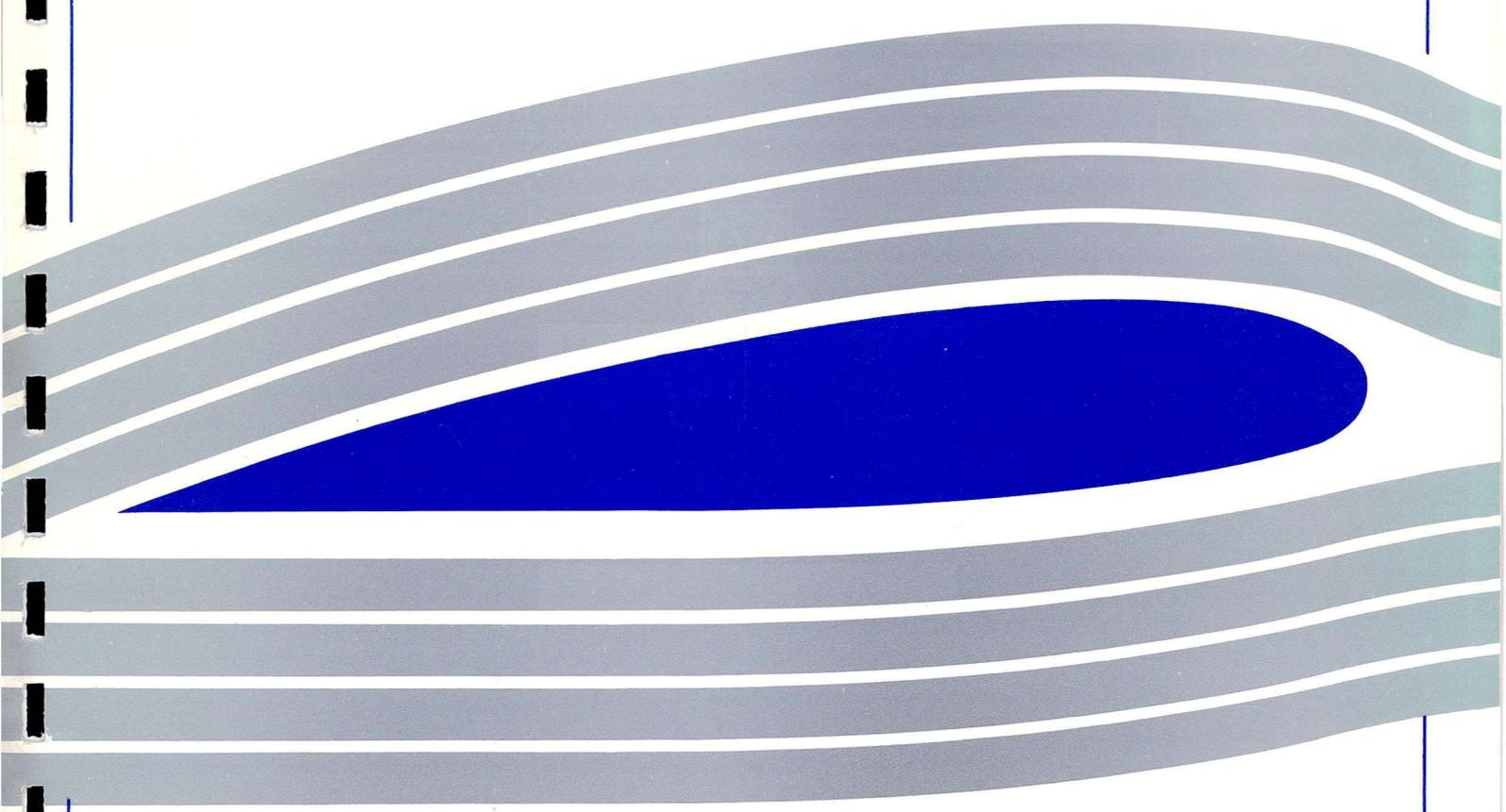




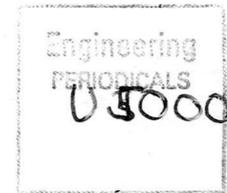
University of Glasgow  
DEPARTMENT OF  
**AEROSPACE  
ENGINEERING**

**Solution of the Euler Unsteady Equations  
Using Deforming Grids**

L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben,  
K.J. Badcock, and B.E. Richards







# Solution of the Euler Unsteady Equations Using Deforming Grids

L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben,  
K.J. Badcock, and B.E. Richards

Department of Aerospace Engineering,  
University of Glasgow,  
Glasgow, G12 8QQ, U.K.

Aero. Report 9704

February 20, 1997



2052

# Solution of the Euler Unsteady Equations Using Deforming Grids

L. Dubuc, F. Cantariti, M. Woodgate, B. Gribben,  
K.J. Badcock, and B.E. Richards

Department of Aerospace Engineering,  
University of Glasgow,  
Glasgow, G12 8QQ, U.K.

Aero. Report 9704

February 19, 1997

## 1 Introduction

Unsteady flows are at the centre of many engineering problems, such as helicopter aerodynamic, acoustic and fluid-structure interaction problems, and free surface problems. Owing to the cost of experiments, computational tools have an important role to play in this area. However, the application of CFD methods is very complex and its practical use for 3D unsteady problems, such as the rotor environment or full aircraft simulation, is unlikely in the near future. Nevertheless, progress are being made in this direction and studies of three-dimensional flows governed by the Reynolds-Averaged Navier-Stokes equations are being reported in the literature with increasing frequency.

An important issue in the development of a CFD tool aimed at practical engineering application is its capability of handling complex geometries. Over the years, improvement in numerical algorithms, along with increasing computing power, has led to a number of numerical methods with a sufficient level of maturity and reliability to make the solution of the Reynolds-Averaged Navier-Stokes equations possible for a wide range of flow problems. However, due to the great difficulty associated with the grid generation process for complex multi-component configurations, these methods are often restricted to relatively simple geometries, therefore not satisfying the industrial needs. At present, computational tools which combine good flexibility and generality in terms of complex geometries with accurate and efficient flow solvers are still rare.

The grid generation issue is of crucial importance when considering complex geometries. Amongst the most commonly used approaches are the structured grid approach and the unstructured grid approach. Structured grids allow easy implementation and calculation management, but have the disadvantage that grid generation is considerably more difficult for

complex geometries, as single block structured grids can only be created on domains which can be mapped onto rectangular parametric space. Alternatively, the unstructured grid approach is more flexible from the point of view of grid generation since no constraints exist on the point connectivity. This, however, results in a more complex code and data structure.

A useful compromise between the two approaches is provided by the multiblock technique [37, 42]. The multiblock grid consists of an unstructured arrangement of structured grids, where the generation of each structured block is made easier by the partitioning of the computational domain. The multiblock approach allows an extension of the methodologies developed for single block structured grids. The block decomposition also provides a natural partition of the problem for parallel processing. Examples in the literature of successful applications of multiblock methods for complex geometries can be found in [18, 25, 26, 31, 32, 35, 36, 39]. However, the generation of multiblock structured grids, although based on single block algorithms, is still a relatively difficult and time-consuming task, especially in 3D.

The grid deformation also becomes an important issue when solving unsteady problems where the mesh has to be deformed to conform to the instantaneous body shape. In most cases (e.g., single aerofoils), rigid body motions can easily be treated by moving the mesh rigidly in response to the motion of the body. However, this approach is no longer applicable if the body deforms as in an aeroelastic problem. The same holds if the outer boundaries of the mesh are fixed multiblock boundaries or if more complex deformations are considered, an example if this arises when there is relative motion of the different elements of a multi-component configuration. To tackle such problems, efficient grid regeneration and grid deformation techniques are required.

A number of methods exist for deforming the grid, e.g., methods based on a spring analogy [34] or elliptic smoothing, but they generally involve either a complete re-generation of the mesh around a deformed geometry or require the solution of a large system of equations for the displacements. This approach has efficiency drawbacks in the context of a rapid unsteady flow solver, for which the grid deformation process must be cheap and represent only a small fraction of the overall CPU time required by the flow solver.

Efficient remeshing techniques which do not require complete regeneration of the grid are rare. An algebraic technique based on the transfinite interpolation algorithm was used in [21] to deform the mesh at each step around a single aerofoil undergoing an oscillating pitching motion. The same strategy was also employed in the case of an oscillating trailing edge flap [20], and in [23] was extended to 3D for unsteady wing problems.

In the work presented here, a novel moving mesh technique allowing more general deformations around more complex geometries is described. The method is based on a TFI algorithm based on a multi-dimensional interpolation of the grid point displacements and is incorporated within a multi-block environment.

Another important issue for a practical CFD code is the efficiency of the method, especially for unsteady flow computations for which a large number of calculations is required to obtain

converged unsteady solutions. Explicit methods are simple to implement, but for many problems the allowable time step for stability is much smaller than that required for accuracy. For unsteady flows, where most of the acceleration techniques develop to speed up steady flow calculations cannot be used as they destroy time accuracy, this results in a very large number of time-steps to reach convergence. Implicit schemes, on the other hand, allow much larger time steps, but the work required per time step may be large, particularly in 3D. In [12], the progress made in the development of an implicit code capable of solving transonic turbulent flows for 2D aeroelastic problems was presented. The performance of the method, called AF-CGS, compared favourably with other existing codes using explicit multi-grid methods.

With the introduction by Jameson [30] of a dual-time approach, the use of explicit methods for unsteady flow computations has regained some popularity. The method uses an implicit real-time discretisation, but at each real time step marches the solution in pseudo-time to a steady state through an explicit time-marching scheme. The acceleration techniques of steady flow calculations can then be used, since the marching process is done in pseudo-time. This approach has led to considerable improvements compared to previous explicit methods based on single time-discretisation.

Although originally developed to accelerate standard explicit time-marching schemes, the dual-time method can be used in conjunction with implicit schemes for the solution of the steady-state problems in pseudo-time.

The general approach chosen by the CFD group at the University of Glasgow is to use high order upwind differencing schemes to provide accuracy and robustness and to use implicit methods to provide efficiency. Benefiting from the general experience gained through the development of 2D and 3D implicit steady and unsteady code [10, 12, 13, 17], a considerable effort is now being made to develop a generally applicable parallel multiblock (PMB) flow solver [9, 17].

The code is based on a cell-centred finite volume method and uses high-order upwind discretisation for the convective fluxes. Following the idea of Jameson, a dual-time approach is used to discretise the unsteady equations. The large sparse linear system which arises for the implicit time discretisation is then solved efficiently by using a Conjugate Gradient type method. The preconditioning strategy is a crucial factor in the efficiency and the success of such CG methods. In [12, 17], an Alternating Direction Implicit (ADI) factorisation was used as a preconditioner for the solution of the linear system at each iteration. This approach proved successful on a number of steady and unsteady aerofoil test problems [8, 12, 17]. In order to enhance the parallel efficiency of the method, which depends largely on the inter-block communication required by the solution algorithm, we use here a Block Incomplete Lower-Upper (BILU) factorisation for the preconditioner which allows decoupling between the blocks for the solution of the linear system. Previous results obtained with the parallel multiblock code have been reported in [9, 17, 28] for some standard aerofoil test cases and for some demonstration cases for complex 2D geometries. Encouraging results were obtained for all cases. We propose here to extend the application of the method to unsteady flow

problems for complex geometries by using a novel moving grid technique which allows rapid deformation of the grid .

Comparison between results obtained by the present method and experimental data will be shown for a number of pitching aerofoil test cases selected from the AGARD standard aeroelastic configurations. Results for a demonstration test case for the Williams aerofoil with an oscillating flap will also be presented.

## 2 Two-Dimensional Governing Equations

The two-dimensional Euler equations in Cartesian co-ordinates  $(x, y)$  can be written in non-dimensional conservative form as

$$\frac{\partial \mathbf{W}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = 0 \quad (1)$$

where  $\mathbf{W}$  denotes the vector of conservative variables,

$$\mathbf{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}$$

$\mathbf{F}$  and  $\mathbf{G}$  denote the convective fluxes,

$$\mathbf{F} = \begin{bmatrix} \rho U \\ \rho u U + p \\ \rho v U \\ U(\rho E + p) + x_t p \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \rho V \\ \rho u V \\ \rho v V + p \\ V(\rho E + p) + y_t p \end{bmatrix}$$

In the above equations,  $\rho$ ,  $p$  and  $E$  denote the density, the pressure and the specific total energy, respectively.  $u$  and  $v$  are the two components of the Cartesian velocity, and  $U$  and  $V$  are the contravariant velocities which are defined by

$$U = u - x_t, \quad V = v - y_t$$

where  $x_t$  and  $y_t$  are the grid speeds in the  $x$  and  $y$  directions respectively.

The equations are discretised using a cell-centred finite volume method which transforms the partial differential equations into a set of ordinary differential equations which can be written as

$$\frac{\partial}{\partial t}(V_{i,j} \mathbf{W}_{i,j}) + \mathbf{R}_{i,j}(\mathbf{W}) = 0 \quad (2)$$

where  $t$  is the time,  $W_{i,j}$  is the vector of conservative variables,  $V_{i,j}$  is the control volume and  $R_{i,j}(W)$  is the flux residual for the cell  $(i, j)$  which contains all the terms arising from the spatial discretisation. The convective fluxes are discretised using Osher upwind Finite Difference Splitting scheme together with a MUSCL variable interpolation to provide second or third order accuracy in space. The Von Albada limiter is used to ensure monotonic solutions around shock waves. Central differencing is employed for the discretisation of the viscous terms. Far-field boundary conditions are treated by the characteristic boundary method based on the Riemann invariants.

### 3 Implicit Dual-Time Method

The original implicit dual-time approach was introduced by Jameson [30] and allows an implicit discretisation to be used in real time with the solution at the new time level being obtained through an iteration in pseudo-time. This permits the acceleration techniques of steady flow, such as local time stepping, residual smoothing, multigrid or implicit methods to be used to obtain the updated solution. This also allows the real time step to be chosen based on accuracy requirements alone without stability restrictions. The dual-time method can be obtained from a steady Euler solver by means of a few modifications and is therefore very attractive when extending steady flow methodologies to unsteady problems. The method has been used mostly in conjunction with multigrid algorithms and applied to both the Euler [15, 30] and Navier-Stokes equations [4], on structured and unstructured grids, and using either rigid grids or general mesh deformation algorithms [20].

In this work, we propose to apply the dual-time method in conjunction with an implicit time-stepping method for the solution of the steady-state problem in pseudo-time. In [12] implicit time stepping was used without pseudo-time iterations. However, the solutions obtained by this method, which uses only a single time discretisation, are only first order accurate in time. By using a dual-time approach with pseudo-time iterations it is possible to improve the time accuracy to second order.

Time accuracy can be further enhanced by using a third-order implicit time discretisation rather than the second-order discretisation used here, as shown by [29]. However, this is achieved at the expense of additional storage for an extra time level of accuracy, which is not always necessary considering the reasonably good accuracy obtained in general with second-order time discretisation.

We consider here the unsteady governing equations written in discrete form as shown in equation (2). This set of equation is then discretised in time by using a fully implicit time discretisation (in real time) to give

$$\frac{d}{dt}(V_{i,j}^{n+1}W_{i,j}^{n+1}) + R_{i,j}(W^{n+1}) = 0, \quad (3)$$

where the superscript  $n+1$  denotes the time level  $(n+1)\Delta t$  of the approximation (in real time). Following Jameson [30], the time derivative is approximated by a second-order backward

difference discretisation, so equation (3) becomes

$$\frac{3V_{i,j}^{n+1}W_{i,j}^{n+1} - 4V_{i,j}^nW_{i,j}^n + V_{i,j}^{n-1}W_{i,j}^{n-1}}{2\Delta t} + R_{i,j}(W^{n+1}) = 0, \quad (4)$$

This equation for  $W_{i,j}^{n+1}$  is non-linear and therefore cannot be solved analytically. At this stage, it is convenient to redefine a new residual  $R^*$ , referred to as unsteady residual and defined by :

$$R_{i,j}^*(W^{n+1}) = \frac{3V_{i,j}^{n+1}W_{i,j}^{n+1} - 4V_{i,j}^nW_{i,j}^n + V_{i,j}^{n-1}W_{i,j}^{n-1}}{2\Delta t} + R_{i,j}(W^{n+1}) = 0 \quad (5)$$

This new equation can be seen as the solution of a steady state problem which can then be solved with a time-marching method by introducing a derivative with respect to a fictitious pseudo-time  $t^*$ ,

$$\frac{\partial W_{i,j}^{n+1}}{\partial t^*} + \frac{1}{V_{i,j}^{n+1}}R_{i,j}^*(W^{n+1}) = 0 \quad (6)$$

The steady state solution to equation (6) satisfies

$$\frac{\partial W_{i,j}^{n+1}}{\partial t^*} = 0 \quad (7)$$

which means it also satisfies  $R_{i,j}^*(W^{n+1}) = 0$  and hence is also the solution of the unsteady equation (5). The pseudo-time problem can then be solved by using any time-marching method designed to solve steady-state problems, utilising any of the standard acceleration techniques.

## 4 Implicit Unfactored Method

The steady state problem in pseudo-time to be solved at each global time step is

$$\frac{\partial W_{i,j}}{\partial t^*} + \frac{1}{V_{i,j}}R_{i,j}^*(W) = 0 \quad (8)$$

where  $t^*$  is the pseudo-time,  $W_{i,j}$  is the approximation to  $W_{i,j}^{n+1}$  and  $R_{i,j}^*(W)$  is the unsteady residual as defined by equation (5) and which contains the terms arising from the spatial discretisation and those arising from the implicit time discretisation,

$$R_{i,j}^*(W) = \frac{3V_{i,j}W_{i,j}}{2\Delta t} - \frac{2V_{i,j}^nW_{i,j}^n}{\Delta t} + \frac{V_{i,j}^{n-1}W_{i,j}^{n-1}}{2\Delta t} + R_{i,j}(W). \quad (9)$$

Using an implicit time discretisation on the pseudo-time  $t^*$ , equation (8) becomes

$$\frac{\Delta W_{i,j}}{\Delta t^*} = \frac{W_{i,j}^{m+1} - W_{i,j}^m}{\Delta t^*} = -\frac{1}{V_{i,j}}R_{i,j}^*(W^{m+1}) \quad (10)$$

where the superscript  $m+1$  denotes the time level  $(m+1)\Delta t^*$  (in pseudo-time). In this equation, the flux residual on the right hand side is evaluated at the new time level  $(m+1)$

and is therefore expressed in terms of the unknown solution at this new time level. In order to get an expression in terms of known quantities only (i.e., solution at the previous time levels), the term  $R^*(W^{m+1})$  is linearised with respect to the time variable  $t^*$  :

$$\begin{aligned} R^*(W^{m+1}) &\approx R^*(W^m) + \frac{\partial R^*}{\partial t} \Delta t^* \\ &\approx R^*(W^m) + \frac{\partial R^*}{\partial W} \frac{\partial W}{\partial t^*} \Delta t^* \\ &\approx R^*(W^m) + \frac{\partial R^*}{\partial W} \Delta W \end{aligned}$$

where  $\Delta W = W^{m+1} - W^m$  and, according to equation (9),

$$\frac{\partial R^*}{\partial W} = \frac{\partial R}{\partial W} + \frac{3V}{2\Delta t} \mathbf{I} \quad (11)$$

Substituting the above in equation (10), one implicit pseudo time step can be written as:

$$\left[ \left( \frac{V}{\Delta t^*} + \frac{3V}{2\Delta t} \right) \mathbf{I} + \frac{\partial R}{\partial W} \right] \Delta W = -R^*(W^m) \quad (12)$$

Equation (12) can now be rewritten in terms of the vector of primitive variables  $\mathbf{P}$  and in terms of the flux residual components  $R_\xi, R_\eta$  to give:

$$\left[ \left( \frac{V}{\Delta t^*} + \frac{3V}{2\Delta t} \right) \frac{\partial W}{\partial \mathbf{P}} + \frac{\partial R_\xi}{\partial \mathbf{P}} + \frac{\partial R_\eta}{\partial \mathbf{P}} \right] \Delta \mathbf{P} = -R^*(W^m), \quad (13)$$

where the terms  $\frac{\partial R_\xi}{\partial \mathbf{P}}$  and  $\frac{\partial R_\eta}{\partial \mathbf{P}}$  denote the flux Jacobians associated with the discretised flux in the  $\xi$  and  $\eta$  direction respectively.

Equation (13) is the unfactored linear system which arises from the implicit time discretisation in pseudo-time. This linear system is similar to that obtained for a standard steady state problem (without dual-time) and can therefore be solved using the same method as that used for steady state problems. The difference between the linear system given by equation (13) and that associated with a standard steady state problem lies first in the definition of the flux residual on the right hand side of equation (13) which not only contains the terms arising from the spatial discretisation of the fluxes but also the terms arising from the discretisation of the time derivative, and secondly in the presence of an extra term  $3V/2\Delta t$  on the diagonal of the Jacobian matrix on the left hand side of equation (13). With these two modifications in the definition of the linear system, the problem can be solved by following the same strategy as that normally used for a standard steady state problem in real time [6, 9, 11].

## 5 Resolution of the Unfactored Linear System

If the equation for the updates is written in the form

$$Ax = b \quad (14)$$

where

$$\mathbf{A} = \left[ \left( \frac{V}{\Delta t^*} + \frac{3V}{2\Delta t} \right) \frac{\partial \mathbf{W}}{\partial \mathbf{P}} + \frac{\partial \mathbf{R}_\xi}{\partial \mathbf{P}} + \frac{\partial \mathbf{R}_\zeta}{\partial \mathbf{P}} \right] \quad (15)$$

$$\mathbf{b} = \frac{3V\mathbf{W}}{2\Delta t} - \frac{2V^n\mathbf{W}^n}{\Delta t} + \frac{V^{n-1}\mathbf{W}^{n-1}}{2\Delta t} + \mathbf{R}_\xi + \mathbf{R}_\eta \quad (16)$$

then a conjugate gradient method can be used to solve the preconditioned system

$$\mathbf{C}^{-1}\mathbf{A}\mathbf{x} = \mathbf{C}^{-1}\mathbf{b}, \quad (17)$$

where  $\mathbf{C}^{-1} \approx \mathbf{A}^{-1}$  is called the preconditioner. If  $\mathbf{C}^{-1}$  closely approximates  $\mathbf{A}^{-1}$  then the conjugate gradient method will converge quickly. However, the effort and the cost to calculate  $\mathbf{C}^{-1}$  must be minimised for overall efficiency.

In the present work, we use the Generalised Conjugate Gradient (GCG) method to solve the system (17) as described in [5]. The GCG method has been written to minimise the number of matrix vector and preconditioner vector multiplications at the expense of some extra vector operations. The preconditioning strategy is based on a Block Incomplete Lower-Upper (BILU) factorisation of the Jacobian matrix with the sparsity pattern of L and U defined with respect to the sparsity of the unfactored matrix  $\mathbf{A}$ . In order to minimise the solution coupling between the blocks in a multiblock grid, the BILU factorisation is decoupled between the blocks [9, 14].

For the high order spatial reconstruction schemes, the left hand side (LHS) matrix is constructed using terms associated with a first order spatial scheme only, while using a higher order right hand side (RHS). This not only has the advantage of reducing the storage from nine blocks per cell down to five blocks when high order MUSCL extrapolation is used, it also has the effect of reducing greatly the number of GCG steps per iteration, and thereby increases the overall efficiency of the method [16].

The present implicit method, also implemented in parallel, has proved to be very efficient, comparing favourably with other explicit (with multigrid) or implicit techniques.

## 6 Geometric Conservation Law

When computing the flow on a moving grid, the cell areas also vary in time and it is therefore important to discretise the time-dependent metrics carefully in order to maintain the conservative properties of the scheme. If the cell areas are calculated analytically in terms of the grid node positions, numerical errors will be introduced in the solution algorithm which will increase with time. To avoid such numerical errors, the cell areas must be integrated forward in time by using the same method as that used to solve the physical conservation laws [2, 3, 19, 33, 41]. This is achieved by introducing a Geometric Conservation Law (GCL) which is derived from the continuity equation and reads,

$$\frac{\partial}{\partial t} \int_{\Omega} dV - \oint_{\partial\Omega} \mathbf{v} \cdot \mathbf{n} d\Sigma = 0 \quad (18)$$

where  $V$  is the cell area,  $v$  is the grid speed,  $n$  is the normal area vector and  $\partial\Sigma$  is the boundary surface of the control volume  $\Omega$ . Using the same second-order time discretisation as for the flow equation, equations (18) becomes

$$\frac{3V_{i,j}^{n+1} - 4V_{i,j}^n + V_{i,j}^{n-1}}{2\Delta t} - \oint_{\partial\Sigma} v \cdot n d\Sigma = 0 \quad (19)$$

The above geometric conservation law must be satisfied at all time on a moving grid. This law states that the change in area of each control volume between  $t^n$  and  $t^{n+1}$  must be equal to the area swept by the cell boundary during  $\Delta t = t^{n+1} - t^n$ . The volume  $V_{i,j}^{n+1}$  at the new time step can then be computed by

$$V_{i,j}^{n+1} = \frac{4V_{i,j}^n}{3} - \frac{V_{i,j}^{n-1}}{3} + \frac{2\Delta t}{3} [(\xi_t)_{i+1/2,j} - (\xi_t)_{i-1/2,j} + (\eta_t)_{i,j+1/2} - (\eta_t)_{i,j-1/2}] \quad (20)$$

where

$$\begin{aligned} \xi_t &= -(\xi_x x_t + \xi_y y_t) \\ \eta_t &= -(\eta_x x_t + \eta_y y_t) \end{aligned}$$

Note that this is an explicit equation for  $V_{i,j}^{n+1}$  since the terms  $\xi_t$  and  $\eta_t$  are prescribed from the node values. Using the GCL to calculate the volumes rather than calculating them geometrically (or analytically) ensures that large errors are not encountered when solving the physical conservation laws. This formally introduces an error into the values obtained for the volumes, but this is small in comparison with the numerical errors in the solution procedure [2, 3, 19, 33, 41]. The GCL needs to be evaluated only once every global time step to calculate the new cell areas.

## 7 Deforming Grid Algorithm for Multi-block Applications

For unsteady computations, the mesh must be deformed once per (real) time step. Therefore, a fast and efficient way of deforming the grid at each time level of the calculation is required. For simple geometries and simple deformation, such as single aerofoils in oscillating pitching motion, the grids can be rigidly rotated with the aerofoil. However, for more complex geometries, such as multi-component aerofoils where the relative motion of the different components must be taken into account, it is necessary to regenerate the grid.

In a multi-block approach, the flow domain is split up into blocks of simple shapes and structured grids are generated in each block with grids in adjacent blocks being matched at common interfaces. With such an approach, the generation of the grid in each block is made easier and the overall quality of the mesh is improved.

However, the generation of a good quality multi-block grid still remains a very time consuming task, and it is in general easier to deform the initial grid rather than regenerate a complete new grid at each time step in order to account for the motion and the deformation of the configuration. In doing so, the grid deformation process is completely independent of

the generation of the initial grid, for which any suitable technique can be used. As well as being fast and simple, an important feature of the deforming grid procedure is that it must maintain the overall quality of the initial grid by introducing minimal distortion around the aerofoil surface. It is important to keep the grid as rigid as possible in the near wall regions by introducing cell distortion towards the far field, or at least in the regions of low gradients where the flow is not changing rapidly.

Also, for multi-block meshes, the complete mesh does not need to be deformed at each time level, but only the mesh blocks embedding the moving surfaces. In some cases, additional blocks, not directly adjacent to the moving surfaces, may be deformed in order to obtain smoother grids after deformation.

Methods based on the algebraic interpolation of the grid displacements are very attractive as they generally preserve the overall quality of the initial grid as well as being cheap to calculate and easy to formulate. Such methods, also referred to as perturbation methods, have been presented in the literature. In [22, 24], application of a moving grid algorithm using grid displacements was described for single element aerofoils in oscillating pitching motion, where only one boundary was allowed to move. In that case, the interpolation of the displacements across the grid can be performed in one direction only, between the moving solid boundary (aerofoil or wing surface) and the fixed far-field boundary, by using an appropriate interpolation function. The choice of adequate control parameters for different types of grids, such as Euler or Navier-Stokes grids, was also addressed therein. However, the direction of interpolation is an additional constraint which is not desirable here when extending the application of the method to multi-block configurations where the block topology and the orientation of the blocks can arise randomly. An extension of the method allowing four moving boundaries and interpolation in each direction is therefore necessary in order to provide sufficient flexibility and sufficient generality. Also, due to the complexity in the block connectivity, and in order to avoid a sequential procedure with unnecessary communication between blocks, it is preferable to employ a method which can treat all the blocks independently.

Amongst the several approaches which were investigated, a method based on the interpolation of the block corner displacements appeared to be the most suitable here. If the displacements of the block corners are known, the displacements of the block faces, and then of the interior points within each block, can be interpolated. By using the same interpolation procedure at all block boundary interfaces, the perfect matching of the block boundaries is guaranteed even if the blocks are treated independently.

The technique employed here to deform the grid in each block is a modified version of the transfinite interpolation method (TFI) based on the grid displacements (see figure 1). The TFI grid generation algorithm is a very popular algebraic grid generation technique which effectively interpolates grid points in the computational domain from prescribed points along the block boundaries. The algorithm can equally be applied to the grid point displacements by interpolating the displacements across the grid from the prescribed or calculated displacements along the block boundaries.

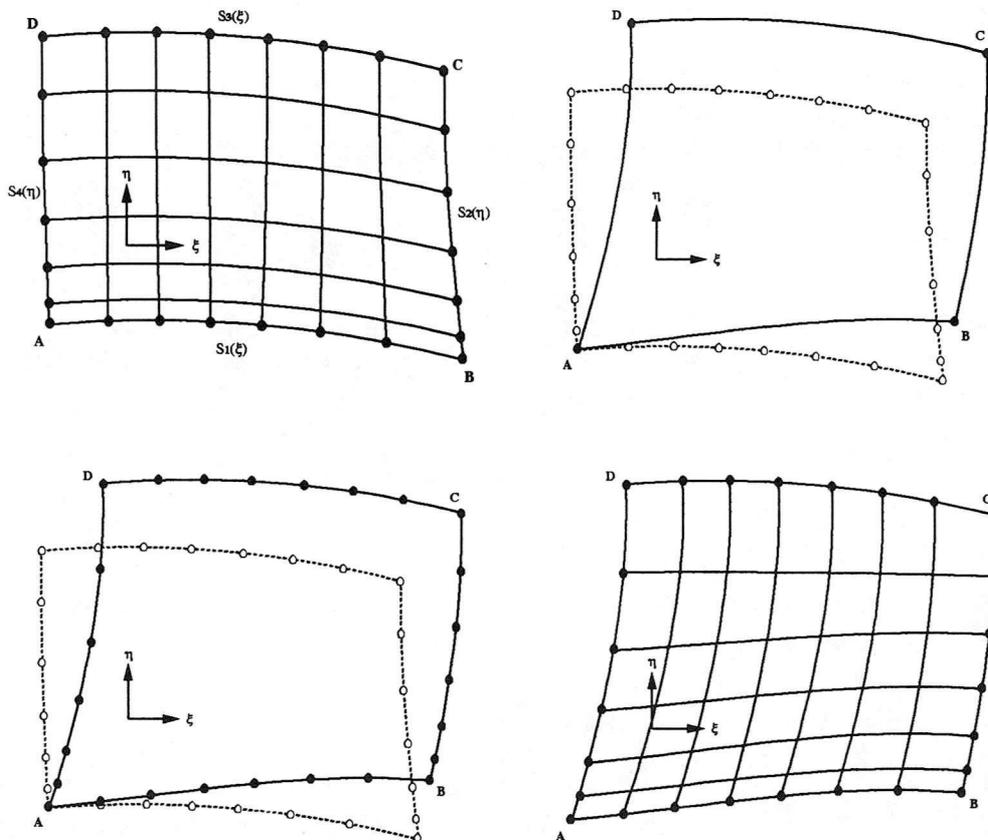


Figure 1: TFI Method of Perturbations

## 8 TFI Method of Perturbations

### 8.1 Displacement of the Block Corners

We first need to determine the displacements of the four block corners (or block vertices). In order to identify a moving block from a fixed block, we introduce a new parameter MOVE in the grid file which is set to one for each moving block and to zero for all fixed blocks. For each block corner, a search is made over its neighbours, and if at least one of the neighbouring blocks surrounding this corner point (i.e., all blocks having this point as a vertex) is fixed (i.e., block flagged with MOVE=0), then no displacement is allowed for this point. Otherwise, the corner point is moved according to the motion of the solid surface. The displacement of all points lying on a moving surface is assumed to be known. In the present work, we consider only rigid motions for oscillating pitching aerofoils and oscillating flaps, but the application of the method can be easily extended to more complex and more general deformation. ▶

## 8.2 Displacement of the Block Faces

The displacements of the four corner points are then used to interpolate the displacement of all the points along the block boundary. We denote by  $x$  and  $dx$  the position vector and displacement vector respectively associated to the grid points of the mesh,

$$x = \begin{bmatrix} x(\xi, \eta) \\ y(\xi, \eta) \end{bmatrix}, dx = \begin{bmatrix} dx(\xi, \eta) \\ dy(\xi, \eta) \end{bmatrix}$$

Let A and B be the two end-points of a block boundary with respective displacements denoted by  $dx_A$  and  $dx_B$  respectively. The displacement  $dx$  of any point P along this boundary can then be obtained by the following weighted formula as :

$$dx = \left(1 - \frac{a}{c}\right) dx_A + \left(1 - \frac{b}{c}\right) dx_B$$

where  $a = \|\vec{AP}\|$ ,  $b = \|\vec{BP}\|$  and  $c = \|\vec{AB}\|$ . Here, the distances are calculated with the previous grid point coordinates. If both end-points are fixed (i.e., zero displacement), then the whole boundary face remains fixed.

## 8.3 Displacement of the Interior Points

Following the original formulation of the TFI algorithm described by Gordon and Hall [27], the general transfinite interpolation method results in a recursive algorithm which is here applied to the grid point displacements:

$$dx(\xi, \eta) = f_1(\xi, \eta) + \phi_1^0(\eta) [dx_{b1}(\xi) - f_1(\xi, 0)] \\ + \phi_2^0(\eta) [dx_{b3}(\xi) - f_1(\xi, 1)]$$

where

$$f_1(\xi, \eta) = \psi_1^0(\xi) dx_{b4}(\eta) + \psi_2^0(\xi) dx_{b2}(\eta)$$

and  $dx_{b1}, dx_{b2}, dx_{b3}$ , and  $dx_{b4}$  are the interpolated displacements along the four boundary faces. The functions  $\psi$  and  $\phi$  are the blending functions in the  $\xi$  and  $\eta$  directions respectively. These functions are given by the grid point distributions along each block faces as

$$\psi_1^0(\xi) = 1 - s_1(\xi) \\ \psi_2^0(\xi) = s_3(\xi) \\ \phi_1^0(\eta) = 1 - s_4(\eta) \\ \phi_2^0(\eta) = s_2(\eta)$$

Case	Aerofoil	$M_\infty$	$\alpha_m$	$\alpha_0$	$k$	$x_m$
CT1	NACA 0012	0.6	$2.89^\circ$	$2.41^\circ$	0.0808	0.273
CT2	NACA 0012	0.6	$3.16^\circ$	$4.59^\circ$	0.0811	0.273
CT5	NACA 0012	0.755	$0.016^\circ$	$2.51^\circ$	0.0814	0.25

Table 1: AGARD test cases examined in this report

where  $s_1(\xi)$  is the stretching function on the block face  $\eta = 0$ ,  $s_2(\eta)$  on the block face  $\xi = 1$ ,  $s_3(\xi)$  on the block face  $\eta = 1$ ,  $s_4(\eta)$  on the block face  $\xi = 0$ . The coordinates of the new grid point are then simply obtained by

$$x(\xi, \eta) = x_0(\xi, \eta) + dx(\xi, \eta)$$

where  $dx$  is the interpolated displacement and  $x_0$  is the vector position for the initial undisturbed grid.

## 9 Results for NACA0012 Aerofoil

### 9.1 Description of the Test Cases

In this report, we present some results which have been obtained using the dual-time method for a series of standard pitching aerofoil test cases selected from the AGARD database [1]. The different test cases examined in this report for the NACA0012 aerofoil are listed in table 1. For these test cases, the periodic motion of the aerofoil is defined by the angle of attack as a function of time as

$$\alpha(t) = \alpha_m + \alpha_0 \sin(\omega t)$$

where  $\alpha_m$  is the mean incidence,  $\alpha_0$  is the amplitude of the pitching oscillation and  $\omega$  is the angular frequency of the motion which is related to the reduced frequency  $k$  by

$$k = \frac{\omega c}{2U_\infty}$$

where  $c$  is the aerofoil chord and  $U_\infty$  is the freestream velocity. For all cases, the aerofoil oscillates about its quarter chord. In table 1,  $M_\infty$  is the free stream Mach number and  $x_m$  is the location of the moment center about which the pitching moment is calculated.

All the results given in this report for the NACA0012 aerofoil, except those related to the grid refinement study, were obtained with the same grid. The grid used here is an C-type Euler grid which consists of  $129 \times 33$  grid points with 97 points on the aerofoil, the far-field boundary being situated at approximately 15 chords from the aerofoil surface. The average spacing for the first layer of points from the aerofoil surface varies between 0.0027 at the leading edge and 0.010 at the trailing edge.

Case	Grid handling Technique	GCL	Number of pseudo time steps per cycle	CPU/cycle (in Work Units)
CT1	Rotation	No	505	2116
	Deformation	No	507	2030
	Deformation	Yes	503	2091
CT2	Rotation	No	666	2835
	Deformation	No	670	2706
	Deformation	Yes	667	2801
CT5	Rotation	No	622	2639
	Deformation	No	621	2815
	Deformation	Yes	617	2891

Table 2: Effect of grid deformation on the efficiency of the method

For all unsteady calculations, an initial solution is first obtained by solving for the steady flow at the mean incidence, before setting the aerofoil in motion to solve the unsteady flow equations. The initial solution at each time step is obtained by linear extrapolation from the last two solutions at the previous time steps. At each global time step, the solution is marched in pseudo-time until some specified tolerance for the norm of the unsteady residual is reached. Here, a reduction of three orders of magnitude was found to be sufficient to achieve a converged solution at each global time step. No gain in accuracy for the solution in terms of normal force and moment coefficients was noticed when using a higher degree of convergence.

## 9.2 Effect of Grid Deformation

We first examine the effect on the solution accuracy and the performance of the method when using grid deformation compared to grid rotation and also the effect of using the Geometric Conservation Law to account for area changes.

Comparison is made for the three test cases CT1, CT2 and CT5 and results are shown in figures 3, 4 and 5 respectively in terms of normal force and pitching moment coefficients. The results were obtained on a  $128 \times 32$  grid consisting of three blocks, where, in the case of grid deformation, all blocks are deforming. The results were obtained here with 80 steps per cycle. No significant differences can be noticed for the normal force coefficient, while the loop for the moment coefficient shows a slight sensitivity in the results between the two grid handling techniques, probably caused by a slightly different location of the shock wave. It is also interesting to note that the use of the Geometric Conservation Law does not show any improvement of the results for all these cases.

Table 2 shows a comparison of the number of implicit iterations and CPU time (expressed in terms of Work Units) required to compute one complete cycle in each case. Here again, no significant effect is observed between the results obtained on a rigidly rotating grid and those

obtained on a deforming grid, with or without GCL.

Additional computations were carried out to further investigate the effect of grid deformation and grid distortion. Two identical grids consisting of six blocks (two layers of three blocks wrapped around the aerofoil) were considered, with a different number of moving blocks, either six or three.

When all six blocks are deforming, the three inner blocks nearer to the aerofoil surface are mainly rotated rigidly with the aerofoil, therefore creating deformation and distortion only in the outer blocks, away from the aerofoil surface. On the other hand, when only the three inner blocks are allowed to deform, the deformation of the grid had to be absorbed over a smaller region, creating larger distortion in the near-aerofoil region.

However, results obtained in that case also showed that the deformation of the grid had no significant effect on the solution accuracy and on the overall efficiency of the method compared to the results obtained on a rigidly rotating grid, therefore suggesting that the grid deformation strategy employed here and based on a TFI algorithm is a perfectly suitable approach to handle deforming grids for similar cases. Also, the use of the GCL has proved to have no significant effect on the solution accuracy nor the overall performance of the method for these cases. Further investigations are being carried out to show if this holds for more complex geometries.

### 9.3 Results

In the following, we use grid deformation with GCL for all cases. We present here the results for the three AGARD test cases listed in table 1. Results are shown in terms of normal force and moment coefficient loops and instantaneous pressure distributions which are available at a number of points during the pitching cycle. For case CT5, the moment coefficient is calculated about the aerofoil quarter chord, whereas for case CT1 and CT2 the moment coefficient is calculated about 0.273 chord. It was indicated in previous numerical studies [2, 12, 19] that better agreement for the moment coefficient loop was obtained by using this corrected value, suggesting a possible error in the location of the moment center quoted for the experiments.

- AGARD Case CT1

Figures 6 and 7 show the predicted pressure distributions for case CT1 at eight different incidences during the cycle plotted against the measured pressure distributions as given in the AGARD report [1]. Excellent agreement is obtained all throughout the cycle with a sharp and accurate prediction of the shock wave which develops on the upper surface of the aerofoil as the incidence increases. A slight overprediction of the shock strength is observed near the maximum incidence (see figure 6(3)).

The loops for the normal force coefficient  $C_n$  and the moment coefficient  $C_m$  are shown in figure 12 for 20, 40, 80 and 160 steps per cycle for the third cycle of the computation. Relatively good agreement is obtained compared to the experimental values although a slight overprediction of the normal force is observed throughout the cycle. Good agreement is obtained for the moment coefficient computed at the corrected moment center position of 0.273c. The present results are in good agreement with previous

published results obtained for Euler calculations [2, 12, 19, 20]. The difference between prediction and experiment may be explained by the fact that the viscous effects have been neglected here when they might not be completely insignificant. However, it is not clear why such discrepancies appeared in the normal force coefficient loop when the pressure distributions seem to be in perfect agreement with the experiment. It has been suggested in previous numerical studies [19] that the experimental data for these AGARD test cases were not fully reliable which could partly explain the mismatch between prediction and experiment.

- **AGARD Case CT2**

Comparison of the predicted and experimental pressure distributions for case CT2 is shown in figures 8 and 9 for eight different incidences during the cycle. Very good agreement is obtained for this test case where the amplitude of the pitching oscillation and the maximum incidence are higher than for the previous test case. Some minor discrepancies can be observed around the shock waves at the high incidence. The shock strength is well predicted but the shock location is predicted slightly downstream of the experiment. Excellent agreement is obtained at lower incidence.

Plots for the normal force and pitching moment coefficients are shown in figure 13 for 20, 40, 80 and 160 steps per cycle. Good agreement is obtained here for the normal force coefficient and the moment coefficient computed at  $0.273c$ . Here again, the differences between the predicted and the experimental loops can be explained by the neglect of the viscous effects and perhaps also by some uncertainties in the experimental data.

All results compare relatively well with those given in [20] for Navier-Stokes calculations.

- **AGARD Case CT5**

Case CT5 is a more challenging test case due to a higher value of the freestream Mach number which may make the viscous effects more significant here. The flow is characterised by the presence of a strong shock wave which develops alternatively on the upper and lower surface of the aerofoil. Comparison of the predicted and experimental pressure distributions for case CT5 is shown in figures 10 and 11 for eight different incidences during the cycle. Larger discrepancies can be observed for this test case between prediction and experiment. An overprediction of the pressure jump across the shock is observed for almost all incidences with a slight oscillation of the pressure at the foot of the shock. Also, the shock is predicted slightly upstream of the experiment which is consistent with most results (Euler or Navier-Stokes) given in previous publications [2, 3, 12, 29, 41]. However, the present results show significant improvement compared to those obtained in [12] with a thin-layer Navier-Stokes solver.

The comparison of the normal force and moment coefficient is shown in figure 14 for the third cycle of the computation. The normal force coefficient  $C_n$  is clearly underestimated here all throughout the cycle, possibly due to the overprediction of the shock

Case	Number of time steps per cycle	Number of pseudo time steps per cycle	Average number of pseudo time steps per real time step	CPU/cycle (in Work Units)
CT1	160	634	3.9	2643
	80	503	6.3	1740
	40	389	9.2	1649
	20	274	13.7	1250
	10	189	18.9	915
CT2	160	917	5.7	3860
	80	667	8.3	2746
	40	463	11.6	2275
	20	506	25.3	2783
CT5	160	901	5.6	3814
	80	617	7.7	3028
	40	437	10.9	2201
	20	292	14.6	1160
	10	198	19.8	956

Table 3: Effect of the size of the time step on the flow solver efficiency

strength and its location a little too far upstream, resulting in an under-estimation of the pressure coefficient compared to the experiment. Some of the discrepancies can be associated with the neglect of the viscous terms. The results obtained for the  $C_m$  are relatively good, although it is clear that its value is also affected by the wrong prediction of shock location.

#### 9.4 Effect of the size of the time step

The size of the global time step is one of the numerical parameters which affects the efficiency of the CG algorithm as well as the time-accuracy of the solution. However, the dual-time method provides here a second-order discretisation in time compared to the first-order discretisation associated with all single time-discretisation methods based on time linearisation, therefore allowing larger time steps to be used. Results obtained with a single time-discretisation method for similar studies were given in [12]. It was observed therein that the smaller the time step, the easier the linear system is to solve at each iteration at the cost of an increased number of steps for each cycle for the solution of the flow equations.

Here, we investigate the effect of the size of the time step on the efficiency and the accuracy of the flow solver. The results of this investigation are given in table 3 for each AGARD test case in terms of the number of pseudo time steps and CPU time required to compute one cycle when varying the size of the time step (or the number of steps per cycle).

For all cases, we note that, as the size of the global time step increases, although the average number of implicit iterations per time step increases, the overall number of implicit

Case	CGS Tolerance	Number of pseudo time steps per cycle	CPU/cycle (in Work Units)
CT1	0.001	458	2235
	0.01	503	2177
	0.05	600	2361
	0.1	694	3054
CT2	0.001	620	2950
	0.01	667	2794
	0.05	772	3574
	0.1	874	3738
CT5	0.001	590	2736
	0.01	617	2584
	0.05	674	2971
	0.1	752	3035

Table 4: Effect of the CG tolerance on the flow solver efficiency

iterations per cycle decreases, leading to a significant reduction in the overall CPU time. Here, the CPU time required to compute one cycle is expressed in terms of Work Units, where one work unit is the time required to compute one explicit iteration.

The results of the present method using dual-time compare favourably with those given in [12] for the AF-CGS code using a single time discretisation approach. With the AF-CGS code, and for the same cases run on similar grids, the restrictions associated with the time linearisation and the conditioning of the linear system are clear. For the best cases, the minimum number of time steps required to compute one cycle was about 800 for case CT1 and 570 for case CT5. By comparison, the present method allows very large time steps to be used, resulting in the number of time steps per cycle being as little as 20, and even 10 in some cases.

The loops for the normal force and moment coefficients for all three cases are given in figures 12, 13 and 14 respectively for varying number of steps per cycle. The loops are plotted for the third cycle of the computation only.

All three cases present similar behaviour with respect to the size of the global time step, with a very good agreement for the normal force coefficient even for relatively large time steps, and a more sensitive behaviour for the moment coefficient, characterised by a slight deterioration of the  $C_m$  prediction as the time step increases. However, the benefit of using a second-order discretisation in time is clear when looking at the good accuracy obtained at large time steps.

## 9.5 Effect of the CG tolerance

Another numerical parameter which has an influence on the efficiency of the CG method

Grid name	Grid size	Number of cells on aerofoil surface	Number of cells along wake line	Average distance of first cells
grid1	96 × 24	72	12	0.0039–0.015
grid2	128 × 32	96	16	0.0027–0.010
grid3	192 × 48	144	24	0.0018–0.0071
grid4	256 × 64	192	32	0.0014–0.0054

Table 5: Grids used for the grid refinement study

is the CG tolerance (i.e., the convergence criterion used for the solution for the linear system). Table 4 shows the results obtained for different CG tolerances varying from 0.001 to 0.1. The results given here were obtained on a  $128 \times 32$  grid for 80 steps per cycle.

For all three cases considered here, we note that a stricter tolerance results in a larger number of CG steps at each implicit iteration but an overall reduction of the number of implicit iterations per cycle, whereas less strict tolerance require less CG steps per iterations but more iterations per cycle to converge to the solution. This can be explained by the fact that larger (i.e., less strict) tolerances result in a linear system which converges quicker but presumably to a less accurate solution leading to degraded pseudo-time convergence. Looking at the CPU time required, the optimum convergence criterion for the CG algorithm is here 0.01. This value was used for all the results presented in this report.

## 9.6 Effect of grid refinement

Finally we examine the effect of mesh refinement by carrying out comparisons of the average number of implicit iterations per cycle and the CPU time per cycle and per grid point for different grid sizes. Four grids are considered here for this grid dependence study which are presented in table 5. All four grids were extracted from the same grid by interpolating the stretching functions in order to preserve the good quality of the initial grid, in terms of smoothness and grid line orthogonality. The spacing of the first layer of grid points from the aerofoil surface increases from the leading edge to the trailing edge of the aerofoil. The spacing at the leading edge and trailing edge for each grid are also given in table 5. Also, all grids consist of three blocks. Note that the  $128 \times 32$  grid (i.e., grid 2) is the grid used to obtain all the other results given in this report.

The results are given in table 6 and were obtained for 80 steps per cycle. As expected, the number of implicit iterations required to compute one cycle increases as the mesh is refined, leading to a significant increase of the overall CPU time.

It is interesting to note that the number of pseudo time steps per cycle does not increase dramatically. Also, the CPU per cycle and per grid point decreases as the size of the mesh increases, suggesting thereby no deterioration of the efficiency of the method on fine meshes, which represents an improvement compared to the previous results obtained in [12] with the

Case	Mesh	Number of pseudo time steps per cycle	CPU/cycle (in Work Units)	CPU/cycle/g.p. (in Work Units)
CT1	96 × 24	472	1941	0.84
	128 × 32	503	2176	0.53
	192 × 48	613	2708	0.29
	256 × 64	638	3176	0.19
CT2	96 × 24	620	3065	1.33
	128 × 32	667	3281	0.80
	192 × 48	783	3489	0.38
	256 × 64	869	4310	0.26
CT5	96 × 24	546	2353	1.02
	128 × 32	617	2590	0.63
	192 × 48	779	3285	0.35
	256 × 64	888	4807	0.29

Table 6: Effect of grid refinement on flow solver efficiency

AF-CGS code (using single-time discretisation), for which it was observed that the performance of the CG method degraded as the size of the problem increased, therefore requiring smaller time steps on finer meshes for the linear solver to converge within the specified number of CG steps.

Figures 15, 16 and 17 show the corresponding normal force and moment coefficients for the various grids considered here. Perfect matching is obtained with all grids for the normal force coefficient, whereas the moment coefficient exhibits a slight sensitivity to the size of the grid employed. However, these results suggest that no significant gain in accuracy is obtained when using fine grids, and that sufficient accuracy can be obtained on a relatively coarse grid (typically 128 × 32) for Euler computations at similar flow conditions.

### 9.7 Multi-Element Demonstration : Williams Airfoil

The above test cases are all single element aerofoils and do not represent a major challenge for the grid deformation and grid re-generation process. In order to demonstrate the performance of the grid deformation technique described in the previous section, we consider here a demonstration case for the Williams aerofoil (Configuration B) [43] with an oscillating flap. Several test cases were considered for this configuration with various amplitudes for the flap deflection up to 15 degrees. The mesh consists of 15 blocks with 11,228 grid points (10,367 mesh cells). The blocks which are allowed to deform are selected by the user by means of a graphical interface. The choice is made to retain the overall quality of the grid. For this particular problem, a minimum of four blocks for very small flap deflections and up to a maximum of ten blocks for larger amplitudes (over 10 degrees) were considered. It was found,

Case	Aerofoil	$M_\infty$	$\alpha_m$	$\alpha_0$	$k$	$x_m$
1	Williams	0.58	$0^\circ$	$5^\circ$	0.0814	0.25
2	Williams	0.58	$7^\circ$	$7^\circ$	0.0814	0.25

Table 7: Demonstration cases for Williams aerofoil with oscillating flap

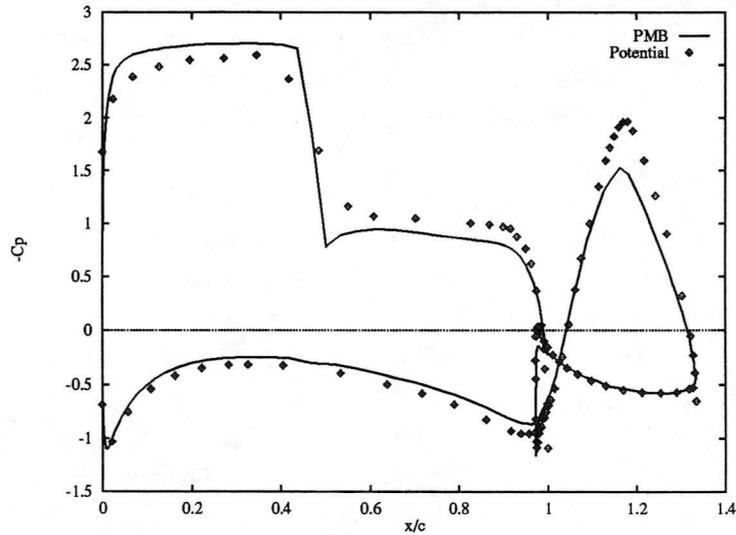


Figure 2: Pressure coefficient for Williams Aerofoil (Configuration B)  
Comparison for steady computation :  $M_\infty = 0.58$ ,  $\alpha = 0$

in all cases, that better deformed grids were obtained when allowing a large number of blocks to be deformed, allowing the distortion of the mesh to be reduced and spread over a larger flow region.

The flow conditions used for the steady computation about the Williams aerofoil are a freestream Mach number of 0.58 and a zero incidence angle, which results in a supercritical flow. Previous computations of this test case [40] indicate a strong shock wave on the upper surface of the main aerofoil situated at approximately 50% chord of the aerofoil, with a local Mach number reaching 1.5 just ahead of the shock.

Comparison has been made with the steady version of the code for the Williams aerofoil (Configuration B) and compared with previous results obtained with an analytical full potential solution [40] and other computations [38]. Excellent agreement with the results of Stolcis and Johnston [38] was obtained for the  $C_p$  distribution (see figure 2).

We present here the results for the two cases listed in table 7, where  $M_\infty$  is the free-stream Mach number,  $\alpha_m$  is the mean deflection angle of the flap,  $\alpha_0$  is the amplitude of the flap oscillation,  $k$  is the reduced frequency and  $x_m$  is the moment center used to calculate the

pitching moment. In all cases, the deflection of the flap is counted with respect to the position of the flap of the original Williams aerofoil (Configuration B), which is positive when the flap is deflected downward and negative when deflected upward. The flap is rigidly rotated about the point situated near its leading edge with coordinates  $x = 0.98$  and  $y = -0.07$ . A view and close up of the original and deformed grid around the flap is shown in figures 18 and 19.

For all cases, an initial solution was first obtained by solving a steady state problem at the mean deflection angle and the flap was then set in motion to solve the unsteady problem for the oscillating flap. The calculations are continued until a periodic solution is obtained, usually after two or three cycles. For large flap deflection problems, it was found that a reduction of the implicit CFL number (down to approximately 50) was necessary to obtain the initial steady state solution, after which the implicit CFL number was increased up to 250 for the rest of the unsteady computation.

The results obtained for case 1 are shown in figure 20 in terms of lift coefficient and pitching moment coefficient calculated about the quarter-chord of the main aerofoil. We note that the converged solution is obtained in less than two cycles. Figures 21 and 22 show the corresponding  $C_p$  distributions at eight different angles during the cycle. As the flap is deflected downward, the shock wave located on the upper surface of the main aerofoil moves downstream and the pressure plateau ahead of the shock increases slightly, resulting in an increase of lift. At +5 degrees deflection, the shock is located at about 55% of the main aerofoil chord. When the flap is deflected upward, the shock moves back upstream, up to about 35% chord.

Case 2 is significantly more difficult due to the large deformation involved, with a flap deflection reaching 14 degrees with respect to the original position of the flap (i.e., undisturbed grid). Figure 23 shows the results in terms of lift coefficient and pitching moment coefficient. The  $C_p$  distributions at eight different angles are shown in figures 24 and 25. As in the previous case, large flap deflections result in a shock moving downstream and a slightly higher pressure plateau ahead of the shock, resulting in a significant increase of lift. At maximum flap deflection (14 degrees), the shock is located at approximately 70% of the main aerofoil chord, whereas at minimum deflection (corresponding to a zero deflection angle), the shock is situated at about 50% chord. The  $C_p$  distributions also show that the extra lift generated by the flap itself increases significantly as the flap moves downward. For this particular test case, the lift coefficient varies of about  $\pm 20\%$  during the cycle.

Figure 26 shows some pressure contours for case 2 at four different angles. The displacement of the strong shock wave on the upper surface of the main aerofoil is clearly visible here.

## 10 Conclusions

The capability of an unfactored implicit method for the solution of the two-dimensional Euler equations on moving meshes has been demonstrated. The results obtained confirm the applicability of the current time stepping strategy, which combines a dual-time approach to

discretise the unsteady equations with an implicit time stepping method for the solution of the steady state problem in pseudo-time. The method is based on a General Conjugate Gradient solution of the linear system with a BILU factorisation used for the preconditioner. This preconditioning strategy allows decoupling of the blocks for the solution of the linear system, resulting in a very efficient parallel implementation of the method.

The method was incorporated within a multi-block environment and was used in conjunction with a general moving grid technique which allows rapid and efficient deformation of the grid in the case of complex geometries. Very encouraging results were obtained for a demonstration test case consisting of a two-element aerofoil with an oscillating flap. The new strategy employed, based on a transfinite interpolation of the grid displacements, is not restricted to rigid body motion and simple geometries and it therefore provides a useful tool for computing aeroelastic problems for complex geometries.

Results were presented for several pitching aerofoil flows (AGARD test cases) and good agreement was noted with both experiment and previous computations reported in the literature. Improved performance was obtained compared to [12], not only due to the improved performance of the CG solver and its preconditioner, but also to the use of the dual-time approach for the solution of the unsteady equations.

A grid refinement study was carried out and showed no deterioration of the method as the size of the problem is increased, which again represents an improvement over [12]. Large time steps can be used, corresponding to a minimum of 10 or 20 steps per cycle for some cases. Finally, the use of a Geometric Conservation Law to account for cell area changes was shown to have no significant effect on the accuracy and convergence properties of the method for the flow problems considered here.

The validation of the present method for unsteady Euler computations represents only an intermediate step in the development of a general flow solver capable of tackling turbulent viscous flows for complex geometries, and areas of future work will include the implementation of the viscous terms for modeling the Navier-Stokes equations, along with the implementation of a two-equation  $k - \omega$  turbulence model.

## References

- [1] 'Compendium of Unsteady Aerodynamic Measurements', *AGARD-R-702*, (1982).
- [2] C.B. Allen, 'Central-Difference and Upwind-Biased Schemes for Steady and Unsteady Euler Aerofoil Computations', *Aero. Journal*, Vol. 99, (March 1995).
- [3] C.B. Allen, 'Adaption by Grid Motion for Unsteady Euler Aerofoil Flows', *AGARD Paper 35, Proceedings 77th Fluid Dynamics Panel Meeting and Symposium, Progress and Challenges in CFD Methods and Algorithms, Seville*, (October 1995).
- [4] A. Arnone, M.-S. Liou and L.A. Povinelli, 'Multigrid Time Accurate Integration of Navier-Stokes Equations', *AIAA Paper No. 93-3361*, (1993).
- [5] O. Axelsson, 'Iterative Solution Methods', *Cambridge University Press*, (1994).
- [6] K.J. Badcock, 'AF-CGS-P : An Unfactored Method in Parallel for Turbulent Pitching Aerofoil Flows', *University of Glasgow, Aero Report 9323*, (September 1993).
- [7] K.J. Badcock, 'Newton's Method for Laminar Aerofoil Flows', *University of Glasgow, Aero Report 9330*, (1993).
- [8] K.J. Badcock, I.C. Glover and B.E. Richards, 'Convergence Acceleration for Viscous Airfoil Flows Using an Unfactored Method', in *Second European Conference on CFD*, pp. 333-341. *ECCOMAS*, (1994).
- [9] K.J. Badcock, S. Porter and B.E. Richards, 'Unfactored Multiblock Methods - Part 1 : Initial Method Development', *University of Glasgow, Aero Report 9511*, (1995).
- [10] K.J. Badcock, X. Xu, L. Dubuc and B.E. Richards, 'Preconditioners for High Speed Flows in Aerospace Engineering', in *Numerical Methods for Fluid Dynamics V.*, *Oxford University Press*, pp. 287-294, (February 1996).
- [11] K.J. Badcock, G. Sim and B.E. Richards, 'Aeroelastic Studies using Transonic Flow CFD Modelling', *Paper 18, CEAS, International Forum on Aeroelasticity and Structural Dynamics 1995*, (26-28 June 1995).
- [12] K.J. Badcock and A.L. Gaitonde, 'An Unfactored Implicit Moving Mesh Method for The Two-Dimensional Unsteady Navier-Stokes Equations', *International Journal for Numerical Methods in Fluids*, Vol. 23, pp. 607-631, (1996).
- [13] K.J. Badcock, B.E. Richards and M.A. Woodgate, 'Factored-Unfactored Method for 3D Transonic Flows', in *Computational Fluid Dynamics '96*, pp. 605-610, *J.-A. Desideri et al (Editors), John Wiley & Sons*, (1996).
- [14] K.J. Badcock, W. McMillan, M.A. Woodgate, B.J. Gribben, S. Porter and B.E. Richards, 'Integration of an Implicit Multiblock Code into a Workstation Cluster Environment',

*Parallel Computational Fluid Dynamics: Algorithms and Results using Advanced Computers.* P. Schiano et al. (Eds.), Elsevier Science B.V. Amsterdam, pp. 408-415, (1996).

- [15] A.A. Belov, L. Martinelli and A. Jameson, 'A Novel Fully Implicit Multigrid Driven Algorithm for Unsteady Incompressible Flow Calculations', *Computational Fluid Dynamics '94*, Published in 1994 by John Wiley & Sons Ltd, (1994).
- [16] F. Cantariti, M. Woodgate and K. Badcock, 'Approximate Jacobians for the Euler and Navier-Stokes Equations', *University of Glasgow, Aero Report 97??*, (1997).
- [17] L. Dubuc, K.J. Badcock, M. Woodgate and B.E. Richards, 'Implicit Navier-Stokes Simulations of Unsteady Flows', *in conference proceedings, Conference on Unsteady Aerodynamics, Royal Aeronautical Society, London*, (17-18 July 1996).
- [18] G. Freskos and O. Penanhoat, 'Numerical Simulation of the Flow Field Around Supersonic Air-Intakes', *Journal of Engineering for Gas Turbines and Power*, Vol. 116, pp. 116-123, (January 1994).
- [19] A.L. Gaitonde, 'A Dual-Time Method for the Solution of the Unsteady Euler Equations', *Aero. Journal*, Vol. 98, (October 1994).
- [20] A.L. Gaitonde, 'A Dual-Time Method for the Solution of the 2D Unsteady Navier-Stokes Equations on Structured Moving Meshes', *University of Bristol, Department of Aerospace Engineering, Report No. 716*, (May 1995).
- [21] A.L. Gaitonde and S.P. Fiddes, 'A Moving Mesh System for the Calculation of Unsteady Flows', *AIAA Paper 93-0641*, (1993).
- [22] A.L. Gaitonde and S.P. Fiddes, 'A Comparison of a Cell-Centre Method and a Cell-Vertex Method for the Solution of the Two-Dimensional Unsteady Euler Equations on a Moving Grid', *Journal of Aerospace Engineering*, Vol. 209, (1995).
- [23] A.L. Gaitonde and S.P. Fiddes, 'A Three-Dimensional Moving Mesh Method for the Calculation of Unsteady Transonic Flows', *The Aeronautical Journal of the Royal Aeronautical Society*, (May 1995).
- [24] A.L. Gaitonde and D.P. Jones, 'Parallel Implementation of a Dual-Time Moving Mesh Method for the 2D Unsteady Navier-Stokes Equations', *University of Bristol, Report No. 736*, (December 1995).
- [25] F. Ghaffari, J.M. Luckring, J.L. Thomas, B.L. Bates and R.T. Biedron, 'Multiblock Navier-Stokes Solutions about the F/A-18 Wing-LEX-Fuselage Configuration', *Journal of Aircraft*, Vol. 30, No. 3, pp. 293-303, (May-June 1993).
- [26] P.A. Gnoffo, K.J. Weilmuenster, S.J. Alter, 'Multiblock Analysis for Shuttle Orbiter Re-entry Heating from Mach 24 to Mach 12', *Journal of Spacecraft and Rockets*, Vol. 31, No. 3, pp. 367-377, (May-Jun 1994).

- [27] W.J. Gordon and C.A. Hall, 'Construction of Curvilinear Coordinate Systems and Applications of Mesh Generation', *International Journal of Numerical Methods in Engineering*, Vol. 7, pp. 461-477, (1973).
- [28] B.J. Gribben, 'Progress Report: Application of the Multiblock Method in Computational Aerodynamics', *University of Glasgow, Aero Report 9621*, (1996).
- [29] R. Heinrich, K. Pahlke and H. Bleecke, 'A Three Dimensional Dual-Time-Stepping Method for the Solution of the Unsteady Navier-Stokes Equations', *Conference on Unsteady Aerodynamics (17-18 July, 1996)*, Royal Aeronautical Society, London, (1996).
- [30] A. Jameson, 'Time Dependent Calculations Using Multigrid with Applications to Unsteady Flows Past Airfoils and Wings', *AIAA Paper 91-1596*, (June 1991).
- [31] G. Kalitzin, A.R.B. Gould, J.J. Benton, 'Application of Two-Equation Turbulence Models in Aircraft Design', *AIAA 96-0327, 34th Aerospace Sciences Meeting and Exhibit, Reno, NV*, (Jan 15-18, 1996).
- [32] M. Kathong, R.E. Smith, S.N. Timari, 'Application of Multiple Grids Topology to Supersonic Internal/External Flow Interactions', *Journal of Aircraft*, Vol. 27, No. 3, pp. 245-252, (March, 1990).
- [33] M. Lesoinne and C. Farhat, 'Geometric Conservation Laws for Aeroelastic Computations Using Unstructured Dynamic Meshes', *AIAA Paper 95-1709-CP*, (1995).
- [34] K. Nakahashi and G.S. Deiwert, 'Self Adaptive Grid Method with Application to Airfoil Flow', *AIAA Journal*, Vol. 25, pp. 513-520, (1987).
- [35] T.E. Nelson, D.W. Zingg and G.W. Johnston, 'Compressible Navier Stokes Computations of Multielement Airfoil Flows Using Multiblock Grids', *AIAA Journal*, Vol. 32, No.3, pp. 506-511, (March 1994).
- [36] S. Rill and K. Becker, 'Simulation of Transonic Flows over Twin-Jet Transport Aircraft', *Journal of Aircraft*, Vol. 29, No. 4, pp. 640-646, (July-August 1992).
- [37] A. Rizzi, P. Eliasson, I. Linblad, C. Hirsch, H. Lacor and J. Haeuser, 'The Engineering of Multiblock/Multigrid Software for Navier-Stokes Flows on Structured Meshes', *Computers and Fluids*, Vol. 22, Nos. 2-3, pp. 341-367, (1992).
- [38] L. Stolcis and L.J. Johnston, 'Solution of the Euler Equations on Unstructured Grids for Two-Dimensional Compressible Flow', *Aeronautical Journal*, pp. 181-195, (June/July 1990).
- [39] Th. Streit, 'Euler and Navier-Stokes Solutions for Supersonic Flow Around a Complex Missile', *Journal of Spacecraft and Rockets*, Vol. 31, No. 4, pp. 600-608, (July-Aug, 1994).

- [40] A. Suddhoo and I.M. Hall, 'Inviscid Compressible Flow Past a Multiple Element Aerofoil', *AGARD CP 365*, (1984).
- [41] V. Venkatakrisnan and D.J. Mavriplis, 'Implicit Method for the Computation of Unsteady Flows on Unstructured Grids', *AIAA Paper 95-1705-CP*, (1995).
- [42] N.P. Weatherhill and C.R. Forsey, 'Grid Generation and Flow Calculations for Aircraft Geometries', *Journal of Aircraft*, Vol. 22, No. 10, pp. 855-860, (October 1985).
- [43] B.R. Williams, 'An Exact Test Case for the Plane Potential Flow About Two Adjacent Lifting Aerofoils', *ARC R& M 3717*, (1973).

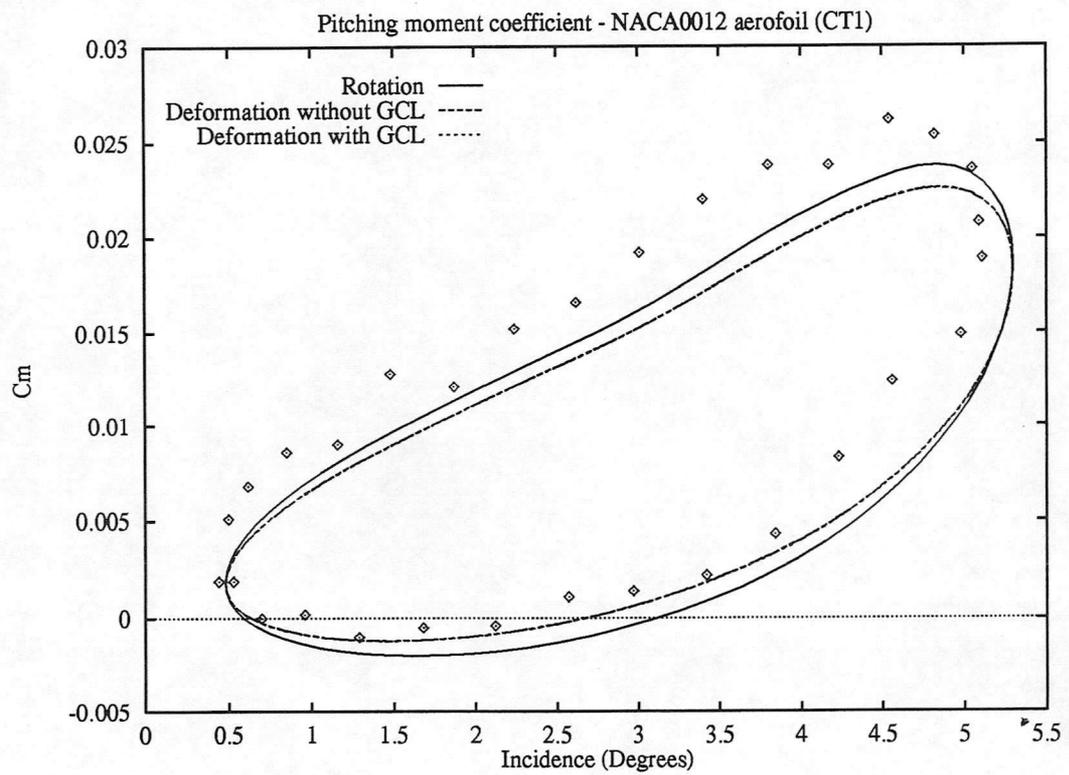
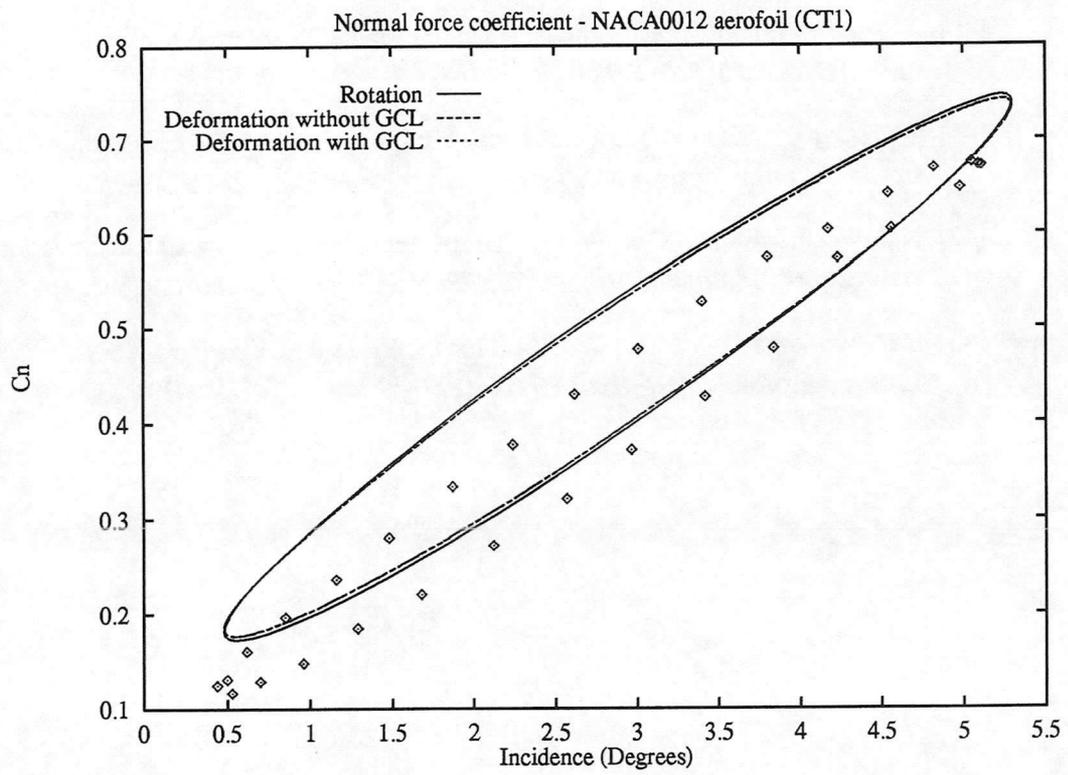


Figure 3: Effect of grid deformation on solution accuracy for case CT1

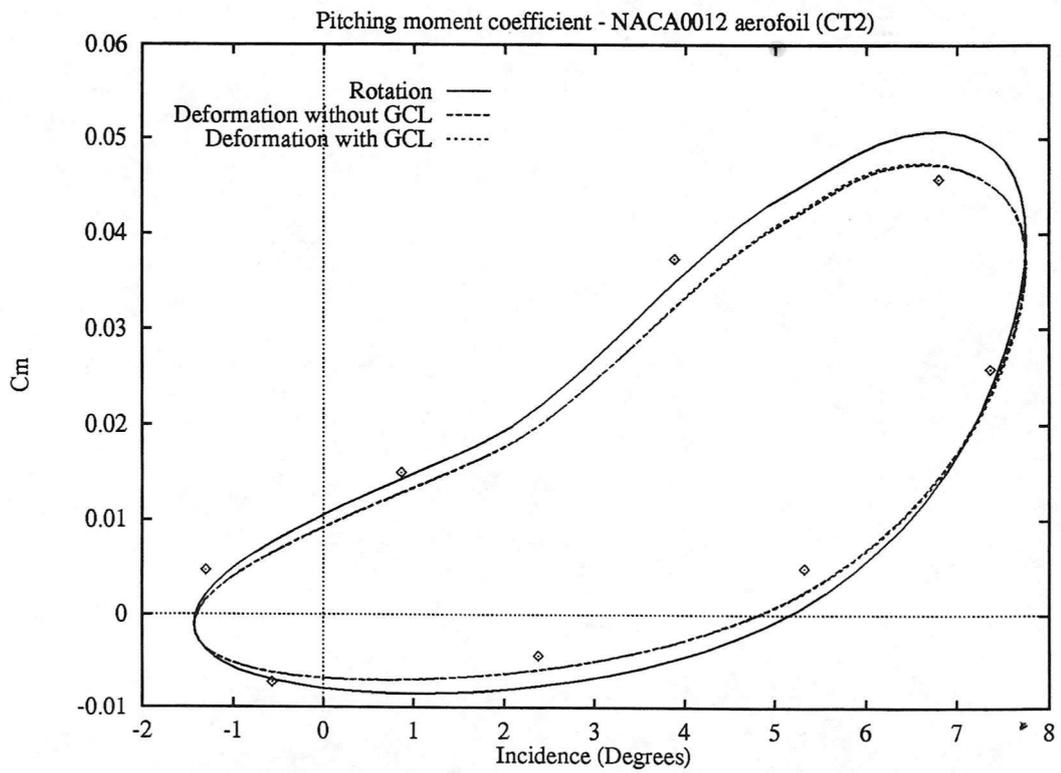
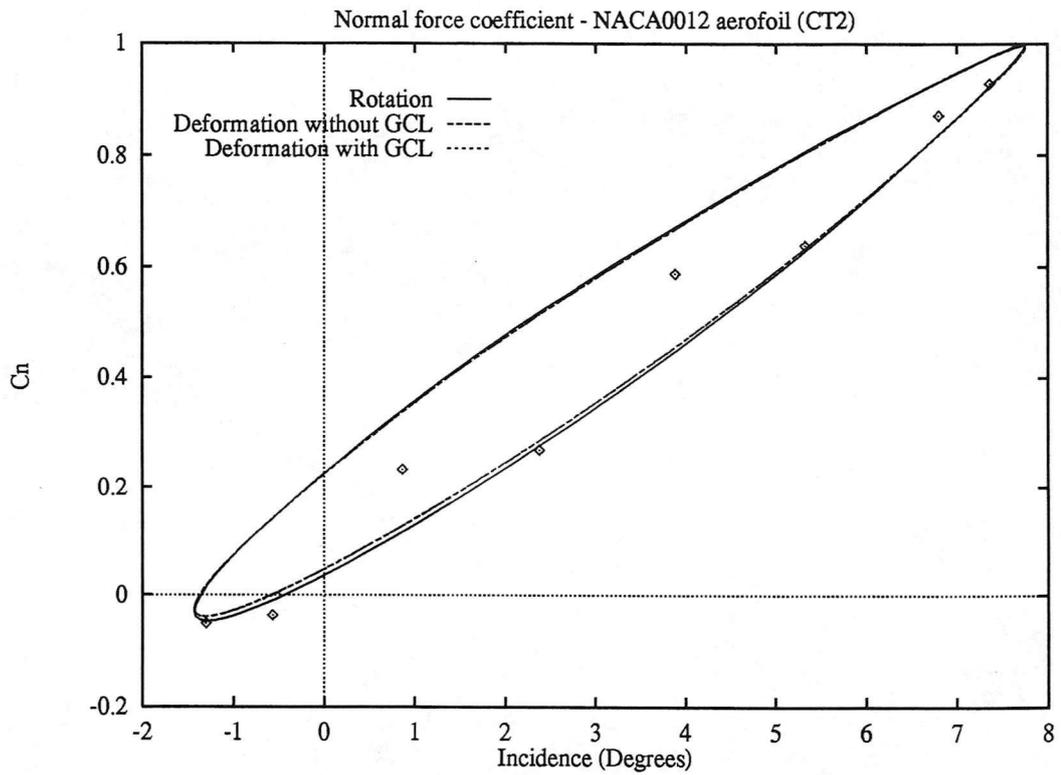


Figure 4: Effect of grid deformation on solution accuracy for case CT2

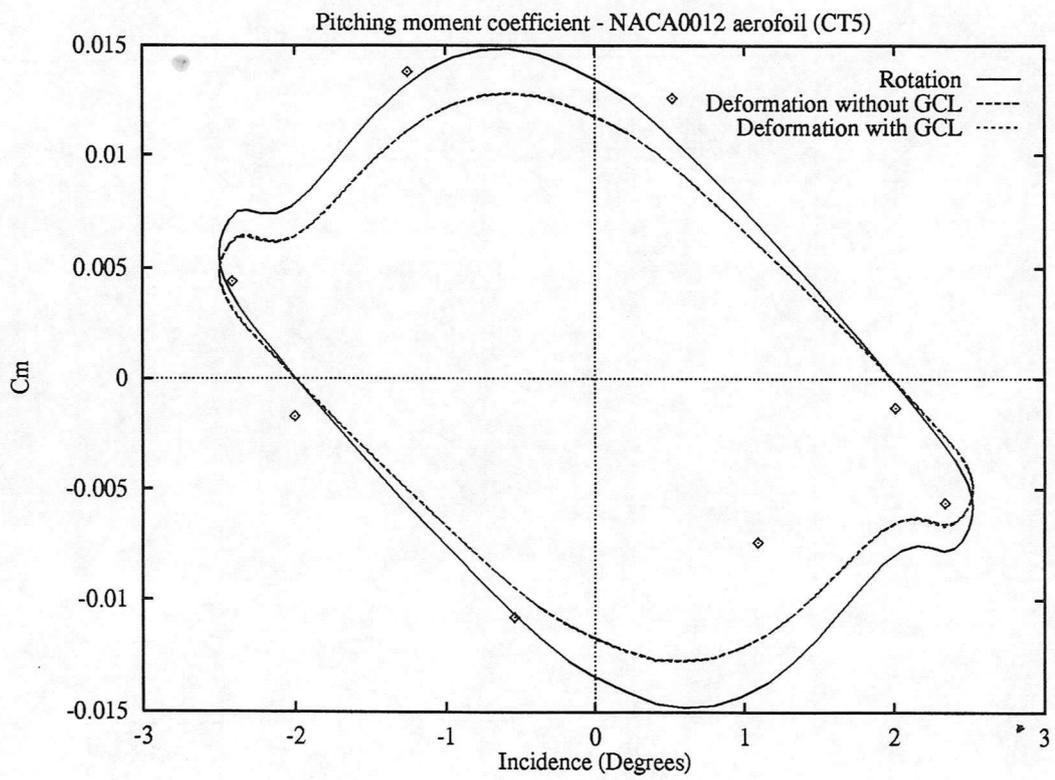
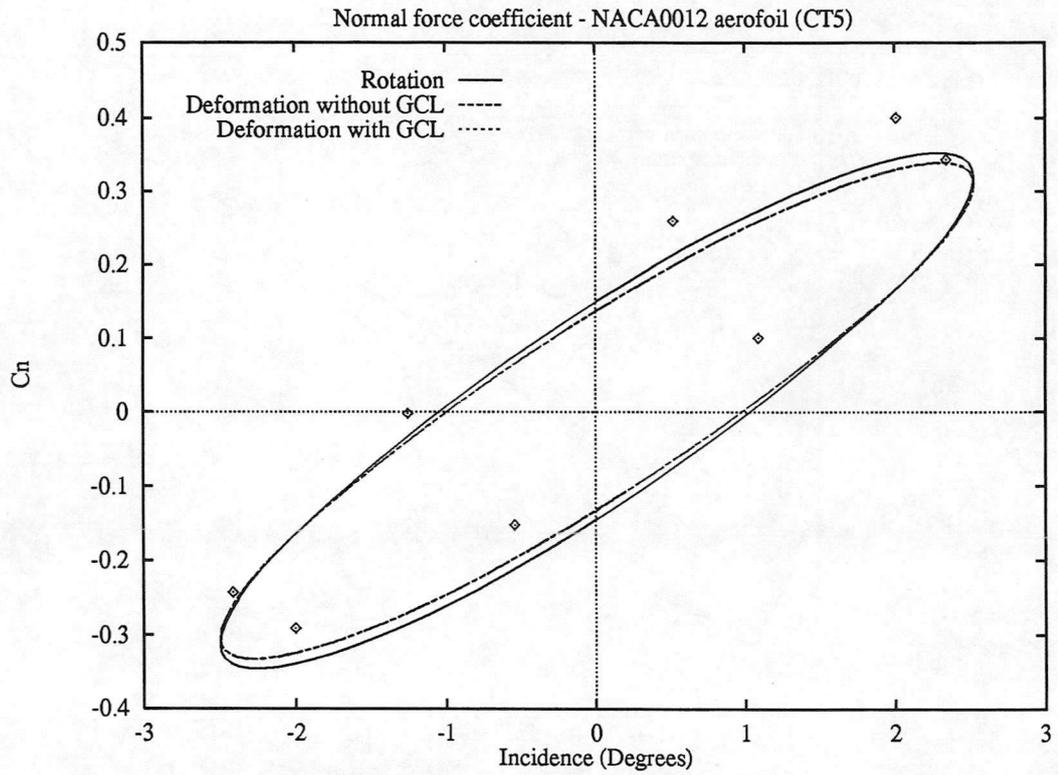
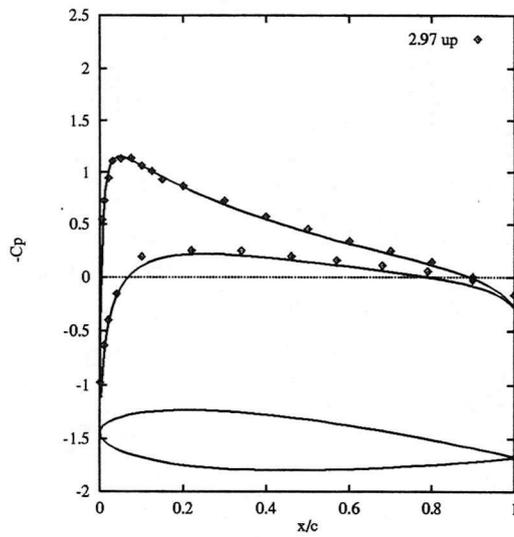
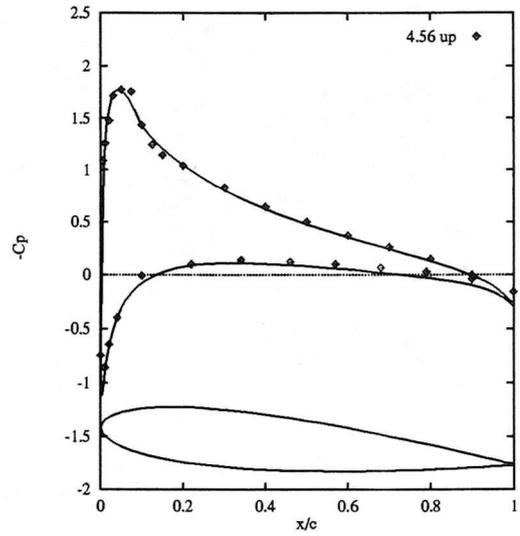


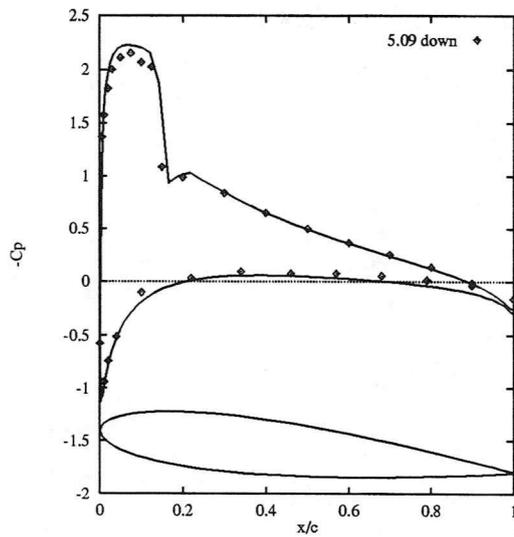
Figure 5: Effect of grid deformation on solution accuracy for case CT5



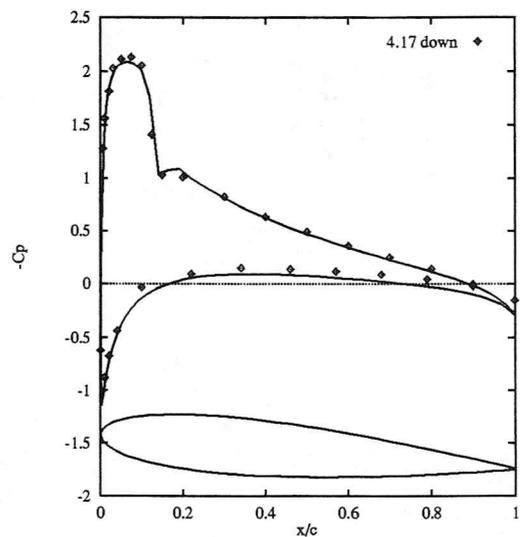
(1)  $\alpha = 2.97^\circ up$



(2)  $\alpha = 4.56^\circ up$

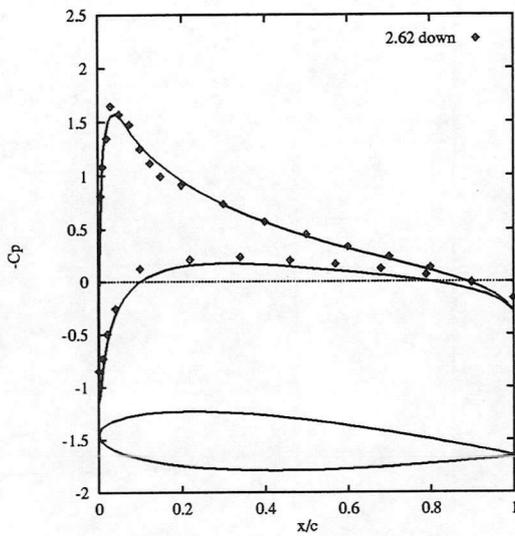


(3)  $\alpha = 5.09^\circ down$

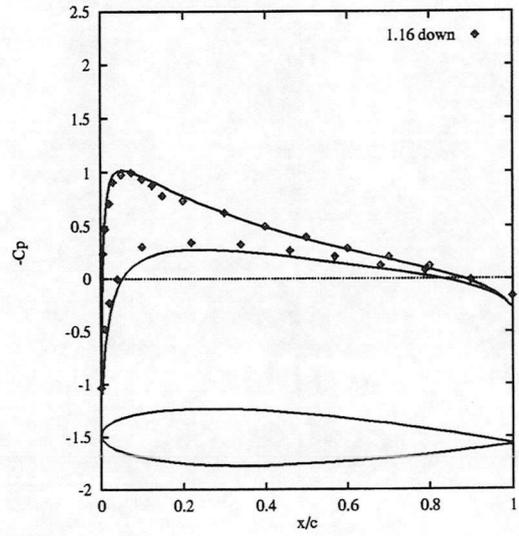


(4)  $\alpha = 4.17^\circ down$

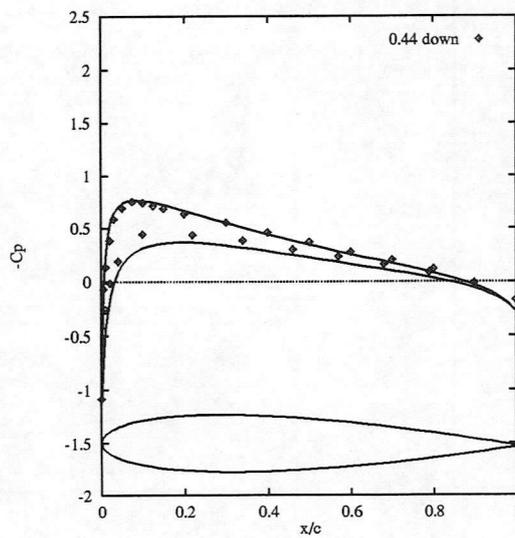
Figure 6: Instantaneous pressure distributions for NACA 0012 aerofoil  
 AGARD Test Case 1  
 $M_\infty = 0.6$ ,  $\alpha = 2.89^\circ + 2.41^\circ \sin(\omega t)$ ,  $k = 0.0808$



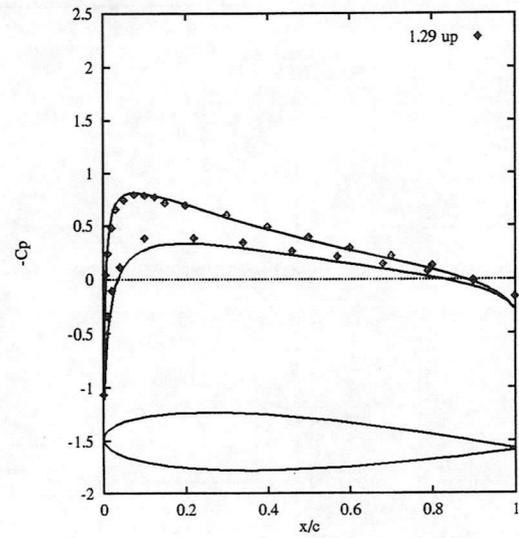
(5)  $\alpha = 2.62^\circ$  down



(6)  $\alpha = 1.16^\circ$  down

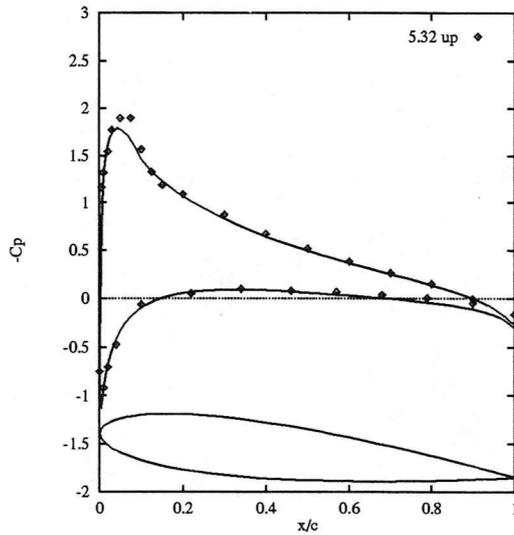


(7)  $\alpha = 0.44^\circ$  down

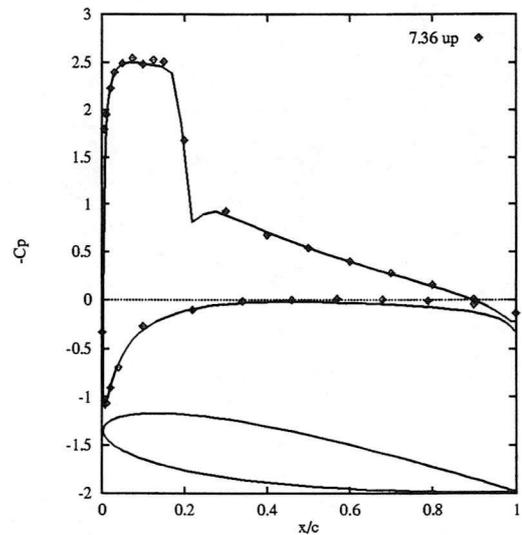


(8)  $\alpha = 1.29^\circ$  up

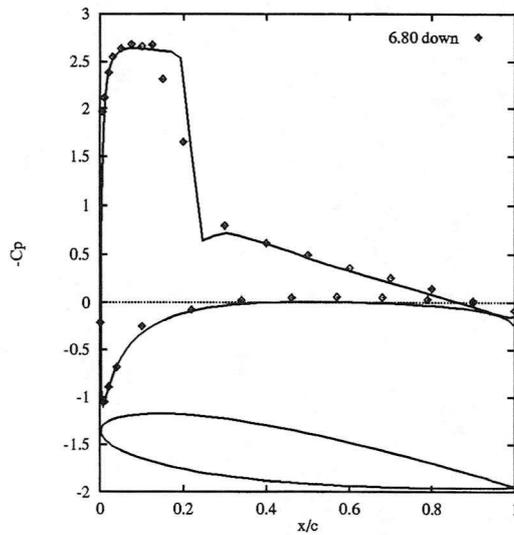
Figure 7: Instantaneous pressure distributions for NACA 0012 aerofoil  
 AGARD Test Case 1  
 $M_\infty = 0.6$ ,  $\alpha = 2.89^\circ + 2.41^\circ \sin(\omega t)$ ,  $k = 0.0808$



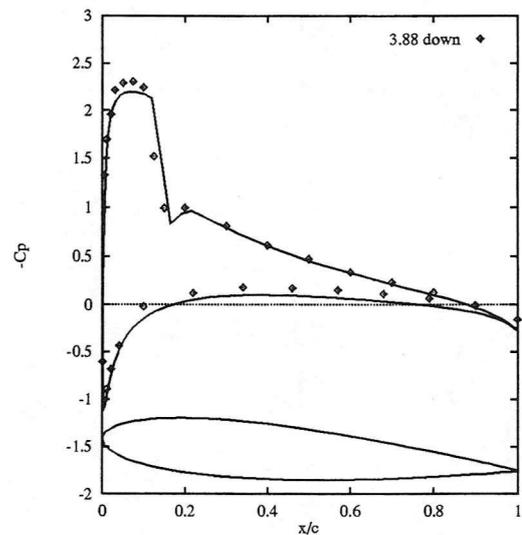
(1)  $\alpha = 5.32^\circ \text{ up}$



(2)  $\alpha = 7.36^\circ \text{ up}$

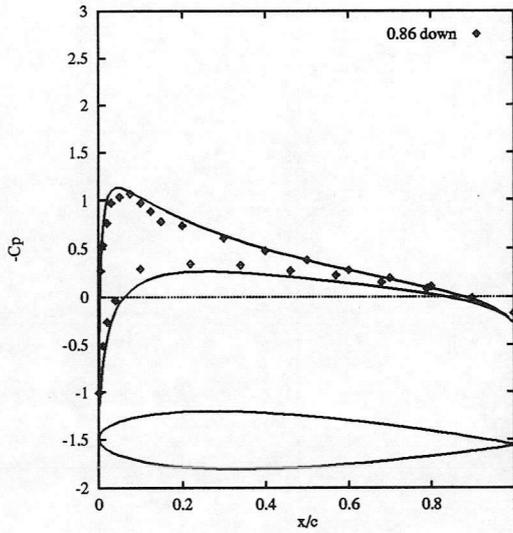


(3)  $\alpha = 6.80^\circ \text{ down}$

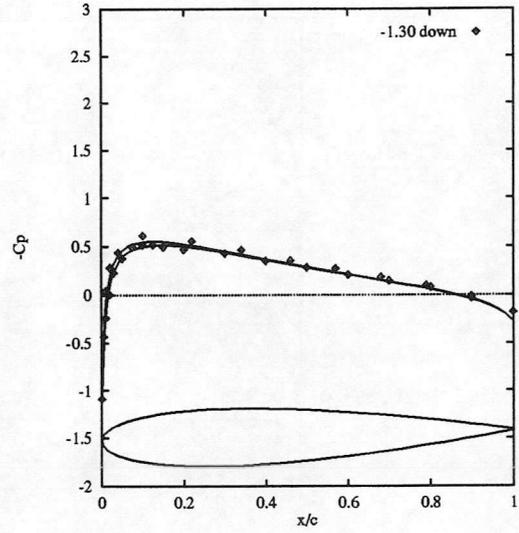


(4)  $\alpha = 3.88^\circ \text{ down}$

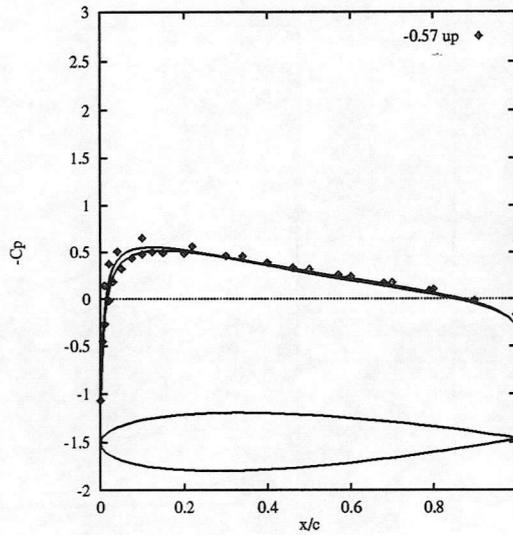
Figure 8: Instantaneous pressure distributions for NACA 0012 aerofoil  
 AGARD Test Case 2  
 $M_\infty = 0.6$ ,  $\alpha = 3.16^\circ + 4.59^\circ \sin(\omega t)$ ,  $k = 0.0811$



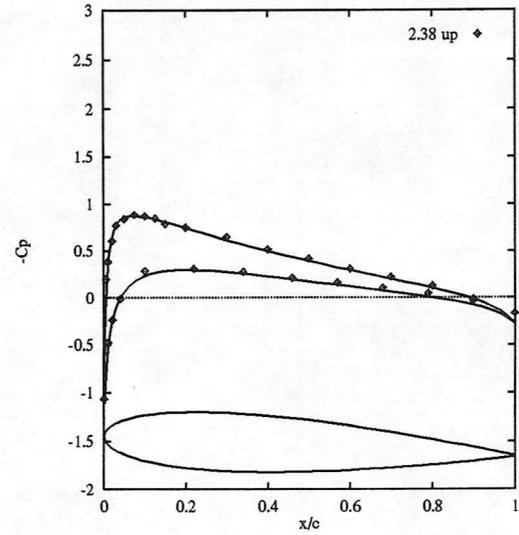
(5)  $\alpha = 0.86^\circ \text{down}$



(6)  $\alpha = -1.30^\circ \text{down}$



(7)  $\alpha = -0.57^\circ \text{up}$



(8)  $\alpha = 2.38^\circ \text{up}$

Figure 9: Instantaneous pressure distributions for NACA 0012 aerofoil  
 AGARD Test Case 2  
 $M_\infty = 0.6$ ,  $\alpha = 3.16^\circ + 4.59^\circ \sin(\omega t)$ ,  $k = 0.0811$

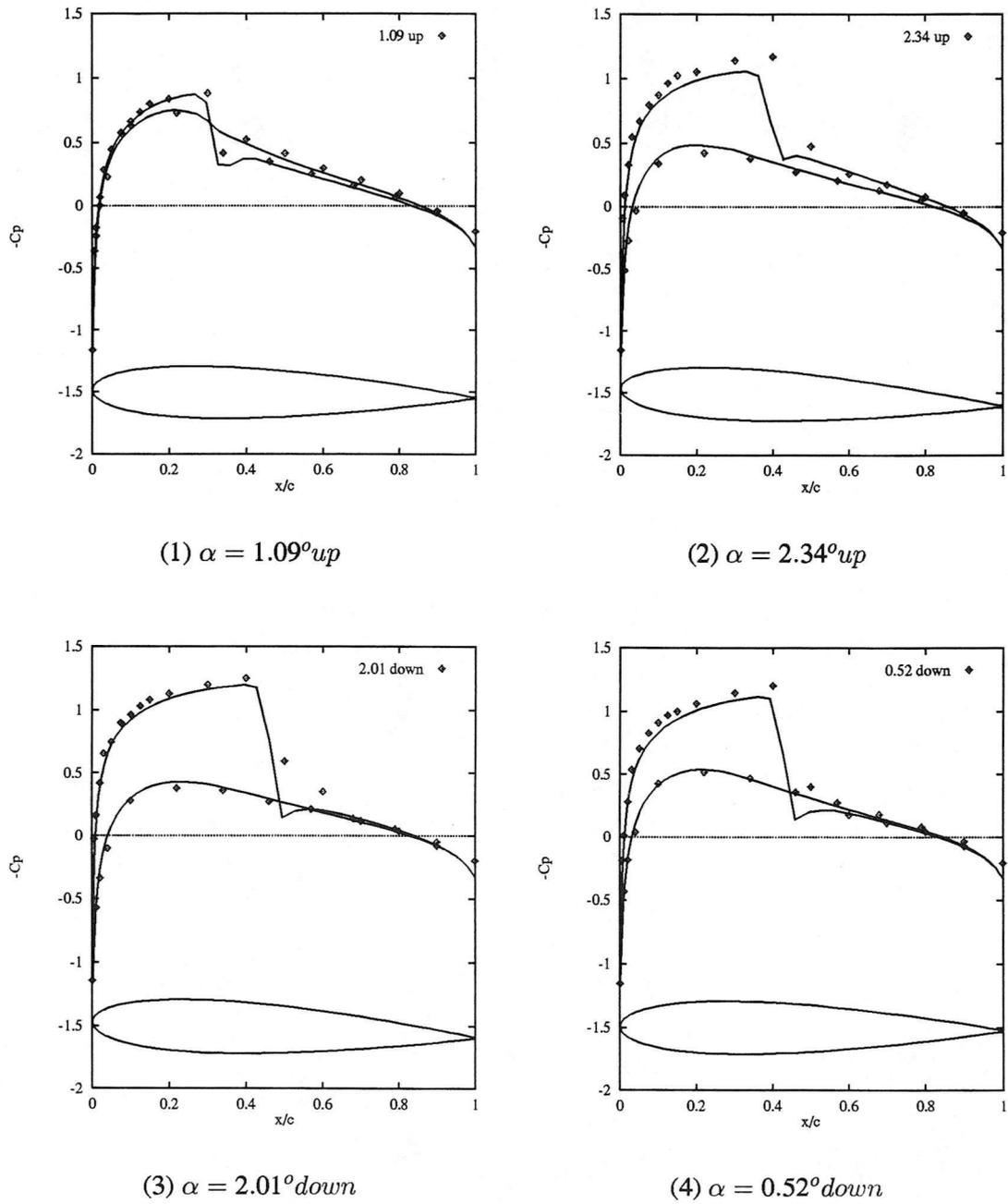
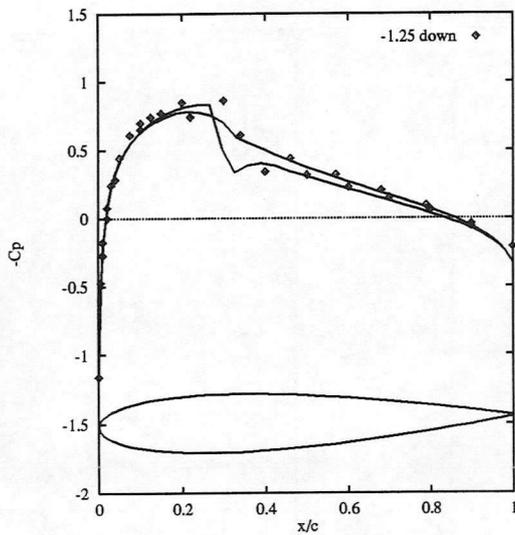
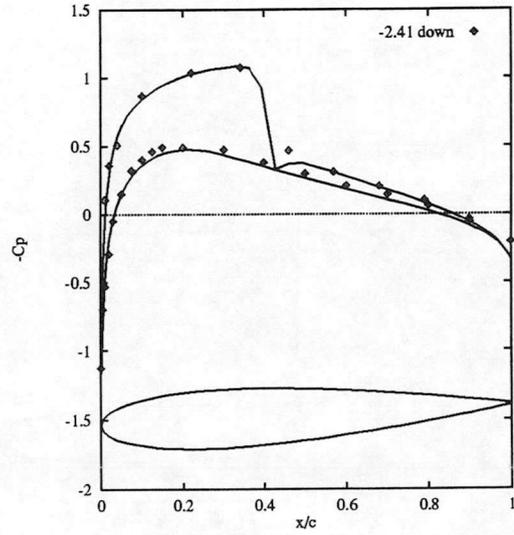


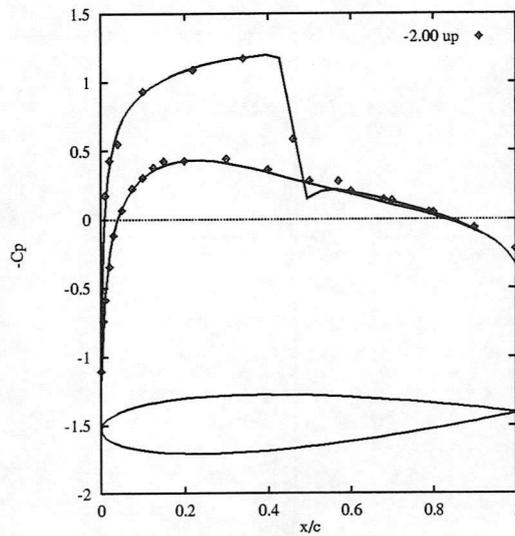
Figure 10: Instantaneous pressure distributions for NACA 0012 aerofoil  
 AGARD Test Case 5  
 $M_\infty = 0.755, \alpha = 0.016^\circ + 2.51^\circ \sin(\omega t), k = 0.0814$



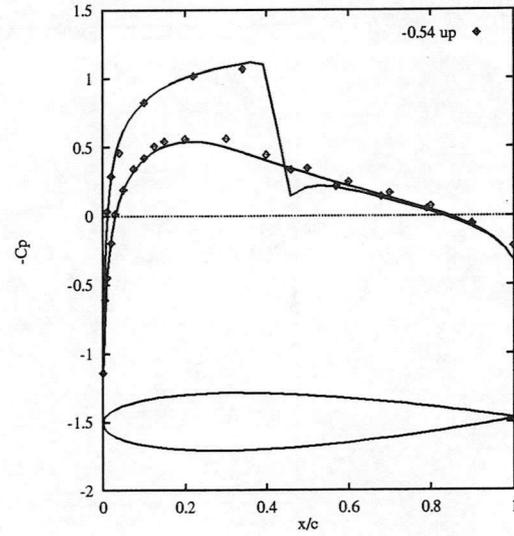
(5)  $\alpha = -1.25^\circ \text{down}$



(6)  $\alpha = -2.41^\circ \text{down}$



(7)  $\alpha = -2.00^\circ \text{up}$



(8)  $\alpha = -0.54^\circ \text{up}$

Figure 11: Instantaneous pressure distributions for NACA 0012 aerofoil  
 AGARD Test Case 5  
 $M_\infty = 0.755$ ,  $\alpha = 0.016^\circ + 2.51^\circ \sin(\omega t)$ ,  $k = 0.0814$

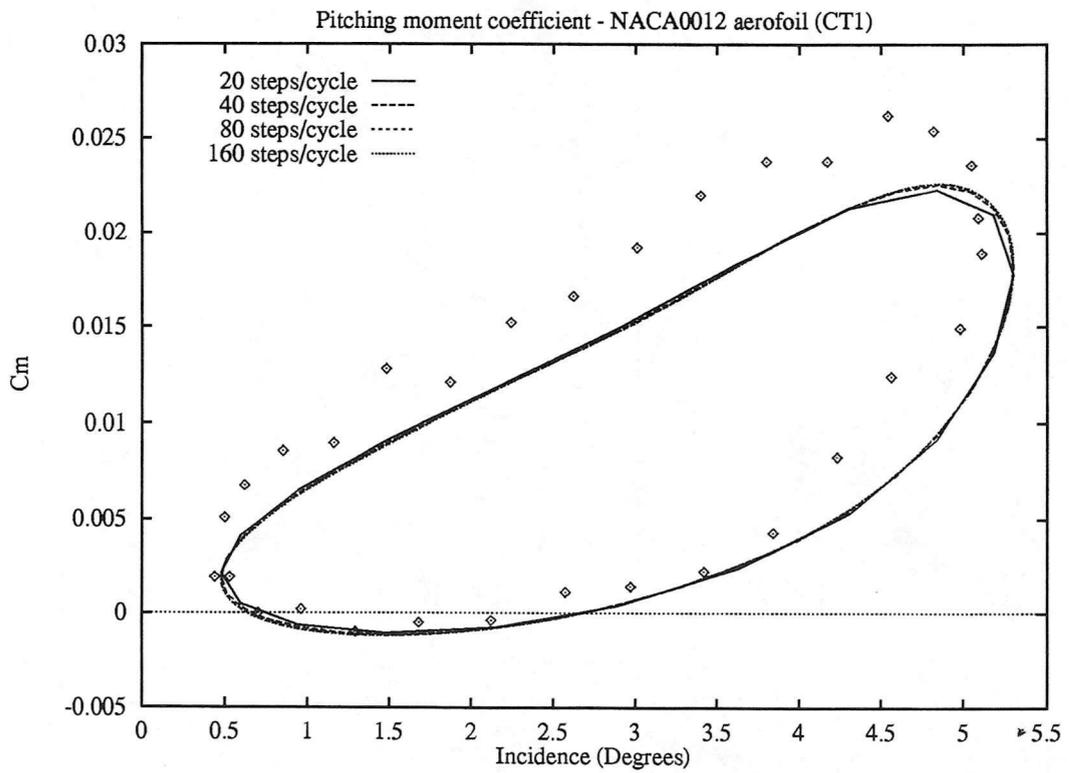
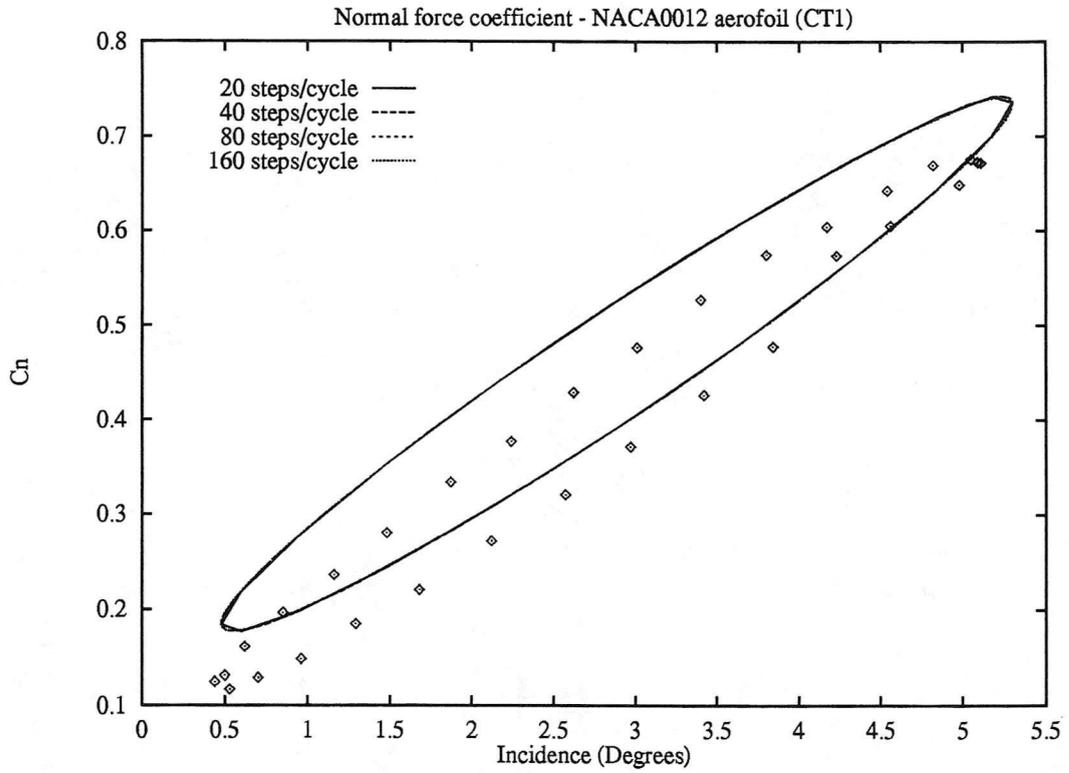


Figure 12: Effect of time step : Integrated normal force (top) and pitching moment coefficient (bottom) for case CT1

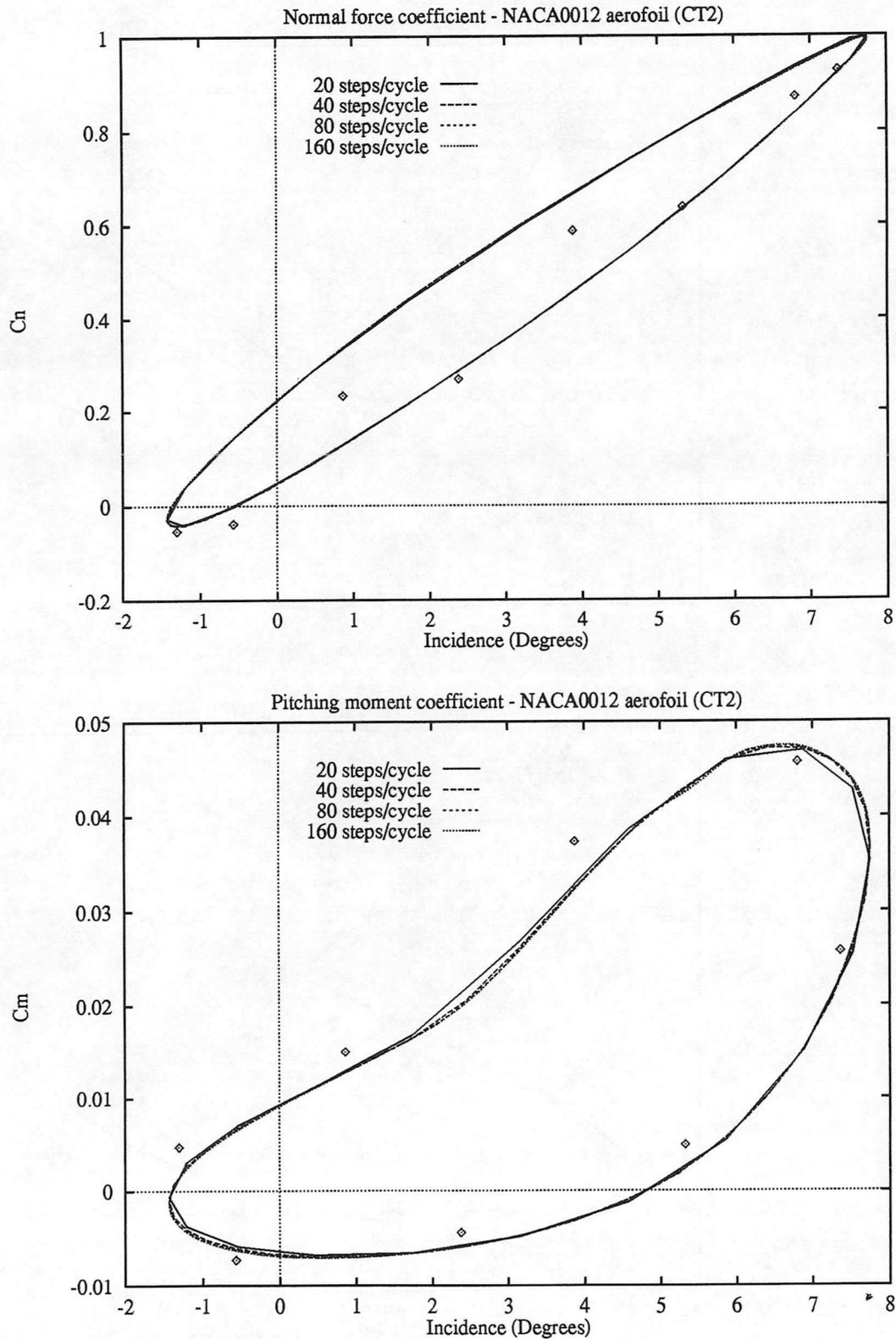


Figure 13: Effect of time step : Integrated normal force (top) and pitching moment coefficient (bottom) for case CT2

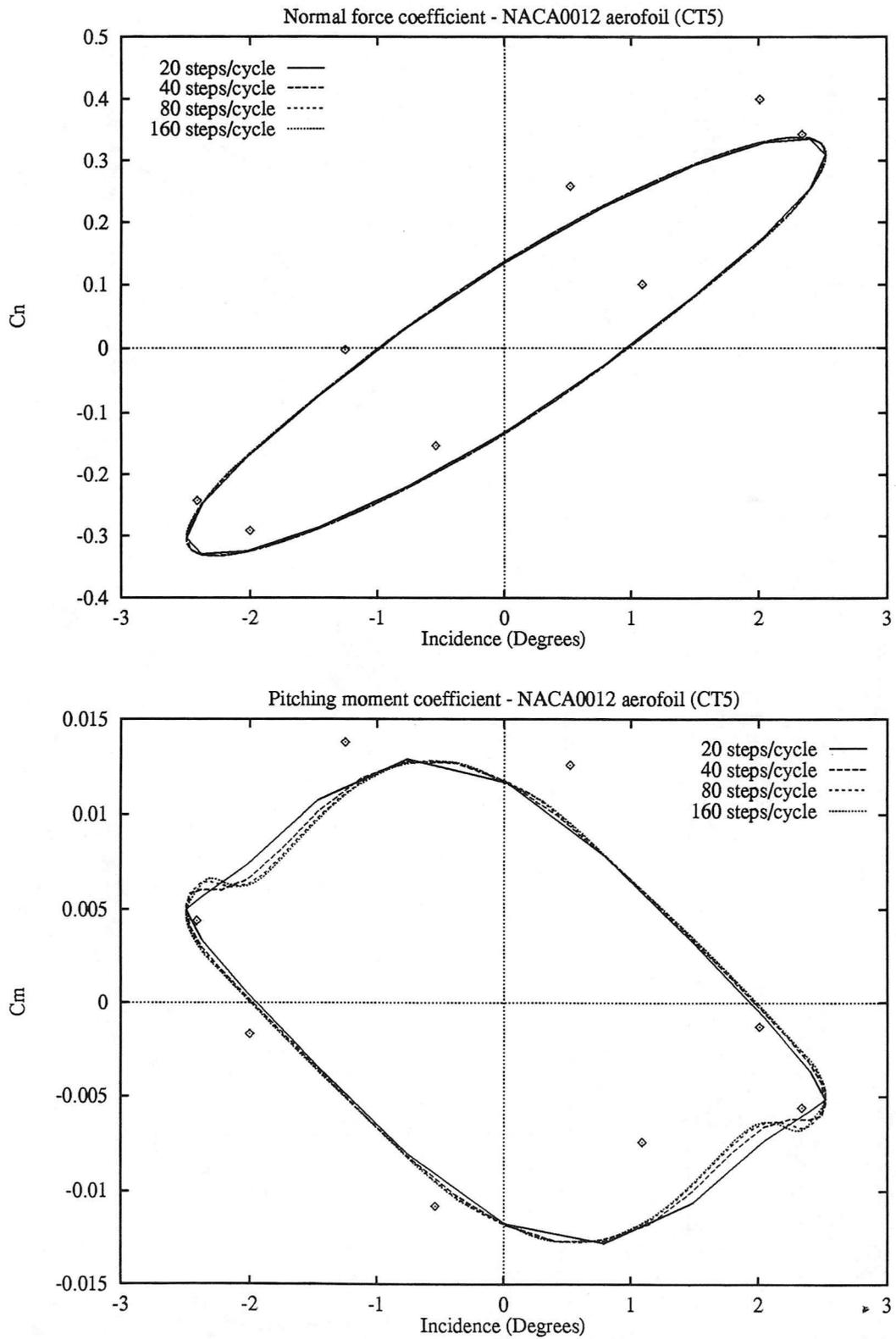


Figure 14: Effect of time step : Integrated normal force (top) and pitching moment coefficient (bottom) for case CT5

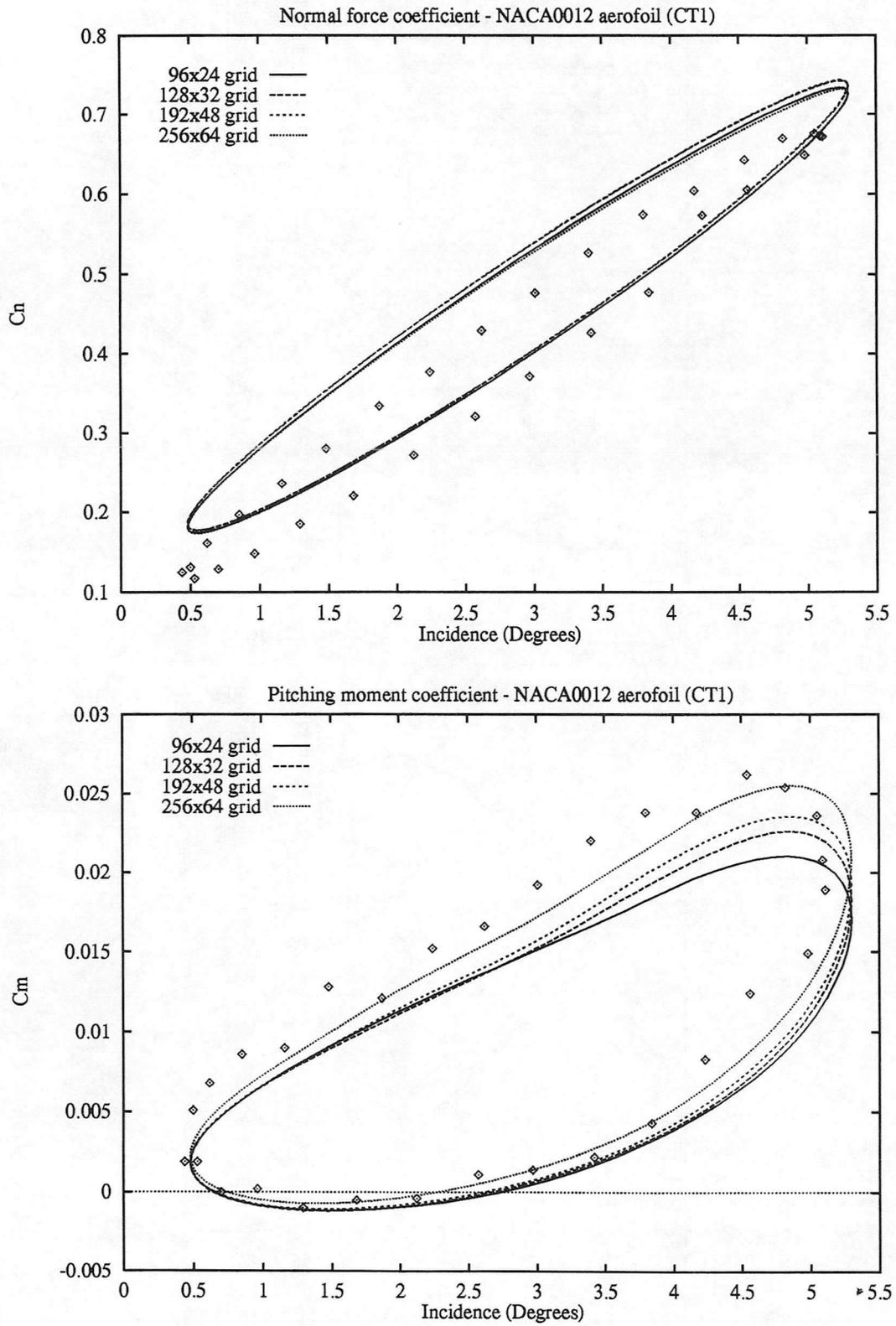


Figure 15: Grid refinement study : Integrated normal force (top) and pitching moment coefficient (bottom) for case CT1

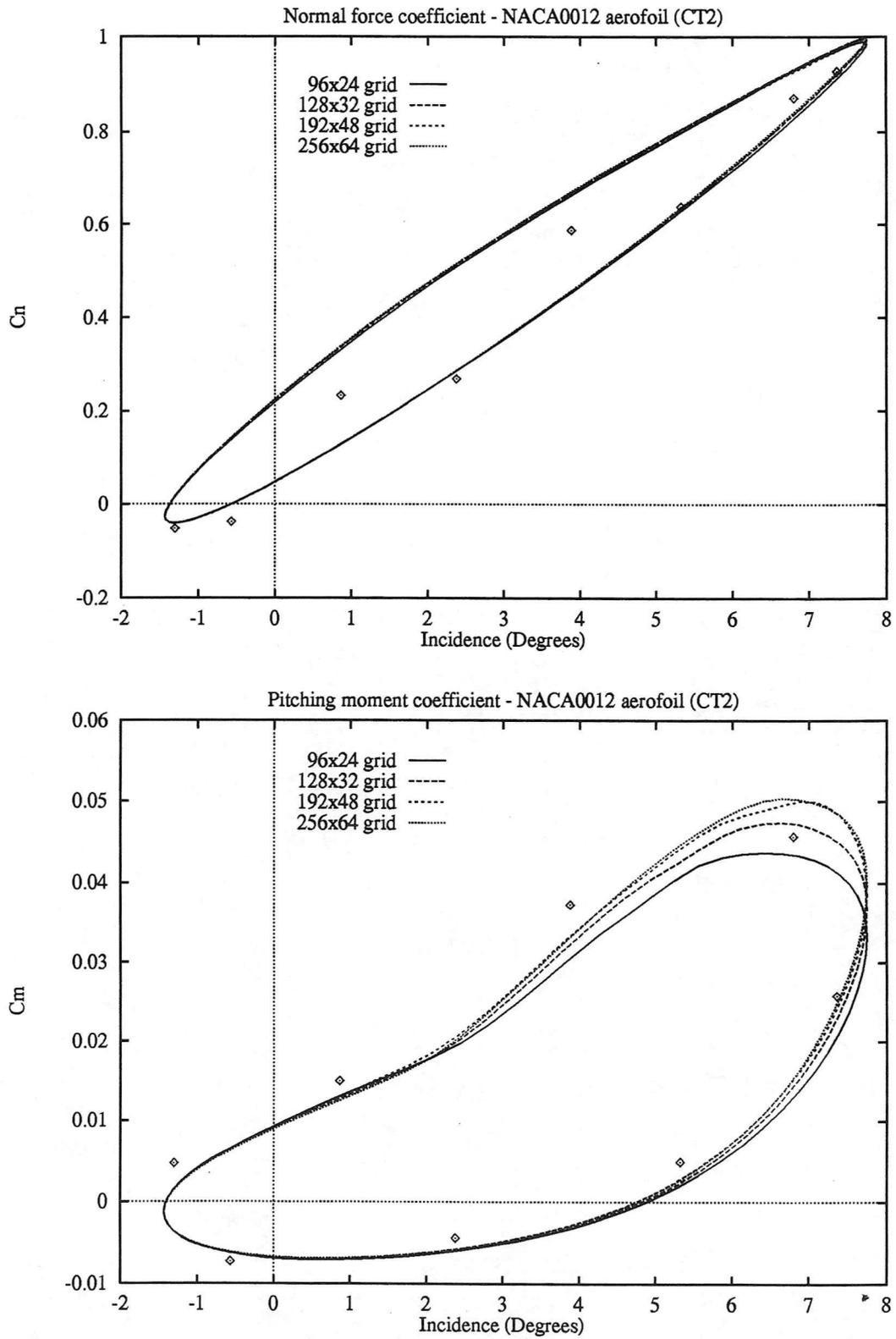


Figure 16: Grid refinement study : Integrated normal force (top) and pitching moment coefficient (bottom) for case CT2

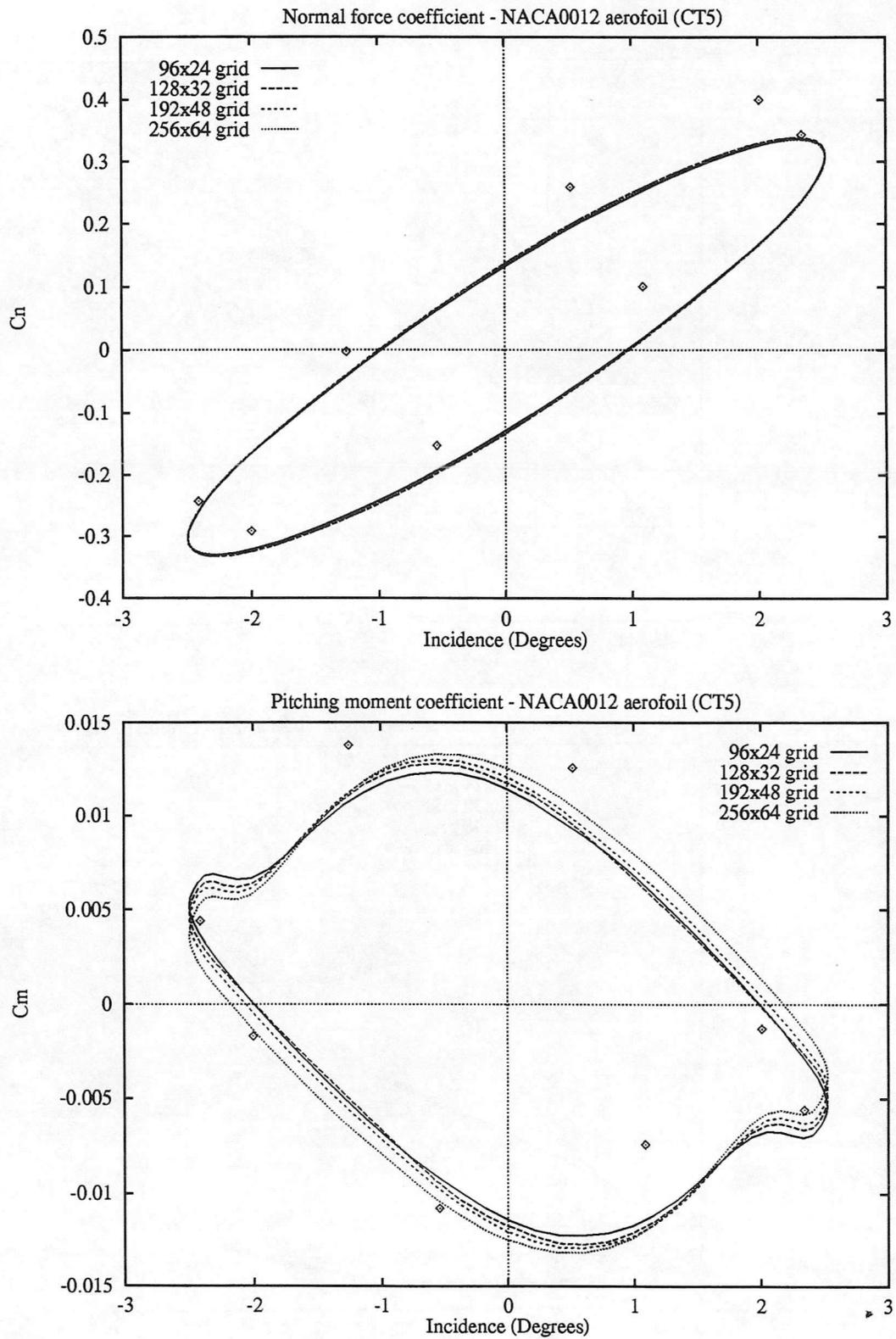


Figure 17: Grid refinement study : Integrated normal force (top) and pitching moment coefficient (bottom) for case CT5

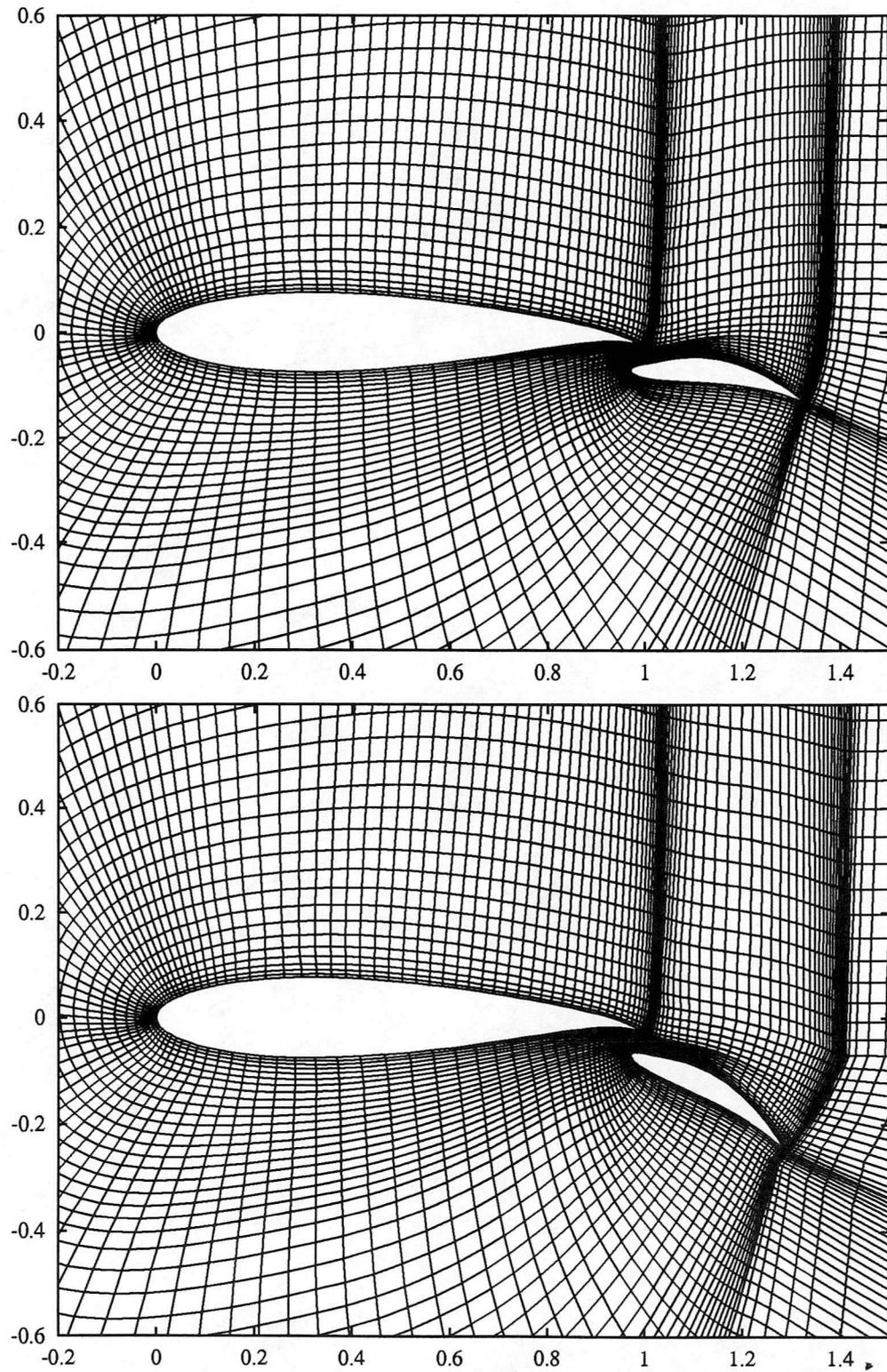


Figure 18: Multi-block grid for the original Williams aerofoil (top) and 20 degrees flap deflection case (bottom)

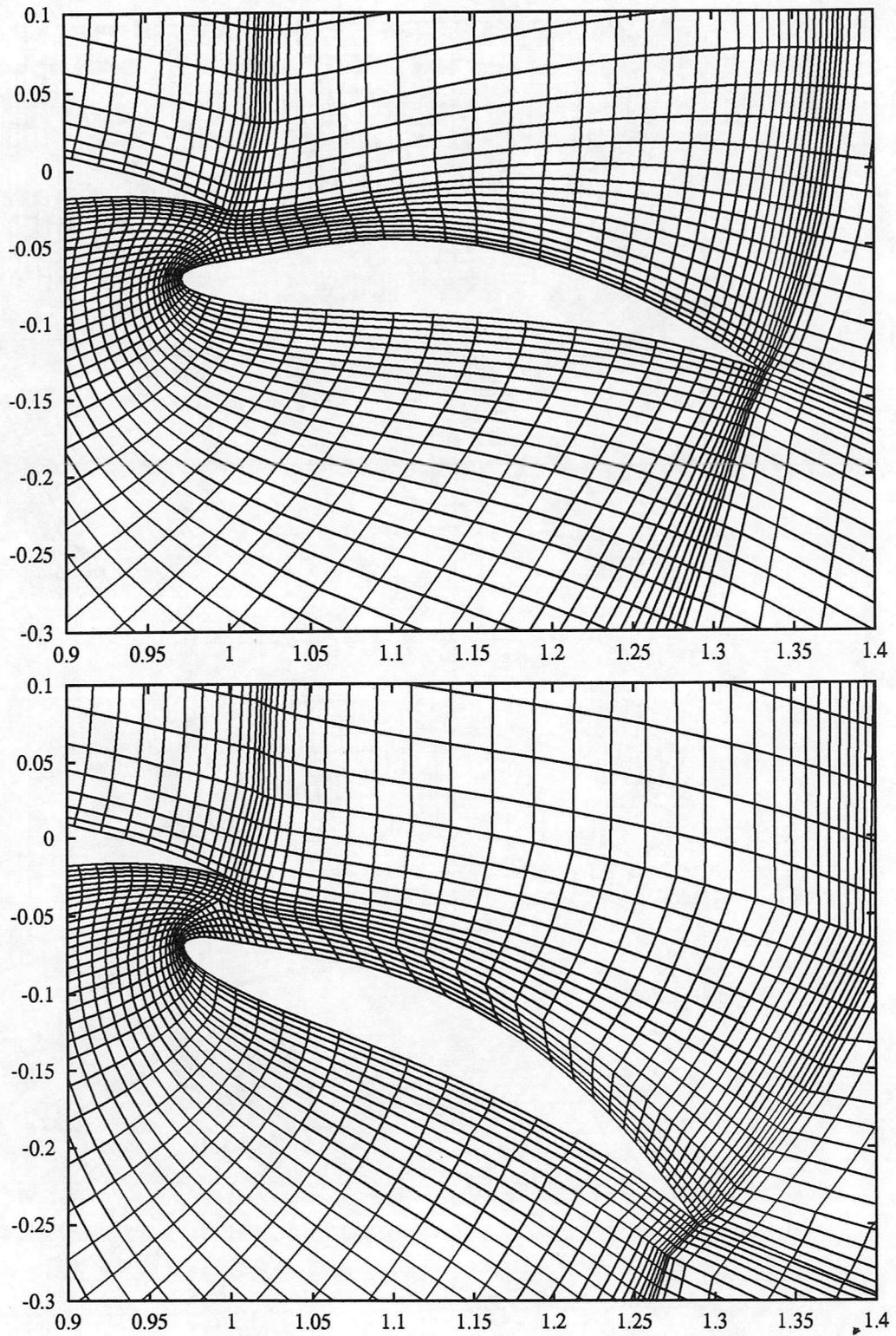


Figure 19: Close-up view of the multi-block grid for the original Williams aerofoil (top) and 20 degrees flap deflection case (bottom)

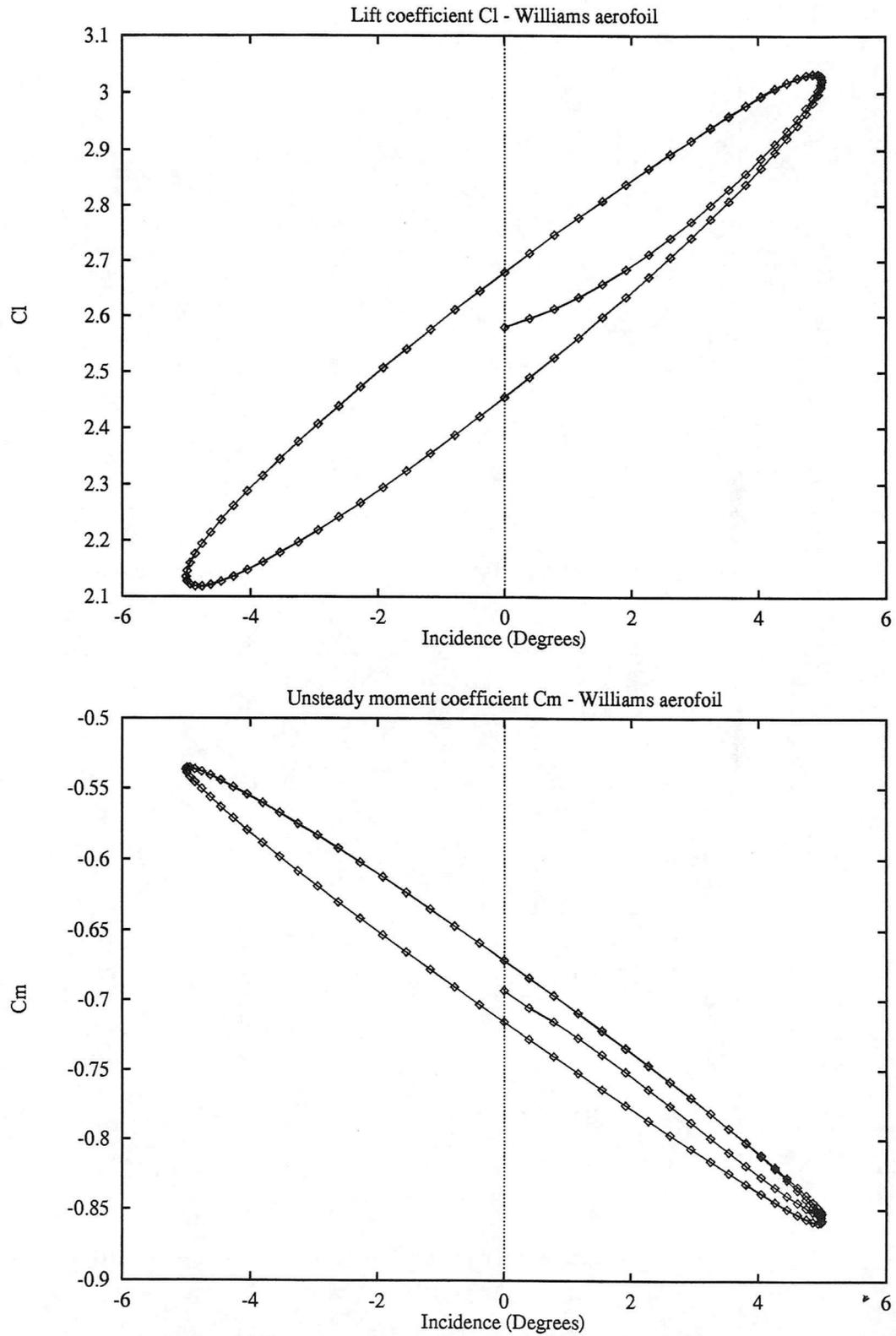
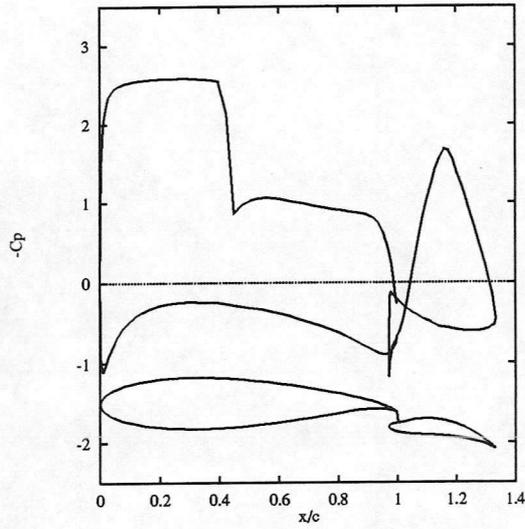
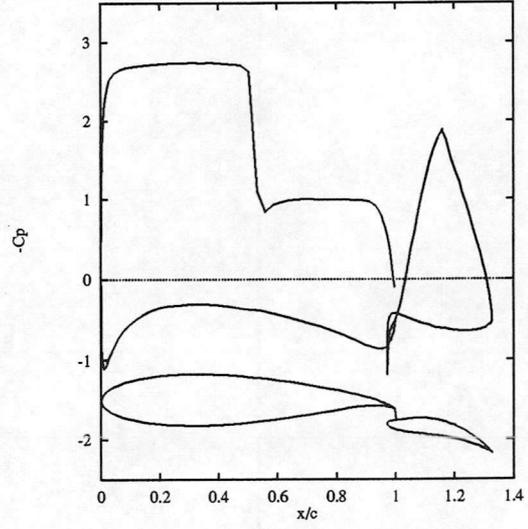


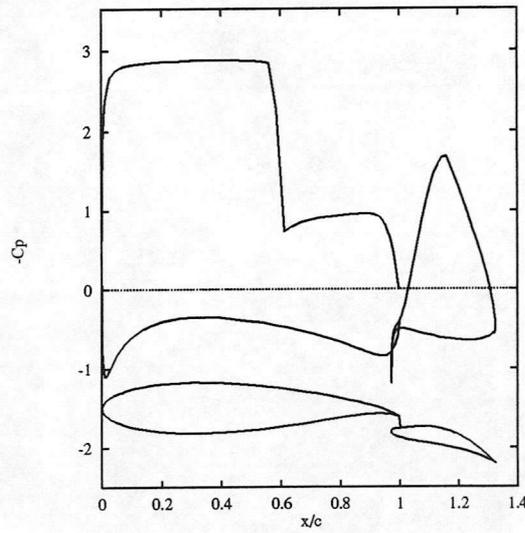
Figure 20: Williams aerofoil : Lift coefficient (top) and pitching moment coefficient (bottom)  
 $M_\infty = 0.58, \alpha = 0^\circ + 5^\circ \sin(\omega t), k = 0.0814$



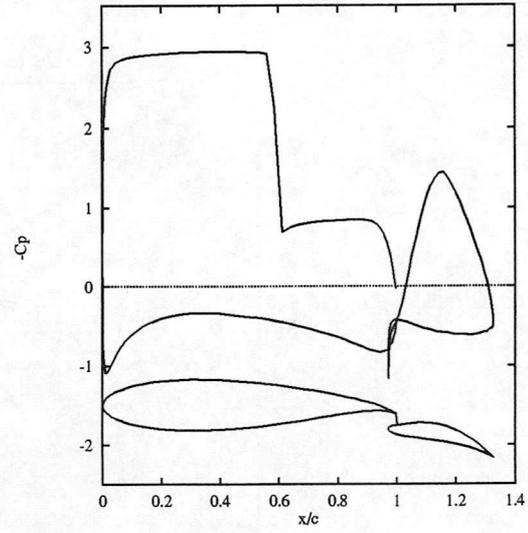
(1)  $\alpha = 0^\circ \nearrow$



(2)  $\alpha = 3.5^\circ \nearrow$

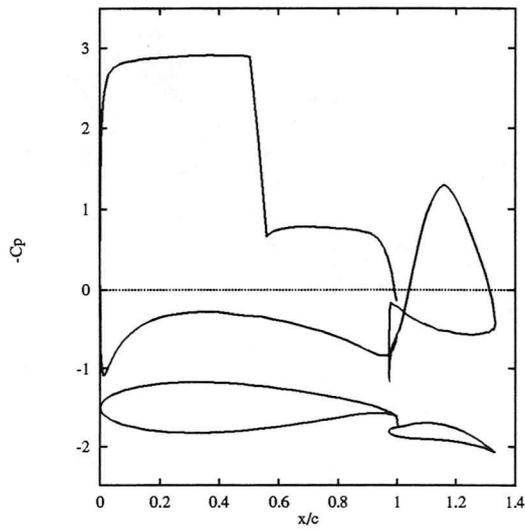


(3)  $\alpha = 5^\circ$

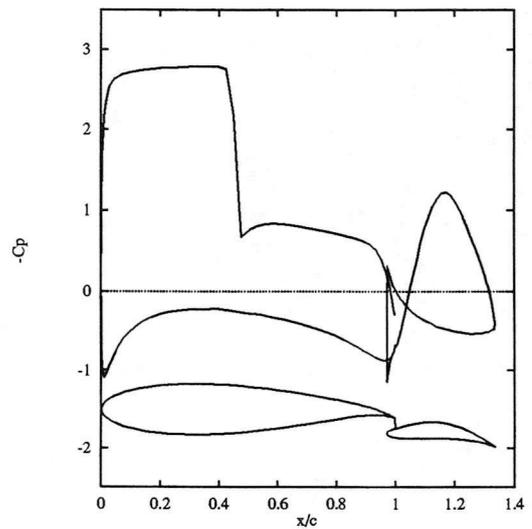


(4)  $\alpha = 3.5^\circ \searrow$

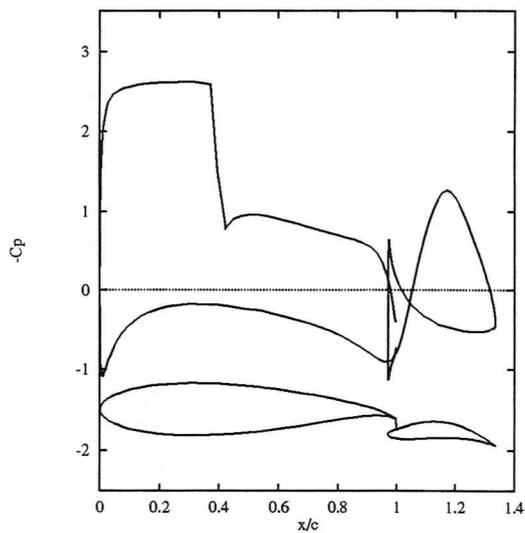
Figure 21: Pressure distribution for Williams aerofoil with oscillating flap  
 $M_\infty = 0.58$ ,  $\alpha = 0^\circ + 5^\circ \sin(\omega t)$ ,  $k = 0.0814$



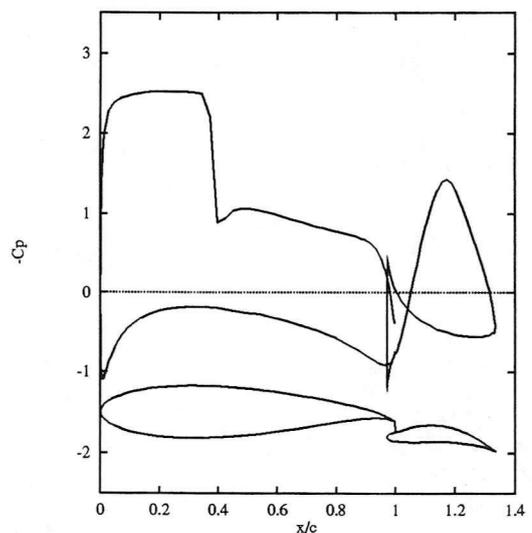
(5)  $\alpha = 0^\circ \searrow$



(6)  $\alpha = -3.5^\circ \searrow$



(7)  $\alpha = -5^\circ$



(8)  $\alpha = -3.5^\circ \nearrow$

Figure 22: Pressure distribution for Williams aerofoil with oscillating flap  
 $M_\infty = 0.58$ ,  $\alpha = 0^\circ + 5^\circ \sin(\omega t)$ ,  $k = 0.0814$

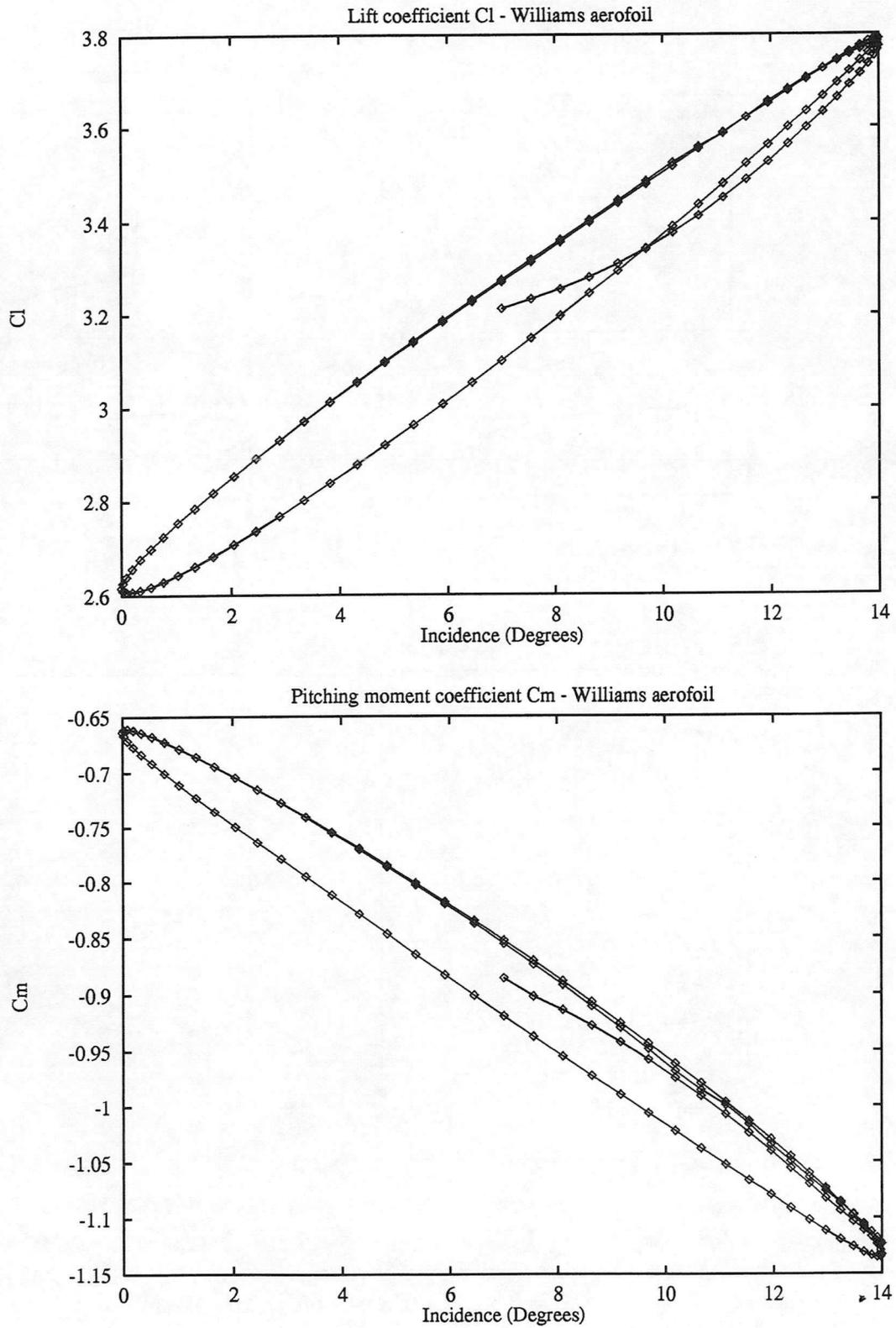
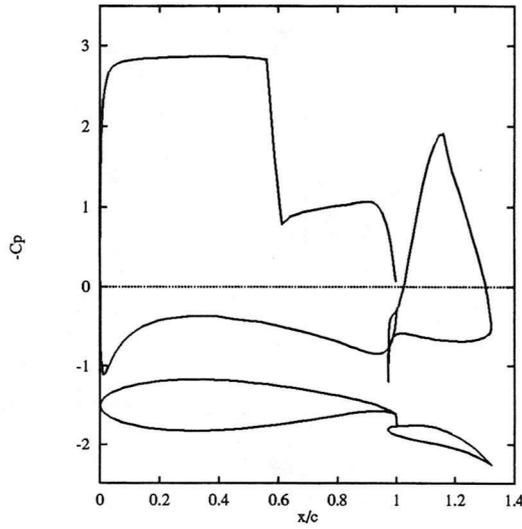
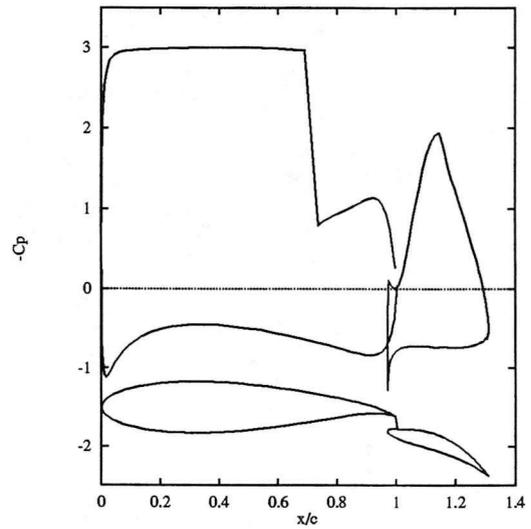


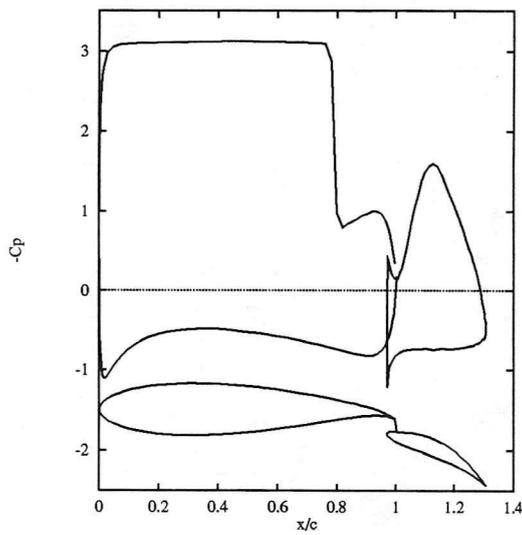
Figure 23: Williams aerofoil : Lift coefficient (top) and pitching moment coefficient (bottom)  
 $M_{\infty} = 0.58$ ,  $\alpha = 7^{\circ} + 7^{\circ} \sin(\omega t)$ ,  $k = 0.0814$



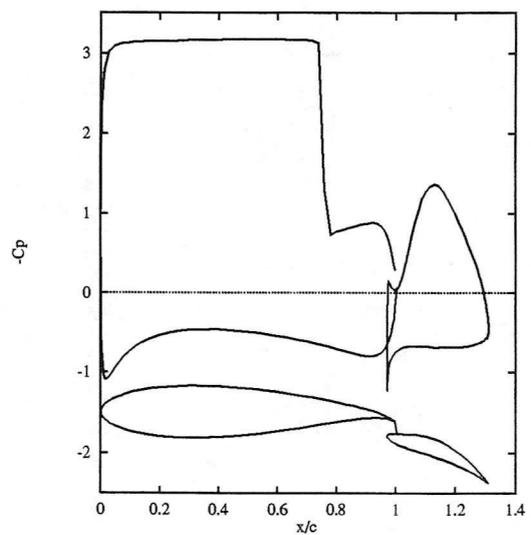
(1)  $\alpha = 7^\circ \nearrow$



(2)  $\alpha = 12^\circ \nearrow$

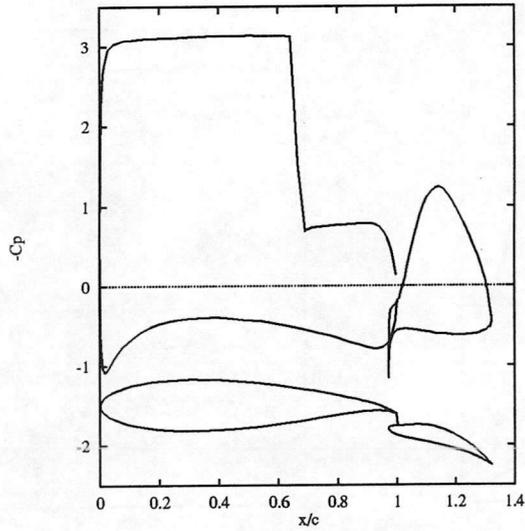


(3)  $\alpha = 14^\circ$

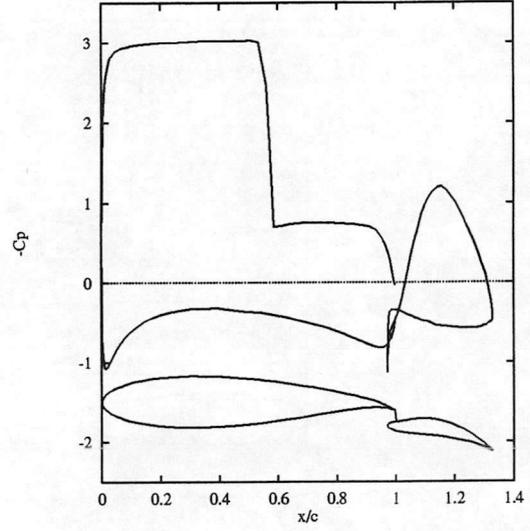


(4)  $\alpha = 12^\circ \searrow$

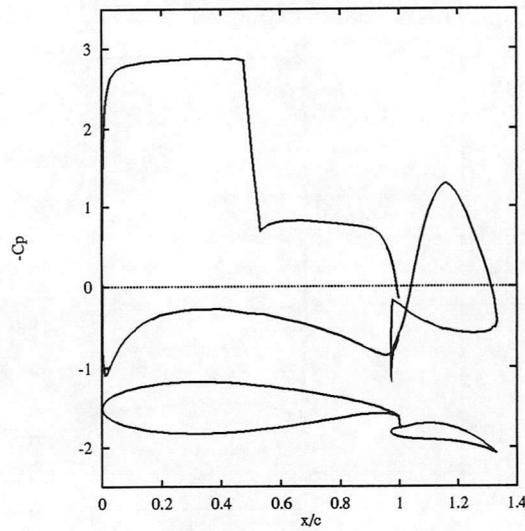
Figure 24: Pressure distribution for Williams aerofoil with oscillating flap  
 $M_\infty = 0.58$ ,  $\alpha = 7^\circ + 7^\circ \sin(\omega t)$ ,  $k = 0.0814$



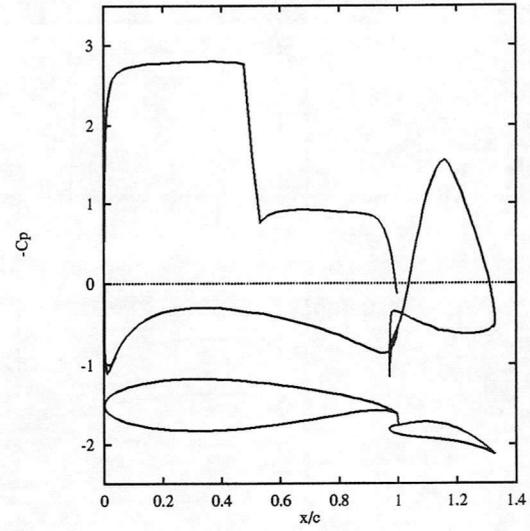
(5)  $\alpha = 7^\circ \searrow$



(6)  $\alpha = 2^\circ \searrow$

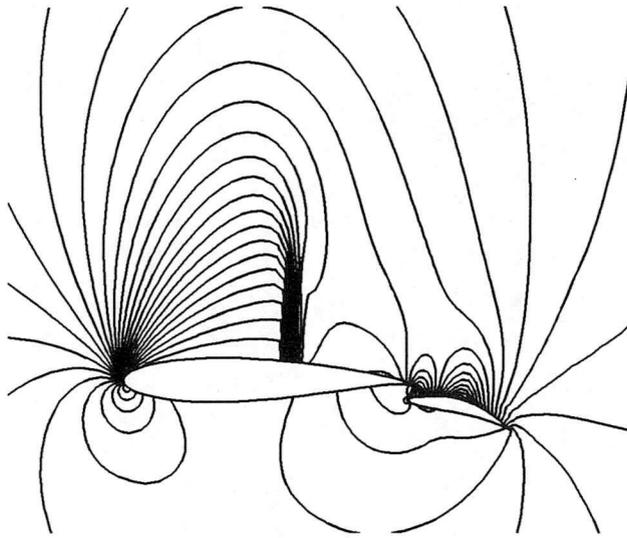


(7)  $\alpha = 0^\circ$

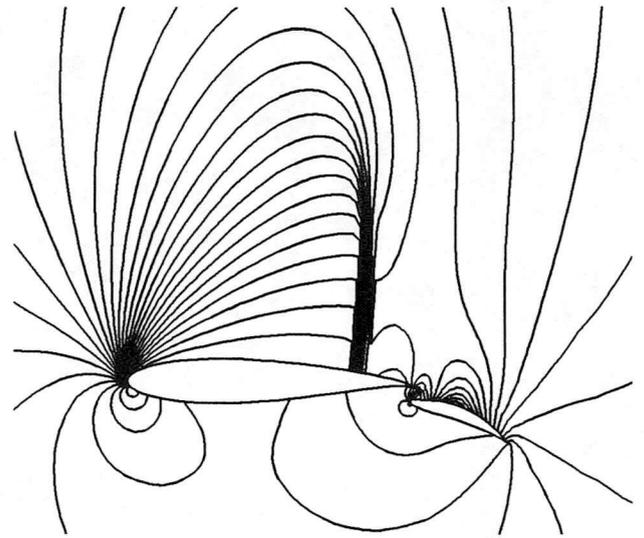


(8)  $\alpha = 2^\circ \nearrow$

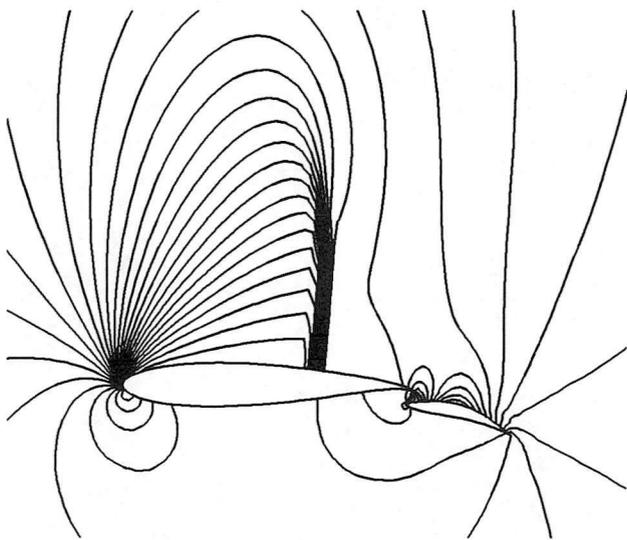
Figure 25: Pressure distribution for Williams aerofoil with oscillating flap  
 $M_\infty = 0.58, \alpha = 7^\circ + 7^\circ \sin(\omega t), k = 0.0814$



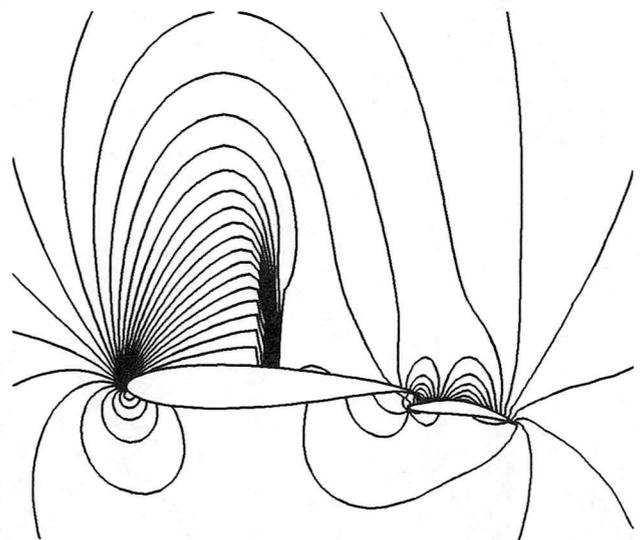
$\alpha = 7.0^\circ \nearrow$



$\alpha = 14.0^\circ$



$\alpha = 7.00^\circ \searrow$



$\alpha = 0.0^\circ$

Figure 26: Pressure contours for Williams aerofoil with oscillating flap  
 $M_\infty = 0.58, \alpha = 7^\circ + 7^\circ \sin(\omega t), k = 0.0814$

