ELSEVIER

Comput. Methods Appl. Mech. Engrg. 190 (2001) 3771–3796

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

# Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations

C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira *, A.J.H. Goddard

*Computational Physics and Geophysics Group, T.H. Huxley School of Environment, Earth Science and Engineering, Imperial College of Science, Technology and Medicine, Royal School of Mines Building, Prince Consort Road, London SW7 2BP, UK*

Received 29 November 1999

**Abstract**

A method for optimising a pre-existing mesh of tetrahedral finite elements is described. It is based on a series of mesh connectivity and node position searches of the landscape defining mesh quality. A Riemannian metric, reflecting the a posteriori error measure, is used to calculate element size and shape. A functional is defined which embodies both shape and size quality of an element with respect to the metric, and is used to gauge mesh quality. A heuristic-based local search strategy is adopted – local in the sense that it has no hill-climbing abilities. The paper presents applications of the method to complex, steady-state and time-dependent problems which highlight its anisotropic, feature-capturing abilities. Numerical evidence is provided which suggests that the computational complexity (time) of the proposed algorithm varies linearly with the number of elements or nodes of the finite element mesh. 

*Keywords:* Riemannian metric; A posteriori error measure; Finite element; Tetrahedral mesh; Optimisation heuristics; Anisotropic adaptivity

## 1. Introduction

A common problem encountered by numerical practitioners is that the computational grid or mesh used in a numerical simulation has to be generated a priori to the solution procedure. It is therefore difficult to resolve adequately the local physical features at a first attempt, and the mesh often needs to be adapted to enable the solution procedure to satisfy the resolution requirements. Importantly, this also allows the reduction in the computational effort which is crucial for complex applications. Developing ways of systematically modifying or adapting computational meshes to the underlying physics of problems has been the subject of increased interest in recent years and now forms one of the most active areas of computational research.

Mesh adaptivity or optimisation relies on the derivation of an appropriate error measure which dictates how the mesh is to be modified. Several error measures which serve as a criterion for mesh modification have been proposed ranging from using interpolation-based methods (which are a subset of explicit methods, see, for example, [26,33,38]), for calculating a posteriori error measures, to implicit methods which can correctly propagate the error through the mesh, see [3,36]. An additional consideration, when adapting the mesh is that not only node density, but also the alignment of the mesh relative to the solution is important. For example, in boundary layers or along shocks the mesh may need to be refined in one

* Corresponding author. Tel.: +44-171-594-9319; fax: +44-171-594-9341.
*E-mail addresses:* c.pain@ic.ac.uk (C.C. Pain), c.oliveira@ic.ac.uk (C.R.E. de Oliveira).

0045-7825/01/$ - see front matter 
PII: S0045-7825(00)00294-2

direction only, e.g., normal to the boundary layer. Hessian-based adaptivity methods are capable of fulfilling this requirement. They have two main attractive features: they are independent of the specific problem being solved and use a non-euclidean metric enabling the meshes to be built truly automatically.

Once the error measure has been derived, the nodes are redistributed according to this measure. This step in the adaptive procedure is quite complex algorithmically, and is intrisically dependent on the mesh generation method and geometrical and material boundary conforming requirements. Unstructured tessellation methods can deal with geometries of arbitrary complexity and they render themselves naturally to adaptivity. There are three main approaches to the generation of unstructured tetrahedral meshes: the Delaunay triangulation scheme [18], the Advancing Front technique [28,30] and the Quadtree/Octree approach [24]. Delaunay tessellation is a procedure for forming optimal tesselations using an appropriate metric for measuring distances. It leads to robust algorithms which can easily incorporate adaptivity. Region-wise coarsening/refinement can be achieved with the Advancing Front technique in which both new nodes and triangles are regenerated in the region [24]. The main drawback of this technique is the fact that the mesh has to be regenerated at every mesh adaptation, which can be computationally expensive for large problems. Octree methods by nature do not offer scope for aligning the mesh anisotropically with the solution.

In general, 2-D Delaunay meshes are of good quality and can lead to useful mesh adapting schemes for triangular [39] and quadrilateral [10] elements. However, in 3-D when nodes are evenly distributed, the element aspect ratio can become very large [18]: small inscribed spheres (that is thin 'slivers') can be generated, despite all the edge lengths of an element being approximately equal with respect to the desired metric. Local mesh connectivity transformations have been used in [23] to obtain a mesh that satisfies the Delaunay property along with a modification to optimise the interior dihedral angle of the resulting tetrahedral elements and discourage thin 'slivers'. Further work on 3-D Delaunay meshing can be found [11,12,14,37].

The fact that the Delaunay kernel permits element 'slivers' motivated the authors to discard it in favour of mesh optimisation methods. Such methods are guaranteed to converge and have proven relatively insensitive to round-off error, at least in our implementation. This research does not represent the first use of mesh optimisation though. For example, [16] demonstrated successful mesh optimisation involving swapping (mesh connectivity adjustments) and smoothing (node position adjustments) of a mesh, and that using either of these independently leads to severely reduced mesh quality. In particular, interior element angles are highly sub-optimal with node movement alone. Also, it was demonstrated that purely optimisation-based node movement is CPU-expensive when compared with simpler methods such as Laplacian-based smoothing, and with little gain in mesh quality over the latter. This has motivated us to use approximate optimisation techniques for mesh smoothing (node movement). In addition, Buscaglia and Dari [13] also uses optimisation methods for 2-D meshes. They used, as we do here, the quality of the poorest element (as defined by some functional) as a measure of the overall mesh quality. This was all achieved in the context of a Riemannian metric based on a Hessian. The Hessian was calculated by introducing explicit artificial diffusion and used both weak and strong finite element forms for the Hessian calculation. In [18] the mesh connectivity/node movement operations, required by an optimisation method, are described in detail.

In what follows, a method is presented which adapts the mesh to the solution without sacrificing the integrity of the boundary (geometry), or internal boundaries (regions) of the domain. It circumvents the complexities of boundary-conforming Delaunay methods [17] by operating on the existing mesh. The error measure employed is based on the curvature of the solution and provides a directional measure. The objective is to obtain a mesh which has a uniform interpolation error in any direction. This is accomplished with use of a metric which is related to the Hessian of the solution field. Appropriate scaling of the metric [14] enables the resolution of multi-scale phenomena as encountered in transient incompressible fluids calculations. The resulting metric is used to calculate element size and shape. The mesh optimisation method is based on a series of mesh connectivity and node position searches of the landscape defining mesh quality which is gauged by a functional. The mesh modification thus fits the solution field(s) in an optimal manner.

The details of the method are presented in the next section. Numerical assessment of its performance is presented in Section 3. The mesh optimisation/adaptivity method described here is of general applicability.

It can be used for applications ranging from obtaining good quality meshes without prior knowledge of the solution to adapting the meshes to follow transients and to iteratively adapt the finite element solution/mesh towards an optimal steady state. Examples of this are presented in Section 4 with application to unsteady fluid flow and steady-state radiation transport problems.

## 2. A mesh optimisation and adaptivity method

The mesh adaptivity approach described below is based on a variational functional which gauges the quality of the mesh including both element size and shape as objectives. The functional is optimised using local searches of the mesh connectivity and node positions. An appropriate definition of a metric based on the Hessian allows distances in the desired Euclidean space to be calculated, which in turn allows the element (length) size to be controlled by, for example, a specified interpolation error.

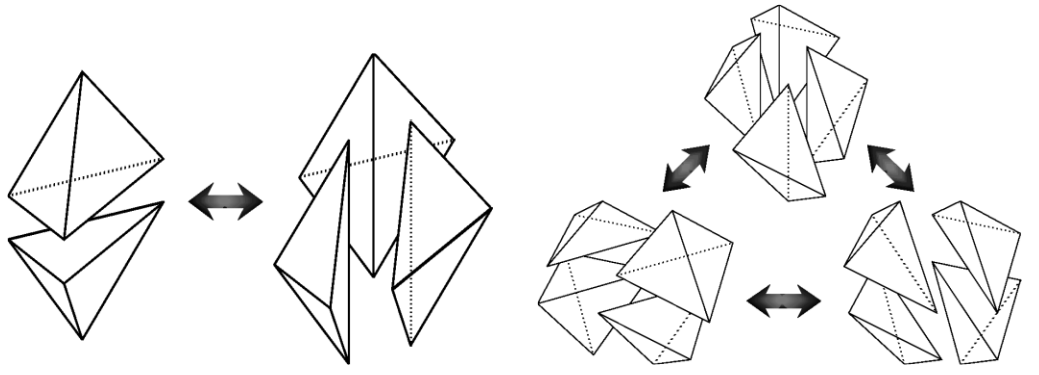
### 2.1. Optimisation heuristics

The optimisation method aims to improve, locally, the worst element. The overall objective function is the element quality associated with the worst element of the mesh. Within the optimisation, each element in the mesh is visited in turn and the following operations are performed within the vicinity of the element: (1) edge collapsing; (2) edge splitting; (3) face to edge (Fig. 1(b)) and edge to face (Fig. 1(a)) swapping; (4) edge swapping; (5) node movement.

The mesh optimiser proposes a new neighbourhood mesh configuration (local mesh transformation) which is a small change in the mesh, using one of the above operations. This is accepted as the current mesh configuration if the change in the maximum functional associated with all the elements affected by the change, is negative and less than a smallness parameter $\kappa$. In addition, if the maximum functional value of all the elements that would be affected by a local mesh transformation is less than a certain threshold value $\mathscr{F}_t$ (a value of 0.15 is used here) then this mesh transformation is not considered. The values of $\kappa$ and $\mathscr{F}_t$ are defined bearing in mind the element quality, the most common of which (after mesh optimisation) is about 0.2 measured by Eq. (5), see Section 4. That is,

$$\max_{e' \in \mathscr{E}'}\{F_{e'}\} - \max_{e \in \mathscr{E}}\{F_e\} \leqslant -\kappa \quad \text{and} \quad \max_{e \in \mathscr{E}}\{F_e\} > \mathscr{F}_t \tag{1}$$

for elements in the set $\mathscr{E}$ and element functionals $F_e \; \forall e \in \mathscr{E}$ (which measure the quality of elements $e$) that will be effected by a proposed local mesh transformation; $\mathscr{E}'$ is the set of, and $F_{e'}$ are the functional values of the elements changed or created by the proposed local mesh transformation. A positive non-zero value of $\kappa$ is used to avoid problems with round-off error and also provides some additional control over the CPU requirements of the algorithm ($\kappa = 0.01$ is the default and is used in the applications).



(a) Face to edge (b) Edge swapping

Fig. 1. Digram showing: (a) edge to face and face to edge swapping; (b) edge to edge swapping with four elements.

However, if

$$\max_{e' \in \mathscr{E}'}\{F_{e'}\} - \max_{e \in \mathscr{E}}\{F_e\} < 0 \quad \text{and} \quad \max_{e \in \mathscr{E}}\{F_e\} > \mathscr{F}_t \tag{2}$$

then the proposed local transformation is still accepted if

$$\frac{1}{K'}\sum_{e' \in \mathscr{E}'}\{F_{e'}\} - \frac{1}{K}\sum_{e \in \mathscr{E}} F_e < -\kappa \quad \text{and} \quad \max_{e \in \mathscr{E}}\{F_e\} > \mathscr{F}_t, \tag{3}$$

in which $K$ is the number of elements in $\mathscr{E}$ of the current mesh that will be affected by the proposed transformation, and $K'$ are the number of elements of the proposed mesh that are different from the current mesh. It is convenient to define *Criterion* 1: either Eq. (1) or Eqs. (2) and (3) are satisfied for a local mesh transformation.

During the optimisation each element is visited in turn, sweeping across all the elements of the mesh, in a similar manner to a Gauss–Seidel iteration [20]. When an element is visited all the mesh connectivity optimisation operations above are performed in turn, and out of these the best proposed configuration is accepted, as long as Criterion 1 is met, otherwise the mesh is not altered. If Criterion 1 is not satisfied for operations (1)–(5) listed above, then the element is marked not to be visited again. In this way the optimisation does not repeat, unnecessarily, moves to new mesh configurations. Any new elements that are created (including elements that change shape due to node movement) are added into the end of the list of elements so they will be analysed during the current 'sweep' to see if they and their neighbouring elements can be improved. Once the last element in the element list is visited the mesh optimisation is finished.

A record is maintained of all operations that did not improve (according to Criterion 1) the quality of the local functional so that these operations are not repeated needlessly in neighbouring elements etc. When an element is visited the element is marked not to be visited again and the operations of face to edge swapping are considered for its four faces. If any of these operations are accepted then the old elements are deleted and are replaced by new elements.

Whenever an element is created all nodes and edges of this new element are unmarked. An element can be created partly from existing nodes and edges and partly from new nodes and edges. The absence of the mark says that the corresponding node, edge or element is a candidate for a local transformation. Each node, edge and element has its own mark for this purpose.

If an edge of an element is visited and edge operations (edge swapping, edge collapsing, edge splitting) are proposed, but none are accepted, then this edge is marked not to be considered again for local mesh edge transformation. If an edge is accepted the edge is deleted along with all affected elements which are then replaced by new elements.

When a node is moved all surrounding elements are deleted and replaced by new elements which are added into the end of the element list.

### 2.2. The functional – gauge of mesh quality

The functional used to gauge the quality of the mesh is

$$\mathscr{F} = \|F\|_p, \tag{4}$$

in which $p$ is the norm (the even infinity norm is used), $F$ is a vector of length equal to the number of elements, and the component $F_e$ of vector $F$ associated with element $e$ is

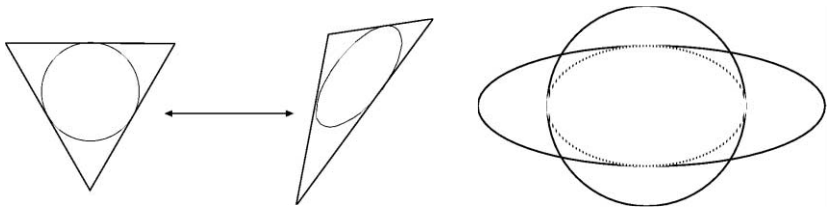
$$F_e = \frac{1}{2}\sum_{l \in \mathscr{L}_e}(\delta_l)^2 + \mu(q_e)^2, \tag{5}$$

where $\mathscr{L}_e$ is the set of edges of element $e$ and $\mu = 1$. The first term in this functional gauges the size of element $e$ and the second term, involving $q_e$, its shape.

In Eq. (5), $\delta_l$ for edge $l$ is defined as

$$\delta_l = r_l - 1,$$

where $r_l$ is the length, with respect to the metric $M(\Omega)$ defined in Euclidean space $\Omega$, of edge $l$.



(a) The mapping (b) Metric superposition

Fig. 2. (a) Mapping between Euclidean metrics; (b) the metric $M_1$ has been distorted to become a circle and the metric $M_2$ has been distorted using the same operations. The maximal inner ellipsoid is shown as a dashed line.

$q_e$ in Eq. (5) is defined as,

$$q_e = \left( \frac{\alpha}{\rho_e} - 1 \right), \tag{6}$$

where $\alpha = 1/(2\sqrt{6})$, $\rho_e$ is the radius of the inscribed sphere of element $e$ in Euclidean space with respect to $M(\Omega)$ (see Fig. 2(a)). The scalar $\alpha$ has been chosen such that $q_e = 0$ for an ideal element (aspect ratio of unity in relation to the metric $M$). The two terms of Eq. (5) work together in the sense that $(q_e)^2$ requires the element $e$ edge lengths to be approximately unity for it to be a good measure of the element shape quality while $\sum_{l \in \mathscr{L}_e} \delta_l^2$ when optimised, will force the edge lengths of element $e$ to be approximately unity with respect to $M$.

### 2.3. Calculating the functional

The length $r_l$ of edge $l$, with respect to $M(\Omega)$, is given by

$$r_l = \int_a^b (s_l(t)^{\mathrm{T}} M(a + s_l(t)) s_l(t))^{1/2} \, \mathrm{d}t, \tag{7}$$

in which $a$ and $b$ are positions of the two nodes associated with edge $l$, $s_l(t)$ is the position in $\Omega$, relative to point $a$, along edge $l$. The magnitude of $s_l(t)$ is the arc length parameter for edges that are straight lines. For the purposes of integration $s_l(t)$ is associated with parameter $t$.

Eq. (7) is used to calculate the distance along an edge. $M(\Omega)$ has been discretised node-wise, and has a linear variation within each element and so $M(\Omega)$ has a linear variation along each edge. All the edges of the meshes used in this work are on straight lines. Therefore, the application of one point quadrature is gauged sufficient to obtain $r_l$, via

$$r_l = (v_l^{\mathrm{T}} M_l v_l)^{1/2}, \tag{8}$$

in which $v_l$ is a vector in a direction along edge $l$ with a magnitude equal to the edge length, and $M_l$ is the edge-centred metric

$$M_l = \frac{1}{2} \sum_{i \in \mathscr{N}_l} M_i.$$

In this equation, the set $\mathscr{N}_l$ contains the two nodes of edge $l$. It is convenient to also define an element-centred metric

$$M_e = \frac{1}{4} \sum_{i \in \mathscr{I}_e} M_i, \tag{9}$$

where $\mathscr{I}_e$ are the set of nodes at the vertices of element $e$. Since $M_i$ is positive definite for all nodes $i$, both $M_l, \forall l$ and $M_e, \forall e$ are guaranteed to be positive definite. The node-wise metrics $M_i$ are calculated by in-

terpolating the metric $M(\Omega)$ from the original mesh onto the latest mesh. Thus, interpolation is required every time a node is created or moved in the mesh optimisation.

In order to calculate the in-sphere radius $\rho_e$ the face areas and volume of element $e$ will first need to be obtained. A triangle of sides length $a, b, c$ has an area of

$$\tilde{A} = (4a^2b^2 - (a^2 + b^2 + c^2))^{1/2}. \tag{10}$$

The values of the distances $a, b, c$ are obtained with respect to the metric $M$ using Eq. (8). The volume of the tetrahedral element $e$ is given by

$$V_e = \frac{1}{6}|\hat{r}_a \ \hat{r}_b \ \hat{r}_c|, \tag{11}$$

where $\hat{r}_a$, $\hat{r}_b$ and $\hat{r}_c$ are the vectors of three edges of element $e$ with a common node. Note that $V_e$ is signed, so it can be used to check if an element has turned 'inside-out' during a local mesh transformation, and that the edge order $a, b, c$ must be consistent for all tetrahedra. The volume $\tilde{V}_e$ of an element in metric space is approximated by

$$\tilde{V}_e = (\det(M_e))^{1/2} V_e, \tag{12}$$

where $V_e$ is the element volume in Euclidean space and $M_e$ is the element averaged metric, given by Eq. (9). The radius (with respect to $M$) of the inscribed sphere of element $e$ can then be calculated using

$$\rho_e = \frac{3\tilde{V}_e}{\tilde{A}_1^e + \tilde{A}_2^e + \tilde{A}_3^e + \tilde{A}_4^e},$$

where $\tilde{A}_1^e, \tilde{A}_2^e, \tilde{A}_3^e, \tilde{A}_4^e$ are the areas of the four faces of element $e$ calculated from Eq. (10).

### 2.4. The metric

The previous section discussed the formation of the functional from a given metric. In this section the metric is derived from a solution field variable and an error norm based on the interpolation error. In 1-D the interpolation error $\epsilon$ is defined as

$$\epsilon = \hat{\gamma} h_e^2 \left| \frac{\partial^2 \psi}{\partial x^2} \right| \tag{13}$$

(see [40]), where $h_e$ is the length of element $e$, $\psi$ is the solution variable and $\hat{\gamma}$ is an $O(1)$ scalar, in fact $\hat{\gamma} = 1$ is used here.

In multi-dimensions

$$\epsilon = \hat{\gamma} v^{\mathrm{T}} |H| v$$

for some Hessian $H$ (assumed independent of Euclidean space, in this case) and vector $v$ representing the desired direction and length. The Hessian $H$ can be replaced by a modified Hessian $Q$ (see later on in this section).

For a required (specified) interpolation error $\hat{\epsilon}$ a metric $\hat{M}$ is defined such that an element size (length) is unity if it has the desired interpolation error $\hat{\epsilon}$. Thus,

$$\hat{M} = \frac{\hat{\gamma}}{\hat{\epsilon}} |H|. \tag{14}$$

The length of vector $v$ with respect to $\hat{M}$ is then calculated from

$$\|v\|_{\hat{M}} = (v^{\mathrm{T}} \hat{M} v)^{1/2},$$

for spatially independent $\hat{M}$. Thus, in 1-D the length $d_e$ with respect to $M$ of an element $e$ of length $h_e$ is

$$d_e = \left( h_e^2 \frac{\hat{\gamma}}{\hat{\epsilon}} \left| \frac{\partial^2 \psi}{\partial x^2} \right| \right)^{1/2}.$$

The metric $\hat{M}$ must be modified to take into account maximum and minimum element sizes. The result is a metric $M$ defined as

$$M = V^{\mathrm{T}} \tilde{\Lambda} V, \tag{15}$$

where $V$ (rotation matrix) contain the eigenvectors of the metric $\hat{M}$ given by Eq. (14). The eigenvalues $\tilde{\lambda}_j$, $\forall j \in \{1,2,3\}$ of $M$ are based on the eigenvalues $\lambda_j$, $\forall j \in \{1,2,3\}$ of metric $\hat{M}$ but modified as follows:

$$\tilde{\lambda}_j = \max\left\{ \lambda'_j, \frac{1}{a^2} \max_{i=1}^{3} \lambda'_i \right\} \quad \forall j \in \{1,2,3\} \tag{16}$$

with

$$\lambda'_j = \min\left\{ \frac{1}{h_{\min}^2}, \max\left\{ |\lambda_j|, \frac{1}{h_{\max}^2} \right\} \right\} \quad \forall j \in \{1,2,3\}, \tag{17}$$

in which $a$ is the maximum aspect ratio (100 is the default and has been used in Section 4). The maximum aspect ratio of an element is thus controlled through the metric by capping the larger size(s) associated with an element. This corresponds to increasing the smaller eigenvalue moduli, in Eq. (16). Also $h_{\min}$ and $h_{\max}$ in Eq. (17) are the minimum and maximum element sizes (more specifically, edge lengths) tolerated in the mesh. Since $\tilde{\lambda}_j$, $\forall j \in 1,2,3$ are positive, $M$, given by Eq. (15), is positive definite.

The Hessian, for a field $\psi(\Omega)$, in 2-D is

$$H = \begin{pmatrix} (\partial^2\psi/\partial x^2) & (\partial^2\psi/\partial x \partial y) \\ (\partial^2\psi/\partial y \partial x) & (\partial^2\psi/\partial y^2) \end{pmatrix}$$

and similarly in higher dimensions $H = \nabla^{\mathrm{T}} \nabla \psi$.

In order to calculate an error norm it is often convenient to normalise the Hessian with the magnitude of the solution $\psi$ so that the resulting error norms are not biased towards the larger values of $|\psi|$. This takes the form of

$$Q = \frac{1}{\max(|\psi|, \psi_{\min})} H \tag{18}$$

for some positive $\psi_{\min}$. The use of $\psi_{\min}$ in this equation avoids problems with round-off error (dividing by zero or a small number) and also provides control or a cut-off point for the magnitude of $\psi$ below which fine mesh resolution is deemed unnecessary.

### 2.5. Hessian calculation

Hessian calculation can be troublesome, particularly due to accuracy considerations near boundaries. In [13] an explicit diffusion was introduced in the Hessian calculation to suppress spurious oscillations resulting from a weak finite element formulation. However, the weak formulation requires some additional information to yield the second derivative at and normal to, the boundaries.

Galerkin methods are repeatedly applied to calculate the first derivatives, so at a node $i$:

$$\left. \frac{\partial \psi}{\partial x} \right|_i \approx q_{x_i} = L_i^{-1} \int N_i (\partial \psi / \partial x) \, \mathrm{d}V$$

and similarly for $q_{y_i}$, $q_{z_i}$, thus

$$(\partial\psi/\partial x) \approx q_x = \sum_j N_j q_{x_j},$$

etc. $N_j$ are the finite element basis functions and $L$ is the row summed lumped mass matrix, see [40]. The second-order terms that make up the Hessian

$$H_i = \begin{pmatrix} q_{xx_i} & q_{xy_i} & q_{xz_i} \\ q_{yx_i} & q_{yy_i} & q_{yz_i} \\ q_{zx_i} & q_{zy_i} & q_{zz_i} \end{pmatrix}$$

centred on node $i$ are calculated by for example

$$q_{xx_i} = L_i^{-1} \int N_i \frac{\partial q_x}{\partial x}\, \mathrm{d}V, \quad q_{xy_i} = L_i^{-1} \int N_i \frac{\partial q_x}{\partial y}\, \mathrm{d}V.$$

Accuracy is still reduced at boundaries because of the reduced size (roughly half the size of the stencil away from the boundaries) of the supporting stencil.

### 2.6. Superposition of metrics

Often one requires to solve problems with two or more solution fields, each field having its own metric, reflecting its own required mesh resolution. In this formulation each field can also have its own error requirements, which will in turn be reflected in the metric associated with that field.

For simplicity, it is important to superimpose the metrics so that the mesh adaptivity method has only to deal with a single metric field. The alternative would be to calculate the distance associated with each metric (or solution field) and determine some representative distance from these. We take the approach of using the finest mesh resolution dictated by any of these fields. Thus, the minimum distance would be the representative mesh resolution distance. Using this philosophy, two metrics can be superimposed by deforming one of the metrics so it can be represented by the unit sphere (that is the identity matrix). The second metric is deformed using the same operations, then the superimposed metric is chosen to be the maximal ellipsoid which fits inside both the unit sphere and the ellipsoid, see Fig. 2(b). This maximal ellipsoid or metric is then transformed back to the original system for use with the mesh adaptivity algorithm. The transformation of a vector $v'$ in Euclidean space associated with metric $\mathscr{M}$ to $v$ in Euclidean space, is $v' = \Lambda^{-1/2}\mathscr{V} v$ and will be used repeatedly to superimpose the metrics.

For the deformation of a metric into a sphere, the metric is chosen which requires the least distortion, measured by

$$\frac{\max_{i\in\{1,2,3\}} \lambda_i}{\min_{i\in\{1,2,3\}} \lambda_i}, \tag{19}$$

in which $\lambda_i$ are the eigenvalues of the metric. Suppose the metrics $\mathscr{M}_1$, $\mathscr{M}_2$ are to be superimposed, and $\mathscr{M}_1$ is the least distorted, defined by Eq. (19), of the two metrics and

$$\mathscr{M}_1 = \mathscr{V}_1^{\mathrm{T}} \Lambda_1 \mathscr{V}_1, \quad \mathscr{M}_2 = \mathscr{V}_2^{\mathrm{T}} \Lambda_2 \mathscr{V}_2.$$

$\mathscr{M}_2$ is mapped to a space in which $\mathscr{M}_1$ is represented by a unit sphere and so becomes,

$$\hat{\mathscr{M}}_2 = \Lambda_1^{-1/2} \mathscr{V}_1 \mathscr{M}_2 \mathscr{V}_1^{\mathrm{T}} \Lambda_1^{-1/2} = \mathscr{Q}^{\mathrm{T}} \Lambda \mathscr{Q},$$

in which $\mathscr{Q}, \Lambda$ are the eigenvectors and eigenvalues of $\hat{\mathscr{M}}_2$. Now the eigenvalues $\Lambda$ (or $\lambda_i, \forall i \in \{1,2,3\}$) are limited using $\hat{\lambda}_i = \max\{1, \lambda_i\}$ so that the resulting metric $\hat{\mathscr{M}}$ (with eigenvalues $\hat{\Lambda}$ or $\hat{\lambda}_i, \forall i \in \{1,2,3\}$) defines the maximal inner ellipsoid, see Fig. 2(b). $\hat{\mathscr{M}} = \mathscr{Q}^{\mathrm{T}} \hat{\Lambda} \mathscr{Q}$ is transformed back to the original space to obtain the final metric

$$M = \mathscr{V}_1^{\mathrm{T}} \Lambda_1^{1/2} \hat{\mathscr{M}} \Lambda_1^{1/2} \mathscr{V}_1.$$

Superimposing $n > 2$ metrics is performed by repeating the above procedure, $G(\mathscr{M}_1, \mathscr{M}_2)$, for each additional metric

$$M = G(\cdots G(G(\mathscr{M}_1, \mathscr{M}_2), \mathscr{M}_3) \cdots \mathscr{M}_n).$$

### 2.7. Predicting the number of elements

It is important to be able to predict the number of elements that will be generated for a given metric field $M(\Omega)$, so that the mesh adaptivity and finite element solution CPU times (or memory requirements) can in turn be predicted. If mesh adaptivity is anticipated to produce an excessive number of elements then the metric $M(\Omega)$ can be scaled such that a maximum number of elements is not exceeded by the mesh optimiser.

Now in metric space, $M(\Omega)$ , the optimal element volume is $\gamma = (1/\sqrt{72})$. Assuming that all the elements, after optimisation of the mesh, have a volume $\gamma$ then the number of elements $E_{\text{new}}$ after optimisation, can be calculated from

$$E_{\text{new}} = \frac{\sum_{e=1}^{E_{\text{old}}} \tilde{V}_e}{\gamma}, \tag{20}$$

where $E_{\text{old}}$ is the number of elements in the original mesh to be adapted and so $e$ loops over the original mesh elements in the above equation. Now if $E_{\text{new}} \geqslant \theta E_{\max}$ then a new metric $M_{\text{new}}(\Omega) = \beta M(\Omega)$ replaces $M(\Omega)$ for use in the mesh adaptivity algorithm for some scalar $\beta$. None unity $\theta$ acknowledges the fact that Eq. (20) is not exact – we have found that $\theta \approx 0.85$. In Eq. (20), $\tilde{V}_e$ is the volume, given by Eq. (12), of element $e$ of the original mesh with respect to $M(\Omega)$. Since the required number of elements is $E_{\max}$, the scalar $\beta$ must satisfy

$$\theta E_{\max} = \frac{\sum_e (\det(\beta M_e))^{1/2} V_e}{\gamma} = \frac{\sum_e \beta^{3/2} (\det(M_e))^{1/2} V_e}{\gamma}$$

and thus

$$\beta = \left( \frac{\gamma \theta E_{\max}}{\sum_e (\det(M_e))^{1/2} V_e} \right)^{2/3}.$$

### 2.8. Edge collapsing

This operation deletes all elements that belong to an edge and collapses the two nodes of the edge, usually to the centre of the edge. It may be worthwhile to simultaneously optimise the new node position, even if this is restricted to being along the edge to be collapsed and possibly further restricted to three points: at one of the two nodes; at the other node; at the mid-point of the edge. However, to reduce the size of the number of configurations to search through and the CPU requirements of mesh optimisations, the edge is collapsed to a node at its mid-point when possible. There are situations when an edge can only be collapsed to one of the nodes, for example, if one of the nodes of the edge is used to define the surface of the domain or an interior surface bounding regions of different material properties. If both nodes of an edge lie on one of these surfaces, and the surface will change shape with edge collapsing then collapsing this edge is not permitted.

### 2.9. Edge splitting

Edge splitting involves inserting a node at the centre of an edge and generating new elements surrounding the split edge. Again, it is possible to simultaneously optimise the new node position as well as split the edge. However, this is not done here, again, for computational efficiency.

### 2.10. Edge and face-edge swapping

Face to edge swapping can be performed if two tetrahedra share a common face and their combined interior is convex. This is achieved by introducing an edge between the two nodes that are not shared, producing three tetrahedral elements, see Fig. 1(a). Conversely, it is easy to see that the inverse operation of edge to face swapping is also possible.

In general, an edge of the mesh can be removed and the surrounding $S$ elements replaced by $2S-4$ elements. In the edge to face example, $S=3$, two elements replace three. For $S$ greater than 4, more elements are created than are removed. This, and the fact that regions with edges surrounded by many elements are of poor quality, suggests that there is an upper bound, above which edge swapping is rarely worthwhile. This was determined as $S>7$ in [16]. The number of possible ways that elements can be re-connected after deleting an edge is shown diagrammatically in [16] and is given by

$$C_{\mathrm{r}}=\frac{(2S-4)!}{(S-1)!(S-2)!} \tag{21}$$

(see [18]). An optimisation method would therefore have to search through a large number of connectivity permutations for large $S$. Thus, to make the algorithm efficient, $S$ has been limited to 4, see Fig. 1(b). Four is the minimum value of $S$ that leads to good quality optimised meshes. If an edge not on the surface of the domain has four surrounding elements it has only two possible re-connections, see Fig. 1(b). This edge swapping is unique in that it is its own inverse. On the surface of a domain with four coplanar points belonging to two joined triangles on neighbouring tetrahedra, edge swapping operations are performed between the two possible re-connections.

### 2.11. Node movement

Node movement, or smoothing, is often used to improve mesh quality, see [18]. This is perhaps most commonly achieved by visiting each node and re-positioning it to the centroid of all the surrounding nodes (nodes that share an edge with visited node). In general, the centroid is calculated with respect to a metric. This is analogous to Laplacian smoothing but in a distorted space with a diffusion tensor reflecting this distortion/metric.

In this work, the objective of movement of an individual node is to move it to a position which minimises the worst element quality given by Eq. (5), of an element surrounding this node. Thus, the implied functional is the even infinity norm of all the functionals associated with these surrounding elements. Unfortunately, this functional is not amenable to differentiation and therefore its roots cannot be found readily by methods that rely on differentiation. However, if this functional is approximated with a functional that has the differential properties required (e.g., using the two-norm instead of the infinity norm), then the large number of non-linear optimisation methods in the literature [7] becomes applicable, e.g., Newton Raphson iteration, gradient methods with line searches. However, non-differential methods can also be effective, e.g., choosing an optimal element for each face of the ball obtained by removing the node. Some average of the new node positions associated with each optimised element is then adopted as the node position, see [18].

In this work, the aim is to equate, without decreasing the local mesh quality, the edge lengths given by $r_l=(v^{\mathrm{T}}M_l v_l)^{1/2}$ of the set of edges $\mathscr{L}_i$ connected to node $i$. This can be achieved by minimising

$$E_i=\frac{1}{2}\sum_{l\in\mathscr{L}_i} r_l^2=\frac{1}{2}\sum_{l\in\mathscr{L}_i} v_l^{\mathrm{T}}Mv_l. \tag{22}$$

If this is differentiated w.r.t the position $p^i$ of the node $i$ under consideration and using $v_l=p^i-y_l^i$, then

$$\frac{\partial E_i}{\partial p^i}=\sum_{l\in\mathscr{L}_i} M_l v_l,$$

where $y_l^i$ is the node position of the node connecting node $i$ via edge $l$. Thus, at the minimum given by Eq. (22)

$$\sum_{l\in\mathscr{L}_i} M_l p^i = \sum_{l\in\mathscr{L}_i} M_l y_l^i - q^i = 0,$$

with $q^i = \sum_{l\in\mathscr{L}_i} M_l y_l^i$. Let $A^i = \sum_{l\in\mathscr{L}_i} M_l$, then $p^i$ can be calculated from $A^i p^i = q^i$. This equation is amended by introducing a diagonal matrix $D^i$, chosen to achieve diagonal dominance and ensure insensitivity to round-off error when solving the resulting equation

$$(D^i + A^i)(p^i - \hat{p}^i) = q^i - A^i \hat{p}^i,$$

in which $\widehat{p^i}$ is the previous position of node $i$ and

$$D^i_{jk} = \begin{cases} \max\{A^i_{jj}, (1+\sigma)\sum_{m=1,m\neq j}^{3} |\, A^i_{jm}\,|\} & \text{if } j = k, \\ 0 & \text{if } j \neq k. \end{cases} \tag{23}$$

$\sigma = 0.01$ is used in this work. Relaxation of $p^i$ using $x^i = wp^i + (1-w)\hat{p}^i, w \in (0,1]$ is applied to Eq. (24) to obtain

$$(D^i + A^i)(x^i - \hat{p}^i) = w(q^i - A^i \hat{p}^i), \tag{24}$$

which is solved for the new node position $x^i$ ($w = 1/2$ is used here).

It is important to use relaxation because the functional Eq. (22), used for node movement, is not entirely consistent with minimising the worst element quality given by Eq. (4), among all the surrounding elements.

*2.11.1. Node movement constrained to a surface or curve*

A similar procedure to that described above is used to move nodes constrained to an internal or external surface of the domain. Since the surface mesh is comprised of triangles – a series of plane surfaces – two cases of node movement need to be considered: (i) node movement on a tangent plane to the surface; (ii) node movement along a line between a group of triangles.

Along a tangent plane, case (i), the new node position is some linear combination of two orthogonal vectors in this plane $u_1^i$ and $u_2^i$ say. If $u_1^i$ is a unit vector along an edge of one of the triangles containing node $i$ to be moved, then $u_2^i = (1/u_i^*)u_1^i \times (u_1^i \times t_2^i)$ where $t_2^i$ is measured along another edge of the same triangle and $u_*^i$ is a scalar chosen to make $u_2^i$ a unit vector. Suppose

$$p^i = \alpha_1^i u_1^i + \alpha_2^i u_2^i + \hat{p}^i$$

is the new proposed node position and the new vector of an edge $l$, connected to node $i$, is given by

$$v_l^i = p^i - x_l^i = \alpha_1^i u_1^2 + \alpha_2^i u_i^2 + \hat{p}^i - x_l^i.$$

Using this equation in Eq. (22) and introducing the matrix $U^i = (u_1^i \;\; u_2^i)$ it is easy to see that

$$E_i = \frac{1}{2}\left[\alpha^{i\mathrm{T}} \hat{M}^i \alpha^i + g^{i\mathrm{T}} \alpha^i + \alpha^{i\mathrm{T}} g^i\right], \tag{25}$$

with

$$g^i = \sum_{l\in\mathscr{L}_i} U^{i\mathrm{T}} M_l (x_l^i - \hat{p}^i), \quad \alpha^i = \begin{pmatrix} \alpha_1^i \\ \alpha_2^i \end{pmatrix}$$

and $\hat{M}^i = U^{i\mathrm{T}} \sum_{l\in\mathscr{L}_i} M_l U^i$. Setting the first differential of Eq. (25) to zero and introducing relaxation via $\alpha_c^i = w\alpha^i + (1-w)\hat{\alpha}^i$, the following equation for $\alpha_c^i$ is obtained:

$$(D^i + \hat{M}^i)\alpha_c^i = -wg^i,$$

in which $D^i$ is a matrix used to ensure a certain level of diagonal dominance, (see Eq. (23)). The new node position $x^i$ of node $i$, is then calculated from

$$x^i = U^i \alpha_c^i + \hat{p}^i - x_l^i.$$

When the node movement has no constraints then

$$U^i = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

for all nodes $i$ not on the boundary or internal surface of the domain. In addition, if the node $i$ is to be constrained to lie on a line then $x^i = \alpha^i_{c_1} u^i_1 + \hat{p}_i$ which is the equation of the line with $u^i_1$ the unit vector tangent to the line and $U^i = u^i_1$.

### 2.12. Data structures

The mesh optimisation method has been implemented with a memory manager consisting of two linked lists containing node, edge and element information. This allows the efficient deletion/insertion of elements, edges and nodes from/to the mesh as the optimisation proceeds. It guarantees efficient use of memory resources and is one of the novel features of the adaptive method. One linked list contains nodal information (including co-ordinates and metrics) and one contains element and edge information. Other data structures for meshing are described in [15,18,27].

#### 2.12.1. Description

Both linked lists are made of data "cells", which contain information about the item (or part of item) it references, as well as pointers to the next item (or part of item), and the previous item (or part of item).

Each nodal cell, see Fig. 5, contains information for only one node regarding the co-ordinates of the node, the metric for the node, the type of node (i.e., whether it is on an immutable 'geometry' surface, whether it is on a geometry edge) as well as the pointers to the locations in the list for the next and previous nodes. In Fig. 5 the part of the data cell labelled 'contnr' stores the element of the original mesh in which the node is contained (used for interpolation). Other parts of the data cell contain the node number assigned to it and node movement and deletion constraints associated with internal or external surfaces of the domain.

Information for an edge (Fig. 3) is stored within a single cell of the second linked list, with two references into the nodal list for the nodes at each end of the edge, as well as geometry information (similar to the nodes), and the pointers to the next and previous item in the list. The flags in this cell indicate whether this is an edge or element cell and assuming it is an edge the flags contain geometrical information about the edge, such as whether or not it is used to define the shape of the domain or an internal surface and whether this edge can be deleted. One flag indicates whether the edge has been visited previously and the optimisation chose not to alter the elements surrounding the edge. The flags also indicate if the data cell is currently empty or not. The edge length with respect to $M$ is also contained in the cell (denoted as 'functnl' in Fig. 3) along with some spare memory used in the mesh optimisation routine.

Information for an element (Fig. 4) spans three cells of the second list, with four references into the same list for the neighbouring elements, six references for the edges that make up the element, and the usual next/previous pointers. Note that the first cell of an element has a 'next' pointer that points to the cell containing the second part of information for that element, which itself has a 'next' pointer that references the third

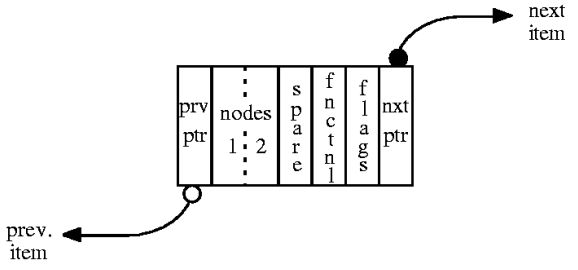



Fig. 3. Data structure for edge information, part of the combined edge-element linked list.

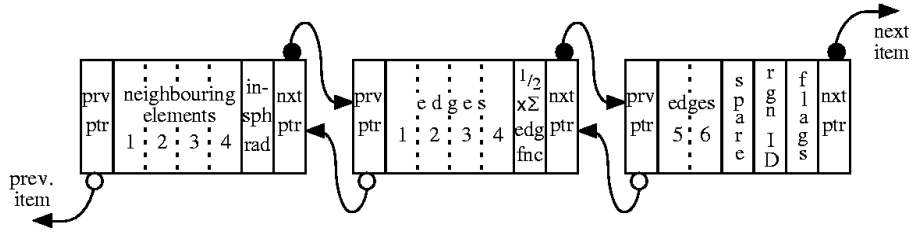



Fig. 4. Data structure for element information, part of the combined edge-element linked list.

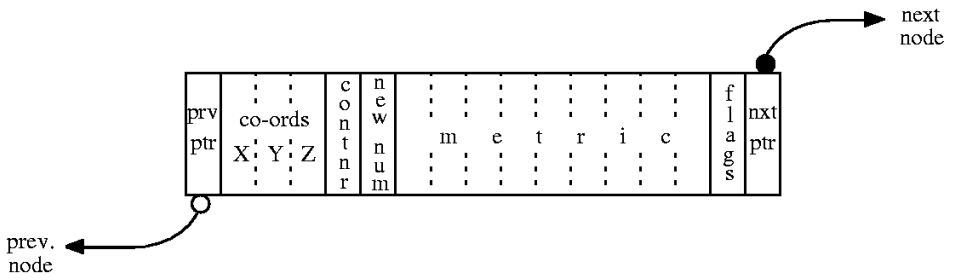



Fig. 5. Data structure for node information, part of node linked list.

and final part of information for that element. The flags in an element cell (see Fig. 4) contain similar information to the flags associated with an edge. It also includes information about: flags; a region number which is used to recognise internal surfaces of the domain; the in-sphere radius w.r.t $M$ of the element; and the sum $1/2\sum_{l\in\mathscr{L}_e}(\delta_l)^2$ which forms part of Eq. (5).

The final cell containing a valid item (or part of item) will point on to an empty (free) cell. Each empty cell then points on to further empty cells, until all the cells of the list have been visited – the last cell has an 'end marker' for its next pointer. Fig. 5 gives an example of a list containing edges, elements and free space. In addition there is a special 'start pointer' which gives the location in the list of the first item, and a 'free pointer', which points to the first free cell.

*2.12.2. Adding and removing an item*

A new item is added in the cell pointed to by the free pointer (in the case of an element, it spans two further free cells). The next and previous pointers need no updating, but the free pointer must be updated to point to the next free cell in the list (which is pointed to by the newly-filled cell).

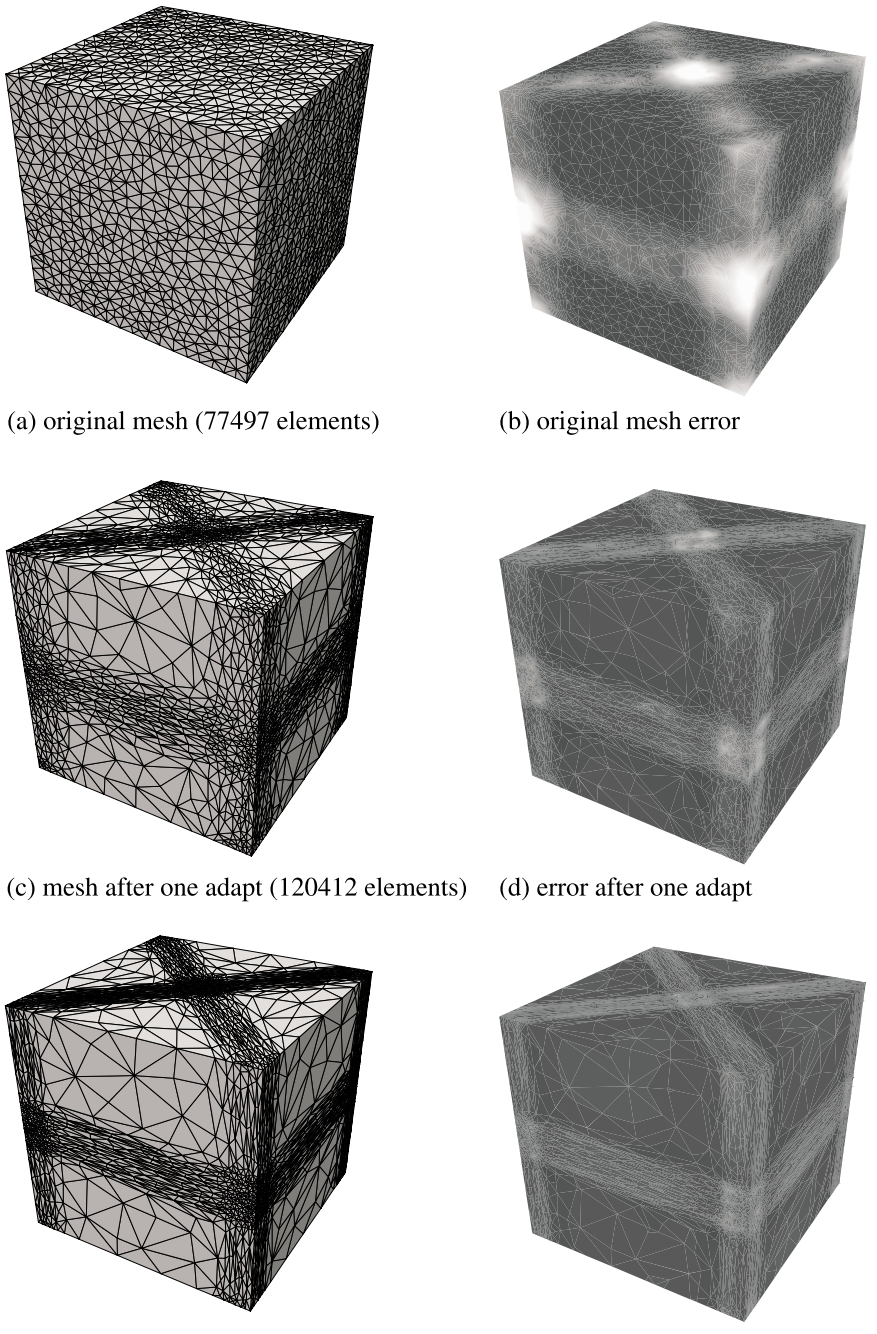
An item is removed by severing its place in the list (i.e., changing the previous and next cells to point around it), and inserting it into the used/free interface – i.e., changing its next pointer to point to the current free pointer, and its previous pointer to point to the last filled cell (given by the previous pointer of the cell pointed to by the current free pointer). The free pointer is then updated to point to the newly-freed cell. In the case of element removal, three cells are freed and the first one is pointed to by the current free pointer, with the other two following on in the same order as for the element. It should be noted that if the item being removed is the first in the list (i.e., pointed to by the start pointer) then the start pointer must be updated to point to the next item (unless the removed item is the only one in the list – which should never happen).

This technique for addition and removal has the advantage of guaranteeing that just-freed cells are the first to be re-used. Thus, the lower parts of the memory or linked list are used first. Since a removed item is pointed to by the free pointer, it will be the next to be filled if an item is added.

## 3. Performance analysis of the mesh adaptivity method

In this section, we analyse the performance of the adaptivity method by applying it to a relatively simple problem. This problem was designed to highlight the anisotropic meshing capabilities of the method. The

domain is a box and the mesh should adapt to follow the features of three fields containing three perpendicular ridges, as described below. Interpolation errors are set to be the same for each field and the three metrics associated with these are superimposed to obtain one metric on which the mesh is adapted. The initial mesh has five elements, which is then optimised, based on a homogeneous isotropic metric, to produce the mesh shown in Fig. 6(a) from which the adaptivity was initiated. The simple box problem has



(a) original mesh (77497 elements) (b) original mesh error

(c) mesh after one adapt (120412 elements) (d) error after one adapt

(e) mesh after two adapts (128752 elements) (f) error after two adapts

Fig. 6. Evolution of mesh and solution errors to accurately represent the three ridges. The maximum errors in (b), (d) and (f) are 31.70, 10.19 and 4.84, respectively. The minimum errors in (b), (d) and (f) $1.45 \times 10^{-2}$, $3.31 \times 10^{-2}$ and $4.02 \times 10^{-2}$, respectively.

been adapted with a minimum and maximum element sizes of 0.02 and 1.0, respectively. In addition, the maximum aspect ratio has been controlled, through the metric, to be 100.

Each ridge is defined by

$$f_k = a(q_k - b^2) + c \quad \text{with } b = 0.07,\ c = 1, \tag{26}$$

in which the three planes $q_k = 0$ are defined by

$$q_1 = (x-y)^2, \quad q_2 = (1-x-y)^2, \quad q_3 = \left(\frac{1}{2} - z\right)^2,$$

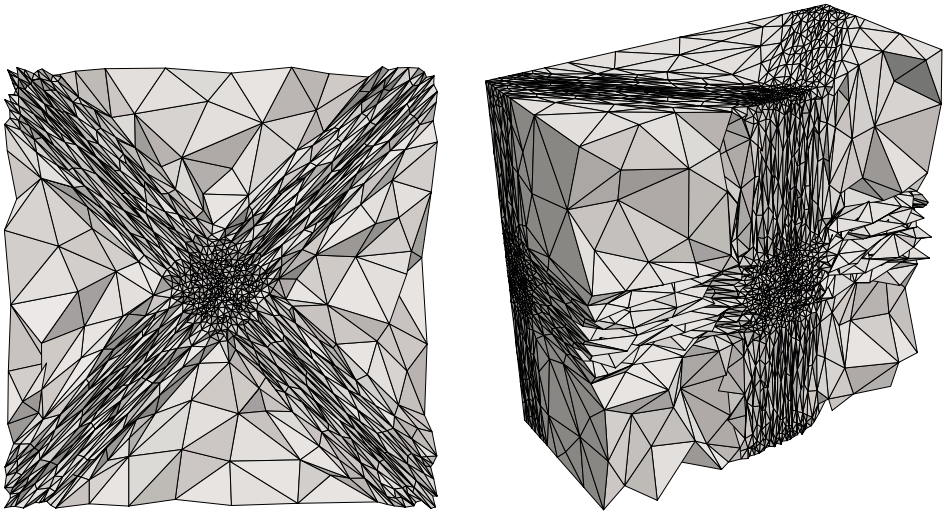
where

$$a = \begin{cases} 75 & \text{if } q_k \leqslant 0.07, \\ \frac{1}{3} & \text{otherwise} \end{cases}$$

so $\sqrt{q_k}$ is the perpendicular distance from the given plane $k$. These fields are held on the meshes shown in Figs. 6(a), (c), and (e) and are exact at the nodes but have a linear variation within an element. This inaccuracy leads to inaccuracy in the Hessian used to calculate the errors, and thus, after one adaptation of the mesh, Fig. 6(c), the mesh resolution does not accurately reflect the fields given by Eq. (26), but has an improved (relative to the mesh shown in Fig. 6(a)), capacity to do so. Again the fields, given by Eq. (26), on this new mesh are more accurate and thus the mesh can more readily represent the Hessian which is used to help guide mesh refinement, to obtain the mesh after the second adaptation, see Fig. 6(e).

Notice that, where the ridges cross on the surface of the domain, Fig. 6(c) and (e) the aspect ratio of the elements is about unity and that the error measure, which is simply the volume of the element in metric space (see Eq. (12)) normalised with the ideal element volume $(1/\sqrt{72})$, shown on the surface of the domain in Figs. 6(b), (d) and (f) becomes more uniform with each mesh adaptation.

Figs. 7(a) and (b) show the mesh, after the final adaptation, inside the domain on horizontal and vertical planes through the centre of the domain, respectively. Fig. 7(a) clearly shows the anisotropy of the mesh along the ridges and an area where the mesh has aspect ratio of unity (on the plane) where the ridges cross. In Fig. 6(b) it is seen that the mesh in this area (of unit aspect ratio on the plane shown in Fig. 6(a)) is highly



(a) mesh inside domain (b) mesh inside domain

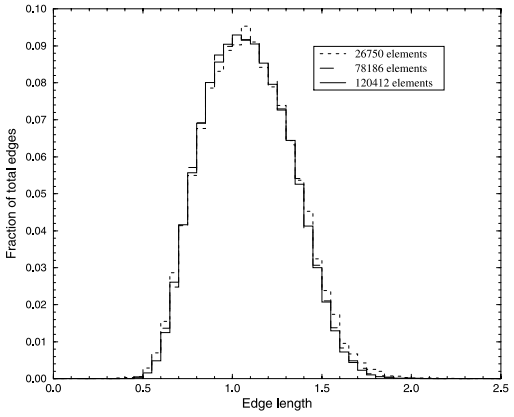
Fig. 7. Adapted mesh of a cube to follow anisotropically three ridges which are perpendicular to each other. The adapted mesh has 128752 elements and 24449 nodes. (a) and (b) show the mesh inside of the domain with (a) central horizontal cut, top view (b) central vertical cut.

anisotropic in the perpendicular direction. These mesh features demonstrate the effectiveness of the metric supposition method.

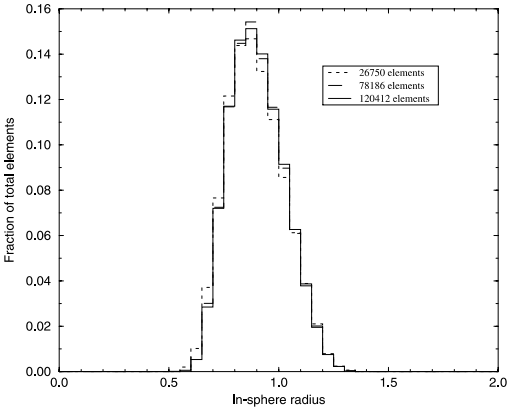
The mesh has been adapted, from that shown in Fig. 6(a), once to resolve the three fields with three different interpolation error requirements. Fig. 8(a) shows the resulting distribution of edge lengths and Fig. 8(b) the normalised in-sphere radii relative to the metric, which is our gauge of element shape quality. A unit normalised in-sphere radius is optimal in which the in-sphere radius has been normalised with the optimal radius $(1/\sqrt{72})$. The distribution of the overall element qualities (judged by Eq. (5)) is seen in Fig. 8(c). Notice that the distributions, Figs. 8(a)–(c) have converged to approximately the same curve.

One of the five original tetrahedra, that make up the initial domain, and that has a face on one of the faces of the cube was given a different region ID which means that the shape of this region must be maintained throughout mesh adaptation. The resulting mesh optimisation has difficulty in conforming to the initial boundary of the domain as well as optimising their element qualities. However, the ability to do this is important when optimising meshes to conform to geometrically complex shapes or internal boundaries of the domain. Thus, the element qualities are generally lower than those for the case with no internal boundaries, see distributions in Fig. 8(d). This was demonstrated for the intermediate interpolation error requirements of the three mesh adaptations whose results are shown in Figs. 8(a)–(c).

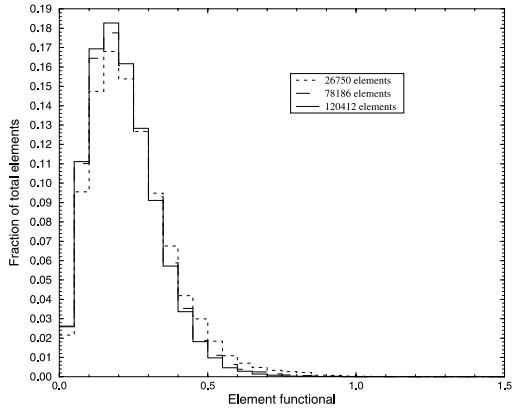
The parameters dictating the speed of the optimisation $(\mathscr{F}_t, \kappa)$ (see Section 2.2), were increased from the default values $(0.15, 0.01)$ to $(0.25, 0.05)$ to speed up the mesh optimisation. Mesh optimisations were then



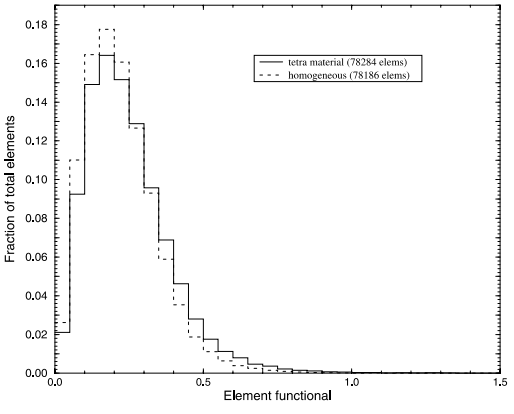

(a) edge length




(b) normalised element in-sphere radius

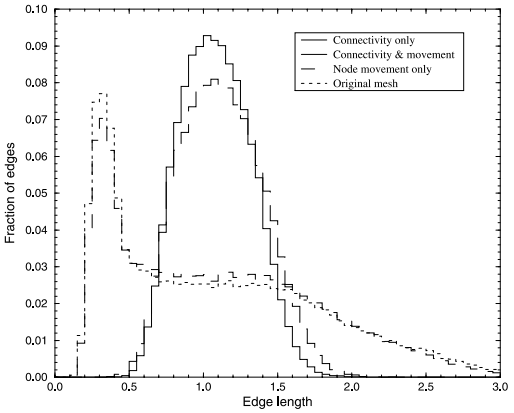



(c) element functionals

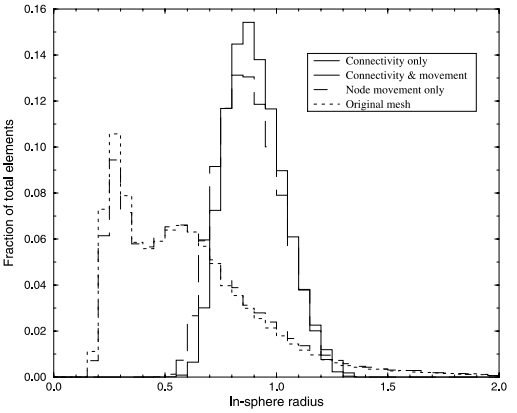



(d) element functionals

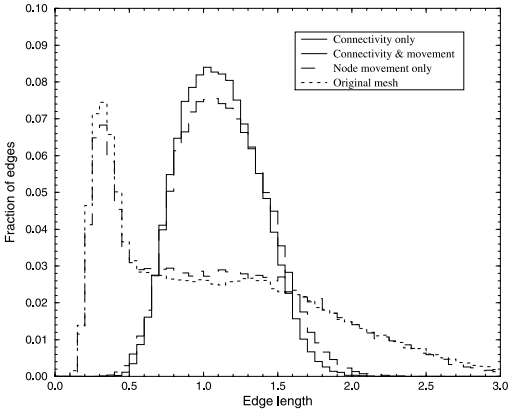
Fig. 8. Mesh adapted to more closely follow the three fields each containing a ridge like structure. Three meshes have been obtained with varying interpolation error requirements: (a) histogram of the edge lengths in metric space (relative to the ideal length); (b) the in-sphere radius distribution (used to define the quality of an element); (c) the overall functional judging the quality of the elements. (d) is the same as (c) except that a tetrahedral region has been placed within the domain which the elements must conform to.



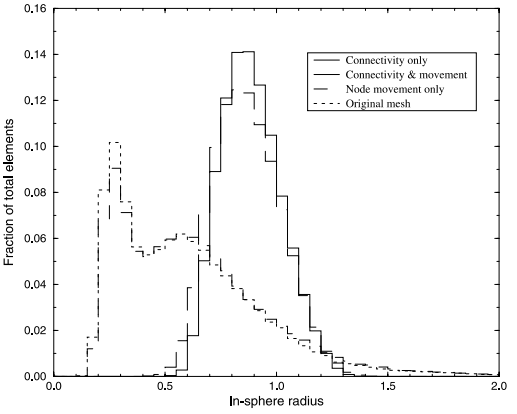

(a) edge length (slow optimisation)




(b) element shape (slow optimisation)

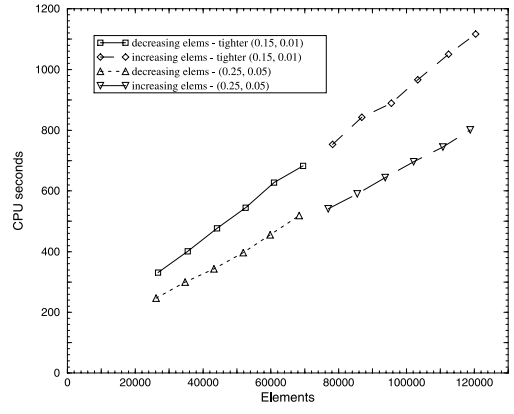



(c) edge length (fast optimisation)

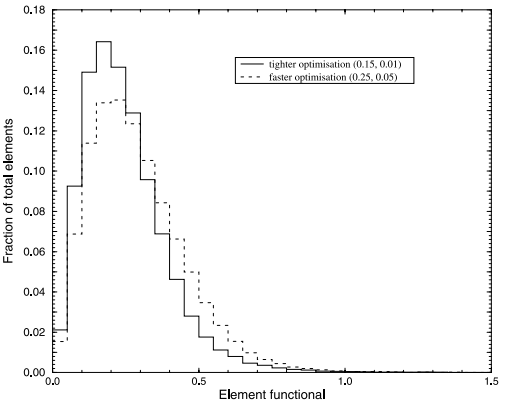



(d) element shape (fast optimisation)

Fig. 9. (a) and (b) Slow high quality optimisation ($\mathscr{F}_t = 0.15$, $\kappa = 0.01$); and fast (c) and (d) low quality optimisation ($\mathscr{F}_t = 0.25$, $\kappa = 0.05$). The uniform mesh containing three ridges has been adapted to more closely follow the three fields. Three mesh adaptations were performed with: node movement only; mesh connectivity only; both mesh connectivity and node movement. The original mesh statistics are also provided for comparison.




(a) CPU time




(b) mesh quality comparison

Fig. 10. (a) CPU requirements to adapt the mesh from a uniform mesh of 77,497 elements to a mesh adapted to three ridges as the interpolation error is varied ($\mathscr{F}_t = 0.25$, $\kappa = 0.05$); (b) difference in the quality of the mesh for fast and slow optimisations ($\mathscr{F}_t = 0.15$, $\kappa = 0.01$).

performed for the intermediate error requirement. The resulting edge and in-sphere radii are shown in Fig. 9 with: mesh connectivity operations only; the default algorithm with both mesh connectivity and node movement operations; node movement only; and for the original mesh. Clearly the mesh quality is far superior with both mesh connectivity and node movement.

A comparison of the distribution of the element functionals (given by Eq. (5)) for slow and fast optimisations are shown in Fig. 10(b) for the intermediate error requirements, along with the CPU requirements of the algorithm, Fig. 10(a), as the number of elements (or required interpolation error) is varied. It should be noticed that the CPU requirements of mesh adaptivity varies approximately linearly with the number of elements produced. This linearity in CPU requirements is expected to carry over to general steady state and transient problems. In these cases the CPU time for mesh optimisation is reduced because the starting mesh is often of good quality, unlike the original mesh shown in Fig. 6(a).

## 4. Applications of mesh optimisation and adaptivity

As mentioned earlier, the mesh optimisation/adaptivity method presented above is of general applicability. To demonstrate this we describe next its application to transient CFD problems and a steady-state radiation transport problem.

### 4.1. Transient incompressible flow

The primitive variable form of the Navier–Stokes equation is given by

$$\frac{\mathrm{D}u}{\mathrm{D}t} = v\nabla^2 u - \nabla p, \tag{27}$$

$$\nabla \cdot u = 0, \tag{28}$$

where $p$ is the pressure, $u$ the velocity containing three components and $D$ is the total derivative.

The well-known difficulty in satisfying the Ladyzhenskaya–Babuska–Brezzi condition [5,8], associated with mixed formulations can be circumvented by using a shared pressure/velocity mesh and introducing explicitly, a stabilisation term [4,22,34,35]. This stabilisation term takes its motivation from the success of the Poisson equation approach, see [1,2,21], for solving the incompressible Navier–Stokes equations. This approach, which involves deriving and solving an elliptic differential equation for pressure, has been demonstrated to introduce implicitly a fourth-order (in pressure) stabilisation term into the continuity equation, the magnitude of which depends linearly on a product of the square of the element/cell length and the time step size [35]. In the incompressible transient fluids problems solved here a consistent discretisation (assemble both continuity and momentum equations then solve the resulting discretised equations) is used and therefore the fourth-order stabilisation term is introduced explicitly into the continuity equation. This approach is a natural beneficiary of mesh adaptivity since its main drawback is the fact that the continuity equation can be poorly satisfied in regions with a large spatial variation of pressure. But since the offending term depends on the square of the mesh spacing, adapting the mesh to follow variations in pressure, among other things, ensures that this term remains small.

The momentum equations (27) and (28) are discretised by using the Crank–Nickolson time stepping method and Galerkin discretisation in space, as outlined in [29]. However, we lump the resulting mass matrix in the momentum equations. The non-linear terms of the form $u_x(\partial u_x/\partial x)$ are linearised using $\tilde{u}_x(\partial u_x/\partial x)$ with $\tilde{u}_x$ taken from the previous time step and $u_x$ the $x$-component of velocity. The resulting set of linear equations are of the form

$$\left(B_l + \frac{1}{2}\triangle t A\right)\left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\triangle t}\right) = A\mathbf{u}^n + C\mathbf{p}^{n+1/2}, \tag{29}$$

$$C^{\mathrm{T}}\mathbf{u}^{n+1} + K\mathbf{p}^{n+1/2} = 0, \tag{30}$$

in which the matrix $B_l$ is the row sum lumped mass matrix and **u**, **p** are vectors containing the unknown velocities and pressures centred on the nodes and the superscript represents the time level at which the variables are defined. As well as applying Greens theorem to the second-order terms, it is also applied to the pressure term thus introducing pressure into the natural boundary conditions which are of the form

$$\nu\frac{\partial u}{\partial n} - n_x p = 0, \quad \nu\frac{\partial v}{\partial n} - n_y p = 0, \quad \nu\frac{\partial w}{\partial n} - n_z p = 0. \tag{31}$$

The are used as outlet boundary conditions. The stabilisation matrix $K$ takes on the form

$$K = Q^{\mathrm{T}} M_l^{-1} Q,$$

in which

$$Q_{ij} = \frac{1}{2}\int_V (\nabla N_i)^{\mathrm{T}} \overline{\overline{h}} \nabla N_j \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z,$$

where the integral is over the domain $V$, $N_i$ are the finite element shape functions associated node $i$ and the tensor $\overline{\overline{h}}$ is a discretisation of the distance across the element in various directions.

*4.1.1. Applications of mesh adaptivity*

In this section the flow past a single infinite cylinder, which is a 2-D benchmark flow problem, and 3-D flow past a bundle of cylinders (representing a sub-set of a boiler tube bank) is simulated with the aid of mesh adaptivity. The Reynolds numbers used in both simulations is 200 and the non-dimensional time step size is 0.025. In both simulations the mesh adapts every 10 time steps and the limit on the aspect ratio was set to 1000.

For transient problems, such as these, at, say every $n$ time steps, the mesh is adapted if $F_e \geqslant 3$, see Eq. (5). In practice, for the examples presented this criterion is always met and so we always adapt the mesh every 10 time steps. However, for these transient problems there is little change in the fields some of the domain from one mesh adaptation to the next and so the mesh optimisation method has a good quality initial mesh which speeds up the mesh optimisation. The element quality distributions for these examples were found to be very similar to that shown in Fig. 8(b) with the slower optimisation and so these results are not shown here.

*4.1.1.1. 2-D flow past a single cylinder*. The diameter of the cylinder is unity and the domain extent, shown in Fig. 11(a), is 30 diameters long 15 wide; the centre of the cylinder is placed 7.5 diameters from the inlet boundary on the left. At the inlet boundary, left hand side boundary of Fig. 11(a), the inlet velocity of unity in the direction normal to this boundary was specified and all other velocity components were set to zero. The top and bottom boundaries of the three-dimensional box shaped domain were 0.1 diameters away from each other and on which symmetry boundary conditions were applied in order to ensure the flow remained 2-D. At the sides of the domain, again symmetry boundary conditions were applied and a no slip boundary condition is enforced on the cylinder. Figs. 11(a)–(d) show the evolution of the mesh spanning just over half a shedding cycle. The outlet boundary on the right most side of Fig. 11(a) has boundary conditions (31) applied to it. In Fig. 12(a) the pressure at a time level of 123 time units is plotted and one can clearly see the low pressure regions in the centre of the vortices shed off the cylinder. The velocity vectors at this time level are shown in Fig. 12(b) and a close up of the mesh and velocities are shown in Figs. 12(c) and (d), respectively.

The interpolation error was set to $\hat{\epsilon} = 0.03$ (see Eq. (14)) and the value of $\psi_{\min}$ in Eq. (18) was set to 0.25. The minimum and maximum element lengths ($h_{\min}$ and $h_{\max}$) were set to 0.0025 and 4, respectively.

The speed of the flow was measured at three points 2.5, 5 and 20 diameters directly down-stream of the cylinder centre. Fig. 13(a) shows the speed of the flow at these three points and it can be deduced that the non-dimensional frequency (Strouhal no.) is 0.183 which is in agreement with [6,19]. In addition Fig. 13(b)

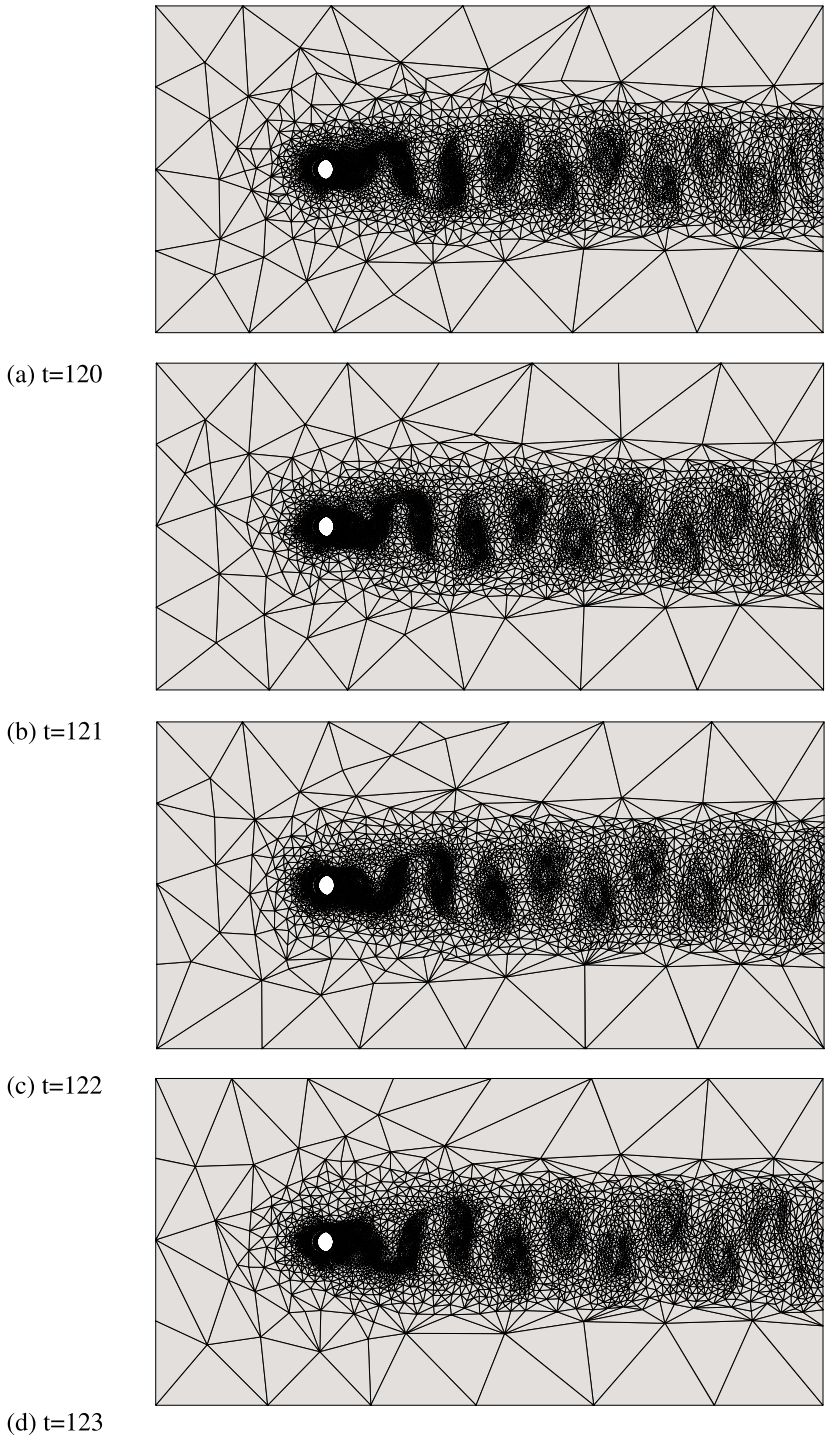
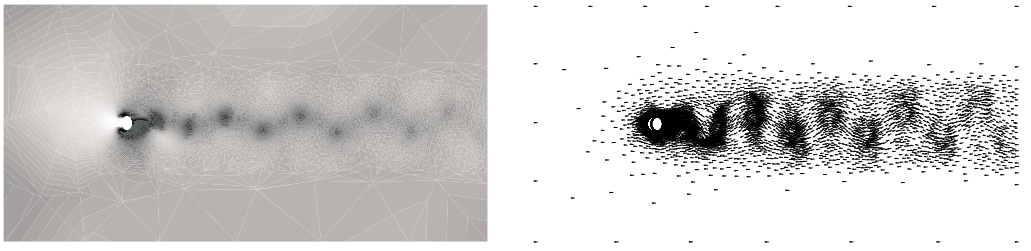
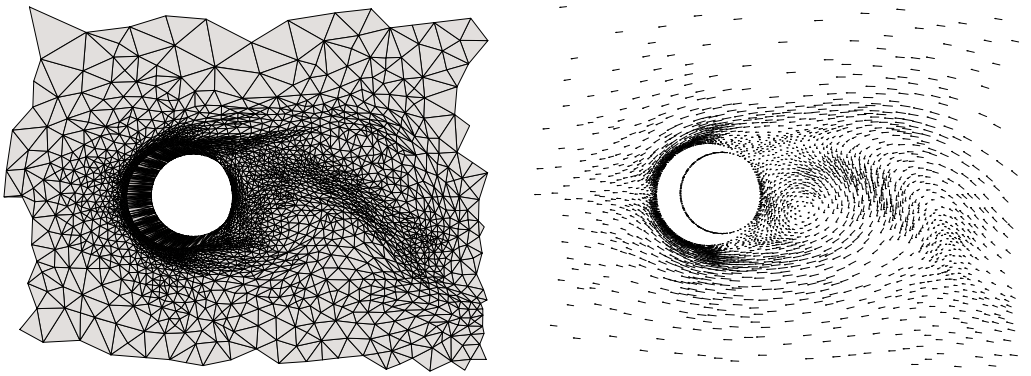



Fig. 11. Mesh (looking down from above the domain) at four time levels into the simulation – time levels span approximately half one shedding cycle.

shows how the number of elements changes over time into the simulation as the flow features evolve. Notice that the number of elements is initially relatively large, due to the high gradients caused by the impulsive initialisation of the flow. The number of elements decreases then increases as the complexity of the flow increases and the non-linearities advect down-stream.

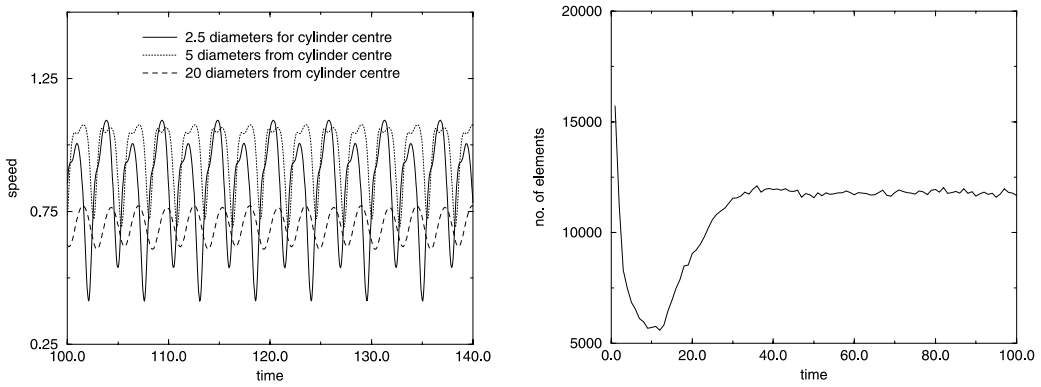


(a) pressure (viewed from above) (b) velocity



(c) mesh near cylinder (d) velocity vectors near cylinder

Fig. 12. Fields at $t = 123$ time units into simulation. (a) pressure with a linear scale from in the range $(-0.87, 0.60)$ dark is low pressure and light is high pressure (b) velocity vectors – maximum velocity 1.6 (c) mesh in the vicinity of the cylinder (d) velocity vectors in vicinity of cylinder.

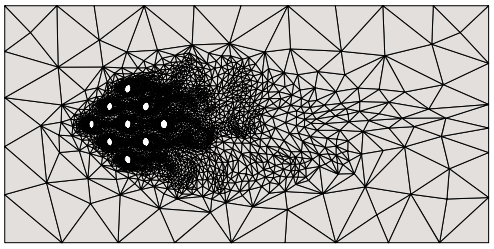



(a) the speed at three points (b) number of elements

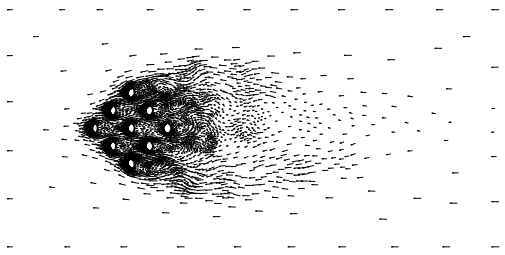
Fig. 13. Flow past a single cylinder: (a) variation with time of speed at points 2.5, 5 and 20 diameters directly downstream of the cylinder centre (Strouhal no. 0.183); (b) variation in the number of elements with time.

*4.1.1.2. Three-dimensional flow past a bundle of cylinders*. The diameter of each cylinder is unity and the domains horizontal extent is shown in Fig. 14(a). The bottom left and top right corners of the domain have horizontal coordinates of $(-11, -11)$ and $(33, 11)$, respectively. The centres of the cylinders are aligned with the $z$-coordinate and placed at $x$- and $y$-coordinates (in cylinder diameters) given by the following nine pairs:

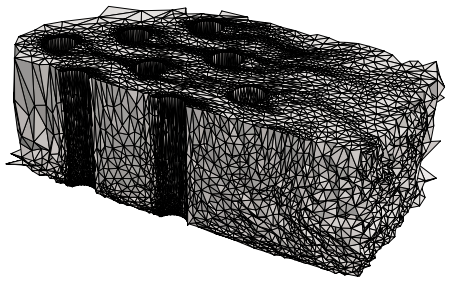
$$(0,0), (2a,0), (-2a,0), (-a,a), (a,a), (0,2a), (-a,-a), (a,-a), (0-2a),$$



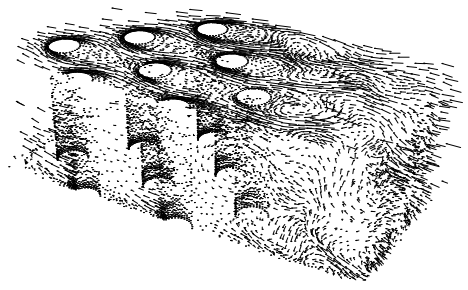
(a) mesh on surface of domain



(b) velocity on surface of domain

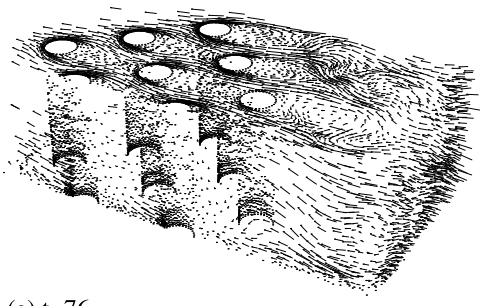


(c) mesh near cylinders

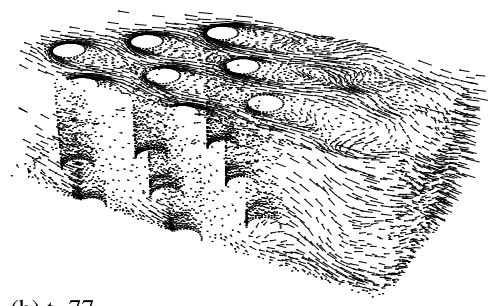


(d) velocity vectors near cylinders

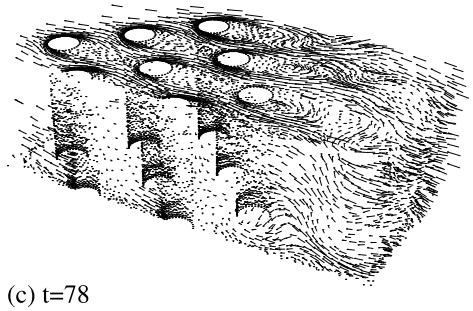
Fig. 14. Mesh and velocity field at $t = 80$ units into the simulation of flow past a bundle of cylinders. The mesh has 65,844 nodes and 342,828 elements: (a) top view of mesh on surface of domain; (b) top view of velocity vectors on surface of domain (maximum velocity 1.85); (c) mesh near cylinders; (d) velocity vectors near cylinders (maximum velocity is 1.82).



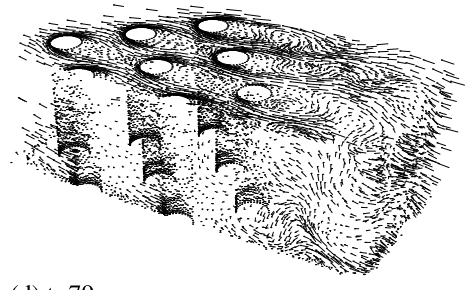
(a) t=76



(b) t=77



(c) t=78



(d) t=79

Fig. 15. Velocity vectors in the vicinity of the nine cylinders at: (a) $t = 76$ time units; (b) $t = 77$ time units; (c) $t = 78$ time units; (d) $t = 79$ time units.

in which $a = 1.679558$. The depth of the domain is 4.21 diameters and at the top of the domain $z = 4.21$ a symmetry boundary conditions is imposed and at the bottom $z = 0$ a no slip boundary conditions is enforced, which is also enforced on the cylinders. A symmetry boundary condition is also imposed on the

sides of the domain $y = -11$ and $y = 11$. At the inlet boundary, left hand side boundary of Figs. 14(a) and (b), the inlet velocity of unity in the direction normal to the boundary was specified and all other velocity components were set to zero. At the outlet boundary on the right most side of Fig. 14(a) boundary conditions given by Eq. (31) are applied.

Fig. 14 show the mesh and velocity vectors at 80 time units into the simulation. At which point it has reached a quasi steady state. Notice that the fluid tends to avoid the group of cylinders by moving around them, see Fig. 14(b), thus producing a large scale flow governed by a the a length scale dictated by the maximum distance between any two cylinders in the horizontal. Relatively small scale flow features are also generated immediately behind the cylinders. Figs. 14(c) and (d) show the mesh and velocity vectors in the vicinity of the cylinders, showing the detailed flow features picked up by mesh adaptivity including the highly anisotropic boundary layer at the bottom of these pictures. To give an idea of the dynamics of the flow the velocity vectors are shown in the vicinity of the cylinders at four time levels in Fig. 15. Each mesh adaptation takes approximately 5 minutes of CPU and the $t = 80$ was achieved in 24 h of computing time.

### 4.2. A steady-state neutron radiation transport problem

Modelling the transport of radiation through matter is important in many engineering applications, such as the analysis of nuclear reactors, the design of radiation shields, nuclear waste management, human medical dosimetry and nuclear down-well logging. This is not an easy numerical problem as geometries can be quite complex and a seven-dimensional phase space has to be addressed in full. We describe here the solution of the problem concerning the prediction of the neutron distribution in a MAGNOX nuclear reactor fuel assembly.

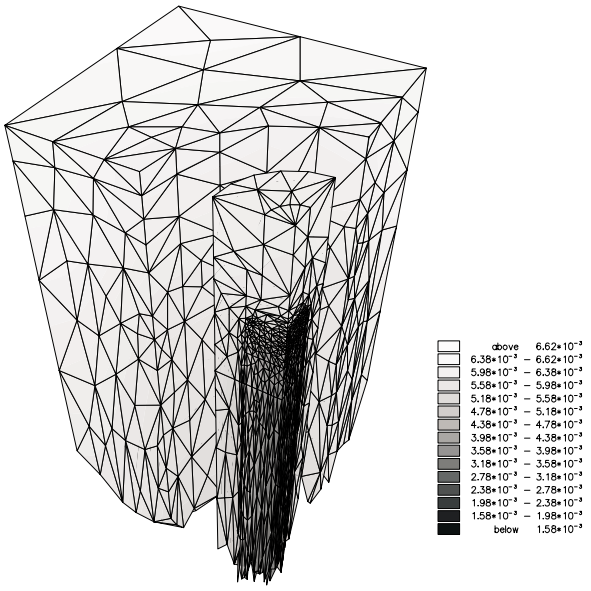
The multi-speed, linear transport equation which governs the way in which radiation propagates through matter is [25]

$$\nabla \cdot \boldsymbol{\Omega}\psi + H\psi = S,$$

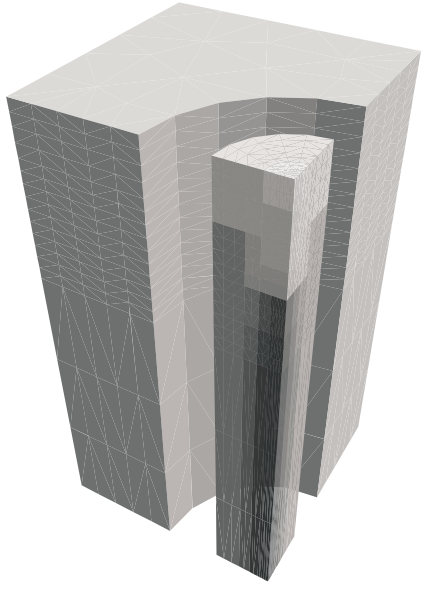
where $\psi$ is the angular flux, $\boldsymbol{\Omega}$ represents the direction of particle flow, $H$ is the scattering-removal operator and $S$ a given distributed source. This equation is solved via a combined variational finite element-spherical harmonics method which has been incorporated into the EVENT [31] code. The method gives rise to sparse, symmetric, positive-definite systems of linear equations for the moments of the angular expansion at each of the nodes of the finite element mesh. The equations are solved via nested preconditioned conjugate iterative procedures [32].

The main requirement for this type of problem is that the mesh must be adapted to capture the features of the radiation distribution in space, angle and energy. The interpolation error of the neutron scalar flux (also shown if Fig. 16(a)) of the sixth energy group (thermal energy) was used to obtain the metric through which the mesh is adapted. This was used in absence of suitable weights for the metrics associated with each of the scalar neutron fluxes in the six energy groups (fields).

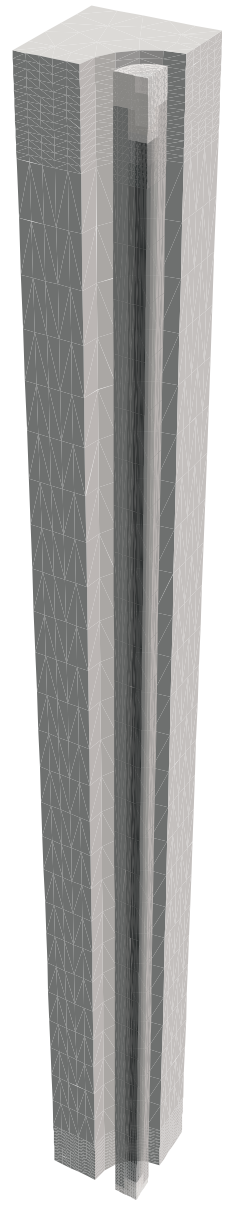
The mesh is adapted through a series of adaptive steps by adapting the mesh and refining the solution, to a final mesh converged solution. The final adapted mesh is seen for a MAGNOX nuclear reactor fuel lattice [9] at the topmost part of the domain in Figs. 16(a) and (b) shows the regions of different material properties which the mesh in Fig. 16(a) must conform to. A MAGNOX lattice cell consists of a uranium metal fuel rod, with associated metallic etc. cladding regions separated by a coolant gas 'void' from a graphite moderator region. The whole domain is shown in Fig. 16(c) with regions shaded according to their material properties. At the bottom of the lattice of Fig. 16(c) the mesh is a near mirror image of that shown in Fig. 16(a) and along the elongated main body of the lattice, the mesh is highly anisotropic (lengthened) in the vertical direction. The final adapted mesh has 28692 elements and 6047 nodes. This simulation took approximately 30 min of CPU time on an Compaq XP1000 ALPHA (EV6) workstation for a calculation consisting of 6 neutron energy groups and 5 angular variables per energy group.




(a) mesh at top of domain for group 6 scalar flux



(b) different materials



(c) materials across whole domain

Fig. 16. Diagrams showing the adapted mesh (with 28692 elements and 6074 nodes) and regions of different material of the MAGNOX lattice: (a) the adapted mesh at the upper part of domain with the scaler flux of the sixth energy group superimposed; (b) regions of different material properties at the top most part of the domain; (c) whole domain showing materials.

## 5. Conclusions

The mesh optimisation method described here provides high quality anisotropic meshes for finite element solvers and the indications are that the implementation can have a time complexity that varies linearly with the number of elements. The methods have been designed to be insensitive to round-off error and the mesh adaptivity algorithm as a whole has proven to be robust (the mesh optimisation method does not fail to produce valid elements). This is important for long transient calculations, for example. To make the

algorithm efficient, the number of local mesh transformations considered has been minimised subject to the constraint that high quality meshes must be obtained.

It is shown that mesh adaptivity offers a natural partnership for the pseudo compressibility method used here to solve the incompressible Navier–Stokes equations on a shared velocity/pressure mesh, because it offers a solution for the problem associated with this method of poor continuity satisfaction in regions with large variations in pressure. Anisotropic adaptivity in particular offers substantial computational improvements over fixed mesh methods. There were no significant difficulties in solving, with iterative solution methods, either the radiation or fluids problems on the highly distorted anisotropic meshes produced by adaptivity. For calculations with many field variables of different types, more work is required to obtain suitable weightings of the metrics associated with each field.

## Acknowledgements

Support by the UK Engineering and Physical Sciences Research Council and British Energy Plc. through grant GR/K/73466, is gratefully acknowledged.

## References

[1] S. Abdallah, Numerical solutions for the pressure Poisson equation with Neumann boundary conditions using a non-stagged grid, I, J. Comput. Phys. 70 (1987) 182–192.
[2] S. Abdallah, Numerical solutions for the incompressible Navier–Stokes equations in primative variables non-stagged grid II, J. Comput. Phys. 70 (1987) 193–202.
[3] M. Ainsworth, J.T. Oden, A posteriori error estimation in finite element analysis, Comput. Methods Appl. Mech. Engrg. 142 (1997) 1–88.
[4] S.W. Armfield, Finite difference solutions of the Navier–Stokes equations on staggered and non-staggered grids, Comput. Fluids 20 (1991) 1–17.
[5] I. Babuska, Error bounds for finite element method, Numer. Math. 16 (1971) 322–333.
[6] E. Berger, R. Willie, Periodic flow phenomena, Ann. Rev. Fluid. Mech. 4 (1972) 313–329.
[7] C.M. Bishop, Neural network for pattern recognition, Oxford University Press, New York, 1995.
[8] F. Brezzi, On the existance, uniqueness and approximation of saddle-point problems arising from Lagrange multipliers, RAIRO Ser. Rouge Anal. Numer R 2 (1979) 129–151.
[9] BNFL Magnox, private communication, May, 1999.
[10] H. Borouchaki, P.J. Frey, Adaptive triangular-quadrilateral mesh generation, Int. J. Num. Methods Engrg. 41 (1998) 915–934.
[11] H. Borouchaki, P.L. George, Optimal Delaunay point insertion, Int. J. Num. Methods Engrg. 39 (1996) 3407–3437.
[12] H. Borouchaki, S.H. Lo, Fast Delaunay triangulation in three dimensions, Comput. Methods Appl. Mech. Engrg. 128 (1995) 153–167.
[13] G.C. Buscaglia, E.A. Dari, Anisotropic mesh optimization and its application in adaptivity, Int. J. Num. Methods Engrg. 40 (1997) 4119–4136.
[14] M.J. Castro-Diaz, F. Hecht, B. Mohammadi, O. Pironneau, Anisotropic unstructured mesh adaption for flow simulations, Int. J. Num. Methods Fluids 25 (1997) 475–491.
[15] L. de Floriani, Data structures for encoding triangulated irregular networks, Adv. Engrg. Software 9 (1987) 122–128.
[16] L.A. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, Int. J. Num. Methods Engrg. 40 (1997) 3979–40002.
[17] P.L. George, F. Hecht, E. Saltel, Automatic mesh generator with a specified boundary, Comput. Methods Appl. Mech. Engrg. 92 (1991) 269–288.
[18] P.L. George, Delaunay Triangulation and Meshing: Application to Finite Elements, Editions HERMES, Paris, 1998.
[19] J. Gerrard, The wakes of cylindrical bluff bodies at low Reynolds numbers, Philos. Trans. Roy. Soc. 288 (1978) 351–367.
[20] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, Maryland, 1983.
[21] P.M. Gresho, S.T. Chan, M.A. Christon, A.C. Hinmarsh, A little more on stabilized $Q_1Q_1$ for transient viscous incompressible flow, Int. J. Num. Methods Fluids 21 (1995) 837–856.
[22] T.J.R. Hughes, L.P. Franca, A new finite element formulation for computational fluid dynamics: VII. The Stokes problem with various well-posed boundary conditions: symmetric formulations that converge for all velocity/pressure spaces, Comput. Methods Applied Mech. Engrg. 65 (1987) 85–96.
[23] B. Joe, Three-dimensional triangulations from local transformations, SIAM J. Sci. Stat. Comput. 10 (1889) 718–741.
[24] Y. Kallinderis, P. Vijayan, Adaptive refinement-coarsening scheme for three-dimensional unstructured meshes, AIAA J. 31 (1993) 1440–1447.
[25] E.E. Lewis W.F. Miller Jr., Computational Methods of Neutron Transport, Wiley, New York, 1984.

[26] R. Lohner, K. Morgan, K.O. Zienkiewicz, An adaptive finite element procedure for compressible high speed flows, Comput. Methods Appl. Mech. Engrg. 51 (1985) 441–465.
[27] R. Lohner, Some useful data structures for the generation of unstructed grids, Comm. App. Numer. Methods 4 (1988) 123–135.
[28] R. Lohner, P. Parikh, Generation of three-dimensional unstructured grids by the advancing-front method, Int. J. Num. Methods Fluid 8 (1988) 1135–1149.
[29] S. Mansoorzadeh, C.C. Pain, C.R.E. de Oliveira, A.J.H. Goddard, Finite Element simulations of incompressible flow past a heated/cooled sphere, Int. J. Num. Methods Fluids 28 (1999) 903–915.
[30] P. Moller, P. Hansbo, On advancing front mesh generation in three dimensions, Int. J. Num. Methods Engrg. 38 (1995) 3551–3569.
[31] C.R.E. de Oliveira, An arbitrary geometry finite element method for multigroup neutron transport with anisotropic scattering, Prog. Nuclear Energy 18 (1986).
[32] C.R.E. de Oliveira, C.C. Pain, A.J.H. Goddard, Parallel domain decomposition methods for large-scale finite element transport modelling, in: Proceedings of Int. Conf. on Math. Comput. Reactor Physics, Envir. Anal. Portland, USA, 1995.
[33] J. Peraire, M. Vahdati, K. Morgan, O.C. Zienkiewicz, Adaptive remeshing for compressible flow computations, J. Comput. Phys. 72 (1987) 449–466.
[34] C.M. Rhie, W.L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge seperation, AIAA J 21(11) (1983).
[35] F. Sotiropoulos, S. Abdallah, The discrete continuity equation in primitive variable solutions of incompressible flow, J. Comput. Phys. 95 (1991) 212–227.
[36] T. Strouboulis, J.T. Oden, A posteriori error estimation of finite element approximations in fluid mechanics, Comput. Methods Appl. Mech. Engrg. 78 (1990) 201–242.
[37] N.P. Weatherill, Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints, Int. J. Num. Methods Engrg. 37 (1994) 2005–2039.
[38] J. Wu, J.Z. Zhu, J. Szmelter, O.C. Zienkiewicz, Error estimation and adaptivity in Navier–Stokes incompressible flows, Comput. Mech. 6 (1990) 259–270.
[39] X. Xu, C.C. Pain, A.J.H. Goddard, C.R.E. de Oliveria, An automatic adaptivity meshing techniques for Delaunay triangulations, Comput. Methods Appl. Mech. Engrg. 161 (1998) 297–303.
[40] O.C. Zienkiewicz, R.L. Taylor, The finite element method, vol. 2, fourth ed., McGraw-Hill, New York, 1991.