INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **54**:965–993
Published online 21 March 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1475

Stabilized edge-based finite element simulation of free-surface flows

Renato N. Elias and Alvaro L. G. A. Coutinho*, †

Center for Parallel Computations and Department of Civil Engineering, Federal University of Rio de Janeiro, P.O. Box 68506, RJ 21945-970-Rio de Janeiro, Brazil

SUMMARY

Free-surface flows occur in several problems in hydrodynamics, such as fuel or water sloshing in tanks, waves breaking in ships, offshore platforms, harbours and coastal areas. The computation of such highly nonlinear flows is challenging since free-surfaces commonly present merging, fragmentation and breaking parts, leading to the use of interface-capturing Eulerian approaches. In such methods the surface between two fluids is captured by the use of a marking function which is transported in a flow field. In this work we present a three-dimensional parallel edge-based incompressible SUPG/PSPG finite element method to cope with free-surface problems with volume-of-fluid (VOF) extensions to track the evolving free surface. The pure advection equation for the scalar marking function was solved by a fully implicit parallel edge-based SUPG finite element formulation. We studied variants of this formulation, considering the effects of discontinuity capturing and a particular tangent transformation designed to increase interface sharpness. Global mass conservation is enforced adding or removing mass proportionally to the absolute value of the normal velocity of the interface. We introduce a parallel dynamic deactivation algorithm to solve the marking function equation only in a small region around the interface. The implementation is targeted to distributed memory systems with cache-based processors. The performance and accuracy of the proposed solution method were tested with several validation problems. Copyright © 2007 John Wiley & Sons, Ltd.

Received 20 December 2006; Revised 6 February 2007; Accepted 7 February 2007

KEY WORDS: free surface; interface capturing; stabilized finite elements; edge-based formulation; volume-of-fluid

*Correspondence to: Alvaro L. G. A. Coutinho, Center for Parallel Computations and Department of Civil Engineering, Federal University of Rio de Janeiro, P.O. Box 68506, RJ 21945-970-Rio de Janeiro, Brazil.
†E-mail: alvaro@nacad.ufrj.br

Contract/grant sponsor: Petroleum National Agency (ANP, Brazil)
Contract/grant sponsor: MCT/CNPq
Contract/grant sponsor: The Brazilian Council for Scientific Research

1. INTRODUCTION

Free-surface flows occur in many hydrodynamics problems. Sloshing of liquids in tanks, wave breaking in ships, offshore platforms, harbours, coastal areas, green water on decks are important examples of this class of problems [1]. The main computational challenge when solving such highly nonlinear problem is determining the evolution of the interface location. There is a large number of numerical methods devoted to the computation of free-surface problems. These methods are frequently classified as interface tracking and interface-capturing methods (see [2]).

Interface tracking methods are based on a Lagrangian framework where the moving interface or boundary is explicitly tracked by the computational grid or by the particles of meshless methods which must be deformed or moved in order to follow the fluid flow. The deforming-spatial-domain/stabilized space–time finite element formulation (DSD/SST) proposed by Tezduyar [3, 4] and Tezduyar *et al*. [5, 6] is a mesh-based example of interface tracking method. Although accurate, the interface tracking methods are computationally expensive. The mesh-based Lagrangian formulations, such as the DSD/SST, suffer when the moving interface turns the mesh highly distorted obligating the frequent use of a mesh rebuilding procedure. Furthermore, most of these methods fail when the interface presents folds, fluid fragmentation, bubbles and cusps since the mesh tracking of such phenomena is practically unfeasible. As a remedy for this problem, the meshless Lagrangian methods have been applied to the solution of free-surface flows. Koshizuka *et al*. [7] and Violeau and Issa [8] are examples of the application of particle-based formulations such as the smoothed particle hydrodynamics (SPH) methods to the simulation of free-surface problems. However, meshless methods still present a high computational cost since they need to compute the interaction between the particles using search algorithms. Moreover, these methods are hard to parallelize, and the post-processing usually employs specific algorithms to represent the interface. As a compromise between the advantages offered by mesh based and meshless methods, Del Pin *et al*. present in [9] the particle finite element method (PFEM) applied to free-surface flows. In this method the critical parts of the continuum are discretized with particles while the remaining parts are treated by a Lagrangian finite element formulation. Another technique mixing Lagrangian and Eulerian flavours was proposed in [10] by Takizawa *et al*. In this work the authors enhanced the constrained interpolation profile (CIP) method for solving hyperbolic equations with a meshless Soroban grid. The resulting formulation was used to treat fluid–object and fluid–structure interaction in the presence of free surfaces.

As a cost-effective alternative to interface tracking methods, interface-capturing methods have emerged. Interface-capturing methods are Eulerian in their concept, thus they rely on a unique and fixed computational grid to capture the interface evolution. In this class of methods the interface is represented by a scalar function which marks the regions filled with the fluids involved. In other words, the interface position is implicitly captured in a scalar marking function value, and the interface evolution is determined by the additional cost of solving an advection equation for the marker. As opposed to interface tracking methods, interface-capturing methods require little effort to represent all complicated features of moving interfaces. Additionally, the parallel implementation and post-processing of interface-capturing methods are straightforward. The main drawback of interface-capturing methods is the need to average the fluid properties at the interface cells (elements) due to the discontinuity of the Eulerian representation of the interface. Moreover, the accuracy and computational cost of interface-capturing methods are typically associated with grid resolution, properties of the marking function chosen to represent the interface and numerical methods for solving the fluid flow and marking function advection. The well-known

DOI: 10.1002/fld

volume-of-fluid (VOF) scheme, firstly proposed by Hirt and Nichols [11] for Cartesian grids, is an interface-capturing technique which employs a step function ranging from 0 to 1 to represent the fraction of fluid within the grid cells. In this sense, the interface is implicitly represented by the partially filled cells. The main issues associated with VOF methods include the difficulty in advecting a discontinuous step function and the accurate modelling of surface tension effects. Level set methods [12] implement free-surface flows in a different manner than VOF by changing the marking function employed to represent the interface. In level set methods, the original problem dimension is augmented by the use of a signed distance function to represent the interface. Therefore, the fluids are associated with the range of the distance function signs while the interface is implicitly represented by the zero level set. In this way, the level set method overcomes the problem of advecting a discontinuous function posed by VOF while surface tension effects can be readily modelled using the distance function properties. However, the level set method suffers when the distance function loses its properties and must be rebuilt. In fact, the success of level set method lies in its ability of building and keeping a signed distance function without losing its properties. Most of the original VOF and level set methods are devoted to Cartesian grid formulations. The enhanced-discretization interface-capturing method, firstly proposed by Tezduyar [13], and the work of Lohner *et al*. presented in [14], both are examples of unstructured grid formulations based on the finite element method to solve free-surface flows using a VOF marking function. In [15] Sunitha *et al*. proposed a stabilized finite element formulation to treat bubble dynamics problems using a level set marking function to represent the fluid phases. In [16] Shepel and Smith proposed a less diffusive approach to the original interface-capturing method implemented in a finite volume commercial code [17]. In this work the authors used a level set-based marking function transported by a streamline-upwind/Petrov–Galerkin (SUPG) finite element formulation [18], while the fluid flow is solved by finite volumes.

In this work we extend our edge-based stabilized finite element solver to deal with free-surface problems. The main characteristics of our incompressible flow solver [19–21] are: SUPG and pressure-stabilizing/Petrov–Galerkin (PSPG) [3, 22] stabilized finite element formulation; implicit time marching scheme with adaptive time stepping control; advanced inexact Newton solvers; edge-based data structures to save memory and improve performance; support to message passing and shared memory parallel programming models; and large eddy simulation (LES) extensions using a classical Smagorinsky model. We introduce VOF extensions in this flow solver to track the evolving free surface [11, 13, 14]. The advection equation for the free-surface marking function is solved by a fully implicit parallel edge-based SUPG finite element formulation. We investigate the effectiveness of incorporating discontinuity capturing for solving the steep gradients of the VOF marking function and a tangent transformation standard in CIP methods [23]. A global mass conservation algorithm is introduced to enforce that the mass of the species involved are correctly represented as the solution evolves. The computational effort to solve the marking function is limited to a narrow band around the free surface by a parallel dynamic-deactivation (PDD) scheme, extending the features of the original scheme introduced by Lohner and Camelli [24].

The remainder of this paper is organized as follows: the first and second sections present the incompressible flow and interface-capturing governing equations, respectively; the third section summarizes the solution procedures employed and the fourth shows results obtained for standard global mass conservation test cases: the Zalesak's disk [25], the disk stretching [26] and the three-dimensional deformation field [27] where we compare the stabilized formulation with discontinuous Galerkin (DG) [28], ENO, WENO and particle level sets methods [29, 30]. Next, we simulate the dam-break problem and the wave impact on a container [1, 31], where a water column is

DOI: 10.1002/fld

suddenly released, hitting an obstacle. We compare our results with the available experimental and numerical results obtained with different methods (finite volumes, finite elements, smooth particle hydrodynamics) showing that the present scheme is fast, simple and accurate. The final remarks and conclusions are summarized in the last section.

2. GOVERNING EQUATIONS

2.1. Incompressible flow

Let $\Omega \subset \mathbb{R}^{n_{\text{sd}}}$ be the spatial domain, where n_{sd} is the number of space dimensions. Let Γ denote the boundary of Ω. We consider the following velocity–pressure formulation of the Navier–Stokes equations governing the incompressible flow of two immiscible fluids:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} - \mathbf{f}\right) - \nabla\cdot\boldsymbol{\sigma} = \mathbf{0} \quad \text{on } \Omega\times[0,t_f] \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{on } \Omega\times[0,t_f] \tag{2}$$

where ρ and $\mathbf{u}$ are the density and velocity, $\mathbf{f}$ is the body force vector carrying the gravity acceleration effect and $\boldsymbol{\sigma}$ is the stress tensor given as

$$\boldsymbol{\sigma}(p,\mathbf{u}) = -p\mathbf{I} + \mathbf{T} \tag{3}$$

where p is the hydrostatic pressure, $\mathbf{I}$ is the identity tensor and $\mathbf{T}$ is the deviatoric stress tensor

$$\mathbf{T} = 2\mu\boldsymbol{\varepsilon}(\mathbf{u}) \tag{4}$$

and $\boldsymbol{\varepsilon}(\mathbf{u})$ is the strain rate tensor defined as

$$\boldsymbol{\varepsilon}(\mathbf{u}) = \tfrac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^{\mathrm{T}}) \tag{5}$$

In the present work a LES approach to turbulence is considered by the use of a classic Smagorinsky turbulence model [32]. In this model, the viscosity μ is augmented by a subgrid-scale viscosity μ_{SGS} proportional to a norm of the local strain rate tensor and to a filter width h defined here as the cubic root of the element volume

$$\mu_{\text{SGS}} = \rho(C_{\text{S}}h)^2(2\boldsymbol{\varepsilon}(\mathbf{u}):\boldsymbol{\varepsilon}(\mathbf{u}))^{1/2} \tag{6}$$

where C_{S} is the Smagorinsky constant, ranging from 0.1 to 0.2.

The essential and natural boundary conditions associated with Equations (1) and (2) can be imposed at different portions of the boundary Γ and represented by

$$\mathbf{u} = \mathbf{g} \quad \text{on } \Gamma_g \tag{7}$$

$$\mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h} \quad \text{on } \Gamma_h \tag{8}$$

where Γ_g and Γ_h are complementary subsets of Γ.

Let us assume that we have some suitably defined finite-dimensional trial solution and test function spaces for velocity and pressure, $S^h_{\mathbf{u}}$, $V^h_{\mathbf{u}}$, S^h_p and $V^h_p = S^h_p$. The finite element formulation of Equations (1) and (2) using SUPG and PSPG stabilizations for incompressible fluid flows can be

DOI: 10.1002/fld

written (see Tezduyar and Osawa [33]) as follows: find $\mathbf{u}^h \in S_{\mathbf{u}}^h$ and $p^h \in S_p^h$ such that $\forall \mathbf{w}^h \in V_{\mathbf{u}}^h$ and $\forall q^h \in V_p^h$

$$\int_\Omega \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h - \mathbf{f} \right) \mathrm{d}\Omega + \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}(p^h, \mathbf{u}^h) \, \mathrm{d}\Omega - \int_\Gamma \mathbf{w}^h \cdot \mathbf{h} \, \mathrm{d}\Gamma + \int_\Omega q^h \nabla \cdot \mathbf{u}^h \, \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \frac{1}{\rho} [\tau_{\mathrm{SUPG}} \rho \mathbf{u}^h \cdot \nabla \mathbf{w}^h + \tau_{\mathrm{PSPG}} \nabla q^h] \cdot \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \mathbf{u}^h \right) - \nabla \cdot \boldsymbol{\sigma}(p^h, \mathbf{u}^h) - \rho \mathbf{f} \right] \mathrm{d}\Omega^e$$
$$+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_{\mathrm{LSIC}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h \Omega^e = 0 \tag{9}$$

In the above equation the first four integrals on the left-hand side represent terms that appear in the Galerkin formulation of problem (1)–(8), while the remaining integral expressions represent the additional terms which arise in the stabilized finite element formulation. Note that the stabilization terms are evaluated as the sum of element-wise integral expressions, where n_{el} is the number of elements in the mesh. The first summation corresponds to the SUPG term and the second to the PSPG term. We have evaluated the SUPG and PSPG stabilization parameters according to Tezduyar *et al.* [22], as follows:

$$\tau_{\mathrm{SUPG}} = \tau_{\mathrm{PSPG}} = \left[\left(\frac{2\|\mathbf{u}^h\|}{h} \right)^2 + 9 \left(\frac{4\nu}{h^2} \right)^2 \right]^{-1/2} \tag{10}$$

here $\mathbf{u}^h$ is the local velocity vector, and ν is the kinematic viscosity.

In Equation (9), the last summation is the least-squares incompressibility constraint (LSIC) term [34], added to prevent oscillations in high Reynolds number flows. The LSIC stabilization parameter is

$$\tau_{\mathrm{LSIC}} = \frac{\|\mathbf{u}^h\| h}{2} \tag{11}$$

The discretization of Equation (9) leads us to a nonlinear system of equations to be solved at each time step.

2.2. *Interface capturing*

The VOF method was firstly proposed by Hirt and Nichols [11] for finite difference schemes. The idea of the method is to define a scalar marking function $\phi(\mathbf{x}, t)$ over the computational domain in such a manner that its value at a certain point $\mathbf{x} \in \Omega$ and instant $t \in [0, t_f]$ indicates the fraction of the fluids involved. Thus, a scalar marking function can be employed to capture the position of the interface between the fluids by simply using the fluids fraction relationship.

The VOF can be stated as: assuming the value 1 in regions filled with fluid A, e.g. water, and the value 0 in regions filled with fluid B, e.g. air, the position of the fluid interface will be defined by the isovalue contour $\phi(\mathbf{x}, t) = \phi_{\mathrm{c}}$, where $\phi_{\mathrm{c}} \in [0, 1]$. The value $\phi_{\mathrm{c}} = 0.5$ is usually assumed. Finally, the function $\phi(\mathbf{x})$ is driven by a velocity field $\mathbf{u}$ satisfying the following transport equation,

DOI: 10.1002/fld

given in conservative form as:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = 0 \tag{12}$$

In the VOF formulation the fluid density and viscosity, employed in the fluid flow solution, are interpolated across the interface as follows:

$$\rho = \phi(\mathbf{x}, t)\rho_B + [1 - \phi(\mathbf{x}, t)]\rho_A \tag{13}$$

$$\mu = \phi(\mathbf{x}, t)\mu_B + [1 - \phi(\mathbf{x}, t)]\mu_A \tag{14}$$

where subscripts A and B denote the values corresponding to each fluid.

The finite element formulation of Equation (12) can be written as follows: find $\phi^h \in S^h_\phi$, such that, $\forall w^h \in V^h_\phi$

$$\begin{aligned}
&\int_\Omega w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h \right) \mathrm{d}\Omega + \int_\Omega w^h (\nabla \cdot \mathbf{u}^h)\phi^h \, \mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_{\mathrm{SUPG}} \mathbf{u}^h \cdot \nabla w^h \left(\frac{\partial \phi^h}{\partial t} + \mathbf{u}^h \cdot \nabla \phi^h + (\nabla \cdot \mathbf{u}^h)\phi^h \right) \mathrm{d}\Omega^e \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \delta \nabla w^h \cdot \nabla \phi^h \, \mathrm{d}\Omega = 0
\end{aligned} \tag{15}$$

where S^h_ϕ and V^h_ϕ are standard test and weight finite element spaces. The first two integrals represent the Galerkin formulation of Equation (12), while the first element-wise summation represents the SUPG and the second summation term is the nonlinear discontinuity-capturing term, useful when sharp gradients and/or boundary layers are present in directions other than the streamlines [35]. The evaluation of τ_{SUPG} and δ stabilization terms follows the definitions described in [18, 35], respectively. The discretization of Equation (15) leads us to a non-linear, due to the discontinuity-capturing term, ordinary differential equation system. In the particular case of the consistent approximated upwind (CAU) method for the discontinuity-capturing term [35] we have

$$\delta = h \frac{|R(\phi^h)|}{\|\nabla \phi^h\|} \quad \text{if } \|\nabla \phi^h\| \neq 0 \tag{16}$$

where $R(\phi^h)$ is the residual of Equation (12)

$$R(\phi^h) = \frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) \tag{17}$$

In the following sections the above formulation will be evaluated for the advection of the VOF step function. In order to avoid non-physical results, values lying outside the range [0, 1] were truncated with the following function:

$$\phi^h = \min[\max[\phi^h, 0], 1] \tag{18}$$

A very simple and promising method to suppress interface diffusion and undulation was proposed in [23] for the cubic-interpolated propagation (CIP) method. The idea is to substitute the original

DOI: 10.1002/fld

step function ϕ by F_ϕ, where F_ϕ is a function of ϕ. In [23] a function F_ϕ based on a tangent transformation, is given

$$F_\phi = \tan[(1-\varepsilon)\pi(\phi - 0.5)] \tag{19}$$

and the marking function recovered by the inverse transformation, that is

$$\phi = \frac{\arctan F_\phi}{(1-\varepsilon)\pi} + 0.5 \tag{20}$$

where ε is a small number to avoid infinite values when ϕ is equal to 0 or 1. Neglecting the term $(1-\varepsilon)$, F_ϕ would lie in the range from $-\infty$ to $+\infty$ for $\phi = 0$–1. The merit of this transformation is due to the tangent function properties which shows a regular behaviour in regions where the function ϕ experiences a rapid change [23]. Furthermore, the diffusion and undulation which occur when advecting the function F_ϕ instead of the original one, will be readily limited in a range from 0 to 1 due to the tangent function. Another property of this transformation is that most of the values will be kept concentrated near to $\phi = 0$–1, consequently increasing interface sharpness (for more details see [23]). In the following sections we will evaluate the efficiency of this transformation when used in a finite element context. For the purpose of our evaluations we adopted the SUPG formulation to advect function (19) with $\varepsilon = 0.01$. Other methods to improve interface calculations in terms of interface sharpness and mass conservation may be seen in [3, 36].

3. SOLUTION PROCEDURE

The computational solution kernels consist of predictor/multicorrector time integration schemes as described in [18, 37] for both incompressible fluid flow and interface transport equations. The generalized trapezoidal rule is employed in the time discretization. The nonlinearities due to the convective term on the Navier–Stokes equation are treated by an Inexact Newton-GMRES algorithm as described in Elias *et al*. [38]. In this solution algorithm, at the beginning of the nonlinear iterations in each time step, the algorithm computes large linear tolerances, producing fast nonlinear steps. As the iterations progress towards the solution, the inexact nonlinear method adapts the GMRES tolerances to reach the desired accuracy. A nodal-block diagonal preconditioner is employed for the flow and a simple diagonal preconditioner for the marker. Moreover, for both the fluid flow and the marking function advection we use an adaptive time stepping procedure based on a proportional-integral-derivative (PID) controller (see [39] for further details). Most of the computational effort spent in this solution procedure is due to the matrix–vector products within the GMRES driver for both flow and marker. To improve the computational efficiency with respect to standard element-by-element and sparse matrix–vector storage schemes, we adopt an edge-based data structure in order to minimize indirect memory addressing, diminish floating point operation counts (flops) and memory requirements, as described in Elias *et al*. [20] and Coutinho *et al*. [19] for both the Navier–Stokes equations and the marking function advection. Further computational gains are obtained from data preprocessing performed by the EdgePack library—a package to improve cache reutilization based on reordering and grouping techniques [40]. All computational kernels are parallelized covering most of distributed and shared memory systems as well as vectorized and pipelined processors [21].

DOI: 10.1002/fld

3.1. Enforcing mass conservation

The most challenging feature for a good interface-capturing method resides in its ability to preserve the mass of the species involved. In VOF methods the mass loss can be associated with reasons such as: interface smearing due to numerical diffusion of the step function; inexact divergence free velocity field and boundary conditions; undulations in the solution of the marking function advection. Level set methods suffer when the marking function lose its signed distance property. These problems have been reported by many authors and are still the subject of several researches. A global mass conservation algorithm has been introduced in [41] for a moving Lagrangian interface and tested later in mold filling applications [42]. In this work we have followed the procedure proposed by Lohner *et al.* [14] to overcome mass losses. In this procedure the mass lost/gained are found by comparing the expected value, composed by the initial mass plus the inlet and outlet fluxes, at the end of each time step. Therefore, the values to be added or removed are made proportional to the absolute value of the normal velocity of the interface given by

$$u_n = \left| \mathbf{u} \cdot \frac{\nabla \phi}{\|\nabla \phi\|} \right| \tag{21}$$

The amount computed from Equation (21) guarantees that the mass correction will act mainly in regions where the interface is moving faster while keeping the stationary regions untouched. Therefore, the portion of mass correction corresponding to each element is computed and projected onto the global nodes by L_2 projection.

In order to guarantee the effectiveness of the mass correction method we must have an accurate procedure for computing the volume of the fluid phases. In this work we have adopted a method which makes use of the interface explicitly determined for those elements lying on the interface. To introduce the method, firstly we need to identify the elements as filled, partially filled or empty. This can be done by computing an element key as illustrated in Figure 1.

While computing the element key the nodes are marked with value 1 if $\phi \geqslant 0.5$, otherwise, it is marked as 0, thus the key is found by summation of the node marks. Note that the empty elements (key 0) can be skipped from the computation since the void region will be the difference between the domain and the filled region. Moreover, for elements partially filled (keys 1, 2 and 3) the filling volume can be found from the sub-tetrahedra and wedges formed by the interface plane cutting the original element.

3.2. Parallel dynamic deactivation

We introduce here another computational artefact to further improve the overall efficiency of the present free-surface solver, the PDD technique for solving the marking function. This technique is

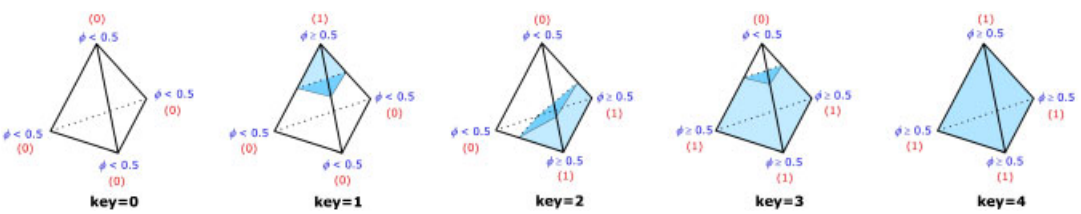

Figure 1. Scheme for the computation of the element keys according to the filling level.

DOI: 10.1002/fld

an extension of the DD procedure, which is an algorithm that restricts the computation on regions where a defined gradient is found. It was firstly presented by Lohner and Camelli in [24] for contaminant transport problems. In this paper we have extended the original DD scheme for the parallel computation of free-surface flows. Since the marking function employed on VOF methods presents steep gradients, the DD algorithm catches and restricts the computations only on regions around the interface. Consequently, most of the computational effort that would be necessary to solve the interface transport over the whole domain is considerably saved. Moreover, a buffer zone around the interface is built to assure that the interface is kept within the enabled region in each time step. It is important to emphasize that although the computational costs associated with the transport problem are recognizably lesser than those spent by the Navier–Stokes solution, a similar approach can be employed to restrict the computations only on regions filled by the aimed fluid during the incompressible flow solution phase. Our DD implementation follows the edge-based adaptive implicit/explicit (AIE) algorithm described by Souza *et al*. in [43] and originally introduced in [44]. AIE algorithms select groups of edges (elements) where implicit and explicit time integration algorithms will be applied according to stability and accuracy criteria. In the present context the set of active elements initially selected by the DD algorithm is based upon the following criteria:

$$\|\nabla\phi\|^e \geqslant \eta \|\nabla\phi\|_m \tag{22}$$

where $\|\nabla\phi\|^e$ is the Euclidean norm of the element gradient solution, $\|\nabla\phi\|_m$ is the average gradient norm computed for the whole computational grid, and $\eta \in [0, 1]$ is a parameter which controls how the element selection must work. Alternatively, we could select the interface element following the procedure described in Section 3.1 for the global mass conservation algorithm instead of using Equation (22). Note though that, for interface-capturing methods based on level sets, since the marking function has a constant gradient, the selection criteria based on the average gradient must be changed such that it selects the elements lying within a defined distance from the interface. This procedure is named as *narrow band* in the level set terminology [45]. After the initial selection, a buffer zone is built by selecting a number of layers, n_{layers}, with the corresponding active nodes, elements and edges. Note that this parameter can be set according to the Courant–Friedrichs–Lewy (CFL) condition in order to assure that interface does not cross the buffer zone. The DD implementation relies on the simple code modification sketched in Algorithm 1.

ORIGINAL LOOP:	MODIFIED LOOP:

`do ie = 1,ntent`
`  ! ...computations...`
`enddo`
	`Selects active entities using Eq. 22`
	
	`Insert active entities in the ActiveEntities list`
	`do ie = 1,nae`
	`  ia = ActiveEntities(ie)`
	`  ! ...computations...`
	`enddo`

Algorithm 1. Modifications required for the dynamic deactivation implementation

In Algorithm 1, **`ntent`** is the number of entities for the whole domain, **`ActiveEntities`** is an integer list which holds the index of those entities selected as active, and **nae** is the number of active entities (nodes, edges and elements). For the marking function advection, **nae** is the number of these entities surrounding the interface. Note that no further modifications are required. In the PDD extension, each processor estimates its own number of finite element

DOI: 10.1002/fld

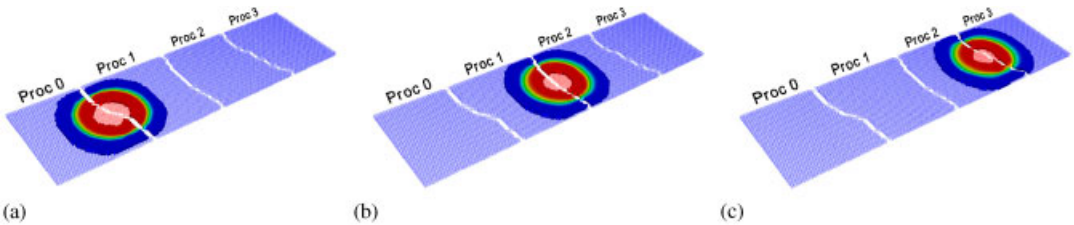

Figure 2. Parallel dynamic deactivation in the advection of a circle.

entities enabled for computation, and the loops are constrained in the same way as described in Algorithm 1. The selection of active entities is made after solving the flow equations. The PDD is illustrated in Figure 2, where the evolution of the active elements during the advection of a circle is shown by the solid coloured regions. If there are no enabled entities in some processor, it becomes idle as shown for processors 2–3, 0–3 and 0–1 in Figure 2(a), (b) and (c) respectively.

It is worthwhile to mention that even producing unbalanced partitions, the PDD will produce computational gains since it can drastically reduce the overall effort (computation and communication) to solve the problem. Moreover, the PDD does not spend any further effort in repartitioning, renumbering and redistributing the entities over the processors. In fact, the PDD partitions are produced and controlled by the use of lists of active entities which restrict the main loops of the solver.

4. MASS CONSERVATION TESTS

This section presents the solution of a set of problems designed to evaluate the ability of the proposed numerical scheme to advect the VOF step function while keeping the desired characteristics for a free-surface solver: interface sharpness and global mass conservation. The problems presented are challenging since they impose different scenarios where the interface must keep its original shape after being submitted to severe velocity fields. The tests comprise only the solution of the transport equation when submitted to an analytical velocity field. The linear tolerance of 10^{-3} is adopted for the GMRES solver in all cases, and the multi-correction step is stopped after the relative residual decreases three orders of magnitude or the maximum number of five multi-corrections is reached. Furthermore, we use unstructured three-dimensional meshes of tetrahedra to run all cases, even for two-dimensional problems, where the model is represented only by a thin three-dimensional section. The results are compared with those listed in [28] for the DG method, in [29] which shows improvements to the WENO (weighted essentially non-oscillatory) scheme with a particle level set formulation, and [45] which is based on the classical structured grid ENO scheme. The computations were carried out in a Dell Precision 370 machine (Intel Pentium 4 EM64T 3.6 GHz, 4 GB of memory, Intel Fortran Compiler 9.1 and Windows XP professional x64).

The accuracy of the solution is accessed in terms of area or volume loss of the filled region with respect to the initial data. Thus, being the volume (area) of the region completely filled with fluid, computed as described in Section 3.1, the relative amount of fluid lost/gained at a given

DOI: 10.1002/fld

time step can be found by

$$\tilde{V}(\%) = 100 \times \frac{V_1^{t_n} - V_1^{t_0}}{V_1^{t_0}} \tag{23}$$

where t_0 and t_n represent the initial and current time steps, respectively, and $\tilde{V}$ is the relative volume (area) fluctuation.

4.1. Zalesak's rotating disk

The Zalesak's rotating disk [25] has been widely used to evaluate the accuracy of advection solvers [25, 28, 29]. The problem consists of a slotted disk centred at (50,75) with a radius of 15. The slot has a width of 5, and length of 25. The constant rotational velocity field is given by

$$\begin{aligned} u_x &= (\pi/314)(50 - y) \\ u_y &= (\pi/314)(x - 50) \\ u_z &= 0 \end{aligned} \tag{24}$$

and the disk completes one revolution at each 628 time units. Since this work aims three-dimensional problems, we employ only a two-dimensional narrow slab with dimensions $100 \times 100 \times 0.5$. For this problem we have employed a mesh with element length of 0.32 in average which corresponded to 283 920 elements, 469 086 edges and 93 467 nodes (Figure 3). The time step was set to 1, corresponding to an average CFL number of 1.18. The DD parameter η is set to 0.90 while the number of additional layers is set to 5.

Table I summarizes the results obtained in this work, considering as the basic scheme for marking the SUPG stabilization with the truncation function given by Equation (18), added or not with the discontinuity-capturing (CAU) term and advecting only the tangent transformed function (T). In all cases, the results include formulations with and without the global mass conservation algorithm, which is denoted as MC. The obtained area losses or gains were compared to those published in [28–30, 45].

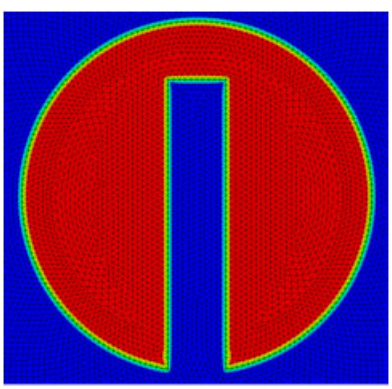

Figure 3. Initial Zalesak's disk for the mesh employed—average element length 0.32.

DOI: 10.1002/fld

Table I. Zalesak's rotating disk—area loss or gain after one revolution.

Method	h	Area	Area loss/gain	CPU time (s)
Exact	—	581.69	—	—
(a) BASIC	0.32	579.65	-0.35	260.33[¶]
(b) BASIC + MC	0.32	581.69	-3.3×10^{-5}	302.83[¶]
(c) BASIC + CAU	0.32	574.53	-1.23	648.28[¶]
(d) BASIC + CAU + MC	0.32	581.57	-2.0×10^{-2}	689.70[¶]
(e) BASIC + TF	0.32	568.03	-2.35	309.58[¶]
DG TRI(3)*	4.0	580.40	$+0.30$	49.27[‖]
DG TRI(4)*	4.0	583.27	-0.18	101.22[‖]
DG TRI(5)*	4.0	581.01	$+0.20$	211.51[‖]
DG TRI(3)*	2.0	581.22	$+0.16$	466.06[‖]
DG TRI(4)*	2.0	582.17	-0.0050	955.5[‖]
DG TRI(5)*	2.0	582.21	-0.0017	2040.03[‖]
WENO(5)[†]	1.0	613.0	-5.3	—
WENO(5)[‡]	1.0	580.4	$+0.31$	134.043
WENO(5)[‡]	0.5	581.7	$+0.08$	840.038
ENO(3)[§]	1.0	—	—	—

*Discontinuous Galerkin [28].
[†]Level set [29, 30].
[‡]Particle level set [29, 30].
[§]Level set [45].
[¶]Intel Pentium-4 3.6 GHz.
[‖]Intel Pentium-4 2.4 GHz.

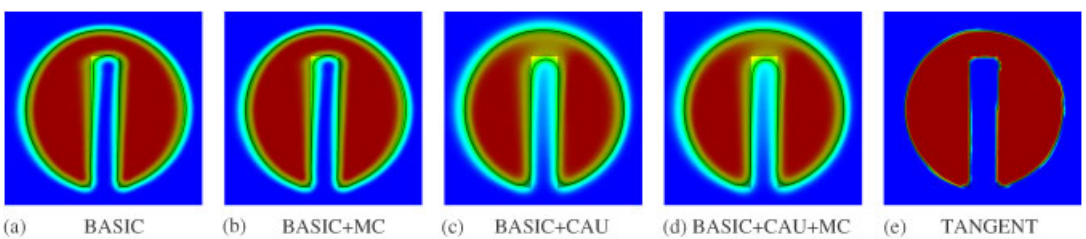

Figure 4. Zalesak's rotating disk problem after one revolution.

Table I shows that, among the schemes proposed in this work, the basic scheme was the fastest, however, the most accurate solutions were those obtained with the mass correction algorithm. The BASIC + MC scheme, when compared against DG TRI(4) [28], produced almost no losses, being three times faster. In opposition, the solution schemes employing the discontinuity-capturing term produced less accurate results and spent more time since the discontinuity-capturing term turns the advection equation nonlinear. However, it is important to observe that the global mass conservation algorithm was effective to remedy the lack of accuracy presented by this formulation.

For each case listed in Table I, Figure 4 shows the marking function contour plot after one revolution. In this figure the shaded area corresponds to the original shape of the slotted disk while the region bounded by black lines is the disk appearance after one revolution (628 time steps).

DOI: 10.1002/fld

We may see in Figure 4 that the discontinuity-capturing formulation was the most diffusive while the tangent transformed function formulation practically does not present any diffusion. Moreover, for this formulation, the interface is kept restricted within the element scale, leading the interface to a jigsaw appearance. It is also important to note that although the global mass conservation algorithm is efficient to preserve area, it has almost no effect in preserving the slotted disk shape.

4.2. *Disk stretching*

The disk stretching problem, also named as *single vortex* and *vortex-in-a-box* problem, was firstly introduced by Bell *et al.* [26] as a more severe test case than the Zalesak's rotation disk for testing the ability of the numerical scheme to resolve thin filaments. These thin filaments occur when a disk is submitted to the analytical stretching flow field

$$\psi = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y) \tag{25}$$

which gives rise to the following velocity field:

$$\begin{aligned} u_x &= \frac{\partial \psi}{\partial y} = \sin(2\pi y)\sin^2(\pi x) \\ u_y &= -\frac{\partial \psi}{\partial x} = -\sin(2\pi x)\sin^2(\pi y) \end{aligned} \tag{26}$$

The computational grid was built with 91 105 nodes, 453 826 edges and 271 900 elements in a thin domain of size $[0, 1] \times [0, 1] \times [0, 0.005]$ where a disk of radius 0.15 is initially placed at $(0.5, 0.75)$. Therefore, the velocity field starts to distort and stretch out the disk up to a very thin curl is rendered. For the purpose of error analysis, LeVeque [27] suggested to multiply the velocity field by

$$g(t) = \cos(\pi t/T) \tag{27}$$

where T is the period at which the flow returns to its initial state, consequently the disk should recover its original shape. Following the authors [28, 29] we have adopted a reversal period of $T = 8$ in the error analysis. Figure 5 shows the sequence corresponding to the time units 0, 2, 4, 6, 8, and the periods $0.0T$, $T/4$, $T/2$, $3T/4$, T, respectively. For this case, a fixed time step of 0.005 was chosen which corresponded to an average CFL number of 0.78.

In Figure 5 the marking function was extruded in order to make the interface sharpness more evident. The shadowed cylinder refers to the original shape. Note that the maximum stretching is

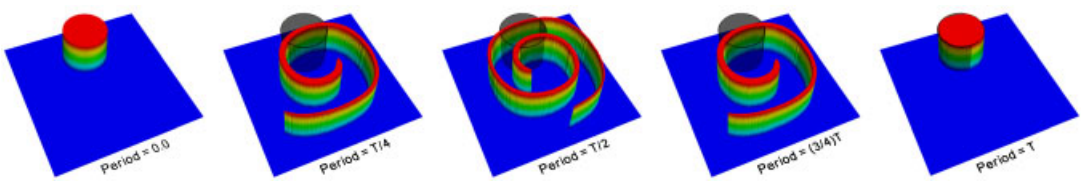

Figure 5. Disk stretching problem—time evolution for the scheme employing the tangent function.

DOI: 10.1002/fld

Table II. Disk stretching—area loss or gain.

Method	h	Area	% Area loss	CPU time (s)
Exact	—	0.07063	—	—
(a) BASIC	3.3×10^{-3}	0.07094	+0.449	1028.12[§]
(b) BASIC + MC	3.3×10^{-3}	0.07063	-1.215×10^{-4}	1135.45[§]
(c) BASIC + CAU	3.3×10^{-3}	0.05089	−27.950	2552.28[§]
(d) BASIC + CAU + MC	3.3×10^{-3}	0.07044	−0.261	2650.18[§]
(e) BASIC + TF	3.3×10^{-3}	0.07094	+0.446	2522.093
DG TRI(4)*	1/64	0.07053	+0.24	8600.21[¶]
DG TRI(4)*	1/32	0.07130	−0.85	815.09[¶]
DG QUAD(4)*	1/32	0.07120	−0.71	301.83[¶]
DG TRI(6)*	1/16	0.07188	−1.56	355.16[¶]
WENO(5)†	1/128	0.42500	+39.8	
WENO(5)‡	1/128	0.07020	+0.71	885.94
WENO(5)‡	1/256	0.07050	+0.32	6609.78

*Discontinuous Galerkin [28].
†Level set [29, 30].
‡Particle level set [29, 30].
§Intel Pentium-4 3.6 GHz.
¶Intel Pentium-4 2.4 GHz.

reached at period $T/2$. Moreover, the snapshots corresponding to the periods $T/4$ and $3T/4$ as well as 0.0 and T should be as similar as possible. Table II lists the comparisons between area losses/gains for the schemes proposed in this work with those found in [28–30, 45].

Table II shows us that, following the results obtained for the Zalesak's rotating disk problem, the global mass conservation algorithm combined with the basic scheme yielded the most accurate results when compared to other formulations. Although the computational time was almost 4 times slower than that reported for the DG QUAD(4) in [28], we should consider that the computations were carried out in a finer and three-dimensional mesh. Figure 6 (left) groups the area loss data listed in Table II during the transient solution while Figure 6 (right) shows the band of active entities following the marking function advection for the DD algorithm at instant $T/2$, computed with the BASIC + MC formulation. Note that the number of entities tends to increase as the disk is stretched.

Figure 6 (left) shows that, for this problem, the discontinuity-capturing formulation has drastically failed and presented an area loss of about 30%. For the other schemes the area loss is less than 1%, even for those formulations where no mass correction was employed. Figure 7 shows the snapshots for the formulations suggested in this work at instants $T/2$ and T.

We can observe in Figure 7 how difficult is to keep the original area in this problem, since the original shape is severely distorted up to a very thin filament. Once again, the formulation based on the discontinuity-capturing term is the most diffusive, spending also more computational time. The SUPG formulation, associated with the truncation function (Equation (18)), reached a good result spending less CPU time and with area losses smaller than 0.5%. The global mass conservation algorithm showed its efficiency when associated with the basic scheme, almost recovering the original disk area with a slight additional computational cost. For this problem, the solution with tangent transformation showed a good result even turning the aspect of the interface cogged.

DOI: 10.1002/fld

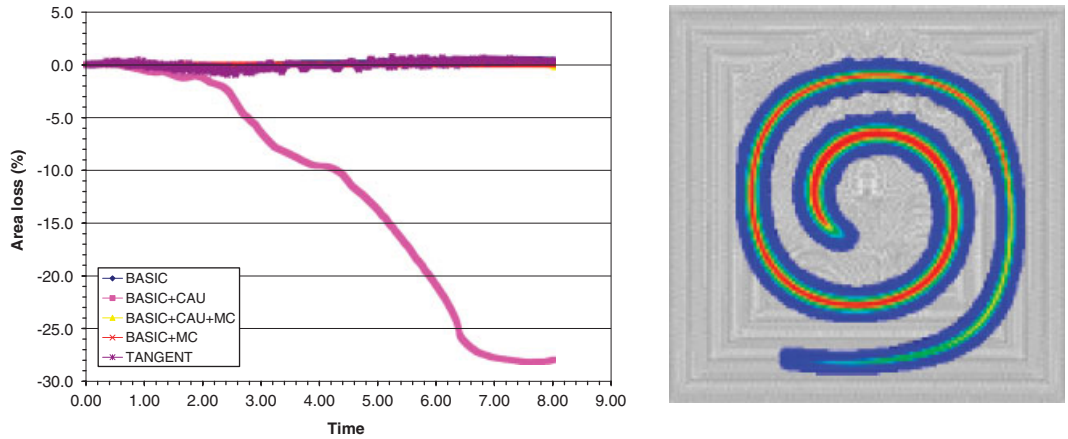

Figure 6. Disk stretching—(left) area loss per time and (right) visualization of the active region (solid filled region).

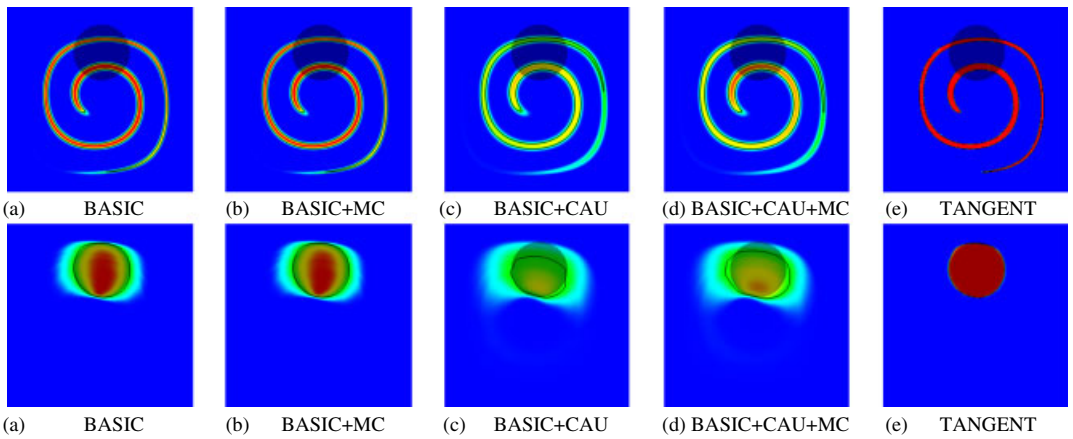

Figure 7. Disk stretching—snapshots for the solution at time $T/2$ and T.

4.3. Three-dimensional deformation field

The three-dimensional deformation field, firstly proposed by LeVeque in [27], is indeed the most challenging conservation problem since it superposes similar deformation fields in the planes x–y and x–z simultaneously imposing severe deformation states to the original shape. Following LeVeque [27], let us consider the velocity field

$$\begin{aligned} u_x &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)g(t) \\ u_y &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)g(t) \\ u_z &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)g(t) \end{aligned} \tag{28}$$

DOI: 10.1002/fld

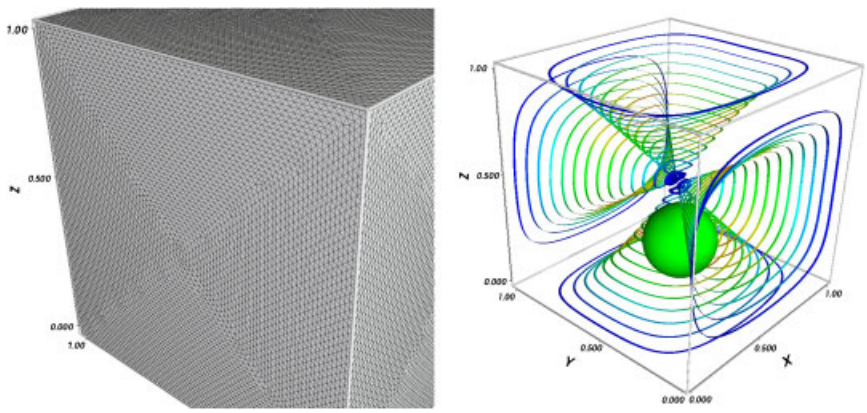

Figure 8. Three-dimensional deformation field—(left) unstructured mesh detail and (right) main vortices and initial shape.

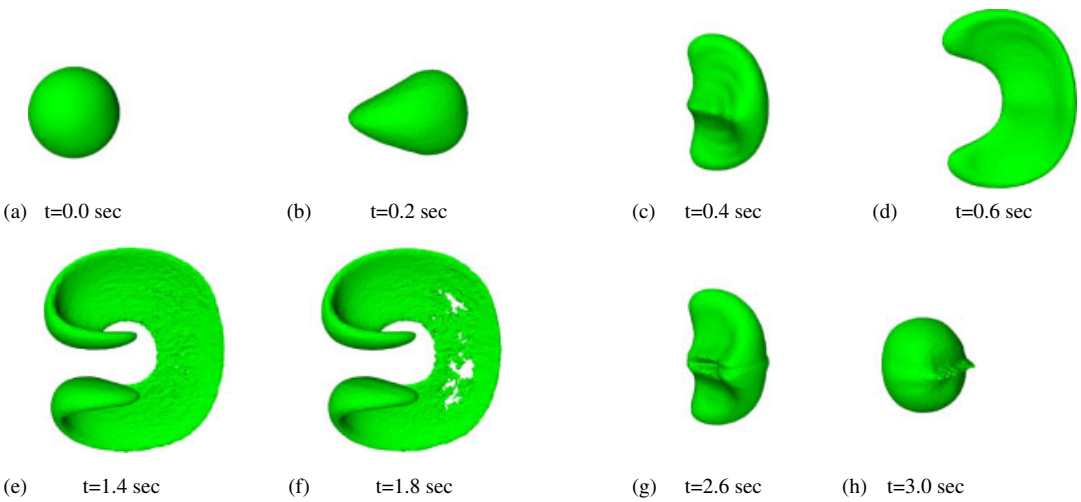

Figure 9. Three-dimensional deformation field—time evolution for the BASIC + mass correction scheme.

where the term $g(t)$ has the same definition of that presented in Section 4.2 and is used to reverse the flow field in a period of time equal to T. For this case we used a period T equal to 3. The problem comprises a sphere with radius 0.15 placed at (0.35, 0.35, 0.35) in a unit computational domain. Therefore, an unstructured mesh is built using 50 line divisions as baseline size as shown in Figure 8 (left) which gives rise to elements with $h \approx 7.5 \times 10^{-3}$ on average. The time step is fixed to 0.02, the GMRES tolerance is 10^{-3} and the maximum number of multi-correction steps is set to 5. The flow field given by Equation (28) forms two rotating vortices (see Figure 8, right) which while carving the opposite sides of the sphere (Figure 9(b)), it squeezes the sphere forming

DOI: 10.1002/fld

Table III. Three-dimensional deformation field—volume loss and performance for various formulations.

Method	h	Volume		CPU time (h)
		Loss at $T/2$	Loss at T	
(a) BASIC	7.5×10^{-3}	−6.8260	−8.4467	0.37§
(b) BASIC + MC	7.5×10^{-3}	+0.0026	−0.5767	0.45§
(c) BASIC + CAU	7.5×10^{-3}	−42.6175	−64.4269	0.74§
(d) BASIC + CAU + MC	7.5×10^{-3}	−0.1873	+0.9578	0.81§
(e) TANGENT	7.5×10^{-3}	−0.9010	−4.2039	0.39§
DG HEX(4)*	1/24	+1.7	+1.63	8.3¶
DG HEX(4)*	1/32	+1.56	+1.53	26.4¶
DG TET(4)*	1/24	+0.85	+0.64	41.1¶
WENO(5)†	1/100	+49.0	+80.0	—
WENO(5)‡	1/100	−1.9	+2.6	—

*Discontinuous Galerkin [28].
†Level set [29, 30].
‡Particle level set [29, 30].
§Intel Pentium-4 3.6 GHz.
¶Intel Pentium-4 2.4 GHz.

a pancake as seen in Figure 9(c) and (d). Afterwards, the top and bottom sphere lobes are caught up by the main vortices transforming the surface in a very slim and stretched shape (see Figure 9(e) and (f)). At this stage, the interface forming both sides of the stretched shape becomes as thin as the grid size, causing the collapse of some regions of the surface and volume losses consequently (Figure 9(f)). After $T/2$ periods of time the flow field is reverted in order to recover the original sphere (Figure 9(f) and (h)). Following the procedure employed for the previous examples we have compared our volume loss/gain results in Table III with those reported in [28–30] using different formulations.

Table III shows that although we have employed a finer mesh, all methods investigated here reached the end of the simulation much faster than the others, even considering the differences between the CPUs of the systems employed. Comparing the case using the BASIC+CAU+MC with that reported by [28] using the DG HEX(4) formulation the CPU time was 10 times faster. Moreover, comparing the most accurate solutions in Table III we can note that the BASIC + MC scheme was 91 times faster for a mesh 5.5 times coarser than the DG TET(4) formulation presenting similar volume losses. This good computational performance can be attributed to the DD algorithm which saves most of the computational costs. It is worthwhile to mention that the same computational device could also be readily extended to be used with other formulations possibly bringing similar savings.

The transient volume losses for the cases listed in Table III are illustrated in Figure 10. Once again the BASIC + CAU scheme presented the worst and more diffusive result losing more than half of the original volume. Figure 11 shows the snapshots corresponding to time $T/2$ and T for the formulations listed in Table III.

We may see from Figure 11 that the interface almost disappears when the maximum stretching is reached for the formulations using the discontinuity-capturing term. It is also important to note how the global mass conservation algorithm is able to overcome this problem recovering the mass

DOI: 10.1002/fld

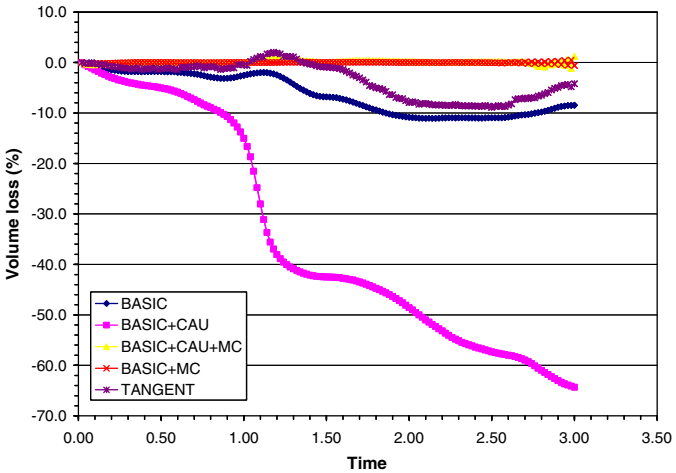

Figure 10. Three-dimensional deformation field—volume loss per time.

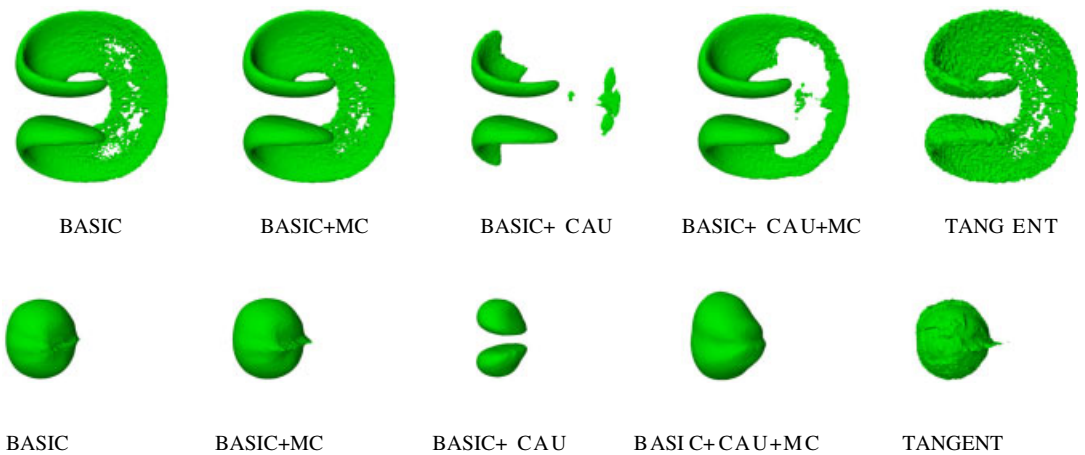

Figure 11. Three-dimensional deformation field—snapshots for the solution at time $T/2$ and T.

loss. Although the formulation using the tangent transformation has reached a good conservation result, it introduced some undesirable wrinkles to the original interface.

5. FREE-SURFACE PROBLEMS

This section aims the discussion of test problems employed to evaluate the performance and robustness of the proposed scheme regarding solution correctness, global mass conservation of the discretization method and the PDD performance in some challenging free-surface problems.

DOI: 10.1002/fld

5.1. *Dam-break problem*

The collapse of a water column is a well-known problem, widely employed to validate free-surface codes based on interface-capturing methods since it has experimental results (see [7, 46] for details) and several simulation results from different numerical methods (see for instance [14, 47, 48]). The problem consists of a water column initially sustained by a dam which is suddenly removed. The water falls under the influence of gravity ($g = 9.81\ \mathrm{m/s^2}$), acting vertically, and flows downward until hitting the opposite wall producing a sloshing effect. The model, as shown in Figure 12, is a box with dimensions $4a \times a \times 2.4a$, where a is a parameter, assumed here to be equal to 0.146 m, following Reference [7]. The water column has dimensions $a \times a \times 2a$. The unstructured mesh was built with 46 766 nodes, 251 807 tetrahedra and 306 597 edges. The density of water is $\rho_w = 1000\mathrm{kg/m^3}$ and the dynamic viscosity $\mu_w = 0.01\mathrm{kg/(m\,s)}$. The density of the air was assumed to be $\rho_a = 1\mathrm{kg/m^3}$ and the dynamic viscosity $\mu_a = 0.0001\mathrm{kg/(m\,s)}$. The maximum inexact-Newton tolerance was set to 10^{-1} while the fixed GMRES tolerance for the advection equation was 10^{-3}. A maximum number of inexact-Newton and multi-correction steps of 4 were chosen, and the nonlinear and multi-correction loops were assumed as converged after a relative residual decrease of 3 orders of magnitude.

The validation results are accessed from the water column leading edge as well as the height of the water column plotted against the dimensionless time as shown in Figure 13. These results when compared to those given in [7, 14, 46–48] show that the proposed scheme is in good agreement with the experimental and numerical reference data. Moreover, from the snapshots shown in Figure 14 for the instants 0.2 and 0.4 s, we can qualitatively see the good results obtained with our code when compared to those presented by Koshizuka *et al*. in [7] in their experiment.

In order to estimate how the formulations influence the volume conservation we present in Figure 15 the transient volume loss for the BASIC and BASIC + CAU schemes, with and without using the global mass conservation algorithm. All computations were carried out using a fixed

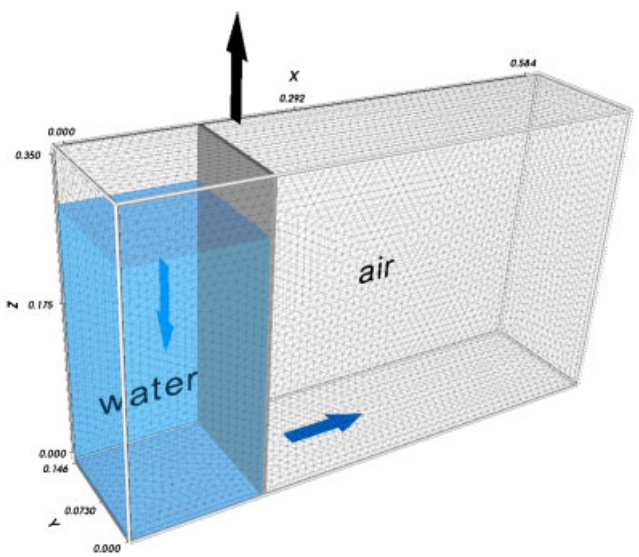

Figure 12. Model for the collapse of a water column problem.

DOI: 10.1002/fld

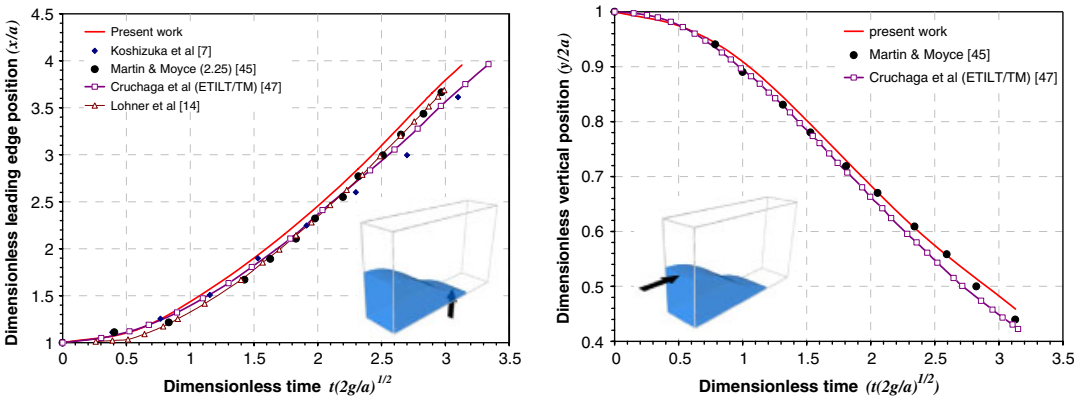

Figure 13. Dam-break validation—(left) leading edge position and (right) water column height.

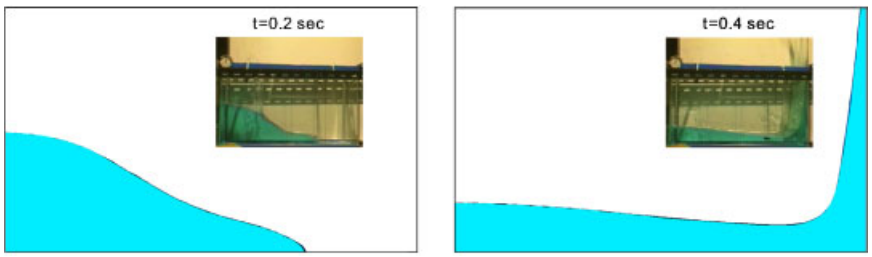

Figure 14. Snapshots for the simulation instant 0.2 and 0.4 s compared with the experimental results presented by Koshizuka *et al.* [7].

time step of 0.01, which resulted in a CFL ranging from 0.3 to 16 for the time interval of interest. Table IV lists the volume lost or gained and the CPU time spent to solve the dam-break problem.

Note that CPU times are almost the same for all schemes. This is due to the fact that most of the computational cost is spent for solving the incompressible Navier–Stokes equations while the costs associated with the marking function transport using PDD is negligible.

Differently from what was observed for the mass conservation tests, given in the previous section, the BASIC and BASIC + CAU showed similar volume losses.

Figure 16 (left) shows the amount of active edges during the simulation for a run with two processors while Figure 16 (right) illustrates how the PDD works. We should emphasize that, in the PDD algorithm, the amount of any finite element entity enabled or disabled has a direct relationship with the computational effort employed to solve the transport equation. Note that the PDD algorithm works enabling and disabling the finite element entities as the solution front advances in such a way that the water column leading edge is followed by the layer of active entities. Moreover, at the beginning of the solution procedure, the whole computational effort is spent only on process 0 while process 1 is kept idle. A good load balance is reached only after 0.27 s and is lost afterwards. This result suggests that even without a good parallel load balance, the PDD algorithm can be faster due its ability to reduce, considerably, the number of equations following the solution regions with high gradients.

DOI: 10.1002/fld

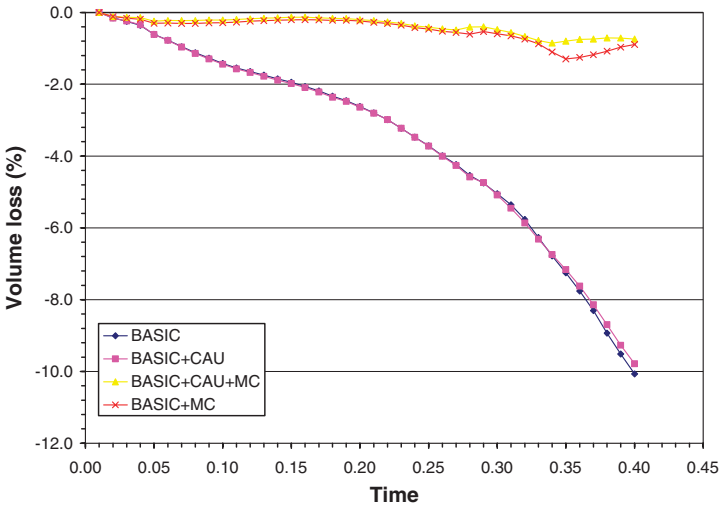

Figure 15. Volume loss/gain for different formulations.

Table IV. Dam-break problem—volume loss and performance.

Method	Volume loss	CPU time (s)
BASIC	−10.06	840.61
BASIC + MC	−0.89	797.48
BASIC + CAU	−9.78	811.11
BASIC + CAU + MC	−0.74	849.86

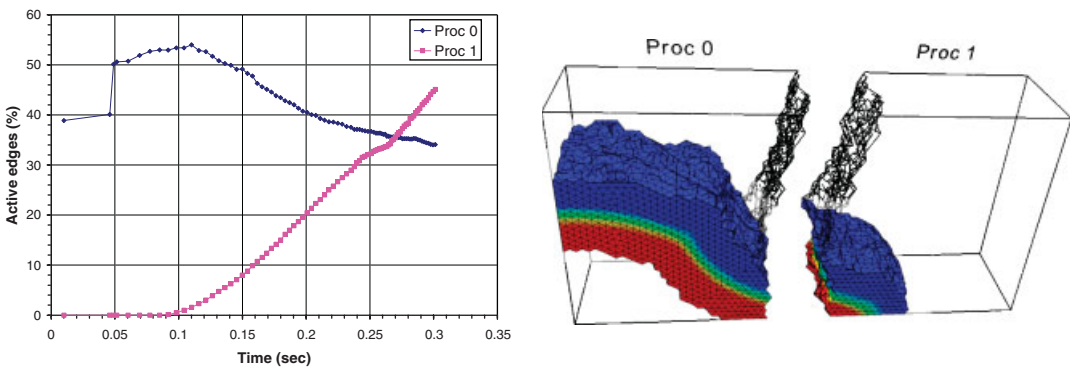

Figure 16. Dam-break problem—(left) Percentage of active edges per processor and (right) snapshot showing the band of active entities crossing the parallel partitions.

5.2. *Wave impact on a container*

The problem of a wave formed by a dam break interacting with a box is a more challenging and realistic application where fluid fragmentation and turbulence effects take part. This problem

DOI: 10.1002/fld

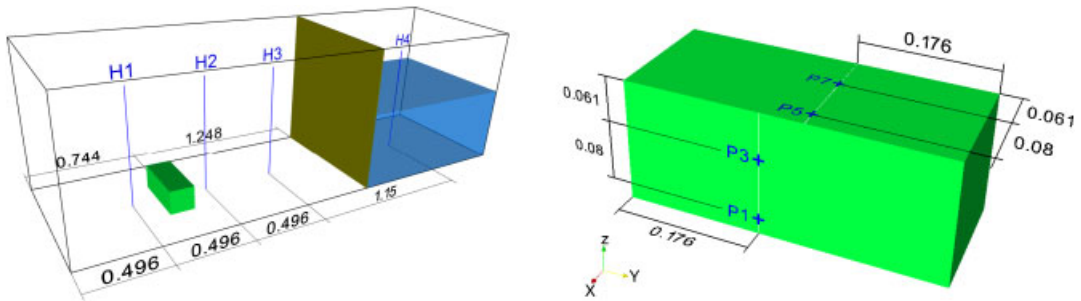

Figure 17. Wave impact on a container—(left) computational model and (right) container detail.

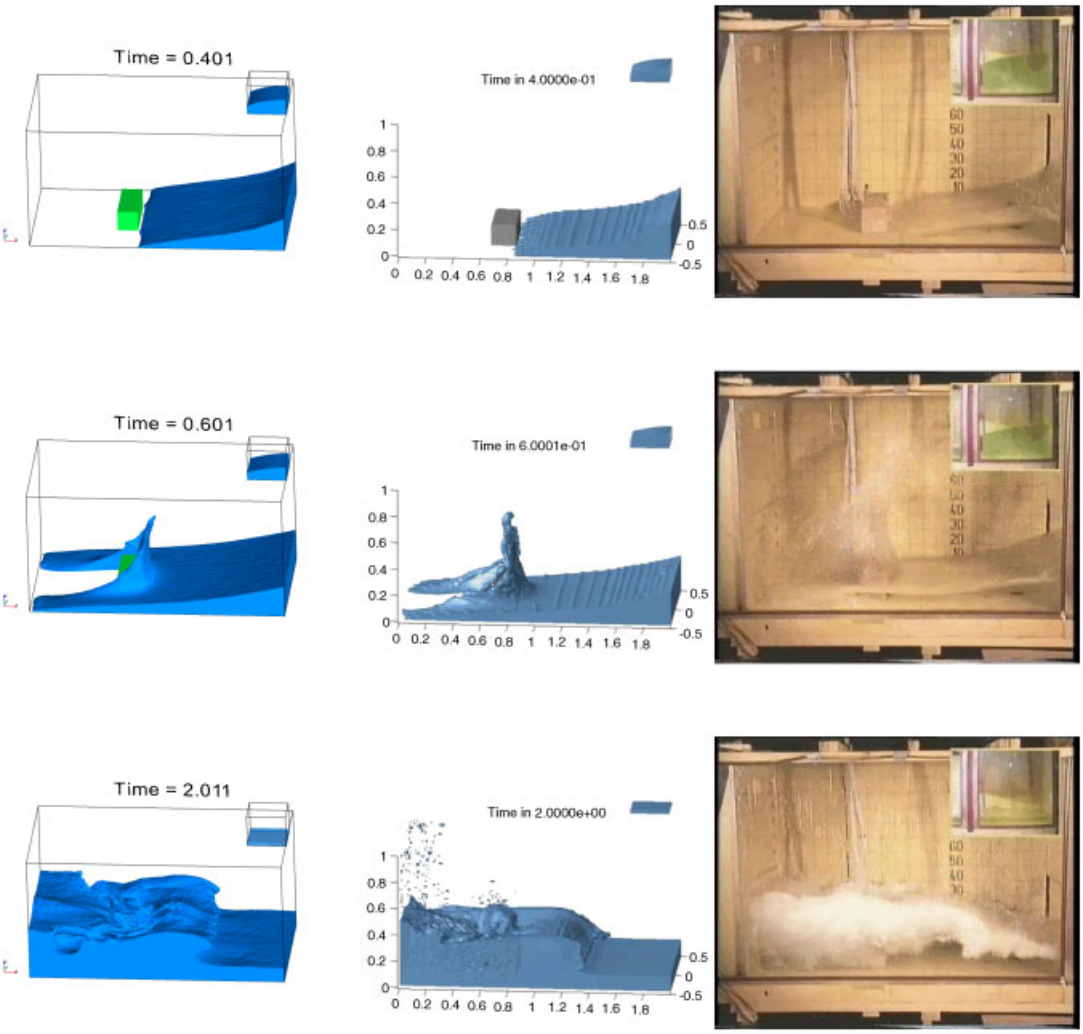

Figure 18. Snapshots for the simulation of the wave impact on a container.

DOI: 10.1002/fld

was experimentally studied by the Maritime Research Institute Netherlands (MARIN) to evaluate the effects of green water flow over the deck of ships. Numerical results are also available for a VOF method employing Cartesian grids in the works of Kleefsman [1] and Kleefsman *et al.* [31]. The model comprises a simple box with dimensions $3.22 \times 1 \times 1$ m where a water column measuring $1.228 \times 1 \times 0.55$ m is supported by a door as shown in Figure 17 (left). The simulation starts when the door is removed, instantaneously releasing the water column, which begins to flow under the gravity action. Therefore, the fluid accelerates until hitting a box with dimensions $0.161 \times 0.403 \times 0.161$ m representing a container model in scale. At this stage the flow turns turbulent in regions close to the box and the interface starts to fragment and fold. The experiments are carried out for 6 s which is enough for the water column to hit the opposite wall twice. The free slip boundary condition is assumed for all walls of the tank as well as for the container faces. The initial condition consists of the column fully filled with water ($\phi = 1$) while the rest of the domain is considered as completely drained ($\phi = 0$). The fluids properties are: $\rho_{\text{water}} = 1000.0\,\text{kg/m}^3$, $\mu_{\text{water}} = 0.001\,\text{kg/(m s)}$, $\rho_{\text{air}} = 1.0\,\text{kg/m}^3$ and $\mu_{\text{air}} = 0.01\,\text{kg/(m s)}$. The air is considered more viscous than the water in order to avoid unstable regions in the air phase. However, in our tests we have not detected any great influence in the water flow arising from this assumption. The validation results are accessed in terms of the height of the water in four measuring stations, H1, H2, H3 and H4, placed at 0.496, 0.992, 1.488 and 2.638 m of the tank origin, respectively, and the pressure at the points P1, P3, P5 and P7 on the container as illustrated in Figure 17 (right). At these locations experimental data are available in [49]. For this

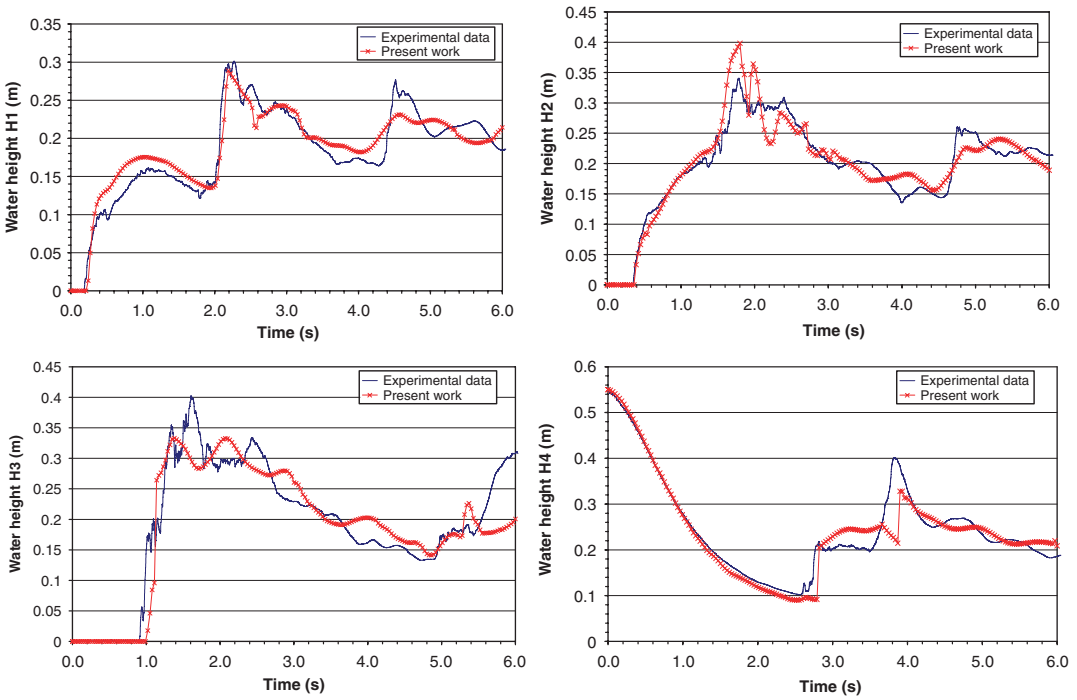

Figure 19. Water impact on a container—water height time histories at measuring stations H1, H2, H3, H4.

DOI: 10.1002/fld

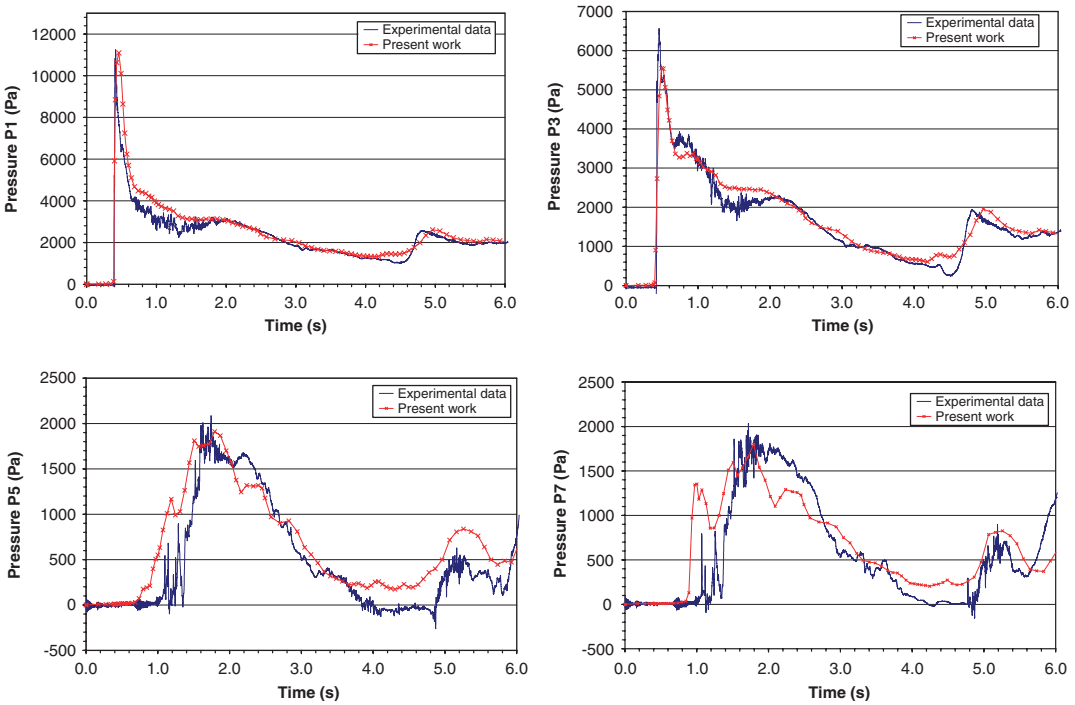

Figure 20. Water impact on a container—pressure time histories at points P1, P3, P5, P7.

problem an unstructured grid with 1 194 961 elements, 1 460 365 edges and 209 272 nodes was built. The maximum inexact-Newton linear tolerance was set to 10^{-1} while the linear tolerance for the advection equation was 10^{-3}. The nonlinear loops were stopped after the relative residual as well as the relative step increment decreased 3 orders of magnitude. The same stop criterion was adopted to the transport multi-correction loop. The PID controller was set to adapt the time step to the CFL range of 0.5–1.5, and the DD algorithm was set to select five layers of entities more than those selected by a η value of 0.8. The turbulence effects were taken into account by the use of a classical Smagorinsky model with C_s equal to 0.1 associated with the LSIC stabilization term. The transport of the marking function was solved by the BASIC scheme with the global mass conservation algorithm (BASIC + MC). The computations were performed in message passing parallelism using eight cores of Itanium-2 Montecito CPUs (1.6 GHz/8 MB) in a SGI Altix 450 system which spent 15 h.

Figure 18 shows water elevation snapshots for the simulation of the wave impact over a container. The leftmost figures are the results of the present work, the numerical results in the central figures and the experimental results (right) are taken from Kleefsman [1] and Kleefsman *et al.* [31]. We may observe a good agreement between the numerical simulation and experiment. In the simulation results of [1, 31] the free surface has some small ripples, which are not present in our results, since the free surface is by construction piecewise linear. In Figure 19 time histories of the water height at the four locations given in Figure 17 (left) are compared with the experimental

DOI: 10.1002/fld

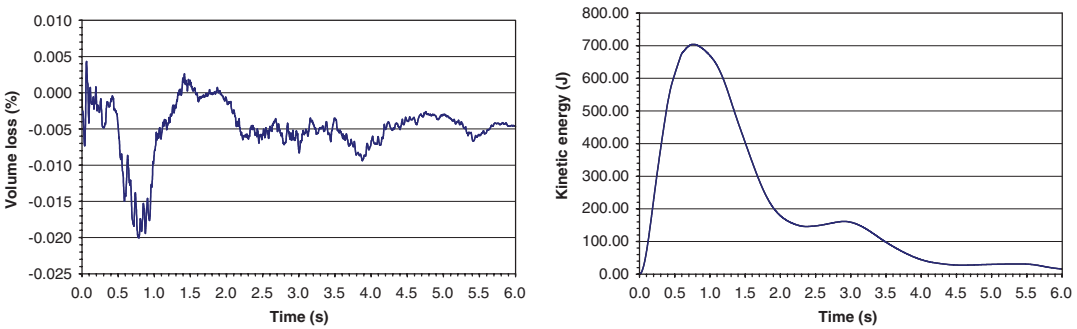

Figure 21. Water impact on a container—(left) water volume loss history and (right) kinetic energy of the wave.

results available in the SPH European Research Interest Community [49]. The agreement between simulation and experiments is very good, particularly at probes H2 and H4. The simulation correctly captures the instant that the wave hits the box and the magnitude of the impact pressure, as can be seen in Figure 20, which shows pressure time histories for both the present simulation and the experimental results in [49], at measuring stations P1, P3, P5 and P7. However, there are small discrepancies between numerical and experimental results at both stations located at the top of the box (P5 and P7). Kleefsman [1] and Kleefsman *et al*. [31] reported pressure spikes in their computed pressure time histories at the same locations. In the present simulations no pressure spikes were observed. It is worthwhile to mention that the numerical results presented by Kleefsman in [1] are very accurate compared to the experimental results, however, the Cartesian grid employed by this author was about 6 times finer than the unstructured grid used in our tests, and the computations were conducted in a CFL number of 0.75. The water volume loss was negligible during all the simulation and ranged on the order of the magnitude of the nonlinear tolerance. Figure 21 (left) shows that the maximum value of 0.02% was found just at the moment the wave head reaches the box. At this moment the wave begins to slosh, fragment and fold until it hits the opposite wall of the tank and starts the reverse flow towards to the reservoir. This moment coincides with the maximum kinetic energy developed by the wave as plotted in Figure 21 (right).

6. CONCLUSIONS

In this work we have presented an extension of a fully implicit parallel edge-based incompressible flow solver for free-surface flows on unstructured grids. The unsteady three-dimensional Navier–Stokes equations were discretized with a SUPG/PSPG stabilized finite element formulation. The main characteristics of this flow solver are: implicit time marching scheme with adaptive time stepping control; advanced inexact Newton solvers; edge-based data structures to save memory and improve performance; support to message passing and shared memory parallel programming models. Into this flow solver we introduced volume-of-fluid (VOF) extensions to track the evolving free surface. The advection equation for the scalar marking function was solved by a fully implicit

DOI: 10.1002/fld

parallel edge-based SUPG finite element formulation. We have studied variants of this formulation considering the effects of discontinuity capturing and a particular tangent transformation designed to increase interface sharpness. Global mass conservation is enforced adding or removing mass proportionally to the absolute value of the normal velocity of the interface. This guarantees that the mass correction will act mainly in regions where the interface is moving faster, while keeping the stationary regions untouched. This procedure is accurate provided the volume phases are computed correctly. Hence, we have presented an accurate parallel algorithm to compute the volume of each fluid phase, which makes use of the interface explicitly determined for those elements lying on the interface. We introduced here another computational artefact to further improve the overall efficiency of the present free-surface solver, the parallel dynamic deactivation (PDD) technique for solving the marking function. This technique restricts the computation to regions where a defined gradient is found. Since the marking function employed on VOF methods presents steep gradients, the DD algorithm catches and restricts the computations only on regions around the interface. In the PDD extension, each processor estimates its own number of finite element entities enabled for computation and the loops are constrained to the number of active entities (nodes, edges and elements). Consequently, the overall effort (computation and communication) to solve the problem is drastically reduced.

The resulting computational strategy for the marking function was validated using three standard global mass conservation test cases: the Zalesak's disk; the disk stretching; and the three-dimensional deformation field. We have investigated numerically the effects of introducing discontinuity capturing and solving the marking function with the tangent transformation, the accuracy of the global mass conservation scheme and the computational gains achieved by the DD algorithm. We have compared our results with the discontinuous Galerkin, ENO, WENO and particle level sets methods. We observed that the combination of SUPG and the global mass conservation scheme produced the fastest and most accurate results. Area and volume losses are of the same order of the best discontinuous Galerkin and particle level set methods. However, although the tangent transformation produced solutions where the interface between the fluids is of the order of the mesh size, we have observed undesired wrinkles in the interface. The DD algorithm yielded faster solutions with no loss of accuracy. The combined effect of all these features resulted in very efficient solutions.

In the sequel we validated the whole solution procedure solving the classical dam-break problem and the problem of the wave impact on a container. In the dam-break problem good agreement was observed between our results and other numerical and experimental results for the time histories of the water column leading edge as well as the height of the water column. We noticed in this case that the global mass conservation procedure is very effective, keeping the volume loss below 1% for solutions considering or not discontinuity capturing. Indeed, thanks to the edge-based data structure and the PDD algorithm, the computational effort to solve the marker using any formulation is negligible compared to solving the Navier–Stokes equations. We also investigated a more challenging and realistic application, the simulation of a wave formed by a dam break interacting with a box, where fluid fragmentation and turbulence effects take part. We continued to obtain a good agreement between the present numerical simulation and the available experimental data for the height of the water in four measuring stations, and pressure at several points over the box. Besides, in our simulation the free surface has no ripples, as opposed to numerical results reported earlier, since the free surface is by construction piecewise linear. In addition, no pressure spikes were observed in our simulations, unlike previously reported numerical results.

DOI: 10.1002/fld

ACKNOWLEDGEMENTS

The authors would like to thank the financial support of the Petroleum National Agency (ANP, Brazil) and MCT/CNPq, The Brazilian Council for Scientific Research. The Center for Parallel Computations (NACAD) at the Federal University of Rio de Janeiro provided the computational resources for this research.

REFERENCES

1. Kleefsman KMT. Water impact loading on offshore structures: a numerical study. *Ph.D. Thesis*, University of Groningen, Groningen, 2005.
2. Tezduyar TE. CFD methods for three-dimensional computation of complex flow problems. *Journal of Wind Engineering and Industrial Aerodynamics* 1999; **81**:97–116.
3. Tezduyar TE. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics* 1992; **28**:1–44.
4. Tezduyar TE. Computation of moving boundaries and interfaces and stabilization parameters. *International Journal for Numerical Methods in Fluids* 2003; **43**:555–575.
5. Tezduyar TE, Behr M, Liou J. A new strategy for finite element computations involving moving boundaries and interfaces—the deforming-spatial-domain/space–time procedure: I. The concept and the preliminary numerical tests. *Computer Methods in Applied Mechanics and Engineering* 1992; **94**:339–351.
6. Tezduyar TE, Behr M, Mittal S, Liou J. A new strategy for finite element computations involving moving boundaries and interfaces—the deforming-spatial-domain/space–time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders. *Computer Methods in Applied Mechanics and Engineering* 1992; **94**:353–371.
7. Koshizuka S, Tamako H, Oka Y. A particle method for incompressible viscous flow with fluid fragmentation. *Computational Fluid Mechanics Journal* 1995; **113**:134–147.
8. Violeau D, Issa R. Numerical modelling of complex turbulent free-surface flows with the SPH method: an overview. *International Journal for Numerical Methods in Fluids* 2007; **53**:277–304.
9. Del Pin F, Idelsohn S, Oñate E, Aubry R. The ALE/Lagrangian particle finite element method: a new approach to computation of free-surface flows and fluid–object interactions. *Computers and Fluids* 2007; **36**:27–38.
10. Takizawa K, Yabe T, Tsugawa Y, Tezduyar TE, Mizoe H. Computation of free-surface flows and fluid–object interactions with the CIP method based on adaptive meshless Soroban grids. *Computational Mechanics* 2006; DOI:10.1007/s00466-006-0093-2
11. Hirt CW, Nichols BD. Volume of fluid (VOF) methods for the dynamics of free boundaries. *Journal of Computational Physics* 1981; **39**:201–225.
12. Sethian JA. *Level Set Methods and Fast Marching Methods*: *Evolving Interfaces in Computational Geometry*, *Fluid Mechanics*, *Computer Vision*, *and Materials Science*. Cambridge University Press: Cambridge, U.K., 1999.
13. Tezduyar T, Aliabadi S, Behr M. Enhanced-discretization interface-capturing technique (EDICT) for computation of unsteady flows with interfaces. *Computer Methods in Applied Mechanics and Engineering* 1998; **155**:235–248.
14. Lohner R, Yang C, Onate E. On the simulation of flows with violent free-surface motion. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:5597–5620.
15. Nagrath S, Jansen KE, Lahey Jr RT. Computation of incompressible bubble dynamics with a stabilized finite element level set method. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4565–4587.
16. Shepel SV, Smith BL. New finite-element/finite-volume level set formulation for modelling two-phase incompressible flows. *Journal of Computational Physics* 2006; **218**:479–494.
17. *CFX-4 User Documentation*. AEA Technology, Harwell, Oxfordshire, U.K. 2000.
18. Brooks AN, Hughes TJR. Streamline-upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equation. *Computer Methods in Applied Mechanics and Engineering* 1982; **32**:199–259.
19. Coutinho ALGA, Martins MAD, Sydenstricker RM, Elias RN. Performance comparison of data-reordering algorithms for sparse matrix–vector multiplication in edge-based unstructured grid computations. *International Journal for Numerical Methods in Engineering* 2006; **66**:431–460.
20. Elias RN, Coutinho ALGA, Martins MAD. Inexact Newton-type methods for the solution of steady incompressible viscoplastic flows with the SUPG/PSPG finite element formulation. *Computer Methods in Applied Mechanics and Engineering* 2006; **195**:3145–3167.

21. Elias RN, Martins MAD, Coutinho ALGA. *Parallel Edge-based Inexact Newton solution of Steady Incompressible 3D Navier–Stokes Equations*. Lecture Notes in Computer Science, vol. 3648. Springer: Berlin, 2005; 1237–1245.
22. 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.
23. Yabe T, Xiao F. Description of complex and sharp interface with fixed grids in incompressible and compressible fluid. *Computers and Mathematics with Applications* 1995; **29**(1):15–25.
24. Lohner R, Camelli F. Dynamic deactivation for advection-dominated contaminant transport. *Communications in Numerical Methods in Engineering* 2004; **20**:639–646.
25. Zalesak ST. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics* 1979; **31**:335–362.
26. Bell J, Collela P, Glaz H. A second-order projection method for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1989; **85**:257–283.
27. LeVeque R. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis* 1996; **33**:627–665.
28. Marchandise E, Remacle JF, Chevaugeon N. A quadrature-free discontinuous Galerkin method for the level set equation. *Journal of Computational Physics* 2006; **212**:338–357.
29. Enright D, Fedkiw R, Ferziger J, Mitchell I. A hybrid level set method for improved interface capturing. *Journal of Computational Physics* 2002; **183**:83–116.
30. Enright D, Losasso F, Fedkiw R. A fast and accurate particle level set method. *Computers and Structures* 2005; **83**(6–7):479–490.
31. Kleefsman KMT, Fekken G, Veldman AEP, Iwanowski B, Buchner B. A volume-of-fluid based simulation method for wave impact problems. *Journal of Computational Physics* 2005; **206**:363–393.
32. Smagorinsky J. General circulation experiments with primitive equations. Part I: The basic experiment. *Monthly Weather Review* 1963; **91**:99–152.
33. Tezduyar TE, Osawa Y. Finite element stabilization parameters computed from element matrices and vectors. *Computer Methods in Applied Mechanics and Engineering* 2000; **190**:411–430.
34. Tezduyar TE. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering* 2001; **8**:83–130.
35. Galeão AC, do Carmo EGD. A consistent approximate upwind Petrov–Galerkin method for convection-dominated problems. *Computer Methods in Applied Mechanics and Engineering* 1988; **68**(1):83–95.
36. Tezduyar TE, Sathe S. Enhanced-discretization selective stabilization procedure (EDSSP). *Computational Mechanics* 2006; **38**:456–468.
37. 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.
38. Elias RN, Martins MAD, Coutinho ALGA. Parallel edge-based solution of viscoplastic flows with the SUPG/PSPG formulation. *Computational Mechanics* 2006; **38**:365–381.
39. Valli AMP, Carey GF, Coutinho ALGA. Control strategies for timestep selection in fe simulation of incompressible flows and coupled reaction–convection–diffusion processes. *International Journal for Numerical Methods in Fluids* 2005; **47**:201–231.
40. Martins MAD, Elias RN, Coutinho ALGA. *EdgePack*: *A Parallel Vertex and Node Reordering Package for Optimizing Edge-based Computations in Unstructured Grids*. Lecture Notes in Computer Science, vol. 4395. Springer: Berlin, 2007; 292–304.
41. Cruchaga M, Celentano D, Tezduyar T. A moving Lagrangian interface technique for flow computations over fixed meshes. *Computer Methods in Applied Mechanics and Engineering* 2001; **191**:525–543.
42. Cruchaga M, Celentano D, Tezduyar T. Computation of mould filling processes with a moving Lagrangian interface technique. *Communications in Numerical Methods in Engineering* 2002; **18**:483–493.
43. Souza DAF, Martins MAD, Coutinho ALGA. Edge-based adaptive implicit/explicit finite element procedures for three-dimensional transport problems. *Communications in Numerical Methods in Engineering* 2005; **21**:545–552.
44. Tezduyar TE, Liou J. Adaptive implicit–explicit finite element algorithms for fluid mechanics problems. *Computer Methods in Applied Mechanics and Engineering* 1990; **78**:165–179.
45. Sussman M, Fatemi E, Smereka P, Osher S. An improved level set method for incompressible two-fluid flows. *Computers and Fluids* 1998; **27**:663–680.
46. Martin JC, Moyce WJ. An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Philosophical Transactions of the Royal Society of London*, *Series A*, 1952; **244**:312–324.

DOI: 10.1002/fld

47. Greaves DM. Simulation of viscous water column collapse using adapting hierarchical grids. *International Journal for Numerical Methods in Fluids* 2005; **50**(6):693–711.
48. Cruchaga MA, Celentano DJ, Tezduyar, TE. Moving-interface computations with the edge-tracked interface locator technique (ETILT). *International Journal for Numerical Methods in Fluids* 2005; **47**:451–469.
49. SPH European Research Interest Community. http://wiki.manchester.ac.uk/spheric/index.php/SPHERIC_Home_Page (18th December 2006).

DOI: 10.1002/fld