INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng (in press)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.2079

Simple finite element-based computation of distance functions in unstructured grids

Renato N. Elias, Marcos A. D. Martins 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, Rio de Janeiro, RJ 21945-970, Brazil

SUMMARY

A distance field is a representation of the closest distance from a point to a given surface. Distance fields are widely used in applications ranging from computer vision, physics and computer graphics and have been the subject of research of many authors in the last decade. Most of the methods for computing distance fields are devoted to Cartesian grids while little attention has been paid to unstructured grids. Finite element methods are well known for their ability to deal with partial differential equations in unstructured grids. Therefore, we propose an extension of the fast marching method for computing a distance field in a finite element context employing the element interpolation to hold the Eikonal property ($\|\nabla\varphi\| = 1$). A simple algorithm to develop the computations is also presented and its efficiency demonstrated through various unstructured grid examples. We observed that the presented algorithm has processing times proportional to the number of mesh nodes. Copyright © 2007 John Wiley & Sons, Ltd.

Received 12 April 2006; Revised 20 March 2007; Accepted 21 March 2007

KEY WORDS: distance function; finite elements; level set; Eikonal solvers; reinitialization

1. INTRODUCTION

The level set method is a technique that makes use of implicit interface representations for capturing its behavior within a velocity field. Since it was conceived, the level set method has shown its versatility in solving a wide variety of problems such as: image reconstruction, collision detection, shortest path computation, multiphase flow and others [1, 2]. In level set methods, a distance

*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, Rio de Janeiro, RJ 21945-970, Brazil.
†E-mail: alvaro@nacad.ufrj.br, alvaro@coc.ufrj.br

Contract/grant sponsor: Petroleum National Agency (ANP, Brazil)
Contract/grant sponsor: Center for Parallel Computations (NACAD)
Contract/grant sponsor: Brazilian Council of Technological and Scientific Development—CNPq; contract/grant number: CT-PETRO/PROSET 500.196/02-8

Copyright © 2007 John Wiley & Sons, Ltd.

function is generally employed as a geometric tool to initialize and keep a surface with some desirable properties such as smoothness and constant known gradient.

A distance function, also called distance field, can be defined as a scalar field representation where at each point within this field we know the distance from that point to the surface of an object [3]. Unfortunately, for practical problems, using discrete versions of the level set method, the distance field cannot be easily determined through analytical equations and numerical methods for computing distance functions must be applied.

The most straightforward and naïve way for computing distance fields is through the use of a geometric brute force algorithm where the point-to-point distance is computed throughout the computational grid and the minimum distance for each point is stored. However, brute force algorithms are infeasible due their high computational cost and the study of more efficient methods for computing distance functions is still an open research area. Furthermore, while most of the available methods for computing distance fields are developed for Cartesian grids, little attention has been paid to unstructured meshes. Some of the existing methods are based on the solution of partial differential equations while others are purely based on geometric calculations. We suggest Reference [3] for a recent survey about distance function methods in three dimensions, related mostly to computer graphics applications. Rouy and Tourin [4] proposed to numerically solve the interface transport equation where the origin, or level set 0, is employed as 'boundary' condition and the equation integrated in time for each direction, inward and outward, toward steady state. Sussman *et al*. [5] improved this method by considering it as an initial value problem and embedded the function signs to specify and transport the interface in both directions simultaneously. However, methods based on the solution of the hyperbolic transport equation have drawbacks associated with the difficulty of keeping the interface fixed during the time integration. Moreover, the method proposed by Sussman and co-workers also suffers when the initial data are steep near the interface and some kind of smoothness procedure must be applied to the initial condition [1]. Sussman and Fatemi [6] suggested an improvement to keep the interface fixed during the solution procedure. They noticed that for incompressible multiphase flow the amount of each phase must be preserved and proposed to add a correction term to the original equation.

The first work addressing triangulated versions of the level set method is due to Barth and Sethian [7]. In this work the authors considered a stabilized space–time Galerkin least-squares finite element method with shock capturing to solve the Hamilton–Jacobi and Eikonal equations. Some finite element works addressing the use of distance fields with level set methods have been recently published. Most of them employed some of the techniques previously cited to compute and keep a distance field function. Shepel and co-workers [8] presented a level set finite element implementation in two commercial codes to track interfaces. Nagrath *et al*. [9, 10] employed a stabilized finite element method to solve bubble dynamics problems through a level set approach. In these works the authors computed and held the distance fields in unstructured grids by applying finite element versions of the scheme proposed by Sussman and Fatemi [6], solving the level set equation by the Streamline-Upwind/Petrov–Galerkin (SUPG) formulation. Recent works propose other finite element methods to solve the level set equation, based either on an assumed gradient formulation [11] or a discontinuous Galerkin formulation [12]. Vector level sets have been used to describe crack propagation in [13, 14]. In this approach only nodal data are used to describe the crack; no geometrical entity is introduced for the crack trajectory, and no partial differential equations need to be solved to update the level sets. Belytschko *et al*. [15] developed a new algorithm for smoothing the surfaces in finite element formulations of contact impact. In their method smoothing is done implicitly by constructing signed distance functions for the interacting

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

bodies. These functions are then employed for computing variables needed for implementation of contact impact.

Sethian [16] employed the crossing-time idea to develop the fast marching method (or FMM for short). The FMM is a technique for computing the arrival time of a front, as e.g. a balloon, inflating in the normal direction at a set of grid points [3]. This is done by solving the Eikonal equation from a given boundary condition. The Eikonal equation is

$$\|\nabla T\| F = 1 \tag{1}$$

where $F \geqslant 0$ is the speed of the front, and T is the arrival time of the front. Given a point $\mathbf{p}$, the arrival time $T(\mathbf{p})$ is the time at which the skin of the balloon passed $\mathbf{p}$. However, if $F = 1$, the front moves at unit speed, and the arrival time at $\mathbf{p}$ is simply the distance from $\mathbf{p}$ to the closest point on the front at time 0 [3]. In order to advance the computational front, the FMM implementation employs Cartesian grids and an algorithm based on the use of min-heap lists (see Sedgewick [17] for details) and binary tree operations computing and advancing the solution of Equation (1) at each grid point while keeping the heap list updated. We refer the interested reader to [2] for more details about FMM.

Extensions of the basic FMM method have added capabilities to handle unstructured grids. First, Kimmel and Sethian [18] introduced a first-order scheme to unstructured acute triangulations in $\mathbb{R}^2$ and on two-dimensional surfaces, using the resulting technique to compute shortest path geodesics on triangulated manifolds. Later, Sethian and Vladimirsky [19] built second-order methods for arbitrary unstructured meshes in $\mathbb{R}^n$. Another variant have been recently proposed by Gross *et al*. [20] for quadratic tetrahedra. They have shown, that in the finite element simulation of two-phase incompressible flow with surface tension, FMM outperforms methods based on pseudo-time stepping of the Eikonal equation. Motivated by the nice properties observed in the Cartesian FMM and its unstructured grid extensions, we propose a novel method for computing distance functions in unstructured grids. Our main concern is to develop a fast method, easy to implement, accurate at least in the region near the interface (also called *narrow band* in a level set context) and readily applicable to finite element multiphase flow solvers. In order to reach these desirable features, we propose to employ the finite element interpolation itself to hold the Eikonal property at element level. We also propose an advancing front algorithm that instead of using heap-lists is simply based on a list of elements that can be used to compute the distance, and advances inserting, enabling and disabling new elements according to the algorithm evolution. This algorithm uses simple data structures present in most finite element mesh generators and adaptive codes (see, for instance, [21]). Furthermore, this new method, while relaxing some of the FMM assumptions, inherits some of its features, such as: the interface is kept fixed during the distance computation, the method marches in both directions from the interface considering the distance signs, there is no need to explicitly determine where the interface is, the computation can be restricted within a narrow band to decrease computational costs and the method can be also employed to solve other problems related to arrival time, shortest path or distance functions computation.

The outline of this work is as follows: in the first part of Section 2 we discuss how to compute the distance function using finite element interpolations to impose the satisfaction of Eikonal equation at element level. In the second part, the algorithm employed to evolve the computations is detailed and illustrated. In Section 3 some examples are presented to validate and evaluate the performance of our proposal and in the last section our conclusions and final remarks are presented.

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

2. DISTANCE FUNCTION COMPUTATION

This section presents a novel method for computing distance functions in unstructured meshes. The method is based on the use of finite element interpolation to hold the Eikonal property ($\|\nabla\varphi\| = 1$) at element level over the whole computational domain. We also propose an algorithm to evolve the computations that can be easily implemented and extended to two-dimensional and three-dimensional cases.

2.1. Solving the Eikonal equation element by element

Let us assume that the finite element version of the Eikonal equation computed at element level is as follows:

$$\|\nabla\varphi^{\mathrm{e}}\| = \|\mathbf{B}^{\mathrm{T}}\mathbf{d}\| = 1 \tag{2}$$

where φ^{e} is the element distance function, $\mathbf{d}$ is the element distance vector and $\mathbf{B}$ is the discrete gradient operator. For a linear tetrahedral element $\mathbf{d}$ and $\mathbf{B}$ can be expressed as

$$\mathbf{d}^{\mathrm{T}} = [d_1 \;\; d_2 \;\; d_3 \;\; d_4] \tag{3}$$

$$\mathbf{B} = \frac{1}{6V}\begin{bmatrix} N_{1,x} & N_{1,y} & N_{1,z} \\ N_{2,x} & N_{2,y} & N_{2,z} \\ N_{3,x} & N_{3,y} & N_{3,z} \\ N_{4,x} & N_{4,y} & N_{4,z} \end{bmatrix} \tag{4}$$

where V is the element volume and $N_{i,j}$ is the partial derivative of the element interpolation at node i and direction j, given, respectively, as

$$\begin{aligned}
&N_{1,x} = (z_{43}y_{23} - z_{23}y_{43}), \quad N_{1,y} = (z_{42}y_{32} - z_{32}y_{42}), \quad N_{1,z} = (y_{43}x_{23} - x_{43}y_{23}) \\
&N_{2,x} = (y_{31}z_{41} - y_{41}z_{31}), \quad N_{2,y} = (x_{41}z_{31} - x_{31}z_{41}), \quad N_{2,z} = (y_{41}x_{31} - y_{31}x_{41}) \\
&N_{3,x} = (y_{41}z_{21} - y_{21}z_{41}), \quad N_{3,y} = (x_{21}z_{41} - x_{41}z_{21}), \quad N_{3,z} = (y_{42}x_{12} - x_{42}y_{12}) \\
&N_{4,x} = -(N_{1,x} + N_{2,x} + N_{3,x}), \quad N_{4,y} = -(N_{1,y} + N_{2,y} + N_{3,y}) \\
&N_{4,z} = -(N_{1,z} + N_{2,z} + N_{3,z})
\end{aligned} \tag{5}$$

$$6V = x_{21}(z_{31}y_{14} - y_{31}z_{14}) + x_{31}(z_{12}y_{14} - y_{12}z_{14}) + x_{41}(z_{12}y_{31} - y_{12}z_{31}) \tag{6}$$

and

$$x_{ij} = x_i - x_j, \quad y_{ij} = y_i - y_j, \quad z_{ij} = z_i - z_j \tag{7}$$

Thus, the distance function gradient at element level is simply

$$\begin{bmatrix} \varphi^{\mathrm{e}}_{,x} \\ \varphi^{\mathrm{e}}_{,y} \\ \varphi^{\mathrm{e}}_{,z} \end{bmatrix} = \begin{bmatrix} N_{1,x}d_1 + N_{2,x}d_2 + N_{3,x}d_3 + N_{4,x}d_4 \\ N_{1,y}d_1 + N_{2,y}d_2 + N_{3,y}d_3 + N_{4,y}d_4 \\ N_{1,z}d_1 + N_{2,z}d_2 + N_{3,z}d_3 + N_{4,z}d_4 \end{bmatrix} \tag{8}$$

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

which lead us to the following constraint in order to hold the Eikonal equation at element level,

$$(\varphi^{\mathrm{e}}_{,x})^2 + (\varphi^{\mathrm{e}}_{,y})^2 + (\varphi^{\mathrm{e}}_{,z})^2 = 1 \tag{9}$$

It is easy to realize that given three known distances within a tetrahedron we can readily compute the fourth distance solving the quadratic constraint described in Equation (9). This method is an extension, in a finite element context, of the FMM initially proposed by Sethian [16]. Expanding Equation (9) to compute the distance at the fourth element node we arrive at

$$(d_x + N_{4,x}d_4)^2 + (d_y + N_{4,y}d_4)^2 + (d_z + N_{4,z}d_4)^2 = 1 \tag{10}$$

where

$$d_x = N_{1,x}d_1 + N_{2,x}d_2 + N_{3,x}d_3 \tag{11}$$

$$d_y = N_{1,y}d_1 + N_{2,y}d_2 + N_{3,y}d_3 \tag{12}$$

$$d_z = N_{1,z}d_1 + N_{2,z}d_2 + N_{3,z}d_3 \tag{13}$$

The quadratic form of Equation (10) is

$$\underbrace{(N_{4,x}^2 + N_{4,y}^2 + N_{4,z}^2)}_{a} d_4^2 + \underbrace{(2d_xN_{4,x} + 2d_yN_{4,y} + 2d_zN_{4,z})}_{b} d_4 + \underbrace{(d_x^2 + d_y^2 + d_z^2 - 1)}_{c} = 0 \tag{14}$$

and following FMM, we will assume that the distance of the fourth node can be taken as the maximum of the two real roots, thus

$$d_4 = \max\left(\frac{-b + \sqrt{b^2 - 4ac}}{4ac}, \frac{-b - \sqrt{b^2 - 4ac}}{4ac}\right) \tag{15}$$

Note that the roots of Equation (14) will be the possible values to hold a unitary gradient within the element as required by the Eikonal equation. It is also important to emphasize that the computational solution of the quadratic equation (14) requires a careful treatment in order to avoid roundoff errors [22]. However, in many situations Equation (14) does not have any real roots. In two dimensions, the inability to solve the quadratic equation (14) corresponds to an inability to tilt the plane representing the distance approximation within an element at an appropriate angle (see [18] for further details). Possible solutions for this problem, as indeed pointed out in References [18, 19], involve looking for other elements to compute, modifying locally the mesh, using high-order approximations or a combination of them. Here, we will simply skip the elements where we found imaginary roots, approximating the missing nodal distance by a suitable defined interpolation involving the surrounded known nodes, as described below.

2.2. *The algorithm*

In this section we describe the algorithm employed to evolve Equations (14) and (15) over the whole computational domain. We will describe the algorithm for triangulated meshes in order to make it easier to understand, but the reader should note that the extension for three-dimensional cases is straightforward. The algorithm is based on the FMM proposed

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

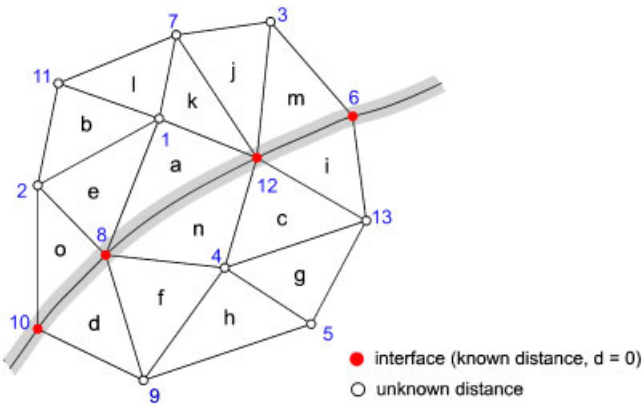

Figure 1. Triangulated mesh and interface.

by Sethian [16] and its unstructured grid variations [18, 19]. Thus, let us consider the triangulated mesh in Figure 1.

The filled red nodes represent origin nodes (level set zero in the level set context) where the distances are known ($d = 0.0$) and the open black nodes represent the unknown nodes. Elements are identified by letters while nodes are represented by numbers. Taking as an example, consider the node '1' in Figure 1. Note that only the element 'a', which has two known nodes, is capable of producing the distance for the node '1'. Furthermore, after computing the distance for this node we will be capable of computing also the distances for the nodes '2' and '7' through the elements 'e' and 'k', respectively. This is the main idea of the proposed algorithm: solve any computable element; those that have at least two known nodes for triangulated meshes or three for tetrahedral meshes; and make other elements available to later computations according to the algorithm evolution. Note that we are not concerned about which node or element is closer to the interface, as is done in the min-heap list employed in classical implementations of the FMM, but we are trying to compute a valid solution from any computable element instead. Moreover, the computable elements will arise, naturally, from those closest to the interface and will march inwards and outwards according to the algorithm evolution. Another important issue to discuss is that, after solving node '1' from element 'a' and node '2' from element 'o', the element 'e' might be discarded from the computation. This assumption will clearly reduce the number of algorithm iterations.

Now we can describe the main components involved in our algorithm. From Figure 1 we can easily build the following arrays:

`ElementsToSolve:`

a	d	i	m	n	o							

`KnownNodes:`

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
2	0	1	2	1	1	0	0	2	1	1	0	2	2	2

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

ElementsPerNode:

1	a	k	l	b	e		
2	o	e	b				
3	j	m					
4	h	g	c	n	f		
5	g	h					
6	m	i					
7	l	k	j				
8	d	f	n	a	e	o	
9	d	f	h				
10	o	d					
11	b	l					
12	c	i	m	j	k	a	n
13	i	c	g				

where `ElementsToSolve` contains the elements with at least two nodes (for triangulated meshes) with known distances, `KnownNodes` is an element list employed to count the number of known nodes for each element and `ElementsPerNodes` is a table to hold the elements sharing a node. Besides the structures accounted here we need one vector to store the status of each node (enabled or disabled for computation). This additional information is available or is readily built as derived data structures from typical finite element arrays [21]. Thus, the algorithm can be summarized in the steps given in Box 1.

Remarks

- Since the `ElementsToSolve` array represents a buffer to store elements to be used during the computations it can be set with any desired size and restarted when necessary.

Box 1. Algorithm for computing distance fields in unstructured meshes.

```
i = 0
WHILE there are elements in ElementsToSolve DO :
   Recover an element from ElementsToSolve(i)
   IF ( number of known nodes is 2)  THEN
         Compute the unknown node solving Eq. (14)
      IF ( real roots were found) THEN
            Stores the computed distance
            Turns the node recently computed off
            FOR each element sharing the node recently computed DO :
               Recover the element from ElementsPerNodes
               Increment KnownNodes for the element recovered
               IF ( element reaches 2 known nodes) THEN
                  Insert the element in ElementsToSolve
            ENDIF
         END FOR
      ENDIF
   ENDIF
   increment i
END WHILE
```

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

- At the end of the solution procedure a few nodes with unknown distances may remain due to the discarded elements with complex roots for Equation (15). For these nodes, we can estimate a valid distance computing its value at the baricenter of the polyhedron formed by the known nodes surrounding the unknown node. This task can be easily accomplished since we have stored the necessary data in the `ElementsPerNodes` array.
- We are restricting ourselves to examples where the interface passes exactly through the element nodes. Note that in practical problems the interface commonly cuts the element edges. In these cases the interface elements can be treated as an element where all of its nodes have known distances and the algorithm would remain the same as we have described.
- When working with signed distance fields, the interface elements are easily found through the search of those elements where a signal change is detected. Note that this task can be accomplished without having to represent the interface explicitly.
- In order to keep the algorithm as simple as possible we have been adopting to solve the unknown node always in the fourth local position of the element and directly using Equation (14) to compute the distance value. This can be accomplished by applying a circular shift on the element nodal connectivity.
- The proposed algorithm, as observed in the FMM and its variants, does not change the position of the original interface. It is an attractive feature for multiphase flow problems employing level set methods since we guarantee that the masses of the phases are preserved during the redistancing procedure.

Figure 2(a)–(f) illustrates the first six iterations of the proposed algorithm corresponding to the mesh and data structures given previously. In this figure the filled black nodes are nodes where the iteration is being performed, blue elements are those with two known nodes (enabled for computation) and red elements are elements discarded from the computation since their nodes are already known (disabled for computation elements).

From Figure 2 we can see that the algorithm advances following the `ElementsToSolve` list including and excluding new elements according to the distances evaluation. We can also note that the algorithm does not march in a sequential manner. It develops the computation by selecting elements inward and outward indifferently.

3. NUMERICAL TESTS

In this section we will present some numerical tests in order to evaluate the accuracy and efficiency of our scheme to construct valid distance fields in unstructured meshes formed by linear tetrahedral finite elements. The algorithm, coded in Fortran 90, runs in a Dell Precision 370 workstation (Intel Pentium 4 3.6 GHz/ 1 Mb/800 MHz, 4 Gb of DDR2 533 MHz SDRAM and Windows XP Professional).

3.1. Sphere expansion and contraction

In this test, the surface of a sphere with unitary radius centered in a cube with side 4 is used to evaluate the accuracy of the proposed method. For this problem the analytical solution of the

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

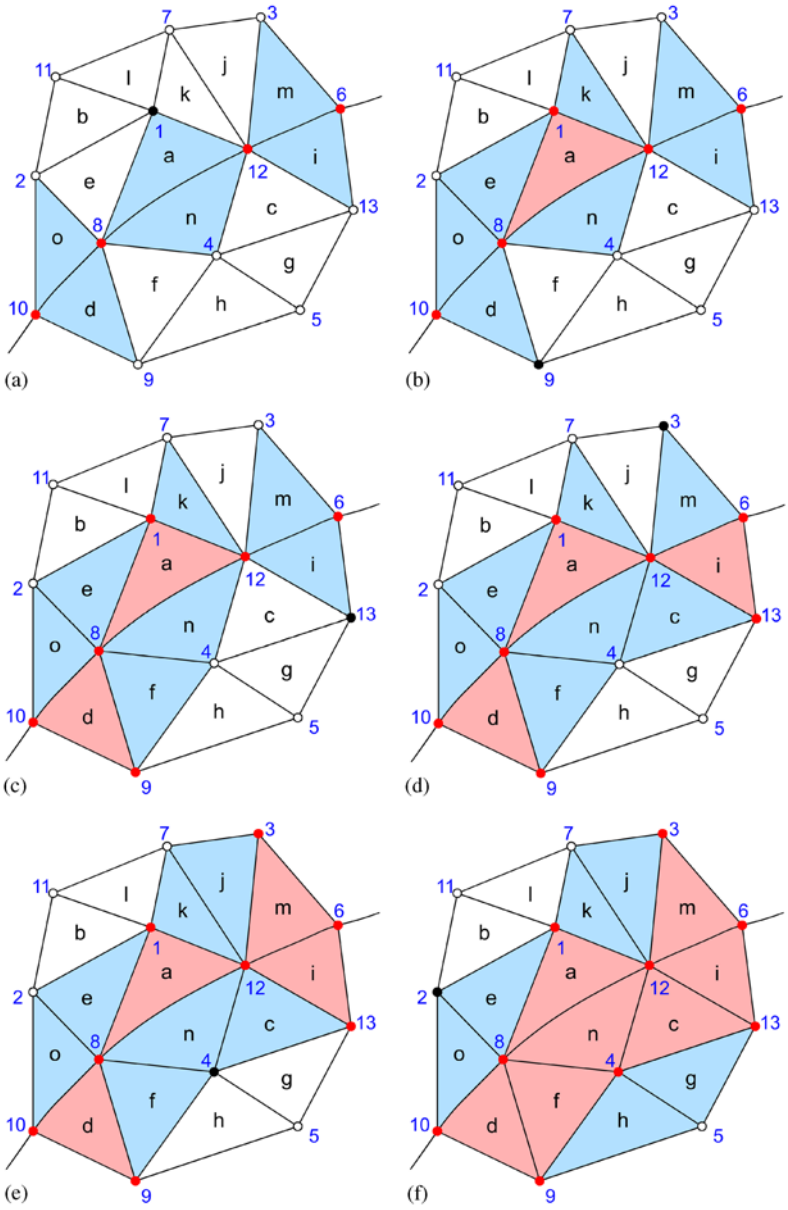

Figure 2. Illustration of the first six steps of the algorithm presented in Section 2.2: (a) computing node '1' from element 'a', enabling elements 'e' and 'k' to compute nodes '2' and '7' and disabling element 'a'; (b) computing node '9' from element 'd', enabling element 'f' to compute node '4' and disabling element 'd'; (c) computing node '13' from element 'i', enabling element 'c' to compute node '4' and disabling element 'i'; (d) computing node '3' from element 'm', enabling element 'j' to compute node '7' and disabling element 'm'; (e) computing node '4' from element 'n', enabling elements 'h' and 'g' to compute node '5' and disabling elements 'f' and 'c'; and (f) computing node '2' from element 'o', enabling element 'b' to compute node '11' and disabling elements 'o' and 'e'.

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

unsigned distance field can be computed from the following equation:

$$d(\mathbf{x}) = \begin{cases} \|\mathbf{x}\| - r & \text{if } \|\mathbf{x}\| \geqslant r \\ r - \|\mathbf{x}\| & \text{if } \|\mathbf{x}\| < r \end{cases} \tag{16}$$

where $\mathbf{x}$ is the position vector and r is the radius of the sphere. The computations were carried out in three different unstructured meshes with element sizes listed in Table I. Figure 3 shows external surface clips for the tetrahedral meshes employed.

The unsigned distances, computed with SPH3, for this problem are shown in Figure 4(a) through isocontours while the details within a narrow-band region corresponding to 4 times the average edge length (h) inward and outward are given in Figure 4(b).

In order to evaluate the accuracy of the proposed method, the relative error norms of the computed distance fields were compared to the analytical solution given by Equation (16) for the three meshes employed. These errors, according to the distance from the interface, are illustrated in Figure 5.

The results show that the proposed method is able to compute accurate distance fields even in the worst case, that is, for the coarsest mesh (SPH1) and computing the distance field in the full range. The errors, considering the whole domain, ranged from 0.59% in the best case (SPH3 mesh) to 1.21% in the worst case (SPH1 mesh).

In level set methods it is common to define a region around the interface, known as narrow band, and restrict the computations within this region in order to reduce computational efforts. In this sense, Figure 5 shows that our method is able to find errors lesser than 0.2% for narrow bands of $2.5h$, $4h$, and $8h$ for the SPH1, SPH2, and SPH3 meshes, respectively. These results

Table I. Mesh parameters for the problem of computing a distance function on a sphere centered in a cube.

Mesh	Element size	Elements	Nodes
SPH1	0.20	48 550	26 366
SPH2	0.15	159 623	28 977
SPH3	0.10	505 458	89 303

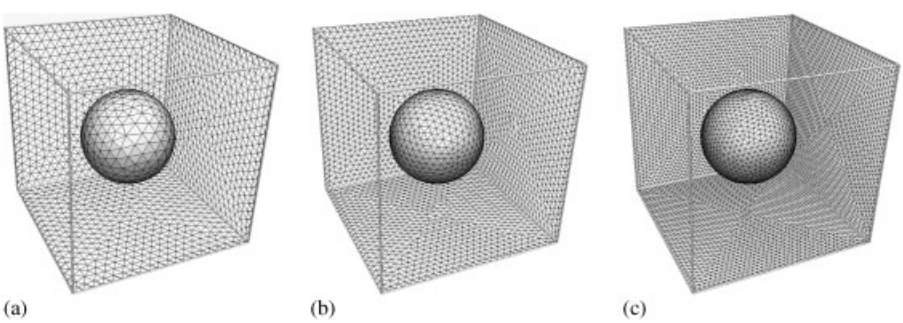

Figure 3. External surface clips of the tetrahedral meshes: (a) SPH1: $h = 0.20$; (b) SPH2: $h = 0.15$; and (c) SPH3: $h = 0.10$.

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

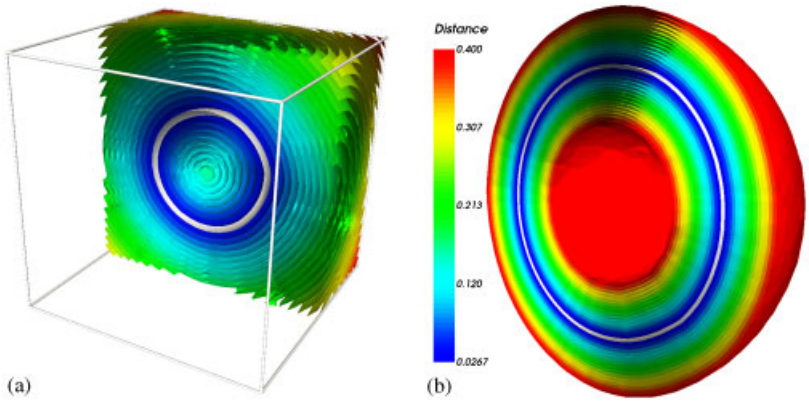

Figure 4. Unsigned distance function for the problem of expand/contract a sphere: (a) computed unsigned distance field and (b) narrow band $[-4h, +4h]$.

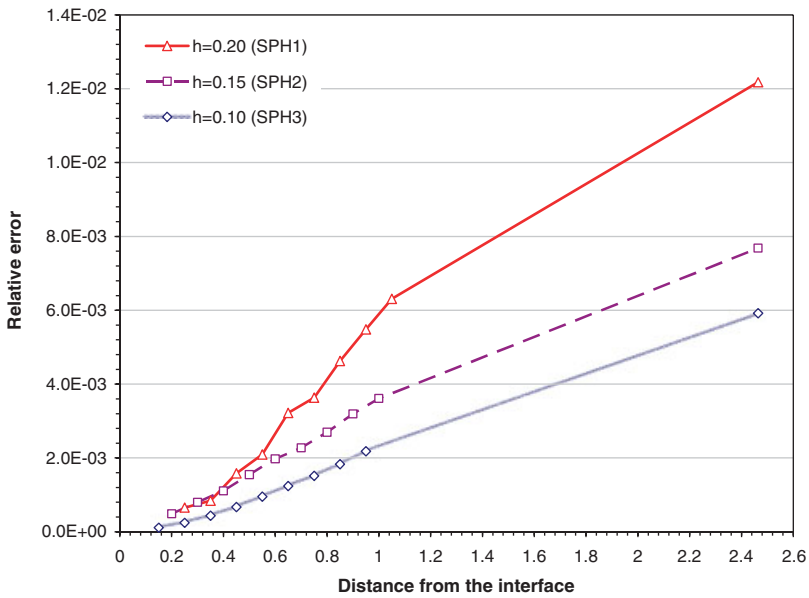

Figure 5. Relative error norms according to the distance from the interface for increasingly refined meshes.

suggest that the proposed scheme is sufficiently accurate to be readily employed in finite element solvers based on level set methods.

3.2. *Cube expansion and contraction*

In this case the surface of a box with side 2, centered in a cubic domain with side 4, is expanded and contracted. This problem is particularly interesting for testing the efficiency of the algorithm

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Table II. Mesh parameters for the problem of computing a distance function on a cube centered in a box.

Mesh	Element size	Elements	Nodes
CUB1	0.20	60 534	11 408
CUB2	0.15	158 057	28 727
CUB3	0.10	497 595	87 830

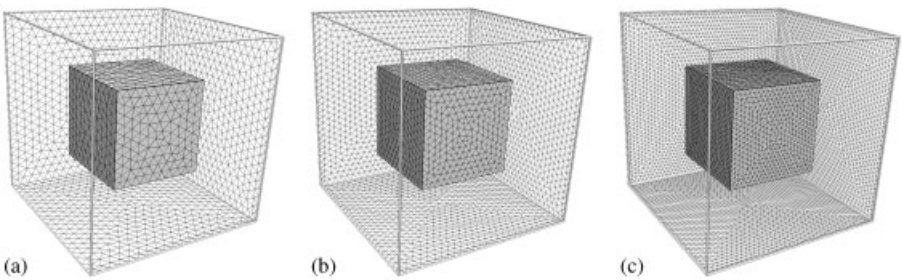

Figure 6. External surface clips of the tetrahedral meshes: (a) CUB1: $h = 0.20$; (b) CUB2: $h = 0.15$; and (c) CUB3: $h = 0.10$.

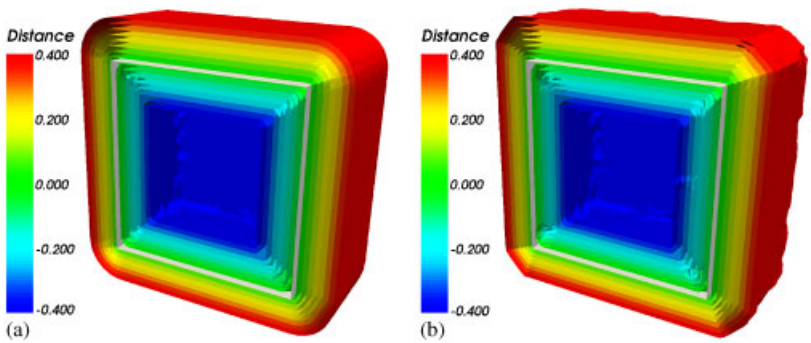

Figure 7. Signed distance function corresponding to a narrow band of $[-4h, +4h]$ for the problem of expand/contract a cube: (a) analytic and (b) computed.

to overcome geometries with corners and sharp edges. The element sizes of the different cases evaluated are listed in Table II while the corresponding meshes are shown in Figure 6(a)–(c).

The solution illustrated in Figure 7(b) shows that the algorithm suffers when folds and corners are present and the accuracy begins to be lost in regions far from the interface. This assumption is confirmed through Figure 8, where the relative error norms are plotted according to the distance from the interface.

For this case, the results presented in Figure 8 show that the proposed algorithm was able to obtain a distance field with errors around 4% considering the whole domain. The results also

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

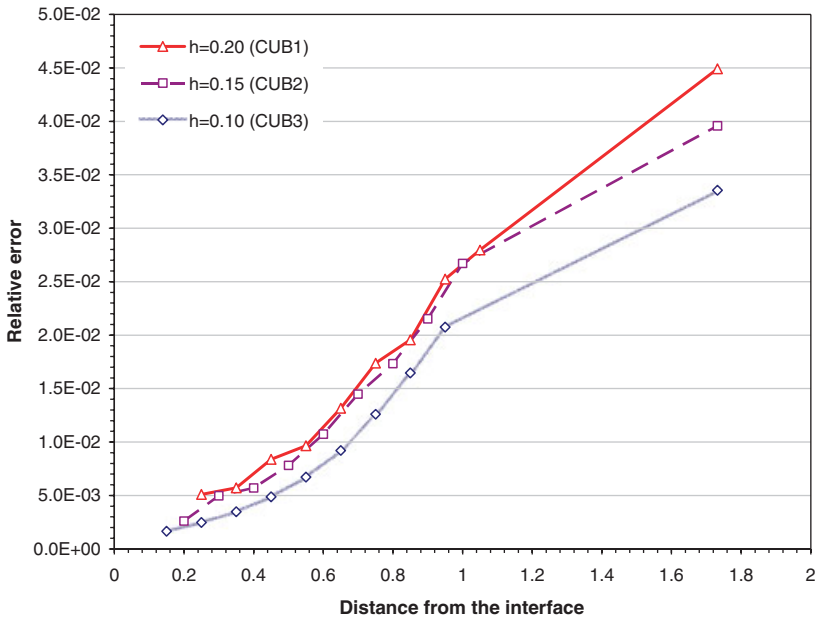

Figure 8. Relative error norms according to the distance from the interface for increasingly refined meshes.

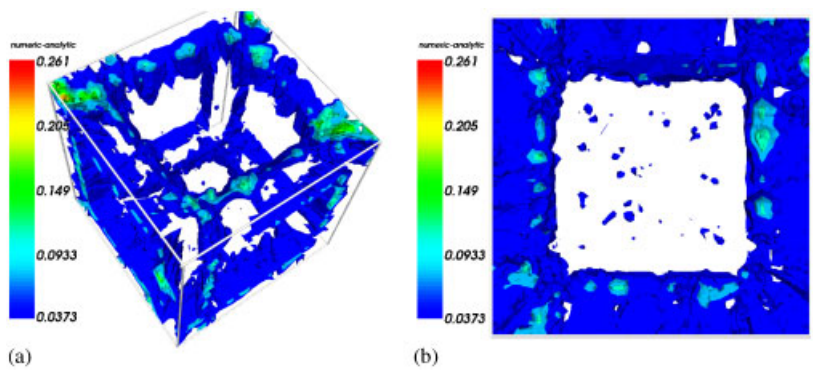

Figure 9. Differences between the analytic and computed solutions.

reveal that errors beneath 1% were obtained within a narrow band of $3h$, $4h$, and $7h$ for the CUB1, CUB2, and CUB3, respectively.

The differences between the computed and the analytical solutions for CUB3 mesh are shown in Figure 9(a) and (b). This figure shows us that larger differences are concentrated in regions far from the interface mainly in areas with edges. Furthermore, in regions close to the interface, differences larger than 0.04 were not verified.

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Table III. Computational parameters.

			Nodes		Iterations	Time (s)
Model	Direction	Elements	Total	Known	Full model	Full model
sph1	in/out	60 749	11 469	321	11 264	0.0468
sph2	in/out	159 623	28 977	686	28 474	0.1406
sph3	in/out	505 458	89 303	1326	88 783	0.4375
cub1	in/out	60 534	11 408	668	11 056	0.0468
cub2	in/out	158 057	28 727	1300	28 307	0.1406
cub3	in/out	497 595	87 830	2684	88 390	0.4375
dam	in/out	251 807	46 766	814	47 266	0.2188
indy	out	1 602 025	309 657	73 035	244 782	1.3281
subwp	out	817 608	292 984	33 685	117 980	0.6094
cyl3D	out	446 662	81 991	1260	94 245	0.4219
yf17	out	528 915	97 104	6394	98 030	0.4531
LeMans	out	10 264 863	1 858 246	111 972	2 142 343	11.9575

3.3. Real-world test problems

In this section the complexity and performance of the proposed algorithm are discussed based on the computation of distance fields for various finite element real-world models. These models were chosen since they are challenging for any distance function algorithm due the presence of folds, apexes, locus points, corners among others geometric singularities. The models range from simple and coarse meshes (e.g. *sph1*, *cub1*, and *dam*) to highly detailed and fine models (e.g. *indy*, *subwp*, and *lemans*) to evaluate the performance of our algorithm. The main computational parameters for all meshes are listed in Table III. The number of initially known nodes comprises those nodes where the level set 0 lies. All timings and number of iterations listed in Table III refer to the algorithm running on the whole computational domain. For some models (the last five models in Table III) the distance function was computed only in the outward direction. The level sets obtained for a narrow band around $4h$ (h based on the average edge length) are illustrated in Figure 10(a)–(f).

Table III shows that, for most of the models, the proposed algorithm was able to find a distance field in a number of iterations lesser than the number of total nodes in the mesh (*nnos*). We can also remark that while in the *subwp* case the algorithm carried out a number of iterations corresponding to $0.4 \times nnos$, for the *LeMans* model $1.15 \times nnos$ iterations were necessary to compute the desired distance field. Nevertheless, in all cases computer times were proportional to the number of nodes in the meshes. The timings data in Table III showed us that for most models the algorithm was fast and found solutions in less than 2 s. Note that this performance tends to improve if we restrict the solution within a narrow band.

4. CONCLUSIONS

This work proposed a new method for computing distance functions in unstructured grids imposing the satisfaction of Eikonal equation at element level. In order to evolve the solution throughout a computational grid a new advancing front algorithm is also proposed. The algorithm employs a list of elements available for computing the distance, while updating this list according to the

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

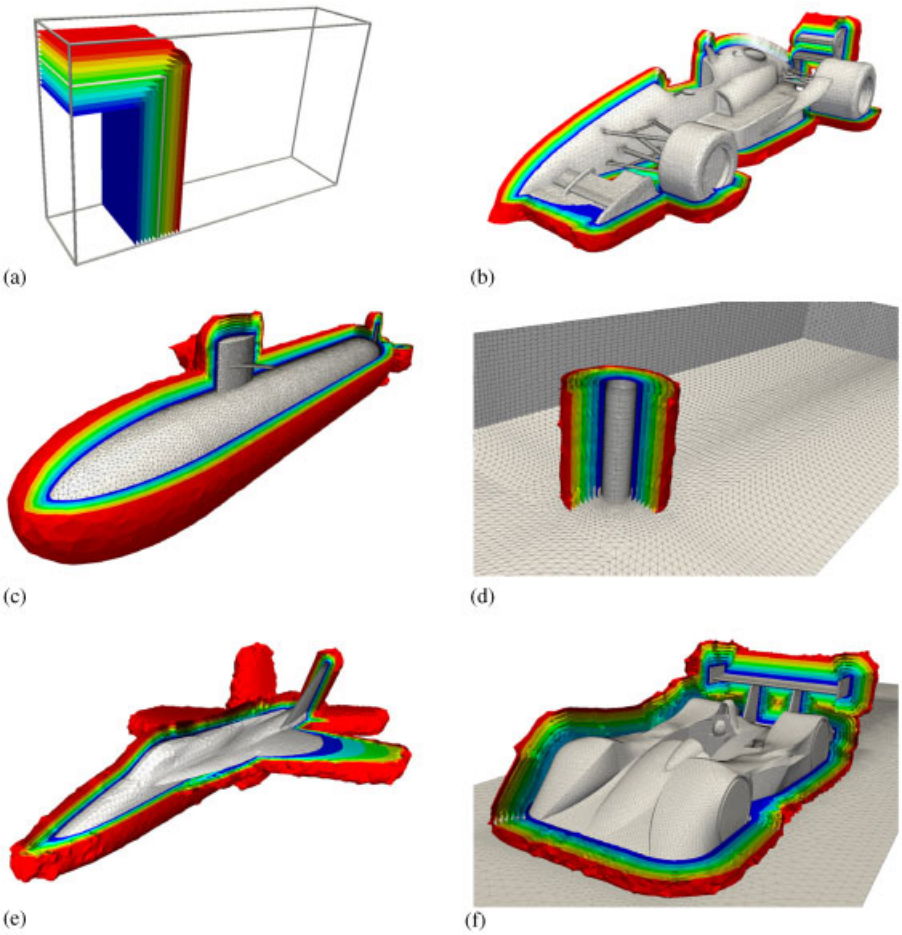

Figure 10. Distance fields for several real-world problems: (a) dam break; (b) Indycar car; (c) Los Angeles class submarine; (d) flow around a circular cylinder; (e) YF17; and (f) Le Mans car.

algorithm march, inserting, enabling and disabling new elements for further computations. The method is easy to implement and can be readily employed in finite element solvers since all information necessary is available or is readily built as derived data structures. The tests showed that the proposed method is efficient even for large models and can be further speeded up by restricting the computations within a narrow band. The results also demonstrated that the method was able to find accurate solutions in computer times proportional to the number of nodes in the several meshes examined.

ACKNOWLEDGEMENTS

The authors would like to thank the financial support of the Petroleum National Agency (ANP, Brazil), the Center for Parallel Computations (NACAD) at the Federal University of Rio de Janeiro. Dr Marcos

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme

Martins gratefully acknowledges the financial support of the Brazilian Council of Technological and Scientific Development—CNPq, through grant CT-PETRO/PROSET 500.196/02-8.

REFERENCES

1. Osher S, Fedkiw R. *Level Set Methods and Dynamic Implicit Surfaces*. Applied Mathematical Sciences, vol. 153 (1st edn). Springer: Berlin, 2002.
2. Sethian JA. *Level Set Methods and Fast Marching Methods—Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge Monographs on Applied and Computational Mathematics (2nd edn). Cambridge University Press: NY, U.S.A., 1999.
3. Jones MW, Bærentzen JA, Sramek M. 3D distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics* 2006; **12**(4):581–599.
4. Rouy E, Tourin A. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis* 1992; **29**:867–884.
5. Sussman M, Smereka P, Osher S. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics* 1994; **114**:146–159.
6. Sussman M, Fatemi E. An efficient interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on Scientific Computing* 1999; **20**:1165–1191.
7. Barth TJ, Sethian JA. Numerical schemes for the Hamilton–Jacobi and level set equations on triangulated domains. *Journal of Computational Physics* 1998; **145**(1):1–40.
8. Shepel SV, Smith BL, Paolucci S. Implementation of a level set interface tracking method in the FIDAP and CFX-4 codes. *Transactions of the ASME* 2005; **127**:674–686.
9. Nagrath S, Jansen KE, Lahey RTJ. Computation of incompressible bubble dynamics with a stabilized finite element level set method. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**:4565–4587.
10. Nagrath S, Jansen KE, Lahey RTJ, Akhatov I. Hydrodynamic simulation of air bubble implosion using a level set approach. *Journal of Computational Physics* 2006; **215**:98–132.
11. Mourad HM, Dolbow J, Garikipati K. An assumed-gradient finite element method for the level set equation. *International Journal for Numerical Methods in Engineering* 2005; **64**(8):1009–1032.
12. Marchandise E, Remacle JF, Chevaugeon N. A quadrature-free discontinuous Galerkin method for the level set-equations. *Journal of Computational Physics* 2006; **212**:338–357.
13. Ventura G, Xu JX, Belytschko T. A vector level set method and new discontinuity approximations for crack growth by EFG. *International Journal for Numerical Methods in Engineering* 2002; **54**:923–944.
14. Ventura G, Budyn E, Belytschko T. Vector level sets for description of propagating cracks in finite elements. *International Journal for Numerical Methods in Engineering* 2003; **58**:1571–1592.
15. Belytschko T, Daniel WJT, Ventura G. A monolithic smoothing-gap algorithm for contact-impact based on the signed distance function. *International Journal for Numerical Methods in Engineering* 2002; **55**:101–125.
16. Sethian JA. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences* 1995; **93**(4):1591–1595.
17. Sedgewick R. *Algorithms*. Addison-Wesley: Reading, MA, 1988.
18. Kimmel R, Sethian JA. Computing geodesic paths in manifolds. *Proceedings of the National Academy of Sciences* 1998; **95**(15):8431–8435.
19. Sethian JA, Vladimirsky A. Fast methods for the Eikonal and related Hamilton–Jacobi equations on unstructured meshes. *Proceedings of the National Academy of Sciences* 2000; **97**(11):5699–5703.
20. Gross S, Reichelt V, Reusken A. A finite element based level set method for two-phase incompressible flows. *IGPM Report Nr. 243*, RWTH Aachen, Germany, May 2004 (ftp://ftp.igpm.rwth-aachen.de/pub/reports/pdf/IGPM243.pdf).
21. Löhner R. *Applied CFD Techniques*. Wiley: New York, 2001.
22. Heath MT. *Scientific Computing*: *An Introductory Survey* (2nd edn). McGraw-Hill: New York, 2002.

Copyright © 2007 John Wiley & Sons, Ltd.

DOI: 10.1002/nme