Report no. 95/17

Implicit time accurate solutions on unstructured dynamic grids

P. I. Crumpton   M. B. Giles

In this paper an unstructured multigrid algorithm is used as an iterative solution procedure for the discrete equations arising from an implicit time discretisation of the unsteady Euler equations on tetrahedral grids. To calculate unsteady flows due to oscillating boundaries, a novel grid movement algorithm is introduced, in which an elliptic equation with a nonlinear diffusion coefficient is used to define the displacement of interior grid nodes. This allows large grid displacements to be calculated in a single step. The multigrid technique uses a edge–collapsing algorithm to generate a sequence of grids, and a pseudo–timestepping smoother. On the coarser grids, no grid motion is used. Instead, surface normals are rotated consistently and transfer/interpolation weights are based on the time-averaged grid coordinates. A 2D NACA0012 test case is used to validate the program. 3D results are presented for the M6 wing and a full aircraft configuration.

This work was presented at the 12th AIAA conference on computational fluid dynamics 1995 (Paper AIAA 95-1671).

This work was performed within Oxford Parallel, with financial support from Rolls–Royce plc, DTI and SERC.

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England   OX1 3QD

April, 1997

1 Introduction

The eventual aim of this work is to model the unsteady aeroelastic behaviour of complex configurations, such as a complete aircraft. In moving towards this goal the present work concentrates on the prediction of the unsteady inviscid response to a body oscillating at low reduced frequency. An example of this type of problem is wing flutter or the periodic pitching of an aeroplane.

The spatial geometry is discretised using an unstructured tetrahedral grid. To account for the movement of some of the boundary surfaces, the surface and interior grid nodes are moved in a consistent manner, keeping the connectivity of the grid fixed. One of the key aspects of this paper is the manner in which this consistent displacement is computed, enabling very large displacements to be made in a single step.

A consequence of the low reduced frequency is that explicit methods have a stringent CFL restriction which leads to very large CPU requirements [1]. Following the approach of Jameson [2] this is avoided by using an implicit time discretisation, with the resulting equations being solved using a multigrid method. This employs pseudo-timemarching purely as a iterative smoothing technique. Particular attention is paid to two aspects of this procedure, the generation of a sequence of grids by recursively 'collapsing' edges of the original fine grid, and the formulation of the coarse grid discrete operator when the fine grid is in motion.

Although an efficient solver is employed to minimise the computational cost of each time step, CPU requirements for complex configurations are still very high. To reduce the otherwise high computation time, a distributed-memory parallel computer is employed, as in [3]. The efficient utilisation of this hardware is very simply achieved by use of the OPlus library [4] through the straightforward insertion of OPlus FORTRAN 77 subroutine calls [5].

The content of this paper is as follows; problem formulation, the grid movement algorithm, the implicit solver, results, discussion and conclusions.

2 Problem formulation

The integral form of the unsteady compressible Euler equations for a deforming control volume is

$$\frac{d}{dt}\iiint_{V(t)} u\,dV = -\iint_{\partial V(t)} (\boldsymbol{F} - u\dot{\boldsymbol{x}}) \cdot d\boldsymbol{A}, \tag{2.1}$$

where $u = (\rho, \rho u, \rho v, \rho w, E)^T$ is the usual vector of conserved variables, $\boldsymbol{F} = (f, g, h)^T$ are the inviscid fluxes and $\dot{\boldsymbol{x}}$ is the velocity of the moving boundary.

Using an unstructured tetrahedral grid with the flow variables defined at the vertices, the flux integral for an individual tetrahedron labelled α can be defined as

$$R_\alpha = \sum_\beta (\boldsymbol{F}_\beta - u_\beta \dot{\boldsymbol{x}}_\beta) \cdot \boldsymbol{A}_\beta.$$

The summation is over the four faces of the terahedron. On each face $\boldsymbol{A}_\beta$ is the triangular face area vector and the fluxes are based on an average of the values at the three corner nodes of the face.

Using this definition, and a standard second order backward difference for the time derivative, gives the algorithm

$$\begin{aligned} \frac{3}{2\Delta t} V_j^n U_j^n - \frac{2}{\Delta t} V_j^{n-1} U_j^{n-1} + \frac{1}{2\Delta t} V_j^{n-2} U_j^{n-2} & \\ + \sum_\alpha D_{\alpha,j} R_\alpha(U^n) &= 0 \end{aligned}$$

The summation is over the cells surrounding node j. V_j^n is the nodal volume, defined as one quarter of the volume of the surrounding cells, at time level n. $D_{\alpha,j}$ is a distribution matrix, defining how the residual R_α is to be distributed to the four corners of cell α. In a Galerkin approximation this would simply be $\frac{1}{4}I$.

Defining the nonlinear operator M as

$$M_j(U^n) = \frac{3}{2\Delta t} V_j^n U_j^n + \sum_\alpha D_{\alpha,j} R_\alpha(U^n).$$

and the right hand side source term as

$$f^n = \frac{2}{\Delta t} V_j^{n-1} U_j^{n-1} - \frac{1}{2\Delta t} V_j^{n-2} U_j^{n-2}$$

the equations for time level n can be written as

$$M_j(U^n) = f_j^n, \quad \forall j \quad \text{for } n = 1, 2, 3, ... \tag{2.2}$$

This nonlinear system of coupled equations can now be solved by introducing a pseudo-unsteady term

$$V_j^n \frac{\partial U_j}{\partial \tau} + M_j(U) = f_j^n \tag{2.3}$$

and 'time-marching' the solution using local pseudo-timesteps $\Delta\tau_j$ until U_j converges to U_j^n.

This is the basis of several approaches to solving the coupled implicit time-marching equations (e.g. [1]). In this work we follow the ideas of Jameson in using a multigrid procedure to accelerate this pseudo-timestepping evolution to reduce the computational cost of each implicit timestep. Using an existing multigrid Lax-Wendroff solver which has proven to be computationally efficient and robust for steady flows around complex geometries, the distribution matrices in Eq(2.2) are taken to be Lax-Wendroff distribution matrices involving the local pseudo-timestep $\Delta\tau_j$ and the pseudo-time derivative of U_j is approximated by a simple forward difference in pseudo-time. In the quasi-steady limit in which the real timestep $\Delta t \to \infty$, the whole procedure for a single time level of the unsteady evolution reduces to the standard steady-state Lax-Wendroff multigrid algorithm.

A blend of second and fourth difference smoothing is added to the basic Lax-Wendroff algorithm. Details of this are given in Reference [6] which shows the importance of constructing a fourth difference smoothing operator which does not affect linear functions on irregular meshes.

3 Grid movement

The present work models the unsteady aerodynamic response induced by the periodic oscillation of a complex configuration. Consequently, the movement of the grid is pre–determined and can be expressed as

$$\boldsymbol{x}(t) = \boldsymbol{x}_o + \sin(\omega t)(a^+ \delta\boldsymbol{x}^+ + a^- \delta\boldsymbol{x}^-) \tag{3.1}$$

where

$$\begin{aligned} a^+ &= \frac{1}{2}(1 + sgn(\sin(\omega t)) \\ a^- &= \frac{1}{2}(1 - sgn(\sin(\omega t)) \ . \end{aligned}$$

Here, $\boldsymbol{x}_o$ are the co–ordinates of an unperturbed grid, and $\delta\boldsymbol{x}^+$ and $\delta\boldsymbol{x}^-$ are the maximum deflections at the two extremes of the periodic motion. The connectivity of the grid used to discretise $\boldsymbol{x}(t)$ is taken to be that of the original unperturbed grid $\boldsymbol{x}_o$. It is assumed that if $\boldsymbol{x}$, $\boldsymbol{x}+\delta\boldsymbol{x}^+$ and $\boldsymbol{x}+\delta\boldsymbol{x}^-$ are all valid grids in the sense of containing only cells with positive volume, then so is $\boldsymbol{x}(t)$; to date, this has always been the case.

The displacements $\delta\boldsymbol{x}^\pm$ are specified on the surfaces of the complex body, and then some heuristic method is needed to assign values for interior grid points. One popular technique uses a spring analogy [1], where each edge of the displaced computational mesh is modelled as a spring in equilibrium. The mesh at the boundary is moved by the prescribed amount, and the spring system is solved to reach a new equilibrium. This is usually performed in conjunction with an explicit time marching procedure for the gas dynamics, which has a severe CFL restriction. Consequently, the mesh need only be moved a cell thickness or so on the boundary. In the work in this paper, because an implicit solver is being used this timestep restriction is avoided and the spring analogy was found to be ineffective for the large perturbations required. An alternative was to move the mesh to the required deflection via many internal steps, but this was thought to be unattractive, since the problem would become even more severe for future viscous calculations in which the surface cells would be extremely thin. Therefore, the following alternative approach was developed.

The displacement $\delta\boldsymbol{x}$ from grid $\boldsymbol{x}_o$ is prescribed by the following equation

$$\nabla \cdot (k(\boldsymbol{x}_o + \delta\boldsymbol{x}) \nabla \delta\boldsymbol{x}) = 0 \qquad (3.2)$$

with Dirichlet boundary conditions. The nonlinear diffusion coefficient k is a constant in each cell α of mesh $\boldsymbol{x}_o$, and is given by

$$k_\alpha(\boldsymbol{x}_o + \delta\boldsymbol{x}) = \frac{1}{\max(Vol(\boldsymbol{x}_o + \delta\boldsymbol{x}, \alpha), \epsilon)} \quad .$$

The function $Vol(\boldsymbol{x}, \alpha)$ returns the volume of cell α of grid $\boldsymbol{x}$, and ϵ is a small positive number needed to prevent k from becoming negative. The basic idea behind this choice of k is that relatively small cells will have a relatively large diffusion coefficient resulting in relatively small gradients in $\delta\boldsymbol{x}$. Therefore these small cells, which are usually very near to the surface of the body, tend to undergo rigid body motion (a combination of translation and rotation) in common with the local surface boundary. This avoids the possibility of small cells having very large changes in volume, even leading to negative volumes, due to rapid variations in $\delta\boldsymbol{x}$.

Equation (3.2) is discretised using a straightforward Galerkin finite element approximation on the unperturbed mesh $\boldsymbol{x}_o$. The resulting nonlinear system is solved using a under–relaxed Jacobi iteration, with the non–linear k being evaluated at a previous iteration. For large boundary deflections, the initial guess to the Jacobi iteration is found from interpolation of the solution from a coarser grid, which is used within the multigrid method. This greatly enhanced the robustness of this method. For complex configurations, the convergence of the Jacobi iteration can be poor, but with the multigrid initial guess, the perturbed mesh is valid (in the sense that all volumes are positive) long before numerical convergence. Consequently, the iterative procedure is usually stopped after a fixed number of iterations.

Fig. 9 shows the grids resulting from a pitching of a civil aeroplane configuration $\pm 15^o$. The grids all have 0.75×10^6 cells. This code, like the flow solver, has been developed using the OPlus library, and consequently executes in parallel. The time taken to move the mesh is negligible in comparison to the CPU time of the unsteady Euler solver. The figure shows how the clustering of cells is maintained during the mesh movement without excessive skewing of tetrahedra.

This approach has links with the spring analogy grid movement, [1], where the spring coefficient is scaled inversely proportionately to the length of an edge. The resulting computational molecule is clearly linked with a diffusion operator. The success of the current approach is believed to be the choice of k which is defined over cell volumes, which are used in the criteria for a valid mesh. In contrast, the spring analogy uses only edges, which are not directly linked to the cells in any way during the mesh movement process.

4 Solution of implicit equations

4.1 Multigrid approach

As outlined in an earlier section, a multigrid procedure is used to iteratively solve the fully-coupled nonlinear system of equations that arise from the implicit time discretisation of the unsteady flow equations. Multigrid has mainly been used for structured meshes, however, it is now being increasingly used for unstructured meshes. The four basic components needed for multigrid are

1. A prolongation operator I_h^H which transfers corrections from coarse to fine grids;
2. A restriction operator I_h^H which transfers residuals and solutions from fine to coarse grids;
3. A coarse grid operator $M_H(Q_H)$ which is strongly related to the fine grid operator $M_h(Q_h)$ (usually by being the same approximation operator but applied to a coarser grid);
4. A smoother or relaxation scheme which damps high frequency error modes.

Within a CFD framework the four major approaches to construct these operators are outlined below.

1. An independent sequence of grids can be produced by a black box grid generator and then linked together. This has been adopted by several authors ([2], [7], [3]). This allows the flexibility of using any grid based solver, however, the generation of the sequence of grids is not automatic, and needs a tight coupling between the grid generator and the multigrid method.
2. An increasingly popular approach is agglomeration, which has been successfully applied to very complex problems [8]. Here a coarse grid matrix is constructed by the "agglomeration" of fine grid volumes. This is completely automatic, but is somewhat reliant on an edge-based data structure.
3. Another strategy is to produce fine grids from a coarse grid, preferably in some sort of adaptive refinement procedure [9]. This seems an attractive proposition, however, this requires a strong coupling between the grid refinement and the configuration surface spline definition.
4. The philosophy adopted here, as in [10], is to use an automatic point removal algorithm, which is completely automatic, needing no interaction with the grid generation process. The resulting grid sequence can be used by any grid based algorithm, including those which use an edge based data structure.

The methodology used to remove points is very straightforward, and is based on "collapsing edges", see Fig. 1. Given an edge with nodes i and j of a mesh the algorithm proceeds as follows:

1. construct T, a list of all cells connected to the current edge;
2. construct F, a list of the external faces of the set T;
3. reconnect the nodes in the faces of F with a new node $\boldsymbol{x}_k$, if all tetrahedra have positive volume; otherwise reject the edge collapse as being invalid.

The choice of $\boldsymbol{x}_k$ depends on the location of the two nodes. If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are both interior nodes then $\boldsymbol{x}_k$ is defined to be the average of the two. On the other hand, if $\boldsymbol{x}_i$ is an interior node and $\boldsymbol{x}_j$ lies on a boundary then $\boldsymbol{x}_k$ is set equal to $\boldsymbol{x}_j$ to preserve the geometry of the boundary surface as much as possible. Extending this idea to include situations in which a node may lie on more than one boundary surface (such as at a wing/fuselage junction) the general rule is to count the number of surfaces on which each node lies. If they are equal then $\boldsymbol{x}_k$ is assigned the average coordinates; if they are unequal $\boldsymbol{x}_k$ is assigned the coordinates of the node lying on more surfaces. An example of the grids produced near boundaries is given in Fig. 10.

It now remains to choose which edges are collapsed. This is done with a simple strategy which begins by 'colouring' all nodes so that no two nodes of the same colour are connected by an edge. (This is very similar to the colouring of edges for vector computations to avoid indirect addressing conflicts in updating nodal data.)

1. for all nodes i of the first colour, collapse the shortest edge connected to i
2. if the number of nodes remaining is still above the desired level, repeat with the next colour if it exists, otherwise re–colour and begin again.

The motivation behind colouring the nodes is to visit nodes, and therefore remove nodes, uniformly over the mesh. The shortest edge is chosen to make the collapsed grid as isotropic as possible. Isotropic grids, without any highly stretched cells, enable the smoother to damp equally in all directions, hopefully ensuring fast and robust multigrid convergence.

At the end of this point removal method, for each coarse grid point j a list of all the fine nodes that have been collapsed onto the coarse node R_j^H is constructed. Using this information the restriction of a node based quantity ϕ_j is defined as the volume weighting

$$\phi_j^H := \frac{\sum\limits_{i\in R_j^H} V_i^h \phi_i^h}{\sum\limits_{i\in R_j^H} V_i^h}$$

and the prolongation is the simple injection process

$$\forall i \in R_j^H, \quad \phi_i^h := \phi_j^H \quad .$$

The mesh quality of the coarse grid can be poor, but to date it has not been necessary to implement grid smoothing, such as face swapping, for the coarse grids. These are only used to accelerate the convergence of the fine mesh computation and the multigrid algorithm has proven to be sufficiently robust to cope with the coarse grids. For steady-state applications, a standard FAS multigrid method is used with the restriction and prolongation operators defined above, and with a coarse grid operator which is similar to the fine grid operator except in two respects. One is the use of a fixed level of second difference smoothing instead of the adaptive blend of second and fourth difference smoothing. The other concerns the imposition of flow tangency at surface grid nodes. On the fine grid, a local surface normal is calculated from the surrounding boundary faces, but on the coarser grid levels it is obtained by restriction from the fine grid because of the poor quality of the surface definition on the coarse grids, see Fig. 5.

4.2 Treatment of moving meshes

The previous section has described the generation of a sequence of fixed grids, with prolongation and restriction weights relating variables between grids. When the fine grid moves, the most natural treatment is to move the coarse grids by restricting the fine grid deflections thus relating the fine and coarse grids at their time-averaged position to define the coarse grid coordinates at all times during the oscillation. In this manner the coarse grids oscillate consistently with the fine grid. During a multigrid iteration for each implicit timestep, the fine grid residual is tranferred onto the coarser grid, the coarse grid equations are solved (applying the multigrid recursively on even coarser grid levels) taking into account the motion of the coarse grid and then finally the changes on the coarse grid are interpolated back onto the fine grid.

For small amplitude oscillations this procedure works well, but at large amplitudes there can be problems with the volume of some deforming coarse grid cells becoming negative. This can lead to complete failure of the multigrid iterative solution procedure.

To overcome this problem, the multigrid iteration on all coarse grid levels is performed on fixed grids corresponding to the time-averaged position. To understand why this is justified it is necessary to consider the discrete operator on the moving fine grid as operating on the unsteady flow variables, grid coordinates and grid velocities, $M(U, \boldsymbol{x}, \dot{\boldsymbol{x}})$. For small amplitude oscillations about the time-average solution, this operator can be approximately linearised, giving

$$M \approx \frac{\partial M}{\partial U}\tilde{U} + \frac{\partial M}{\partial \boldsymbol{x}}\tilde{\boldsymbol{x}} + \frac{\partial M}{\partial \dot{\boldsymbol{x}}}\tilde{\dot{\boldsymbol{x}}}$$

where $\widetilde{U}, \widetilde{\boldsymbol{x}}, \tilde{\dot{\boldsymbol{x}}}$ are the unsteady perturbation quantities and the Jacobian matrices, $\frac{\partial M}{\partial U}, \frac{\partial M}{\partial x}, \frac{\partial M}{\partial \dot{x}}$ are evaluated on the stationary time-averaged grid with the time-averaged flow variables.

On the coarser grids, using the FAS multigrid method, the equation to be solved is

$$M_H(U_H) = M_H(I_h^H U_h) + I_h^H(f_h - M_h(U_h)) \tag{4.1}$$

where $f_h - M_h(U_h)$ represents the remaining residual on the finer grid and $I_h^H U_h$ is the transferred solution onto the coarser grid which is used as the starting point for the coarse grid computation. If the coarse grid operator M_H is taken to be the same as the fine grid operator on a moving coarse grid, then using the linear approximation derived above the same grid motion terms appear in both $M_H(U_H)$ and $M_H(I_h^H U_h)$, and so cancel. Hence, to linear order the same result is obtained by keeping the coarse grid fixed.

Even for large amplitude oscillations of the fine grid, this procedure solving the coarse grid multigrid equations on the fixed grids always accelerates the convergence of the overall iterative solution of the implicit fine grid solutions. It obviously avoids the possibility of negative cell volumes on the coarser grids, and cannot affect the final converged solution that is obtained on the moving fine grid.

5 Results and discussion

5.1 NACA0012

The first case studied is a pitching NACA0012 airfoil, with freestream conditions: $\alpha = 0.016$, $M_\infty = 0.755$. The airfoil is rotated about a quarter chord, to $\pm 2.51^o$ with reduced frequency $\omega = 0.1682$. The sequence of grids used for this calculation can be seen in Fig. 2. The fine grid was generated by extending a 2D triangular grid into a 3D prismatic grid that was then split into tetrahedra. Coarse grids were generated by removing three quarters of the fine grid points. The sequence is summarised below:

grid	1	2	3
# cells	51690	12933	2432
# nodes	13326	3256	806

Fig. 3 shows the lift coefficient plotted against the time dependent angle of attack $\alpha(t)$ for 4 periods, with 64 timesteps per period. Clearly a periodic solution has been achieved. For this case the experimental data from [11] is shown; this level of agreement is typical for an inviscid approximation, see [12].

For this test problem a study was performed to examine the convergence as the size of the implicit timestep is reduced. Fig. 4 shows the final periodic solution for calculations with 8, 16, 32 and 64 timesteps per period, the 32 and

64 timesteps per period are indistinguishable to plotting accuracy. To quantify the order of convergence, the r.m.s. change in the unsteady lift as the timestep is halved,

$$E = \left(\frac{\omega}{2\pi} \int_0^{2\pi/\omega} \left(C_L^{\Delta t}(t) - C_L^{\Delta t/2}(t) \right)^2 dt \right)^{1/2},$$

is presented in the table below. The results confirm the second order convergence expected because of the second order backward time discretisation.

$\omega \Delta t / 2\pi$	1/8	1/16	1/32
E	0.0451	0.0128	0.00283

5.2 M6 wing

The second test case is a a pitching M6 wing with freestream conditions: $\alpha = 3.06$, $M_\infty = 0.84$. For this case, the unsteady oscillation is a rotation of $\pm 5^o$ about an axis perpendicular to the symmetry plane at the root mid-chord. The grid sequence was generated by removing $\frac{7}{8}^{th}$ of the finer grid points; these grids are shown in Fig. 5 and are tabulated below.

grid	1	2	3
# cells	251721	32294	3277
# nodes	46331	5578	671

This case was run with a range of reduced frequencies, $\omega = 0.1, 0.02, 0.004$, all with 16 timesteps per period. All frequencies achieved a periodic solution after 4 periods. Fig. 6 shows the periodic lift variation plotted against $\sin(\omega t)$ for all frequencies. The expected trend is observed, with a reduced level of hysteresis as the frequency is reduced. In the quasi-steady limit there would be no hysteresis, with the lift coefficient being a direct function of the angle of attack.

The convergence histories for the first 17 timesteps is shown in Fig. 7. Here a work unit is equivalent to a single grid relaxation. Robust convergence of 4 orders of magnitude is achieved within 400 single grid iterations.

5.3 Aircraft configuration

The final calculation demonstrates the ability to perform calculations on a complex configuration. The rotation of the aeroplane is less severe than that shown in Fig. 9, only $\pm 2.5^o$, as the Euler solver will not give sensible solutions at high angles of attack at which viscous effects will prevail. The grid sequence used for this calculation is shown in Fig. 10 and is summarised below.

grid	1	2	3
# cells	746227	94821	9991
# nodes	137094	16308	2025

Fig. 8 shows the lift (evaluated in grid units) plotted against non-dimensional time for a reduced frequency of 0.01 (based on a root chord of 333 grid units), clearly a periodic solution has been obtained. For each physical timestep the residual was reduced by 3–4 orders of magnitude, which took between 10 and 20 minutes elapsed time on an 8 processor IBM SP1.

6 Parallel computing

Despite the use of the multigrid procedure to solve the implicit equations at each timestep of the unsteady flow calculation, the overall computational requirements to compute a few periods of oscillation are still very substantial. As a consequence, practical calculations for complex geometries require the use of parallel computing to produce results within an acceptable timescale. This is true both for 'production' calculations and during code development.

For the work in this paper, the code has been developed and maintained using the OPlus parallel harness [4], which allows a single source FORTRAN code to be executed on a wide variety of parallel hardware; the calculations in this paper were performed on an 8-processor distributed memory IBM SP1 and a 4-processor shared-memory Silicon Graphics Power Challenge. Although the same source code can also be compiled and executed sequentially, the code was written from the start using the OPlus harness, and almost all algorithm development and code debugging was performed using parallel computations. Sequential computation for realistic 3D grids would have severely slowed the development process.

The distributed visualisation software, pV3, [13] was vital in the development process. By interfacing it to the distributed OPlus computation it was possible to observe the evolution of the solution as it was being computed in parallel. This was invaluable in understanding the behaviour of the computation and thereby improving the numerical smoothing formulation and the multigrid algorithm.

Parallel efficiency statistics have not been generated for the computations in this paper. The parallel efficiency for steady-state multigrid calculations has been demonstrated previously for both the IBM SP1 and the Silicon Graphics Power Challenge [3].

7 Conclusion

A mesh movement algorithm has been introduced, that robustly moves complex grids through large deflections. This is based on a nonlinear elliptic p.d.e. with a diffusion coefficient that is inversely proportional to the cell volume on the displaced grid.

A point removal strategy based on collapsing edges is demonstrated to be a robust and automatic way of producing grid sequences suitable for use in multigrid. This strategy has the flexibility of being applied to any grid based smoother. Here it is shown to be an efficient tool for solving equations arising from the implicit time discretisation of the Euler equations.

To avoid problems due to negative cell volumes, the coarse grid operator is applied on the stationary time-averaged grid for which all cell volumes are positive. For small amplitudes of unsteadiness this is shown to be mathematically equivalent to a coarse grid operator defined on the deforming coarse grid.

Unsteady solutions are compared to experimental data for an oscillating 2D NACA0012 aerofoil. Further tests on an M6 wing and an aircraft configuration illustrate the applicability of the method to 3D and complex geometries.

Acknowledgements

This work was performed within Oxford Parallel with financial support from Rolls–Royce plc, DTI and SERC. We gratefully acknowledge the use of the unstructured grid generator of Jaime Peraire and Joaquim Peiro, and the visualisation software, Visual3 and pV3, of Bob Haimes.

References

[1] R.D. Rausch, J.T. Batina, and H.T.Y. Yang. Three–dimensional time–marching aeroelastic analysis using an unstructured–grid Euler method. *AIAA Journal*, 31(9), 1993.

[2] A. Jameson. Time dependent calculations using multigrid, with applications to unsteady flows past airfoils, wings, and helicopter rotors. AIAA Paper 91–1596, 1991.

[3] P.I. Crumpton and M.B. Giles. Aircraft computations using multigrid and an unstructured parallel library. AIAA Paper 95–0210, 1995.

[4] D. A. Burgess, P. I. Crumpton, and M. B. Giles. A parallel framework for unstructured mesh solvers. Proceedings of ECCOMAS Conference on CFD, 1994.

[5] P.I. Crumpton and M.B. Giles. OPlus programmer's manual. Oxford University Computing Laboratory, 1993.

[6] P.I. Crumpton. An efficient cell vertex method for unstructured tetrahedral grids. Proc. ICFD Conference in Oxford (To appear), 1995.

[7] D.J. Mavriplis. Three dimensional unstructured multigrid for the Euler equations. ICASE Report No. 91-41, 1991.

[8] V. Venkatakrishnan and D.J. Mavriplis. Agglomeration multigrid for the three–dimentional Euler equations. ICASE Report No. 94–5, 1994.

[9] T. Barth. Randomized multigrid. AIAA Paper 95–0207, 1995.

[10] E. Morano, H. Guillard, A. Dervieux, M.P. Leclercq, and Stoufflet B. Faster relaxations for non-structured multigrid with Voronoi coarsening. ECCOMAS 92, Vol. 1 pp. 69–74, 1992.

[11] R.H. Landon. Compendium of unsteady aerodynamic measurements. AGARD Report No. 702, 1983.

[12] M. Meister. Development of an implicit finite volume scheme for the computation of unsteady flow fields on unstructured moving grids. Proc. ICFD Conference in Oxford (To appear), 1995.

[13] R. Haimes. pV3: A distributed system for large scale unsteady CFD visualisation. AIAA Paper 94–0321, 1994.

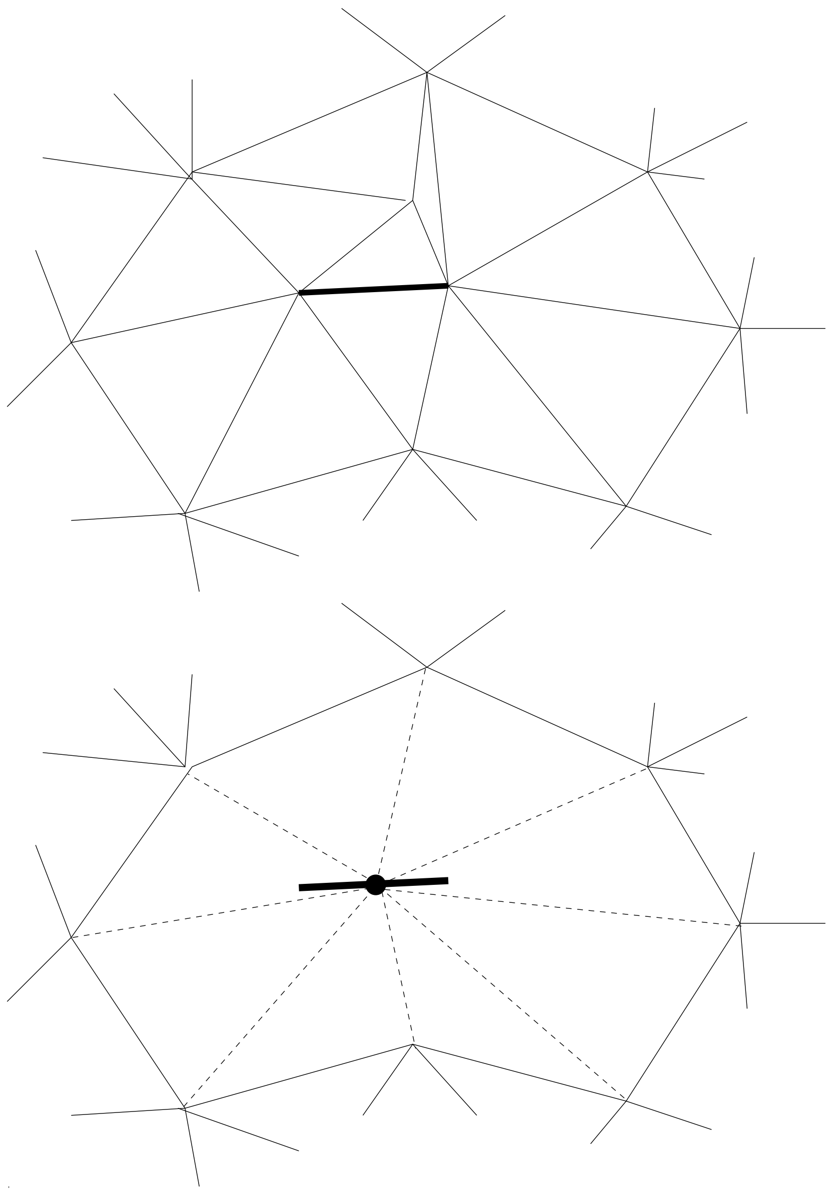

Figure 1: Edge collapsing procedure

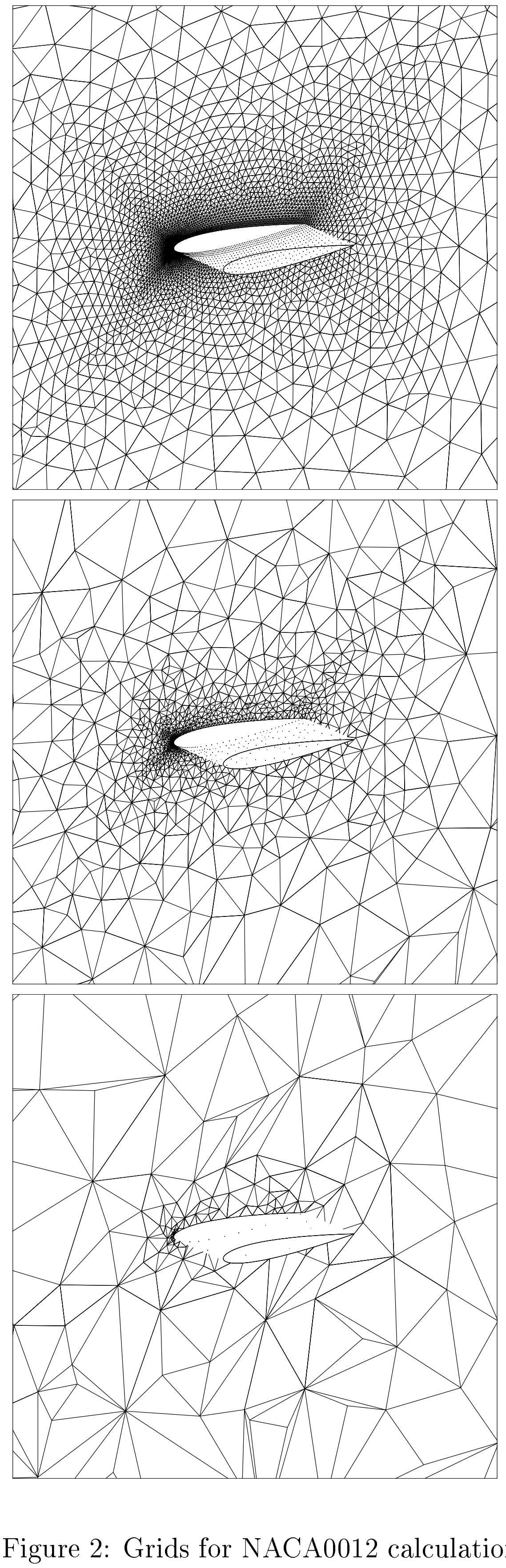

Figure 2: Grids for NACA0012 calculation

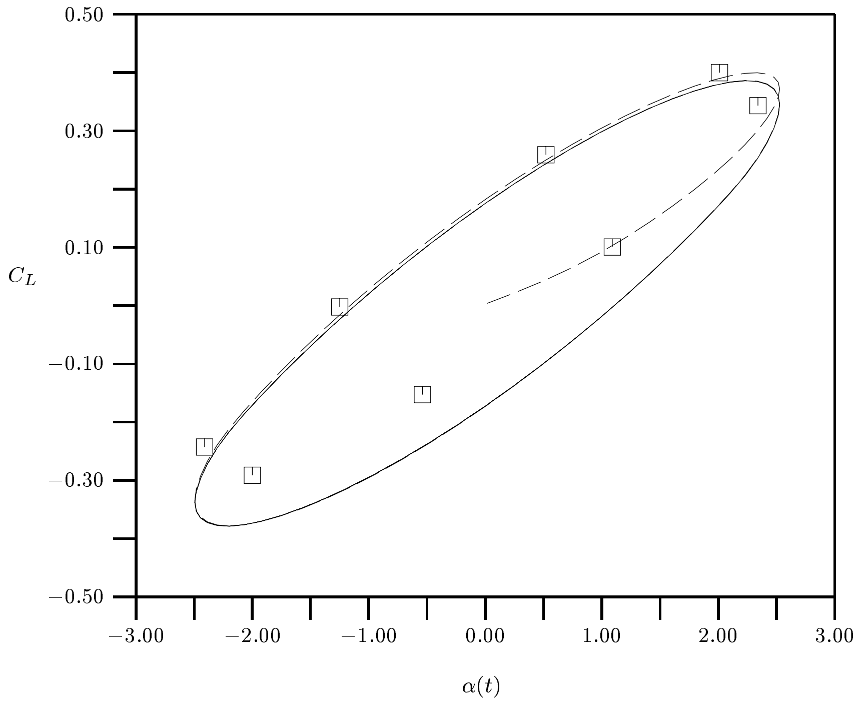

Figure 3: Lift variation for NACA0012 calculation

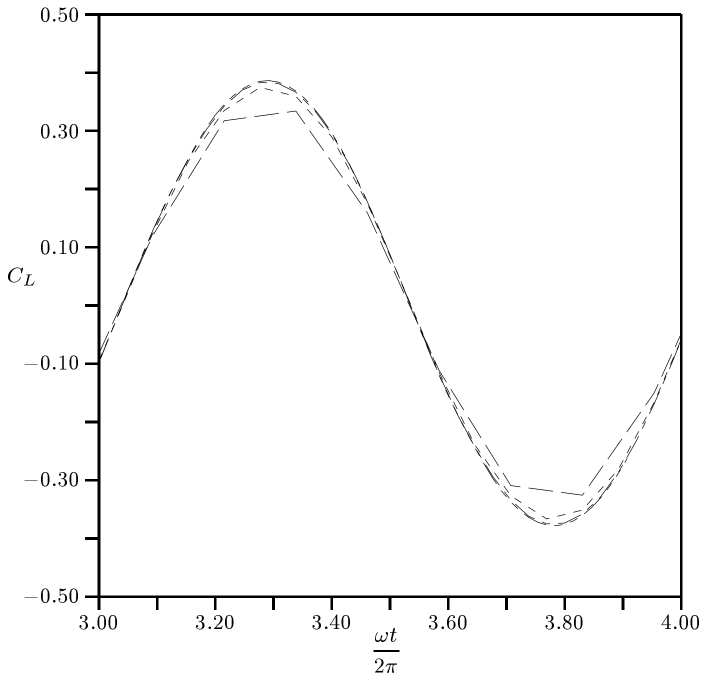

Figure 4: Unsteady lift variation for 8,16,32,64 timesteps per period.

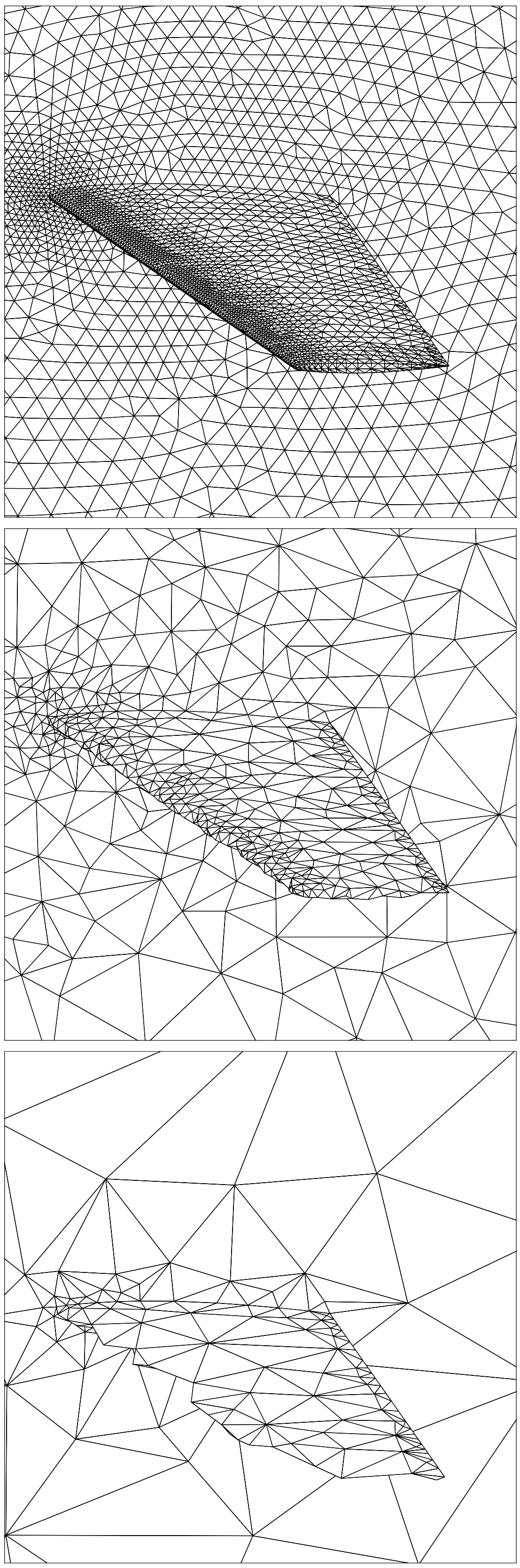

Figure 5: Grids for M6 wing calculation

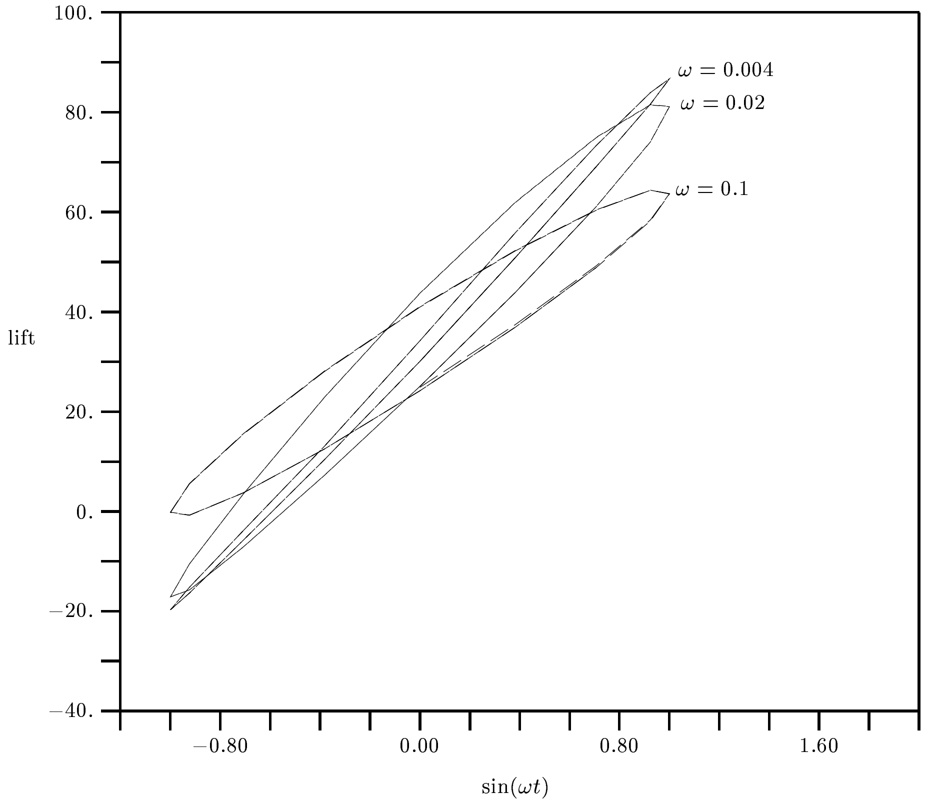

Figure 6: Lift variation for M6 wing calculations

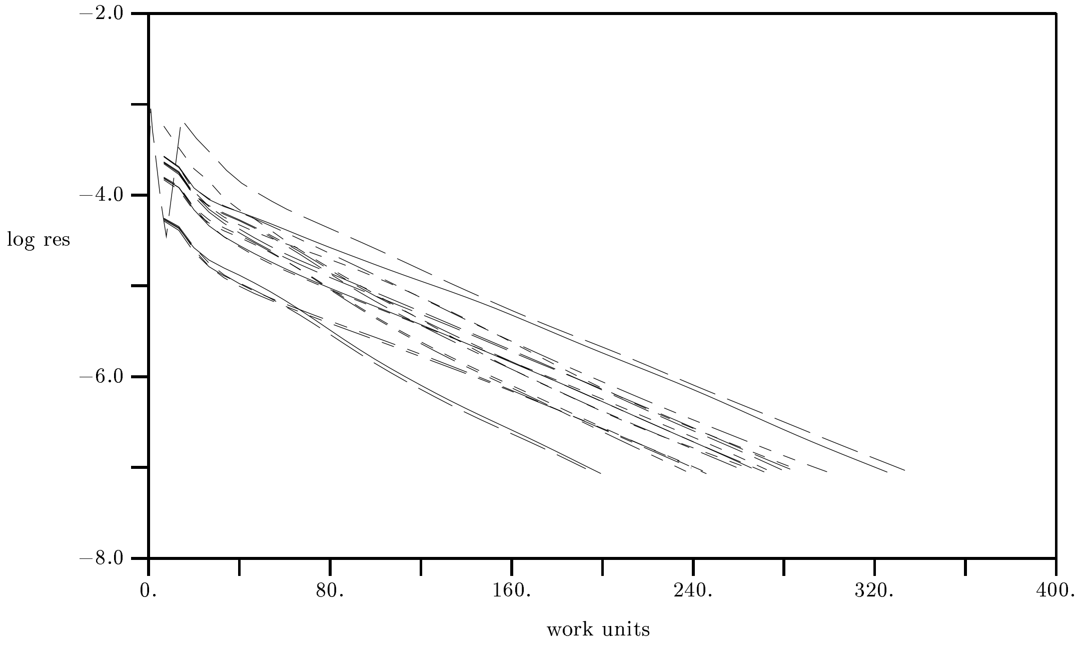

Figure 7: Convergence history for first 17 timesteps for M6 wing problem

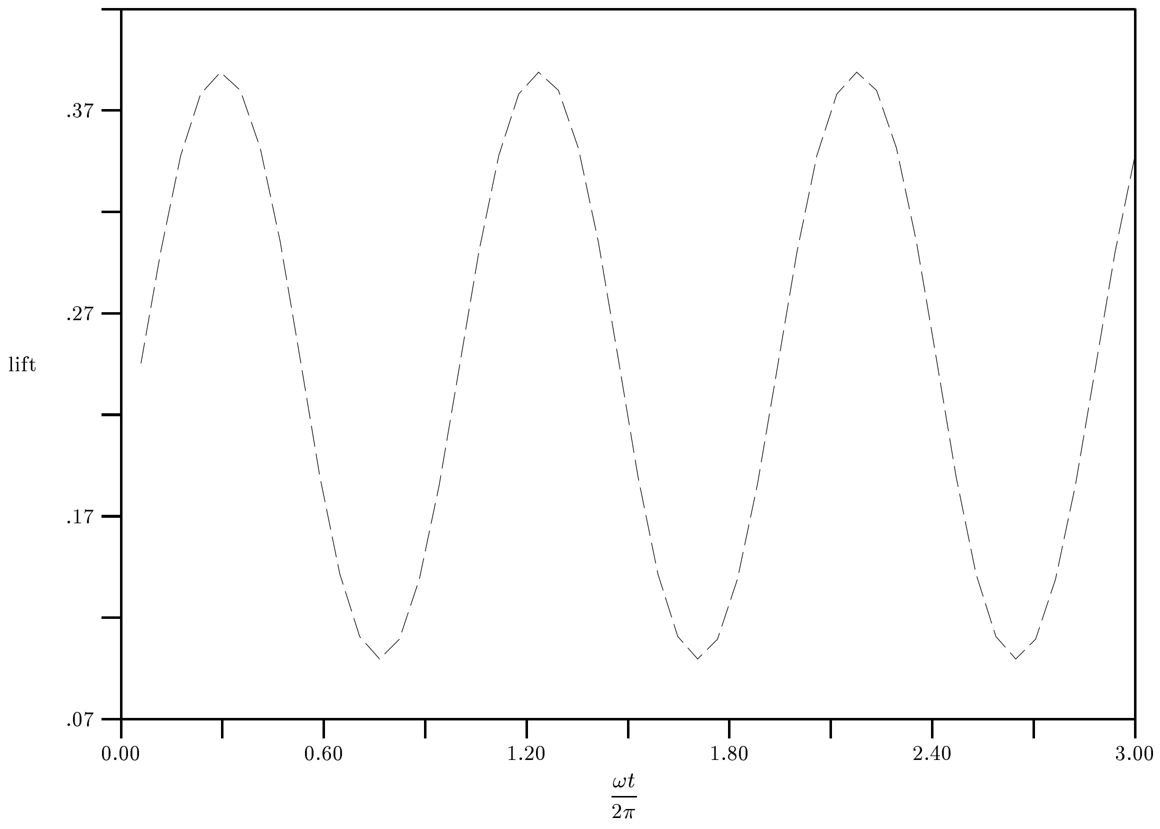

Figure 8: Unsteady lift for the aircraft configuration.

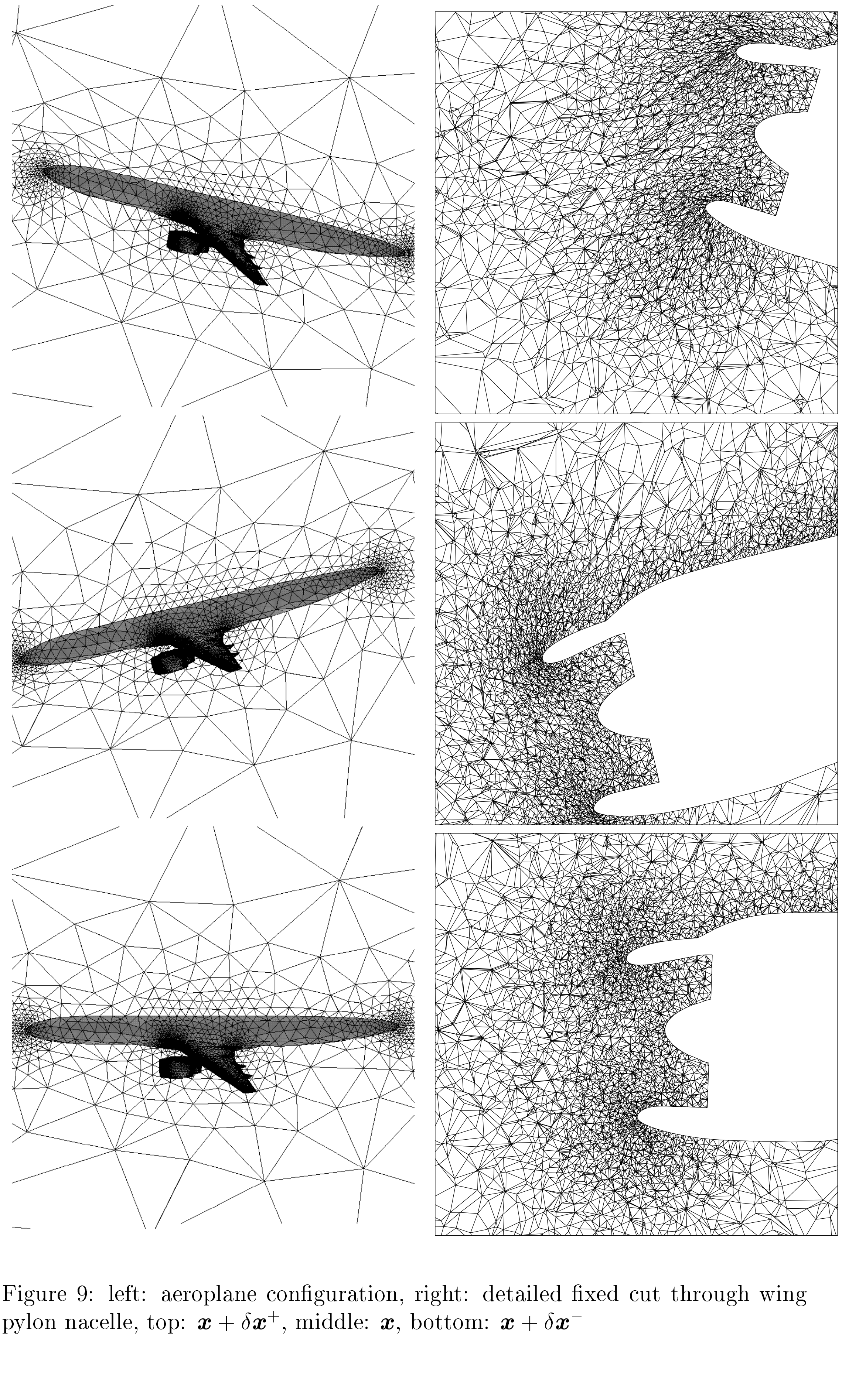

Figure 9: left: aeroplane configuration, right: detailed fixed cut through wing pylon nacelle, top: $\boldsymbol{x} + \delta \boldsymbol{x}^{+}$, middle: $\boldsymbol{x}$, bottom: $\boldsymbol{x} + \delta \boldsymbol{x}^{-}$

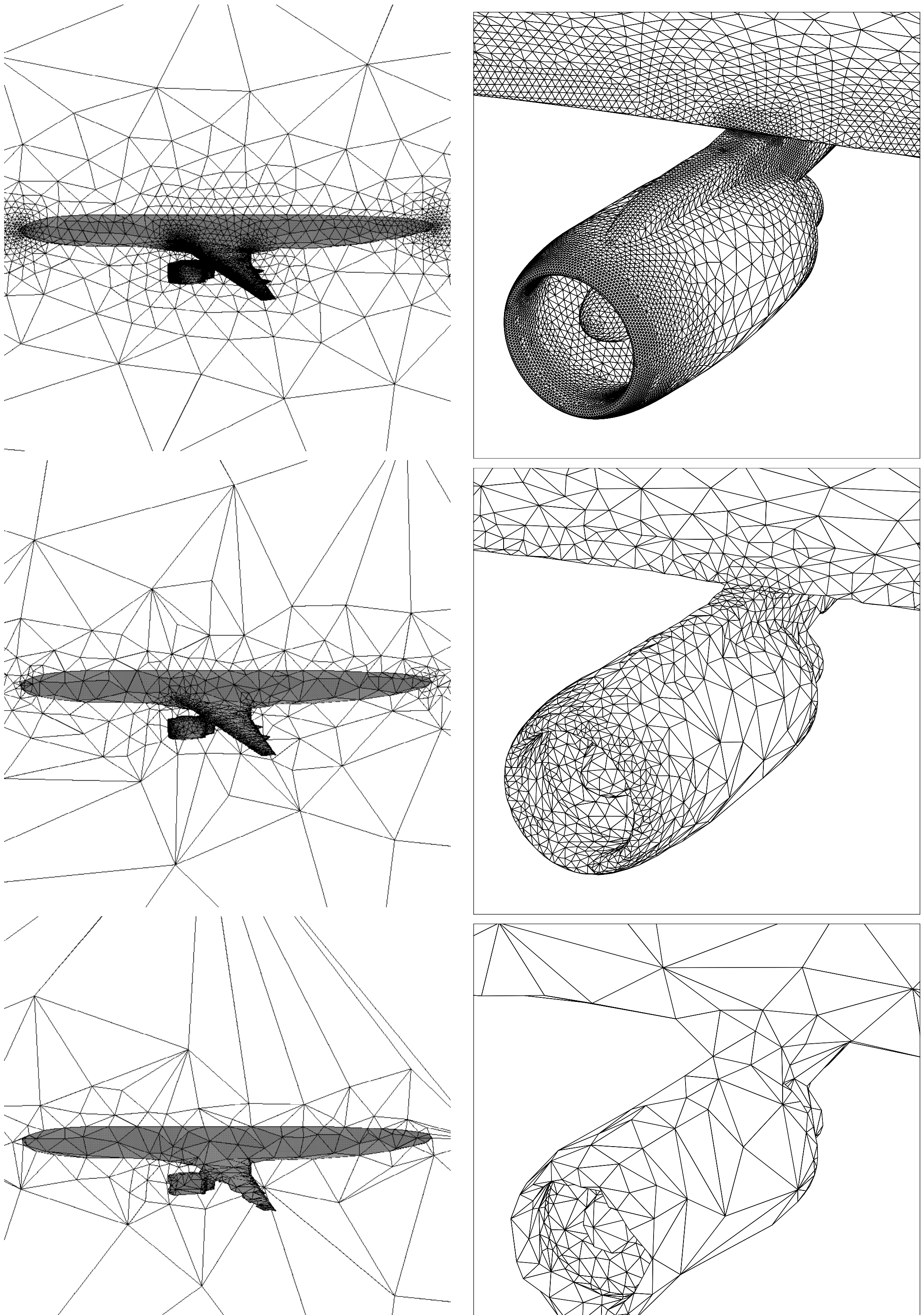
Figure 10: Sequence of grids for aircraft configuration