INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2007; **53**:1243–1266
Published online 6 July 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1255

A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing

O. Hassan[1,*,†], K. A. Sørensen[2], K. Morgan[1] and N. P. Weatherill[1]

[1]*Department of Civil Engineering, University of Wales, Wales, Swansea SA2 8PP, U.K.*
[2]*Numerical Simulation, EADS Military Aircraft, 81663 Munich, Germany*

SUMMARY

A method for the solution of time-dependent, turbulent, compressible flows involving geometries that change in time is presented. The governing equations are discretized using a finite volume method using a special dual mesh definition, specially tailored for the hybrid meshes used. A geometrically conservative way of treating these meshes is formulated. The discretized equations are implicit in time, and are solved by a dual time approach. The subiterations involved are performed using multigrid acceleration with explicit relaxation. For moving geometries, the mesh is deformed using a spring analogy approach combined with local remeshing. The approach is demonstrated for several flows of practical interest. Copyright © 2006 John Wiley & Sons, Ltd.

Received 2 November 2005; Revised 9 March 2006; Accepted 10 March 2006

KEY WORDS: Navier–Stokes; time accurate; remeshing; deforming mesh; turbulent; hybrid unstructured mesh; agglomerated multigrid

1. INTRODUCTION

Many problems of industrial and academic relevance involve compressible fluid flows around geometries which change with time. Manoeuvering aircraft not only experience geometry changes through the deployment of control surfaces, but also through aeroelastic effects which, for flexible aircraft, may change the aerodynamic characteristics significantly. In both military and civilian aircraft industries the simulation of multiple bodies in relative motion is of great interest. Such scenarios occur when stores are released from aircraft, the simulation of propellers or turbines, the simulation of active sensors or in formation flying such as air refuelling, to mention only a few. As can be seen from these areas of interest, the problems may involve motion that is essentially

*Correspondence to: O. Hassan, Department of Civil Engineering, University of Wales, Wales, Swansea SA2 8PP, U.K.
†E-mail: o.hassan@swansea.ac.uk

known *a priori* or motion directly coupled with the flowfield and dynamics of the problem. Some problems may be well approximated by simulating the geometry changes as the movement of two or more rigid bodies, while others involve significant deformation of components.

There are mainly two approaches currently in use for the simulation of these types of problems. The Chimera method [1] consists of supplying each separate body with a dedicated mesh. The meshes are then numerically combined during the calculation. The approach is in use for both structured [2] and unstructured [3] grids. For practical use, this approach usually requires fully automatic hole-cutting procedures, where the meshes are merged automatically for each geometry change. While robust and fast, these approaches are normally not conservative and experience is often needed to ensure adequate numerical resolution through the entire simulation. The approach works well for problems involving large relative motion of components and allows for small component deformation by coupling with a mesh deformation approach. However, for problems involving large deformations of components which cannot be broken down into smaller entities in relative motion, the method is only applicable in combination with other approaches.

The alternative presented here, involves the use of a single, consistent, mesh for each time step. With time, the mesh needs to adapt to the changing geometry. If the geometry changes are small, mesh deformation approaches may be applied, keeping the connectivity unchanged between time steps but deforming the elements so as to conform to the correct geometrical situation [4–7]. Such approaches are fast and have good numerical characteristics, as long as the deformed meshes are of good quality. For simulations involving large movements, however, these methods are usually not capable of retaining an adequate level of local mesh quality [8]. In such cases, the use of alternative techniques becomes necessary. Mesh quality enhancement techniques, which modify the structure of the mesh, can be utilized in regions where the mesh exhibits bad quality measures [9]. An alternative approach is to regenerate the entire mesh when it becomes necessary to do so to prevent unacceptable degrees of mesh distortion [10]. This approach is expensive, as it requires the frequent involvement of a mesh generation procedure, and thus also repeatedly introduces global interpolation errors between time steps.

The approach described in this paper is based on combining the mesh deformation method with a local remeshing approach [11–14]. For regions where the mesh deformation strategy produces meshes of locally adequate quality, the approach is equivalent to a normal mesh deformation approach. However, in regions where the local element quality is deteriorated, a hole is cut in the mesh which is subsequently remeshed using an unstructured mesh generator. In this way the remeshing effort is normally considerably smaller than for global remeshing strategies. The solution on the mesh at the previous time level is interpolated to the new mesh and the solution is advanced on this mesh. This approach has the advantage of reducing the amount of remeshing required, thus minimizing the regions where interpolation errors are introduced.

To date, this approach has only been applied to the solution of unsteady inviscid flows with moving boundary components. In this paper, we present an extension to enable the simulation of the unsteady viscous flows with moving boundary components to be carried out. The mesh movement algorithm has to be modified to handle the highly stretched elements which are employed in the boundary layers. In addition, when the distance between the various components imposes a restriction on the number of generated boundary layers, it is necessary to locally regenerate the viscous layers during the remeshing process. It will be demonstrated that the combined procedure is currently feasible, in terms of both computational costs and memory requirements. The practical examples which are included show that the method is robust and applicable to geometries of a complication level experienced in industry.

2. PROBLEM FORMULATION

The time-dependent, compressible Navier–Stokes equations in integral form on a three-dimensional Cartesian domain $\Omega \subset \mathbb{R}^3$, with surface $\partial\Omega$, can be expressed as

$$\int_{\Omega} \frac{\partial U_i}{\partial t}\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega} F_{ij} n_j \,\mathrm{d}\mathbf{x} = \int_{\partial\Omega} G_{ij} n_j \,\mathrm{d}\mathbf{x}, \qquad i = 1, \ldots, 5, \;\; j = 1, 2, 3 \tag{1}$$

where n_j is the outward unit normal vector to $\partial\Omega$ and the unknown vector of the conservative variables is given by

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho\varepsilon \end{bmatrix} \tag{2}$$

Here ρ denotes the fluid density, u_i the ith component of the velocity vector and ε the specific total energy. The inviscid and viscous flux tensors are defined by

$$\mathbf{F}_j = \begin{bmatrix} \rho u_j \\ \rho u_1 u_j + p\delta_{1j} \\ \rho u_2 u_j + p\delta_{2j} \\ \rho u_3 u_j + p\delta_{3j} \\ u_j(\rho\varepsilon + p) \end{bmatrix} \tag{3}$$

and

$$\mathbf{G}_j = \begin{bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ u_k\tau_{kj} - q_j \end{bmatrix} \tag{4}$$

respectively. In these definition,

$$\tau_{ij} = -\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{5}$$

is the deviatoric stress tensor, μ is the dynamic viscosity coefficient and

$$q_j = -k \frac{\partial T}{\partial x_j} \tag{6}$$

is the heat flux where k is the thermal conductivity and T is the absolute temperature. It is assumed that the viscosity varies with temperature according to Sutherland's law

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + 110}{T + 110} \tag{7}$$

where the temperature is expressed in degrees Kelvin and (μ_0, T_0) is a reference state. It is assumed that the Prandtl number

$$Pr = \frac{c_p \mu}{k} \tag{8}$$

is constant and equal to 0.72. In this expression, c_p is the specific heat at constant pressure. The system is closed by assuming the gas to be calorically perfect, setting

$$p = \rho R T \tag{9}$$

and

$$\varepsilon = c_v T + \tfrac{1}{2} u_k u_k \tag{10}$$

where R is the real gas constant and $c_v = c_p - R$ is the specific heat at constant volume. Throughout this work, the ratio of the specific heats,

$$\gamma = \frac{c_p}{c_v} \tag{11}$$

is set to $\gamma = 1.4$. For turbulent calculations, the Spalart–Allmaras one equation model [15] is employed, with the turbulent Prandtl number set equal to 0.9.

3. HYBRID UNSTRUCTURED MESH GENERATION

The process of mesh generation begins with the discretization of the computational domain into a mesh of tetrahedral elements. The element size distribution is governed by a user-specified mesh control function [16]. Hybrid unstructured meshes are constructed by merging certain elements of this tetrahedral mesh. The steps involved in this procedure can be summarized as:

1. Quadrilateral elements are generated on the wall surfaces. This is accomplished by an indirect method of combining triangles, coupled with a splitting scheme to guarantee meshes consisting of quadrilateral elements only. Mesh cosmetics and Laplacian smoothing are employed to improve the surface mesh quality.
2. The advancing layer method [17] is employed to generate stretched elements adjacent to those boundary surface components which represent solid walls. The height and number of layers is specified by the user, in such a way that the expected boundary layer profile is

adequately resolved. The implementation starts from a triangular mesh on the surface of the wall, which means that each quadrilateral element has to be treated as two triangles in the generation process. The layers are constructed by generating points along prescribed lines and connecting the generated points, by using advancing front mesh generation concepts, to form tetrahedral elements. For the first layer, the prescribed lines are normal to the surface, while for succeeding layers, smoothing of the line directions is employed. Surface intersections require special treatment [17]. Point generation ceases before the prescribed number of layers is reached if an intersection appears or if the local mesh size is close to that specified in the user–specified mesh distribution function.

3. The remainder of the domain is meshed using a standard isotropic Delaunay procedure [18].
4. Tetrahedra are merged to form prisms and pyramids. Starting at the boundary, the concept employed while generating the advancing layers is used to determine the point to be generated from each boundary point. For each layer, each boundary triangle is considered in turn and a prism is generated if all three vertices of the triangle have generated new points, a pyramid is generated if two of the vertices have successfully generated new points and a tetrahedron will be generated if there is only one vertex generating a new point.
5. Hexahedral elements are generated by using the fact that pairs of triangles on the wall are the result of splitting each of the original quadrilateral elements generated on the surface. The two prisms constructed on these two triangles will be combined to form a hexahedron. This combination can only be performed if the two tetrahedra in the isotropic mesh containing the two triangles grown from the original quadrilateral surface elements can be merged into a pyramid. When this condition is not satisfied, it is possible to overcome the problem by inserting a point at the centroid of the hexahedron of the final layer and then splitting this hexahedron into five pyramids and two tetrahedra.

4. EQUATION DISCRETIZATION

The discretization of Equation (1) is accomplished by using a cell vertex finite volume method. This requires the construction of a dual mesh, in which each cell of the dual is associated with a single vertex of the hybrid mesh.

4.1. Dual mesh construction

The median dual, illustrated in Figure 1, when used on the hybrid meshes generated by the procedure in Section 3 may produce control volumes that do not contain the associated node. This may occur locally in regions of high surface curvature or at the interface between the inner and outer meshes. To avoid this, the information contained in the original tetrahedral mesh is used. The quadrilateral faces of the merged elements are in general not planar and must be defined accordingly. By taking advantage of the information available from the simplex mesh, the quadrilateral element faces are defined as the union of the two triangular faces originating from the tetrahedral mesh. In this way, the surfaces of the elements are defined as closed collections of planar triangular facets, which are valid due to the validity of the original tetrahedral mesh. The centroid of a quadrilateral face is defined as the midpoint of the edge shared between the two triangular facets making up the quadrilateral face. Centroids of triangular faces are taken as the nodal average, since the triangles

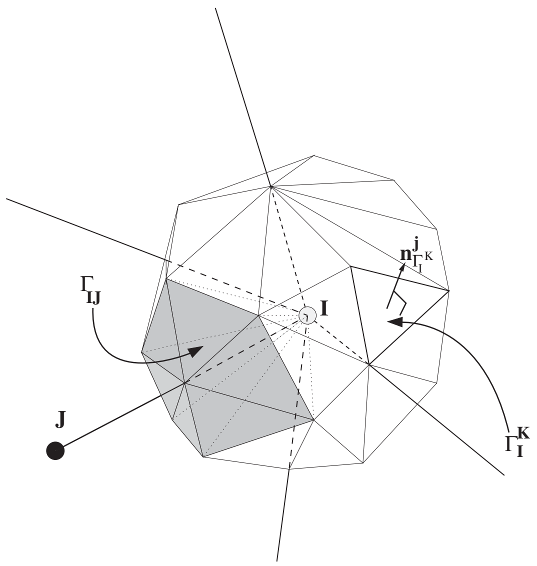

Figure 1. Illustration of the dual mesh surrounding an internal node I.

are considered planar. With such well-defined element surfaces, it is now possible to define element centroids that ensure the validity of the dual mesh:

- For a pyramid element, the centroid is defined to be in the triangular merging face between the two tetrahedra making up the element, according to the formula

$$\mathbf{x}_m = \tfrac{1}{5}(2\mathbf{x}_1 + 2\mathbf{x}_3 + \mathbf{x}_5) \tag{12}$$

  where the local node indexes 1, 3 and 5 denote the nodes on the diagonal of the quadrilateral base and pyramid top, respectively.
- The centroid of a prism element is defined as the nodal average of the midpoints of the quadrilateral face centroids in the element.
- The centroid of a hexahedral element is defined as the midpoint of the diagonal edge of the common quadrilateral face of the two prisms making up the element.

Illustration of the construction of the dual is shown in Figures 2, 3 and 4 for pyramids, prisms and hexahedra, respectively. In the figures, the left illustration shows the position of the face centroids as open circles and the element centroid as a filled circle. The illustrations on the right show the part of the dual mesh surrounding a node that is contained within the element. In two dimensions, the centroid of a quadrilateral element is taken as the midpoint of the diagonal originating from the triangular mesh. These definitions of the dual mesh coincide with the median dual mesh for regular elements.

4.2. Treatment of inviscid fluxes

To perform the numerical integration of the inviscid fluxes, a set of coefficients is calculated for each edge using the dual mesh segment associated with the edge. The values of these coefficients

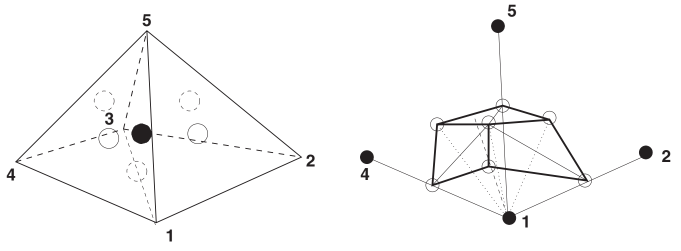

Figure 2. Illustration of the construction of the dual mesh for a pyramidal element.

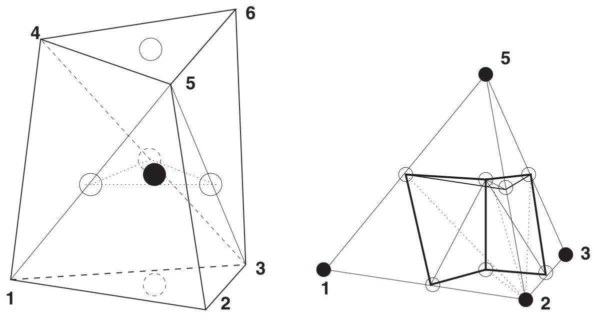

Figure 3. Illustration of the construction of the dual mesh for a prismatic element.

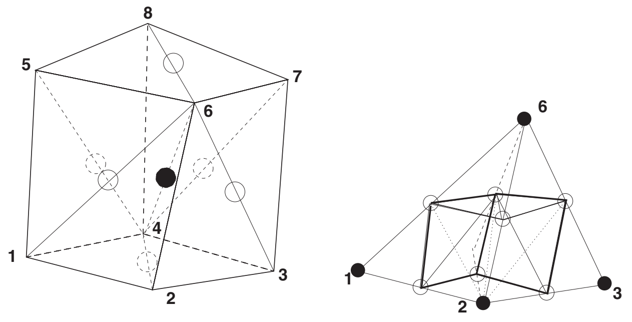

Figure 4. Illustration of the construction of the dual mesh for a hexahedral element.

for an internal edge are given by the formula

$$C_j^{IJ} = \sum_{K \in \Gamma_{IJ}} A_{\Gamma_I^K} n_j^{\Gamma_I^K} \tag{13}$$

where $A_{\Gamma_I^K}$ is the area of facet Γ_I^K and $n_j^{\Gamma_I^K}$ is the outward unit normal vector of the facet from the viewpoint of node I, see Figure 1. In a similar way, the contribution to boundary integrals from the computational boundary can be expressed using a coefficient

$$D_j^{IJ} = \sum_{K \in \Gamma_{IJ}^B} A_{\Gamma_I^K} n_j^{\Gamma_I^K} \tag{14}$$

where Γ_{IJ}^B is the set of dual mesh faces on the computational boundary touching the edge between nodes I and J. In this case, $n_j^{\Gamma_I^K}$ denotes the normal of the facet in the outward direction of the computational domain.

The numerical integration of a flux over the dual mesh segment associated with an edge is then performed by assuming the flux to be constant, and equal to its approximated value at the midpoint of the edge, over the segment, i.e. a form of midpoint quadrature. The calculation of a surface integral for the inviscid flux over the control volume surface for node I is then given by the formula

$$\int_{\partial\Omega_I} F_{ij} n_j \, \mathrm{d}\mathbf{x} \approx \sum_{J \in \Lambda_I} \frac{C_j^{IJ}}{2} (F_{ij}^I + F_{ij}^J) + \sum_{J \in \Lambda_I^B} D_j^{IJ} F_{ij}^I \tag{15}$$

where Λ_I denotes the set of nodes connected to node I by an edge and Λ_I^B denotes the set of nodes connected to node I by an edge on the computational boundary. The last term, thus, only appears if I is a boundary node.

4.3. Treatment of viscous fluxes

The integral in Equation (1) that contains the viscous fluxes may be represented in a similar manner as

$$\int_{\partial\Omega_I} G_{ij} n_j \, \mathrm{d}\mathbf{x} \approx \sum_{J \in \Lambda_I} \frac{C_j^{IJ}}{2} (G_{ij}^I + G_{ij}^J) + \sum_{J \in \Lambda_I^B} D_j^{IJ} G_{ij}^I \tag{16}$$

However, the viscous fluxes involve gradients of the flow variables and the nodal values of these gradients need to be determined before this formula can be employed. The evaluation of the gradient of a function may be performed in several ways within the finite volume framework. Here, the approach described in Reference [19] is applied, employing a compact stencil in the edge directions.

This is achieved by splitting the evaluation of the derivative at an edge midpoint

$$\partial_j^h u_i^{IJ} = \tfrac{1}{2}(\partial_j^h u_i^I + \partial_j^h u_i^J) \tag{17}$$

where $\partial_j^h u_i^I$ denotes the standard finite volume first-order derivative with respect to x_j of the ith component of the velocity at node I, into two components, tangential and normal to the edge, as

$$\partial_j^h u_i^{IJ} = \partial_j^h u_i^{IJ}|_t + \partial_j^h u_i^{IJ}|_n \tag{18}$$

where

$$\partial_j^h u_i^{IJ}|_t = r_k^{IJ} \partial_k^h u_i^{IJ} r_j^{IJ} \tag{19}$$

is the component in the direction of the edge and

$$\partial_j^h u_i^{IJ}|_n = \partial_j^h u_i^{IJ} - r_k^{IJ} \partial_k^h u_i^{IJ} r_j^{IJ} \tag{20}$$

is the normal component. Here

$$r_j^{IJ} = \frac{1}{l_{IJ}}(x_j^J - x_j^I) \tag{21}$$

is the unit directional vector in the edge direction from node I to node J and

$$l_{IJ} = |\mathbf{x}^J - \mathbf{x}^I| \tag{22}$$

is the edge length. The component of the derivative along the edge is then replaced by a finite difference approximation

$$\hat{\partial}_j^h u_i^{IJ}|_t = \frac{1}{l_{IJ}}(u_i^J - u_i^I) r_j^{IJ} \tag{23}$$

which is second-order accurate at the edge midpoint, but only requires two nodes in its evaluation, as compared to four for the form of Equation (19). The scheme

$$\int_{\partial\Omega_I} \frac{\partial u_i}{\partial x_j} n_k \, \mathrm{d}\mathbf{x} \approx \sum_{J\in\Lambda_I} C_k^{IJ}(\partial_j^h u_i^{IJ}|_n + \hat{\partial}_j^h u_i^{IJ}|_t) + \sum_{J\in\Lambda_I^B} D_k^{IJ} \partial_j^h u_i^I \tag{24}$$

can thus be employed in the evaluation of the viscous fluxes, where the gradients along edges are evaluated with the compact finite difference scheme and the remaining components are calculated in the standard finite volume manner.

The viscous terms contribute mostly in the boundary layers, which are characterized by high gradients normal to the wall and, except for localized regions such as around shocks, relatively small gradients tangential to the wall. The introduction of quasi-regular quadrilateral/hexahedral/prismoidal meshes with grid lines parallel and normal to the wall ensures that the high gradients are captured by the compact stencil and that the five point stencil terms are marginalized. The use of these meshes, thus, essentially reduces the current treatment of the viscous terms to the well known finite difference three-point scheme used for structured meshes.

4.4. ALE formulation

For problems involving mesh deformation, the spatial discretization of the governing equations is affected. One way of solving such problems is to retain the Eulerian form of the governing Equations (1), but allowing the control volumes to move independently of the flow. This approach is termed arbitrary Lagrangian–Eulerian (ALE) [20]. By allowing the control volumes to be time-dependent, the governing Equation (1) can be written as

$$\int_{\Omega(t)} \frac{\partial U_i}{\partial t} \mathrm{d}\mathbf{x} + \int_{\partial\Omega(t)} F_{ij} n_j \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega(t)} G_{ij} n_j \, \mathrm{d}\mathbf{x} \tag{25}$$

and, by applying the generalized Leibnitz rule on the time term, the equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} U_i \, \mathrm{d}\mathbf{x} + \int_{\partial\Omega(t)} (F_{ij} - v_j U_i) n_j \, \mathrm{d}\mathbf{x} = \int_{\partial\Omega(t)} G_{ij} n_j \, \mathrm{d}\mathbf{x} \tag{26}$$

results, where v_j is the velocity of the control volume boundary. The Eulerian description of flows incorporating moving meshes is, therefore, achieved by the addition of an extra flux term, termed the ALE flux. The time derivative term is approximated as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_I(t_n)} U_i\,\mathrm{d}\mathbf{x}\bigg|_{t=t_n} \approx \partial_h^t(VU_i)^{I,n} \equiv \frac{1}{\Delta t}(V_I^n U_i^{I,n} - V_I^{n-1}U_i^{I,n-1}) \tag{27}$$

for a first-order scheme and as

$$\partial_h^t(VU_i)^{I,n} = \frac{1}{\Delta t}\left(\frac{3}{2}V_I^n U_i^{I,n} - 2V_I^{n-1}U_i^{I,n-1} + \frac{1}{2}V_I^{n-2}U_i^{I,n-2}\right) \tag{28}$$

for a second-order scheme.

An important property of Equation (26) is geometric conservation. Geometric conservation ensures that the control-volume movement itself has no direct effect on the fluxes, in the sense that if the unknown field is constant, the solution does not change in time, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)}\mathrm{d}\mathbf{x} - \int_{\partial\Omega(t)} v_j n_j\,\mathrm{d}\mathbf{x} = 0 \tag{29}$$

It is desirable to retain this quality numerically, after discretization. This was first shown in Reference [21] and since advocated in References [8, 22]. For a numerical scheme, the discrete ALE term for node I at time level n is termed $P_i^{I,n}$, so that

$$P_i^{I,n} \approx \int_{\partial\Omega_I(t_n)} v_j^n U_i^n n_j^n\,\mathrm{d}\mathbf{x} \tag{30}$$

The numerical integration of this term is performed in the usual way of summing the edge contributions. It is, therefore, convenient to introduce the numerical ALE coefficients S_{IJ}^n, T_{IJ}^n, for the edge between nodes I and J, such that

$$P_i^{I,n} = \sum_{J\in\Lambda_I}\frac{S_{IJ}^n}{2}(U_i^{I,n} + U_i^{J,n}) + \sum_{J\in\Lambda_I^B} T_{IJ}^n U_i^{I,n} \tag{31}$$

The task of finding ALE coefficients that render the numerical scheme geometric conservative is not trivial and has been treated by several authors [6, 8, 21, 22]. Here the approach of Reference [6] is followed. This was originally stated for simplex meshes, but is described here in a form suitable for the hybrid meshes introduced in Section 3, with the control volume definitions of Section 4.1.

The key point in Reference [6] is the assumption that each node in the mesh moves in a linear fashion between time levels n and $n+1$. Under this assumption, the movement of a triangular facet can be easily described. An illustration of the dual mesh facet Γ_I^K is given in Figure 5. The points $\mathbf{x}_{c1}$, $\mathbf{x}_{c2}$ and $\mathbf{x}_{c3}$ are situated at the vertices of the facet and are moved in a linear fashion in space–time from time level n to time level $n+1$. The vertices may be element centroids, face centroids or edge midpoints for any kind of mesh, with a dual mesh consisting of planar triangular facets.

It can be shown [6], that the volume swept over by triangular facet Γ_I^K between time levels n and $n+1$ is equal to the volume shown in Figure 5, which is given as

$$\delta V_{\Gamma_I^K}^{n+1,n} = \int_{t_n}^{t_{n+1}}\int_{\Gamma_I^K} v_j^{\Gamma_I^K} n_j^{\Gamma_I^K}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \tag{32}$$

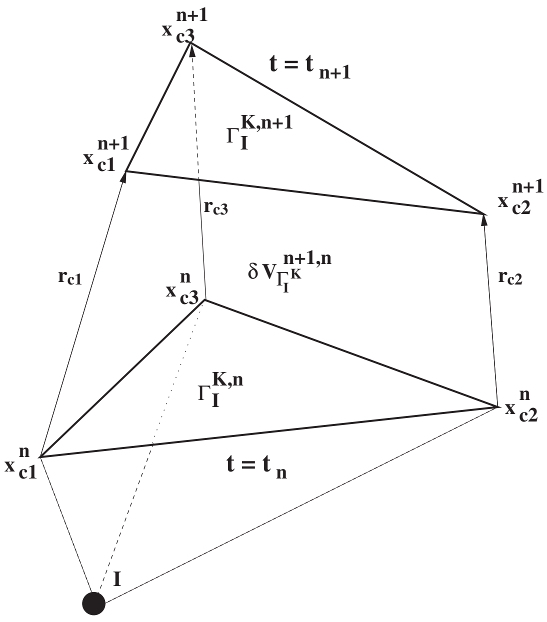

Figure 5. Illustration of the terminology used for the movement of a triangular facet of the dual mesh for node I.

This may be evaluated as, [6, 22],

$$\delta V_{\Gamma_I^K}^{n+1,n} = \tfrac{1}{9}(A_{\Gamma_I^K}^{n+1} n_j^{\Gamma_I^K,n+1} + A_{\Gamma_I^K}^{n} n_j^{\Gamma_I^K,n} + A_{\Gamma_I^K}^{*} n_j^{\Gamma_I^K,*})(r_j^{c1} + r_j^{c2} + r_j^{c3}) \tag{33}$$

where

$$A_{\Gamma_I^K}^{n+1}\mathbf{n}_{\Gamma_I^K}^{n+1} = \tfrac{1}{2}(\mathbf{x}_{c2}^{n+1} - \mathbf{x}_{c1}^{n+1}) \times (\mathbf{x}_{c3}^{n+1} - \mathbf{x}_{c1}^{n+1}) \tag{34}$$

$$A_{\Gamma_I^K}^{n}\mathbf{n}_{\Gamma_I^K}^{n} = \tfrac{1}{2}(\mathbf{x}_{c2}^{n} - \mathbf{x}_{c1}^{n}) \times (\mathbf{x}_{c3}^{n} - \mathbf{x}_{c1}^{n}) \tag{35}$$

$$A_{\Gamma_I^K}^{*}\mathbf{n}_{\Gamma_I^K}^{*} = \tfrac{1}{4}(\mathbf{x}_{c2}^{n+1} - \mathbf{x}_{c1}^{n+1}) \times (\mathbf{x}_{c3}^{n} - \mathbf{x}_{c1}^{n}) + \tfrac{1}{4}(\mathbf{x}_{c2}^{n} - \mathbf{x}_{c1}^{n}) \times (\mathbf{x}_{c3}^{n+1} - \mathbf{x}_{c1}^{n+1}) \tag{36}$$

and

$$\mathbf{r}_{c1} = \mathbf{x}_{c1}^{n+1} - \mathbf{x}_{c1}^{n} \tag{37}$$

$$\mathbf{r}_{c2} = \mathbf{x}_{c2}^{n+1} - \mathbf{x}_{c2}^{n} \tag{38}$$

$$\mathbf{r}_{c3} = \mathbf{x}_{c3}^{n+1} - \mathbf{x}_{c3}^{n} \tag{39}$$

A local geometric conservation law

$$\delta V_{\Gamma_I^K}^{n+1,n} = \Delta t\, v_j^{\Gamma_I^K,*} n_j^{\Gamma_I^K,*} A_{\Gamma_I^K}^{*} \tag{40}$$

can now be applied to the facet, where $v_j^{\Gamma_I^K,*}$ is an averaged facet velocity, $A^*_{\Gamma_I^K}$ is the averaged facet area and $n_j^{\Gamma_I^K,*}$ is the average normal between time levels n and $n+1$. The right-hand side of this equation can be recognized as the contribution of the numerical ALE term for a single facet, multiplied by the time step. Since the dual mesh definitions described in Section 4.1 are constructed by collections of triangular facets, the formulation of a geometric conservative scheme for a hybrid mesh is now straightforward. By setting

$$\Delta t S_{IJ}^{n+1} = \sum_{K\in\Gamma_{IJ}} \delta V_{\Gamma_I^K}^{n+1,n} \tag{41}$$

and, similarly, for the computational boundary

$$\Delta t T_{IJ}^{n+1} = \sum_{K\in\Gamma_{IJ}^B} \delta V_{\Gamma_I^K}^{n+1,n} \tag{42}$$

a numerical ALE flux that satisfies a geometric conservation law is found, since, by summing over the edges connected to a given node I,

$$\Delta t \sum_{J\in\Lambda_I} S_{IJ}^{n+1} + \Delta t \sum_{J\in\Lambda_I^B} T_{IJ}^{n+1} = \sum_{J\in\Lambda_I}\left[\sum_{K\in\Gamma_{IJ}} \delta V_{\Gamma_I^K}^{n+1,n} + \sum_{K\in\Gamma_{IJ}^B} \delta V_{\Gamma_I^K}^{n+1,n}\right] = V_I^{n+1} - V_I^n \tag{43}$$

Here, the last equality must be true since the control volume is closed. This ensures that the numerical geometric conservation law is satisfied when the backward Euler time discretization of Equation (27) is used.

If the second-order time discretization given in (28) is applied, however, a modified version of the ALE flux has to be employed to ensure numerical geometric conservation. Now, the modification

$$\hat{S}_{IJ}^{n+1} = \tfrac{3}{2} S_{IJ}^{n+1} - \tfrac{1}{2} S_{IJ}^n \tag{44}$$

$$\hat{T}_{IJ}^{n+1} = \tfrac{3}{2} T_{IJ}^{n+1} - \tfrac{1}{2} T_{IJ}^n \tag{45}$$

is adopted, [9], which makes the numerical ALE coefficient second order in time.

The velocity boundary conditions are applied by using the formula

$$\mathbf{u}_I^{w,n+1} = \frac{1}{\Delta t}(\mathbf{x}_I^{n+1} - \mathbf{x}_I^n) \tag{46}$$

for the wall velocity, if the first-order scheme is used, and

$$\mathbf{u}_I^{w,n+1} = \frac{1}{\Delta t}\left(\frac{3}{2}\mathbf{x}_I^{n+1} - 2\mathbf{x}_I^n + \frac{1}{2}\mathbf{x}_I^{n-1}\right) \tag{47}$$

if the second-order scheme is used. This is consistent with the dual mesh velocity definitions used in the two time discretization approaches.

For remeshed regions, the coordinates of the current time level are interpolated back to the previous time level. A new previous time level mesh, with the current mesh connectivities, is thus created. The mesh discretization is then performed in the same way as for the moved regions of the mesh. In the current implementation, the first-order time discretization scheme is used if remeshing has been performed.

4.5. *Artificial dissipation*

For stabilization, the JST scheme [23] is applied, where a third-order biharmonic term of the form

$$H_i^I \equiv \sum_{J\in\Lambda_I} H_i^{IJ} = \sum_{J\in\Lambda_I} D_{IJ}(E_j^J - E_j^I) \tag{48}$$

where

$$E_j^I = \sum_{K\in\Lambda_I} (U_j^K - U_j^I) \tag{49}$$

and the diagonal dissipation matrix is given by,

$$D_{IJ} = \frac{m_{IJ}}{|\Lambda_I| + |\Lambda_J|} \min\left(\frac{V_I}{\Delta\tau_I}, \frac{V_J}{\Delta\tau_J}\right) \tag{50}$$

Here $|\Lambda_I|$ is the number of edges connected to node I, V_I is the volume (area in two dimensions) of the control volume containing node I and $\Delta\tau_I$ is the local time step used in the explicit Runge–Kutta iterations of the solver, defined by [24]

$$\Delta\tau_I = V_I \left[\sum_{J\in\Lambda_I}\left(\|C_j^{IJ}\|\lambda_{\max}^I + \frac{2\|C_j^{IJ}\|^2(v_I + v_I^t)}{V_I}\right)\right]^{-1} \tag{51}$$

where

$$\lambda_{\max}^I = \frac{|u_j^I C_j^{IJ}|}{\|C_j^{IJ}\|} + c_I \tag{52}$$

Here c_I is the speed of sound at node I and $m_{IJ} = \varepsilon_4$ is a user-specified constant.

For regions with discontinuities, a first-order harmonic term is added, with the form

$$Q_i^I = \sum_{J\in\Lambda_I} \varepsilon_2 \alpha_{IJ} N_{IJ}(U_j^J - U_j^I) \tag{53}$$

The pressure switch

$$N_{ij}^{IJ} = \max(|\Delta p_I|, |\Delta p_J|)\delta_{ij} \tag{54}$$

where

$$\Delta p_I = 12\frac{\sum_{K\in\Lambda_I}(p_K - p_I)}{\sum_{K\in\Lambda_I}(p_K + p_I)} \tag{55}$$

ensures that this term is only significant in regions of strong pressure gradients.

To ensure that the biharmonic dissipation term disappears in the vicinity of discontinuities, the quantity, m_{IJ} in Equation (50) is now determined as

$$m_{IJ} = \max(0, \varepsilon_4\delta_{ij} - \varepsilon_2 N_{ij}^{IJ}) \tag{56}$$

The constants ε_2 and ε_4 are both set equal to 0.1 for the calculations in this paper.

4.6. Multigrid procedure

The discrete system obtained with the above describe procedure can be written as

$$R_i^I(\mathbf{U})=0, \qquad I\in[1,N],\ i\in[1,n] \tag{57}$$

A FAS multigrid procedure [25] is employed to produce the solution in an efficient manner. This consists of solving the equation system using a cascade of successively coarser meshes. The fine mesh equation

$$\mathbf{R}_f(\mathbf{U}_f)=0 \tag{58}$$

is first relaxed by a three-stage Runge–Kutta explicit procedure, yielding, for cycle s, an approximation of the solution on the fine mesh level, $\mathbf{U}_f^{s^*}$. On the next coarser level, the equation

$$\mathbf{R}_{c1}(\mathbf{U}_{c1})=\mathbf{S}_{c1} \tag{59}$$

is formed with the source term defined as

$$\mathbf{S}_{c1}=\mathbf{R}_{c1}(\mathbf{I}_{c1}^f\mathbf{U}_f^{s^*})-\mathbf{I}_{c1}^f(\mathbf{R}_f(\mathbf{U}_f^{s^*})) \tag{60}$$

where $\mathbf{I}_{c1}^f$ is a mapping of vectors from the fine to coarse mesh level, termed a restriction operator. Equation (59) is now relaxed, either by a Runge–Kutta step, or a multigrid step involving another coarse mesh level, $c2$. In this way, the equation is solved using several coarse mesh levels, each mesh level responsible for eliminating errors that appear locally as high frequency. The result is a coarse mesh approximation to Equation (59), $\mathbf{U}_{c1}^{s^*}$. The correction from the coarse mesh level on the fine mesh is given by,

$$\mathbf{U}_f^{s+1}=\mathbf{U}_f^{s^*}+\mathbf{I}_f^{c1}(\mathbf{U}_{c1}^{s^*}-\mathbf{I}_{c1}^f\mathbf{U}_f^{s^*}) \tag{61}$$

where $\mathbf{I}_{c1}^f$ is a mapping of vectors from the coarse to fine mesh level, termed a prolongation operator. This procedure is iterated until convergence. For the work presented here, typically five multigrid levels were applied. The multigrid cells are generated on the dual mesh level, creating the coarse grid cascade by merging dual mesh cells [26, 27]. The volume weighted average

$$U_I^{c1}=\frac{1}{V_I^{c1}}\sum_{J\in M_I}V_J^f U_J^f \tag{62}$$

is used as the restriction operator, where V_I denotes the volume at node I and M_I is the set of all fine mesh control volumes contained within a coarse mesh cell. The simple point injection scheme

$$U_I^f=U_J^{c1} \tag{63}$$

is used for prolongation. These formulae take advantage of the nested mesh property resulting from the agglomeration procedure. A more detailed description of the solution procedure is given in Reference [28].

5. MESH UPDATE PROCEDURE

The interest in this paper is the simulation of flows for which the geometry under consideration is moving with time. This means that the generated mesh will need to change to allow for the alteration in the shape of the computational domain. This can be achieved by remeshing the entire computational domain at each time step [10]. However, this is not only computationally expensive but may also result in reduced accuracy, due to the effects of interpolation errors. In addition, certain desirable features, such as geometric conservation, are difficult to ensure if the mesh structure changes. It is, therefore, good practice to avoid remeshing whenever possible and, instead, strive to retain the mesh connectivity. However, special care has to be taken when using both mesh movement and remeshing on meshes which contain hybrid anisotropic elements generated near the boundary layers.

5.1. Mesh movement

An obvious method to account for boundary movement, while retaining the mesh connectivity, is to use mesh movement [7]. For external flows, this is usually achieved by fixing the mesh on the far field boundary, while moving the mesh nodes on the geometry. The interior mesh nodes are then moved accordingly to achieve the desired mesh quality. A number of different mesh movement approaches have been investigated in the literature, but the approach used here assumes that the edges of the mesh behave like springs connecting the nodes [5, 7, 11]. The nodes on the moving geometry are moved in small increments, with the interior nodes being moved to ensure internal equilibrium for each increment. This is accomplished by solving the system

$$\sum_{J\in\Lambda_I} k_{IJ}(d_{IJ} - d_{IJ}^{\text{prev}}) = 0 \tag{64}$$

where d_{IJ}^{prev} and d_{IJ} are the lengths of the edge between nodes I and J before and after the movement, respectively, and k_{IJ} is the spring coefficient, which is taken to be the inverse of the edge length. With this approach, it is also possible to obtain an additional level of control over the mesh movement by varying the spring constants of the edges. This control is required when the algorithm is applied on meshes which contains boundary layer regions, where stretching of up to 1000 is present [29]. In this case, the moving mesh algorithm can easily produce elements with negative volume. The regeneration of these elements will result in elements which are, in general, identical to the original elements being moved rigidly. Hence, to avoid the complication of regenerating the boundary elements, the spring coefficient of the edges located in the boundary layers is increased to force a rigid movement of the elements in that region. In order to ensure the validity of the moved elements, this increase will be applied uniformly for all the elements of each layer. The increase in the edge coefficient is linearly relaxed for the last 5% of the layers of the anisotropic region. Typically 50 increments are employed with this procedure. The approach is robust and fast, usually requiring about 10–15% of the total CPU time required to solve the system time accurately. However, to accelerate the application of the mesh movement algorithm on hybrid viscous meshes, the nodes in the first 50% of the boundary layers are moved in the same prescribed manner as the boundary points. This normally results in a 50% saving in the CPU time required for the mesh movement procedure.

5.2. Local remeshing

For certain simulations, it is impossible to avoid remeshing if the necessary mesh quality is to be maintained [11, 30]. An example would be store separation, where the geometry splits into several parts that move relative to each other. For such a problem, as the distance between two bodies increases, the mesh obtained by mesh movement is likely to become too coarse in the direction of movement and the resulting stretched mesh will adversely effect the solution accuracy. However, the regions for which mesh movement is inappropriate are usually relatively small and this is utilized by applying local remeshing only. The regions that will be remeshed are determined by using a mesh quality indicator which is taken to be the ratio of the minimum dihedral angle of an element at the current time level and the minimum dihedral angle in the original mesh. Based on this indicator, elements are removed, creating one or more holes in the mesh. Each of these holes is meshed using the hybrid mesh generation scheme, with the triangulation of the hole surface providing the initial surface triangulation [11, 14] for each hole. The application, of the remeshing procedure has to consider the requirement of the boundary layers. Despite the fact that the scheme adopted for mesh movement prevents the requirement for the regeneration of the existing elements, in the case where the geometry contains components which are moving toward each other, some boundary layer elements will need to be removed. Also if the original configuration contains small gaps, which restrict the initial number of layers generated in the boundary region to reach the desired number of layers, the remeshing procedure should allow for further layers to be added if the gaps between the moving component increases due to the boundary movement. In certain circumstances, it may not be possible to recover the surface triangulation of the hole and, in this case, another layer of elements is removed from the original mesh and the process is repeated. The unknown field is transferred from the previous mesh level by linear interpolation.

5.3. Adaptation procedure

The complete adaptation procedure may be described in terms of the following steps:

- Starting from the initial grid, determine the layer on which each point in the boundary layer lies, $NLAY$.
- Calculate the minimum dihedral angle for each element in the isotropic region.
- Loop over the physical time steps:
    - Update the coordinates of the moving nodes and the nodes in the first 50% of the boundary layer region.
    - Update the coordinates of the mesh control function which are assigned to the moving components.
    - Apply the deforming mesh algorithm with the edges in the boundary layers having large edge coefficients to ensure a near rigid movement of the elements in that region.
    - Compute the new minimum dihedral angle of all elements.
    - If any dihedral angle is negative or the number of elements with small dihedral angle is greater that 5% of the total number of elements in the isotropic region then
        - — Mark these as elements to be removed.
        - — Group the adjacent marked elements into regions. This is achieved by recursive colouring starting from any marked element.
          For each region:
            - Identify the boundary faces of the region.
            - Identify the boundary faces which are on the interface between isotropic elements and the boundary layer elements.
            - If any interface face has intersected any of the boundary layer elements, then the element containing the face is added to the list of removed elements and the boundary face list is updated. This will guarantee that no intersecting elements

will remain in the case of a narrowing gap due to the movement of the boundary components.
If the layer number of the node of an interface face is less than the desired number of layers, further layers are grown from the face and the boundary face list is updated.
Using the final boundary face list, the remainder of the hole is remeshed using the isotropic Delaunay triangulation.
For any added boundary layers, merge tetrahedral elements to form hexahedral, prism and pyramid elements.
Interpolate the solution for all the newly generated nodes.
Renumber the newly generated nodes and elements, add the mesh to the global mesh and compute the minimum value of the dihedral angle for each newly generated isotropic element.

6. EXAMPLES

Selected computational examples of increasing complexity are presented, illustrating the various aspects of the approach.

6.1. Oscillating NACA64A010

Turbulent transonic flow around a NACA64A010 aerofoil is considered. The Mach number is 0.796 and the Reynolds number is 1.3×10^7. The initial angle of attack is $0°$ and the aerofoil is prescribed a sinusoidal movement with amplitude of $1.01°$ and Strouhal number of 15.567. As there is no geometric deformation or relative motion of components, the geometry time dependency is achieved by rotating the mesh in a rigid fashion. Experimental results are available for this testcase from Reference [31]. The calculation was performed by stacking a two-dimensional hybrid mesh five times, creating hexahedra and prisms. The resulting mesh consists of 173 720 nodes, 54 784 hexahedra and 166 920 prisms. The farfield boundary is placed a distance of about 15 chord lengths from the aerofoil surface. A layer of the stacked mesh is shown in Figure 6. The initial spacing in the boundary layer is 0.5×10^{-5} chord lengths. A source is placed in the region of the shock at steady-state and zero angle of attack. A fixed number of 300 multigrid cycles was performed for every physical time step, which is more than sufficient to ensure convergence for the lift and drag values. For this example, 32 physical time steps per cycle were employed. The estimated speedup using multigrid accelerated time stepping compared to explicit time stepping is about 175 for this case, assuming that the explicit scheme only needs to perform 32 mesh movement and preprocessing stages. This large number is a consequence of the small grid spacing in the boundary layers, severely slowing down explicit time stepping schemes. Figure 7 shows a comparison between the computed lift polar, i.e. lift coefficient vs phase angle, and that of the experiments. Good correlation can be observed. Also included are plots of the pressure coefficients on the aerofoil surface at different phase angles, Figure 8. Again, it can be observed that there is good correspondence between the computation and the experimental results. The calculation was performed using 16 R14000 CPUs for the solver stage and one processor for the mesh movement and preprocessor stages. One movement cycle required about 5 h of wall-clock time.

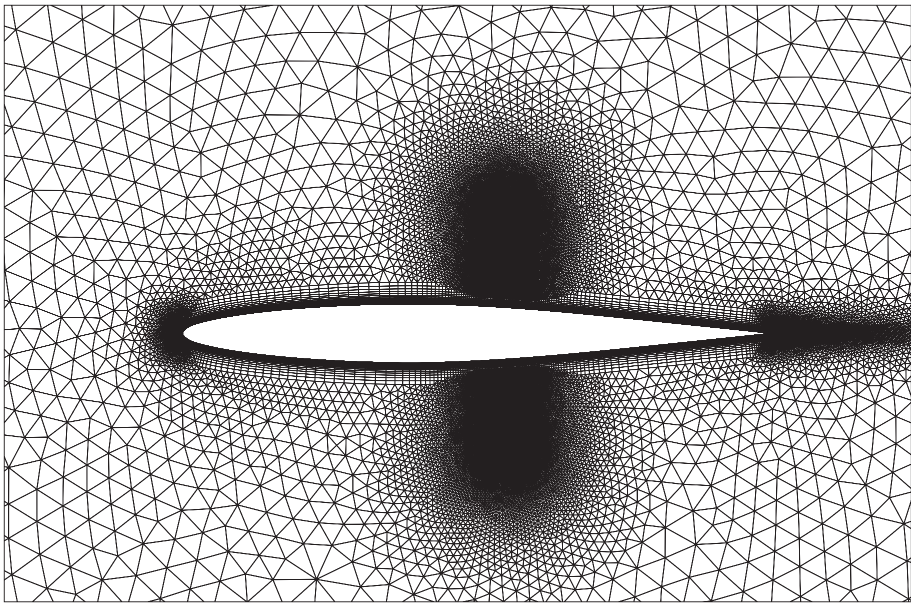

Figure 6. Mesh used in the transient turbulent NACA64A010 testcase.

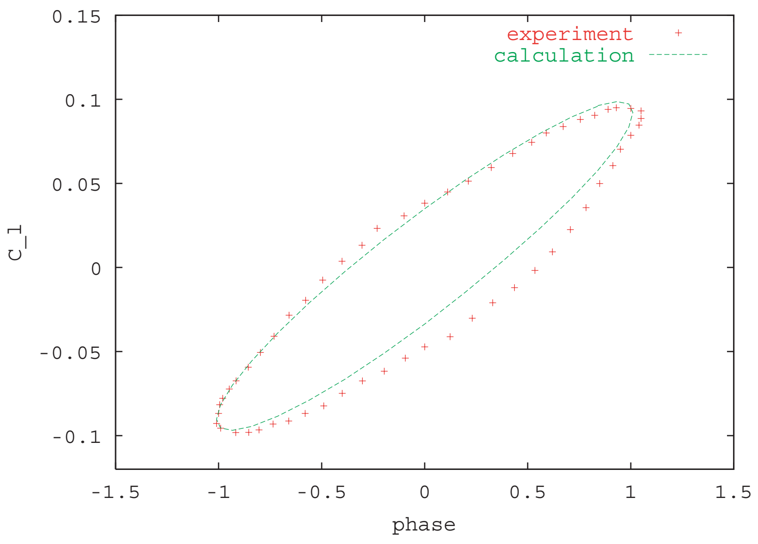

Figure 7. Lift polar for the transient turbulent NACA64A010 testcase.

6.2. *Generic fighter store release*

The following example considers a store release problem where two containers are dropped from an F16 fighter configuration. As the Euler equations are employed, the mesh consists of tetrahedra

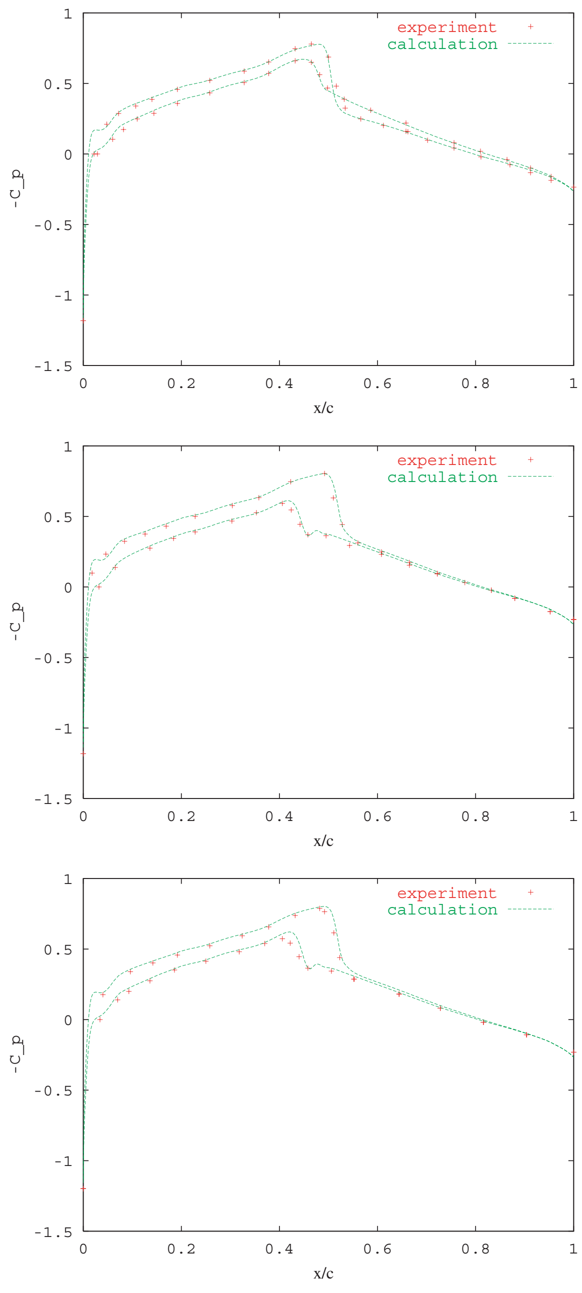

Figure 8. Comparison between computed and experimental surface pressure distributions for phase angles of 0, $\pi/2$ and $3\pi/2$, respectively, for the transient turbulent NACA64A010 calculation.

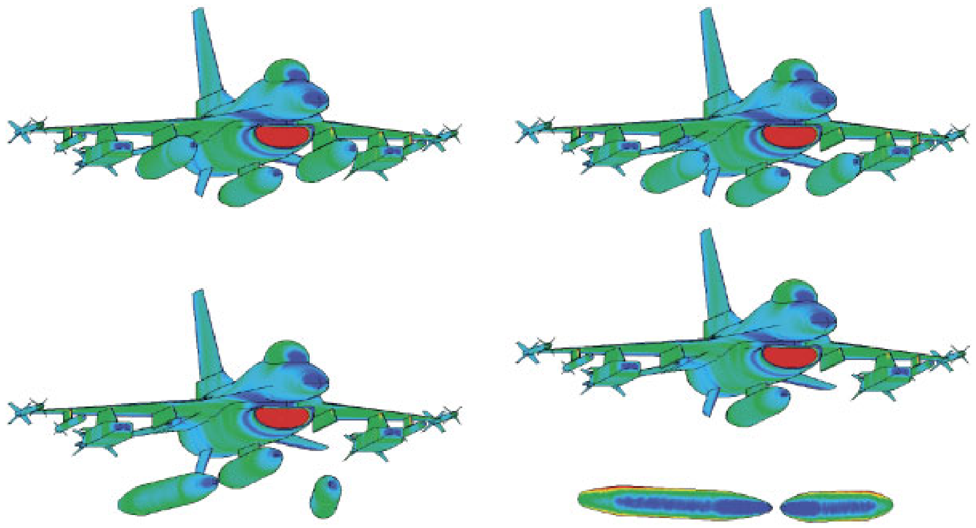
Figure 9. Snapshots of the generic F16 store release calculation. The surface pressure distributions at 0.29 s intervals are shown.

only and no boundary layers. The stores are considered completely filled with a substance of density 1250 times that of the freestream air density. The freestream Mach number is 0.5, the angle of attack is zero and the Froude number is 1333. The rigid body movement of the tanks is calculated by integrating the pressure field on the stores and using second-order accurate time integration. Local remeshing is employed and each mesh consists of about 2.7 million tetrahedral elements. The farfield boundary is placed about 10 aircraft lengths from the aircraft surface. Fifteen timesteps were performed, each employing 50 multigrid cycles. The entire calculation took about 8 h of wall-clock time, using 16 R14000 CPUs on the solver module and one R14000 on the preprocessing, mesh movement and local remeshing modules. Four plots of the solution at regular intervals are shown in Figure 9.

6.3. Separation of a shuttle and booster configuration

The numerical procedure which has been outlined above has been used to simulate the transient separation of a generic shuttle vehicle and a rocket booster. A turbulent steady-state simulation is performed, at a free stream Mach number of 0.85 and 0° angle of attack and the Reynolds number is 1.0×10^7 using the Spalart–Allmaras one equation model. The mesh employed for this portion of the simulation consists of 2.9 million elements. A view of the surface triangulation on the symmetry plane and the pressure contours on the surface of the shuttle and booster configuration are shown in Figure 10. The two bodies then begin to separate according to a prescribed translation, in the symmetry plane, together with a prescribed rotation of the shuttle about a point on its tail. The computation proceeds until the shuttle rotates through 15° and this required 20 local remeshing steps with 300 multigrid cycles per step. To demonstrate the requirement for local remeshing, a simulation was carried out using mesh movement only. Figure 11 shows a cut through the mesh and detail of the symmetry plane near the nose of the shuttle and booster configuration at the

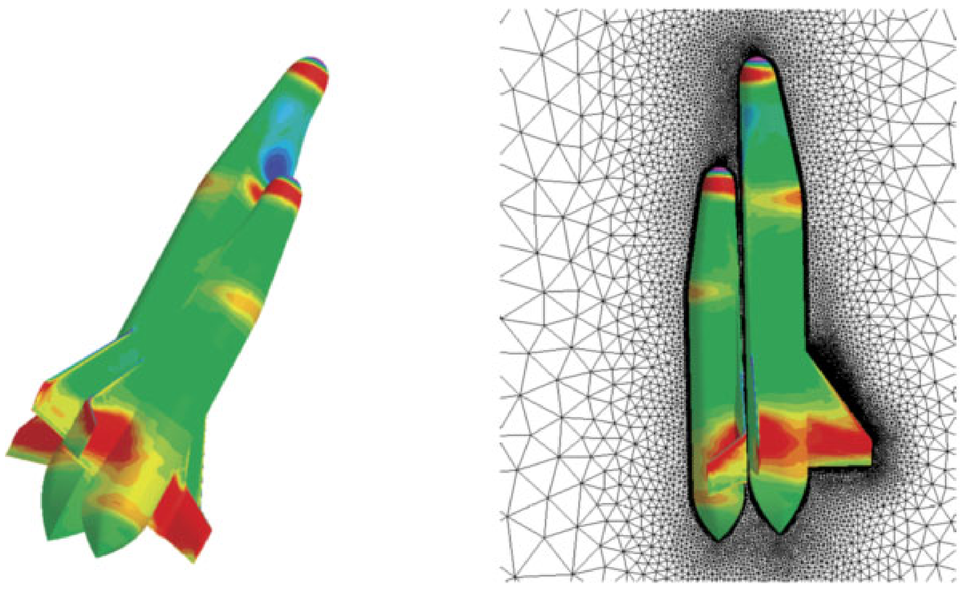

Figure 10. Separation of a shuttle and booster configuration: initial mesh and solution.

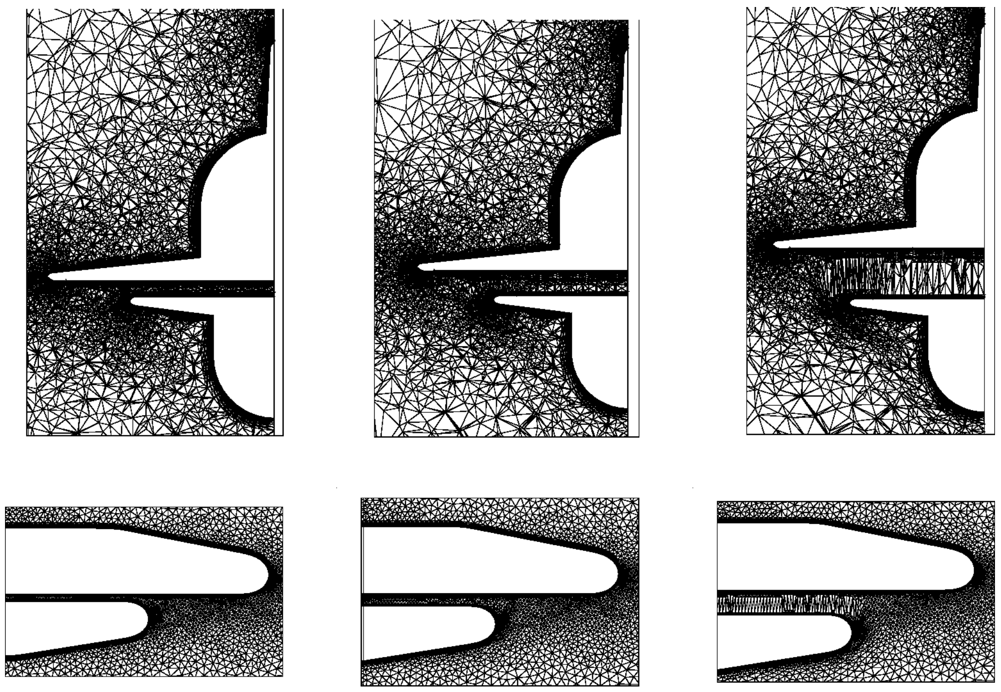

Figure 11. Separation of a shuttle and booster configuration: details of the meshes obtained without the use of local remeshing.

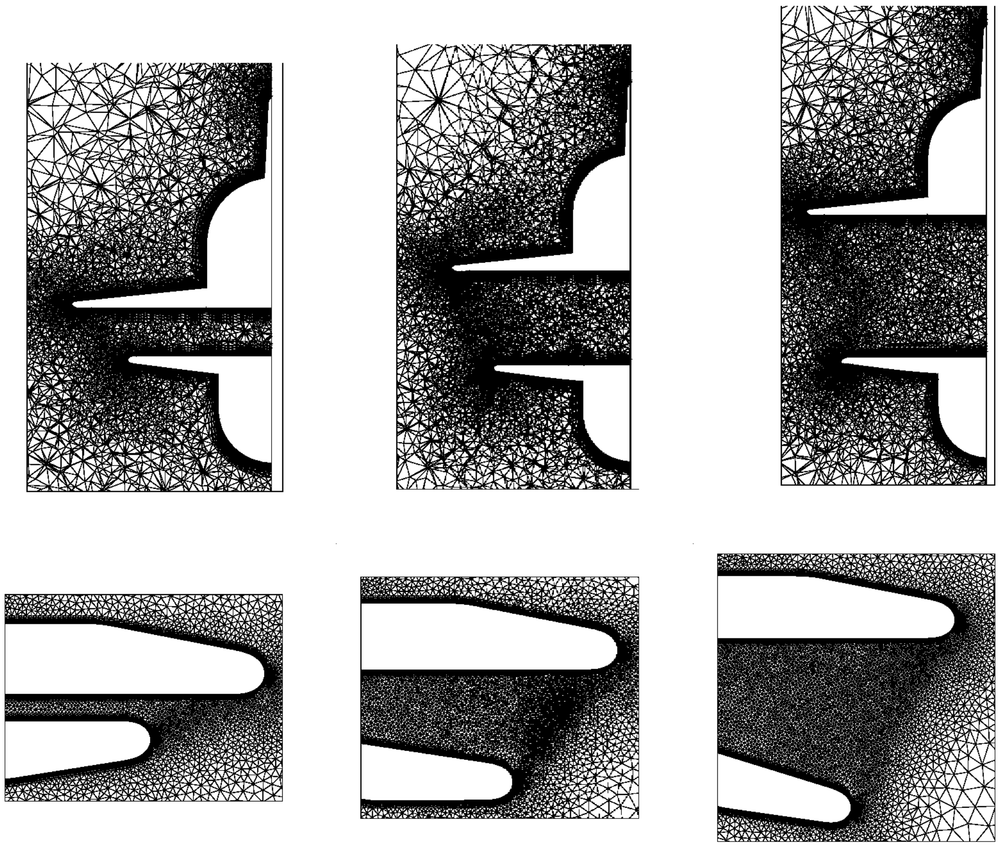
Figure 12. Separation of a shuttle and booster configuration: details of the meshes obtained using the local remeshing procedure.

start of this simulation and after the first few mesh movements. The solution failed to advance any further due to the self-intersection of the volume mesh. Details of the meshes resulted from employing both mesh movement and local remeshing are shown in Figure 12. High-quality meshes with the appropriate number of layers to resolve the boundary layer region after the widening of the gap between the shuttle and the booster are observed. Views of the surface triangulation on the symmetry plane and the pressure contours on the surface of the shuttle and booster configuration at various stages of the simulation are shown in Figure 13. The final mesh at this stage consists of 3.3 million elements. The simulation required 36 h wall-clock time using 16 SGI R14000 processors for the solver and one processor for the mesh pre-processing and adaptation.

7. CONCLUSIONS

An unstructured tetrahedral mesh-based procedure for the simulation of unsteady viscous compressible flows, with moving boundary components, has been described. Parallelization of the

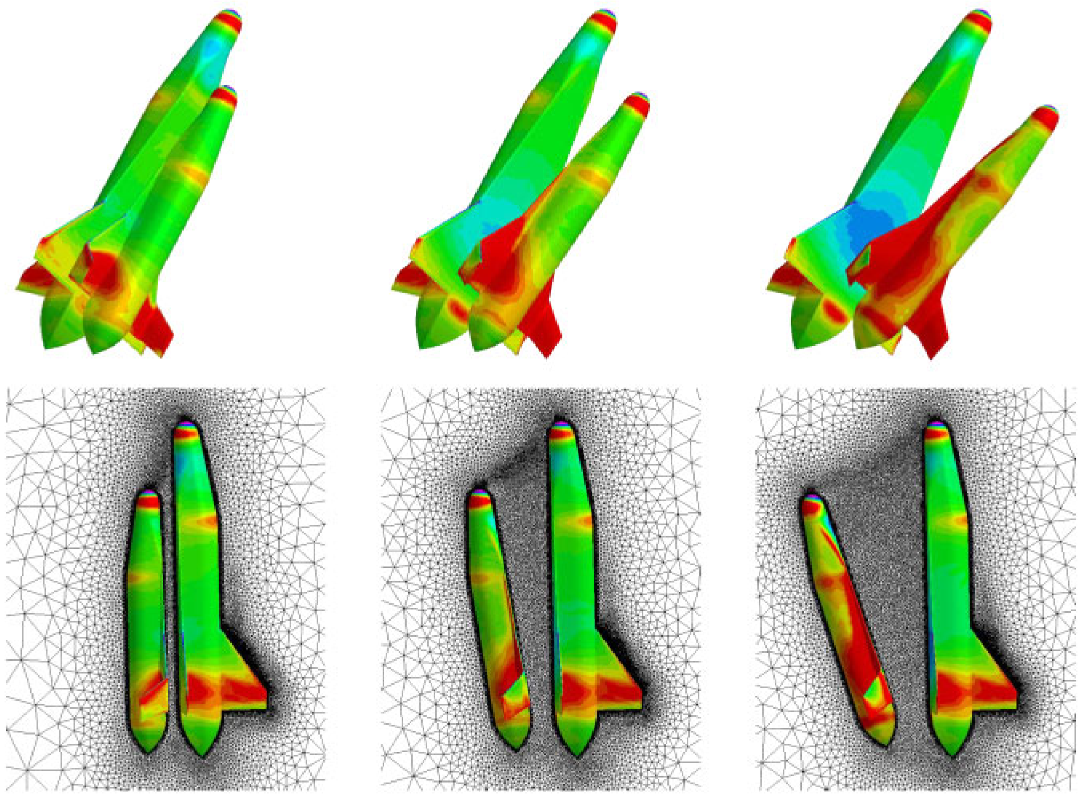

Figure 13. Separation of a shuttle and booster configuration showing meshes on the symmetry plane and the pressure contours on the surface of the configuration at various time intervals.

procedure leads to an efficient approach for flows in which the boundary displacements are small, which can be modelled by a moving mesh system with fixed connectivity. For more general flows, which require local remeshing at various stages to allow for the boundary movement, an efficient parallel implementation of the regeneration of the holes will require further research.

REFERENCES

1. Steger JL, Dougherty FC, Benek JA. *Advances in Grid Generation*, vol. 5. American Society of Mechanical Engineers: Fairfield, NJ, 1983.
2. Rogers SE, Suhs NE, Dietz WE. PEGASUS5: an automated preprocessor for overset-grid computational fluid dynamics. *AIAA Journal* 2003; **41**(6):1037–1045.
3. Nakahashi K, Togashi F, Sharov D. An intergrid-boundary definition method for overset unstructured grid approach. *AIAA Journal* 2000; **38**(11):2077–2084.
4. Johnson AA, Tezduyar TE. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering* 1994; **119**:73–94.
5. Farhat C, Degand C, Koobus B, Lesoinne M. Torsional springs for two dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**:231–245.
6. Nkonga B, Guillard H. Godunov type method on non-structured meshes for three-dimensional moving boundary problems. *Computer Methods in Applied Mechanics and Engineering* 1994; **113**:183–204.

7. Degand C, Farhat C. A three dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures* 2002; **80**:305–316.
8. Koobus B, Farhat C. Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes. *Computer Methods in Applied Mechanics and Engineering* 1999; **170**:103–129.
9. Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. *Journal of Computational Physics* 1993; **127**:380–397.
10. Tezduyar TE, Behr M, Mittal S, Johnson AA. Computation of unsteady incompressible flows with the stabilized finite element methods: space–time formulations, iterative strategies and massively parallel implementations. *New Methods in Transient Analysis*, *AMD*, vol. 143. ASME: New York, 1992.
11. Hassan O, Probert EJ, Morgan K, Weatherill NP. Unsteady flow simulation using unstructured meshes. *Computer Methods in Applied Mechanics and Engineering* 2000; **189**:1247–1275.
12. Löhner R, Yang C, Baum JD, Luo H, Pelessone D, Charman C. The numerical simulation of strongly unsteady flows with hundreds of moving bodies. *International Journal for Numerical Methods in Fluids* 1999; **31**:113–120.
13. Johnson AA, Tezduyar TE. Advanced mesh generation and update methods for 3D flow simulations. *Computational Mechanics* 1999; **23**:130–143.
14. Tremel U. Parallel unstructured adaptive remeshing for moving boundary problems. *Ph.D. Thesis*, University of Wales, Swansea, 2005.
15. Spalart PR, Allmaras SR. A multigrid accelerated hybrid unstructured mesh method for 3D compressible turbulent flow. *AIAA Paper 92-0439*, 1992.
16. Peiró J, Peraire J, Morgan K. FELISA system reference manual. Part 1: Basic theory. *Swansea Report C/R/821/94*, University of Wales, 1994.
17. Hassan O, Morgan K, Probert EJ, Peraire J. Unstructured tetrahedral mesh generation for three-dimensional viscous flows. *International Journal for Numerical Methods in Engineering* 1996; **39**:549–567.
18. Weatherill NP, Hassan O. Efficient three-dimensional Delaunay triangulation with automatic boundary point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering* 1994; **37**:2005–2039.
19. Crumpton PI, Moinier P, Giles MB. An unstructured algorithm for high Reynolds number flows on highly stretched grids. In *Numerical Methods in Laminar and Turbulent Flow*, Taylor C, Cross JT (eds). Pineridge Press: Swansea, 1997; 561–572.
20. Donéa J. *Arbitrary Lagrangian Eulerian Methods*, *Computer Methods for Transient Analysis*. Elsevier: North-Holland, 1983.
21. Thomas PD, Lombard CK. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal* 1979; **17**:1030–1037.
22. Lesoinne M, Farhat C. Geometric conservation laws for problems involving moving boundaries and deforming meshes, and their impact on aeroelastic computations *Computer Methods in Applied Mechanics and Engineering* 1996; **134**:71–90.
23. Jameson A, Schmidt W, Turkel E. Numerical simulation of the Euler equations by finite volume methods using Runge–Kutta timestepping schemes. *AIAA Paper 81-1259*, 1981.
24. Giles MB. Energy stability analysis of multi-step methods on unstructured meshes. *Massachusetts Institute of Technology Report CFDL-TR-87-1*, 1987.
25. Brandt A. Multi-level adaptive solutions to boundary value problems. *Mathematics of Computation* 1977; **21**:333–390.
26. Steve H, Lallemand MH, Dervieux A. Unstructured multigridding by volume agglomeration: current status. *Computers and Fluids* 1992; **21**:397–433.
27. Mavriplis DJ, Venkatakrishnan V. A 3D agglomeration multigrid solver for the Reynolds-averaged Navier–Stokes equations on unstructured meshes. *International Journal for Numerical Methods in Fluids* 1996; **23**:527–544.
28. Sørensen KA. A multigrid accelerated procedure for the solution of compressible fluid flows on unstructured hybrid meshes. *Ph.D. Thesis*, University of Wales, Swansea, 2002.
29. Stein K, Tezduyar TE, Benney R. Automatic mesh update with the solid-extension mesh moving technique. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:2019–2032.
30. Baum JD, Luo H, Löhner R. A new ALE adaptive unstructured methodology for the simulation of moving bodies. *AIAA Paper 94-0414*, 1994.
31. Davis SS. NACA64A010 oscillatory pitching. *AGARD Report R-702*, 1982.