. One practical issue is that although the use of a 'Discrete Laplacian' matrix guarantees an invertible matrix, this is not the case when the continuum form is chosen implying that the pressure needs to be fixed on the Neumann boundary, that is, pressure is to be imposed 'strongly', at least for the solution of the pressure step. This fact implies that, when FS is to be used, the pressure is known beforehand on the Neumann boundary.

Given such situation, it is convenient to avoid integrating by parts the pressure gradient term, using the equivalent formula

$$\int_{\Omega} \mathbf{w} \cdot \nabla p \mathrm{d}\Omega = -\int_{\Omega} p \nabla \cdot \mathbf{w} \mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{w} \cdot p\mathbf{n} \mathrm{d}\Gamma \tag{9}$$

This means that the pressure space should be in $[\mathcal{H}^1(\Omega)]^d$ which is an additional requirement to the smoothness of the function. Such modified form has the important advantage that no boundary integrals need to be computed for the pressure (second integral of the right and side of Equation (9)), which leads to an easier application of the pressure boundary condition on the free surface as will be explained in Section 3.4.

This consideration leads us to the following expression of the residual at a given intermediate time step β (note the use of ∇ instead of $\mathbf{G}$).

$$\mathbf{r}^{\beta} = \mathbf{F}^{\beta} - \nu \mathbf{L}\mathbf{u}^{\beta} - \mathbf{C}(\mathbf{u}^{\beta})\mathbf{u}^{\beta} - \nabla p^{\beta}; \tag{10}$$

Remark 1
Using Equation (9) implies a point-wise application of the normal force on the Neumann boundary instead of its weak imposition. This is an approximation that is acceptable for low viscosity flows for which the term $\int_{\Omega} \mathbf{n} \cdot \nu \nabla \mathbf{u} d\Omega$ is negligible [21].

On the base of such definition, we can now proceed to the time integration. The application of the RK4 scheme to the first equation gives (symbolically)

$$\mathbf{M}\frac{\mathbf{u}^{n+1} - \mathbf{u}^{n}}{\Delta t} = \frac{1}{6}\left[\mathbf{r}(\mathbf{u}^{n}, p^{n}) + 2\mathbf{r}\left(\mathbf{u}^{\beta 1}, p^{\beta 1}\right) + 2\mathbf{r}\left(\mathbf{u}^{\beta 2}, p^{\beta 2}\right) + \mathbf{r}\left(\mathbf{u}^{\beta 3}, p^{\beta 3}\right)\right]; \tag{11}$$

where the intermediate velocities are defined as

$$\begin{aligned}
\mathbf{u}^{\beta 1} &= \mathbf{u}^{n} + \mathbf{M}^{-1}\frac{\Delta t}{2}\mathbf{r}(\mathbf{u}^{n}, p^{n}); \\
\mathbf{u}^{\beta 2} &= \mathbf{u}^{n} + \mathbf{M}^{-1}\frac{\Delta t}{2}\mathbf{r}\left(\mathbf{u}^{\beta 1}, p^{\beta 1}\right); \\
\mathbf{u}^{\beta 3} &= \mathbf{u}^{n} + \mathbf{M}^{-1}\Delta t\, \mathbf{r}\left(\mathbf{u}^{\beta 2}, p^{\beta 2}\right);
\end{aligned} \tag{12}$$

The direct application of such procedure would require knowing the pressure at all of the given sub-steps, as pressure should adjust instantly so to guarantee the incompressibility of all of the sub-step velocities. If we instead assume a linear variation of the pressure within the time step, and we accept that only the end-of-step velocities will be divergence free, we arrive to the definition of a FS splitting in the form [28]

- STEP 1:

$$\mathbf{M}\frac{\tilde{\mathbf{u}} - \mathbf{u}^{n}}{\Delta t} = \frac{1}{6}[\mathbf{r}(\mathbf{u}^{n}, p^{n}) + 2\mathbf{r}\left(\mathbf{u}^{\beta 1}, p^{n}\right) + 2\mathbf{r}\left(\mathbf{u}^{\beta 2}, p^{n}\right) + \mathbf{r}\left(\mathbf{u}^{\beta 3}, p^{n}\right)]; \tag{13}$$

  where $\tilde{\mathbf{u}}$ is the fractional step velocity [26].
- STEP 2: (omitting pressure stabilization)

$$\mathbf{L}(p^{n+1} - p^{n}) = \mathbf{D}\tilde{\mathbf{u}} \tag{14}$$

DOI: 10.1002/fld

- STEP 3:

$$\mathbf{u}^{n+1} = \tilde{\mathbf{u}} + \frac{\Delta t}{2}\mathbf{M}^{-1}\nabla(p^{n+1} - p^n); \tag{15}$$

2.1.1. Edge-based approach. Having made the choice of an explicit scheme for the time integration of the momentum equation, we need to devise a data structure, which is suitable for the fast calculation of the residuals.

For a comprehensive and detailed comparison between the cost of an element-based and that of an edge-based approach the consultation of [3] is recommended. The advantage of such technique for this specific case is that the pre-computation of operators allows a faster computation of the residual when compared with standard integration rules.

The starting point in our work is the systematic use of the partition-of-unity property of the finite element shape functions, which provides the relations

$$\sum_i N_i = 1 \implies N_i = 1 - \sum_{j \neq i} N_j \tag{16}$$

and (as a consequence)

$$\sum_i \nabla N_i = 0 \implies \nabla N_i = -\sum_{j \neq i} \nabla N_j \tag{17}$$

The 'edge-based' approach is obtained by applying such relations for the computation of the discrete operators of interest. In the following, we consider one by one the different terms involved in the calculation of the residua, by expressing the contributions to the entry corresponding to a given node i. The j index indicates one of the nodes 'around' i which share an edge with it.

2.1.1.1. Gradient term. The gradient term (not integrated by parts) which appears in the momentum equation, reads

$$\begin{aligned}
\sum_j \int_\Omega N_i \nabla N_j \, p_j \, \mathrm{d}\Omega &= \sum_{j \neq i} \int_\Omega N_i \nabla N_j \, p_j \, \mathrm{d}\Omega + \int_\Omega N_i \nabla N_i \, p_i \, \mathrm{d}\Omega \\
&= \sum_{j \neq i} \int_\Omega N_i \nabla N_j \, p_j \, \mathrm{d}\Omega - \int_\Omega N_i \left(\sum_{j \neq i} \nabla N_j \right) p_i \, \mathrm{d}\Omega \\
&= \sum_{j \neq i} \int_\Omega N_i \nabla N_j (p_j - p_i) \mathrm{d}\Omega \\
&= \sum_{j \neq i} \boldsymbol{\nabla}_{ij} (p_j - p_i);
\end{aligned} \tag{18}$$

by applying Equation (17), we show how the pressure gradient term can be computed by using the $\boldsymbol{\nabla}_{ij}$ for any edge ij. Note that the term ii is never needed with the proposed approach.

2.1.1.2. Divergence term. The derivation of the divergence term is basically identical to the previous one, with the only difference that a scalar product is involved. Following exactly the same steps as before, it can be readily shown that (taking in account that $\mathbf{D}_{ij} \equiv \mathbf{G}_{ji}^T$)

$$\sum_j \mathbf{D}_{ij} \cdot \mathbf{u}_j = \sum_{j \neq i} \mathbf{D}_{ij} \cdot (\mathbf{u}_j - \mathbf{u}_i) = \sum_{j \neq i} \mathbf{G}_{ji}^T \cdot (\mathbf{u}_j - \mathbf{u}_i). \tag{19}$$

DOI: 10.1002/fld

2.1.1.3. Convection term. The non-linear convection term has to be approximated to fit within the framework of our edge-based formulation. Several possibilities exist to obtain a suitable form to be used in the calculations. One could start by considering the conservative form of the convection operator $\nabla \cdot (\mathbf{u} \otimes \mathbf{u})$.

$$\begin{aligned}\sum_j \int_\Omega N_i \nabla N_j \cdot (\mathbf{u}_j \otimes \mathbf{u}_j) \mathrm{d}\Omega = \\ = \sum_{j \neq i} \int_\Omega N_i \nabla N_j \cdot \mathbf{u}_j \mathbf{u}_j \mathrm{d}\Omega - \int_\Omega N_i \left(\sum_{j \neq i} \nabla N_j \right) \cdot \mathbf{u}_i \mathbf{u}_i \mathrm{d}\Omega\end{aligned} \tag{20}$$

which shows that the convective term can be estimated as

$$\sum_{j \neq i} (\boldsymbol{\nabla}_{ij} \cdot \mathbf{u}_j) \mathbf{u}_j - \sum_{j \neq i} (\boldsymbol{\nabla}_{ij} \cdot \mathbf{u}_i) \mathbf{u}_i . \tag{21}$$

Alternatively, one can start with the non-conservative form of the same equation and use a nodal integration rule as proposed in [30]. This approach estimates the convective term contribution as

$$\sum_{j \neq i} (\boldsymbol{\nabla}_{ij} \cdot \mathbf{u}_i)(\mathbf{u}_j - \mathbf{u}_i) \tag{22}$$

The first approach is 'globally conservative' by construction in the sense that the sum over all of the nodes in the mesh is guaranteed to give zero. This property is only approximately verified by the second technique, because the integration rule is not exact. In practice, both approaches work effectively, nevertheless, the second approximation appears to be slightly more robust and was the one chosen in the current work.

2.1.1.4. 'Weak' gradient term. The migration from a classical finite element to an edge-based implementation requires describing the gradient of a scalar function integrated by parts. Because in the current formulation we made the choice of not integrating by parts the pressure gradient, this is not strictly needed for the implementation of our method. In any case following [31], we obtain

$$\begin{aligned}\sum_j \int_\Omega \nabla N_i N_j p_j &= \sum_{j \neq i} \int_\Omega \nabla N_i N_j p_j \mathrm{d}\Omega + \int_\Omega \nabla N_i N_i p_i \mathrm{d}\Omega \\ &= \sum_{j \neq i} \int_\Omega \nabla N_i N_j p_j \mathrm{d}\Omega - \int_\Omega \left(\sum_{j \neq i} \nabla N_j \right) N_i p_i \mathrm{d}\Omega \\ &= \mathbf{G}_{ij} p_j - \boldsymbol{\nabla}_{ij} p_i .\end{aligned} \tag{23}$$

2.1.1.5. Viscous term. The viscous term in the Navier-Stokes equations requires estimating the scalar Laplacian operator L_{ij}. Although it is possible to store directly on each edge an entry of the type $L_{ij} := \int_\Omega \nabla N_i \cdot \nabla N_j \mathrm{d}\Omega$, we preferred in our implementation to store *a matrix term* of the type

$$\mathbf{L}^d_{ij} := \int_\Omega \nabla N_i \otimes \nabla N_j \mathrm{d}\Omega \tag{24}$$

on each edge of our mesh. The scalar gradient can then be obtained as needed by the trace operator as

$$L_{ij} = Tr \left(\mathbf{L}^d_{ij} \right) \tag{25}$$

which allows writing the viscous term as

$$\sum_{i \neq j} Tr \left(\mathbf{L}^d_{ij} \right) \nu (\mathbf{u}_j - \mathbf{u}_i) \tag{26}$$

DOI: 10.1002/fld

2.1.1.6. 'Special terms'. All the terms needed for the implementation of the Navier-Stokes equation have already been described. In our presentation of the discrete form, we avoided specifying the stabilization used, because this is relatively irrelevant for the description of our proposed algorithm. We would like nevertheless to point that storing the matrix Laplacian $\mathbf{L}_{ij}^d$ instead of its scalar counterpart, is justified only for the implementation of the stabilization operators and of the shock capturing terms. A detailed description of the usage in this context can be found in [31] or [32], nevertheless, the need to store such operator can be understood by considering a SUPG-like stabilization operator. On a given node i, the stabilization operator has a form of the type

$$\sum_{i \neq j} \nabla \cdot (\mathbf{u}_i \otimes \mathbf{u}_i) \nabla (\mathbf{u}_j - \mathbf{u}_i) \tag{27}$$

By using the matrix Laplacian operator, this can be approximated as

$$\sum_{i \neq j} \cdot \left(\mathbf{u}_i \cdot \mathbf{L}_{ij}^d \cdot \mathbf{u}_i \right) \nabla (\mathbf{u}_j - \mathbf{u}_i) \tag{28}$$

which requires considering $\mathbf{L}_{ij}^d$ as the matrix described. Similarly, the matrix form is also useful in the computation of the sub-scale residuals and for the definition of a cross-wind dissipation term which is useful in controlling unwanted numerical oscillations.

Remark 2
The common feature of all the terms described is that they can be evaluated for each node i independently of all the others. This implies that the calculation of the residuals can be performed in parallel for each node of the mesh.

Remark 3
All of the terms involved in the calculations can be computed without even considering the diagonal term, which is consequently not stored.

2.1.2. Improving volume correction. Independently of the time accuracy of the numerical scheme used for the first step, the overall scheme can not be more than second order accurate because of pressure-splitting procedure. Furthermore, the use of the continuum Laplacian operator, mandatory in the context of a semi-explicit scheme, implies some volume loss mostly concentrated in the vicinity of the free surface.

The origin of such loss can be traced back to two distinct phenomena:

a. As observed in [29], the pressure is fixed on the Neumann boundary as this is needed to make the Laplacian resolvable. This implies that it looses the capacity to adapt locally so to attempt guaranteeing the local volume conservation.
b. Equation (7) is generally evaluated at time step $n + 1$ implying that the divergence constrain depends exclusively on the velocity at $n + 1$. Any error in the fulfillment of the divergence constraint at the preceding step ($\mathbf{Du}^n = 0$) is simply discarded and never recovered.

The algorithm we devise for the solution of the free surface problem attempts to minimize the first issue. The idea, as we shall show in Section 3.4, is that the pressure will be fixed on the nodes outside the free surface, thus letting some freedom to the nodes in its proximity.

On the other hand, the fulfillment of the divergence free condition at the present time step (n + 1) and at the previous one (n) are combined in order to overcome the drawback stated in point b. The idea is the following: if no error was made in the past, we shall assume $\mathbf{Du}^n = 0$. However, this assumption is not verified in practice and mass is either created or destructed at a rate of $\mathbf{Du}^n$. Although usually this information is simply discarded, we modify the divergence free condition ($\mathbf{Du}^{n+1} = 0$) in order to sum up the volume variation lost (or gained) at the previous time ($\mathbf{Du}^n$).

In mathematical terms, our proposal is simply to modify the divergence constraint as

$$\mathbf{Du}^{n+1} + \mathbf{Du}^n = 0 \tag{29}$$

DOI: 10.1002/fld

As we will show in some of the examples, this simple modification improves the volume conservation of the overall scheme.

It is interesting to observe how 29 can be written equivalently as

$$\mathbf{Du}^{n+1} + \mathbf{Du}^{n} = 0 \rightarrow \frac{1}{2}\left(\mathbf{Du}^{n+1} + \mathbf{Du}^{n}\right) = 0 \rightarrow \mathbf{Du}^{n+\frac{1}{2}} = 0 \tag{30}$$

Which corresponds to a different time integration rule for the divergence constraint.

3. THE FREE SURFACE PROBLEM

Because the technique we propose is based on the use of a fixed-grid approach, the solution method has to be completed by the choice of a tracking method for the free surface and by the choice of an approach to apply the boundary conditions needed on the free surface. We will next present a brief description of the standard level set approach and continue with a detailed description of our approach to impose the boundary conditions on the free surface.

3.1. The level set concept

The *Level Set Method* was conceived as a methodology to follow moving interfaces. The interface is implicitly defined as the zero-valued iso-surface of a given smooth function [33, 34].

Let us call $\Omega^0 \subset \mathbb{R}^d$ (where d is the space dimension) the global control domain of analysis and $\Omega(t) \subset \Omega^0$ the fluid domain at time step t. The boundary of $\Omega(t)$ is defined by part of $\partial\Omega^0$ and by a moving boundary $\partial\Omega_m(t)$. The latter is defined as

$$\partial\Omega_m(t) := \{\mathbf{x} \mid \varphi(\mathbf{x}, t) = 0\} \tag{31}$$

From now on $\Omega(t) = \Omega$, and $\Omega_m(t) = \Omega_m$ and the explicit indication of time will be omitted for reason of simplicity. Following the same criteria, the fluid domain at a given time step t^n is $\Omega(t^n) = \Omega^n$.

As often done, the level set function is defined as a signed distance function, that is

$$d(\mathbf{x}) := min(|\mathbf{x} - \mathbf{x}_i|) \qquad \forall \mathbf{x}_i \in \partial\Omega_m \tag{32}$$

The level set function, for a given time instant t is defined as

$$\begin{aligned} &\varphi(\mathbf{x}) = d(\mathbf{x}) && \text{if} \quad x \notin \Omega, t \in (0, T); \\ &\varphi(\mathbf{x}) = d(\mathbf{x}) = 0 && \text{if} \quad x \in \partial\Omega_m, t \in (0, T); \\ &\varphi(\mathbf{x}) = -d(\mathbf{x}) && \text{if} \quad x \in \Omega, t \in (0, T); \end{aligned} \tag{33}$$

As detailed in [34], this function inherits all of the properties of implicit surfaces. Moreover, its monotonicity across the interface allows its differentiation.

The fundamental idea of using the level set approach can now be understood considering the mass conservation equation for a variable-density fluid. ρ being the density this states

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \mathbf{u} \cdot \nabla\rho + \rho\nabla \cdot \mathbf{u} = 0 \tag{34}$$

The case of interest is that of $\rho \neq 0$ inside the fluid domain and $\rho = 0$ outside of the free surface, where we may consider that a regularization function is applied to ρ to make it differentiable in space.

Let us split Equation (34) in the following two equations

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} + \mathbf{u} \cdot \nabla\rho = 0 \tag{35}$$

DOI: 10.1002/fld

and

$$\rho\nabla\cdot\mathbf{u}=0\rightarrow\nabla\cdot\mathbf{u}=0 \tag{36}$$

It is easy to understand that if such two equations are verified, then the former one will also be verified. Now, Equation (35) represents the transport of the density with the mean flow velocity. Because the density can behaved rather badly as it approximates a jump, it is convenient to replace it by the transport of a smooth scalar φ (in our case the distance function), which can be used to recover the density distribution at any moment. The problem is thus transferred to the solution of the transport problem

$$\begin{aligned}\partial_t\varphi+\mathbf{u}\cdot\nabla\varphi&=0 &&\text{in}\quad \Omega^0,\quad t\in(0,T),\\ \varphi&=\overline{\varphi} &&\text{on}\quad \partial\Omega_{in},\quad t\in(0,T),\\ \varphi(\mathbf{x},0)&=\varphi_0(\mathbf{x}) &&\text{in}\quad \Omega^0,\end{aligned} \tag{37}$$

where $\partial\Omega_{in}:=\{\mathbf{x}\in\partial\Omega^0\mid\mathbf{u}\cdot\mathbf{n}<0\}$ is the inflow part of $\partial\Omega_m$.

In principle, such equation can be solved using any numerical scheme at hand. Nevertheless, because the choice of using an explicit scheme for the momentum equation was made, we prefer using a RK4 scheme for the solution of the convective system 37 and an Orthogonal Sub-grid Scale stabilization technique.

3.2. Coupling the level set equation and the Navier-Stokes solver

In order to completely define our approach to the solution of the complete free-surface problem, we need to describe the coupling between the Navier-Stokes solver and the newly added level set equation. Conceptually, the velocity obtained from the solution of the Navier-Stokes equation has to be used in convecting φ, whereas the zero iso-surface of the same variables provides the position of the free surface and is consequently needed in prescribing the pressure condition on the Neumann boundary. Many different approaches were proposed over the years to perform such coupling, some based on sub-integration techniques on the cut elements [35] and others based on some form of regularization for the density function in the vicinity of the free surface. Our proposal rises from the observation that, once a continuous pressure distribution is assumed, the old pressure appears only with its gradient in the momentum equation (as we observed before, we prefer not to integrate by part the pressure term). This implies that the momentum equation can be solved approximately without knowing exactly the position of the free surface, provided that an estimate of the pressure gradient is given in any active (or potentially active) area of the fluid domain. On the other hand, the imposition of the zero traction condition on the Neumann Boundary could be applied instead in the pressure correction step through the imposition of idoneous boundary conditions at the level of the pressure Laplacian system.

To complete the algorithm some other ingredients are required as follows:

- An *extrapolation function*: it defines the values of velocity, pressure and gradient of pressure on the nodes in the non-fluid area close to the free surface.
- A tool for the calculation of the *nodal distance* in the whole control domain Ω_0 once the new free surface has been defined;
- A way to choose the pressure to be applied on the outer nodes so that the pressure is approximately zero on the free surface.

3.3. The extrapolation procedure

The extrapolation of the velocities and the pressures from the interior of the domain towards the exterior, represents one of the most critical steps of the calculation process. Our proposal is to define an auxiliary data structure that contains the different layers in the vicinity of the free surface as shown in Figure 1.

DOI: 10.1002/fld

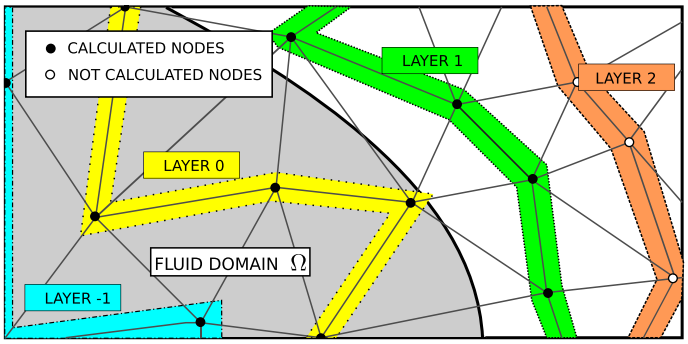


Figure 1. Extrapolation layers and calculated nodes in the time interval $t^n - t^{n+1}$.

An extrapolation domain is set and different layers are defined using the following criteria:

- *LAYER* 0 (L^0) is the first layer of nodes of the fluid domain internal the free surface ($L^0 \in \Omega^n$).
- *LAYER* k (L^k) ($k = 1, 2, \ldots, nl$ [‡]) is the layer of non-fluid nodes neighboring with L^{k-1} ($L^k \notin \Omega^n$)

The fluid velocity and pressure fields on the layers L^k with $k \leqslant 1$ are known from the previous time step t^n (the black nodes in Figure 1). In our proposal, however, such values are not used in performing the extrapolation of pressure, velocity and gradient of pressure, but rather velocity is taken starting from layer L^0 and pressure and pressure gradient from the L^{-1}. The rationale of this choice is that the pressure and pressure gradients in the immediate vicinity of the free surface may show a certain level of spurious oscillations, because the pressure is imposed strongly on layer L^1, and the effect of a non-smooth pressure boundary condition may be still felt on layer L^0. We thus propose to start the extrapolation of the pressure (and pressure gradient) from the inner layer (L^{-1}), which guarantees a much smoother behavior of the extrapolation area.

In symbols, we define the pressure gradient on each node i of a given layer k, as the arithmetic average (avg) of all of its neighbors j which belongs to the layer of lower order

$$\nabla p_i^k := avg\left(\nabla p_j^{k-1}\right) \ \forall k = 0 \ldots nl \ i \in L^k \ j \in L^{k-1} \tag{38}$$

Given such pressure gradients, the pressure is then evaluated on node i so to maintain the extrapolated pressure gradient, that is

$$p_i^k := avg\left(p_j^{k-1} + \mathbf{h}_{ij} \cdot \nabla p_j^{k-1}\right) \ \forall k = 0 \ldots nl \ i \in L^k \ j \in L^{k-1} \tag{39}$$

where $\mathbf{h}_{ij} := \mathbf{x}_j - \mathbf{x}_i$ is the vector from i to j.

The extrapolation of the velocity is performed in a very similar way, with the only difference that the extrapolation starts from layer L^0 not from L^{-1} like in Equations (38) and (39) (see Figure 1 for a graphical representation).

$$u_i^k := avg\left(u_j^{k-1}\right) \ \forall k = 1 \ldots nl \ i \in L^k \ j \in L^{k-1} \tag{40}$$

The extrapolation procedure described provides a prediction of the velocity and pressure field that is likely to be found outside of the pressure domain. Such extrapolation is performed before convecting the distance function and should be extended sufficiently far to cover all of the area upon which the fluid domain is likely to extend during the following time step. We shall remark that the data structure that contains the different layers should be updated every time the distance function is convected.

[‡] nl states for number of extrapolation layers set up by the user.

DOI: 10.1002/fld

We shall also observe that the choice of using the 'strong' form of the pressure gradients in the momentum equation appears at this point to be beneficial. The idea is that, because we made the choice of not integrating by parts the pressure gradient, no boundary integral of the pressure is needed on the free surface (in the solution of the momentum equation), and the only thing needed on any fluid element (including the elements cut by the free surface) is the correct computation of the pressure gradient, which is automatically available once the pressure is extrapolated as described.

Once the convection operation has been performed by solving Equations (37), the level set function is no longer the Euclidean distance function presented in Equation (33). To recover its original nature, a tool to re-evaluate the nodal distance from the new calculated free surface has been developed.

Because of the dynamic nature of the analyzed problem, a redefinition of the fluid domain $\Omega := \{\mathbf{x} \in \Omega_0 | \varphi(\mathbf{x}) \leqslant 0\}$ is necessary at each time step.

For the calculation of the distance field of the domain Ω_0, numerical methods have to be employed because the use of analytical solution is not trivial. The method proposed by Elias, Martins and Coutinho in [36] is taken as reference. It takes its origin from the Fast Marching Method, a technique, first developed by Sethian (see [37]), for the computation of the arrival time of a front. Observe that this approach requires the values of φ to be known on L^0 and L^1 in order to be initialized. This is performed geometrically by computing the Euclidean distance from the zero of the level set function.

3.4. Prescription of the boundary condition on the free surface

Despite its advantages, the pressure extrapolation described in the previous section does not impose in any way the traction-free condition at the free surface. This is done in the second step of the FS procedure (Equation (14)) by fixing the value of the pressure at the time step $n + 1$ so that the pressure field is zero at the free surface. Because the free surface cuts the element at arbitrary position, no nodes are available to directly fix the pressure.

Let us remember that there is no need of computing any boundary integral because we do not integrate by part the pressure gradient term, like is explained in Section 2.1.

Our approach is to consider that the extrapolated pressure gradient is correct in the vicinity of the free surface . We will thus fix the pressure values of the nodes in L^1 so to guarantee that the pressure at the free surface is zero, provided that the pressure gradient is kept fixed. Note that in doing so, we need to recompute the layer L^1, because it does not necessarily coincide with the one used in the extrapolation step.

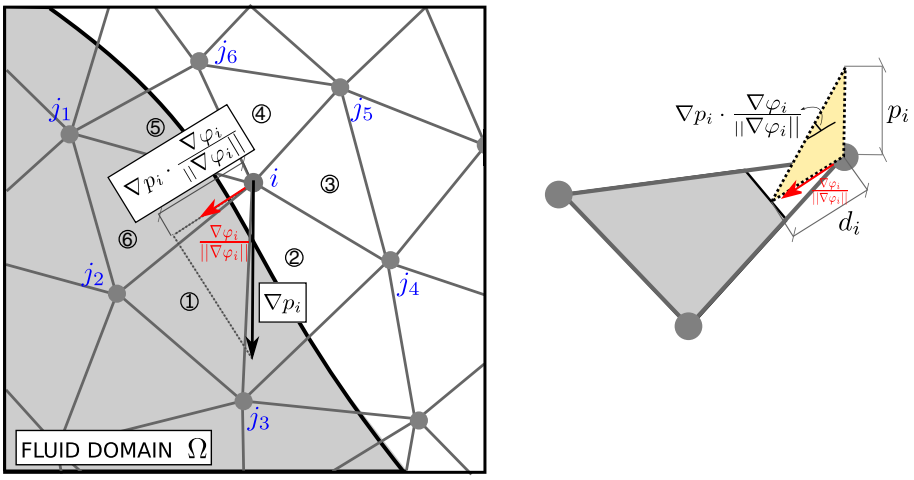


Figure 2. Graphical explanation of the evaluation of pressure on the first layer of non-fluid nodes in order to respect the incompressibility condition.

DOI: 10.1002/fld

In mathematical terms, the component of the pressure gradient of node i ($\in L^1$) in the direction of the distance (which is the gradient of the level set function) is evaluated. Therefore, the pressure of node i is calculated setting zero pressure on the free surface, whose distance from node i is known, and interpolating linearly. Defining as i-patch the cluster of elements whose node i belongs to (elements $1-6$ of Figure 2), the pressure on node i is evaluated like the value of the level set function on node i times its gradient of pressure in the direction of the gradient of the level set function itself.

$$p_i^1 = \nabla p_i^n \cdot \frac{\nabla \varphi_i^{n+1}}{\left\| \nabla \varphi_i^{n+1} \right\|} \varphi_i^{n+1}; \tag{41}$$

where $\nabla \varphi_i^{n+1}$ and $\left\| \nabla \varphi_i^{n+1} \right\|$ are the gradient of the level set function on node i and its L_2 norm, respectively, and φ_i^n is the level set function itself. $\nabla \varphi_i^{n+1}$ is calculated considering the contribution of the gradient of the level set function on each edge concurring on node i. For instance edges ij_p (with $p = 1, 2, .., 6$) of Figure 2.

Remark 4
It is important to observe that Equation (41) is the only point at which the level set function is actually required to be a distance. Because its value is only needed on L^1, it is convenient to recompute it as accurately as possible at every time step. This can be done geometrically for the elements crossed by the zero of the level set function with a minor computational cost.

Remark 5
The correct calculation of the residual of the momentum equation would have required integrating only on the fluid area of the cut elements. This is impossible within an edge-based formulation unless one wants to recompute the edges and loose efficiency. We thus accept to integrate on the whole element area considering that both the body force and the pressure gradient are extrapolated on the outside of the fluid. This approximation is acceptable under the assumption that the volume of the cut elements is small compared with the overall fluid volume and exact for the hydrostatic case where the gradient of pressure and the body force exactly cancel each other (see. Section 5.1 for an empirical verification). Furthermore, the related error goes to zero as the element size is reduced.

3.5. The algorithm

Collecting all the ideas, we arrive to the algorithm:

1. Given the level set function φ^n, *extrapolate* velocity, pressure and pressure gradient so to obtain $\mathbf{u}_{ext}^n$, p_{ext}^n and ∇p_{ext}^n defined as the velocity, pressure and pressure gradient over the extrapolated domain.
2. *Convect* the level set function φ defining the new free surface at t^{n+1} using $\mathbf{u}^n$ and $\mathbf{u}_{ext}^n$. Note that the extrapolated values are only required within a limited number of layers, which are the ones on which the convection will be actually performed.
3. *Re-initialize* (if needed) the *distance* in the whole domain starting from the zero of the level set function at t^{n+1} obtained at step 2.
4. *Solve the momentum* Equation (13). Note that the solution is performed on the domain at the predicted free surface position (φ^{n+1}).
5. Set the approximate *pressure boundary conditions* on $\partial\Omega^{n+1}$ so to guarantee that the pressure is (approximately) zero at the position indicated by the zero of the level set function. In order to do that, the geometric distance is evaluated on L^1.
6. *Solve for the pressure* Equation (14).
7. *Solve for the correction* Equation (15).
8. Move to next time step.

DOI: 10.1002/fld

4. PARALLELIZATION

The edge-based approach used in this work is well suited for parallel processing. One can take advantage of the fact that in each node i, all terms to be calculated are completely independent from other nodes to parallelize the algorithm in its nodal loop. The parallelization of the level set algorithm however requires some modifications in the recalculation of the distance function.

At the moment of writing this paper, the solver is parallelized for shared memory machines using OpenMP. However, the parallel algorithm is also adequate for distributed memory machines parallelization.

4.1. Parallelization of the fluid solver

The first step is the solution of the momentum equation (Equation (13)), which requires four evaluations of the residual within each time step.

In the current work, we made the decision of not using the symmetry of the edge data structure, calculating both edge ij and edge ji as two different edges. This allows computing the residual by a loop of the type

```
r[i] = 0
for i in node_indices (-->perfectly parallel loop!)
 gather data of node i
 for j in "neighbors of i"
  edge = gather edge i,j
  gather data of node j
  r[i] += edge contribution
```

Although this choice approximately doubles the computational work to be performed for each node i, it provides the important advantage that the residual on each node can be computed in parallel, thus allowing a trivial parallelization. Furthermore, the nodal data relative to node i can be cached thus allowing its effective reutilization. We shall remark that other authors (see, e.g., [3]) prefer to take advantage of the symmetry and perform the parallelization of the above loop by using coloring strategies. Such decision is perfectly justified and was proved to provide a very high level of efficiency. In our work, we preferred the first option so to allow a simpler parallelization and future developments as described for example in [32].

The solution of the pressure step (Equation (14)) is performed using an iterative solver and is also trivially parallel as it relies on standard SpMV (Sparse Matrix-Vector product). On the other hand, step 3 of the FS procedure (Equation (15)) has an equivalent form and was parallelized in the same way.

4.2. Parallelization of the reinitialization step

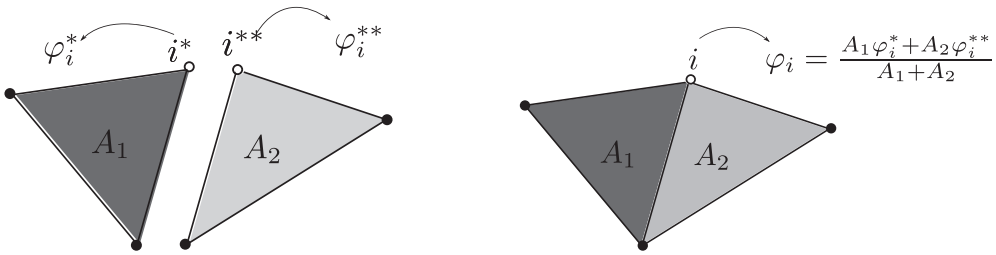
As explained in Section 3, the level set algorithm consists of two parts: convection of the level set function and recalculation of the distance function. Whereas the first part is parallelized using the same methodology described for the momentum equation (see, e.g., [32] for details of the formulation), the second requires more considerations to avoid possible conflicts between threads.

The recalculation of the distance function can be decomposed in two parts:

- The calculation of φ on the nodes of the elements crossed by the free surface, performed geometrically.
- The evaluation of φ on the rest of the domain. This is done using the method presented in [36] and previously commented. We proceed identifying a group of 'active' elements with only a node with unknown distance that we compute for each of such elements.

In both cases, the value to be assigned to each of the unknown nodes is evaluated independently (and in parallel) in every 'active' element and its final nodal value is obtained by performing an area-weighted average of the elemental distances just calculated. This is best explained by considering Figure 3. The level set function of node i, φ_i is calculated independently in the two elements

DOI: 10.1002/fld



(a) Parallel calculation of elemental contribution of φ_i.

(b) Assembling of elemental contribution of φ_i using an area-weighted average.

Figure 3. Graphical explication of the parallel calculation of the level set function φ. The distance value is known on the black nodes and has to be calculated on the white one. In this case, both elements are 'actives' having only one node with unknown distance.

node i belongs to (Figure 3(a)). The elemental contributions are afterwards 'assembled' using an area-weighted average (Figure 3(b)).

The operation is parallelized using a semaphore-based approach and implemented using OpenMP locks.

5. NUMERICAL EXAMPLES

In the next sections, a series of examples are presented to show the good performance of the proposed method.

The good capability of reproducing an hydrostatic pressure distribution in case of a still water simulation, without oscillations or loss of volume, is shown in the first example (Section 5.1).

The volume conservation properties are studied in Section 5.2 where two different models are considered:

- A prismatic vertical channel with an inlet in the bottom face;
- A prismatic vertical channel with a lateral asymmetric inlet;

The improvements in volume conservation explained in sec. 2.1.2 are shown to be crucial in the case of critical conditions (slow velocity and waves).

A comparison between the performance of the present algorithm and a Lagrangian approach (using PFEM [9]) is then presented through two challenging models of a $2d$ flip bucket and a $3d$ dambreak.

Finally, an industrial application of a mold-filling process is presented. In this case, the parallel architecture of the code is shown to be essential.

5.1. Still water example

The first example is a very simple verification of the correct pressure calculation in the case of a tank half filled with water.

Although monolithic methods are typically capable of providing a seamless solution for this kind of problem, the use of pressure-splitting technique can be problematic, leading to the appearance of spurious velocities. The example is stationary, but in order to ensure that no spurious velocities appear, the simulation time is 10 s and the results are taken from the last step of analysis.

Incompressible water has the following characteristics: kinematic viscosity $\nu = 10^{-6}$ m^2/s, density $\rho = 1000$ kg/m^3, gravity is 9.81 m/s in the vertical direction.

The geometry of the model is detailed in Figure 4 and in the same figure the calculated free surface is plotted. The studied meshes are presented in Figures 5 and 6, and they are detailed in Table I. They are intentionally chosen in order to have the free surface not coinciding with the edges of the elements.

DOI: 10.1002/fld

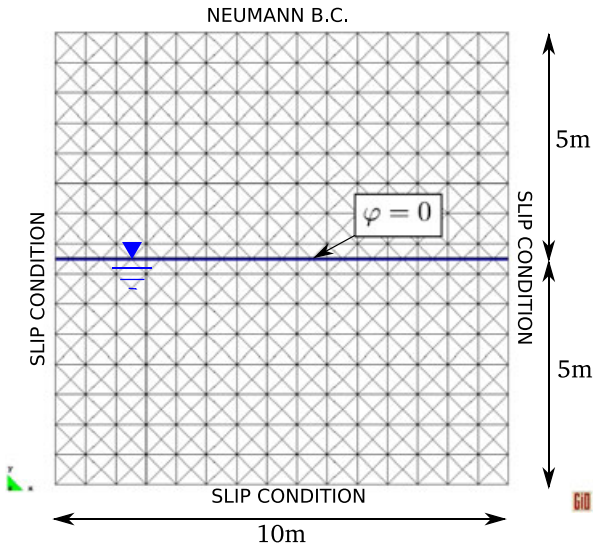
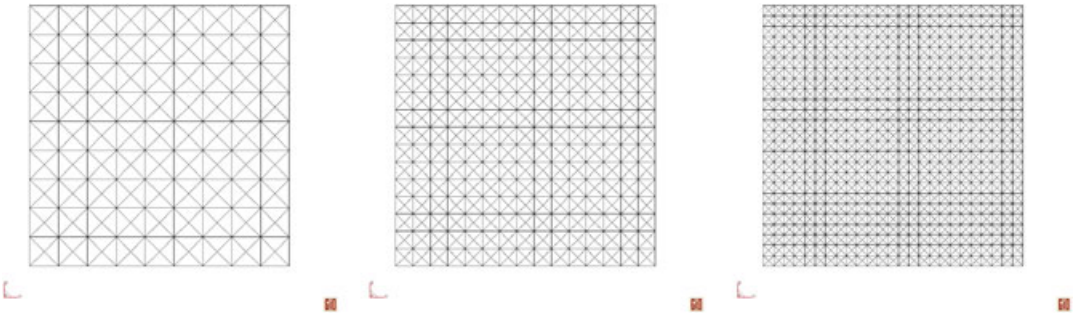
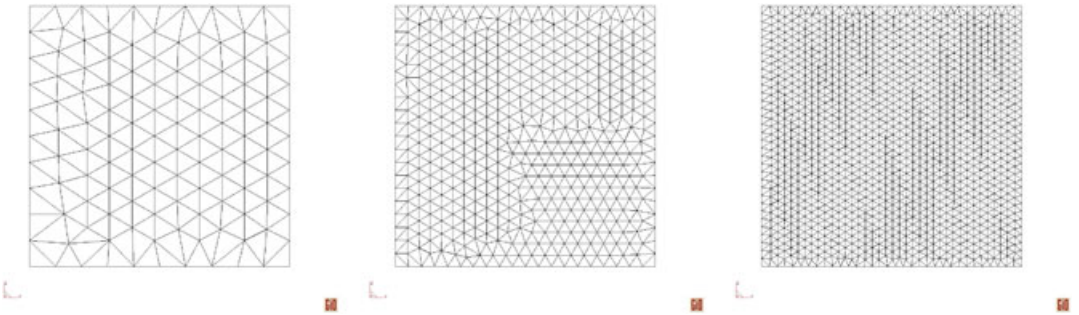


Figure 4. Geometry and boundary condition in the still water example. The blue lines indicate zero value of the level set function φ.



(a) Coarse structured mesh. (b) Medium structured mesh. (c) Fine structured mesh.

Figure 5. Structured meshes considered in the calculation.



(a) Coarse unstructured mesh. (b) Medium unstructured mesh. (c) Fine unstructured mesh.

Figure 6. Unstructured meshes considered in the calculation.

The hydrostatic pressure distribution is not provided at the beginning of the calculations, and as a consequence, the fluid starts from a non-equilibrium configuration and moves slightly before encountering its equilibrium configuration. This results in the appearance of some spurious velocities in the examples. Such velocities are relevant only in the case of a coarse mesh and decrease

DOI: 10.1002/fld

Table I. Still water example.

	STRUCTURED			UNSTRUCTURED		
	Fine	Medium	Coarse	Fine	Medium	Coarse
Number of nodes	1301	481	181	1316	482	125
Number of elements	2500	900	324	2498	882	218
Elemental length [m]	—	—	—	0.3	0.5	1.0
Elements per side	25	13	9	—	—	—

Number of nodes, number of elements, elemental length (unstructured meshes) and number of elements per edge (structured meshes) of the meshes considered in the analysis.

Table II. Order of magnitude of the maximum spurious velocity appearing for the different meshes.

	STRUCTURED			UNSTRUCTURED		
	Fine	Medium	Coarse	Fine	Medium	Coarse
$o(u_{max})[m/s]$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-2}$	$1 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$1 \cdot 10^{-3}$

Table III. Vertical column example.

	STRUCTURED	UNSTRUCTURED
Number of nodes	2981	1210
Number of elements	13800	6117
Elemental length [m]	—	1
Elements per side	$5 \times 5 \times 20$	—

Number of nodes, number of elements, elemental length (unstructured meshes) and number of elements per edge (structured mesh) of the meshes considered in the analysis.

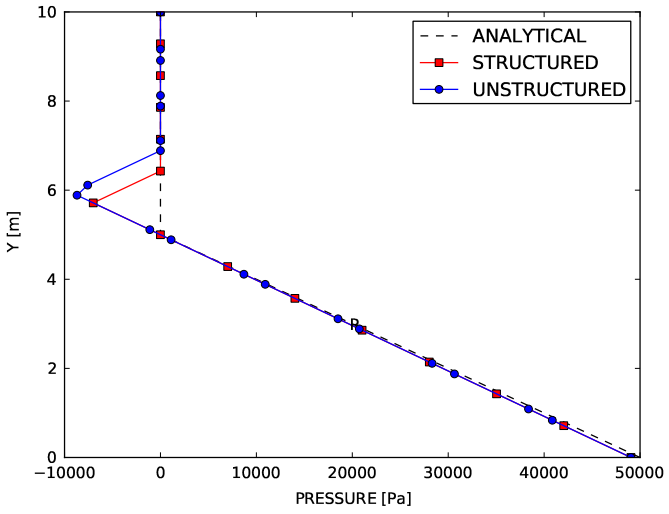


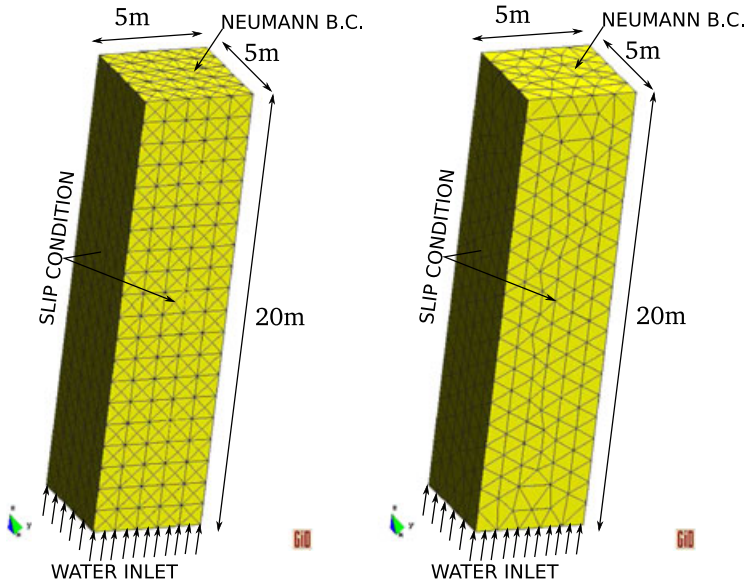
Figure 7. Pressure distribution along the vertical central section in the case of the structured and unstructured coarse mesh shown in Figures 5(c) and 6(c).

very quickly with the refinement of the same. The order of magnitude of their maximum absolute value is shown in Table II. As expected no spurious velocities are present if the hydrostatic pressure distribution is provided.

DOI: 10.1002/fld

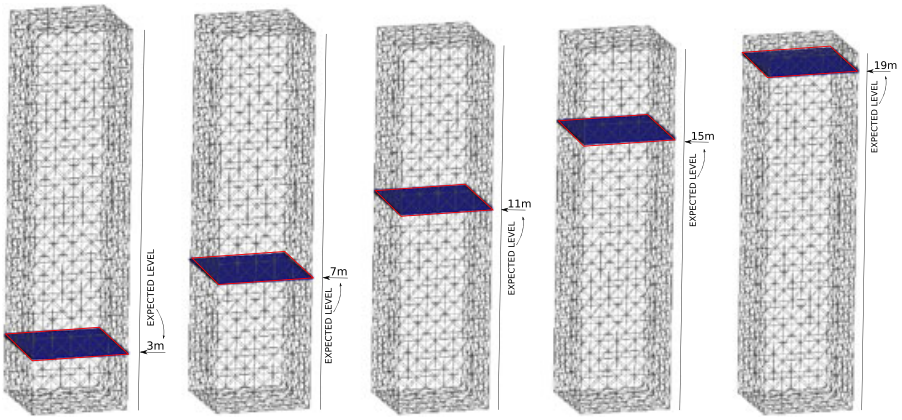
The expected pressure on the free surface has to be zero, and the gradient of pressure has to be constant in the crossed elements and equal to the interior one according to what explained in Section 3.4 and 3.3. This implies that the first layer of non-fluid nodes are expected to have negative pressures. The correct behavior of the algorithm can be appreciated in Figure 7 showing the pressure distribution along a central vertical section in the case of the structured and unstructured coarse meshes.

The integration of the momentum equation on the whole area in this specific case is correct. Pressure gradient in fact exactly balances the body forces both in the fluid and in the non-fluid part of the cut elements and the free surface is correctly calculated.



(a) Structured mesh. Free surface. (b) Unstructured mesh. Free surface.

Figure 8. Geometry and mesh taken into account.



(a) t = 2s (b) t = 6s (c) t = 10s (d) t = 14s (e) t = 18s

Figure 9. Structured mesh. Evolution of the free surface with a 1 m/s bottom incoming velocity. On the right of each snapshot the expected level is indicated.

DOI: 10.1002/fld

5.2. *Vertical column*

The second example is a vertical rectangular column with an inlet in the bottom side and an outlet on the top face. Geometry and conditions of the present examples are taken from [38] (although in that case, the interaction between two fluids with different specific weight was taken into account). Nevertheless, the problem presents the same difficulties of maintenance of a flat free surface both in the transitory and in the stationary regime.

The two mesh studied can be seen in Figure 8: a structured one and an unstructured one. Their details are summarized in Table III.

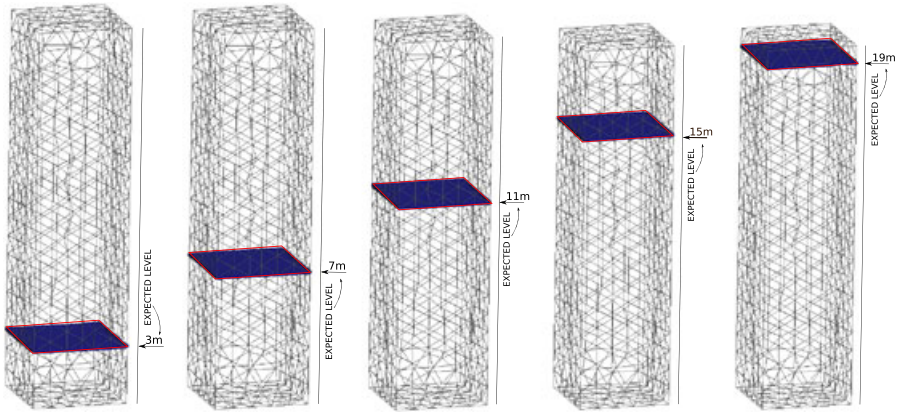


Figure 10. Unstructured mesh. Evolution of the free surface with a 1 m/s bottom incoming velocity. On the right of each snapshot, the expected level is indicated.

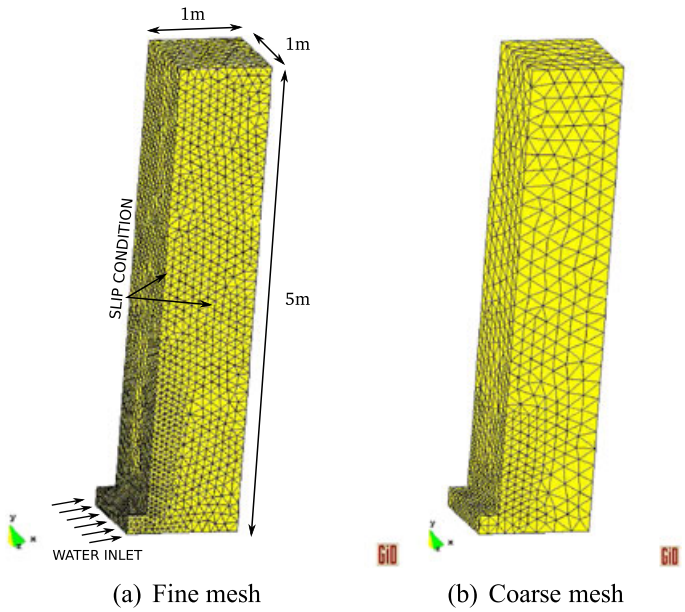


Figure 11. Mesh and geometry of the vertical channel with lateral entrance of water.

DOI: 10.1002/fld

Figures 9 and 10 show the evolution of the free surface (identified with the zero of the level set function ($\varphi = 0$) during the filling process. Considering that the free surface at time $t = 0$ is located at $h = 1$ m from the bottom, and the velocity inlet is $v = 1$ m/s a very good accordance with expected level of the free surface can be noticed at each time step (Figures 9 and 10). In both cases, the expected level of water at 2 s, 6 s, 10 s, 14 s and 18 s is respected, and it is 3 m, 7 m, 11 m, 15 m and 19 m, respectively. No oscillations are observed neither in the unstructured nor in the structured mesh.

If we slightly change the geometry considering a lateral entrance of water, and we decrease the value of inlet velocity to $v_{in} = 0.1$ m/s (see Figure 11 for the details on the geometry and the boundary conditions considered), the improvement of volume conservation explained in Section 2.1.2 plays a relevant role. Two meshes are considered in the calculation: a coarse and a fine one whose characteristics are summarized in Table IV and shown in Figure 13.

In Figure 12(a), the beneficial effect of the volume correction can be appreciated. The expected level of water is compared with the one calculated in the case of running the fine mesh model with and without volume conservation improvements. On the other hand, it is important to observe that, with the volume correction, no relevant changes are observed when a coarser mesh is employed (observe Figure 12(b)).

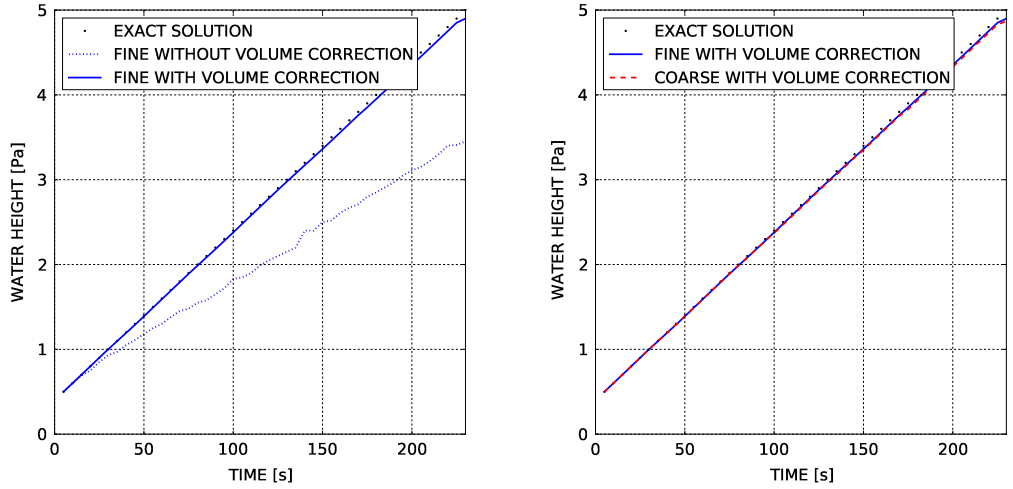
Finally, another important aspect is that the use of the volume correction leads to a flat free surface reducing the oscillations. This can be observed by comparing Figure 13(a,b) where the volume correction is used in both the fine and coarse mesh with Figure 13(c) where it is not used.

Figure 14 shows the speedup of the algorithm as measured on different platforms. The first platform has one Intel *i*7 Quad-core CPU, and the second one has 2 Quad-core AMD opteron CPUs. The mesh uses in the benchmark has 115 919 elements and is hence finer than the one shown in

Table IV. Vertical column with lateral entrance example.

	Fine	Coarse
Number of nodes	12 100	3 050
Number of elements	61 600	14 400

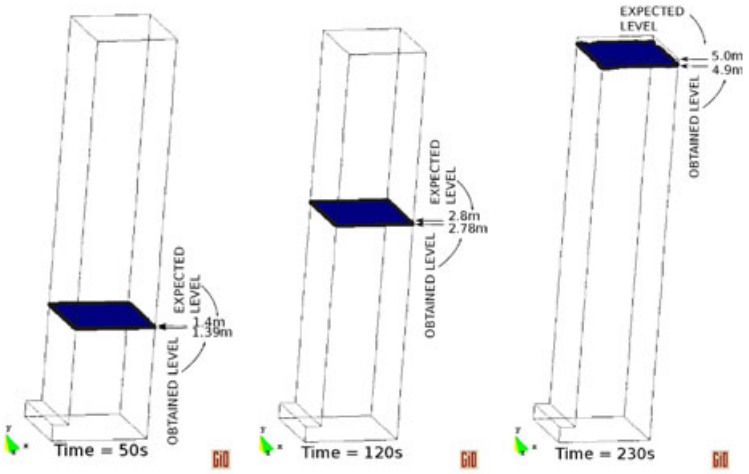
Number of nodes and number of elements of the meshes considered in the analysis.



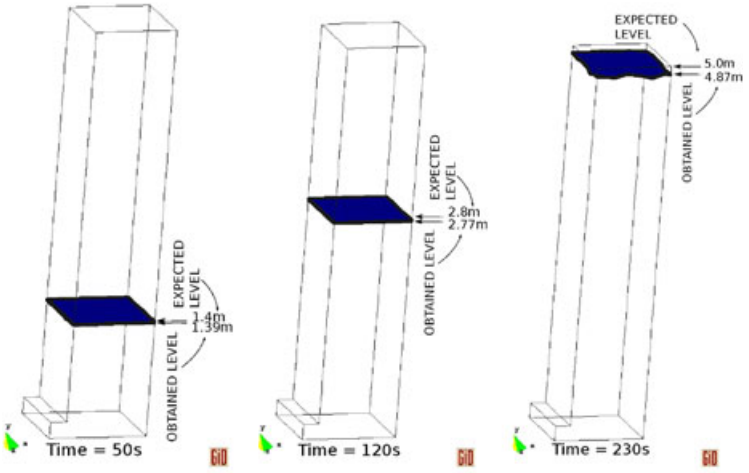
(a) With or without the volume correction. Fine mesh. (b) With volume correction. Coarse and fine mesh.

Figure 12. Vertical column with lateral entrance example. Level of water in function of time.

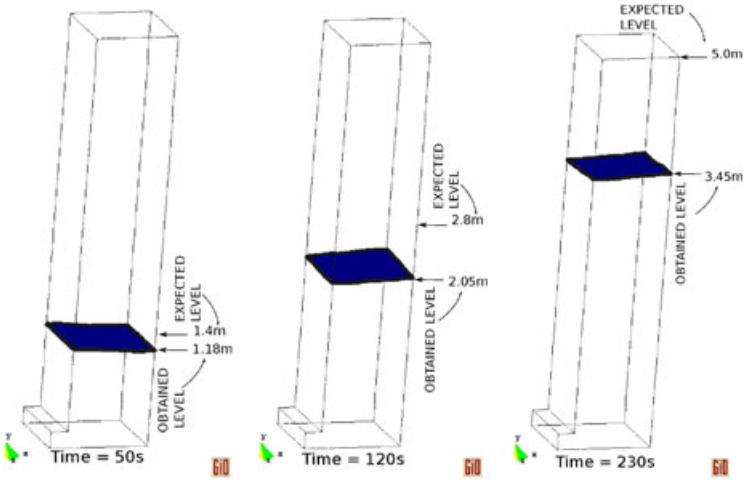
DOI: 10.1002/fld



(a) With volume correction. Fine mesh model



(b) With volume correction. Coarse mesh model



(c) Without volume correction. Fine mesh model

Figure 13. Vertical column with lateral entrance example. Evolution of the free surface at 50 s, 120 s and 230 s.

DOI: 10.1002/fld

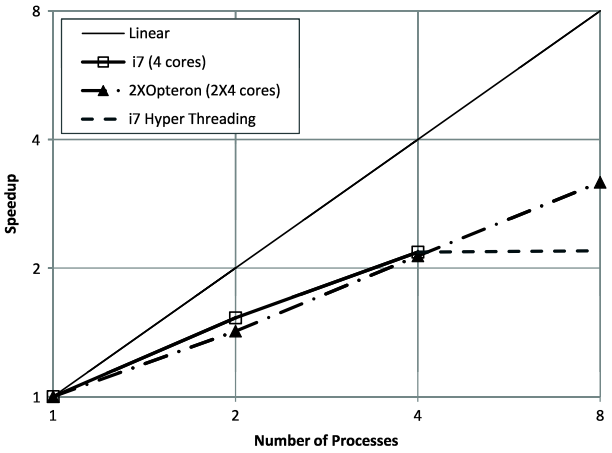


Figure 14. The speedup of the solver over different multi-cores machines.

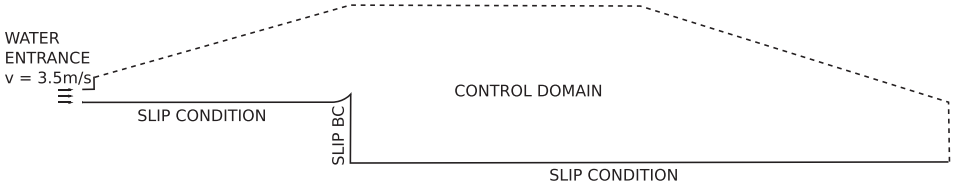


Figure 15. Geometry and boundary condition of the flip bucket example.

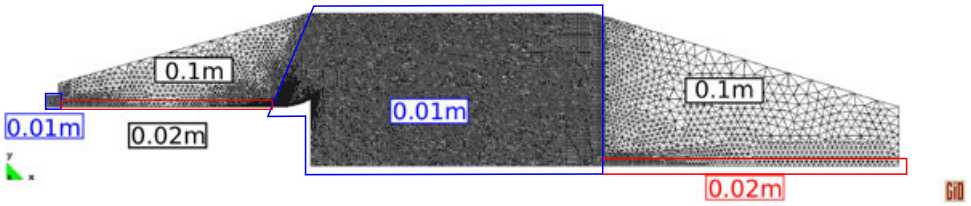


Figure 16. Mesh of the flip bucket example.

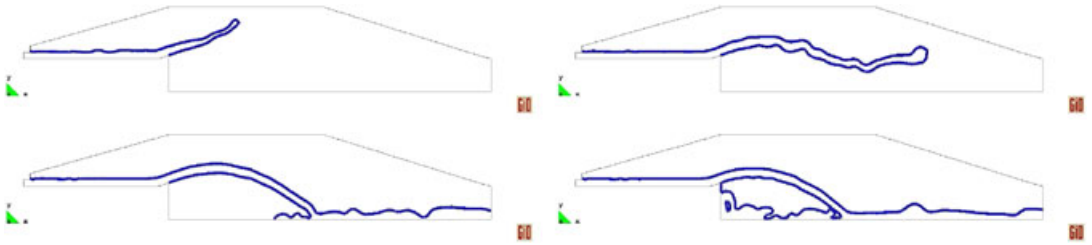


Figure 17. Sequence of the transitory phase of the jet.

the figure. In both cases, the speedup is less than ideal. The main reason for this is that the computations are bandwidth limited and as verified in [32], the maximum bandwidth can be achieved without using all the processors.

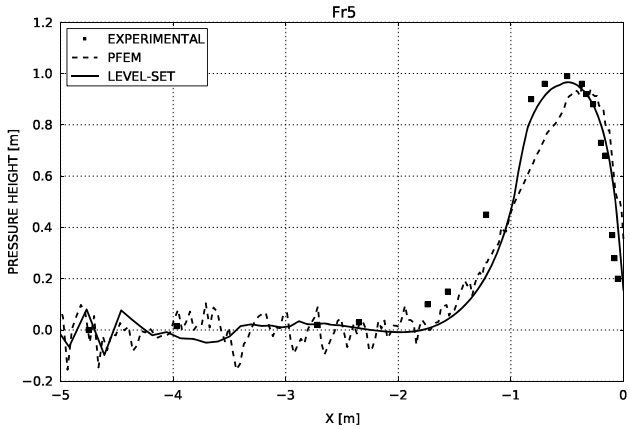
DOI: 10.1002/fld

The specific case of the vertical column was run using relatively few elements ($< 120\,000$). This implies that the OpenMP overhead plays an important role. We shall remark however that a similar speedup is measured for larger examples (see, e.g., the industrial test in Section 5.5) suggesting that the bandwidth limitation presents a stricter limit than the overheads of the parallelization.

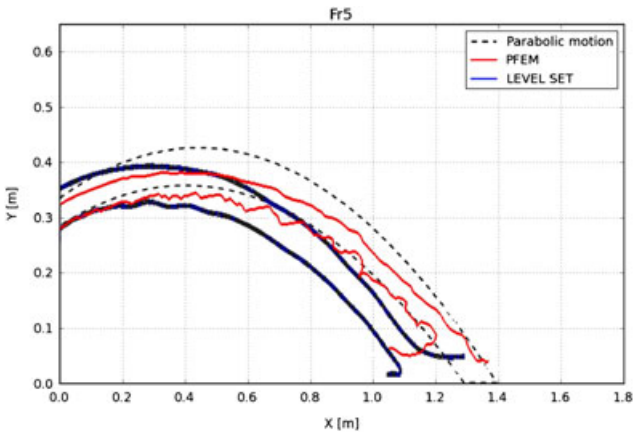
5.3. *Flip bucket*

The present example reproduces an experiment carried out by Prof. W. Hager and coworkers whose results can be found in [39]. The performance of the present level set technique is compared with the results obtained using PFEM [2, 6, 8] and published in [9]. The geometry data, initial and boundary conditions can be found in [9]. The case with Froude number 5 is considered. The control volume and the mesh used can be seen in Figures 15 and 16, respectively.

An entrance of water is imposed in the left side. After a transitory phase shown in Figure 17, the stationary regime is achieved, and pressure is registered on the bucket as shown in Figure 18(a). The jet shape is also compared in Figure 18(b) where the darker line is the level set, whereas the lighter line represents the PFEM results.



(a) Pressure distribution on the bucket. Experimental and numerical comparison



(b) Jet trajectory. Relative comparison.

Figure 18. Level set and PFEM comparison in the pressure head calculation and the jet development.

DOI: 10.1002/fld

A good accordance with experimental pressure along the bucket can be seen in Figure 18(a). The black points are the experimental results found in [39], whereas the continuous line and the dotted line are the level set and the PFEM solutions, respectively.

5.4. 3D dambreak

The present example is a $3D$ dam break already studied by the authors in [9] using PFEM.

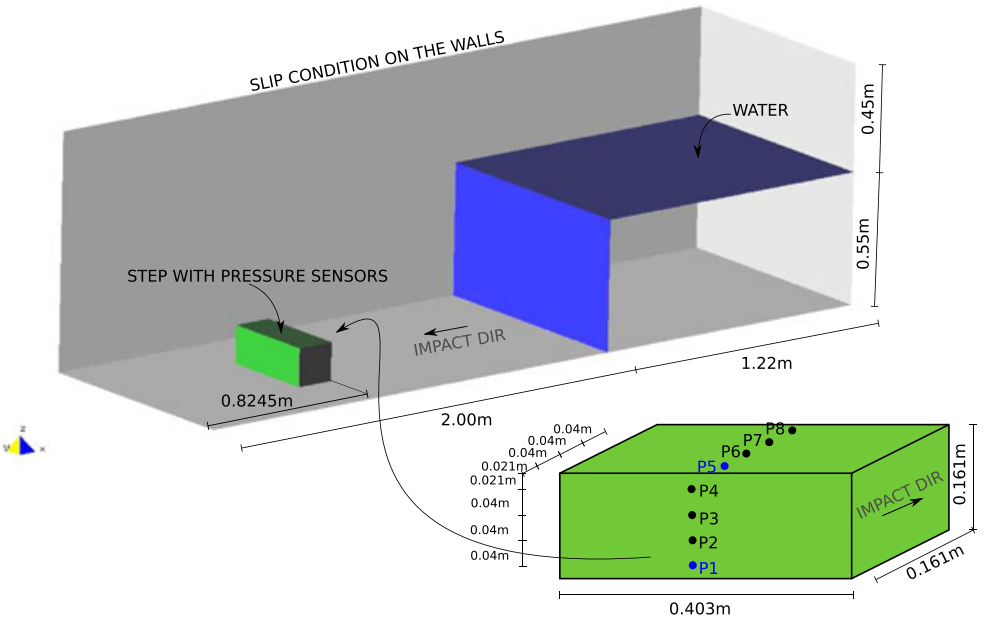
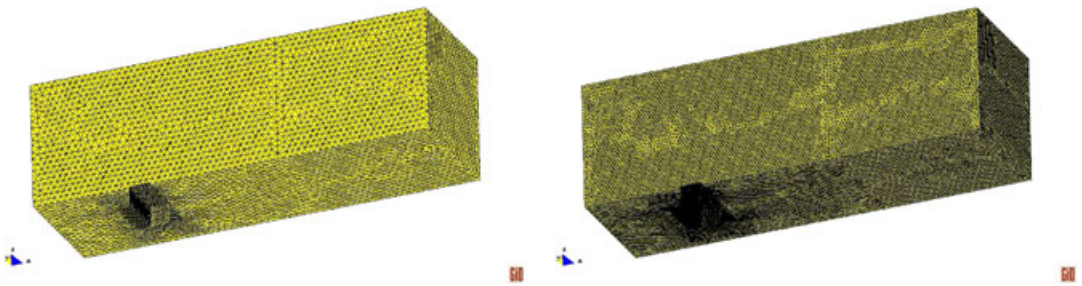


Figure 19. Geometry and boundary condition of the $3D$ dam break example. On the lower left corner a zoom on the pressure sensors distribution on the step

Table V. Dam break example.

	Mesh A	Mesh B
Number of nodes	51 627	392 130
Number of elements	296 157	2 310 984

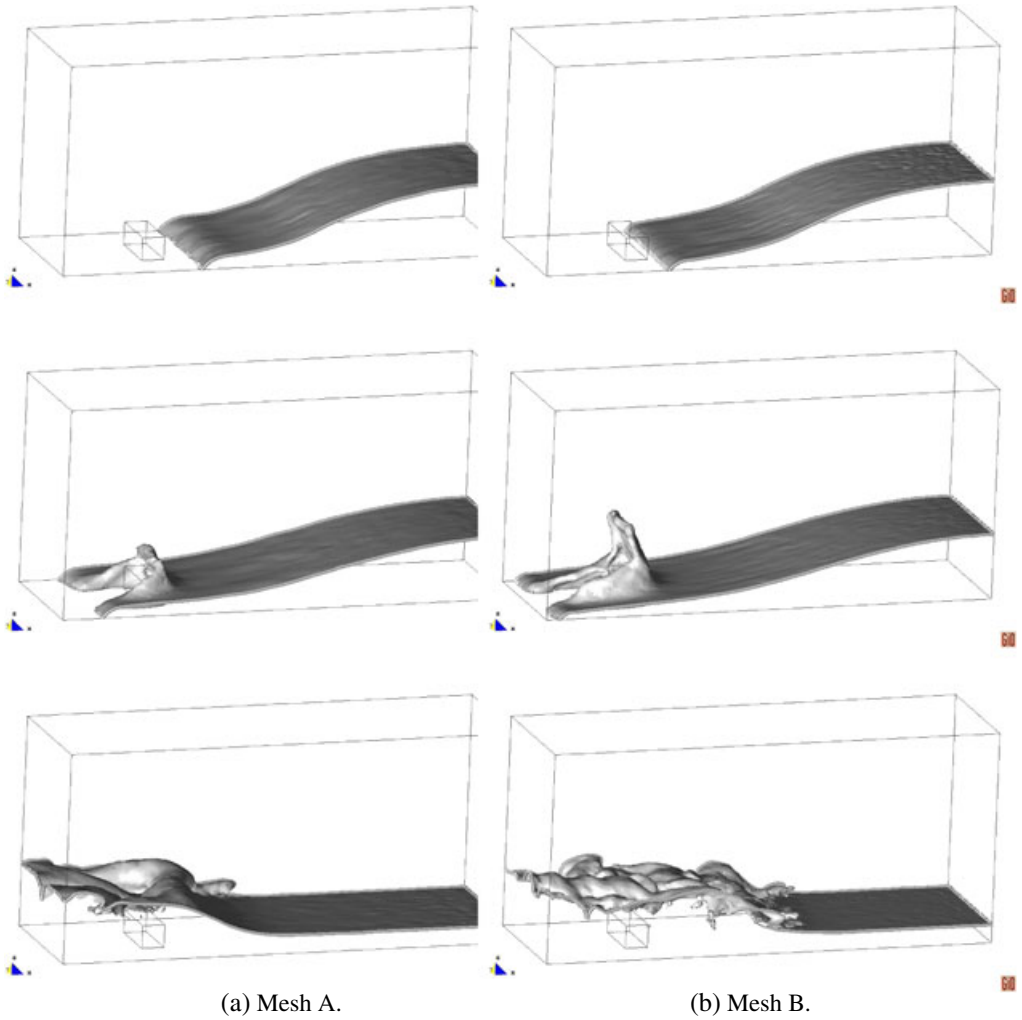
Number of nodes and number of elements of the two meshes considered in the analysis.



(a) Mesh A. (b) Mesh B.

Figure 20. The two meshes considered. On the left Mesh A of 296 157 and Mesh B of 2 310 984 tetrahedra.

DOI: 10.1002/fld



(a) Mesh A. (b) Mesh B.

Figure 21. Evolution of the dam break at 0.4 s, 0.6 s and 2.0 s. Comparison between the results obtained with mesh A and B.

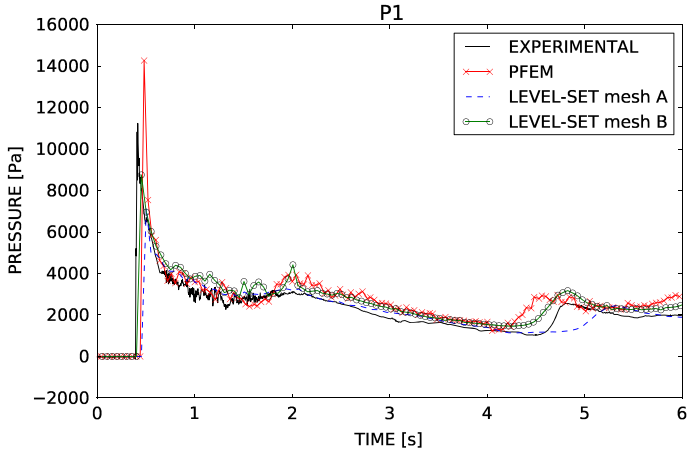


Figure 22. Pressure evolution on pressure sensor P1 (see Figure 19). Level set, PFEM and experimental comparison.

DOI: 10.1002/fld

Data are taken from the experiments performed at the Maritime Research Institute Netherlands for breaking dam flows [40]. Several numerical results of this case study are available in literature for volume of fluid techniques. This is the case of [40] employing Cartesian grids, or [15] using an edge-based approach. Finally, other level set simulations can also be found. Among others, in [16, 41], an application of isogeometric analysis is presented.

The water column is left free to fall over a step where pressure sensors are set following the scheme of Figure 19. The details of geometry can be found in [15].

Two meshes are considered in the present work, their characteristics are detailed in Table V, and they are shown in Figure 20.

A sequence of the falling of the water column can be seen in Figure 21 where the free surface evolution is plotted for the two meshes considered.

The pressure evolution in time obtained with the two meshes is compared in Figures 22 and 23 with experimental results and PFEM results taken form [9].

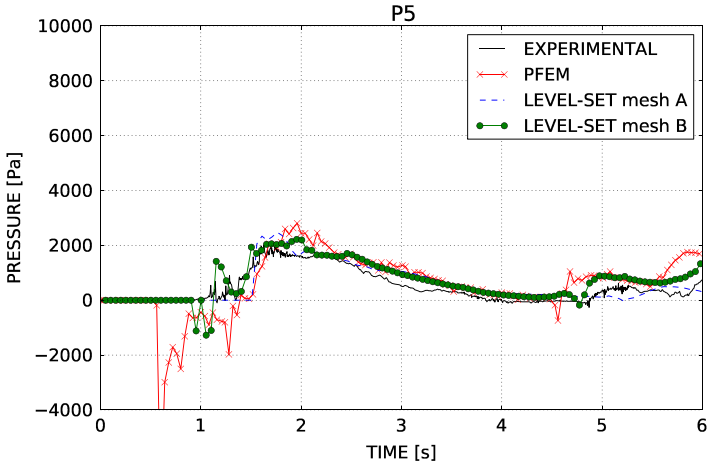
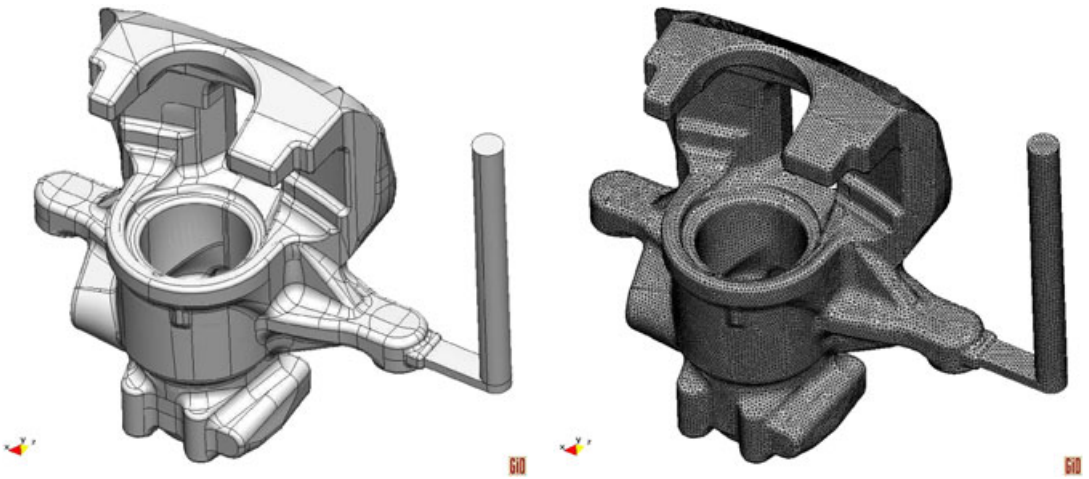


Figure 23. Pressure evolution on pressure sensor P5 (see Figure 19). Level set, PFEM and experimental comparison.



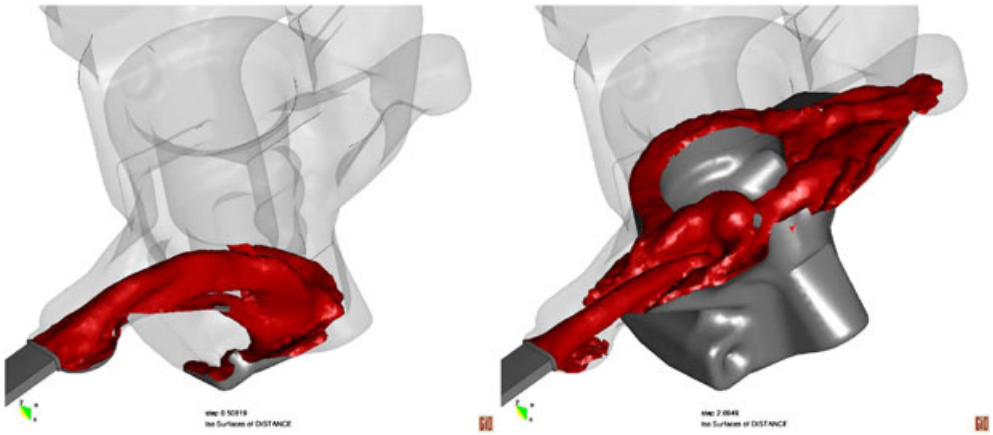
(a) Geometry (b) Mesh

Figure 24. The geometry of the breaking system part and the mesh of 678 047 tetrahedra used for simulation.

DOI: 10.1002/fld

Table VI. Number of nodes, number of elements and elemental length of the mesh used for the braking system part filling simulation.

	UNSTRUCTURED
Number of nodes	134128
Number of elements	678047
Elemental length [m]	0.0015



(a) detail a. (b) detail b.

Figure 25. The sequence of filling for break part simulation.

A better behavior of the Eulerian approach with respect to PFEM can be appreciated. The refinement of the mesh improves the accuracy of the solution and the capability of catching the second pressure waves with a correct timing, whereas a clear delay can be noticed in the case of the coarse mesh.

5.5. *Industrial application. Mold filling*

This example shows the robustness of the method in dealing with complex *industrial* geometries. The geometry shown in Figure 24(a) is a part of vehicles braking system filled with aluminum. The simulation consists of filling in low velocity (gravity filling) with 57 cm^3/s inlet resulting in 8.58 s filling time. This very low filling velocity is an additional difficulty as described in previous examples. The Figure 24(b) shows the unstructured mesh used for simulation. The characteristics of the mesh can be found in Table VI.

Figures 25 and 26 show the results obtained by the simulation. A detail of the free surface can be seen in Figure 25, whereas a sequence of the filling process is presented in Figure 26.

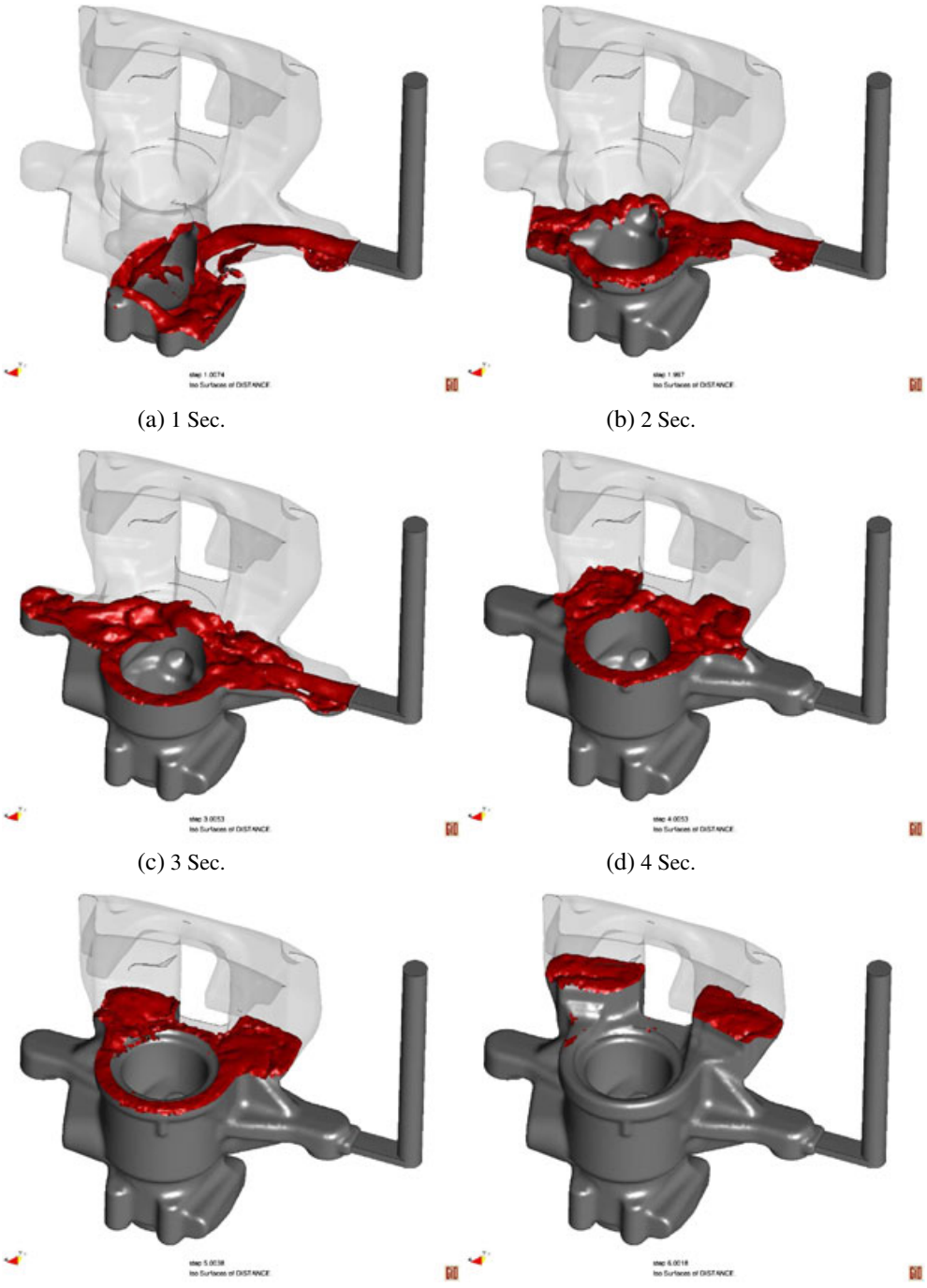
Finally, Figure 27 shows the speedup of the algorithm over two different platforms: one with Intel i7 Quad-core CPU and the other with 2 Quad-core AMD opteron CPUs. The tendency follows the same as the vertical channel. In this example the memory allocation is not optimized for Non-Uniform Memory Access (NUMA) machines, which leads to an asymptotic behavior over AMD NUMA machine. This can be improved using OpenMP first touch policy. In this way, OpenMP would allocate the data touched by each CPU in its local memory preventing poor memory access.

It is interesting to highlight that the same geometry was also used for the simulation a fast filling process, that is, filled in less than 1 s, and the results were also found to be satisfactory.

6. CONCLUSIONS

The current work presents an efficient level set solver, which allows the solution of complex industrial problems. The main new features of the proposed method are the technique used for the

DOI: 10.1002/fld



(a) 1 Sec. (b) 2 Sec.

(c) 3 Sec. (d) 4 Sec.

(e) 5 Sec. (f) 6 Sec.

Figure 26. The sequence of filling for break part simulation.

imposition of the pressure on the free surface, which is constructed on the top of the FS algorithm, and the proposal of an efficient volume-correction technique, which improves the volume conservation properties of the algorithm. The excellent volume conservation properties of the proposed approach are proved via a number of benchmarks, whereas its robustness is shown by tackling the solution of industrial graded mold-filling processes. The results are also compared with a well-established Lagrangian technique proving an excellent agreement, both to experimental results and

DOI: 10.1002/fld

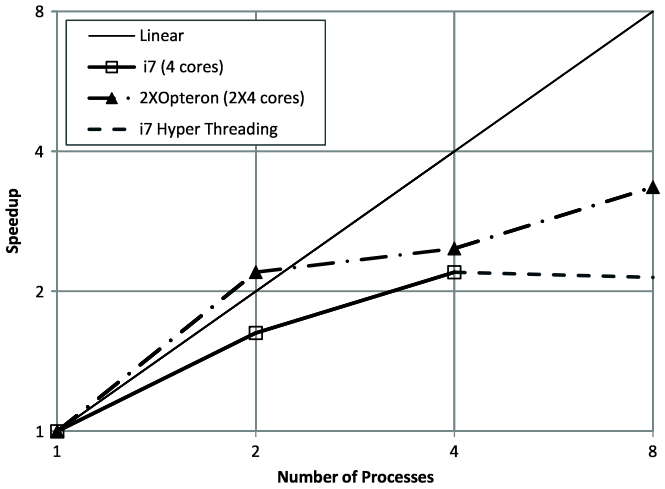


Figure 27. The speedup of the solver over different multi-cores machines.

the reference solution. The solver is implemented in OpenMP targeting commodity hardware, and the implementation is benchmarked and discussed.

ACKNOWLEDGEMENTS

The research was supported by the E-DAM project of the National Plan R+D of the Spanish Ministry of Science and Innovation I+D BIA2010-21350-C03-00.

REFERENCES

1. Fries T, Matthies H. Classification and overview of meshfree methods. *Technical Report*, Technische Universitat Braunschweig, Brunswick, Germany, 2003.
2. Oñate E, Idelsohn S, Celigueta M, Rossi R. Advances in the particle finite element method for the analysis of fluidmultibody interaction and bed erosion in free surface flows. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**:1777–1800.
3. Lohner R. *Applied Computational Fluid Dynamics Techniques*. Wiley: UK, 2001.
4. Osher SJ, Fedkiw RP. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag: Berlin, 2002.
5. Enright D, Fedkiw R, Ferziger J, Mitchell I. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics* 2002; **183**:83–116.
6. Oñate E, Idelsohn S, Pin FD, Aubry R. The particle finite element method. An overview. *International Journal of Computational Methods* 2004; **1**:267–307.
7. Idelsohn S, Oñate E, Pin FD, Calvo N. Fluid-structure interaction using the particle finite element method. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:2100–2123.
8. Idelsohn S, Oñate E, Pin FD. The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *International Journal for Numerical Methods in Engineering* 2004; **61**:964–984.
9. Larese A, Rossi R, Oñate E, Idelsohn S. Validation of the particle finite element method (PFEM) for simulation of free surface flows. *Engineering Computations* 2008; **25**:385–425.
10. Oñate E, Celigueta M, Idelsohn S. Modeling bed erosion in free surface flows by the particle finite element. *Acta Geotechnica* 2006; **1**:237–252.
11. Butler K, Oñate E, Idelsohn S, Rossi R. Modeling polymer melt flow using the particle finite element method (PFEM). *Proceeding of the Interflam Conference 2007* 2007; **2**:929–940.
12. Carbonell J, Oñate E, Suárez B. Modeling of ground exavation with the particle finite element method (PFEM). *ASCE: Journal of Engineering Mechanics* 2008; **136**:455–463.
13. Roubtsova V, Kahawita R. The SPH technique applied to free surface flows. *Computers and Fluids* 2006; **35**:1359–1371.
14. Cleary P, Ha J, Prakash M, Nguyen T. Short shots and industrial case studies: understanding fluid flow and solidification in high pressure die casting. *Applied Mathematical Modelling* 2010; **34**:2018–2033. DOI: 10.1016/j.apm.2009.10.015.

15. Elias R, Coutinho A. Stabilized edge-based finite element simulation of free-surface flows. *International Journal For Numerical Methods in Fluids* 2007; **54**:965–993.
16. Kees C, Akkerman I, Farthing M, Bazilevs Y. A conservative level set method suitable for variable-order approximations and unstructured meshes. *Journal of Computational Physics* 2011; **230**:4536–4558.
17. Coppola H. A finite element model for free surface and two fluid flows on fixed meshes. *Ph.D. Thesis*, Universidad Polititécnica de Cataluña, 2009.
18. Kratos, multiphysics opensource FEM code, 2012. (Available from: http://www.cimne.com/kratos).
19. Dadvand P, Rossi R, Oñate E. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of Computational Methods in Engineering* 2010; **17**:253–297.
20. Buscaglia GC, Ausas RF. Variational formulations for surface tension, capillarity and wetting. *Computer Methods in Applied Mechanics and Engineering* 2011; **200**:3011–3025.
21. Limache A, Idelsohn SR, Rossi R, Oñate E. The violation of objectivity in laplace formulations of the navierstokes equations. *International Journal For Numerical Methods in Fluids* 2007; **54**:639–664. DOI: 10.1002/fld.1480.
22. Codina R. Stabilization of incompressibility and convection through orthogonal sub-scales in finite element method. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:1579–1599.
23. Oñate E, Valls A, García J. Modeling incompressible flows at low and high reynolds numbers via finite calculus-finite element approach. *Journal of Computational Physics* 2007; **224**:332–351.
24. Chorin A. A numerical method for solving incompressible viscous problems. *Journal of Computational Physics* 1967; **2**:12.
25. Temam R. Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionaires (i). *Archives for Rational Mechanics and Analysis* 1969; **32**:135.
26. Codina R. Pressure stability in fractional step finite element methods for incompressible flows. *Journal of Computational Physics* 2000; **170**:112–140.
27. Hairer E, Wanner G. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*. Springer: Berlin, 2004.
28. Ryzhakov P. Lagrangian fe methods for coupled problems in fluid mechanics. *Ph.D. Thesis*, Universitat Politécnica de Cataluña, 2010.
29. Idelsohn SR, Oñate E. The challenge of mass conservation in the solution of free-surface flows with the fractional-step method: problems and solutions. *International Journal for Numerical Methods in Biomedical Engineering* 2010; **26**:1313–1330.
30. Codina R. A nodal-based implementation of a stabilized finite element method for incompressible flow problems. *International Journal for Numerical Methods in Fluids* 2000; **33**:737–766.
31. May M. Implementation of a general algorithm for incompressible and compressible flows within the multi-physics code KRATOS and preparation of fluid-structure coupling. *Diploma Thesis*, Technische Universität München, 2008.
32. Mossaiby F, Rossi R, Dadvand P, Idelsohn S. OpenCL-based implementation of an unstructured edge-based finite element convection-diffusion solver on graphics hardware. *International Journal for Numerical Methods in Engineering* 2012; **89**:1635–1651. DOI: 10.1002/nme.3302.
33. Osher S, Fedkiw RP. Level set methods: an overview and some recent reults. *Journal of Computational Physics* 2001; **169**:463–502.
34. Osher S, Fedkiw RP. *Level Set Methods and Dynamic Implicit Surfaces*. Springer: Berlin, 2003.
35. Coppola Owen H. A finite element model for free surface and two fluid flows on fixed meshes. *Ph.D. Thesis*, UPC, BarcelonaTech, 2009.
36. Elias R, Martins M, Coutinho L. Simple finite element-based computation of distance functions in unstructured grids. *International Journal for Numerical Methods Engineering* 2007; **72**:1095–1110.
37. Sethian J. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences* 1995; **93**(4):1591–1595.
38. Coppola-Owen A, Codina R. Improving eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions. *International Journal For Numerical Methods in Fluids* 2005; **49**:1287–1304.
39. Juon R, Hager W. Flip bucket with and without deflectors. *Journal of Hydraulic Engineering* 2000; **126**:837–845.
40. Kleesfman K, Fekken G, Veldman A, Iwanowski B, Buchner B. A volume -of-fluid based simulation method for wave impact problems. *Journal of Computational Physics* 2005; **206**:363–393.
41. Akkerman I, Bazilevs Y, Kees C, Farthing M. Isogeometric analysis of free-surface flow. *Journal of Computational Physics* 2011; **230**:4137–4152.