INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **42**:697–716 (DOI: 10.1002/fld.532)

Numerical simulation of three-dimensional free surface flows

V. Maronnier[1,2], M. Picasso[1,*,†] and J. Rappaz[1]

[1] *Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland*
[2] *Calcom SA, Parc Scientifique, 1015 Lausanne, Switzerland*

SUMMARY

A numerical model is presented for the simulation of complex fluid flows with free surfaces in three space dimensions. The model described in Maronnier *et al.* (*J. Comput. Phys.* 1999; **155**(2):439) is extended to three dimensional situations. The mathematical formulation of the model is similar to that of the volume of fluid (VOF) method, but the numerical procedures are different.

A splitting method is used for the time discretization. At each time step, two advection problems—one for the predicted velocity field and the other for the volume fraction of liquid—are to be solved. Then, a generalized Stokes problem is solved and the velocity field is corrected. Two different grids are used for the space discretization. The two advection problems are solved on a fixed, structured grid made out of small cubic cells, using a forward characteristic method. The generalized Stokes problem is solved using continuous, piecewise linear stabilized finite elements on a fixed, unstructured mesh of tetrahedrons.

The three-dimensional implementation is discussed. Efficient postprocessing algorithms enhance the quality of the numerical solution. A hierarchical data structure reduces memory requirements.

Numerical results are presented for complex geometries arising in mold filling. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: free surface flows; volume of fluid; finite elements; characteristics methods; mould filling

1. INTRODUCTION

Industrial processes such as casting, injection or extrusion involve complex free surface phenomena that can nowadays be solved numerically using commercial codes. In three-dimensional situations, the motion of the free surface is generally too complex to be handled by front-tracking [1] or Lagrangian methods [2–4]. The Arbitrary Lagrangian–Eulerian method is also difficult to implement since the selection of the mesh velocity is nontrivial for complex flows.

An alternative is to consider an Eulerian approach, which consists in using a fixed mesh but adding an unknown function φ. The function φ can be for instance, the volume fraction

* Correspondence to: M. Picasso, Department de Mathematiques, Ecole Polytechnique Federale de Lausanne, CH-1015 Lausanne, Switzerland.
† E-mail: marco.picasso@epfl.ch

Received 30 July 2001
Revised 14 June 2002

of liquid (the liquid characteristic function) or the pseudo-concentration of liquid. Since the interface moves with the fluid, φ must satisfy the following advection equation

$$\frac{\partial \varphi}{\partial t} + \mathbf{v} \cdot \nabla \varphi = 0$$

where $\mathbf{v}$ is the velocity field of the fluid. When two-phase flows are considered, the velocity field $\mathbf{v}$ is computed in the whole cavity (liquid and gas for instance), thus level set techniques [5, 6] or the pseudo-concentration method [7–12] can be applied. Then, the function φ is smooth, for instance positive in the liquid, negative in the gas and the interface corresponds to the zero level set.

In mould filling applications, the dynamics of the gas surrounding the liquid are not essential to describe the free surface, and the velocity field $\mathbf{v}$ can be computed only in the liquid region of the cavity. Moreover, the surface tension effects can be neglected at first order. Then, the function φ corresponds to the volume fraction of liquid, has value one in the liquid region, zero in the surrounding gas and jumps across the interface. The most famous numerical implementation of this model is the so-called Volume Of Fluid (VOF) method, which was originally devised for finite volumes [13] and recently extended to finite elements [14]. The major difficulty in solving this problem is then due to the fact that the volume fraction of liquid φ is discontinuous across the interface, thus numerical diffusion must be reduced as much as possible.

In this paper, eventhough we follow the mathematical formulation of the VOF method, the numerical treatment of the model is different. An implicit splitting algorithm is used to decouple advection and diffusion phenomena. Then, these two phenomena are solved using two different grids. Advection phenomena (including the motion of the volume fraction of liquid and the prediction of the fluid velocity) are solved using a fixed, structured grid of cubic cells and a forward characteristic method. On the other hand, diffusion phenomena (more precisely a generalized Stokes problem) are solved using continuous, piecewise linear stabilized finite element techniques on a fixed, unstructured mesh of tetrahedrons. The features are then the following. Because of the implicit character of the splitting scheme, the method is unconditionally stable with respect to the Courant-Friedrichs-Lewy (CFL) condition. Moreover, since two different grids are used for diffusion and advection, numerical diffusion in the volume fraction of liquid φ can be reduced by using a smaller grid for advection than for diffusion. Numerical results in two space dimensions have already been presented in Reference [15]. The goal of this paper is to extend the model to three-dimensional situations.

The structure of the paper is the following. In the next section, the governing equations and boundary conditions are presented. In the third section our splitting algorithm is proposed. The fourth section is devoted to the details of the three dimensional implementation. In Section 5, numerical simulations are presented.

2. THE MATHEMATICAL MODEL

2.1. Governing equations

The model presented in this section mainly corresponds to the one presented in the original VOF method [13]. However, as this will be explained in Sections 3 and 4, the numerical treatment of the model is different.

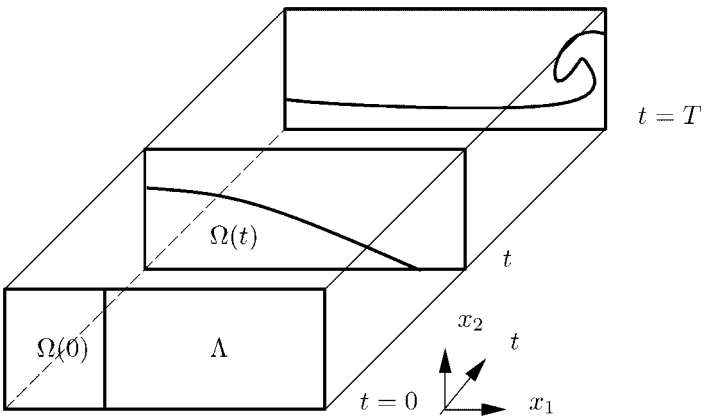

Figure 1. Calculation domain for the broken dam problem in a confined domain. At initial time, the fluid is at rest on the left part of the cavity. It is then free to move and hits the boundary.

Let Λ be a cavity of $\mathbb{R}^3$ in which the fluid must be confined, and let $T>0$ be the final time of the simulation. For any given time t, let $\Omega(t)$ denote the region occupied by the liquid. Finally, let Q_T be the space–time domain containing the liquid and let Σ_T be the space–time free surface between the liquid and the surrounding gas. The notations are reported in Figure 1 in the frame of a two dimensional situation, namely the broken dam problem in a confined domain.

The velocity field $\mathbf{v}: Q_T \to \mathbb{R}^3$ and the pressure field $p: Q_T \to \mathbb{R}$ are assumed to satisfy the time-dependent, incompressible Navier-Stokes equations in conservative form and in the presence of a gravity field $\mathbf{g}$; that is,

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} - 2\mu \operatorname{div} \mathbf{D}(\mathbf{v}) + \nabla p = \rho \mathbf{g} \quad \text{in } Q_T \tag{1}$$

$$\operatorname{div} \mathbf{v} = 0 \quad \text{in } Q_T \tag{2}$$

where $\mathbf{D}(\mathbf{v}) = \frac{1}{2}(\nabla \mathbf{v} + \nabla \mathbf{v}^{\mathrm{T}})$ is the rate of deformation tensor.

Let $\varphi: \Lambda \times [0, T] \to \mathbb{R}$ be the volume fraction of liquid. The function φ equals one if liquid is present and zero if it is not. Since the interface moves with the liquid, the function φ must satisfy (in a weak sense)

$$\frac{\partial \varphi}{\partial t} + \mathbf{v}\cdot\nabla\varphi = 0 \quad \text{on } \Sigma_T \tag{3}$$

From a Lagrangian point of view, the function φ is constant along the trajectories of the fluid particles. More precisely, $\varphi(X(t), t) = \varphi(X(0), 0)$, where $X(t)$ is the trajectory of a fluid particle, thus $X'(t) = \mathbf{v}(X(t), t)$.

2.2. Initial and boundary conditions

The initial conditions are the following. At initial time, the volume fraction of liquid $\varphi(\cdot,0)$ is given, which defines the liquid region at initial time,

$$\Omega(0)=\{x\in\Lambda;\varphi(x,0)=1\}$$

see Figure 1 for notations. The initial velocity field $\mathbf{v}$ is then prescribed in $\Omega(0)$. Let us now turn to the boundary conditions for the velocity field. It is assumed that no forces are acting on the free surface (capillary forces and forces due to external pressure of the surrounding gas are neglected); thus the stress is zero on the free surface,

$$-p\mathbf{n}+2\mu\mathbf{D}(\mathbf{v})\mathbf{n}=0 \quad \text{on } \Sigma_T \tag{4}$$

where $\mathbf{n}$ is the outward unit normal of the free surface. On the boundary of the liquid region being in contact with the walls (that is to say the boundary of Λ; see Figure 1), two cases can be considered. The first case, namely Dirichlet boundary conditions, corresponds to noslip or inflow conditions, the three components of the velocity being imposed. In the second case, slip or zero force boundary conditions apply, according to the flow. More precisely, if the fluid pushes against the wall, then slip boundary conditions are imposed; that is, zero normal velocity and zero tangent stress

$$\text{if} \quad (-p\mathbf{n}+2\mu\mathbf{D}(\mathbf{v})\mathbf{n})\cdot\mathbf{n}<0 \quad \text{then} \quad \mathbf{v}\cdot\mathbf{n}=0 \quad \text{and} \quad (-p\mathbf{n}+2\mu\mathbf{D}(\mathbf{v})\mathbf{n})\cdot\mathbf{t}_i=0, \quad i=1,2$$

where $\mathbf{t}_i$, $i=1,2$, are two unit vectors orthogonal to $\mathbf{n}$. On the other side, if the fluid does not push against the wall, then the stress is zero:

$$\text{if} \quad (-p\mathbf{n}+2\mu\mathbf{D}(\mathbf{v})\mathbf{n})\cdot\mathbf{n}\geqslant 0 \quad \text{then} \quad -p\mathbf{n}+2\mu\mathbf{D}(\mathbf{v})\mathbf{n}=0$$

We have considered this kind of boundary condition in order to prevent the fluid sticking onto the walls.

Let us briefly comment the zero force condition, Equation (4). Consider the situation of Figure 2, namely the filling of a two-dimensional S-shaped cavity (the experiment is described in Reference [15]). When filling the cavity with liquid, the air between the valve and the liquid can escape, thus condition (4) applies on the upper part of the liquid–air interface. However, since a fraction of the air is trapped by the liquid and cannot escape, a resulting force acts on the lower part of the liquid–air interface, this being not considered in our model.

3. TIME DISCRETIZATION: AN IMPLICIT SPLITTING ALGORITHM

A splitting algorithm is used to solve problem (1)–(3), allowing advection and diffusion phenomena to be decoupled.

Let $0=t^0<t^1<t^2<\cdots<t^N=T$ be a subdivision of the time interval $[0,T]$, define $\tau^n=t^n-t^{n-1}$ the nth time step, $n=1,2,\ldots,N$, τ the largest time step. Given an integer n, assume that approximations φ^{n-1}, $\mathbf{v}^{n-1}$, Ω^{n-1} of $\varphi(t^{n-1})$, $\mathbf{v}(t^{n-1})$, $\Omega(t^{n-1})$ respectively, are available. Then φ^n, $\mathbf{v}^n$, Ω^n are computed by means of a splitting algorithm. Firstly, two advection problems are solved, leading to a prediction of the new velocity $\mathbf{v}^{n-1/2}$ together with the new volume of fraction φ^n, which allows the new computational domain Ω^n to be defined. Secondly, a generalized Stokes problem is solved and the velocity $\mathbf{v}^n$ is obtained in Ω^n.

Figure 2. Filling of an *S*-shaped cavity. The air in the upper part of the cavity is free to escape from the valve. The air trapped by the liquid may exert a force on the liquid, this being neglected in our model.

3.1. Advection step

Solve between time t^{n-1} and t^n the two advection problems

$$\frac{\partial \mathbf{w}}{\partial t} + (\mathbf{w} \cdot \nabla)\mathbf{w} = 0$$

$$\frac{\partial \psi}{\partial t} + \mathbf{w} \cdot \nabla \psi = 0$$

with initial conditions

$$\mathbf{w}(t^{n-1}) = \mathbf{v}^{n-1}$$

$$\psi(t^{n-1}) = \varphi^{n-1}$$

If the effect of the boundary of the cavity Λ is not considered, these two problems can be solved exactly, using the method of Characteristics [16–20], the trajectories of the velocity field $\mathbf{w}$ being straight lines. Indeed, the trajectories are given by $X'(t) = \mathbf{w}(X(t), t)$, but since $\mathbf{w}$ is constant along the trajectories, we have $X'(t) = \mathbf{w}(X(t^{n-1}), t^{n-1}) = \mathbf{v}^{n-1}(X(t^{n-1}))$. Let $\mathbf{v}^{n-1/2}$ denote the solution of the first advection problem at time t^n ($\mathbf{v}^{n-1/2}$ is a prediction of the velocity field), $\mathbf{v}^{n-1/2} = \mathbf{w}(t^n)$. Let φ^n denote the solution of the second advection problem at time t^n, i.e. $\varphi^n = \psi(t^n)$. We thus have

$$\mathbf{v}^{n-1/2}(x + \tau^n \mathbf{v}^{n-1}(x)) = \mathbf{v}^{n-1}(x) \tag{5}$$

$$\varphi^n(x + \tau^n \mathbf{v}^{n-1}(x)) = \varphi^{n-1}(x) \tag{6}$$

for all x belonging to Ω^{n-1}. Once φ^n is known in the cavity Λ, then the liquid region at time t^n is defined by:

$$\Omega^n = \{ y \in \Lambda; \varphi^n(y) = 1 \}$$

3.2. Diffusion step

It remains to solve the following generalized Stokes problem:

$$\rho \frac{\mathbf{v}^n - \mathbf{v}^{n-1/2}}{\tau^n} - 2\mu \operatorname{div} \mathbf{D}(\mathbf{v}^n) + \nabla p^n = \rho \mathbf{g} \quad \text{in } \Omega^n \tag{7}$$

$$\operatorname{div} \mathbf{v}^n = 0 \quad \text{in } \Omega^n \tag{8}$$

with the boundary conditions described in Section 2.2 (Dirichlet, slip, or zero force boundary conditions).

In the simple case when the free surface problem is disregarded ($\varphi = 1$ everywhere in the cavity), then this splitting is closely linked to the so-called Characteristics-Galerkin method [16–18]. The major difference comes from the fact that, in the Characteristics-Galerkin method, the trajectories of the fluid particles are computed in the direction opposite to the flow, whereas our method computes them in the flow direction. Thus, we expect our algorithm to be $O(\tau)$ convergent without any stability restrictions on the time step.

4. SPACE DISCRETIZATION AND IMPLEMENTATION

Since advection and diffusion phenomena are now decoupled, two distinct grids can be used. Finite element techniques are well suited for solving (7), (8) on an unstructured mesh of the cavity containing the liquid. Indeed, the use of finite elements for mould filling applications is important since the shape of industrial moulds may be very complex (engine carters, turbine blades,...). On the other hand, a structured grid of cubic cells is used to implement (5) and (6).

4.1. Advection step

Assume that the grid is made out of cubic cells of size h, each cell being labelled by indices (ijk). Let φ_{ijk}^{n-1} and $\mathbf{v}_{ijk}^{n-1}$ be the approximate value of φ and $\mathbf{v}$ at the centre of cell number (ijk) at time t^{n-1}. According to (5) and (6), the advection step on cell number (ijk) consists in advecting φ_{ijk}^{n-1} and $\mathbf{v}_{ijk}^{n-1}$ by $\tau^n \mathbf{v}_{ijk}^{n-1}$ and then projecting the values on the structured grid. An example of cell advection and projection is presented in Figure 3 in two space dimensions.

This advection algorithm is unconditionally stable with respect to the Courant-Friedrichs-Lewy (CFL) condition, and $O(\tau + h^2/\tau)$ convergent, according to the theoretical results available for the Characteristics-Galerkin method [16–18]. However, this algorithm has two drawbacks. Indeed, numerical diffusion is introduced when projecting the values of the advected cells on the grid (remember that the volume fraction of liquid is discontinuous across the interface). Moreover, if the time step is too large, two cells may arrive at the same place, producing numerical (artificial) compression.

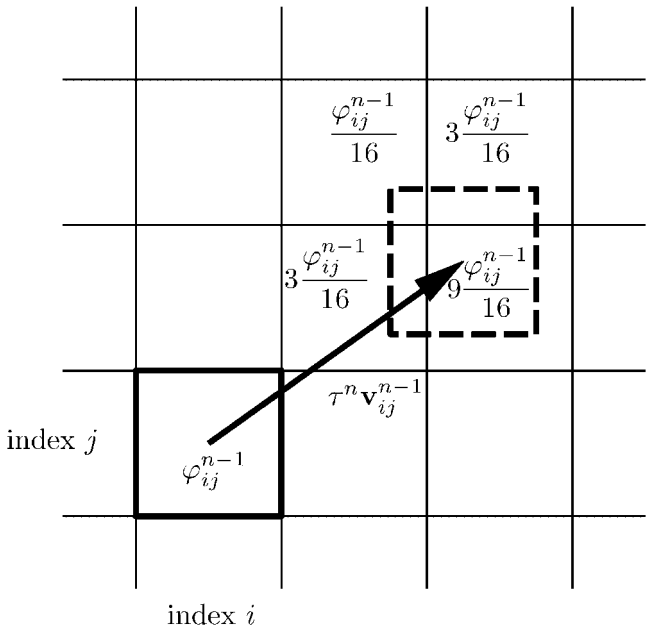

Figure 3. An example of two dimensional advection of φ_{ij}^{n-1} by $\tau^n \mathbf{v}_{ij}^{n-1}$, and projection on the grid. The advected cell is represented by the dashed lines. The four cells containing the advected cell receive a fraction of φ_{ij}^{n-1}, according to the position of the advected cell.

In order to enhance the quality of the volume fraction of liquid, two postprocessing procedures have been implemented, see also Reference [15] for a description in two space dimensions. The first procedure reduces numerical diffusion and is a simplified three-dimensional implementation of the simple linear interface calculation (SLIC) algorithm [21, 22]. The second procedure removes artificial compression using a quick sort algorithm.

4.1.1. Reducing numerical diffusion: a SLIC algorithm. Consider cell number (ijk) being partially filled with liquid (this results from numerical diffusion), let φ_{ijk}^{n-1} be the corresponding volume fraction of liquid, this value being less than one. Instead of advecting φ_{ijk}^{n-1} and then projecting on the grid, the liquid is first pushed on the sides of the cell, then it is advected and projected on the grid. A two-dimensional example is reported in Figure 4.

The critical point is then to decide how to push the volume fraction of liquid in a given cell along the sides of this cell. For a given cell, the choice depends on the volume fraction of liquid of the neighbours, see Figure 5 for an example in a two-dimensional situation.

In the original two-dimensional SLIC algorithm [21, 22], the number of possible cases was $3^4 = 81$, in three space dimensions it should be $3^6 = 729$, which is not reasonable. Therefore a simplified three-dimensional version of the algorithm has been implemented. The volume fraction of liquid in a given cell can be pushed according to the neighbours in three different manners: along a face, along an edge, or against a vertex. The 28 positions of the liquid

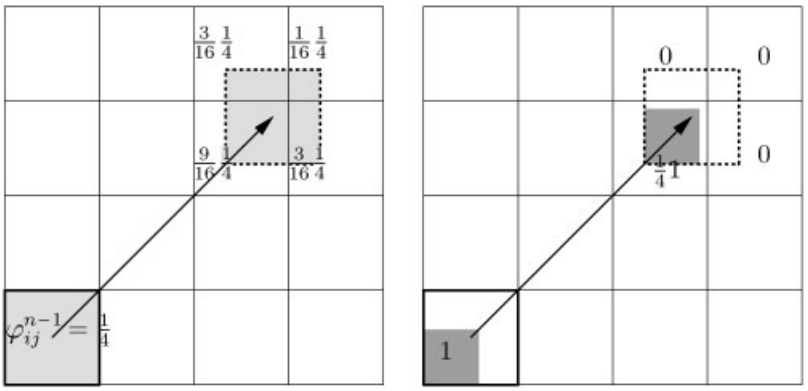

Figure 4. Effect of the SLIC algorithm on numerical diffusion. An example of two dimensional advection and projection when the volume fraction of liquid in the cell is $\varphi_{ij}^{n-1} = \frac{1}{4}$. Left: without SLIC, the volume fraction of liquid is advected and projected on four cells, with contributions (from top to bottom) $\frac{1}{4}\frac{3}{16}$, $\frac{1}{4}\frac{1}{16}$, $\frac{1}{4}\frac{9}{16}$, $\frac{1}{4}\frac{3}{16}$. Right: with SLIC, the volume fraction of liquid is pushed at one corner, then it is advected and projected on one cell only, with contribution $\frac{1}{4}$.

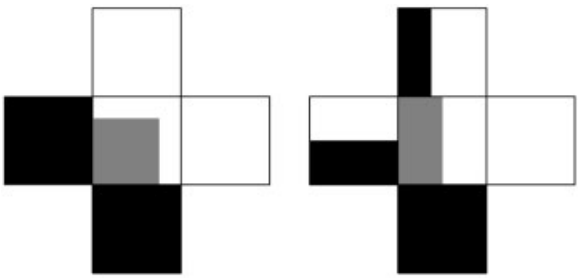

Figure 5. SLIC algorithm. The volume fraction of liquid in a cell partially filled with liquid is pushed according to the volume fraction of liquid of the neighbouring cells. Two examples are proposed. Left: the left and bottom neighbouring cells are full of liquid, the right and top neighbouring cells are empty, the liquid is pushed at the bottom left corner of the cell. Right: the bottom neighbouring cell is full of liquid, the right neighbouring cell is empty, the other two neighbouring cells are partially filled with liquid, the volume fraction of liquid is pushed along the left side of the cell.

within a cell are shown in Figure 6. The last two positions are not considered in the three-dimensional SLIC algorithm although they were in the original two-dimensional one. The full description of the algorithm is available in Reference [23] and we present hereafter the algorithm pertaining to only one of the 28 possible positions.

The situation of Figure 7 is considered. Let (i,j,k) be the indices of a cell partially filled with liquid, thus $0 < \varphi_{ijk}^{n-1} < 1$. We shall say that two cells are in the same state if they are both full of liquid (the volume fraction of liquid is one), both free of liquid (the volume fraction of liquid is zero), or both partially filled with liquid (the volume fraction of liquid

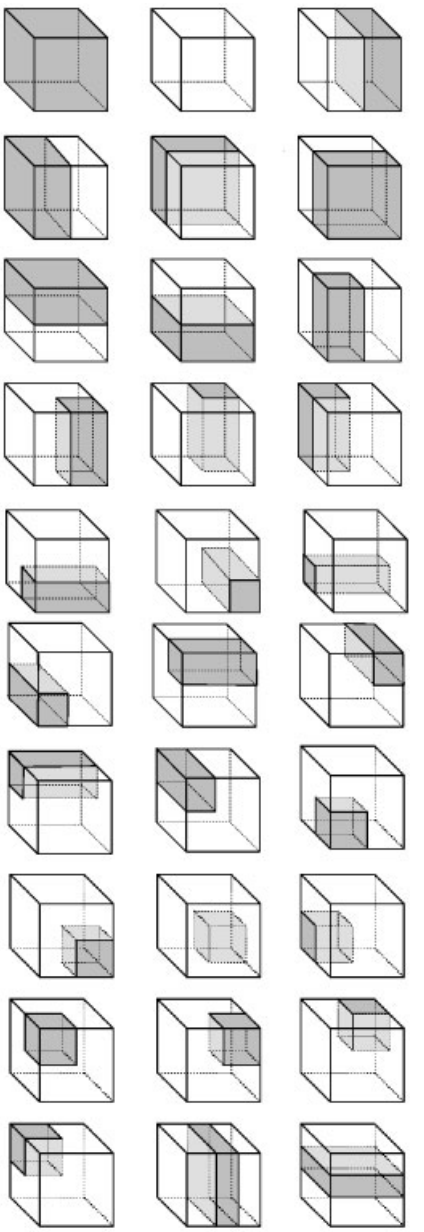

Figure 6. SLIC algorithm. Position of the liquid (shaded region) in a cubic cell partially filled. The volume fraction of liquid is pushed along a face, an edge, or a vertex of the cell according to the neighbours volume fraction of liquid. The possible number of positions is 28. In the first two figures, the cell is full of liquid or empty. The last two figures represent positions considered in the two-dimensional implementation, but discarded in the three dimensional one.

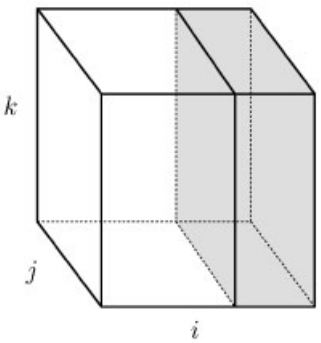

Figure 7. SLIC algorithm: an example. Cell (i,j,k) is partially filled with liquid. The liquid is pushed as in the figure when conditions (9) or (10) are satisfied.

is between zero and one). Then, the liquid in cell (i,j,k) is pushed as in Figure 7 if

$$\varphi_{i-1jk}^{n-1} < 1 \text{ and } \varphi_{i+1jk}^{n-1} = 1$$

$$\text{the cells } (i,j-1,k) \text{ and } (i,j+1,k) \text{ are in the same state} \tag{9}$$

$$\text{the cells } (i,j,k-1) \text{ and } (i,j,k+1) \text{ are in the same state}$$

or if

$$\varphi_{i-1jk}^{n-1} = 0 \text{ and } \varphi_{i+1jk}^{n-1} > 0$$

$$\text{the cells } (i,j-1,k) \text{ and } (i,j+1,k) \text{ are in the same state} \tag{10}$$

$$\text{the cells } (i,j,k-1) \text{ and } (i,j,k+1) \text{ are in the same state}$$

In order to show the importance of the SLIC algorithm, we have reproduced the numerical experiment reported in Reference [21] in three space dimensions, see Figure 8. Consider the cube $[0,20]^3$, meshed into $20 \times 20 \times 20$ cells. At initial time the liquid region is a block of $3 \times 7 \times 3$ cells, then rotates 180 degrees, the centre of rotation being the point $(11.5, 12.5, 11.5)$, the rotation axis being Ox_3. We have reported four experiments along the plane $x_3 = 11.5$, with several time steps. Clearly the SLIC algorithm reduces dramatically numerical diffusion and is essential to the numerical model.

4.1.2. Reducing artificial compression: a decompression algorithm. The algorithm implemented in order to reduce artificial compression is space dimension independent and is essentially the same than the one presented in Reference [15]. In the absence of this decompression algorithm, when the computed values of the volume fraction of liquid φ_{ijk}^n are greater than one, a fraction of the liquid contained in the cavity is lost. The decompression algorithm aims at producing new values φ_{ijk}^n which are bounded by zero and one.

At each time step, all the cells having value φ_{ijk}^n greater than one (strictly) or between zero and one (strictly) are sorted (using a standard quick sort algorithm) according to their

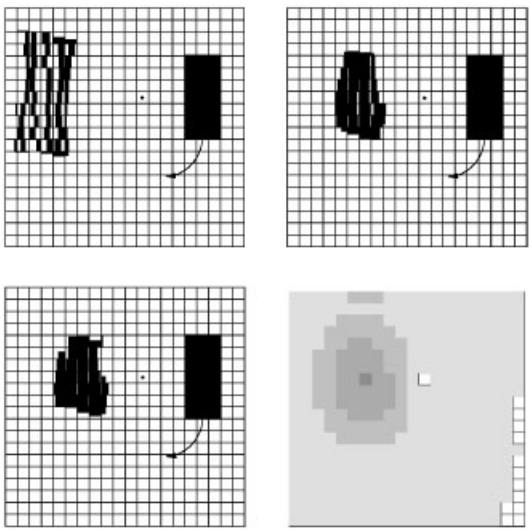

Figure 8. SLIC algorithm. Rotation of a liquid block by 180 degrees. Black corresponds to volume fraction of liquid one, white to volume fraction of liquid zero, grey to intermediate values. Top left: with SLIC, 11 times steps. Top right: with SLIC, 60 times steps. Bottom left: with SLIC, 1686 times steps. Bottom right: without SLIC, 60 times steps.

value φ^n_{ijk}. The cells having value φ^n_{ijk} greater than one are called the dealer cells, whereas the cells having value φ^n_{ijk} between zero and one are called the receiver cells. The decompression algorithm then consists in moving the volume fraction of liquid in excess in the dealer cells to the receiver cells. In the case when all the receiver cells are full but some liquid is still in excess in the dealer cells, then the amount of liquid in excess is stored in a dedicated buffer and is introduced at the next time step. Two-dimensional examples are reported in Reference [15].

4.2. *From cells to finite elements*

Once values φ^n_{ijk} and $\mathbf{v}^n_{ijk}$ have been computed on the cells, values have to be extrapolated at the nodes of the finite element mesh. Then, the new liquid region will be set, and the generalized Stokes problem (7) and (8) will be solved using tetrahedral finite elements. For any vertex P of the finite element mesh let ψ_P be the corresponding basis function (i.e. the continuous, piecewise linear function having value one at P, zero at the other vertices). We consider all the tetrahedrons K containing vertex P and all the cells (ijk) having centre of mass C_{ijk} contained in these tetrahedrons, see Figure 9 for a two-dimensional description. Then, φ^n_P, the volume fraction of liquid at vertex P and time t^n is computed using the following formula:

$$\varphi^n_P = \frac{\displaystyle\sum_{\substack{K \\ P\in K}} \sum_{\substack{ijk \\ C_{ijk}\in K}} \psi_P(C_{ijk})\varphi^n_{ijk}}{\displaystyle\sum_{\substack{K \\ P\in K}} \sum_{\substack{ijk \\ C_{ijk}\in K}} \psi_P(C_{ijk})}$$

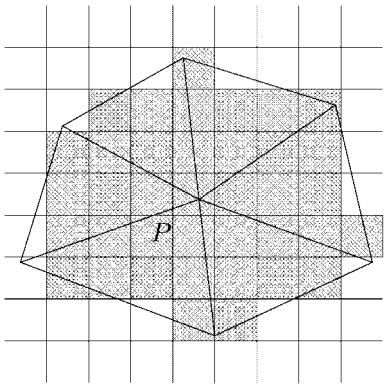

Figure 9. From cells to finite elements. The value of the volume fraction of liquid at vertex P depends on the values computed on the shaded cells.

The same kind of formula is used to obtain the predicted velocity $\mathbf{v}^{n-1/2}$ at the vertices of the finite element mesh.

4.3. *Diffusion step*

Once values of the velocity field $\mathbf{v}^{n-1/2}$ and of the volume fraction of liquid φ^n are available at the vertices of the finite element mesh, the liquid region is defined as follows. An element of the mesh is said to be liquid if (at least) one of its vertices P has a value $\varphi_P^n > 0.5$. The computational domain Ω^n used for solving (7) and (8) is then defined to be the union of all liquid elements.

Note that a velocity has to be guessed for each node having $\varphi_P^n = 0$, but belonging to a liquid element. The velocity is then clearly computed from the values of the neighbouring liquid nodes. Also note that, if a cell has a value φ_{ijk}^n greater than zero, but belongs to a finite element which is not liquid, then it is eliminated from the computations and a small amount of liquid is lost. When this situation occurs, the corresponding amount of liquid is stored in a dedicated buffer and is introduced at the next time step, during the decompression algorithm.

Let us now turn to the finite element techniques used for solving (7) and (8), the boundary conditions being those described in Section 2.2 (Dirichlet, slip, or zero force boundary conditions). The velocity and pressure are continuous, piecewise linear and a stabilized weak formulation is used [24–26], namely

$$\int_{\Omega^n} \frac{\mathbf{v}^n - \mathbf{v}^{n-1/2}}{\tau^n} \mathbf{w}\, \mathrm{d}x + 2\mu \int_{\Omega^n} \mathbf{D}(\mathbf{v}^n) : \mathbf{D}(\mathbf{w})\, \mathrm{d}x - \int_{\Omega^n} p^n \operatorname{div} \mathbf{w}\, \mathrm{d}x - \rho \mathbf{g} \int_{\Omega^n} \mathbf{w}\, \mathrm{d}x$$
$$- \int_{\Omega^n} \operatorname{div} \mathbf{u}^n q\, \mathrm{d}x - \sum_{K \subset \Omega^n} \alpha_K \int_K \left(\frac{\mathbf{v}^n - \mathbf{v}^{n-1/2}}{\tau^n} + \nabla p^n - \rho \mathbf{g} \right) \cdot \nabla q\, \mathrm{d}x = 0$$

Here $\mathbf{w}$ and q are the velocity and pressure test functions, compatible with the boundary conditions. The stability coefficient α_K is defined on each tetrahedron K as a function of the local Reynolds number $Re_K = \rho|\mathbf{v}^{n-1/2}|_\infty h_K/2\mu$ as follows:

$$\alpha_K = \frac{1}{12}\frac{h_K^2}{\mu} \quad \text{if } Re_K \leqslant 3$$
$$= \frac{1}{4Re_K}\frac{h_K^2}{\mu} \quad \text{if } Re_K > 3$$

Note that the stabilizing terms in the weak formulation are added in a consistent manner, $\operatorname{div}\mathbf{D}(\mathbf{v}^n)$ being zero in each tetrahedron since the velocity is piecewise linear.

The degrees of freedom are the three velocity components and pressure at each vertex of the finite element mesh. This finite element procedure is implicit and $O(H^2 + \tau)$ convergent (in the L^2 norm), H being the finite elements spacing and τ the greatest time step. At the moment, all the degrees of freedom are stored in a single matrix (monolithic scheme) and the linear system is solved using a BICGSTAB algorithm and a classical incomplete LU preconditioner. We are looking forward to using decoupling methods in order to reduce memory requirements, see References [27, 28] for a general discussion. Theoretical and numerical investigations have already been published for a time dependent Stokes problem [29]. Both the matrix-pressure method and Incomplete LU factorizations [30, 31] are possible candidates.

Finally, once the new velocity field $\mathbf{v}^n$ is computed at the vertices of the finite element mesh, values are interpolated at the centre of the cells (ijk). A classical Lagrange interpolation formula is used, namely

$$\mathbf{v}_{ijk}^n = \sum_P \psi_P(C_{ijk})\mathbf{v}_P^n$$

where P are the mesh vertices, ψ_P the corresponding basis functions, $\mathbf{v}_P^n$ the corresponding velocities, and C_{ijk} is the cell centre of mass.

4.4. A hierarchic data structure

In number of industrial mould filling applications, the shape of the cavity containing the liquid (the mould, an engine carter for instance) is complex. Therefore, a special data structure has been implemented in order to reduce the memory requirements used to store the cell data. An example is proposed in Figure 10. The cavity containing the liquid is meshed into tetrahedrons. Without any particular cells data structure, a great number of cells would be stored in the memory without being never used. The data structure we have adopted uses three levels to define the cells. At the coarsest level, the so-called window level, the cavity is meshed into blocks, which are glued together. Each window is then subdivided into cubes, this intermediate level is called the block level. Finally, each block is cut into smaller cubes, namely the cells (ijk).

When a block is free of liquid (empty), it is switched off, that is to say the memory corresponding to the cells is not allocated. When liquid enters a block, the block is switched on, that is to say the memory corresponding to the cells is allocated. This data structure was already used in industrial applications pertaining to dendritic solidification [32].

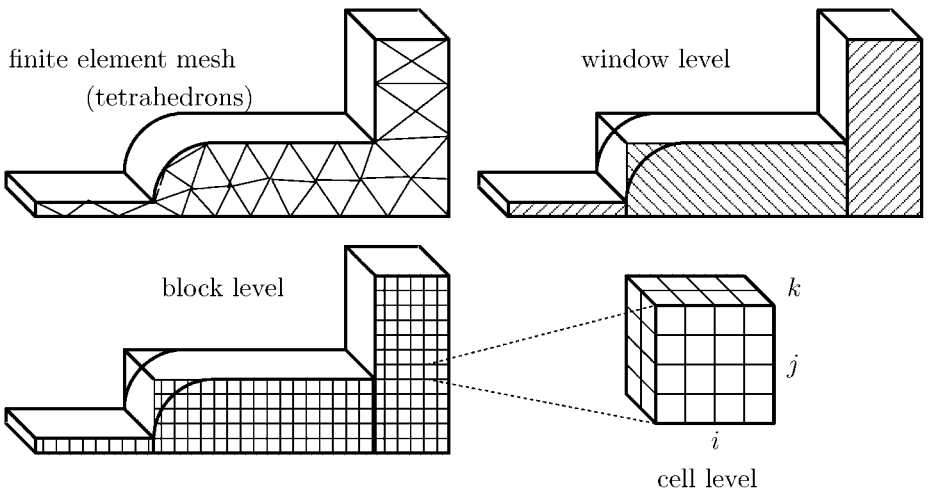

Figure 10. The hierarchical Window-block-Cell data structure used to implement the cells advection.

A graphical interface has been included in the CalcoSoft™ software in order to simplify the handling of this hierarchic data structure.

5. NUMERICAL RESULTS

5.1. Validation: comparison with two-dimensional computations

In order to validate the three-dimensional model, comparisons with the two-dimensional results of Reference [15] have been performed, see also Reference [23]. Only two comparisons are presented here, namely the disk with core test case and the broken dam problem. All the computations below were performed on a PC with Intel Pentium III 866 Mhz CPU and 512 Mb Memory. The results were post-processed with the CalcoSoft™ software.

The disk with core: Water is injected in a disk with a circular obstacle, lying between two horizontal planes. Experimental results are taken from Reference [33]. The dimensions are 0.14×0.16 m, water is entering at velocity 18 m/s, and the density and viscosity are $\rho = 1000\,\mathrm{kg/m^3}$, $\mu = 0.01\,\mathrm{kg/(ms)}$. The 2D mesh has 1879 vertices and 3556 triangles. The 3D mesh is built from the 2D one by elevation, using 5 horizontal layers; it has 9395 vertices and 42672 tetrahedrons. The hierarchical data structure for the cells is made out of a single window containing $54 \times 63 \times 4$ blocks, each block having $4 \times 4 \times 4$ cells, so that the finite element mesh spacing H is roughly three times the cell spacing h. The simulation required 180 time steps to reach the final time (18 ms), the CPU time for the 3D computation was 45 min. In Figure 11 we have reported the liquid region at several times. Clearly the agreement between computations and experiments is very good.

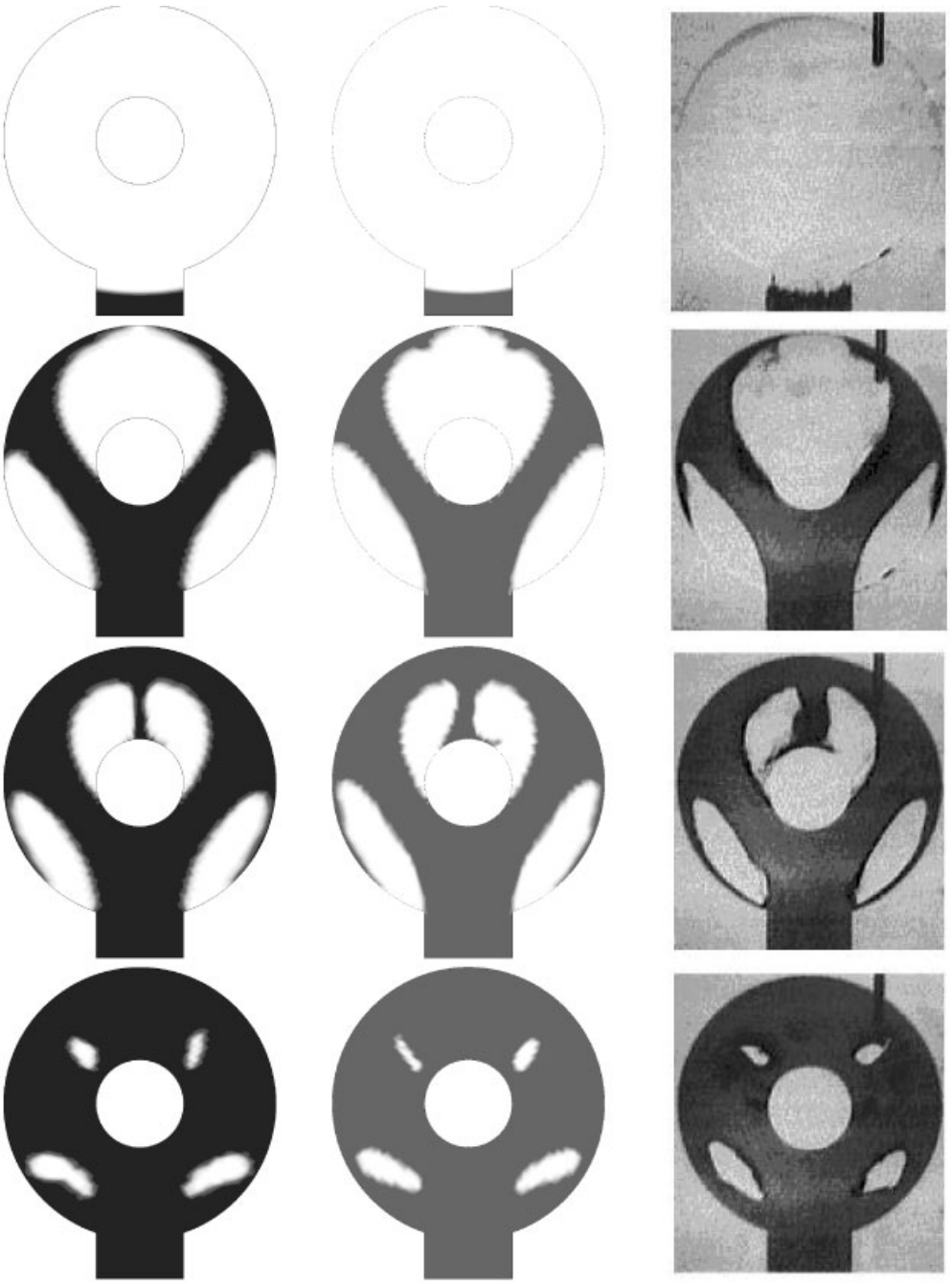

Figure 11. Disk with core: position of the liquid region at time 0, 8.8, 11.8 and 16.2 ms. Left: 2D numerical results, middle: 3D numerical results, right: experimental results [33].

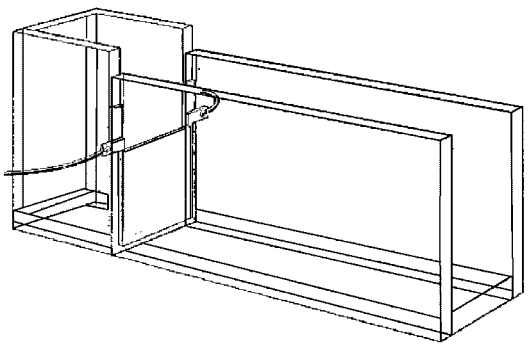

Figure 12. The experimental set-up of the broken dam problem [34].

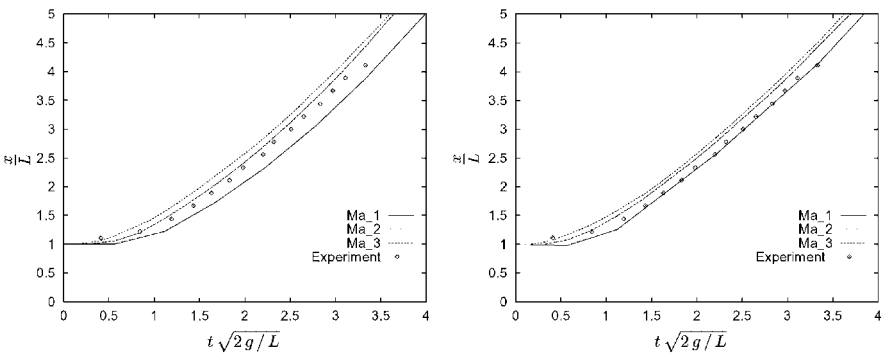

Figure 13. The broken dam problem. Dimensionless position x/L of the liquid front along the bottom of the cavity against dimensionless time $t\sqrt{2g/L}$. Left: 2D results, right: 3D results. The coarsest mesh is Ma_1, L denotes the width of the initial liquid region.

The broken dam problem: A block of water is initially kept in a cavity using a thin paper film. At time zero, the paper film is removed and water is free to move. Experimental results are reported in Reference [34], the experimental setup is reproduced in Figure 12. Three finite element meshes have been used to check convergence. For each mesh, the number of cells is such that the finite element mesh spacing H is roughly three times the cell spacing h. Also, three different time steps were used, the largest being 0.03 s. In Figure 13, we have reported the position of the free surface at several times. Again, the agreement between experimental values, 2D and 3D computations is excellent.

5.2. *Three-dimensional computations*

We are now in position to present three-dimensional computations involving complex shapes of the liquid region. In the first test case, namely the breakage of a dam in a confined cavity, the shape of the cavity is simple but the liquid region becomes complex. The second test case corresponds to the filling of a mould with five arms.

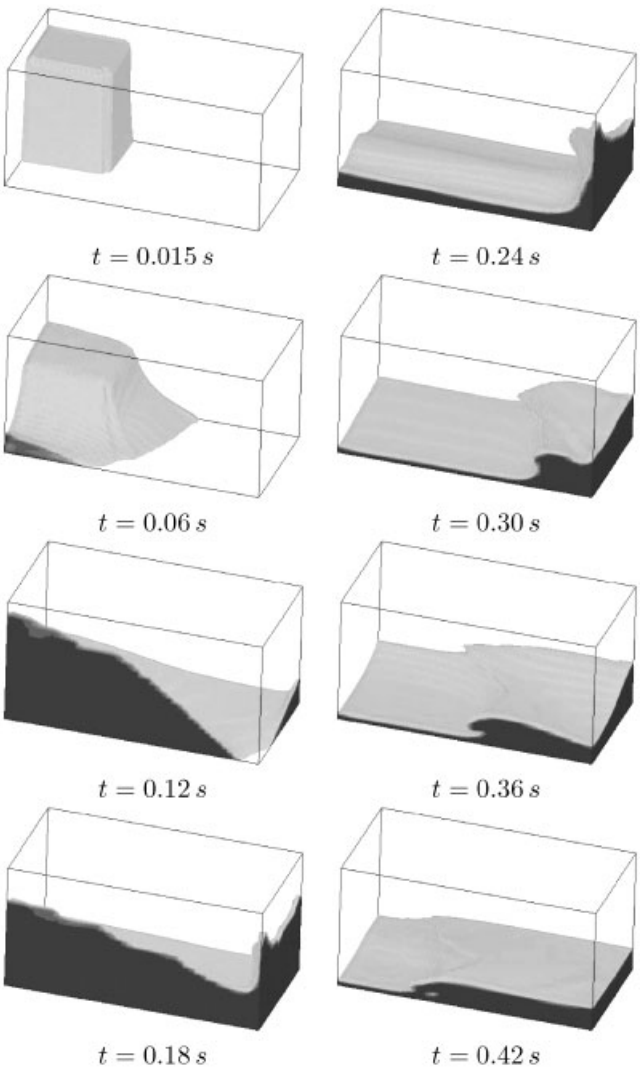

Figure 14. Breakage of a dam in a confined cavity.

The broken dam problem in a confined cavity: We consider a cavity with size $0.09 \times 0.045 \times 0.045$ m and place a block of water with size $0.03 \times 0.02 \times 0.04$ m along one of the vertical edges. The finite element mesh is obtained by cutting the cavity into $45 \times 25 \times 25$ bricks, then cutting each brick in 6 tetrahedrons. A single window is used, with $45 \times 23 \times 23$ blocks made out of $5 \times 5 \times 5$ cells. At initial time water is free to move under action of gravity; it then splashes against the walls of the cavity. The final time is 0.5 s and 500 time steps were used. Numerical results are reported in Figure 14. The CPU time was 277 min.

Filling of a mould with five arms: We consider a mould with five arms, see Figures 15 and 16. Water is entering downwards in a vertical tube at speed 2.2 m/s. The mesh has 3565

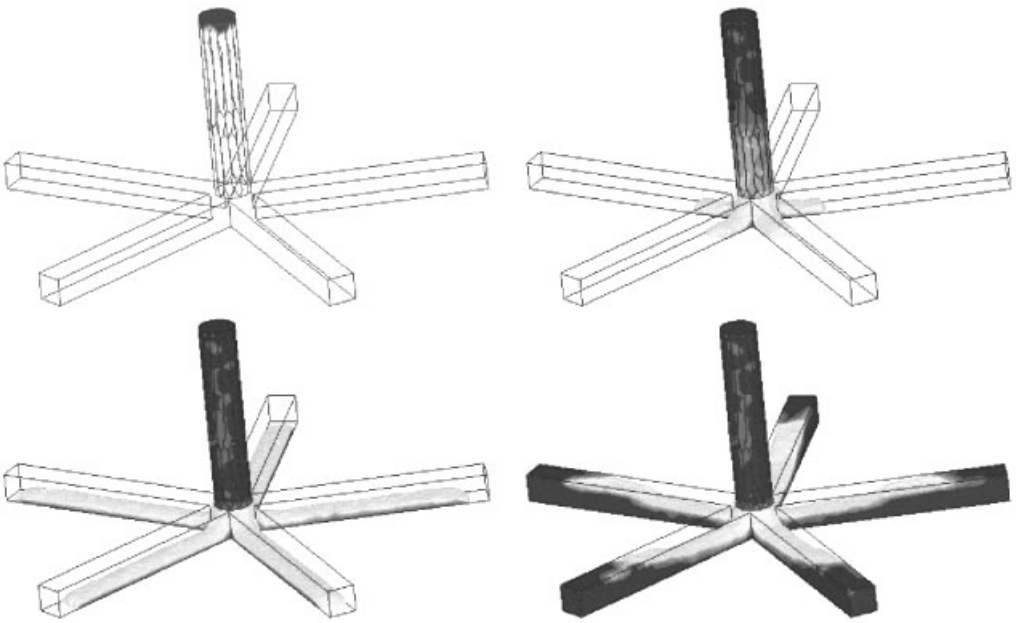

Figure 15. Filling of a mould with five arms: position of the free surface between time 0 and 0.5 s.

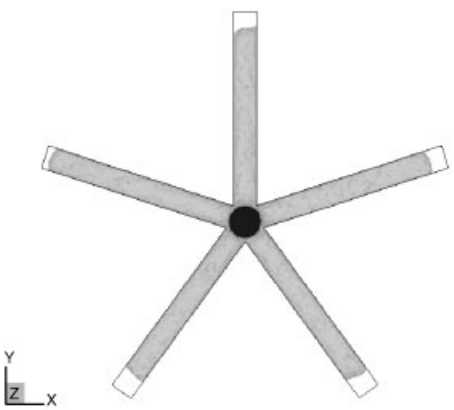

Figure 16. Filling of a mould with five arms: top view of the interface before it hits the end of the five arms.

vertices and 14962 tetrahedrons. As reported in Figure 17, seven windows and 31568 blocks were used to reduce memory requirements, each bloc having $5 \times 5 \times 5$ cells. The final time is 0.5 s and 500 time steps were used. The CPU time was 141 min. For this test case, the cells cannot be aligned with the five arms simultaneously and the goal is to check that filling occurs at the same speed in all arms.

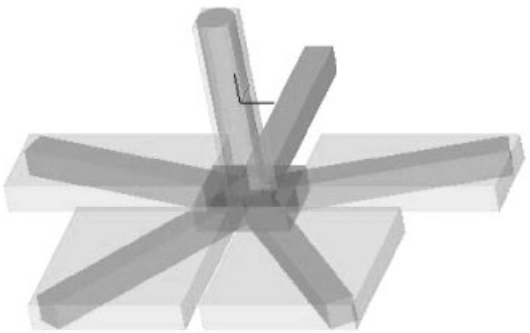

Figure 17. Filling of a mould with five arms: description of the seven windows.

6. CONCLUSION AND PERSPECTIVES

A three-dimensional free surface flow solver has been presented. The volume fraction of liquid is used to describe the interface. A splitting algorithm allows diffusion and convection phenomena to be decoupled. Diffusion is solved with finite elements using an unstructured mesh of tetrahedrons, whereas convection is solved on a grid made out of small cubic cells. Numerical diffusion and compression is minimized using appropriate post-processing algorithms. A hierarchic data structure allows complex geometries to be handled.

Numerical results show the efficiency of this approach. This three-dimensional free surface flow solver has been incorporated in a commercial code, the CalcoSoft^{TM} software, and is used for mould filling simulations.

REFERENCES

1. LeVeque RJ, Shyue K-M. Two dimensional front tracking based on high resolution wave propagation methods. *Journal of Computational Physics* 1996; **123**:354–368.
2. Hansbo P. The characteristic streamline diffusion method for the time-dependent incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1992; **99**:171–186.
3. Hansbo P. Lagrangian incompressible flow computations in three dimensions by use of space–time finite elements. *International Journal for Numerical Methods in Fluids* 1995; **20**:989–1001.
4. Muttin F, Coupez T, Bellet M, Chenot JL. Lagrangian finite-element analysis of time-dependent viscous free-surface flow using an automatic remeshing technique: Application to metal casting flow. *International Journal for Numerical Methods in Engineering* 1993; **36**:2001–2015.
5. Chang YC, Hou TY, Merriman B, Osher S. A level set formulation of Eulerian interface capturing methods for incompressible fluid flows. *Journal of Computational Physics* 1996; **124**(2):449–464.
6. Sussman M, Almgren AS, Bell JB, Colella P, Howell PH, Welcome ML. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics* 1999; **148**(1):81–124.
7. Codina R, Schäfer U, Onate E. Mould filling simulation using finite elements. *International Journal for Numerical Methods in Heat and Fluid Flow* 1994; **4**:291–310.
8. Dhatt G, Gao DM, Ben Cheikh A. A finite element simulation of metal flow in moulds. *International Journal for Numerical Methods in Engineering* 1990; **30**:821–831.
9. Lock N, Jaeger M, Medale M, Occelli R. Local mesh adaptation technique for front tracking problems. *International Journal for Numerical Methods in Fluids* 1998; **28**:719–736.
10. Medale M, Jaeger M. Numerical simulations of incompressible flows with moving interfaces. *International Journal for Numerical Methods in Fluids* 1997; **24**:615–638.
11. Thompson E. Use of pseudo-concentrations to follow creeping viscous flows during transient analysis. *International Journal for Numerical Methods in Fluids* 1986; **6**:749–761.
12. Unverdi SO, Tryggvason G. Computations of multi-fluid flows. *Physica D* 1992; **60**:70–83.

13. Hirt CW, Nichols BD. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics* 1981; **39**:201–225.
14. Mashayek F, Ashgriz N. A hybrid finite-element-volume-of-fluid method for simulating free surface flows and interfaces. *International Journal for Numerical Methods in Fluids* 1995; **20**:1367–1380.
15. Maronnier V, Picasso M, Rappaz J. Numerical simulation of free surface flows. *Journal of Computational Physics* 1999; **155**(2):439–455.
16. Pironneau O. *Finite Element Methods for Fluids*. Wiley: Chichester, 1989.
17. Pironneau O, Liou J, Tezduyar T. Characteristic-Galerkin and Galerkin/least-squares space–time formulations for the advection–diffusion equation with time-dependent domains. *Computer Methods in Applied Mechanics and Engineering* 1992; **100**:117–141.
18. Quarteroni A, Valli A. *Numerical Approximation of Partial Differential Equations*. Number 23 in Springer Series in Computational Mathematics, Springer: Berlin, 1991.
19. Wang H, Dahle HK, Ewing RE, Espedal MS, Sharpley RC, Man S. An ELLAM scheme for advection-diffusion equations in two dimensions. *SIAM Journal on Scientific Computing* 1999; **20**(6):2160–2194.
20. Yabe T, Ishikawa T, Wang PY, Aoki I, Kadota Y, Ikeda F. A universal solver for hyperbolic equations by cubic-polynomial interpolation. II. Two- and three-dimensional solvers. *Computer Physics Communications* 1991; **66**(2–3):233–242.
21. Chorin AJ. Flame advection and propagation algorithms. *Journal of Computational Physics* 1980; **35**:1–11.
22. Noh WF, Woodward P. *SLIC* (*Simple Line Interface Calculation*), Lectures Notes in Physics, vol. 59. Springer: Berlin, 1976; 330–340.
23. Maronnier V. Simulation numérique d'écoulements de fluides incompressibles avec surface libre. *Ph.D. Thesis*, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, 2000.
24. Franca LP, Frey SL. Stabilized finite element methods: II. the incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1992; **99**:209–233.
25. Franca LP, Hughes TJR. Convergence analyses of Galerkin least squares methods for symmetric advective–diffusive forms of the Stokes and incompressible Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 1993; **105**:285–298.
26. Tezduyar TE, Mittal S, Ray SE, Shih R. Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering* 1992; **95**:221–242.
27. Fortin M. Finite element solution of the Navier–Stokes equations. In *Acta Numerica*. Cambridge Univ. Press: Cambridge, 1993; 239–284.
28. Gresho PM, Sani RL. *Incompressible Flow and the Finite Element Method*. Wiley: New York, 2000.
29. Picasso M, Rappaz J. Stability of time-splitting schemes for the Stokes problem with stabilized finite elements. *Numerical Methods for Partial Differential Equations* 2001; **17**(6):632–656.
30. Perot JB. An analysis of the fractional step method. *Journal of Computational Physics* 1993; **108**(1):51–58.
31. Quarteroni A, Saleri F, Veneziani A. Factorization methods for the numerical approximation of Navier–Stokes equations. *Computer Methods in Applied Mechanics and Engineering* 2000; **188**(1–3):505–526.
32. Rappaz M, Desbiolles JL, Gandin CA, Henry S, Semoroz A, Thevoz P. Modelling of solidification microstructures. *Material Science Forum* 2000; **329**(3):389–396.
33. Schmid M, Klein F. Einflußder wandreibung auf das füllverhalten dünner platten. *Preprint*, *Steinbeis Transferzentrum*, *Fachhochschule Aachen*, 1996. See also http://www.fh-aalen.de/arge/HTML/ENGLISH/Markus-E.html.
34. Martin JC, Moyce WJ. An experimental study of the collapse of liquid columns on a rigid horizontal plate. *Philosophical Transactions of Royal Society of London*, *Series A* 1952; **244**:312–324.

