INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2003; **43**:517–536 (DOI: 10.1002/fld.590)

A multigrid accelerated time-accurate inviscid compressible fluid flow solution algorithm employing mesh movement and local remeshing

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

Civil & Computational Engineering Centre, University of Wales Swansea, Swansea SA2 8PP, Wales, U.K.

SUMMARY

An implicit scheme for the time accurate solution of three dimensional compressible fluid flow problems with moving boundaries on unstructured tetrahedral meshes is described. The numerical scheme is nominally second order accurate in both space and time and satisfies a geometric conservation law. For improved computational performance, the implicit equation system is solved by explicit iteration with multigrid acceleration. For the multigrid implementation, the coarse meshes are automatically generated by an agglomeration technique. The change in the solution domain geometry with time is handled by moving the mesh using a spring analogy scheme, with local remeshing performed in regions of reduced mesh quality. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: 3D inviscid flow; time accurate; agglomerated multigrid; unstructured mesh; mesh movement; local remeshing

1. INTRODUCTION

There are many examples of practical problems that require the modelling of compressible time dependent flow with moving boundaries. In the aerospace industry, such applications include the analysis of flutter, store release, buffeting and the deployment of control surfaces. In this paper, an unstructured tetrahedral mesh method is proposed for the solution of this class of problems. This provides a convenient framework for handling the complex geometrical shapes that are frequently encountered, with the spatial discretization of the governing equations accomplished by employing a cell vertex finite volume method.

The resulting equations have often been advanced in time by employing explicit schemes, but this usually requires many time steps and long computational times. Here, the requirement

*Correspondence to: O. Hassan, Civil and Computational Engineering Centre, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, U.K.
† Now at: Aerodynamics—MT632, EADS, Military Aircraft, München, Germany.
‡ E-mail: o.hassan@swansea.ac.uk

Contract/grant sponsor: The Research Council of Norway; contract/grant number: 125676/410

Received 3 March 2002
Accepted 15 January 2003

for a large number of time steps is not normally a result of the demands for time accuracy, but is due to the stringent stability constraints associated with explicit schemes. These constraints arise because small elements are usually unavoidable if the geometry is to be accurately represented and these small elements dictate the time step size for the entire problem. By splitting the computational domain into several sub-domains, according to the size of the allowable local time step, more efficient explicit time accurate schemes [1–3] have been developed. Although these approaches work well for small values of the Strouhal number, they are still restricted by the stability limit of the scheme for slower movement modes.

In this paper, the alternative approach of employing an implicit formulation is adopted. The size of the allowable physical time step is then independent of the spatial discretization which means that the time step used can be selected by considering accuracy criteria only. With such formulations it is imperative, for realistic three dimensional simulations, to use an implementation that avoids the requirement for working with large matrices and, in this context, equation solution by explicit iteration with multigrid acceleration is attractive. For an unstructured tetrahedral mesh, research has already demonstrated how the coarse meshes required for multigrid can be automatically generated [4–6]. The multigrid scheme described in this paper achieves this by using an agglomeration approach [4] in which the control volumes of the meshes are merged together through an edge based method. In this way, good quality nested meshes are automatically created at only a fraction of the time and effort required for manual coarse mesh generation.

There are a number of different approaches that can be employed for handling the meshing issues that arise in the simulation of problems involving moving geometries. If the use of overlapping meshes of the Chimera type [7, 8] is not considered, small changes in the geometry can be approximated by keeping the mesh fixed and perturbing the boundary conditions [9]. This approach provides acceptable results for aerodynamic simulations involving small boundary displacements but, for larger displacements, mesh movement or remeshing techniques must be used. Mesh movement [10–12] adapts the mesh to the deformations imposed by the moving boundary, normally maintaining the mesh connectivity so that issues such as geometric conservation are simplified. An alternative is to apply mesh quality enhancement techniques which modify the mesh structure [13]. The drawback with mesh movement approaches is that meshes of bad quality may result if large movement is involved and, in such cases, the use of remeshing becomes necessary. The remeshing approach [14] is expensive, as it requires the involvement of a mesh generation procedure at every time step and introduces interpolation errors between time steps. The alternative is the method adopted here, which involves a combination of mesh movement and local remeshing [3, 15]. The mesh is first moved, retaining the connectivity, as dictated by the geometry deflection. A mesh quality indicator is applied to decide if there are regions of bad mesh quality present. Such regions are removed, creating holes which are then remeshed using an unstructured mesh generator. The solution of 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.

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 DESCRIPTION

The time dependent compressible Euler equations can be expressed in integral form, on a three dimensional Cartesian domain $\Omega(t) \subset \mathbb{R}^3$ with closed surface $\partial\Omega(t)$, as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)} \mathbf{U}\,\mathrm{d}\mathbf{x} + \int_{\partial\Omega(t)} (\mathbf{F}^j - v^j\mathbf{U})n_j\,\mathrm{d}\mathbf{x} = 0 \quad j = 1,2,3 \tag{1}$$

where the summation convention is employed,

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

and $\mathbf{v} = (v_1, v_2, v_3)$ is the velocity of the surface $\partial\Omega(t)$. Here ϱ denotes the fluid density, u_j the jth component of the fluid velocity vector, p the pressure, ε the specific total energy and n_j the jth component of the outward unit normal vector to $\partial\Omega$. The system is closed by the addition of the calorically perfect gas equations of state

$$p = \varrho RT, \quad \varepsilon = c_v T + \tfrac{1}{2}u_j u_j \tag{3}$$

where R is the real gas constant, T is the total temperature, c_v is the specific heat at constant volume, $\gamma = c_p/c_v = 1.4$ and c_p is the specific heat at constant pressure. At any solid boundary, the relative normal component of the velocity must vanish and this requirement can be expressed in the form

$$u_j n_j^{\mathrm{w}} = u_j^{\mathrm{w}} n_j^{\mathrm{w}} \tag{4}$$

where u_j^{w} and n_j^{w} are the components of the wall velocity and the wall unit normal vector, respectively.

The characteristic parameters for compressible time dependent inviscid flow are the Mach number and the Strouhal number, defined by the expressions

$$M_\infty = \frac{u_\infty}{c_\infty}, \quad St_\infty = \frac{u_\infty t_c}{\ell} \tag{5}$$

In these equations, u_∞ and c_∞ denote the magnitude of the free stream velocity and speed of sound respectively, t_c is a characteristic time scale of the motion and ℓ is a characteristic length scale. For cyclic motion of wings, t_c is taken as the period of one cycle and the mean geometric chord is used as the characteristic length scale. For problems in which gravitational effects are important, an additional characteristic parameter is the Froude number

$$Fr_\infty = \frac{u_\infty^2}{\ell g} \tag{6}$$

where g is the gravitational acceleration.

3. DISCRETIZATION PROCEDURE

The computational domain is represented by an unstructured tetrahedral mesh, generated by a Delaunay method [16]. 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. 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, to strive to retain the mesh connectivity.

3.1. Mesh movement

An obvious method to account for boundary movement, while retaining the mesh connectivity, is to use mesh movement. 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 [3]. 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

$$\sum_{J \in \Lambda_I} k_{IJ}(d_{IJ} - d_{IJ}^{\text{prev}}) = 0 \tag{7}$$

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. 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. 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.

3.2. Local remeshing

For certain simulations, it is impossible to avoid remeshing if the necessary mesh quality is to be maintained. 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 utilised 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 volume of an element at the current time level and the element volume 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 in the normal manner by using a Delaunay scheme, with the triangulation of the hole surface providing the initial surface triangulation [3] for each hole.

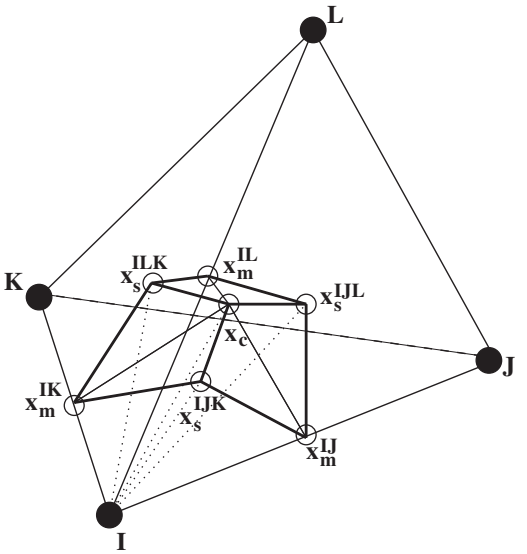

Figure 1. Illustration of that portion of the dual mesh surrounding node I that is contained inside a tetrahedral element.

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.

3.3. The finite volume method

The governing equations are discretized by using a finite volume method, in which the unknowns are located at the vertices of the mesh and the numerical integration is performed over dual mesh interfaces. The dual mesh is constructed by connecting edge midpoints, element centroids and face centroids in such a way that only one node is contained within each control volume. Each edge of the grid is associated with a surface segment of the dual mesh interface between the nodes connected to the edge. This surface is defined using triangular facets, where each facet is connected to the midpoint of the edge, a neighbouring element centroid and the centroid of an element face connected to the edge [17]. This is illustrated, for a tetrahedral element, in Figure 1. In this figure, the midpoint of the edge connecting nodes I and J is denoted by $\mathbf{x}_m^{IJ}$, the centroid of the face with vertices I,J and K is $\mathbf{x}_s^{IJK}$ and the element centroid is designated by $\mathbf{x}_c$. The bold lines on the dual mesh illustrate the boundaries between the edges with which the dual mesh segment is associated and, with this dual mesh definition, the control volume can be thought of as being made up of a set of tetrahedra. The dual mesh interface inside the computational domain surrounding node I is denoted by Γ_I, while the portion of the dual that is situated on the computational boundary is denoted by Γ_I^B. The facets which define the control volume interface surrounding node I are denoted by Γ_I^K, so that $\Gamma_I \cup \Gamma_I^B = \bigcup_K \Gamma_I^K$. The subset of Γ_I associated with the edge connecting nodes I

and J, i.e. the facets touching the edge, is denoted by Γ_{IJ}. The subset of Γ_I^B between nodes I and J is denoted by Γ_{IJ}^B.

To enable the numerical integration of the term involving the inviscid flux vectors $\mathbf{F}^j$ in Equation (1) over the control volume interfaces, a set of coefficients is calculated for each edge using the dual mesh segment associated with the edge. For an internal edge, these coefficients are given by

$$C_{IJ}^j = \sum_{K \in \Gamma_{IJ}} A_{\Gamma_I^K} n_{\Gamma_I^K}^j \tag{8}$$

where $A_{\Gamma_I^K}$ is the area of facet Γ_I^K and $n_{\Gamma_I^K}^j$ is the outward normal to the facet from the viewpoint of node I. In a similar manner, to evaluate the contributions from the facets forming the computational boundary, the coefficients

$$D_{IJ}^j = \sum_{K \in \Gamma_{IJ}^B} A_{\Gamma_I^K} n_{\Gamma_I^K}^j \tag{9}$$

are evaluated. In this case, $n_{\Gamma_I^K}^j$ denotes the normal to the facet in the direction out of the computational domain. When performing the integration, the contribution from the dual mesh segment associated with an edge, is determined by assuming the integrand to be constant over the segment and equal to its approximated value at the midpoint of the edge, i.e. a form of midpoint quadrature is employed. For a general node I, this integral is then evaluated as

$$\int_{\partial\Omega_I} \mathbf{F}^j n_j \, \mathrm{d}\mathbf{x} \approx \sum_{J \in \Lambda_I} \frac{C_{IJ}^j}{2} (\mathbf{F}_I^j + \mathbf{F}_J^j) + \sum_{J \in \Lambda_I^B} D_{IJ}^j \mathbf{F}_I^j \tag{10}$$

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 second term on the right hand side thus only contributes if I is a boundary node. Boundary terms are treated, in the classical finite volume manner, by using a local midpoint rule.

It is proposed to treat the integral in Equation (1) involving the velocity of the control volume interface in a similar fashion. In this case, additional numerical edge coefficients S_{IJ}, T_{IJ}, are introduced such that

$$\int_{\partial\Omega_I} v_j \mathbf{U} n_j \, \mathrm{d}\mathbf{x} \approx \sum_{J \in \Lambda_I} \frac{S_{IJ}}{2} (\mathbf{U}_I + \mathbf{U}_J) + \sum_{J \in \Lambda_I^B} T_{IJ} \mathbf{U}_I \tag{11}$$

The full details of the manner in which these additional edge coefficients are evaluated in this work will be described shortly.

3.4. Time discretization

At a general node I, the time derivative term in Equation (1) may be evaluated as

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\Omega_I(t_n)} \mathbf{U} \, \mathrm{d}\mathbf{x} \bigg|_{t=t_n} \approx \frac{1}{\Delta t} (V_I^n \mathbf{U}_I^n - V_I^{n-1} \mathbf{U}_I^{n-1}) \tag{12}$$

if a first order backward Euler approximation is adopted or as

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_I(t_n)}\mathbf{U}\,\mathrm{d}\mathbf{x}\bigg|_{t=t_n}\approx\frac{1}{\Delta t}\left(\frac{3}{2}V_I^n\mathbf{U}_I^n-2V_I^{n-1}\mathbf{U}_I^{n-1}+\frac{1}{2}V_I^{n-2}\mathbf{U}_I^{n-2}\right)\tag{13}$$

when a three level second order approximation is employed. In these expressions, the superscript n refers to an evaluation at time level $t=t_n$, the time levels are taken to be equally spaced with a time step Δt and V_I is the volume of Ω_I.

3.5. Geometric conservation

An important property of Equation (1) is that of geometric conservation, which implies that, if the unknown field is constant, the solution should not change in time. Mathematically, this implies that

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega(t)}\mathrm{d}\mathbf{x}-\int_{\partial\Omega}v_jn_j\,\mathrm{d}\mathbf{x}=0\tag{14}$$

and it is desirable that the discretized numerical scheme should also exhibit this property [11, 18, 19]. The task of determining the edge coefficients in Equation (11) so that the numerical scheme is geometrically conservative is not trivial and has been treated by several authors [11, 12, 18, 19]. Here the approach of Nkonga and Guillard [12] is adopted and extended to dual meshes which are constructed of assemblies of triangular facets. The key point is the assumption that each node in the mesh moves in a linear fashion between time levels t_n and t_{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 2. The points $\mathbf{x}_{c1}$, $\mathbf{x}_{c2}$ and $\mathbf{x}_{c3}$ are situated at the vertices of the facet and move linearly in space and time, from time level t_n to time level t_{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 is apparent that triangular facets of this form will remain planar under this assumption of linear node movement. The volume swept out by the triangular facet Γ_I^K, between time levels t_n and t_{n+1}, is illustrated in Figure 2 and may be expressed as

$$\delta V_{\Gamma_I^K}^{n+1,n}=\int_{t_n}^{t_{n+1}}\int_{\Gamma_I^K}v_j^{\Gamma_I^K}n_j^{\Gamma_I^K}\,\mathrm{d}\mathbf{x}\,\mathrm{d}t\tag{15}$$

It can be shown [12, 19] that

$$\delta V_{\Gamma_I^K}^{n+1,n}=\tfrac{1}{9}(A_{\Gamma_I^K}^{n+1}n_j^{\Gamma_I^K,n+1}+A_{\Gamma_I^K}^{n}n_j^{\Gamma_I^K,n}+A_{\Gamma_I^K}^{*}n_j^{\Gamma_I^K,*})(r_j^{c1}+r_j^{c2}+r_j^{c3})\tag{16}$$

where

$$A_{\Gamma_I^K}^{n+1}\mathbf{n}^{\Gamma_I^K,n+1}=\tfrac{1}{2}(\mathbf{x}_{c2}^{n+1}-\mathbf{x}_{c1}^{n+1})\times(\mathbf{x}_{c3}^{n+1}-\mathbf{x}_{c1}^{n+1})\tag{17}$$

$$A_{\Gamma_I^K}^{n}\mathbf{n}^{\Gamma_I^K,n}=\tfrac{1}{2}(\mathbf{x}_{c2}^{n}-\mathbf{x}_{c1}^{n})\times(\mathbf{x}_{c3}^{n}-\mathbf{x}_{c1}^{n})\tag{18}$$

$$A_{\Gamma_I^K}^{*}\mathbf{n}^{\Gamma_I^K,*}=\tfrac{1}{4}(\mathbf{x}_{c2}^{n+1}-\mathbf{x}_{c1}^{n+1})\times(\mathbf{x}_{c3}^{n}-\mathbf{x}_{c1}^{n})+\tfrac{1}{4}(\mathbf{x}_{c2}^{n}-\mathbf{x}_{c1}^{n})\times(\mathbf{x}_{c3}^{n+1}-\mathbf{x}_{c1}^{n+1})\tag{19}$$

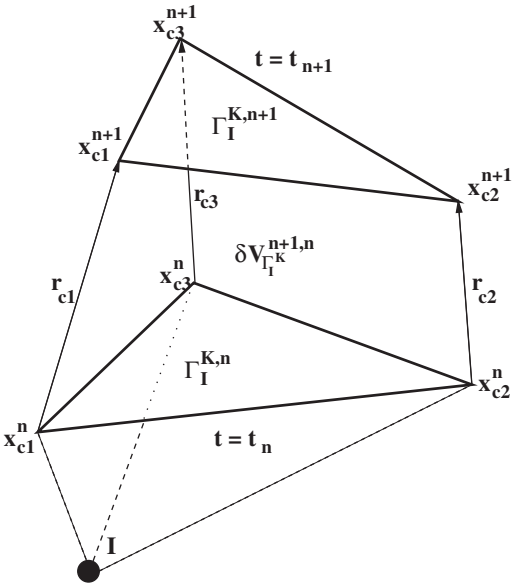

Figure 2. Illustration of the terminology used for the movement of a triangular facet of the dual mesh for node I.

and

$$\mathbf{r}_{c1} = \mathbf{x}_{c1}^{n+1} - \mathbf{x}_{c1}^{n}, \quad \mathbf{r}_{c2} = \mathbf{x}_{c2}^{n+1} - \mathbf{x}_{c2}^{n}, \quad \mathbf{r}_{c3} = \mathbf{x}_{c3}^{n+1} - \mathbf{x}_{c3}^{n} \tag{20}$$

The local geometric conservation law

$$\delta V_{\Gamma_I^K}^{n+1,n} = \Delta t v_{\Gamma_I^K}^{j,*} n_{\Gamma_I^K}^{j,*} A_{\Gamma_I^K}^{*} \tag{21}$$

can now be applied to the facet, where $v_{\Gamma_I^K}^{j,*}$ is an averaged facet velocity, $A_{\Gamma_I^K}^{*}$ is the averaged facet area and $n_{\Gamma_I^K}^{j,*}$ is the average normal between time levels t_n and t_{n+1}. The right hand side of this equation may be recognized as the contribution of a single facet to the integral over $\partial\Omega$ in Equation (14), multiplied by the time step. Since the dual mesh is constructed by collections of triangular facets, it follows that the formulation of a geometrically conservative scheme for the unstructured meshes used here may be obtained by setting

$$\Delta t S_{IJ}^{n+1} = \sum_{K \in \Gamma_{IJ}} \delta V_{\Gamma_I^K}^{n+1,n}, \quad \Delta t T_{IJ}^{n+1} = \sum_{K \in \Gamma_{IJ}^B} \delta V_{\Gamma_I^K}^{n+1,n} \tag{22}$$

Geometric conservation results in this case as, summing over the edges connected to a given node I, produces

$$\Delta t \sum_{J \in \Gamma_I} S_{IJ}^{n+1} + \Delta t \sum_{J \in \Gamma_I} T_{IJ}^{n+1} = \sum_{J \in B_I} \left[\sum_{K \in \Gamma_{IJ}} \delta V_{\Gamma_I^K}^{n+1,n} + \sum_{K \in \Gamma_{IJ}^B} \delta V_{\Gamma_I^K}^{n+1,n} \right] = V_I^{n+1} - V_I^{n} \tag{23}$$

where the last equality is true since the control volume is closed. From this equation, it follows that the geometric conservation law is satisfied numerically when the backward Euler time

discretization of Equation (12) is used. When the alternative second order time discretization of Equation (13) is applied, a modified version of the definition of Equation (22) has to be adopted in order to ensure numerical geometric conservation. Here, the modification

$$\hat{S}_{IJ}^{n+1} = \tfrac{3}{2} S_{IJ}^{n+1} - \tfrac{1}{2} S_{IJ}^{n}, \quad \hat{T}_{IJ}^{n+1} = \tfrac{3}{2}\, T_{IJ}^{n+1} - \tfrac{1}{2}\, T_{IJ}^{n} \tag{24}$$

is used, which makes the coefficients second order in time [13]. Then, using Equation (23), it follows that

$$\frac{3}{2} V_I^{n+1} - 2V_I^n + \frac{1}{2}\, V_I^{n-1} = \sum_{J \in \Lambda_I} \hat{S}_{IJ}^n + \sum_{J \in \Lambda_I^B} \hat{T}_{IJ}^n \tag{25}$$

which is a geometric conservation law valid for the three level time discretization. It should be noted that these internal edge coefficients need only be stored once for each edge as they are anti-symmetric, i.e.

$$S_{IJ}^n = -S_{JI}^n \tag{26}$$

This follows since the volume swept out by each facet is the same for the two control volumes connected by an edge, but of opposite sign. There does not, however, appear to be any direct relationship between T_{IJ}^n and T_{JI}^n and both of these coefficients are calculated and stored.

The boundary condition of Equation (4) is applied by using the expression

$$\mathbf{u}_I^{w,n+1} = \frac{1}{\Delta t}(\mathbf{x}_I^{n+1} - \mathbf{x}_I^n) \tag{27}$$

for the wall velocity when the first order scheme is employed and

$$\mathbf{u}_I^{w,n+1} = \frac{1}{\Delta t}\left(\frac{3}{2}\,\mathbf{x}_I^{n+1} - 2\mathbf{x}_I^n + \frac{1}{2}\,\mathbf{x}_I^{n-1}\right) \tag{28}$$

when the second order scheme is adopted. This is consistent with the dual mesh velocity definitions employed in the two time discretization approaches. For regions that have been remeshed, the coordinates at the current time level are interpolated back to the previous time level, thus creating a new mesh at the previous time level with the connectivity of the current mesh. The mesh discretization is then performed in the same way as for the moved regions of the mesh. In the results presented here, the first order time discretization scheme is used if remeshing has been performed.

3.6. *Artificial dissipation*

As the resulting discretization procedure is essentially central difference in character, the addition of a form of stabilizing dissipation is required, if the scheme is to be practically useful. This is achieved by a method of JST type [20], in which the third order biharmonic term

$$\mathbf{H}_I \equiv \sum_{J \in \Gamma_I} \mathbf{H}_{IJ} = \sum_{J \in \Lambda_I} \mathscr{D}_{IJ}(\mathbf{G}_J - \mathbf{G}_I) \tag{29}$$

where

$$\mathbf{G}_I = \sum_{K \in \Lambda_I} (\mathbf{U}_K - \mathbf{U}_I) \tag{30}$$

is introduced. Here, $\mathbf{H}_{IJ}$ is the artificial dissipation flux and $\mathscr{D}_{IJ}$ is the biharmonic dissipation matrix which is $\mathcal{O}(h^2)$ where h is a typical mesh spacing. The diagonal form

$$\mathscr{D}_{IJ} = m_{IJ}\alpha_{IJ}\mathbf{I} \tag{31}$$

is adopted for the dissipation matrix [20], where $\mathbf{I}$ is the unit matrix,

$$m_{IJ} = \max(0, \varepsilon_4 - \varepsilon_2 N_{IJ}) \tag{32}$$

and

$$\alpha_{IJ} = \frac{1}{K_I + K_J} \min\left(\frac{V_I}{\Delta\tau_I}, \frac{V_J}{\Delta\tau_J}\right) \tag{33}$$

Here K_I is the number of edges connected to node I and $\Delta\tau_I$ is the value of the local time step. The biharmonic dissipation factor, ε_4, is a user-specified parameter and is normally selected to lie within the range $[0.1, 0.2]$. The biharmonic term is not capable of stabilizing the numerical scheme close to discontinuities in the flow and the additional harmonic term

$$\mathbf{Q}_I = \sum_{J\in\Lambda_I} \varepsilon_2 \alpha_{IJ} N_{IJ}(\mathbf{U}_I - \mathbf{U}_J) \tag{34}$$

is added, where ε_2 is the harmonic dissipation parameter. This term is of first order, but only makes a significant contribution in regions of high pressure gradients, due to the incorporation of the pressure switch

$$N_{IJ} = \max(|\partial p_I|, |\partial p_J|) \tag{35}$$

where

$$\partial p_I = 12 \sum_{K\in\Lambda_I} (p_K - p_I) / \sum_{K\in\Lambda_I} (p_K + p_I) \tag{36}$$

The harmonic dissipation parameter is usually assigned a value in the range $[0.2, 0.4]$.

3.7. Discrete equation

When backward Euler time stepping is used, the final form of the discrete equation for the time accurate simulation of inviscid flow is

$$\begin{aligned}
&\frac{1}{\Delta t}(V_I^n \mathbf{U}_I^n - V_I^{n-1}\mathbf{U}_I^{n-1}) + \sum_{J\in\Lambda_I} \frac{C_{IJ}^{j,n}}{2}(\mathbf{F}_I^{j,n} + \mathbf{F}_J^{j,n}) + \sum_{J\in\Lambda_I^B} D_{IJ}^{j,n}\mathbf{F}_I^{j,n} \\
&\quad - \sum_{J\in\Lambda_I} \frac{S_{IJ}^n}{2}(\mathbf{U}_I^n + \mathbf{U}_J^n) - \sum_{J\in\Lambda_I^B} T_{IJ}^n \mathbf{U}_I^n - \sum_{J\in\Lambda_I} \mathscr{D}_{IJ}^n(\mathbf{H}_I^n - \mathbf{H}_J^n) \\
&\quad - \sum_{J\in\Lambda_I} \alpha_{IJ} N_{IJ}^n(\mathbf{U}_I^n - \mathbf{U}_K^n) = 0
\end{aligned} \tag{37}$$

at every node I in the computational domain. If the second order time discretization is used, the first term of this equation is replaced by the right hand side of Equation (13) and S_{IJ}, T_{IJ} in the fourth and fifth terms are replaced by the expressions defined in Equation (24).

4. SOLUTION PROCEDURE

At every time step, this approach results in an equation system of the form

$$\mathbf{R}_f(\mathbf{U}_f) = \mathbf{s}_f \tag{38}$$

where $\mathbf{R}_f$ is the vector of nodal residuals and $\mathbf{s}_f$ is a source term that is independent of $\mathbf{U}_f$. In these expressions, the subscript f refers to a quantity evaluated on the generated mesh. For example, by treating the time derivative term in the governing equations as a source, the scheme of equation (37) may be expressed in this form with

$$\begin{aligned}\mathbf{R}_I(\mathbf{U}^n) = {} & \frac{V^n}{\Delta t}\mathbf{U}_I^n + \sum_{J\in\Lambda_I}\frac{C_{IJ}^{j,n}}{2}(\mathbf{F}_I^{j,n}+\mathbf{F}_J^{j,n}) + \sum_{J\in\Lambda_I^B} D_{IJ}^{j,n}\mathbf{F}_I^{j,n} \\ & - \sum_{J\in\Lambda_I}\frac{S_{IJ}^n}{2}(\mathbf{U}_I^n+\mathbf{U}_J^n) - \sum_{J\in\Lambda_I^B} T_{IJ}^n\,\mathbf{U}_I^n \\ & - \sum_{J\in\Lambda_I}\mathscr{D}_{IJ}^n(\mathbf{H}_I^n-\mathbf{H}_J^n) - \sum_{J\in\Lambda_I}\alpha_{IJ}N_{IJ}^n(\mathbf{U}_I^n-\mathbf{U}_J^n)\end{aligned} \tag{39}$$

where $\mathbf{U}_I$ is the Ith entry in $\mathbf{U}_f$, $\mathbf{R}_I$ is the Ith entry in $\mathbf{R}_f$ and

$$\mathbf{s}_I^n = \frac{V^{n-1}}{\Delta t}\mathbf{U}_I^{n-1} \tag{40}$$

is the Ith entry in $\mathbf{s}_f$. Although the solution of this non-linear system may be accomplished in several ways, we will propose here the use of the FAS multigrid scheme [21], as multigrid has proved to be very effective for the simulation of steady inviscid compressible flow problems. This approach avoids the requirement for linearisation of the discrete system, eliminating the need to store a Jacobian matrix and, thus, reducing the memory requirements significantly compared to those of many other solution schemes.

4.1. Multigrid scheme

We consider a relaxation procedure for the solution of equation (38). The method selected uses an explicit three-stage Runge–Kutta method, with local time stepping, and coefficients 0.6, 0.6 and 1.0. Suppose that the solution estimate following step r is denoted by $\mathbf{U}_f^r$. An additional relaxation step is performed, with the objective of damping high error frequencies, and this produces the improved solution estimate $\mathbf{U}_f^{r^*}$. The objective is now to find the correction $\mathbf{E}_f^r$ such that

$$\mathbf{U}_f = \mathbf{U}_f^{r^*} + \mathbf{E}_f^r \tag{41}$$

On our generated mesh, which we term the fine mesh, the defection for the improved guess at cycle r is defined as

$$\mathbf{D}_f^{r^*} = \mathbf{R}_f(\mathbf{U}_f^{r^*}) - \mathbf{s}_f \tag{42}$$

By subtracting this from Equation (38), and using Equation (41), we deduce that

$$\mathbf{R}_f(\mathbf{U}_f^{r^*} + \mathbf{E}_f^r) - \mathbf{R}_f(\mathbf{U}_f^{r^*}) = -\mathbf{D}_f^{r^*} \tag{43}$$

The fine mesh relaxation scheme will be very efficient at eliminating the highest frequencies, so that the approximation of the fine mesh error on a coarser mesh may be considered smooth. It is, therefore, more efficient to solve this equation on a coarse mesh and, for this purpose, the coarse mesh variable

$$\mathbf{U}_c^r = \mathbf{I}_c^f \mathbf{U}_f^{r^*} + \mathbf{E}_c^r \tag{44}$$

is introduced, where $\mathbf{E}_c^r$ is the coarse mesh representative of the low frequency error components of the iteration. Here, $\mathbf{I}_c^f$ is a restriction operator that maps the fine mesh unknowns on to the coarse mesh. The coarse mesh analogy of Equation (43) is now given by

$$\mathbf{R}_c(\mathbf{U}_c^r) = \mathbf{R}_c(\mathbf{I}_c^f \mathbf{U}_f^{r^*}) - \mathbf{I}_c^f \mathbf{D}_f^{r^*} \tag{45}$$

and, by defining

$$\hat{\mathbf{s}}_c = \mathbf{R}_c(\mathbf{I}_c^f \mathbf{U}_f^{r^*}) - \mathbf{I}_c^f \mathbf{D}_f^{r^*} \tag{46}$$

the coarse mesh equation may be written as

$$\mathbf{R}_c(\mathbf{U}_c^r) = \hat{\mathbf{s}}_c \tag{47}$$

which is of the same form as Equation (38). This means that the fine mesh and coarse mesh equations are similar, with the only difference arising in the form of the source term. This is a major computational advantage when compared with multigrid schemes that work directly with the error on the coarse meshes. As the majority of the computation is related to the evaluation of $\mathbf{R}$, this makes it possible to use many of the same subroutines on the coarse mesh and the fine mesh levels. The three stage Runge–Kutta method is again employed to relax the coarse mesh equation (47), yielding an improved representation $\mathbf{U}_c^{r^*}$ for the coarse mesh unknowns. As this is of the same form as Equation (38), the coarse mesh equation may itself be solved by the FAS multigrid method, thus naturally introducing as many mesh levels as required to resolve all the frequencies in the solution. The coarse mesh error may be approximated as

$$\mathbf{E}_c^{r^*} = \mathbf{I}_c^f \mathbf{U}_f^{r^*} - \mathbf{U}_c^{r^*} \tag{48}$$

and the solution on the fine mesh is corrected according to

$$\mathbf{U}_f^{r+1} = \mathbf{U}_f^{r^*} + \mathbf{I}_f^c \mathbf{E}_c^{r^*} \tag{49}$$

This correction involves a prolongation operator, $\mathbf{I}_f^c$, that maps the coarse mesh variables on to the fine mesh. We have employed a linear restriction mapping, so that the value of a variable on the coarse mesh is produced by the weighted average

$$\mathbf{U}_I^c = \frac{1}{V_I^c} \sum_J \mathbf{U}_J^f V_J^f \tag{50}$$

of the fine mesh unknowns. This formula assumes nested meshes and the summation extends over the control volumes on the fine mesh that make up the coarse mesh control volume.

For prolongation, again assuming nested meshes, the simple point injection scheme

$$\mathbf{U}_I^f = \mathbf{U}_J^c \tag{51}$$

is adopted, where V_J is the coarse mesh volume that includes node I of the fine mesh.

The approach can be thought of as converging the set of steady state equations, with the addition of the time source, at every physical time step. In this way, there are no numerical limitations on the size of the physical time step and it can be set to a value governed by accuracy concerns only. The speedup achieved by using the multigrid accelerated implicit scheme instead of simple [22], or more elaborate [23], explicit time stepping will depend upon the Strouhal number of the flow. Large Strouhal numbers generally favour the implicit approach. Typically, for the Strouhal numbers appearing in many industrial applications, the implicit approach will be well over one order of magnitude faster, for minimal additional memory cost.

4.2. Coarse mesh generation

The nested coarse meshes required for the multigrid procedure described above are obtained by agglomeration. Agglomeration works on the dual mesh of the finite volume scheme, merging control volumes in a manner designed to achieve the local coarseness ratio required [4, 24]. The approach is purely edge based and can therefore be applied on any type of mesh after the edge based data structure has been assembled. Since no mesh generation issues are raised in the procedure, the scheme is completely stable and fast. The agglomeration procedure begins by selecting a seed node from a seed list. Nodes connected to the seed node by edges that haven't already been merged are then grouped together, creating a super node. When this procedure has been completed for all nodes, internal edges in the super nodes are deleted and edges bordering the same two super nodes are merged into super edges by adding the coefficients defined in Equation (8) of the merged edges. In a similar fashion, the boundary coefficients defined by Equation (9) are also added. When the agglomeration loop is complete, super nodes that are only connected to one edge are identified and merged using this connectivity. This can occur if a super node control volume is completely surrounded by another control volume and possibly a part of the boundary of the computational domain.

4.3. Implementation issues

The computational implementation consists of four main components. The first component is mesh movement, for which a geometry file and the surface movement information must be prescribed. The internal nodes of the mesh are then positioned by employing the mesh movement algorithm. The second component is remeshing, in which the local quality of the moved mesh is investigated and remeshing is performed if required. The third component is preprocessing, which sets up the edge based data structure, from the mesh information, generates the coarse grids and constructs the inter grid interpolation arrays for the multigrid procedure. The final component is the equation solving, in which the output from the pre-processor is accessed and the time dependent discrete equations are solved for one physical time step. These components are loosely coupled, employing a driving script and file communication. Such a loose coupling will, inevitably, adversely affect the solution time compared to implementations which do not require I/O communication. However, since relatively few time steps are required for the implicit schemes, this effect is usually small.

5. EXAMPLES

Three examples are included to illustrate the capability of the procedure. The initial example is used to compare the performance of the implicit and explicit schemes for a problem involving a simple geometry undergoing a single prescribed oscillation. The second example involves a more complicated prescribed motion of a realistic aircraft configuration. Both of these simulations are undertaken without the need for remeshing. The final example demonstrates the use of the remeshing capability and involves the simulation of the release of a pair of stores from a complete aircraft configuration.

5.1. ONERA M6 wing

The first example involves flow over an ONERA M6 wing that is undergoing a prescribed oscillation. The free stream Mach number is 0.84 and the initial angle of attack is 3.06°. The mesh employed consists of 264 896 tetrahedral elements. The wing oscillates in pitch, about an axis normal to the midpoint of the root chord, with an amplitude of 5° and Strouhal numbers of 1.68, 3.36, 6.72 and 13.44 are investigated. For this configuration, these Strouhal numbers can be considered to be at the lower end of the range of practical interest. The computations are initiated from free stream conditions and 10 full cycles are calculated. Periodic behaviour is found to occur after five cycles. Within each physical time step, the solution of the implicit scheme is converged to three orders of magnitude and this, typically, requires of the order of 10–30 multigrid cycles, depending upon the number of physical time steps used. Five grid levels are employed in the multigrid procedure. The lift histories of the cyclic motion, for various physical time step sizes, are compared with the results of an explicit scheme in Figure 3. In this figure, the label 1575 refers to the use of an explicit solver. It can be seen

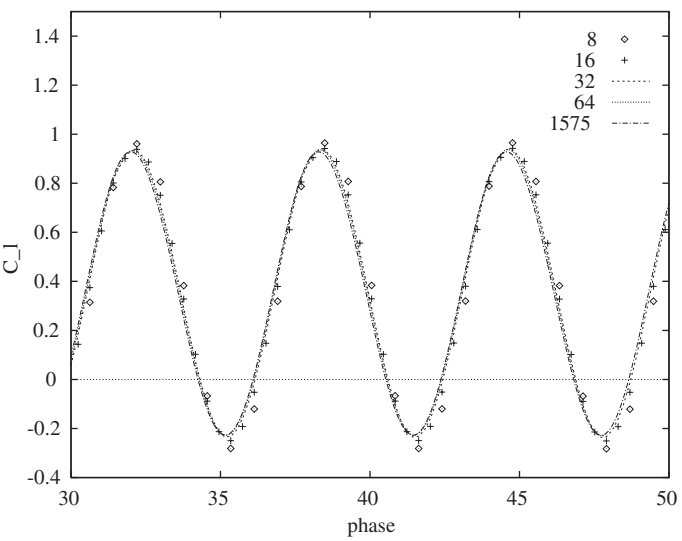

Figure 3. Oscillating ONERA M6 wing: plot of a segment of the lift history obtained using different numbers of time steps per cycle.

Table I. Oscillating ONERA M6 wing: speed up of the implicit solver compared with an explicit procedure for $St = 3.36$.

SPC	Movement time (%)	Preprocessor time (%)	Solver time (%)	Speed up
8	17	10	73	9.2
16	14	13	73	6.4
32	12	16	72	4.3
64	12	21	67	2.8

Table II. Oscillating ONERA M6 wing: speed up of the implicit solver compared with an explicit procedure for different Strouhal numbers.

St	Normed total time (%)	Speed up
1.68	1.00	3.4
3.36	1.05	6.4
6.72	1.08	12.3
13.44	1.10	24.4

that good accuracy is obtained if 16 time steps are used per cycle in this case. A comparison of the speed up of the implicit solver compared to the explicit scheme, for the case when the Strouhal number is 3.36, is shown in Table I. In this table, SPC refers to the number of time steps used per cycle. The percentage of the CPU time required by the various stages of the solution procedure is also shown. It should be noted that these figures include reading and writing from disk. In the explicit solver, the movement and preprocessor stages were called every 10 time steps, with only one movement used per sub step. The error associated with this level of approximation is usually considered negligible for explicit schemes. In spite of this, the speed up of the implicit formulation is considerable. It can also be observed that, as the number of cycles is increased for the implicit scheme, the convergence of the solver stage is quicker and, thus, requires a smaller percentage of the CPU time. The number of sub steps in the grid movement algorithm was varied according to the number of physical time steps used in the calculation, so that each full cycle of the calculation involved 800 movement sub steps. This is more than adequate for this test case. For this problem, the speed up achieved as a function of the Strouhal number, is illustrated in Table II. These figures were produced using 16 steps per cycle. Provided the allowable time step stability limit is smaller than the time step required for a desired accuracy level, the CPU time required for an explicit scheme is directly proportional to the characteristic timescale of the calculation. For the implicit scheme, the dependence of the CPU time of the Strouhal number is markedly smaller. It is observed that, even for a very low Strouhal number of 1.68, significant speedup is achieved with the implicit approach. Plots of the lift polar, which is the variation of the lift coefficient with the phase angle, are shown in Figure 4.

5.2. B60 aircraft

In the second example, the simulation of the flow over an oscillating B60 aircraft geometry is considered. The free stream Mach number is 0.803 and the initial angle of attack is 2.738

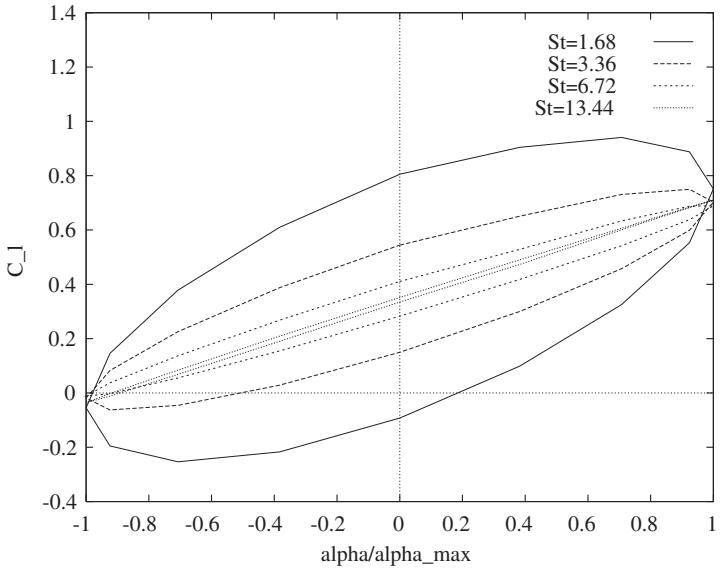

Figure 4. Oscillating ONERA M6 wing: the computed lift polars at different Strouhal numbers.

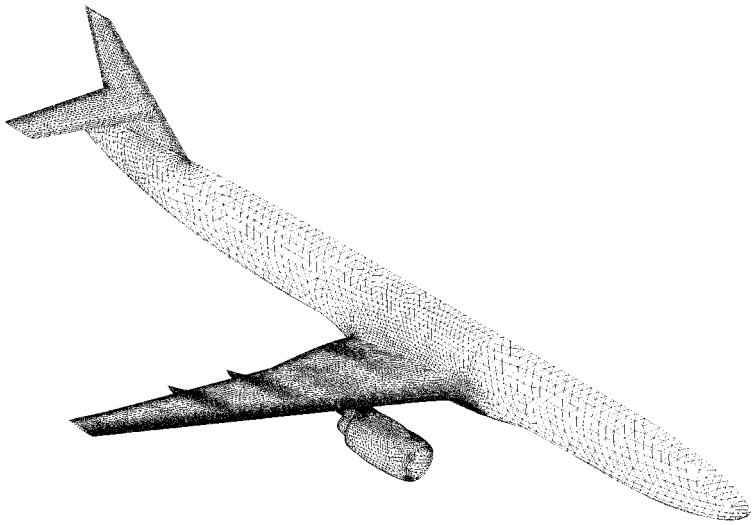

Figure 5. Oscillating B60 configuration: view of the surface mesh employed.

degrees. For this simulation, the wings and engines alone are subjected to a prescribed movement. The wing movement is defined in a piecewise linear fashion, with a pitch amplitude of one degree and a heave amplitude of 2% of the wing semi span at the wing midpoint and a pitch amplitude of five degrees and heave of 6.5% at the wing tip. The wing root is held fixed. The movement is sinusoidal, with a Strouhal number for all movement modes of 13.3, and

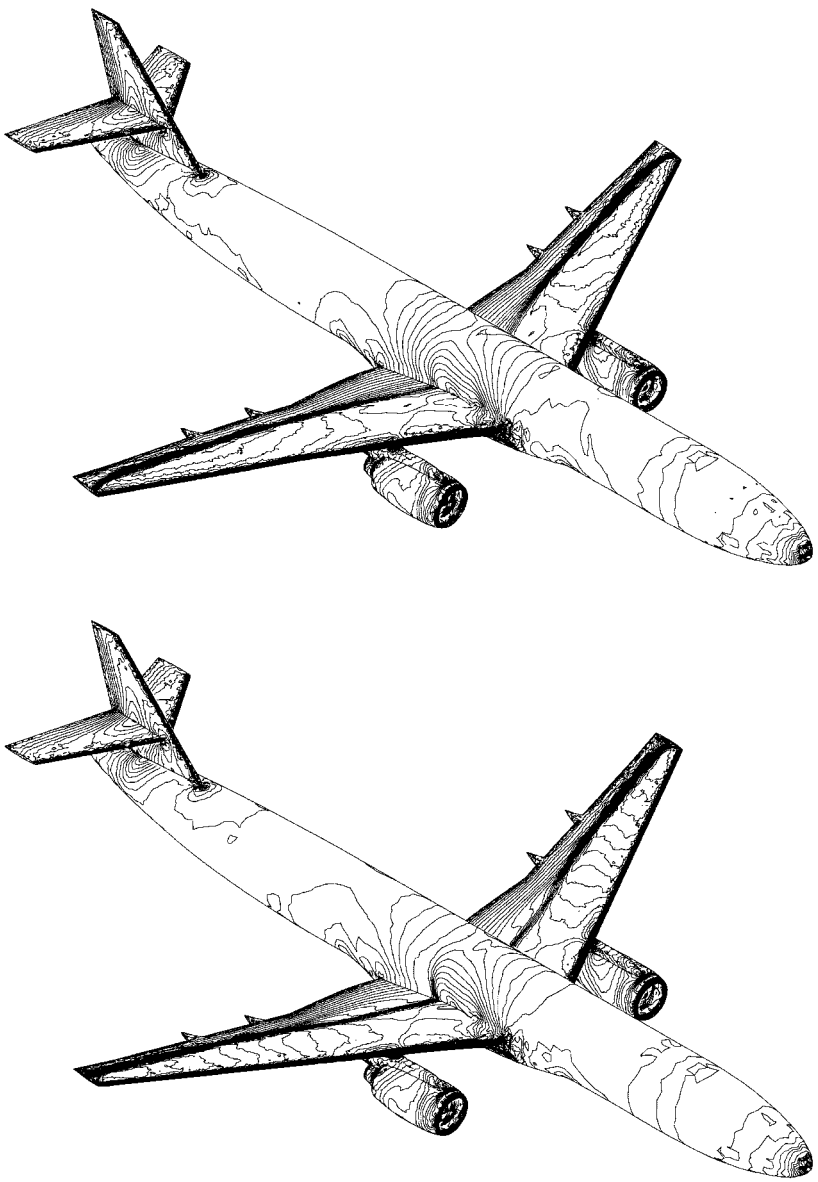

Figure 6. Oscillating B60 configuration: surface contours of pressure at two different times.

32 physical time steps are performed for each cycle. The mesh employed consists of 775 877 tetrahedra and a view of the discretized surface is given in Figure 5. For this example, an estimated speedup of 12.5 is achieved compared to the explicit approach. Snapshots of the surface pressure, at two different times, are given in Figure 6, while the calculated lift polar is shown in Figure 7.

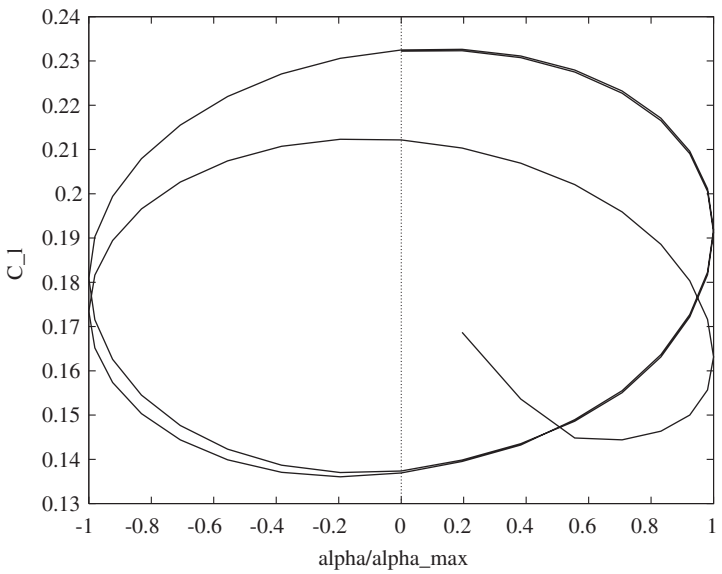

Figure 7. Oscillating B60 configurations: the computed lift polar.

5.3. Store release

The final example involves the simulation of the simultaneous release of two stores from a generic F16 aircraft configuration. The geometrical definition of the aircraft was provided by EADS-M, München, Germany. The stores are assumed to be completely filled with a substance of density 1 250 times that of the free stream air. The free stream 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 a second order accurate time integration. Local remeshing is employed and each mesh consists of about 2.7 million tetrahedral elements. The resulting surface pressure distribution, at four equally spaced time intervals, is shown in Figure 8.

6. CONCLUSION

A method for the simulation of inviscid compressible fluid flow problems involving moving geometries has been described. The method employs a mesh movement/local remeshing approach which is capable of accurately and robustly solving problems involving a wide spectrum of mesh deformation. The multigrid accelerated implicit time stepping equation procedure is competitive with domain decomposed explicit schemes for low Strouhal numbers and superior for time dependent flows with medium and large characteristic times. Future publications will focus on the extension of the method to time dependent turbulent flows.

ACKNOWLEDGEMENTS

K. A. Sørensen acknowledges the sponsorship of The Research Council of Norway, project number 125676/410.

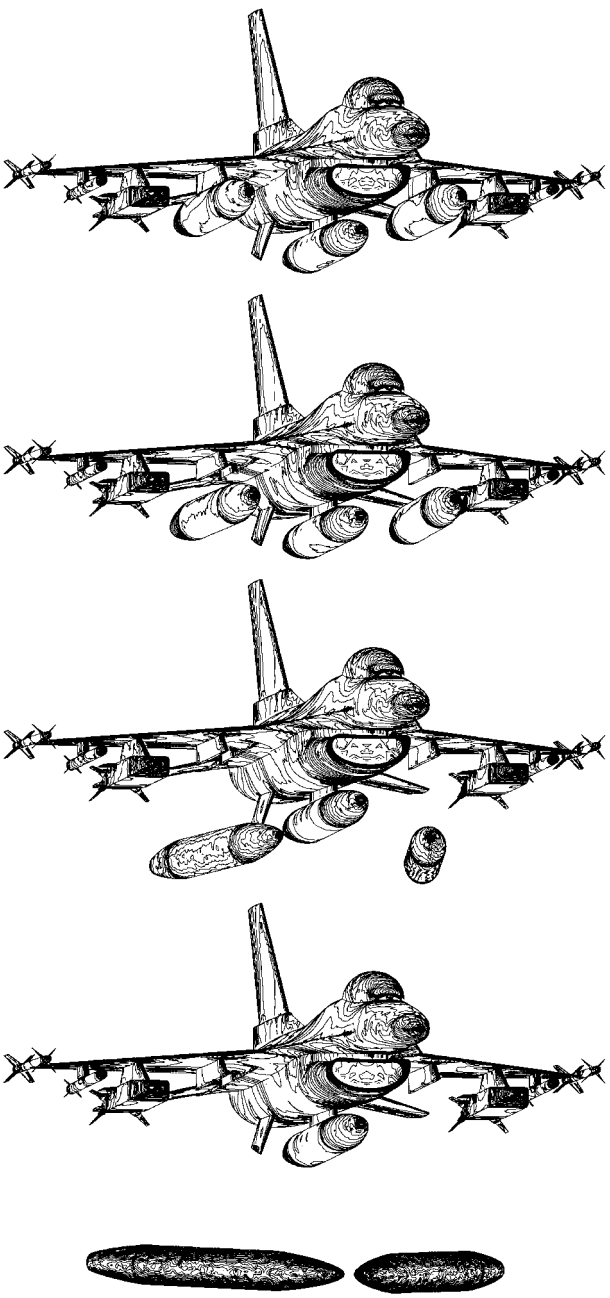

Figure 8. Simultaneous release of two stores from a generic F16 aircraft configuration: computed contours of pressure at 0.29 second intervals.

REFERENCES

1. Löhner R, Morgan K, Zienkiewicz OC. The use of domain splitting with an explicit hyperbolic solver. *Computer Methods in Applied Mechanics and Engineering* 1984; **45**:313–329.
2. Vilsmeier R, Hänel D. Adaptive solutions for unsteady laminar flows on unstructured grids. *International Journal for Numerical Methods in Fluids* 1995; **22**:85–101.
3. 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.
4. Lallemand MH, Steve H, Dervieux A. Unstructured multigridding by volume agglomeration: current status. *Computers & Fluids* 1992; **21**:397–433.
5. Morano E, Guillard H, Dervieux A, Leclercq MP, Stoufflet B. Faster relaxations for non-structured multigrid with Voronoi coarsening. *Proceedings of the ECCOMAS 92 Conference* 1992; **1**:69–74.
6. Müller JD. Coarsening 3-D hybrid meshes for multigrid methods. *Proceedings of the 9th Copper Mountain Multigrid Conference*, 1999.
7. Dougherty FC, Benek JA, Steger JL. On the application of a chimera grid scheme to store separation. *NASA Technical report*, *TM-88193*, 1995.
8. Nakahashi K, Togashi F, Sharov D. An intergrid-boundary definition method for overset unstructured grid approach. *AIAA Paper 99-3304*, 1999.
9. Morgan K, Peraire J, Peiró J. Unstructured grid methods for high speed compressible flows. In *The Mathematics of Finite Elements and Applications: Highlights 1993*, Whiteman JR (ed.). Wiley: Chichester, 1994; 215–241.
10. 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.
11. Koobus B, Fahrat 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.
12. 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.
13. Venkatakrishnan V, Mavriplis DJ. Implicit method for the computation of unsteady flows on unstructured grids. *Journal of Computational Physics* 1996; **127**:380–397.
14. 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* 1992; **143**.
15. 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.
16. 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.
17. Vahdati M, Morgan K, Peraire J. Computation of viscous compressible flows using an upwind algorithm and unstructured meshes. In *Computational Nonlinear Mechanics in Aerospace Engineering*, Atluri SN (ed.). AIAA: Washington, 1992; 479–505.
18. Thomas PD, Lombard CK. Geometric conservation law and its application to flow computations on moving grids. *AIAA Journal* 1979; **17**:1030–1037.
19. Lesoinne M, Fahrat C. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**:71–90.
20. 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.
21. Brandt A. Multi-level adaptive solutions to boundary value problems. *Mathematics of Computation* 1977; **21**:333–390.
22. Hassan O, Probert EJ, Morgan K. Unstructured mesh procedures for the simulation of three-dimensional transient compressible flows with moving boundary components. *International Journal for Numerical Methods in Fluids* 1998; **27**:41–55.
23. Morgan K, Hassan O, Sørensen KA, Weatherill NP. Unstructured grid procedures for 3D transient compressible flow with moving boundaries. In *Innovative Tools for Scientific Computation in Aeronautical Engineering* Périaux J, Joly P, Pironneau O, Oñate E (eds). CIMNE: Barcelona, 2001; 28–37.
24. 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.

