INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 27, 41–55 (1998)

UNSTRUCTURED MESH PROCEDURES FOR THE SIMULATION OF THREE-DIMENSIONAL TRANSIENT COMPRESSIBLE INVISCID FLOWS WITH MOVING BOUNDARY COMPONENTS

O. HASSAN[1]*, E. J. PROBERT[2] AND K. MORGAN[1]

[1] *Department of Civil Engineering, University of Wales Swansea, Swansea SA2 8PP, U.K.*
[2] *EMBS, University of Wales Swansea, Swansea, SA2 8PP U.K.*

SUMMARY

The solution of high-speed transient inviscid compressible flow problems in three dimensions is considered. Discretization of the spatial domain is accomplished by the use of tetrahedral elements generated by Delaunay triangulation with automatic point creation. Methods of adapting the mesh to allow for boundary movement are considered and a strategy for ensuring boundary recovery is proposed. An explicit multistage time-stepping algorithm is employed to advance the flow solution. A number of examples are included to illustrate the numerical performance of the proposed procedures. © 1998 John Wiley & Sons, Ltd.

Int. J. Numer. Meth. Fluids, **27**: 41–55 (1998)

KEY WORDS: unstructured mesh; compressible flow; transient flow; mesh adaptation; moving boundaries; Delaunay triangulation

1. INTRODUCTION

Unstructured mesh methods are now widely employed for the simulation of steady inviscid aerodynamic flows.[1] Domains of complex geometrical shape can be readily represented by unstructured assemblies of elements, and fully automatic unstructured mesh generation procedures exist for the discretization of the domains which are typically encountered in flows of industrial interest. Problems associated with the accuracy of flow algorithms on such general grids are being addressed[2] and recent work has demonstrated the successful extension of the approach to provide an unstructured mesh capability for steady viscous flow analysis.[3] The unstructured mesh approach can naturally accommodate adaptivity, and numerous techniques such as mesh movement, mesh enrichment and adaptive remeshing have been employed for steady inviscid flow analysis.[4] Initial experiences with these adaptivity techniques applied to viscous flow simulation have proved to be encouraging.[5]

Transient inviscid aerodynamic flows with fixed boundary components have also been simulated successfully using the unstructured mesh approach. The flow features detected in such problems may be accurately represented by the employment of mesh adaptation.[6,7] When consideration is given to the simulation of aerodynamic flows involving moving boundary components,[8,9] where the shape of the geometry is changing continuously, the use of some form of mesh adaptation is essential. The

* Correspondence to: O. Hassan, Department of Civil Engineering, University of Wales Swansea, Swansea SA2 8PP, U.K.

CCC 0271–2091/98/010041–15 $17.50 *Received May 1996*

simulation of three-dimensional flow can be computationally expensive and it is necessary to minimize the amount of mesh adaptation which is performed. One possible approach[10] to such problems is to hold the mesh fixed and simulate the motion of a body by modifying the boundary conditions at the surface of the body. This procedure can be expected to be effective if the prescribed movement of the body is sufficiently small. However, in the case of large-scale displacements alternative procedures must be devised.

Here we propose a method using an unstructured grid approach which has been designed to provide an automatic adaptive mesh capability for solving transient compressible flows involving moving boundaries. The mesh generation procedure which is employed is the Delaunay triangulation method with automatic point creation.[11,12] With this approach a distribution of sources can be placed in regions where additional mesh resolution is required[13,14] for the accurate representation of flow features. When a mesh has been generated to cover the domain of interest, the compressible Euler equations can then be solved to simulate the flow over the body of interest. The flow solver is based upon a Galerkin weighted residual approximation,[15] with an edge-based data structure which is the most efficient of the possible alternatives. The solver includes a deforming mesh algorithm to move the mesh in transient computations where the geometry is changing. Local mesh regeneration, employing a method of boundary recovery which ensures mesh conformity, is adopted. A number of numerical examples are included to demonstrate the performance of these methods when they are employed in the analysis of a transient flow with moving boundary components.

2. GOVERNING EQUATIONS

The equations governing 3D compressible unsteady inviscid flow are considered in the conservation form

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^j}{\partial x_j} = 0, \quad j = 1, 2, 3, \tag{1}$$

where the summation convention is employed and

$$\mathbf{U} = \begin{bmatrix} \varrho \\ \varrho u_1 \\ \varrho u_2 \\ \varrho u_3 \\ \varrho \varepsilon \end{bmatrix}, \qquad \mathbf{F}^j(\mathbf{U}) = \begin{bmatrix} \varrho u_j \\ \varrho u_1 u_j + p\delta_{1j} \\ \varrho u_2 u_j + p\delta_{2j} \\ \varrho u_3 u_j + p\delta_{3j} \\ (\varrho \varepsilon + p) u_j \end{bmatrix}. \tag{2}$$

Here δ_{ij} denotes the Kronecker delta, ϱ, p and ε denote the density, pressure and total specific energy of the fluid respectively and u_j is the component of the fluid velocity in the direction x_j of a Cartesian co-ordinate system. The set of equations is completed by the addition of the perfect gas equation of state

$$p = (\gamma - 1)\varrho(\varepsilon - 0{\cdot}5 u_k u_k), \tag{3}$$

where γ is the ratio of specific heats.

Suppose that $\mathbf{U}$ is known at every point of a prescribed region Ω_m at a certain time $t = t_m$, i.e. $\mathbf{U} = \mathbf{U}^m$ at time $t = t_m$. The problem of interest here is then to determine the function $\mathbf{U}$ which satisfies the initial condition, is a solution of the governing equations (1)–(3) in a space–time domain $D = (\Omega(t),\ t_m \leqslant t)$ and satisfies appropriate boundary conditions $\mathbf{F}_n = n_j \mathbf{F}^j = \bar{\mathbf{F}}_n$ on the boundary surface $\Gamma(t)$ of Ω. Here n_j denotes the component, in direction x_j, of the unit outward normal to $\Gamma(t)$.

3. SOLUTION PROCEDURE

3.1. Variational formulation

To develop a numerical solution algorithm, it is convenient to replace the classical formulation of the problem by an equivalent weak variational formulation. To achieve this, we introduce a trial function set $\mathscr{T}$ and a weighting function set $\mathscr{W}$. These sets consist of all suitably smooth functions and, in addition, all members of the trial function set satisfy the initial condition on Ω at time $t = t_m$, i.e.

$$\mathscr{T} = \{\mathbf{U} | \mathbf{U} = \mathbf{U}^m \quad \text{on } \Omega \text{ at } t = t_m\}. \tag{4}$$

A weak variational formulation for the problem is then: find $\mathbf{U}$ in $\mathscr{T}$ such that

$$\int_{t_m}^{T} \int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} W \, \mathrm{d}\Omega \, \mathrm{d}t = \int_{t_m}^{T} \int_{\Omega(t)} \frac{\partial W}{\partial x_j} \mathbf{F}^j \, \mathrm{d}\Omega \, \mathrm{d}t - \int_{t_m}^{T} \int_{\Gamma(t)} W \bar{\mathbf{F}}_n \, \mathrm{d}\Gamma \, \mathrm{d}t \tag{5}$$

for every member W of the weighting function set $\mathscr{W}$ and for all $T > t_m$.

3.2. Approximate variational formulation

Using a fully unstructured tetrahedral mesh generator,[12] the spatial solution domain $\Omega(t)$ is discretized into a mesh of linear tetrahedral elements. Nodal points, numbered $1, 2, \ldots, p$, are located at the element vertices. Subsets $\mathscr{T}^{(p)}$ and $\mathscr{W}^{(p)}$ of the trial and weighting function set respectively are defined by

$$\mathscr{T}^{(p)} = \{\mathbf{U}^{(p)}(x_j, t) | \mathbf{U}^{(p)} = \mathbf{U}_J(t) N_J(x_j, t); \ \mathbf{U}_J(t_m) = \mathbf{U}_J^m\}, \tag{6}$$

$$\mathscr{W}^{(p)} = \{W(x_j, t) | W = a_J N_J(x_j, t)\}, \tag{7}$$

where J takes the values $1, 2, \ldots, p$ in the implied summation. Here N_J is the standard piecewise linear finite element shape function (suitably modified to allow for the mesh movement) which takes the value unity at node J (located at $\mathbf{x} = \mathbf{x}_J(t)$) and zero at every other node, $\mathbf{U}_J$ is the value of $\mathbf{U}^{(p)}$ at node J and a_J is a constant. The superscript m denotes an evaluation at time $t = t_m$. A discrete approximate (Galerkin) variational formulation of the original problem may now be expressed as: find $\mathbf{U}^{(p)}$ in $\mathscr{T}^{(p)}$ such that

$$\int_{t_m}^{T} \int_{\Omega(t)} \frac{\partial \mathbf{U}^{(p)}}{\partial t} N_I \, \mathrm{d}\Omega \, \mathrm{d}t = \int_{t_m}^{T} \int_{\Omega(t)} \frac{\partial N_I}{\partial x_j} \mathbf{F}^j(\mathbf{U}^{(p)}) \, \mathrm{d}\Omega \, \mathrm{d}t - \int_{t_m}^{T} \int_{\Gamma(t)} N_I \bar{\mathbf{F}}_n \, \mathrm{d}\Gamma \, \mathrm{d}t \tag{8}$$

for $I = 1, 2, \ldots, p$. This variational statement requires further manipulation to convert it into a form which is useful for the practical solution of problems involving moving boundaries.

Suppose that the mesh is moving with a velocity

$$\mathbf{v}^{(p)} = (v_1^{(p)}, v_2^{(p)}, v_3^{(p)}), \tag{9}$$

where each velocity component has the piecewise linear representation

$$v_j^{(p)} = v_{jJ} N_J. \tag{10}$$

In this expression, v_{jJ} denotes the component of the velocity of node J in the direction x_J. Then

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega(t)} \mathbf{U}^{(p)} N_I \, \mathrm{d}\Omega = \int_{\Omega(t)} \frac{\partial}{\partial t} (\mathbf{U}^{(p)} N_I) \, \mathrm{d}\Omega + \int_{\Omega(t)} \frac{\partial}{\partial x_j} (v_j^{(p)} \mathbf{U}^{(p)} N_I) \, \mathrm{d}\Omega, \tag{11}$$

where the total time derivative denotes differentiation following the moving mesh. When this result is employed in equation (8), it follows that the variational statement can be expressed in the alternative form:[9] find $\mathbf{U}^{(p)}$ in $\mathscr{T}^{(p)}$ such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \mathbf{U}^{(p)} N_I \, \mathrm{d}\Omega = \int_{t_m}^{T}\int_{\Omega(t)} \frac{\partial N_I}{\partial x_j}\mathbf{F}^{j*}(\mathbf{U}^{(p)}) \, \mathrm{d}\Omega \, \mathrm{d}t - \int_{t_m}^{T}\int_{\Gamma(t)} N_I \bar{\mathbf{F}}_n^* \, \mathrm{d}\Gamma \, \mathrm{d}t \tag{12}$$

for $I = 1, 2, \ldots, p$, where

$$\mathbf{F}^{j*}(\mathbf{U}^{(p)}) = \mathbf{F}^{j}(\mathbf{U}^{(p)}) - v_j^{(p)}\mathbf{U}^{(p)}, \qquad \bar{\mathbf{F}}_n^* = \bar{\mathbf{F}}_n - n_j v_j^{(p)}\mathbf{U}^{(p)}. \tag{13}$$

If the flux terms are also assumed to vary between their nodal values in a piecewise linear fashion, substitution of the assumed form for $\mathbf{U}^{(p)}$ from equation (6) into equation (12) results in the requirement that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \mathbf{U}^{(p)} N_I \, \mathrm{d}\Omega = \int_{t_m}^{T} \mathbf{R}(t) \, \mathrm{d}t, \tag{14a}$$

where

$$\mathbf{R}(t) = \int_{\Omega(t)} \frac{\partial N_I}{\partial x_j} N_K \mathbf{F}_K^{j*} \, \mathrm{d}\Omega - \int_{\Gamma(t)} N_I N_K \bar{\mathbf{F}}_{nK}^* \, \mathrm{d}\Gamma. \tag{14b}$$

The quantities

$$M_{IJ} = \int_{\Omega(t)} N_I N_J \, \mathrm{d}\Omega \tag{15}$$

denote the the entries in the consistent finite element mass matrix. In the current implementation this matrix is replaced by the diagonal lumped mass matrix with entries

$$M_I^{\mathrm{L}} = \int_{\Omega(t)} N_I \, \mathrm{d}\Omega \tag{16}$$

and the solution is advanced over a time step Δt from time t_m to time t_{m+1} via the explicit r-stage time-stepping procedure

$$\begin{aligned}
\mathbf{V}_I^{(0)} &= \mathbf{U}_I^m, \\
M_I^{\mathrm{L}(1)}\mathbf{V}_I^{(1)} &= M_I^{\mathrm{L}(0)}\mathbf{V}_I^{(0)} + \alpha_1 \Delta t \mathbf{R}(t_{m+(0)}), \\
&\vdots \\
M_I^{\mathrm{L}(r)}\mathbf{V}_I^{(r)} &= M_I^{\mathrm{L}(0)}\mathbf{V}_I^{(0)} + \alpha_r \Delta t \mathbf{R}(t_{m+(r-1)}), \\
\mathbf{U}_I^{m+1} &= \mathbf{V}_I^{(r)}.
\end{aligned} \tag{17}$$

In this expression the parameters $\alpha_1, \alpha_2, \ldots, \alpha_r$ are assigned values which are appropriate to the number of states, r, being employed. An edge-based representation is employed for the tetrahedral mesh[2] and the vector $\mathbf{R}$ is evaluated by a loop over the mesh edges. In this fashion it is relatively straightforward to correct the central difference character of the time-stepping scheme by replacing the actual flux function by a consistent numerical flux function. In this work this is achieved by using a Roe flux function over each edge of the mesh to achieve a first-order-accurate scheme. Higher-order accuracy is then attained by using simple TVD ideas.[2]

4. MESH ADAPTIVITY

The generation of an initial mesh of tetrahedral elements is accomplished by the Delaunay triangulation method with automatic point creation. In the automatic point creation procedure the distribution of points is obtained by interpolation of the point spacing from the boundary nodes. The method is enhanced by the use of sources, which provide a mechanism for clustering points. A source has an amplitude, or strength, which specifies the spacing at the source and a decay which determines the rate at which this spacing varies away from the source. A distribution of sources placed in regions where additional mesh resolution is required ensures a mesh which adequately resolves the initial solution. The solution is advanced using the algorithm described in the previous section and at each time step the co-ordinates of the points on the moving boundaries are updated according to the prescribed movement of the geometry. Without a corresponding mesh adaptivity strategy the mesh would become successively distorted and eventually invalid. However, simulations of three-dimensional flow can be computationally expensive and it is advisable to minimize the amount of mesh adaptation which is performed.

4.1. Deforming mesh algorithm

Two methods have been investigated to achieve the mesh adaptivity which is required as the computation progresses. The first method considered is a deforming mesh algorithm which models the mesh as a spring network, with nodal points on the outer boundary held fixed while the instantaneous locations of points on the moving boundary are specified. The locations of the inner nodes of the mesh are then determined by solving the static equilibrium equations that result from summing the forces in the spring system at each node. The mesh-smoothing algorithm thus alters the positions of the interior nodes without changing the topology of the mesh.

In contrast with a standard smoothing procedure, the deforming mesh algorithm does not relocate a point at the centre of gravity of its neighbours. If the co-ordinates of node I at time t_m are given by $\mathbf{x}_I^m$, then the new co-ordinates $\mathbf{x}_I^{m+1}$ at time $t_{m+1} = t_m + \Delta t$ are obtained by the addition of a displacement $\mathbf{d}_I^{m+1}$:

$$\mathbf{x}_I^{m+1} = \mathbf{x}_I^m + \mathbf{d}_I^{m+1}, \tag{18}$$

where $\mathbf{d}_I^{m+1}$ is prescribed on the moving boundaries and is set equal to zero on any fixed boundaries. Hence the co-ordinates on fixed boundaries remain unchanged while the co-ordinates on moving boundaries change according to the specification of the problem under consideration. The new displacement at an internal node I is found by averaging the displacements of surrounding points and iterating to convergence. This iteration takes as its starting value the displacement at I at the previous time step, as follows:

$$\begin{aligned}
&\mathbf{d}_I^{(0)} = \mathbf{d}_I^m,\\
&\quad\vdots\\
&\mathbf{d}_I^{(k)} = \frac{\sum_{p=1}^{n_I} \mathbf{d}_p^{(k-1)}}{n_I}, \quad k = 1, 2, \ldots, q,\\
&\quad\vdots\\
&\mathbf{d}_I^{m+1} = \mathbf{d}_I^{(q)}.
\end{aligned} \tag{19}$$

In this formulation the summation extends over all the nodes surrounding node I and sufficient convergence is normally achieved by taking $q = 5$. By beginning the iterative procedure with the

previous displacement at each node, the motion of the moving boundaries is allowed to spread throughout the whole of the mesh.

4.2. *Local remeshing with boundary recovery*

The deforming mesh algorithm proves very effective for problems involving small amplitudes of oscillation. However, if the problem under consideration involves the large-scale motion of a body, the deforming mesh approach alone cannot be expected to suffice, as some elements will eventually become either so stretched or so flattened that their presence will adversely affect the allowable time step for the next iteration of the flow solver. For such problems a local remeshing procedure, in which distorted elements are removed and those portions of the mesh regenerated, has been implemented.

It is highly possible that distorted elements will occur in several regions and that their removal will result in the creation of a number of separate holes in the mesh. Each hole will be bordered by a collection of faces, some of which might possibly contain boundary points. The regeneration procedure begins with a triangulation on the boundaries, using a Delaunay procedure with automatic point creation[12] applied in two dimensions. Since the positions of the sources have been updated in a similar way to the co-ordinates of the moving body, this new triangulation will show an adaptation of the mesh where additional resolution is required and the resulting triangular faces can be added to the list of faces surrounding the holes to form closed regions. Nodes connected by this list of faces form the starting point for the Delaunay mesh generation which is carried out in three dimensions. Following the automatic point creation and connection by the Delaunay algorithm, the resulting triangulation must be made boundary-conforming. Methods which have been reported previously[12] do not ensure that the original surface triangulation is fully recovered, with the possibility of additional points being introduced on the surface. However, that possibility is unacceptable in the present case, as the 'surface' triangulation bordering a hole is connected to the remainder of the mesh and must be made fully conforming or complications of connection will result. An algorithm has been devised to ensure that the boundary faces surrounding the hole are completely recovered. The algorithm begins by identifying a face in the boundary discretization surrounding the hole which does not appear in the newly generated mesh. A search is made over the edges of this face to determine connected faces which also need to be recovered and a polygon is built enclosing the tetrahedra which contain those faces. Faces in the original boundary discretization which lie within the polygon are then added and, starting with that edge of the polygon with the smallest dihedral angle, a tetrahedron is formed by closing the two adjacent faces to that edge. If this is continued, the polygon can be filled with tetrahedra without creating any new points in the process. If another face which needs to be recovered is identified, the procedure is repeated until all the boundary faces surrounding the hole have been recovered. The algorithm can be implemented as follows.

MFACE = total number of faces

LFACE(1 : NFACE) = list of connected faces to be recovered (where connected faces implies faces sharing an edge)

LTETR(1 : NTETR) = list of tetrahedra intersecting faces to be recovered

LPOLY(1 : NPOLY) = list of faces enclosing the tetrahedra in LTETR

LEDGE(1 : NEDGE) = list of edges in LPOLY

1. *LOOP* over boundary faces, jf = 1 : MFACE
   If any tetrahedron contains face jf *GO TO* 1
   Set NFACE = 0
   Insert jf into LFACE, the list of faces to be recovered

1.1 Consider each face in the list LFACE
LOOP over three edges of face
IF edge does not exist increment NFACE and add face adjacent to edge to LFACE
ENDLOOP
IF new face has been added *GOTO* 1.1

LOOP over faces in LFACE
Set NTETR = 0
LOOP over tetrahedra, je, adjacent to the face
IF je intersects with the face increment NETTR and add je to LTETR
ENDLOOP
ENDLOOP

2. Set NPOLY = 0
LOOP over faces of elements in LTETR
IF face exists in list LPOLY *THEN*
Remove face from list
NPOLY = NPOLY − 1
ELSE
Add face to list
NPOLY = NPOLY + 1
ENDIF
ENDLOOP

3. Add faces to be recovered, NFACE, in the list LFACE, to the list LPOLY and form LEDGE.
4. Select the edge in LEDGE with smallest dihedral angle
Form tetrahedron by closing the two adjacent faces to that edge
Update the list LEDGE
Repeat until polygon is filled with tetrahedra
5. *ENDLOOP*

After the triangulation of each separate hole the nodal values of the solution at newly created points can be obtained by interpolation from a background grid of the deleted elements of the previous mesh using a search process based on an alternating digital tree.[16]

5. ALGORITHM FOR TRANSIENT ANALYSIS

With an efficient flow solution procedure and an automatic mesh adaptivity capability the simulation of pseudotransient and transient flows with moving boundaries can be considered. Here it is assumed that a pseudotransient simulation will consist of a sequence of steady state solutions, corresponding to a sequence of locations in the time history of a moving body. These solutions will be computed on an appropriate sequence of meshes, each new mesh being derived from its immediate precursor by automatic mesh adaptivity. In the case of a truly transient simulation the moving boundaries will be updated at each time step and the mesh adapted accordingly. An algorithm to advance the solution in time can be written as follows.

1. Generate initial mesh using Delaunay triangulation with sources placed appropriately
2. Calculate element volumes, VOL0
3. *LOOP* K = 1,NLOOP

Compute weights and cell volume
LOOP I = 1,NTIME
Until stopping condition is reached:
Advance the solution one time step
IF true transient *THEN*
Update co-ordinates of moving nodes
Update co-ordinates of sources
Apply deforming mesh algorithm
Calculate element volumes, VOLN
Compare VOLN with VOL0
IF VOLN < 0 for any element *OR* $|(\text{VOLN} - \text{VOL0})/\text{VOL0}| > \gamma$ for at least β per cent of elements *THEN*
Mark elements for removal
GOTO 4
ENDIF
ENDIF
ENDLOOP
ENDLOOP

4. *IF* true transient *THEN*
Remove marked elements together with a surrounding layer, producing holes in the mesh bordered by collections of triangular faces
Regenerate holes using the Delaunay method
Interpolate nodal values of the solution on the new portions of the mesh
ELSE
Compute and mark co-ordinates of moving nodes in their new position
Remove layers of elements surrounding moving body until a hole has been created containing all the marked co-ordinates
Update positions of moving nodes and sources
Regenerate hole using the Delaunay method
Interpolate nodal values of the solution of the new portions of the mesh
ENDIF

5. *GOTO* 2

In the application of this algorithm the values of NLOOP and NTIME depend on the problem being considered. For a pseudotransient simulation the value of NLOOP is set equal to the number of stages in the specified time history and NTIME is set equal to the number of time steps required for each steady state solution. In this case the weights and cell volumes are only calculated at the beginning of a computation on any particular mesh, so that the benefits of the edge-based data structure are gained. In the case of a transient simulation the value of NTIME should, strictly speaking, be equal to one. The deforming mesh algorithm is applied at each time step and hence, for full accuracy, the weights and cell volumes should also be recalculated at each time step. Because of the computational expense which this strategy implies, the effect of increasing the value of NTIME has been investigated and a value of NTIME = 5 has been adopted in the calculations reported here. This value has been found to reduce the expense of the calculation significantly without a measurable degradation of the solution quality. The value of NLOOP is set equal to the number of time steps required for the full computation to be completed.

Generally the values of $\gamma = 0{\cdot}2$ and $\beta = 0{\cdot}5$ are preset, so that regeneration will take place if 0·5 per cent of the elements have changed in volume by at least 20 per cent, and these values have been found to work well with a range of applications.

6. NUMERICAL EXAMPLES

The first test case configuration is a missile mounted initially underneath a wing and subsequently released. This example is used to demonstrate the performance of the method when applied to a pseudotransient problem. The time history of the store trajectory is determined by a sequence of data which specify the translational motion of the nose of the missile relative to its original position at time $t = 0$ and the rotational motion in terms of an axis system which is fixed on the missile with its origin at the nose. The freestream flow conditions correspond to a Mach number $M_\infty = 0{\cdot}8$ and the body is initially at an angle of attack $\alpha_0 = 5{\cdot}0^\circ$. A steady state solution is computed by performing 2000 time steps of the flow algorithm on the initial mesh, which consists of 369,862 elements and 63,889 nodes. A detail of this mesh showing the position of the missile relative to the wing, together with the corresponding computed Mach number contours, is shown in Figure 1. This figure shows the mesh and Mach number contours on the surfaces of the wing and the missile. Also shown are a detail of a cut through the mesh on a plane passing through a section of the wing and the missile and the corresponding computed Mach number contours in that plane. A sequence of adapted steady state solutions is computed at subsequent positions in the missile trajectory by applying the algorithm described in the previous section and implementing the appropriate pseudotransient option. Each of these solutions is obtained by performing 2000 time steps of the flow algorithm on a mesh adapted to the position of the missile at the appropriate point in its time history, with the results displayed in Figure 1.

The second example considered is the truly transient development of the flow over a wing with freestream conditions corresponding to a Mach number $M_\infty = 0{\cdot}713$ and an angle of attack $\alpha_0 = 0{\cdot}52$. The wind is oscillating sinusoidally with a motion which is described in terms of the pitch and heave at specified sections along the wing, with those parameters being interpolated linearly at intermediate points. Each section i pitches and heaves sinusoidally as

$$\alpha(t) = \alpha_i \sin(\omega t), \qquad z(t) = z_i \sin(\omega t), \tag{20}$$

where the amplitude of pitch, α_i, is at its maximum value of 1° at the wing tip and the frequency parameter $\omega = 1{\cdot}9873$. The maximum amplitude of heave, $z = 1$ m (scaled for a semispan of 0·73254 m), also occurs at the wing tip. The initial mesh consists of 454,010 elements and 76,227 nodes and the initial solution is the converged steady state solution with no oscillation taking place. Figure 2 shows a detail of the initial mesh on the surface of the wing and in the symmetry plane. The computed pressure coefficient distributions produced at three levels in the third cycle of the computation on a plane passing through a section of the wing taken at 98 per cent of the chord are displayed in Figure 3. The lift at intervals in the third cycle of the computation is shown in Figure 4. During the course of the calculation it was noted that with the parameters set at $\gamma = 0{\cdot}2$ and $\beta = 0{\cdot}5$, no remeshing occurred and the deforming mesh algorithm alone was sufficient to adapt the mesh for the small-scale displacements which this problem entailed.

The final example considered illustrates the performance of the proposed procedure for the truly transient development of the flow over a three-dimensional body which is subject to large displacements. The freestream conditions correspond to a Mach number $M_\infty = 0{\cdot}3$ and the body is initially at zero angle of attack. An initial steady state solution is computed by performing 2000 time steps of the flow algorithm on the initial mesh. The body is then moved through the domain with a

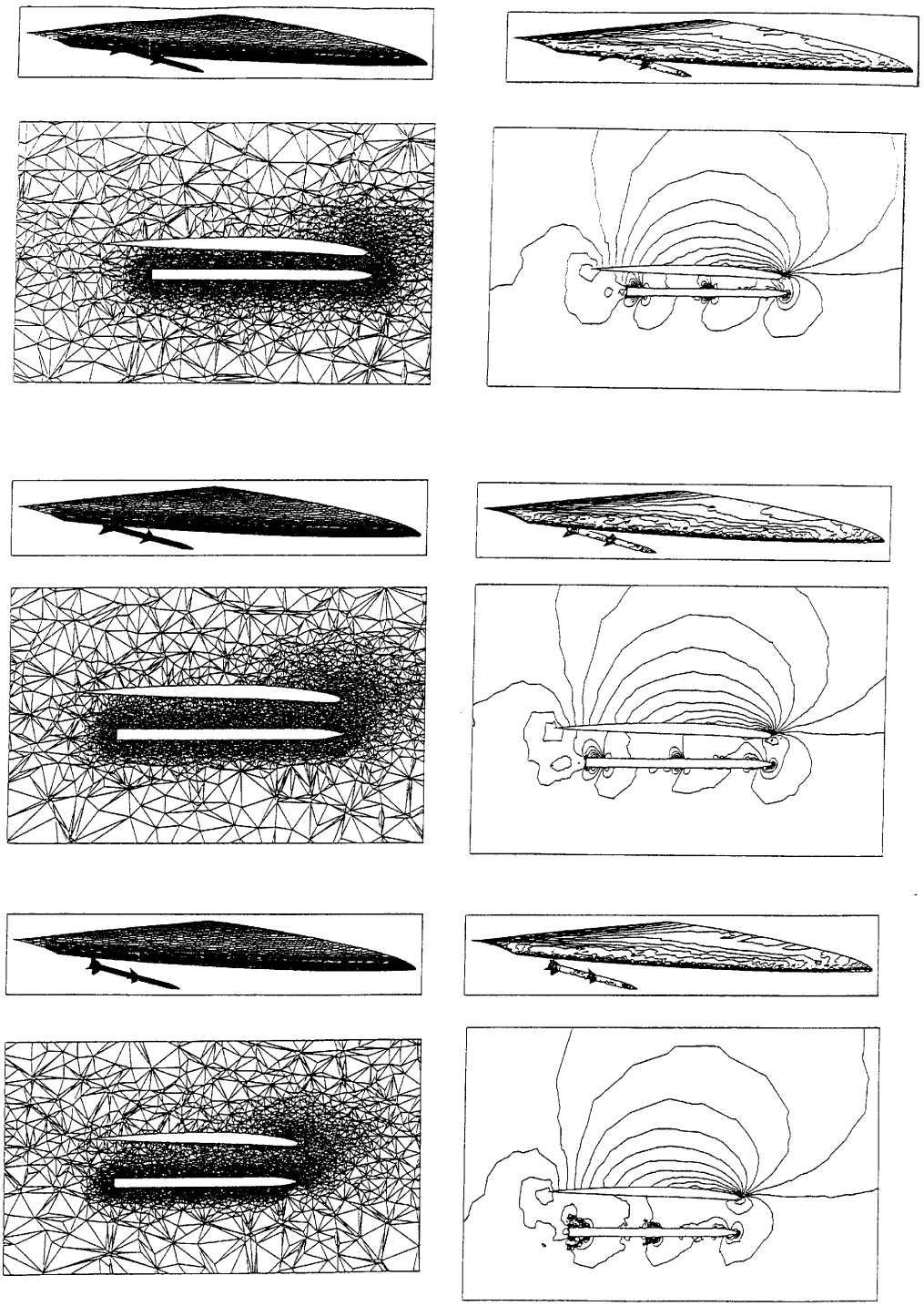

Figure 1. Pseudotransient motion of a missile configuration mounted initially underneath a wing, showing details of sequence of meshes and computed Mach number contours

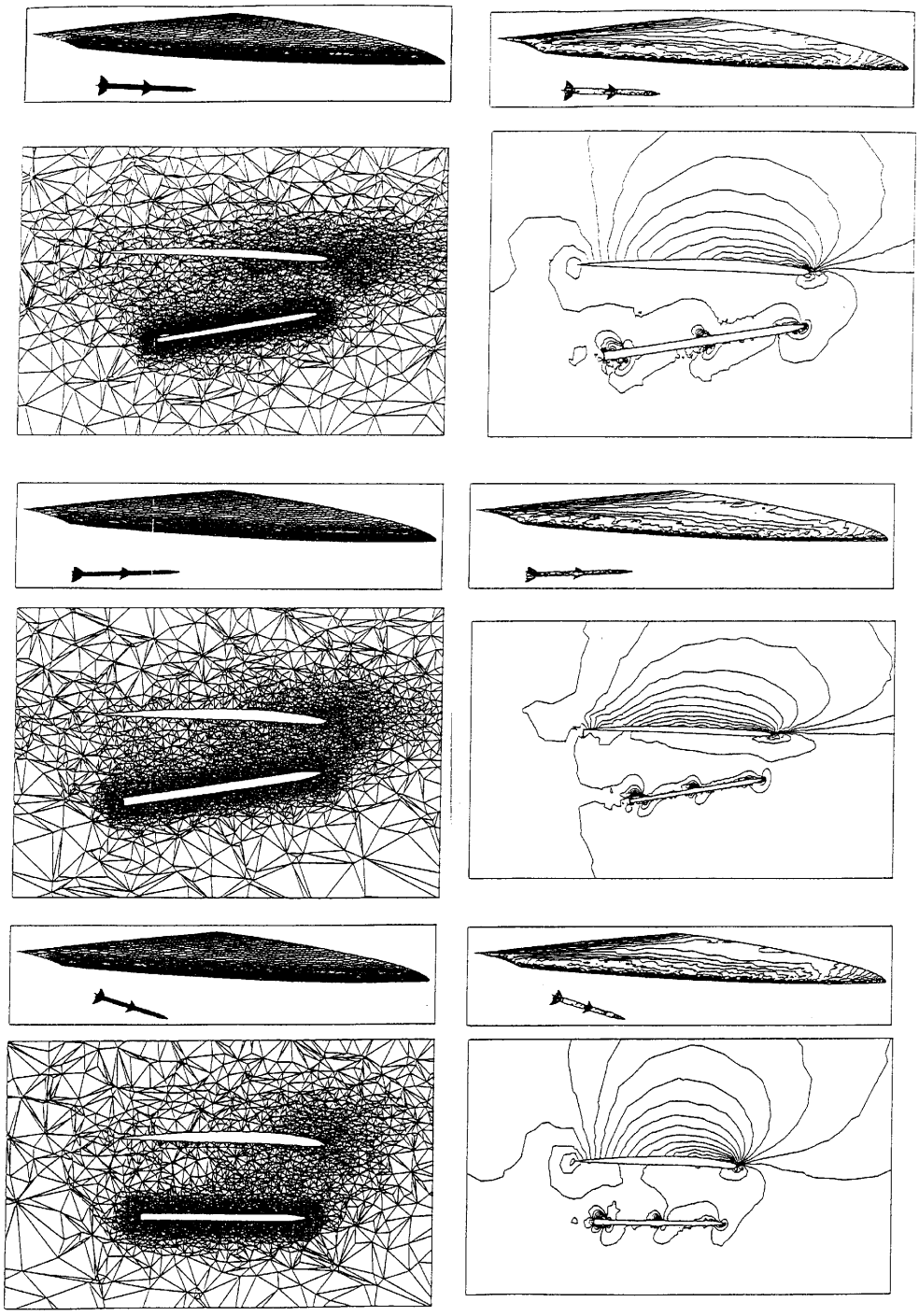

Figure 1. (*continued*)

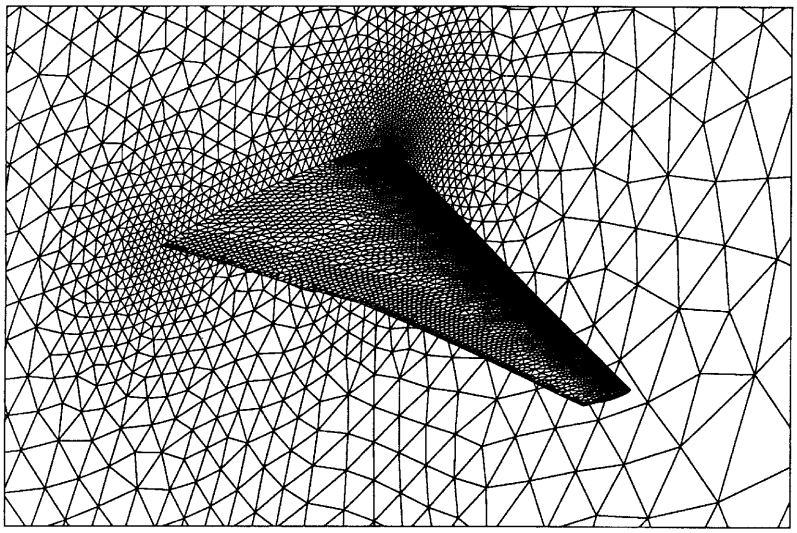

Figure 2. Oscillating wing with freestream conditions $M_\infty = 0{\cdot}713$ and $\alpha_0 = 0{\cdot}52$: detail of surface triangulation of initial mesh

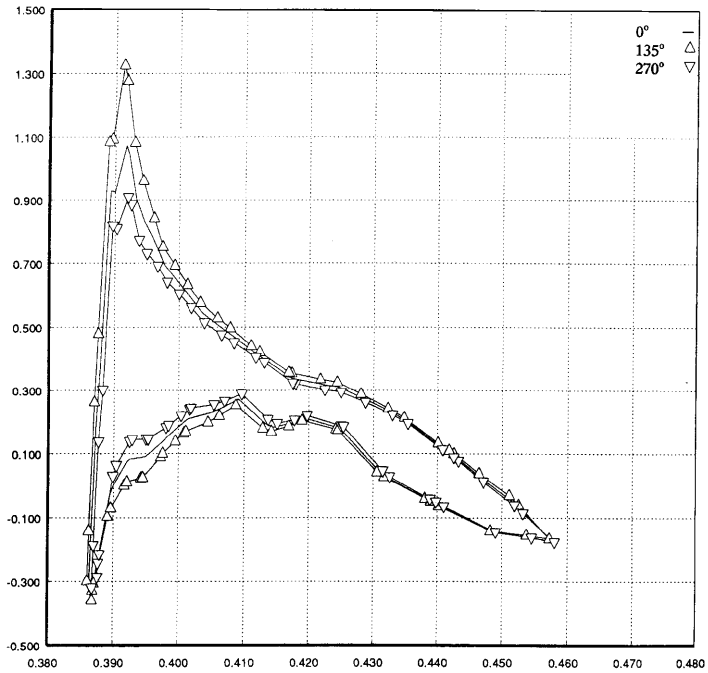

Figure 3. Oscillating wing with freestream conditions $M_\infty = 0{\cdot}713$ and $\alpha_0 = 0{\cdot}52$: distribution of pressure coefficient C_p over wing at a section taken at 98 per cent of chord, at intervals in third cycle of computation

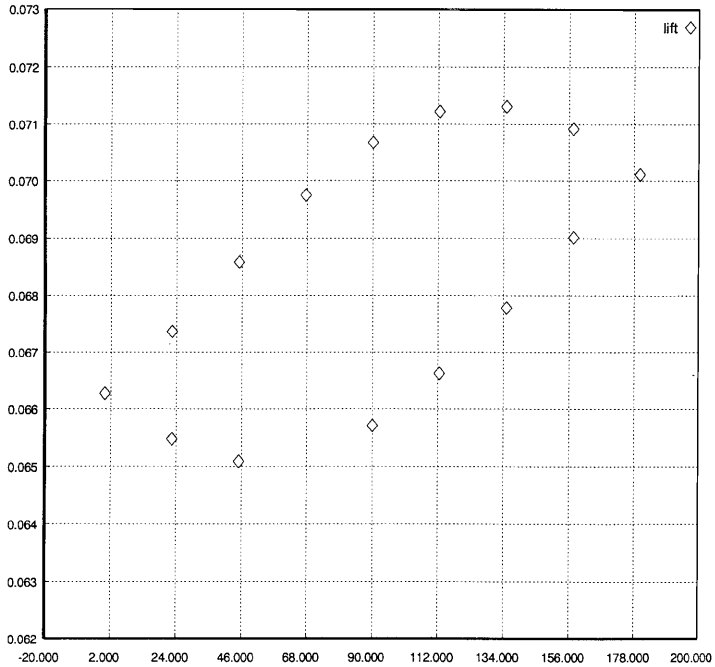

Figure 4. Oscillating wing with freestream conditions $M_\infty = 0{\cdot}713$ and $\alpha_0 = 0{\cdot}52$: lift at intervals in third cycle of computation

prescribed velocity and rotation. A selection of the sequence of meshes produced during the computation and the corresponding computed Mach number contours is shown in Figure 5. During the course of the computation the remeshing parameters were set at $\gamma = 0{\cdot}2$ and $\beta = 0{\cdot}5$ and 10 remeshings were required.

7. CONCLUSIONS

The solution of transient problems involving compressible inviscid aerodynamic flows in three dimensions, in the presence of moving boundary components, by unstructured mesh methods has been considered. A strategy for accomplishing the necessary mesh adaptation has been discussed and a promising approach has been implemented. However, further work is necessary to confirm the accuracy of this approach before it can be confidently employed in the analysis of large-scale practical problems.

ACKNOWLEDGMENT

The authors acknowledge the partial support provided for this work by British Aerospace (AIRBUS) Ltd., Technical Monitor Dr J. Szmelter.

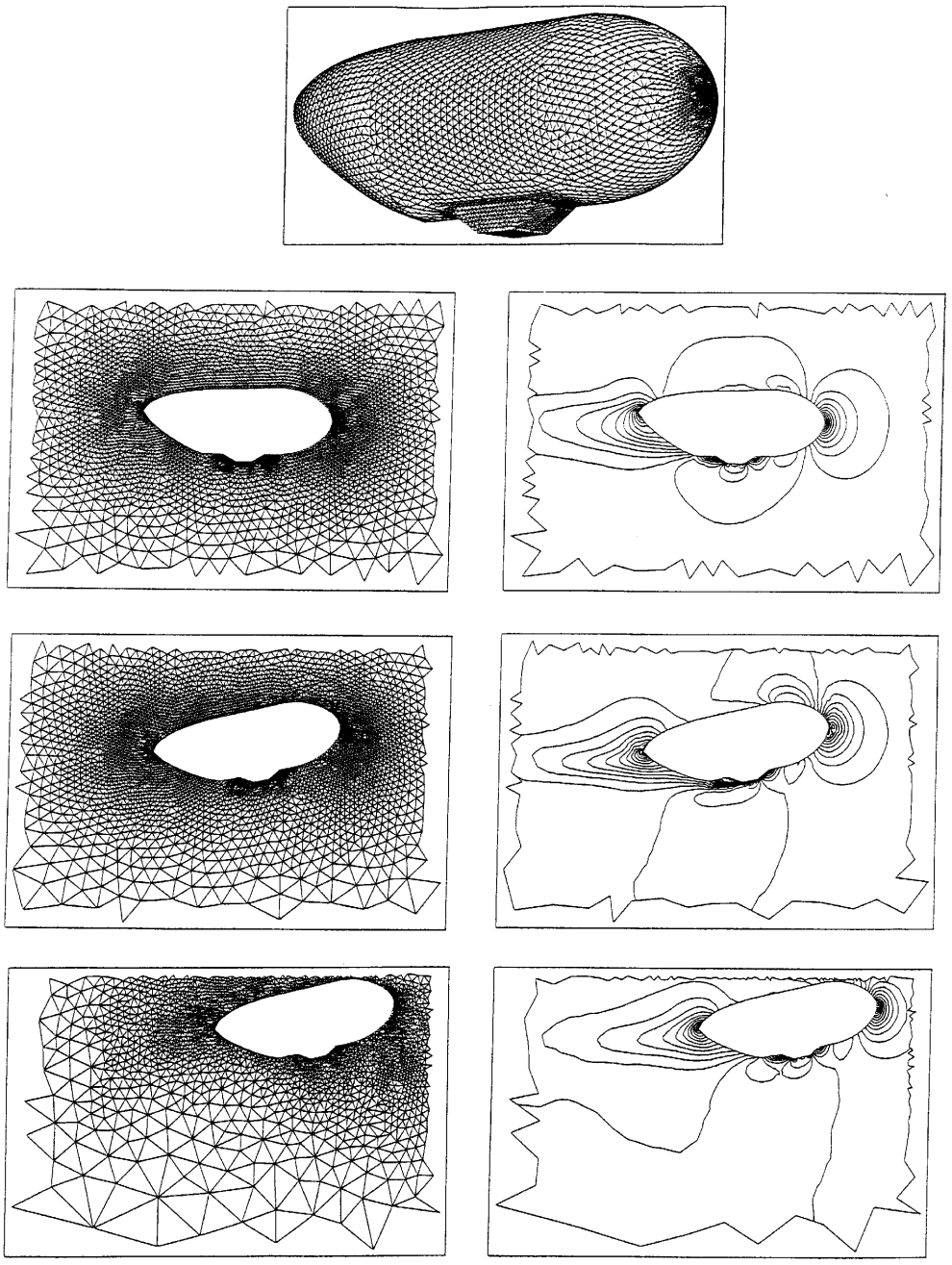

Figure 5. Transient motion of a 3D body subject to large displacements, showing details of mesh and corresponding Mach number contours at different locations

REFERENCES

1. T. Barth and H. Deconinck (eds), *AGARD Rep. R-787*; *Unstructured Grid Methods for Advection Dominated Flows*, AGARD, Paris, 1992.
2. J. Peiró, J. Peraire and K. Morgan, 'FELISA system reference manual. Part 1: Basic theory', *University of Wales Swansea Rep. CR/821/94*, 1994.
3. D. Mavriplis, 'A three dimensional multigrid Reynolds averaged Navier–Stokes solver for unstructured meshes', *AIAA Paper 94-1878*, 1994.
4. J. Peraire, K. Morgan and J. Peiró, 'Unstructured finite element mesh generation and adaptive procedures for CFD', *AGARD Conf. Proc. 464: Application of Mesh Generation to Complex 3-D Configurations*, AGARD, Paris, 1990, pp. 18.1–18.12.
5. O. Hassan, E. J. Probert, K. Morgan and J. Peraire, 'Mesh generation and adaptivity for the solution of compressible viscous high speed flows', *Int. j. numer. methods eng.*, **38**, 1123–1148 (1995).
6. R. Löhner, 'Numerical simulation of a blast inside a Boeing 747', *AIAA Paper 93-3091*, 1993.
7. E. J. Probert, O. Hassan, K. Morgan and J. Peraire, 'An adaptive finite element method for transient compressible flows', *Int. j. numer. methods eng.*, **32**, 1145–1159 (1991).
8. R. Löhner, 'New ALE adaptive unstructured methodology for the simulation of moving bodies', *AIAA Paper 94-0414*, 1994.
9. E. J. Probert, O. Hassan, K. Morgan and J. Peraire, 'An adaptive finite element method for transient compressible flows with moving boundaries', *Int. j. numer. methods eng.*, **32**, 751–765 (1991).
10. K. Morgan, J. Peraire, J. Peiró and O. Hassan, 'Unstructured grid methods for high speed compressible flows', in J. R. Whiteman (Editor), *The Mathematics of Finite Elements and Applications—Highlights 1993*, John Wiley and Sons, Chichester, 215–241, 1994.
11. N. P. Weatherill, 'Delaunay triangulation in computational fluid dynamics', *Comput. Math. Appl.*, **24**, 129–150 (1992).
12. N. P. Weatherill and O. Hassan, 'Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints', *Int. j. numer. methods eng.*, **37**, 2005–2039 (1994).
13. N. P. Weatherill, O. Hassan, M. J. Marchant and D. L. Marcum, 'Adaptive inviscid flow simulations using distributions of sources', *Proc. 8th Int. Conf. on Numerical Methods in Laminar and Turbulent Flow*, Pineridge, Swansea, 1993, pp. 18–23.
14. N. P. Weatherill, O. Hassan, M. J. Marchant and D. L. Marcum, 'Calculation of steady compressible flowfields with the finite element method', *AIAA Paper 93-0341*, 1993.
15. J. Peraire, J. Peiró and K. Morgan, 'Finite element multigrid solution of Euler flows past installated aero-engines', *Comput. Mech.*, **11**, 433–451 (1993).
16. J. Bonet and J. Peraire, 'An alternating digital tree (ADT) algorithm for geometric searching and intersection problems', *Int. j. numer. methods eng.*, **31**, 1–17 (1990).

