Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 194 (2004) 35–56

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A quadtree adaptive method for simulating fluid flows with moving interfaces

Deborah Greaves

Department of Architecture and Civil Engineering, University of Bath, Claverton Down, Bath BA2 7AY, UK

Received 9 October 2002; received in revised form 27 June 2003; accepted 21 August 2003

Abstract

In this paper, a computational method for solving fluid flow problems with moving interfaces is presented. Herein, adaptive quadtree grids are used coupled with the CICSAM [O. Ubbink, Numerical prediction of two fluid systems with sharp interfaces, PhD Thesis, Imperial College of Science, Technology and Medicine, London, 1997] free surface capturing volume of fluid (VoF) method and PLIC reconstruction to interpolate the volume fraction field during refinement and derefinement processes. The combination of high resolution adaptive hierarchical remeshing and CICSAM interface advection is shown to overcome the problems of interface smearing and high CPU intensivity inherent in most VoF schemes. The result is a combination of free surface tracking and free surface capturing in that the interface is effectively tracked by the adapting refinements in the quadtree grid. In this way, a sharp interface is achieved and the advantages of both free surface tracking and capturing are combined. The new method is applied to interface advection examples in translating, rotating and shearing flow fields, and the benefits of using adapting quadtree grids demonstrated.

Keywords: Adaptive grids; Quadtrees; Grid refinement; Volume of fluid; Moving fluid interface

1. Introduction

Moving fluid interface flows occur in many areas of environmental, biological and industrial engineering. Accurate modelling of the moving interface, such as a wave free surface or Taylor bubble, is an extremely challenging problem in CFD because the position of the interface at a given time is not known in advance and must be calculated as part of the solution.

Various techniques have emerged to predict the position of the interface during the solution in time and fall into one of two categories. These are interface tracking methods, which include moving mesh, front tracking and particle tracking schemes; and interface capturing methods, which include volume of fluid (VoF) and level set techniques. Front tracking methods use a piecewise polynomial fit to the

E-mail address: d.m.greaves@bath.ac.uk (D. Greaves).

0021-9991/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.jcp.2003.08.018

interface, which is advected with the flowfield [2]. Moving mesh methods may be accurate and have been applied successfully in simulation of drop formation [3], roll-coating flows [4] and non-breaking free surface waves [5]. However, they encounter problems when the interface turns over on itself during free surface wave breaking with associated air entrainment and break-up into spray. Particle tracking methods are very expensive and not currently practical in three dimensions. Level set and VoF are both front capturing methods and can be used for modelling large-scale deformations of the interface including break up and merging. They differ from front tracking in that the solution is calculated in the combined fluid domains, with the fluid properties changing at the interface. The interface is then located from the zero contour of a distance function in the case of level set [6] and from the volume fraction field in the VoF method [7].

A VoF method consists of two parts: an interface reconstruction algorithm for approximating the interface from the set of volume fractions and a VoF transport algorithm for determining the volume fractions at the new time level from the velocity field and the reconstructed front. The basic method is robust and flexible and VoF schemes are widely used [7–9]. The major drawbacks of this method are its tendency to smear the interface and the high CPU cost due to the need for fine grids and small time steps. The use of high resolution adaptive hierarchical remeshing and CICSAM interface advection used in this work overcomes these drawbacks and combines the benefits of interface tracking and capturing. The interface remains sharp and is effectively tracked by adapting refinements in the quadtree grid.

In this paper, interface tracking schemes for the VoF method are discussed and three schemes described in detail. Calculations of advecting interfaces in a known velocity field are made with the three chosen schemes and the results compared. The interface tracking schemes are calculated on quadtree grids, which adapt to refine the grid at the interface as it is advected. Comparisons are drawn between calculations made on uniform and quadtree adapted grids and the benefits of using quadtree grids demonstrated.

2. The volume of fluid, VoF, method

When considering the incompressible flow of two immiscible fluids, the divergence free velocity field $\underline{u}(x, y, t)$ obeys

$$\nabla \cdot \underline{u} = 0. \tag{1}$$

The location of the two fluids is specified using a volume fraction function, C, with $C = 1$ inside one fluid and $C = 0$ in the other. Cells for which C lies between 0 and 1 contain the interface. The volume conservation of the first fluid can be expressed as

$$\frac{\partial C}{\partial t} + \nabla \cdot (\underline{u}C) = 0. \tag{2}$$

The key to a successful VoF scheme is to solve the volume fraction equation in a way that keeps the interface sharp.

The original VoF scheme of Hirt and Nichols [7] uses an interface construction that approximates a curved interface as horizontal and vertical lines in each interface cell. The fluxing scheme uses a combination of upwinding and downwinding. The advantage of the upwind scheme is that it is stable, but it is diffusive and may spread the interface over many cells. The downwind scheme is unstable, but sharpens the interface and so is advantageous in interface tracking. Various VoF fluxing methods have been developed, most of which aim for a balance between the stability advantages of the upwind scheme and the front sharpening advantages of the downwind scheme.

2.1. SURFER

Lafaurie et al. [10] describe a fluxing scheme in which either upwind or modified downwinding is used depending on the orientation of the interface relative to the fluxing direction. In order to prevent appreciable volume error, it is necessary to direction split the algorithm so that fluxes are calculated in the x-direction for the entire grid and then a y-direction sweep is performed. When the interface is perpendicular to the flow, they use the front sharpening modified downwind scheme and when the interface is parallel to the flow, upwinding is used. The downwind scheme is modified to constrain the volume fraction face value to prevent more fluid being outfluxed than the cell contains and to ensure that the CFL condition is satisfied. Lafaurie et al. [10] also introduced a flotsam indicator field, which is used to ensure that flotsam (a cell partly containing one of the fluids entirely surrounded by cells full of the other fluid) is fluxed by upwinding rather than downwinding, which would lock it in place and prevent it being fluxed. This scheme is implemented in the SURFER program.

The SURFER VoF differencing scheme used for discretisation of the volume fraction Eq. (2) is demonstrated by considering the local grid for x-direction fluxing shown in Fig. 1. The volume fraction equation in discrete form is given by

$$C^{n+1} = C^n + \sum_{k=1}^{k=K} f_k, \tag{3}$$

where k is the cell face orientation and K is the number of cell faces. For a regular rectangular grid in two dimensions, $K = 4$ and $k =$ e, w, n or s. f_k represents the flux of C across the k direction cell boundary. Thus,

$$C^{n+1} = C^n + f_{\mathrm{e}} - f_{\mathrm{w}} + f_{\mathrm{n}} - f_{\mathrm{s}}, \tag{4}$$

and the cell face fluxes are

$$f_k = u_k \frac{\delta t}{\delta x} C_k, \tag{5}$$

where δt and δx are the time step and mesh size, u_k is the velocity at face k and determination of the cell face value of C is critical.

The technique for x-direction fluxing is explained here; the y-direction fluxing is similar. If we consider $u > 0$, the face values for the upwind scheme are $C_{\mathrm{e}}^{\mathrm{upwind}} = C_{\mathrm{C}}$ and $C_{\mathrm{w}}^{\mathrm{upwind}} = C_{\mathrm{W}}$. Referring to Fig. 1, uppercase subscripts indicate values at cell centres and lowercase subscripts indicate cell face values. The corresponding flux at the east face is

$$f_{\mathrm{e}}^{\mathrm{upwind}} = \left[u_{\mathrm{e}} \frac{\delta t}{\delta x} \right] C_{\mathrm{C}}. \tag{6}$$

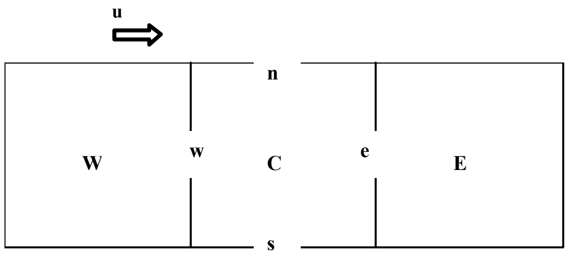

Fig. 1. Diagram for x-direction fluxing.

When the CFL condition is satisfied, $|u|\delta t/\delta x < 1$, the cell will not out-flux more than it contains. The upwind scheme is stable but diffusive and tends to smear the interface over many cells.

For $u > 0$, the downwind scheme leads to $C_{\mathrm{e}}^{\mathrm{downwind}} = C_{\mathrm{E}}$ and $C_{\mathrm{w}}^{\mathrm{downwind}} = C_{\mathrm{C}}$. However, the downwind face flux uses the east neighbour value for C and so could try to extract from the cell more than it contains

$$f_{\mathrm{e}}^{\mathrm{downwind}} = \left[u_{\mathrm{e}}\frac{\delta t}{\delta x}\right]C_{\mathrm{E}}. \tag{7}$$

The downwind scheme is therefore unstable, but it does have the advantage of sharpening the interface. Lafaurie et al. [10] introduced a modified downwind scheme, which prevents the cell out-fluxing more than it contains

$$\begin{aligned}
&\text{If } C_{\mathrm{C}} < f_{\mathrm{e}}^{\mathrm{downwind}}, \quad \text{then } f_{\mathrm{e}}^{\mathrm{downwind}} = C_{\mathrm{C}}.\\
&\text{If } f_{\mathrm{e}}^{\mathrm{downwind}} < C_{\mathrm{C}} < 1 - u_{\mathrm{e}}\frac{\delta t}{\delta x} + f_{\mathrm{e}}^{\mathrm{downwind}}, \quad \text{then } f_{\mathrm{e}}^{\mathrm{downwind}} = \left[u_{\mathrm{e}}\frac{\delta t}{\delta x}\right]C_{\mathrm{E}}.\\
&\text{If } 1 - u_{\mathrm{e}}\frac{\delta t}{\delta x} + f_{\mathrm{e}}^{\mathrm{downwind}} < C_{\mathrm{C}}, \quad \text{then } f_{\mathrm{e}}^{\mathrm{downwind}} = u_{\mathrm{e}}\frac{\delta t}{\delta x} - 1 + C_{\mathrm{C}}.
\end{aligned} \tag{8}$$

The upwind or modified downwind schemes are chosen depending on the angle between the interface normal and the fluxing direction. If the interface is perpendicular to the fluxing direction, the front sharpening aspects of the downwind scheme are advantageous, however, if the interface is parallel the upwind scheme is used. Here, the x-direction fluxing is considered and so if the interface is close to vertical the downwind scheme is used and if the interface is close to horizontal then the upwind scheme is used. A critical angle, θ_{c} is defined such that if $\theta < \theta_{\mathrm{c}}$ then $f_{\mathrm{e}} = f_{\mathrm{e}}^{\mathrm{downwind}}$ and the modified downwind scheme is used; if $\theta > \theta_{\mathrm{c}}$ then $f_{\mathrm{e}} = f_{\mathrm{e}}^{\mathrm{upwind}}$ and the upwind scheme is used. Lafaurie et al. [10] found that the results were best for a critical angle, $1.0 < \theta_{\mathrm{c}} < 1.05$ in radians. A value of $\theta_{\mathrm{c}} = 1.0$ is used in this work. The orientation angle of the interface with the k direction is

$$\theta_k = \arccos(n_k), \tag{9}$$

where n_k is the local approximation to the interface normal

$$\underline{n} = \frac{\nabla^h C}{|\nabla^h C|}, \tag{10}$$

and ∇^h is a finite difference approximation to the gradient operator. The gradient is calculated using

$$\nabla^h = \frac{C_{\mathrm{e}} - C_{\mathrm{w}}}{\delta x} + \frac{C_{\mathrm{n}} - C_{\mathrm{s}}}{\delta y}, \tag{11}$$

where face values are calculated as the average of adjacent cell centre values and zero gradient boundary conditions are applied for C.

2.2. PLIC VoF

The VoF scheme outlined above describes the interface implicitly since the volume fraction data must be inverted to find the approximate interface position. The interface may be reconstructed by SLIC (simple line interface calculation) [11] or by various PLIC (piecewise linear interface calculation) methods [12]. Gueyffier et al. [13] developed a VoF/PLIC method and applied it to simulation of three-dimensional droplets.

A straight line is defined in each interface cell that divides the cell into two parts, each of which contains the correct volume of one of the two fluids. This interface segment is propagated by the flow and the resulting volume of each fluid in the neighbouring cells is calculated. It is necessary both to calculate the volume of a fluid from the equation of the interface segment in the cell and inversely the equation of the interface from the volume of fluid in the cell. The interface normal is calculated from

$$\underline{m} = \nabla^h C. \tag{12}$$

This is similar to the normal vector determined using Eq. (10), but in this case the vector is not normalised. The equation for a straight line with normal $\underline{m}$ is

$$m_x x + m_y y = \alpha, \tag{13}$$

where m_x and m_y are components of $\underline{m}$ and α is a parameter related to the smallest distance between the line and the origin. If $\underline{m}$ is a unit normal then α is equal to the distance between the line and the origin.

The origin is defined at the bottom left hand corner of the cell as shown in Fig. 2. Points E and H are where the line intersects the x- and y-axes at α/m_x and α/m_y, respectively. The area below the line and contained within the cell ABCD is

$$\text{Area} = \frac{\alpha^2}{2m_x m_y}\left[1 - H(\alpha - m_x\delta x)\left(\frac{\alpha - m_x\delta x}{\alpha}\right)^2 - H\big(\alpha - m_y\delta y\big)\left(\frac{\alpha - m_y\delta y}{\alpha}\right)^2\right]. \tag{14}$$

The equation contains the Heaviside step function

$$H(x) = \begin{cases} 0 & \text{for } x < 0, \\ 1 & \text{for } x < 0. \end{cases}$$

The term $\alpha^2/2m_x m_y$ is the area of the triangle AEH and the two terms containing the Heaviside function are the areas that must be subtracted from triangle AEH if either point E or point H lies outside the cell. The interface is propagated by Lagrangian advection from one time step to the next. The time stepping is

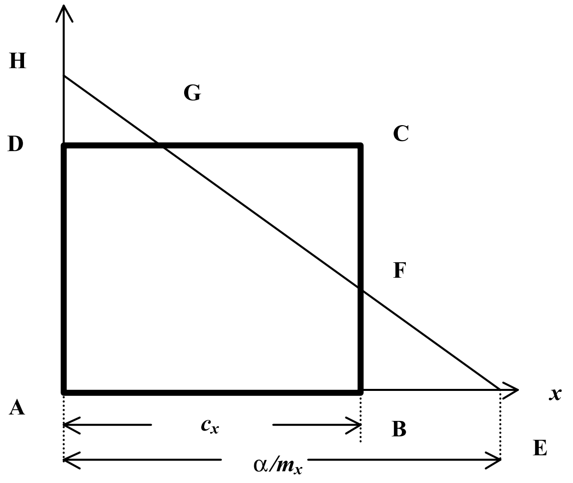

Fig. 2. Cell ABCD is cut by the straight line EH, having normal $\underline{m}$ and parameter α, and contains fluid 1 in region ABFGD and fluid 2 in region FCG.

performed separately in each spatial direction. The x-component of the velocity at each point on the line is a simple linear interpolation between the values on the cell faces, U_w and U_e,

$$u(x) = U_\mathrm{w}\left(1 - \frac{x}{\delta x}\right) + U_\mathrm{e}\frac{x}{\delta x}. \tag{15}$$

During the x-sweep, the x-coordinate of the line changes to the new value

$$x^* = x^n + u(x^n)\delta t = \left[1 + \left(\frac{U_\mathrm{e} - U_\mathrm{w}}{\delta x}\right)\delta t\right]x^n + U_\mathrm{w}\delta t. \tag{16}$$

The superscript n denotes the value at time n and the asterisk is used to denote a fractional step, the x-sweep, to be followed by a similar step in y. For example, in Fig. 3 the interface line *ab* in cell *ABCD* has been advected to *cd*. The new volume of fluid contained in cell *ABCD* and cell *EFBA* as a result of the advection must be calculated. Eq. (16) is inverted to obtain x^n as a function of x^* and the new values of x^n are substituted into (13) to give

$$m_x^n\left[\frac{x^* - U_\mathrm{w}\delta t}{1 + ((U_\mathrm{e} - U_\mathrm{w})/\delta x)\delta t}\right] + m_y^n y^n = \alpha^n. \tag{17}$$

This can be written as

$$m_x^* x^* + m_y^* y^* = \alpha^*, \tag{18}$$

in which the new values of α^* and $\underline{m}^*$ are

$$m_x^* = \frac{m_x^n}{1 + ((U_\mathrm{e} - U_\mathrm{w})/\delta x)\delta t} \tag{19}$$

and

$$\alpha^* = \alpha^n + \frac{m_x^n U_\mathrm{w}\delta t}{1 + ((U_\mathrm{e} - U_\mathrm{w})/\delta x)\delta t}. \tag{20}$$

Also, the volume of fluid that has moved into the right or left neighbouring cell may be calculated. For example, if $\alpha^*/m_x^* > \delta x$, then some of the volume has moved into the right hand cell. This volume can be calculated from (14) after coordinate transformation of

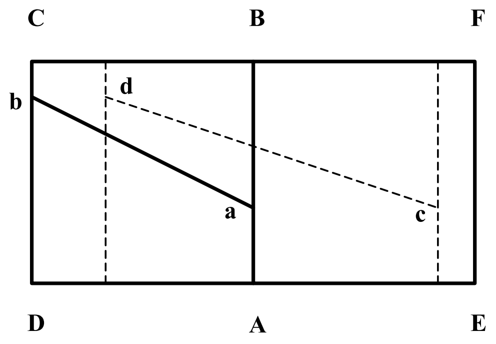

Fig. 3. Lagrangian propagation of the interface during the horizontal advection step.

$$x^* = \delta x + x', \tag{21}$$

where x' is the distance from the left face of the right hand cell. Now Eq. (13) becomes

$$m_x^* x' + m_y^* y = \alpha', \tag{22}$$

where $\alpha' = \alpha^* - m_x^* \delta x$.

2.3. CICSAM VoF

Ubbink [1] derived a compressive differencing scheme for discretisation of the volume fraction equation (2). The scheme is named CICSAM (compressive interface capturing scheme for arbitrary meshes) and is based on the normalised variable diagram (NVD) used by Leonard [14]. Ubbink's [1] scheme combines the convection boundedness criteria (CBC) with the ultimate quickest (UQ) differencing scheme (both defined below), which is a version of Leonard's [15] QUICK scheme. The normalised face value for the CICSAM differencing scheme is calculated by combining these two schemes through a weighting factor derived from the orientation of the interface to the direction of motion. The method is derived for arbitrary meshes and so a direction split approach cannot be taken. Instead Ubbink [1] utilises an implicit multi-dimensional implementation that requires only one sweep through the mesh.

In the CICSAM scheme, the cell face values of the volume fraction used in the discretised volume fraction equation are determined from a combination of the CBC value given by

$$\tilde{C}_{\mathrm{fCBC}} = \begin{cases} \min\left\{1, \frac{\tilde{C}_{\mathrm{D}}}{c_{\mathrm{D}}}\right\}, & \text{when } 0 \leqslant \tilde{C}_{\mathrm{D}} \leqslant 1, \\ \tilde{C}_{\mathrm{D}}, & \text{when } \tilde{C}_{\mathrm{D}} < 0,\ \tilde{C}_{\mathrm{D}} > 1. \end{cases} \tag{23}$$

And the UQ value given by

$$\tilde{C}_{\mathrm{fUQ}} = \begin{cases} \min\left\{\frac{8c_{\mathrm{D}}\tilde{C}_{\mathrm{D}} + (1-c_{\mathrm{D}})(6\tilde{C}_{\mathrm{D}}+3)}{8}, \tilde{C}_{\mathrm{fCBC}}\right\}, & \text{when } 0 \leqslant \tilde{C}_{\mathrm{D}} \leqslant 1, \\ \tilde{C}_{\mathrm{D}}, & \text{when } \tilde{C}_{\mathrm{D}} < 0,\ \tilde{C}_{\mathrm{D}} > 1. \end{cases} \tag{24}$$

Here $\tilde{C}_{\mathrm{D}}$ is the normalised variable for the donor cell, calculated from

$$\tilde{C}_{\mathrm{D}} = \frac{C_{\mathrm{D}} - C_{\mathrm{U}}}{C_{\mathrm{A}} - C_{\mathrm{U}}}, \tag{25}$$

where subscript U indicates the upwind cell, A the acceptor and D the donor cell. These are determined depending on the velocity at a given face. With reference to Fig. 1, when considering the east face of cell C, if $u_{\mathrm{e}} > 0$ then the donor is cell C and cell E is the acceptor; however if $u_{\mathrm{e}} < 0$, then cell E is the donor and cell C the acceptor. The Courant number, c_{D}, is calculated by summing the fluxes over each cell face

$$c_{\mathrm{D}} = \sum_{f=1}^{n} \max\left\{\frac{-F_f \delta t}{V_{\mathrm{D}}}, 0\right\}, \tag{26}$$

where n is the number of faces, F_f is the volumetric flux across a given face calculated from the product of the face velocity and face area, δt is the time step and V_{D} the cell volume. The weighting factor used to combine the CBC and UQ contributions takes into account the orientation of the interface and the direction of motion

$$\gamma_f = \min\left\{k_\gamma \frac{\cos(2\theta_k) + 1}{2}, 1\right\}, \tag{27}$$

where θ_k is the angle calculated in (9) above and k_γ is a constant introduced to control the dominance of the different schemes. Ubbink [1] recommends a value of $k_\gamma = 1$. The normalised face value for the CICSAM differencing scheme is then

$$\tilde{C}_f = \gamma_f \tilde{C}_{\mathrm{fCBC}} + (1 - \gamma_f)\tilde{C}_{\mathrm{fUQ}}. \tag{28}$$

The weighting factor is given by

$$\beta_f = \frac{\tilde{C}_f - \tilde{C}_{\mathrm{D}}}{1 - \tilde{C}_{\mathrm{D}}}, \tag{29}$$

and the face values for the new volume fraction distribution used to solve the discrete volume fraction equation are

$$C_f^* = (1 - \beta_f)\frac{C_{\mathrm{D}}^t + C_{\mathrm{D}}^{t+\delta t}}{2} + \beta_f \frac{C_{\mathrm{A}}^t + C_{\mathrm{A}}^{t+\delta t}}{2}. \tag{30}$$

The superscripts refer to the time level, thus the volume fraction field is advected using a Crank–Nicolson scheme. In some rare cases it is possible for these volume fraction values to have non-physical values, less than 0 or greater than 1. In the event of this occurring, Ubbink [1] recommends a corrector step that corrects the weighting factor β_f. If a negative volume fraction value occurs, this implies that more of fluid 1 has left the donor cell than is available in it. If this occurs the amount of fluid to be convected is reduced by the unboundedness error so that the donor cell is left with a volume fraction equal to zero. When the volume fraction value is greater than 1 this indicates that too much fluid has been convected into the acceptor cell and in this case the amount convected is reduced so that the acceptor cell has a volume fraction value of 1.

3. Quadtree grid generation

Hierarchical grid generation is conceptually easy. The grid is created about a set of discrete seeding points by recursive subdivision of a simple geometric shape that at the root level surrounds the flow domain. Rectangular grids may be generated recursively using the quadtree algorithm, hexahedral grids using the octree algorithm and triangular grids using the tri-tree algorithm. An advantage of hierarchical grids is that they may be stored in a concise grid cell reference numbering system, which contains all the grid information and forms a tree data structure. The data tree can be traversed according to simple rules to obtain grid cell reference numbers of ancestor and neighbour cells: also the grids may be readily adapted by addition or subtraction of grid cells whilst maintaining the overall tree structure.

Samet [16,17] describes the quadtree data structure and its application to spatial data problems. Although quadtree algorithms were first used in image processing, they have also been used extensively as mesh generators. Yerry and Shephard [18] applied quadtree algorithms in creating finite element meshes for structural analysis, as did Messaoud [19] for elliptic partial differential systems. Van Dommelen and Rundensteiner [20] modelled flow past a cylinder using a discrete vortex scheme, with adaptive remeshing based on quadtrees. Multigrid-quadtree meshes have been applied [21] to the solution of species transport and linearised shallow flow problems in complex domains. Quadtree finite element methods using quadrilateral and cubic elements have been used for compressible flows [22] and quadtree finite volume methods for solutions of the Euler equations [23]. In addition, octree grids have been used as a basis for 3-D tetrahedral finite element mesh generation [24].

Greaves and Borthwick [25,26] demonstrated hierarchical grid generation in two dimensions for quadtree grids and in three dimensions for octree grids. Hu et al. [27] presented the use of unstructured triangular tri-tree grids for adaptive solution of Navier–Stokes equations using a finite volume method for arbitrary grids.

3.1. The quadtree algorithm

The quadtree algorithm can be summarised as follows:

(1) Define the set of seeding points, P_n, about which the grid will be generated. The seeding points lie along the interface and their spacing will determine the resolution of the quadtree grid.
(2) Define the unit square or rectangle (root panel) which surrounds the normalised domain of interest.
(3) Divide the root panel into four quadrant panels.
(4) Consider each panel; if the panel contains more than two points, continue with (5), otherwise check the next panel.
(5) Check whether the maximum division level, $M_{\max}$, has been reached. If so, the division of the panel in question is complete, so go to (4) and check the next panel. When all panels considered either have reached the maximum division, $M_{\max}$, or contain less than three points, the mesh generation is complete. Otherwise continue.
(6) Divide the panel into four panels, return to (4) and check the next panel.

Additional panels are generated to regularise the grid such that the maximum panel edge length ratio between two adjacent panels does not exceed two. This will limit the variation of neighbour arrangements encountered when solving discretised equations on the grid. Fig. 4 shows the quadtree grid generated for a circular interface with radius equal to 0.125 centred at $(-0.2, 0.2)$, the origin is at the centre of the root cell. The quadtree grid has a maximum division level equal to 7 and minimum division level equal to 2. In Fig. 4(a) the grid is not regularised and the large difference in size of adjacent cells would make solution of the discrete equations very difficult. In Fig. 4(b) the grid is regularised to limit the edge length ratio; whilst still maintaining a large difference in cell size throughout the grid, adjacent cells are no more than twice the size of one another. A quadtree panel in a regularised grid has eight possible neighbours of the same size and 12 possible neighbours of one level smaller in size.

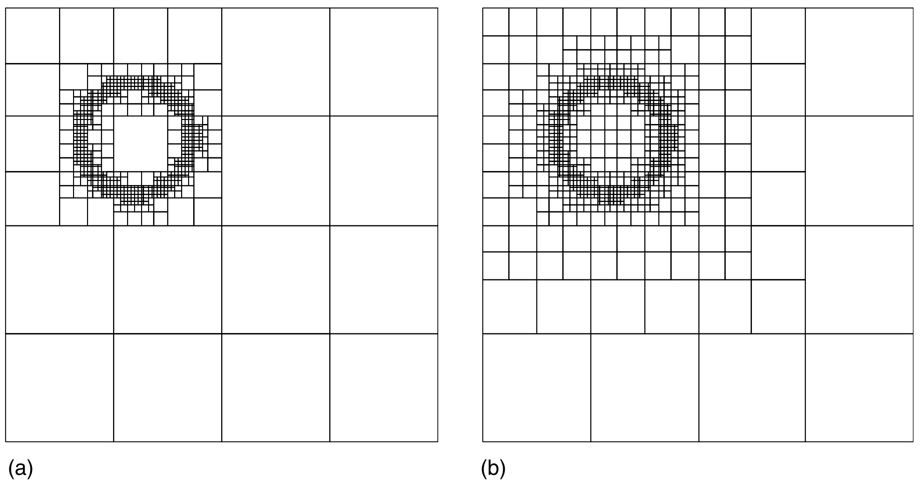

Fig. 4. (a) Quadtree grid without regularisation and (b) regularised quadtree grid.

3.2. The quadtree numbering system

Several numbering systems are possible for storing the quadtree information. Here, the numbering system is essentially due to Van Dommelen and Rundensteiner [20] and enables the tree reference numbers to be stored as an array of binary digits. The reference number, N, for each panel is made up of a sequence of successive binary translations at each generation level and is stored in four arrays of eight digits. The reference number contains information about the panel's location within the tree, and by manipulation leads to neighbour finding within the grid. Full details of the numbering system used here and its manipulation to obtain grid data are described by Greaves [28].

The reference number of a given panel can be regarded as a list of successive orientations NW, SW, NE or SE corresponding to binary translations 00, 01, 10 and 11, which describe the position of the panel within its parent. If the large square in Fig. 5 is the root of the tree, then the four smaller squares are the children of the root cell, created at the first division. The significant part of the reference number for the NE child is given by, $N(I, 1) = 21$ (note that +1 has been added to each significant digit in order to distinguish it from the trailing zeros). Whenever division takes place, four panels are produced having different x and y translations from the centre of the parent cell. These digits correspond to locations NW, SW, NE or SE and are appended to the number of the parent panel being divided to produce the reference number of each child or new panel. Hence the reference numbers of the parent and all ancestor panels are contained within the reference number of a given panel.

3.2.1. Generation level

The number of divisions carried out to produce a panel of given size is called the generation (or division) level of the panel, M. Clearly, all panels of the same size have the same generation level. The reference number is extended by two digits at each division and so the generation level, M, is equal to the number of non-zero digits divided by two.

3.2.2. Parent panel

The reference number of the parent panel of a given panel is found by converting the last two non-zero digits to zero.

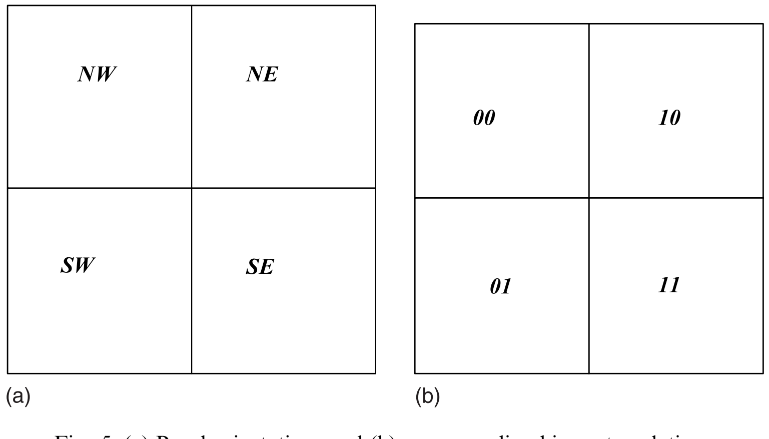

Fig. 5. (a) Panel orientations and (b) corresponding binary translations.

3.2.3. Centre coordinates

The x and y coordinates at the centre of the panel are calculated by successive translations at each level equal to half of the panel edge length, starting from the centre coordinates of the root panel. At each level, M, the panel edge length is equal to 2^{-M}. This calculation is expressed as

$$x = x_{\text{root}} + \sum_{m=1}^{M} \frac{1}{2^{m+1}}(2x(m) - 1) \tag{31}$$

and

$$y = y_{\text{root}} + \sum_{m=1}^{M} \frac{1}{2^{m+1}}(1 - 2\mathrm{y}(m)), \tag{32}$$

where $x(m)$ and $y(m)$ are the x and y direction binary translations at level m.

3.2.4. Neighbour finding

All possible neighbours of any panel can be determined from its reference number. The tree is searched to locate the required neighbour starting from the nearest common ancestor, numbered NCA, of a given panel and its neighbour and ending when a leaf (undivided) panel is reached. The reference number of the NCA shared by two panels is given by the binary digits that are common to both panels. Reference numbers increase down through the generations of a family, so that none of the descendants of a given panel can have a reference number greater than that of the panel's next sibling. Thus, as the search progresses down through the tree at each level, of the two branches between which the reference number of the required panel lies, the branch having the lower reference number is taken. The manipulation of the numbering system to obtain neighbour references is described in detail by Greaves [28].

3.3. Adaptive grid scheme

An advantage of quadtree grids is that they can be readily adapted by the addition and removal of cells throughout a time dependent simulation. In this work, grid refinement is used to follow the movement of the interface. Remeshing of the grid operates by dividing a cell into four if it lies on the interface, i.e. if the value of C lies between 0 and 1. Derefinement also takes place by removing four sibling cells and replacing them with their parent. This only occurs where each of the four sibling cells lies away from the interface and the volume fraction value of each is equal to 0 or 1.

Interpolation and extrapolation of the volume fraction, C, is achieved using PLIC reconstruction. PLIC reconstruction of the interface in the divided cell is transformed to the coordinates of each newly created cell and the volume fraction determined from the equation of the line in each new cell. Fig. 6 shows the interface line segment ab in original cell 0, which after cell refinement lies in new cells 1, 2 and 4 only. The cell height and width in (14) become $\delta x/2, \delta y/2$ (where δx and δy are the dimensions of the parent cell) and the parameter α is transformed as follows for each of the new cells:

$$\begin{aligned}
&\text{Cell 1}: \quad \alpha^* = \alpha - m_y\delta y/2,\\
&\text{Cell 2}: \quad \alpha^* = \alpha,\\
&\text{Cell 3}: \quad \alpha^* = \alpha - m_y\delta y/2 - m_x\delta x/2,\\
&\text{Cell 4}: \quad \alpha^* = \alpha - m_x\delta x/2.
\end{aligned} \tag{33}$$

The grid is only derefined away from the interface, so the volume fraction for each of the removed siblings and their parent will be either 0 or 1.

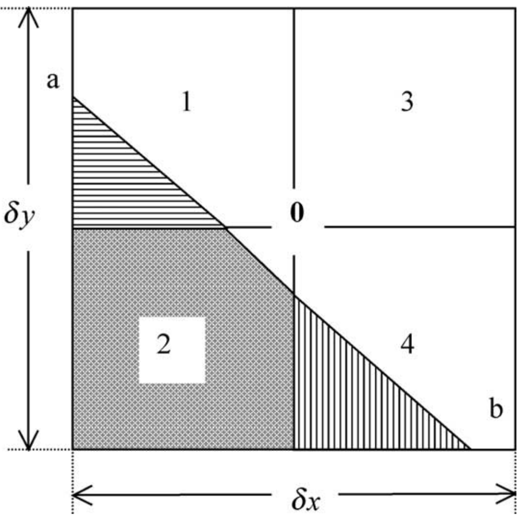

Fig. 6. The shaded areas show the volume fraction of fluid in new cells 1, 2 and 4 after refinement of cell 0.

3.4. Hanging node treatment

Hanging nodes are inherent in quadtree grids and occur at the centre of a cell face where cells of different size meet. In order to conserve fluxes, contributions from all cells neighbouring a given face are used to calculate the fluxes across the face. For example, Eq. (4) applied to cell P in Fig. 7 is

$$C^{n+1} = C^n + f_{e1} + f_{e2} - f_w + f_n - f_s, \tag{34}$$

where f_k is the flux of C across the k-direction boundary.

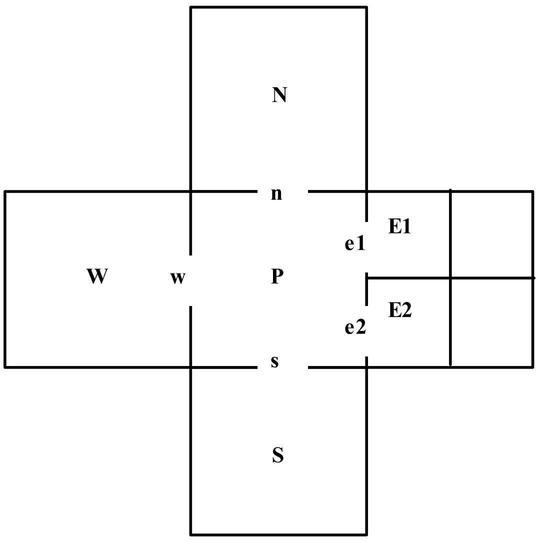

Fig. 7. Hanging node treatment for fluxes.

4. Results

In this section a series of tests are reported to assess the CFD methodology for capturing interfaces between two immiscible fluids. The tests are designed to investigate the accuracy of convection procedures, interface capturing methodologies and grid adaptation. They involve uniform density flow calculations free from gravitational forces and surface tension is not included. The tests involve tracking the progress of fluid bubbles placed in fluid of the same density in a velocity field of uniform, rotating and shear flow. In the case of the uniform and rotating flow, the bubble should be convected through the grid without changing shape. In the case of shearing flow, the bubble will be distorted, but if the velocity is then reversed and the bubble convected back through the same number of time steps, it should return to its original shape.

4.1. Comparison of interface tracking schemes

The first series of tests involve translating circular and square interfaces through a uniform velocity field. The test cases are calculated using the VoF schemes SURFER, PLIC, CICSAM and the accuracy of each method is compared. A regular grid size of 128×128 in a unit square domain is used and the initial contours for the circular interface, positioned at the centre of the grid, are shown in Fig. 8(a). The calculation domain is a unit square and the diameter of the circle, initially placed at the centre of the domain, is 0.01. The circle is translated in a constant velocity field, $u = 1,\ v = -1$ and the time step is determined such that the Courant number is equal to 0.125. In each case, the results are presented by plotting volume fraction contours at $C = 0.05$, $C = 0.4$, $C = 0.6$ and $C = 0.95$. With an accurate interface advection scheme, the interface should be translated intact towards the bottom right-hand corner of the domain. Figs. 8(b)–(d) show the interface at time $t = 0.375$ s calculated using SURFER, PLIC and CICSAM.

The interface translated using SURFER interface tracking is clearly distorted; the results obtained for PLIC and CICSAM are much better and similar to one another, although the original interface is better maintained with CICSAM. These findings are confirmed by calculating the error, which for a grid of n cells is calculated as

$$\text{error} = \left| \frac{\sum_{i=1}^{n} C_i^{\text{final}} \delta x\, \delta y - \sum_{i=1}^{n} C_i^{\text{initial}} \delta x\, \delta y}{\sum_{i=1}^{n} C_i^{\text{initial}} \delta x\, \delta y} \right|, \tag{35}$$

and is summarised for the circle and square translation cases in Table 1. The total volume of fluid should remain constant and so the sum of the volume fraction value multiplied by the cell size over the entire grid should remain constant. The error calculates the normalised difference between the initial and final summed volume of fluid.

Similar results generated for a square interface, size 0.125×0.125, are shown in Fig. 9. Other details are the same as for the circle described above. Here, the sharpness of the corners deteriorates if the interface is smeared over too many cells. As above, the interface calculated using SURFER is clearly distorted, whereas the PLIC and CICSAM schemes both maintain the interface shape well. The errors summarised in Table 1 confirm that CICSAM gives the most accurate solution for interface tracking in translation.

4.2. Comparison of uniform and adapting quadtree schemes

Uniform translation of a square interface is then investigated for the PLIC and CICSAM schemes on uniform and quadtree adapted grids in order to assess the benefits of quadtree grid adaptation.

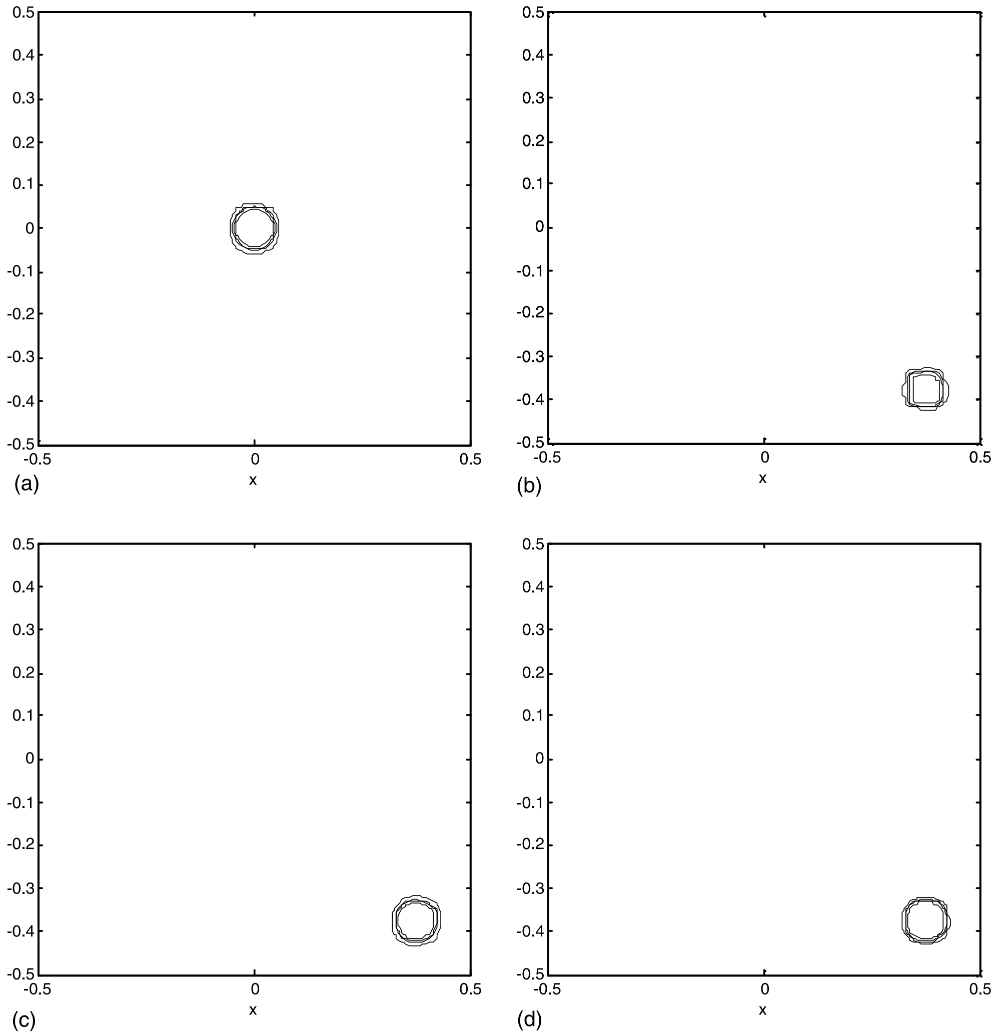

Fig. 8. Advection of circular interface, contours of C plotted at $C = 0.05$, $C = 0.4$, $C = 0.6$ and $C = 0.95$. (a) Initial interface contours, $t = 0$ s, (b) SURFER, $t = 0.375$ s, (c) PLIC, $t = 0.375$ s and (d) CICSAM, $t = 0.375$ s.

Table 1
Comparison of interface tracking schemes

		Error
Translation of circle	SURFER	9.778e^{-4}
	PLIC	1.387e^{-4}
	CICSAM	5.039e^{-6}
Translation of square	SURFER	1.256e^{-3}
	PLIC	2.775e^{-4}
	CICSAM	1.189e^{-5}

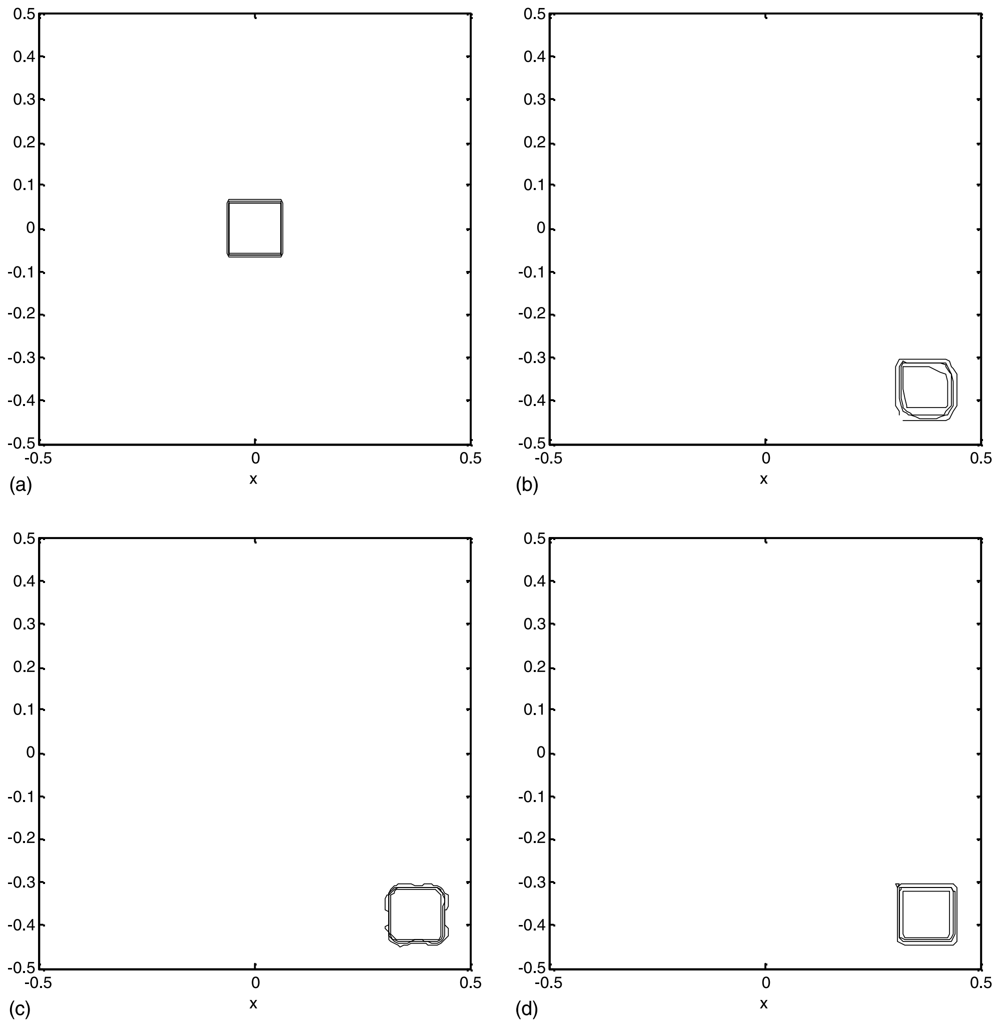

Fig. 9. Advection of square interface, contours of C plotted at $C = 0.05$, $C = 0.4$, $C = 0.6$ and $C = 0.95$. (a) Initial interface contours, $t = 0$ s, (b) SURFER VoF, $t = 0.375$ s, (c) PLIC, $t = 0.375$ s and (d) CICSAM, $t = 0.375$ s.

Here, the quadtree grids have a maximum division level equal to 7 and minimum division level equal to 5. The adapting quadtree grid results are shown in Fig. 10. The volume fraction contours and adapted grid at $t = 0.375$ s calculated using the PLIC scheme are given in Figs. 10(a) and (b), and those calculated using the CICSAM scheme are shown in Figs. 10(c) and (d). Comparing the volume fraction contours in Fig. 10 with those presented in Fig. 9, and by consulting Table 2, it is clear that the adapted grid achieves a solution of very similar accuracy to that calculated on the uniform grid. Table 2 lists the errors calculated using (37). The adapted grid contains typically 1500 cells compared with the uniform grid, which contains 16,384 cells; the CPU time for each calculation is also listed in Table 2 and clearly use of the adapted grid makes a considerable saving in both computer space and time. Using the CICSAM scheme, the quadtree adapted grid size is more than ten times less than the

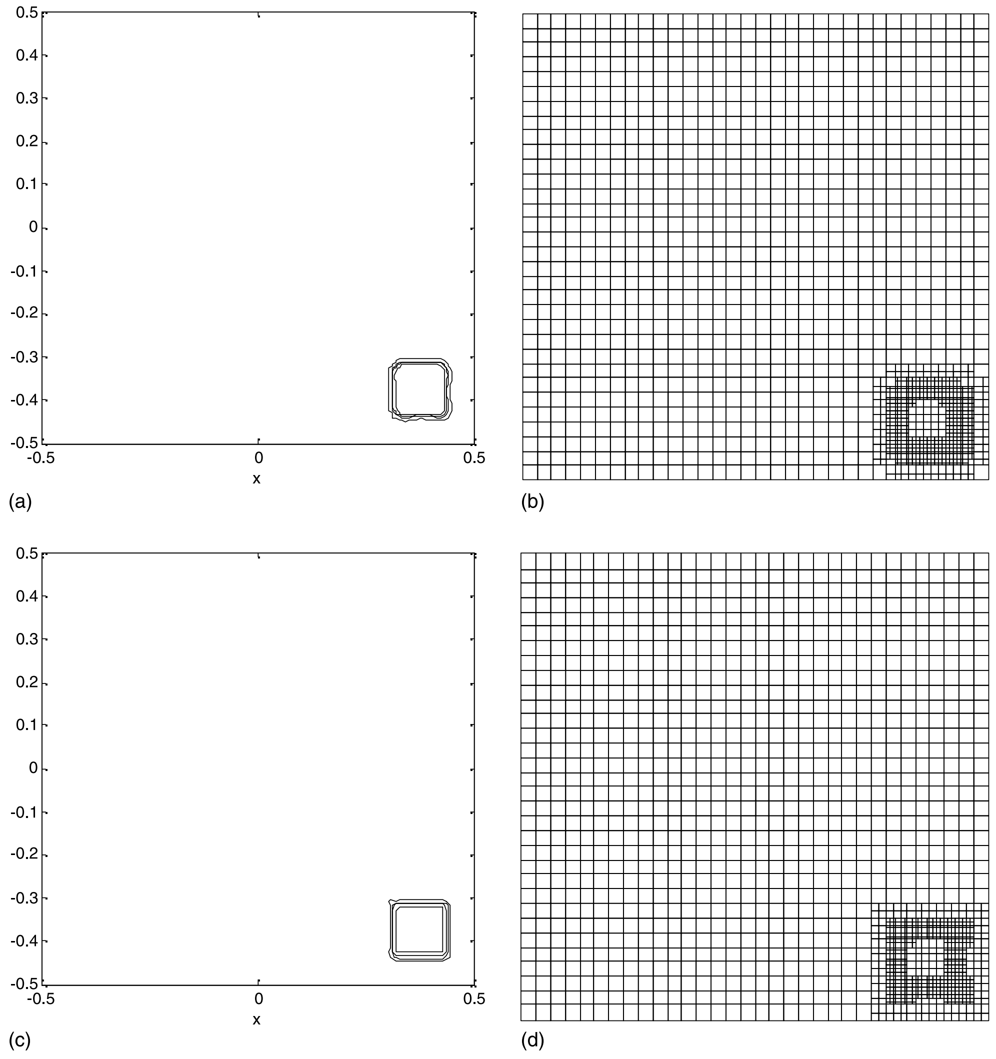

Fig. 10. Advection of square interface on adaptive quadtree grids. (a) PLIC *C* contours, (b) PLIC adapted grid, (c) CICSAM *C* contours and (d) CICSAM adapted grid.

Table 2
Comparison of adapting quadtree with uniform grids

		Error	CPU (s)
Translation of square	PLIC uniform grid	$2.775e^{-4}$	291.56
	PLIC adapted grid	$1.115e^{-4}$	75.99
	CICSAM uniform grid	$1.189e^{-5}$	368.76
	CICSAM adapted grid	$1.178e^{-5}$	77.18

uniform grid for similar accuracy and the CPU used is nearly five times less than for the uniform grid calculation.

4.3. Rotation of square interface

In this example, the square interface is advected in a rotating flow, defined by $u = ru_{\max}\cos\theta$, $v = rv_{\max}\sin\theta$, where $u_{\max} = 1.0$ and $v_{\max} = -1.0$ and the radius at a given point in the grid is measured from the centre of the grid. In this case, the square interface is investigated as this is most likely to be distorted. The square is initially positioned at the centre of the grid and the centre of rotation is also at the centre of the grid. The results for PLIC and CICSAM calculated on quadtree adaptive grids (maximum level 7 and minimum level 5) are given in Fig. 11.

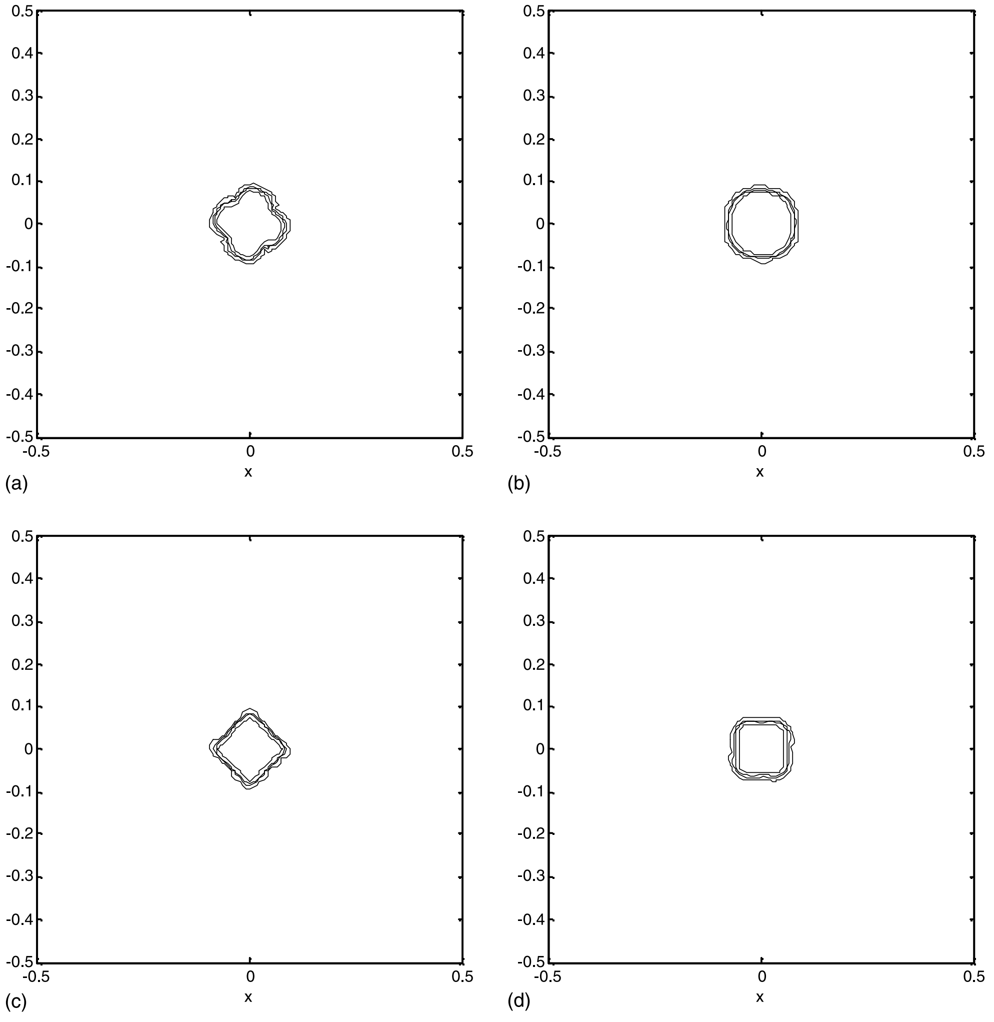

Fig. 11. Rotation of a square interface. (a) PLIC C contours 45° rotation, (b) PLIC C contours 360° rotation, (c) CICSAM C contours 45° rotation and (d) CICSAM C contours 360° rotation.

For the PLIC scheme, the interface is badly smeared at the corners and, after a full revolution of 360°, the originally square interface looks more like a circle. The quadtree CICSAM scheme maintains a sharp interface, although there is some rounding of the corners compared with the translation case. Most of the detail is still sharply defined, however, a finer maximum grid resolution would be required to improve the corner definition and is used for the simulation of a slotted circle in rotating flow described in Section 4.4.

4.4. Rotation of slotted circle

In this case a slotted circular interface is positioned in a rotating velocity field. The circle diameter is 0.25 and its centre is initially positioned at $x = 0.0$, $y = 0.1875$. There is a vertical slot in the bottom of the circle of width 0.03 and height 0.15. The interface is advected using CICSAM differencing on adapting quadtree grids of maximum level 8 and minimum level 5. A finer quadtree resolution is used in this case to aid clear definition of the slot. Results are presented in Fig. 12. Figs. 12(a) and (b) show the initial volume fraction contours and the initial quadtree grid at $t = 0$ s. The volume fraction contours and adapted grid plotted after 180° rotation are shown in Figs. 12(c) and (d) and after 360° rotation are shown in Figs. 12(e) and (f). This case was investigated both by Ubbink and Issa [26] and by Rudman [27].

It is clear from Fig. 12(e) that very little definition has been lost even after 360° rotation. There is some slight rounding of the sharp corners of the slot, but the contours have not spread visibly. Smearing of the interface has been prevented and computational effort minimised by using high resolution adapting quadtree grids.

4.5. Shear flow

A more stringent test for the advection scheme is to operate in a shearing flow field. In this case, also investigated by Ubbink and Issa [29] and by Rudman [30], the velocity is prescribed to be $u = u_{\max} \sin\theta \cos\theta, v = -v_{\max} \cos\theta \sin\theta$, where $u_{\max} = 1.0$ and $v_{\max} = -1.0$. In most real interfacial flow cases, the interface is moving under the influence of fluid shear and the interface deforms considerably. Here, an adapting quadtree grid is used with maximum division 7 and minimum division 5, and the circle of diameter 0.36 is initially positioned at $x = 0.0$, $y = -0.2$. The interface is first advected forward for a given length of time and then the velocities are reversed for the same length of time in order to return the volume fraction field to the initial condition. A perfect advection scheme should result in the same initial volume fraction field.

The results of this test are presented in Fig. 13. The interface is initially advected up to $t = 5.0$ and then up to 10.0 s. Each time the velocities are reversed to return the volume fraction field to its initial condition. In this case the reconstructed interface is plotted rather than the volume fraction contours. In Figs. 13(a) and (b), the initial interface and initial quadtree grids are shown. The grid also has the velocity vectors (scaled by 0.02) superimposed. Figs. 13(c) and (d) show the interface and adapted grid at $t = 5.0$ s and Figs. 13(e) and (f) show the resulting interface and grid after the velocity has been reversed and the interface advected for 5 s in the opposite direction. Similarly, Figs. 13(g)–(j) show the results at $t = 10.0$ s and after stepping back to $t = 0$ s with velocities reversed.

After reversing from $t = 5.0$ s, the resulting circular interface is very close to the initial circular interface, despite being highly distorted at $t = 5.0$ s. At $t = 10.0$ s, the interface has completed nearly two full revolutions, is distorted into a spiral and beginning to break up at the tail end. Even so, when tracked back for a further 10 s with velocities reversed, the resulting interface is close in appearance to the original circle.

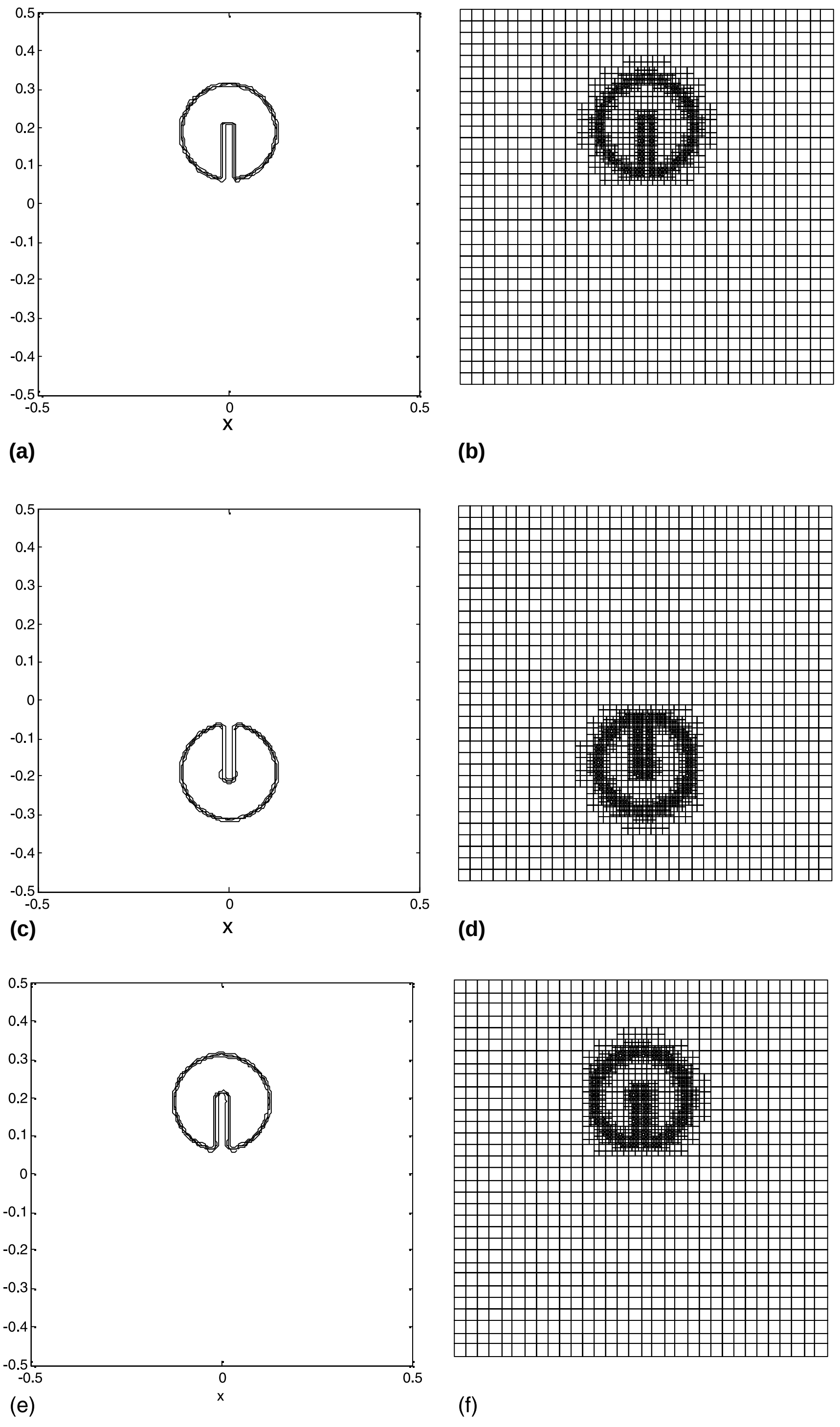

Fig. 12. Rotation of a slotted circle. (a) Initial volume fraction contours, (b) initial grid, (c) C contours after 180° rotation, (d) adapted grid after 180° rotation, (e) C contours after 360° rotation and (f) adapted grid after 360° rotation.

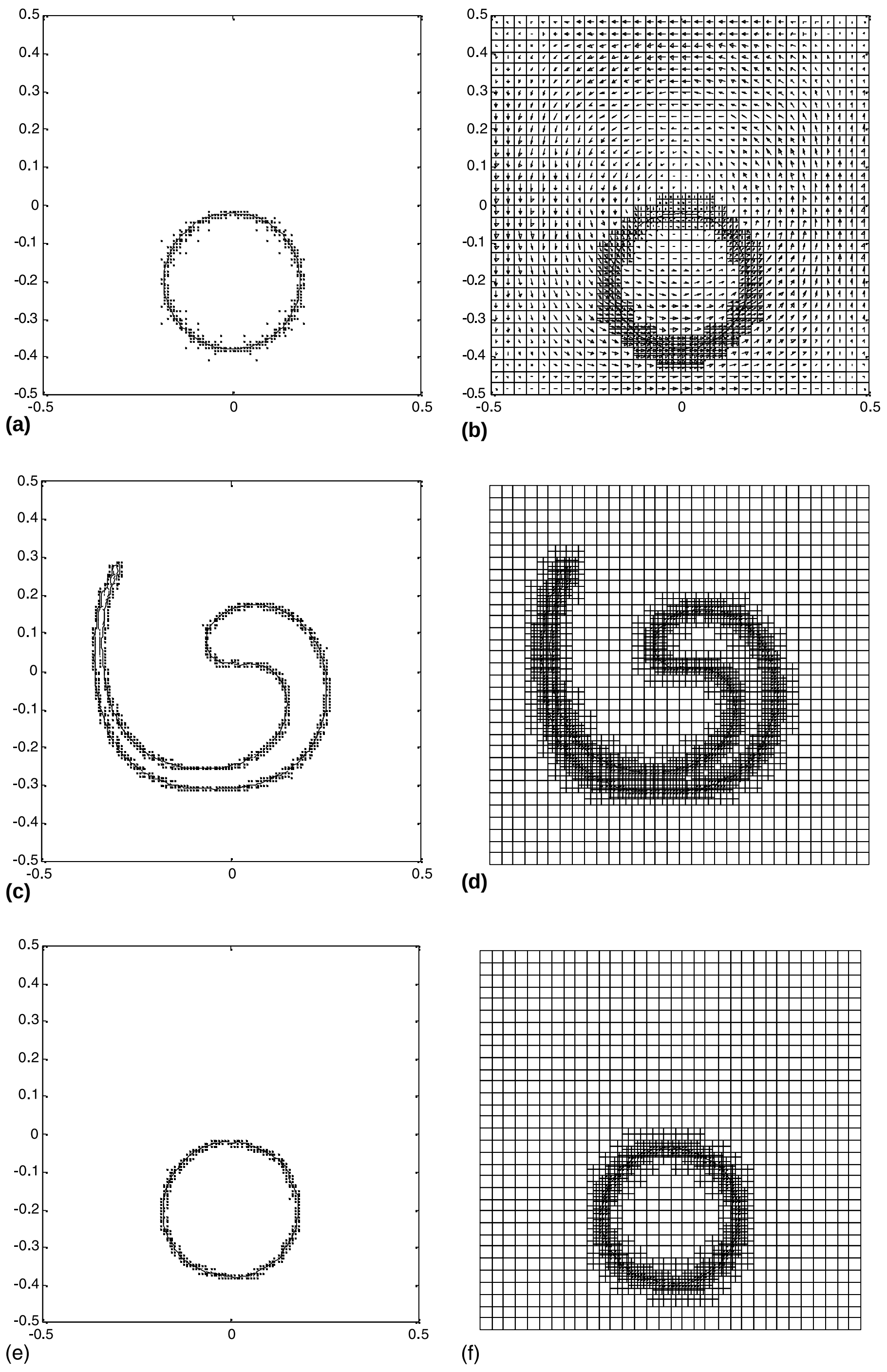

Fig. 13. Circular interface in shear flow. (a) Interface at $t = 0.0$ s, (b) quadtree grid at $t = 0.0$ s, (c) interface at $t = 5.0$ s, (d) adapted grid at $t = 5.0$ s, (e) interface after reversing from $t = 5.0$ to $t = 0.0$ s, (f) adapted grid after reversing from $t = 5.0$ to $t = 0.0$ s, (g) interface at $t = 10.0$ s, (h) adapted grid at $t = 10.0$ s, (i) interface after reversing from $t = 10.0$ to $t = 0.0$ s and (j) adapted grid after reversing from $t = 10.0$ to $t = 0.0$ s.

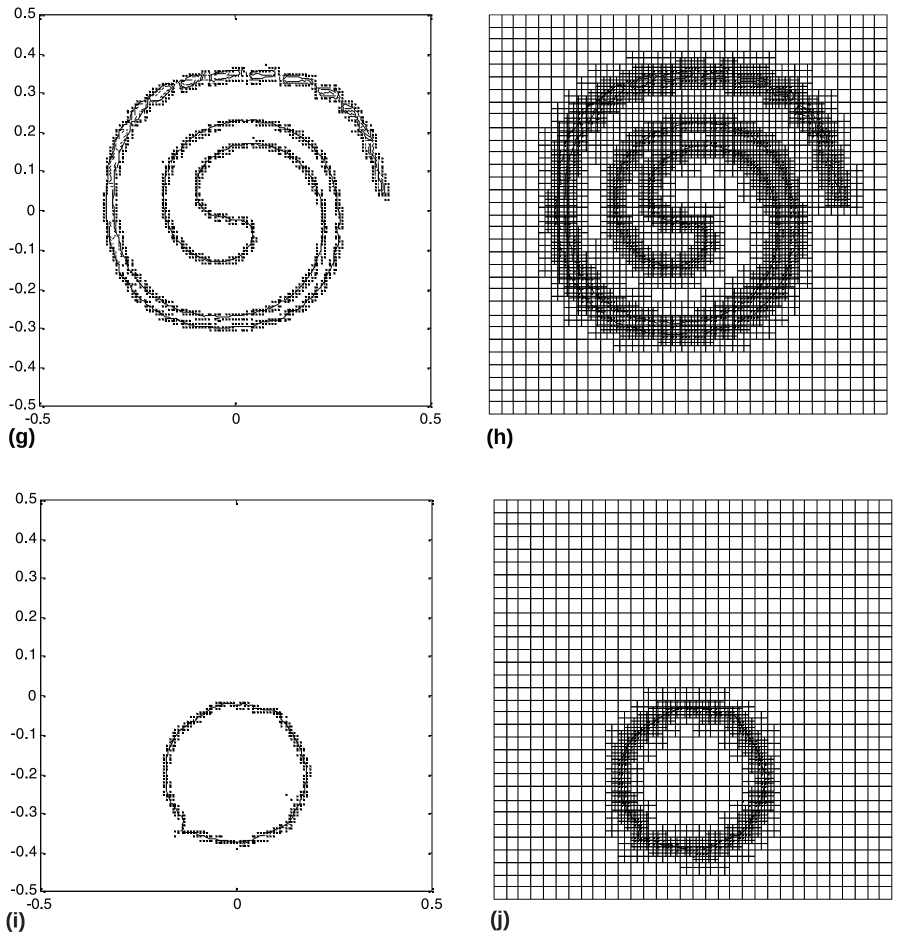

Fig. 13. (*continued*)

5. Conclusions

An adaptive quadtree VoF method has been developed, which uses CICSAM differencing for advection of the interface and PLIC reconstruction for interpolation of the volume fraction as the grid adapts. Three separate interface tracking schemes are investigated and compared before selecting CICSAM, which is demonstrated to be the most accurate. Both PLIC and CICSAM schemes are combined with adaptive quadtree interface tracking and the results compared with those from uniform grid calculations. Quadtree adaptation is shown to provide a significant saving in grid size and computation time. A typical quadtree grid size used here is approximately 10 times less than the equivalent uniform grid and the computation time approximately five times less. The new quadtree-based interface tracking scheme is also applied to more complex interface motion simulations and found to combine successfully high accuracy with sharp definition of the interface even for highly sheared flows.

Acknowledgements

The author is most grateful to the Royal Society for funding this work.

References

[1] O. Ubbink, Numerical prediction of two fluid systems with sharp interfaces, PhD Thesis, Imperial College of Science, Technology and Medicine, London, 1997.
[2] J.M. Hyman, Numerical methods for tracking interfaces, Physica 12D (1984) 396–407.
[3] E.D. Wilkes, S.D. Phillips, O.A. Basaran, Computational and experimental analysis of dynamics of drop formation, Phys. Fluids 11 (12) (1999) 3577–3598.
[4] M.S. Carvalho, L.E. Scriven, Three-dimensional stability analysis of free surface flows: application to forward deformable roll coating, J. Comput. Phys. 151 (1999) 534–562.
[5] D.M. Greaves, A.G.L. Borthwick, G.X. Wu, R. Eatock Taylor, A moving boundary finite element method for fully non-linear wave simulations, J. Ship Res. 41 (1997) 181–194.
[6] D.M. Causon, An efficient front tracking algorithm for multi-component fluid calculations with biomedical applications, Zeit. Angew. Math. Mech. 76 (S1) (1996) 371–372.
[7] C.W. Hirt, B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[8] P. Heinrich, Nonlinear numerical model of landslide-generated water waves, Int. J. Engrg. Fluid Mech. 4 (4) (1991) 403–416.
[9] A. Tomiyama, A. Sou, H. Minagawa, T. Sakaguchi, Numerical analysis of a single bubble by VoF method, JSME Int. J, Ser. B 36 (1993) 51–56.
[10] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with SURFER, J. Comput. Phys. 113 (1994) 134–147.
[11] W.F. Noh, P. Woodward, SLIC (Simple Line Interface Calculations), Lect. Notes Phys. 59 (1976) 330–340.
[12] N. Ashgriz, J.Y. Poo, FLAIR: flux line-segment model for advection and interface reconstruction, J. Comput. Phys. 93 (1991) 449–468.
[13] D. Gueyffier, J. Li, A. Nadim, R. Scardovelli, S. Zaleski, Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows, J. Comput. Phys. 152 (1999) 423–456.
[14] B.P. Leonard, The ULTIMATE conservative difference scheme applied to steady one-dimensional advection, Comput. Methods Appl. Mech. Engrg. 88 (1991) 17–74.
[15] B.P. Leonard, A stable and accurate convective modelling procedure based on quadratic upstream interpolation, Comput. Methods Appl. Mech. Engrg. 19 (1979) 59–98.
[16] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, Reading, MA, 1990.
[17] H. Samet, Applications of Spatial Data Structures, Addison-Wesley, Reading, MA, 1990.
[18] M.A. Yerry, M.S. Shephard, A modified quadtree approach to finite element mesh generation, IEEE Comput. Graph. Appl. 3 (1) (1983) 39–46.
[19] B. Messaoud, Parallel and Adaptive Algorithms for Elliptic Partial Differential Systems, PhD Rensselaer Polytechnic Institute, Troy, NY, 1992.
[20] L. van Dommelen, E.A. Rundensteiner, Fast, adaptive summation of point forces in the two-dimensional Poisson equation, J. Comput. Phys. 83 (1989) 126–147.
[21] J. Józsa, C. Gáspár, Fast, adaptive approximation of wind-induced horizontal flow patterns in shallow lakes using quadtree-based multigrid method, C.M.W.R. Denver, 1992.
[22] D.P. Young, R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Samant, J.E. Bussoletti, A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics, J. Comput. Phys. 92 (1991) 1–66.
[23] D. de Zeeuw, K.G. Powell, An adaptively refined cartesian mesh solver for the Euler equations, J. Comput. Phys. 104 (1993) 56–68.
[24] M.S. Shephard, M.K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, Int. J. Numer. Methods Engrg. 32 (1991) 709–749.
[25] D.M. Greaves, A.G.L. Borthwick, On the use of adaptive hierarchical meshes for numerical simulation of separated flows, Int. J. Numer. Methods Fluids 26 (1998) 303–322.
[26] D.M. Greaves, A.G.L. Borthwick, Hierarchical tree-based finite element mesh generation", Int. J. Numer. Methods Engrg. 45 (1999) 447–471.
[27] Z.Z. Hu, D.M. Greaves, G.X. Wu, Numerical simulation of fluid flows using an unstructured finite volume method with adaptive tri-tree grids, Int. J. Numer. Methods Fluids 39 (2002) 403–440.
[28] D.M. Greaves, Numerical modelling of laminar separated flows and inviscid steep waves using adaptive hierarchical meshes, DPhil Thesis, Oxford University, 1995.
[29] O. Ubbink, R.I. Issa, A method for capturing sharp fluid interfaces on arbitrary meshes, J. Comput. Phys. 153 (1999) 26–50.
[30] M. Rudman, Volume-tracking methods for interfacial flow calculations, Int. J. Numer. Methods Fluids 24 (1997) 671–691.