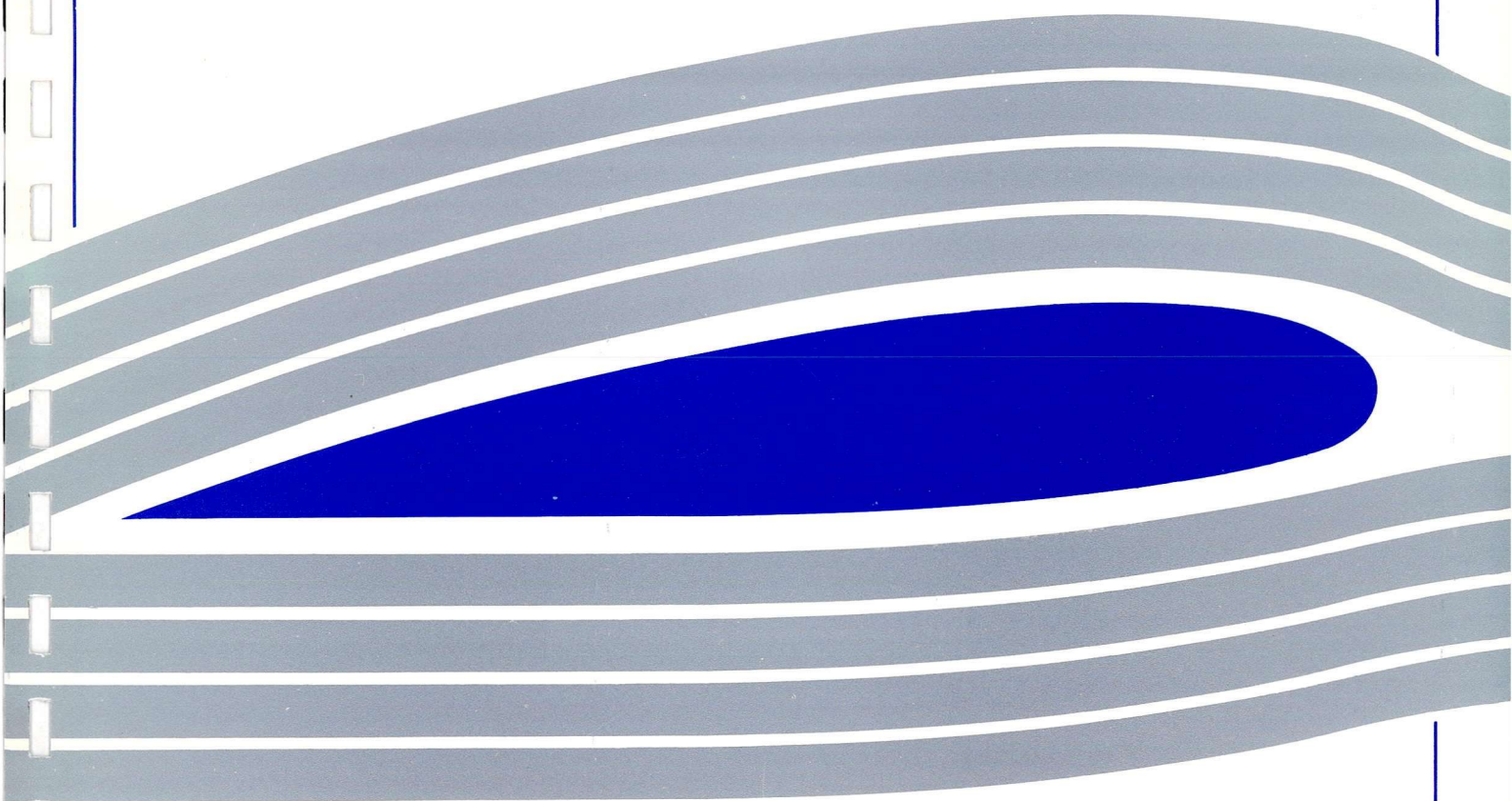
University of Glasgow
DEPARTMENT OF
AEROSPACE
ENGINEERING

Approximate Jacobians for the Solution of the Euler and Navier-Stokes Equations

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

Department of Aerospace Engineering,
University of Glasgow,



Approximate Jacobians for the Solution of the Euler and Navier-Stokes Equations

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

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

Aero. Report 9705

April 25, 1997

Approximate Jacobians for the Solution of the Euler and Navier-Stokes Equations

F. Cantariti, L. Dubuc, B. Gribben, M. Woodgate
K. Badcock and B. Richards
Aerospace Engineering Department
University of Glasgow
Glasgow G12 8QQ

Abstract

This paper describes a method for efficiently solving the steady-state Euler and Navier-Stokes equations. Robustness is achieved through the use of an upwind TVD scheme for discretising the convective terms. The approximate solution is advanced in time implicitly and the linear system arising at each implicit step is solved using a Conjugate Gradient type method. The main emphasis of this paper is on the use of Jacobian matrices associated with a simpler spatial discretisation. This leads to better conditioned linear systems. The resulting method has lower memory and CPU-time requirements when compared with the one using exact Jacobians.

1 Notation

c	aerofoil chord
C_D	drag coefficient
C_L	lift coefficient
C_f	skin friction coefficient
C_p	pressure coefficient
E	total energy per unit mass
$\mathbf{F}, \mathbf{G}$	flux vectors
$\mathbf{H}$	flux matrix
H	total enthalpy per unit mass
h	cell area
$\mathbf{i}, \mathbf{j}$	Cartesian unit vectors
M	Mach number
$\mathbf{n}$	unit outward normal to Ω_s
n	wall normal direction
p	static pressure
Pr	Prandtl number
q_x, q_y	components of heat flux vector
$\mathbf{R}$	vector of residuals
Re	Reynolds number
T	static temperature
t	time
u, v	Cartesian velocity components
$\mathbf{W}$	vector of independent variables
x, y	Cartesian coordinates

Greek Symbols

α	angle of incidence
γ	ratio of specific heats
Δt	time step
$\Delta \mathbf{W}$	vector of conservative updates
μ	molecular viscosity
Ω	flow domain
Ω_s	boundary to the flow domain Ω
ρ	density
τ_{ij}	components of viscous stress tensor

Subscripts

i, j	cell-centre curvilinear coordinates
w	surface condition
∞	freestream conditions

Superscripts

i	convective quantity
ν	diffusive quantity
n	time index

2 Introduction

In the last few years, CFD methods have reached a certain degree of maturity and are being used more routinely for engineering purposes, in conjunction with other traditional techniques such as wind tunnel testing. This is due to several aspects. Firstly, improvements in numerical techniques have led to reduced calculation times. A popular approach

involves multi-stage explicit time-marching schemes, for their good high-frequency damping properties, along with multi-grid acceleration techniques. This approach has been successfully applied to inviscid and viscous flows and results have been widely reported in the literature for both structured and unstructured methods [1], [2], [3], [4], [5], [6]. However, for turbulent flows, there exist some difficulties which are associated with the use of highly stretched meshes and the presence of source terms in the turbulence-transport equations.

Another way is to use implicit schemes to solve the governing flow equations. The most widely used are the Alternate Direction Implicit (ADI) [7] and Approximate Factorisation (AF) [8] techniques. Although these are fairly robust, they suffer from a limitation in the maximum allowable CFL number due to mismatches between explicit and implicit operators and factorisation errors. Hence, some gain can be achieved by trying to solve the linear system arising at each time-step more accurately. Conjugate Gradient (CG) type methods can solve large sparse linear systems efficiently. However, for efficient operation, these techniques require that the eigenvalues of the linear system are clustered around unity, therefore making preconditioning essential. Several preconditioners have been tested, such as ADI preconditioning or Block Incomplete Lower Upper factorisation (BILU) [9], [10], and have been applied to a large variety of steady and unsteady flows [11].

Another reason for the gain in popularity of CFD methods is their increase in accuracy and robustness, requiring less detailed knowledge from the user to set up the calculation and interpret the results. Centred schemes with added artificial dissipation are simple to implement and have been adapted to highly stretched meshes for turbulent flows through the use of ad-hoc scaling functions [4]. However, this approach might break down in the presence of strong shock waves. Modern upwind TVD schemes can overcome this deficiency and have been investigated in detail by Yee [12] for a wide range of flows.

One difficulty associated with implicit methods is the derivation of the Jacobian matrices. This is particularly true when using upwind schemes due to non-linear effects. Although it is possible to derive the exact Jacobian matrices, for example for Roe's scheme [13], it might be more appropriate to use approximated terms to simplify the method. This has been investigated by Barth [13] and was applied to a structured solver by Venkatakrishnan [10], for example. For unstructured methods, approximate Jacobians appear to be the rule due to the increased difficulty in deriving the exact terms [14], [6]. The problem lies in successfully balancing the savings in CPU-time with a possible decrease in stability and reduction in convergence rate.

Following the work done at the University of Glasgow [11], [15], a parallel multi-block Navier-Stokes solver is being developed. The numerical method is presented in the following three sections. Then, our attention is focused on the use of approximate Jacobians for solving both inviscid and viscous flows and computational results are presented for different type of flow conditions. Finally, some conclusion are drawn.

3 Governing Equations

The two-dimensional Cartesian compressible Navier-Stokes equations, written in integral form to facilitate the finite-volume spatial discretisation described below, are given by:

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{W} d\Omega + \oint_{\Omega_s} \mathbf{H}.\mathbf{n} ds = 0 \qquad (1)$$

where Ω is any two-dimensional flow domain, Ω_s is the boundary to this domain and $\mathbf{n}$ is the outward normal vector to Ω_s. The vector $\mathbf{W}$ contains the non-dimensionalised conservative variables:

$$\mathbf{W} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \qquad (2)$$

ρ is the density, (u, v) are the Cartesian velocity components and E is the total energy per unit mass. $\mathbf{H}$ is a matrix containing the flux vectors:

$$\mathbf{H} = \mathbf{F}\mathbf{i} + \mathbf{G}\mathbf{j} \qquad (3)$$

where $\mathbf{i}$ and $\mathbf{j}$ are unit vectors in the two Cartesian directions (x, y). The flux vectors $\mathbf{F}$ and $\mathbf{G}$ are decomposed into convective (^{i}) and viscous diffusive (^{v}) contributions. The convec-

tive parts are equal to:

$$\mathbf{F}^i = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}$$
$$\mathbf{G}^i = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix} \quad (4)$$

where p is the static pressure and H is the total enthalpy per unit mass. The diffusive flux vectors $\mathbf{F}^\nu$ and $\mathbf{G}^\nu$ are given by:

$$\mathbf{F}^\nu = \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + q_x \end{pmatrix}$$
$$\mathbf{G}^\nu = \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} + q_y \end{pmatrix} \quad (5)$$

The components of the stress tensor and of the heat flux vector are modelled in the following way:

$$\begin{aligned} \tau_{xx} &= -\mu \left(2\frac{\partial u}{\partial x} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \right) \\ \tau_{yy} &= -\mu \left(2\frac{\partial v}{\partial y} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \right) \\ \tau_{xy} &= -\mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ q_x &= -\frac{1}{(\gamma - 1)M_\infty^2} \frac{\mu}{Pr} \frac{\partial T}{\partial x} \\ q_y &= -\frac{1}{(\gamma - 1)M_\infty^2} \frac{\mu}{Pr} \frac{\partial T}{\partial y} \end{aligned} \quad (6)$$

In the above relations, γ is the ratio of specific heats, equal to 1.4 for air, Pr is the laminar Prandtl number set to 0.72, T is the static temperature and M_∞ and Re are the freestream Mach number and Reynolds number, respectively. The various flow quantities are related to each other by the perfect gas relations:

$$\begin{aligned} \rho E &= \frac{p}{\gamma - 1} + \frac{\rho}{2}\left(u^2 + v^2 \right) \\ \rho H &= \rho E + p \\ T &= \gamma M_\infty^2 \frac{p}{\rho} \end{aligned} \quad (7)$$

Finally, the laminar viscosity μ is evaluated using Sutherland's law.

4 Boundary Conditions

4.1 Outer Boundary Conditions

At the outer boundary of the computational domain, non-reflecting farfield boundary conditions are determined by using the Riemann invariants for a one-dimensional flow normal to the outer boundary. Furthermore, a vortex correction is introduced to take into account disturbance of the free stream flow by the aerofoil.

4.2 Solid-Wall Boundary Conditions

4.2.1 Inviscid Calculations

For inviscid flows, a single boundary condition needs to be imposed at the wall surface, which is expressed by the slip condition:

$$(u_n)_w = 0 \quad (8)$$

where subscript $_w$ indicates conditions at the wall and n is the wall normal direction. All the other variables, such as the tangential velocity, density and pressure, are extrapolated from interior points using a first order scheme in space.

4.2.2 Viscous Calculations

The surface boundary conditions applied to the Navier-Stokes equations are the no-slip conditions:

$$u_w = v_w = 0 \quad (9)$$

together with the boundary layer approximation:

$$\left(\frac{\partial p}{\partial n} \right)_w = 0 \quad (10)$$

Finally, the wall temperature T_w is given and set equal to the freestream static temperature.

5 Spatial Discretisation

The Navier-Stokes equations are discretised using a cell-centred finite volume approach. The computational domain is divided into a finite number of non-overlapping control-volumes, and the governing equations are applied to each cell in turn. Also, the Navier-Stokes equations are re-written in a curvilinear coordinate system which simplifies the for-

mulation of the discretised terms since body-conforming grids are adopted here.

The spatial discretisation of equation (1) leads to a set of ordinary differential equations in time:

$$\frac{d\mathbf{W}_{i,j}}{dt} = -\mathbf{R}_{i,j} \tag{11}$$

where $\mathbf{R}_{i,j}$ represents the discrete approximation of the convective and viscous flux integrals and subscripts (i,j) are the coordinates of the control-volume in the generalised coordinate system.

The convective terms are discretised in the present work using Osher's upwind scheme [16] for its robustness, accuracy and stability properties. In addition, a MUSCL interpolation is used to provide second-order accuracy and the van Albada limiter prevents spurious oscillations from occurring around shock waves.

The discretisation of the viscous terms requires the value of the velocity components and their derivatives, as well as the derivatives of the static temperature, at the edges of each cell. Cell-edge values of the velocity components are approximated by the average of the two adjacent cell-centre values, as shown below:

$$u_{i+\frac{1}{2},j} = \frac{1}{2}\left(u_{i,j} + u_{i+1,j}\right) \tag{12}$$

Cell-edge values of the derivatives are obtained using Green's formula applied to an auxiliary cell surrounding the considered edge (see Figure 1), for example:

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{1}{h_{aux}} \oint_{\Omega_{saux}} u dy \\ \frac{\partial u}{\partial y} &= \frac{-1}{h_{aux}} \oint_{\Omega_{saux}} u dx \end{aligned} \tag{13}$$

where h_{aux} is the area of the auxiliary cell. The values at the four points a, b, c, d are obtained using the neighbouring cell-centre values:

$$\begin{aligned} u_a &= u_{i,j} \\ u_b &= \frac{u_{i,j-1} + u_{i,j} + u_{i+1,j-1} + u_{i+1,j}}{4} \\ u_c &= u_{i+1,j} \\ u_d &= \frac{u_{i,j} + u_{i,j+1} + u_{i+1,j} + u_{i+1,j+1}}{4} \end{aligned} \tag{14}$$

The choice of the auxiliary cell is guided by the need to avoid odd-even point decoupling and to minimise the amount of numerical viscosity introduced in the discretised equations. Finally, the discretisation of the viscous terms at the wall uses a one-sided auxiliary cell.

6 Implicit Time-Marching Scheme

In the present work, the integration in time of equation (11) to a steady-state solution is performed using an implicit time-marching scheme:

$$\frac{\mathbf{W}^{n+1} - \mathbf{W}^n}{\Delta t} = -\mathbf{R}^{n+1} \tag{15}$$

where n is the current time level, $n+1$ is the new time level and subscripts (i,j) are neglected for clarity. The above equation represents a system of non-linear algebraic equations and to simplify the solution procedure, the flux residual $\mathbf{R}^{n+1}$ is linearised in time as follows:

$$\begin{aligned} \mathbf{R}^{n+1} &= \mathbf{R}^n + \frac{\partial \mathbf{R}}{\partial t}\Delta t + O(\Delta t^2) \\ &\approx \mathbf{R}^n + \frac{\partial \mathbf{R}}{\partial \mathbf{W}}\frac{\partial \mathbf{W}}{\partial t}\Delta t \\ &\approx \mathbf{R}^n + \frac{\partial \mathbf{R}}{\partial \mathbf{W}}\Delta W \end{aligned} \tag{16}$$

where $\Delta\mathbf{W} = \mathbf{W}^{n+1} - \mathbf{W}^n$. Equation (15) now becomes the following linear system:

$$\left(\frac{I}{\Delta t} + \frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right)\Delta\mathbf{W} = -\mathbf{R}^n \tag{17}$$

The complexity of a direct method to compute a linear system is of the order of $\mathcal{N}^3$, which becomes prohibitive when the total number of equations $\mathcal{N}$ becomes large. On the other hand, iterative techniques such as Conjugate Gradient (CG) methods are capable of solving large systems of equations more efficiently in terms of time and memory [15]. CG methods find an approximation to the solution of a linear system by minimising a suitable residual error function in a finite-dimensional space of potential solution vectors. Several algorithms, such as BiCG, CGSTAB, CGS and GMRES, have been tested in [9] and it was concluded that the choice of method is not as crucial as the preconditioning. The current results use a Generalised Conjugate Gradient method [17].

Several preconditioners have been investigated in [9] and in the present method, the

preconditioning strategy is based on a Block Incomplete Lower-Upper factorisation since it appears to be the most promising. The sparsity pattern of the Lower and Upper matrices is defined with respect to the sparsity of the unfactored matrix for simplicity. However, good performance is obtained regarding the convergence of the CG method, regardless of the sparsity pattern adopted [18].

Implicit schemes require particular treatment during the early stages of the iterative procedure. The usual approach in starting the method is to take a small CFL number and to increase it later on. However, it was found that smoothing out the initial flow doing some explicit iterations, and then switching to the implicit algorithm was equally efficient. In the present method, 200 forward Euler iterations are executed before switching to the implicit scheme.

7 Jacobians for Inviscid Calculations

As mentioned in Section (5), the inviscid fluxes are calculated using an upwind scheme. Hence, the numerical flux across an edge depends on the values of the flow variables on either side of the edge. For example, for the interface between cell (i,j) and cell $(i+1,j)$:

$$\mathbf{H}_{i+\frac{1}{2},j} = \mathbf{H}\left(\mathbf{W}^{+}_{i+\frac{1}{2},j}, \mathbf{W}^{-}_{i+\frac{1}{2},j}\right) \qquad (18)$$

where the left and right states are extrapolated using a MUSCL interpolation, leading to the following relations:

$$\begin{aligned}\mathbf{W}^{-}_{i+\frac{1}{2},j} &= \mathcal{F}\left(\mathbf{W}_{i-1,j}, \mathbf{W}_{i,j}, \right. \\ &\qquad \left. \mathbf{W}_{i+1,j}, \mathbf{W}_{i+2,j}\right) \\ \mathbf{W}^{+}_{i+\frac{1}{2},j} &= \mathcal{G}\left(\mathbf{W}_{i-1,j}, \mathbf{W}_{i,j}, \right. \\ &\qquad \left. \mathbf{W}_{i+1,j}, \mathbf{W}_{i+2,j}\right) \end{aligned} \qquad (19)$$

As a result, the flux residual for cell (i,j) is a function of nine points (see Figure 2):

$$\begin{aligned}\mathbf{R}_{i,j} = \mathbf{R}\left(\right. &\mathbf{W}_{i-2,j}, \mathbf{W}_{i-1,j}, \mathbf{W}_{i,j}, \\ &\mathbf{W}_{i+1,j}, \mathbf{W}_{i+2,j}, \mathbf{W}_{i,j-2}, \\ &\left.\mathbf{W}_{i,j-1}, \mathbf{W}_{i,j+1}, \mathbf{W}_{i,j+2}\right)\end{aligned} \qquad (20)$$

The above formulation for $\mathbf{R}_{i,j}$ leads to a Jacobian matrix $\partial\mathbf{R}/\partial\mathbf{W}$ which has nine non-zero entries per row. However, trying to reduce the number of non-zero blocks would have several advantages. Firstly, the memory requirements are lowered. Secondly, the resolution of the linear system by the CG method is faster in terms of CPU-time since all the matrix-vector multiplications involved require less operation counts. Finally, the linear system is easier to solve since the approximate Jacobian matrix is more diagonally dominant.

An approximation to the exact Jacobian arises from neglecting the influence of the MUSCL interpolation:

$$\begin{aligned}\mathbf{W}^{-}_{i+\frac{1}{2},j} &= \mathcal{F}'\left(\mathbf{W}_{i,j}\right) \\ \mathbf{W}^{+}_{i+\frac{1}{2},j} &= \mathcal{G}'\left(\mathbf{W}_{i+1,j}\right)\end{aligned} \qquad (21)$$

The flux residual now becomes a function of only five points (see Figure 3):

$$\begin{aligned}\mathbf{R}_{i,j} = \mathbf{R}'\left(\right. &\mathbf{W}_{i-1,j}, \mathbf{W}_{i,j}, \mathbf{W}_{i+1,j}, \\ &\left.\mathbf{W}_{i,j-1}, \mathbf{W}_{i,j+1}\right)\end{aligned} \qquad (22)$$

This approximation, which is applied only for the derivation of the Jacobian terms, reduces memory requirements and matrix-vector multiplication operation counts to 5/9 of the values using the exact Jacobians.

In this paper, all the calculations were made around the NACA 0012 aerofoil. The computational grids used have a "C" topology and consist of 3 blocks. For inviscid flows, two meshes were generated. The first one contains 97 points along the aerofoil, 16 in the wake and 33 in the direction normal to the aerofoil. The wall distance of the first node is set to $0.003c$ at the leading-edge and to $0.01c$ at the trailing edge, where c is the aerofoil chord. The outer boundary is situated approximately 15 chords away from the aerofoil. The second grid, used for a grid dependency study, was generated by doubling the number of points in each direction and halving the wall distance of the first node.

The first test case considered is a subsonic flow at a Mach number of 0.3 and an angle of attack of 10^o. A typical pressure distribution obtained with the present method is shown in Figure 4 and compares favourably with a potential flow solution [19]. Also, there exists very few differences between the coarse and fine grid solutions, apart from the suction peak near the aerofoil leading-edge.

Figure 5 shows the convergence of the residual, measured by the L^2 norm, versus the

number of work units. A work unit is defined as the CPU-time required to perform one explicit Euler iteration.

When using the exact Jacobian matrix, it was found that there is little to gain in increasing the CFL number above a value of 60 (see Figure 5). As shown in Table 1, this is due to the fact that both the number of implicit steps and the average number of CG iterations needed to solve the linear system at each time step remain almost constant for CFL numbers greater than 60. In any case, it was not possible to start the implicit scheme for CFL values greater than 100 after only 200 explicit iterations for the exact problem. A limit on the size of the starting CFL number arises from the fact that the implicit scheme requires to converge an initial guess which is getting closer to the final solution for increasingly large CFL numbers.

Several conclusions can be drawn from a comparison between the convergence histories of the two approaches. In particular, it is found that when solving the approximate as opposed to the exact problem:

- for the same CFL number, more time steps are required to reach the converged solution. This was expected since at each implicit step, the updates $\Delta \mathbf{W}$ are not a solution of the exact linear system.
- the method seems more robust because higher starting CFL numbers can be used.
- less CG iterations need to be performed at each implicit step due to the fact that the linear system is easier to solve (see Table 1).
- if the CFL number is increased, the number of CG iterations remains roughly constant.

Hence, the total number of work units required to reach a converged solution is reduced by a factor of four when using approximate Jacobians, as shown in Figure 5.

The next test case is a transonic flow at a Mach number of 0.8 and an incidence of 1.25^o. Figure 6 compares a typical pressure distribution obtained with the present method with an inviscid solution [20]. The overall agreement is good, in particular in the shock wave positions and strengths on the upper and lower surfaces. The present results show an under-prediction of the pressure recovery at the trailing edge. This is due to the limiter being "switch-on" in that region and reducing locally the accuracy of the solution. However, this seems a feature of TVD schemes and can also be found in other published results [10]. Figure 6 also shows the pressure distribution for the fine grid. The coarse and fine grid results agree well except near the shocks on the upper and lower surfaces where, as expected, the pressure jumps are more sharply captured on the denser grid. The integrated loads are compared in Table 2 and there is good agreement amongst all the computational results. Concerning the convergence rate, the same conclusions as for the previous case hold here, although the improvement in using approximate Jacobians is now slightly reduced to a factor of three, as seen in Figure 7. This is probably due to the presence of a supersonic flow region, as argued below.

The final inviscid test case is a supersonic flow at a Mach number of 1.8 and an angle of attack of 7^o. The predicted pressure contours are shown in Figure 8. The oscillations in the pressure contours are due to the fact that the computational grid is not aligned with the bow shock. Thus, the propagation of the shock is not treated correctly by Osher's scheme, since it is a one-dimensional approximate Riemann solver. However, such oscillations are also present in Yee's results [12] on a finer grid. The only way to eliminate these oscillations is to adapt the grid to the flow field. Figure 9 compares the present results on the coarse and fine grids and good agreement is found between the two sets of data. The maximum starting CFL numbers for the exact and approximate problems are 10 and 50, respectively. Table 3 shows that the approximate approach requires more implicit steps to converge for all CFL values. Also, the sharp decrease in the total number of work units observed in the two previous test cases is not present here (see Figure 10). This is probably due to the fact that, for supersonic flows, the upwind scheme only selects the upstream points for the calculation of the flux. Therefore, the number of non-zero blocks per row is reduced in the Jacobian matrix and the exact linear system is much easier to solve in this case than for subsonic or transonic flows. This is confirmed in Table 3 where the number of CG iterations per implicit step is reduced from 10-16 for the previous cases to 4. In addi-

Method	Implicit steps to convergence	Average number of CG iterations per implicit step
CFL number = 60		
Exact Jacobians	391	17
Approximate Jacobians	450	4
CFL number = 100		
Exact Jacobians	387	16
Approximate Jacobians	376	5
CFL number = 250		
Exact Jacobians	-	-
Approximate Jacobians	308	5

Table 1: Convergence history details
NACA 0012 aerofoil
$M_\infty = 0.3$, $\alpha = 10^o$

	Grid type	Number of points	C_L	C_D
Salas [20]	O	192 × 39	0.3474	0.0221
Jameson [20]	O	320 × 64	0.3632	0.0230
Present method	C	257 × 65	0.3557	0.0228
Present method	C	129 × 33	0.3542	0.0240

Table 2: Integrated load coefficients
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 1.25^o$

Method	Implicit steps to convergence	Average number of CG iterations per implicit step
CFL number = 10		
Exact Jacobians	393	4
Approximate Jacobians	523	2
CFL number = 25		
Exact Jacobians	-	-
Approximate Jacobians	472	2
CFL number = 50		
Exact Jacobians	-	-
Approximate Jacobians	455	3

Table 3: Convergence history details
NACA 0012 aerofoil
$M_\infty = 1.8$, $\alpha = 7^o$

tion, the exact linear system is so easy to solve that very little is gained in terms of CG iterations by using approximate Jacobians. Nevertheless, the overall cost for solving the exact problem still remains slightly higher than that for the approximate one because of the added number of CG iterations and the extra operations involved in any matrix-vector multiplication.

8 Jacobians for Viscous Calculations

The discretisation of the viscous terms as described in Section 5 leads to a viscous flux residual which is a function of the following nine points: $\mathbf{W}_{i-1,j-1}$, $\mathbf{W}_{i,j-1}$, $\mathbf{W}_{i+1,j-1}$, $\mathbf{W}_{i-1,j}$, $\mathbf{W}_{i,j}$, $\mathbf{W}_{i+1,j}$, $\mathbf{W}_{i-1,j+1}$, $\mathbf{W}_{i,j+1}$ and $\mathbf{W}_{i+1,j+1}$ (see Figure 11). An exact derivation of the inviscid and viscous Jacobians together would involve four more terms in addition to the nine above: $\mathbf{W}_{i-2,j}$, $\mathbf{W}_{i+2,j}$, $\mathbf{W}_{i,j-2}$ and $\mathbf{W}_{i,j+2}$.

However, in view of the computational results presented in the previous section, it seems more interesting from a storage and CPU-time point of view to derive an approximate formulation for the viscous Jacobians based on equation (22). Indeed, such an approach would give savings of 8/13 for the memory requirements and any matrix-vector multiplication operation counts. A simple approximation results from taking into account only the influence of the two points situated either side of the considered edge during the calculation of the viscous flux across a cell interface. For example, in the case shown in Figure 1, the contributions of $\mathbf{W}_{i,j-1}$, $\mathbf{W}_{i,j+1}$, $\mathbf{W}_{i+1,j-1}$ and $\mathbf{W}_{i+1,j+1}$ are neglected and only the terms arising from $\mathbf{W}_{i,j}$ and $\mathbf{W}_{i+1,j}$ are kept. This amounts to making a thin layer approximation for the derivation of the viscous Jacobians in the direction normal to the edge.

As for the inviscid study, three different calculations were made over a range of Mach numbers. For all cases, the coarse computational grids consists of 97 points along the aerofoil, 48 in the wake and 49 in the direction normal to the aerofoil. The fine meshes have 129 points along the aerofoil, 64 in the wake and 65 in the direction normal to the aerofoil. For all the grids, the outer boundary is situated 15 chords away from the aerofoil. The wall distance of the first node for the coarse grids varies from $0.0005c$ at the leading-edge to $0.0007c$ at the trailing-edge for the first two calculations, while for the last case, it is set at the leading- and trailing-edge to $0.002c$ and $0.005c$, respectively. For the fine grids, the wall distance of the first node is half that on the coarse grids.

The first viscous case is a subsonic flow at a Mach number of 0.5, an incidence of 0^o and a Reynolds number of 5000 based on the aerofoil chord. This is a difficult case to predict accurately because of the presence of a small recirculation region near the aerofoil trailing-edge. An overview of the computed solutions can be seen in Figures 12 to 14. The present results are in good agreement with those of Mavripilis and Jameson [21], although the skin friction peak near the aerofoil leading-edge is underestimated on the coarse grid. The predicted coordinate of the separation point of 0.82 is in good agreement with other computational results [21]. The convergence history is shown in Figure 15 and indicates that the method converges reasonably well. In particular, the use of approximate viscous Jacobians does not destabilize the method and similar CFL values to inviscid subsonic cases can be used.

The second test case is a transonic flow at a Mach number of 0.8, an angle of attack of 10^o and a Reynolds number of 500 based on the aerofoil chord. The pressure and skin friction distributions as well as the Mach number contours are plotted in Figures 16 to 18. The present results compare well with those of Muller *et al* [22] except near the aerofoil leading-edge where, again, the skin friction variations are not too well captured on the coarse grid. Although there exists a large recirculation region on the upper surface and downstream of the aerofoil, the convergence rate is good (see Figure 19), and a converged solution is reached in less than 1000 work units.

The final case is a supersonic flow at a Mach number of 2, an angle of attack of 10^o and a Reynolds number of 106 based on the aerofoil chord. The pressure and skin friction distributions and the density contours are plotted in Figures 20 to 22, while the convergence history is given in Figure 23. Again, it was found that the method converges rapidly even when using approximate Jacobians.

9 Conclusions

A method capable of solving the Euler and Navier-stokes equations has been described. These equations are marched in time using an implicit scheme and, at each implicit step, a linear system is solved iteratively using a Conjugate Gradient type algorithm. Our main attention was focused on the derivation and use of approximate Jacobian terms. It has been found that this approach leads to substantial savings in terms of memory and CPU-time requirements for a wide range of flow conditions.

Future work will extend the use of approximate Jacobians to the solution of turbulent-transport equations.

Acknowledgements

This work has been partially supported by a grant under the OST LINK programme Grant DRA ASF/2351U and a joint EPSRC/MOD Scheme Grant GR/K55455.

References

[1] Martinelli, L., Jameson, A. and Grasso, F., A Multigrid Method for the Navier-Stokes Equations, *AIAA Paper 86-0208*, 1986.

[2] Martinelli, L. and Jameson, A., Validation of a Multigrid Method for the Reynolds Averaged Equations, *AIAA Paper 88-0414*, 1988.

[3] Chima, R., Turkel, E. and Schaffer, S., Comparison of Three Explicit Multigrid Methods for the Euler and Navier-Stokes Equations, *AIAA Paper 87-0602*, 1987.

[4] Radespiel, R. and Swanson, R., An Investigation of Cell Centred and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations, *AIAA Paper 89-0548*, 1989.

[5] Mavripilis, D. and Martinelli, L., Multigrid Solution of Compressible Turbulent Flow on Unstructured Meshes Using a Two-Equation Model, *AIAA Paper 91-0237*, 1991.

[6] Lallemand, M., Steve, H. and Dervieux, A., Unstructured Multigridding by Volume Agglomeration: Current Status, *Computers Fluids, vol 21, No 3, pp 397-443*, 1992.

[7] Peaceman, D. and Rachford, H., The numerical Solution of Parabolic and Elliptic Differential Equations, *Journal of the Society for Industrial and Applied Mathematics, vol 3, No 1, pp 28-41*, 1955.

[8] Beam, R. and Warming, R., An Implicit Factored Scheme for the Compressible Navier-Stokes Equations, *AIAA Journal, vol 16, No 4, pp 393-402*, 1978.

[9] Badcock, K., Dubuc, L. and Richards, B., Preconditioners for High Speed Flows in Aerospace Engineering, Numerical Methods for Fluid Dynamics V, pp 287-294, Oxford University Press, 1996.

[10] Venkatakrishnan, V., Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations, *AIAA Journal, vol 29, No 7, pp 1092-1100*, 1991.

[11] Badcock, K. and Gaitonde A., An Unfactored Implicit Moving Mesh Method for the Two-Dimensional Unsteady N-S Equations, *International Journal for Numerical Methods in Fluids, vol 23, pp 607-631*, 1996.

[12] Yee, H., A Class of High Resolution Explicit and Implicit Shock Capturing Methods, Von Karman Institute for Fluid Dynamics, Lecture Series 1989-04, 1989.

[13] Barth, T., Analysis of Implicit Local Linearization Technique for Upwind and TVD algorithms, *AIAA Paper 87-0595*, 1987.

[14] Venkatakrishnan, V., Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters, *Journal of computational Physics, vol 118, pp 120-130*, 1995.

[15] Badcock, K. and Richards, B., Parallel Implicit Upwind Methods for the Aerodynamics of Aerospace Vehicles, Progress and Challenges in CFD Methods and Algorithms, AGARD CP-578, 1995.

[16] Osher, S. and Chakravarthy, S., Upwind Schemes and Boundary Conditions with

Applications to Euler Equations in General Geometries, *Journal of Computational Physics, vol 50, pp 447-481*, 1983.

[17] Axelsson, O., Iterative Solution Methods, Cambridge University Press, 1994.

[18] K. Badcock, W. McMillan, M. Woodgate, B. Gribben, S. Porter and B. 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.

[19] Hirsch, C., Computational Methods for Inviscid and Viscous Flows, Numerical Computation of Internal and External Flows, Volume 2, John Willey, 1990.

[20] AGARD Advisory Report No 211, Test Cases for Inviscid Flow Field Methods, 1985.

[21] Mavripilis, D. and Jameson, A., Multigrid Solution of the Navier-Stokes Equations on Triangular Meshes, *AIAA Journal, vol 28, No 8, pp 1415-1425*, 1990.

[22] Muller, B., Berglind, T. and Rizzi, A., Implicit Central Difference Simulation of Compressible Navier-Stokes Flow Over a NACA0012 Airfoil, Numerical Simulation of compressible Navier-Stokes Flows, Notes on Numerical Fluid Mechanics, Bristeau M. O. *et al.* (Eds.), Volume 18, Vieweg, 1987.

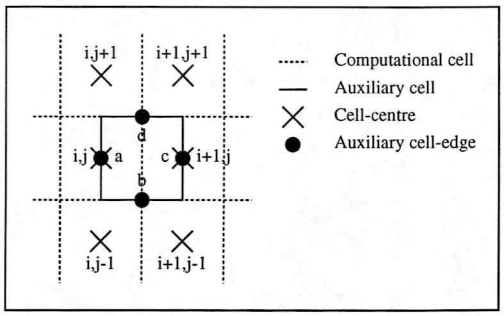


Figure 1: Typical auxiliary cell for viscous flux evaluation

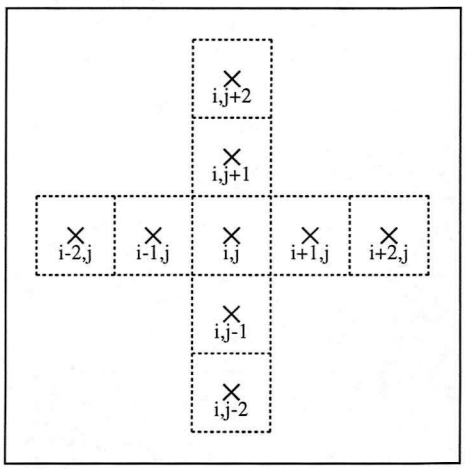


Figure 2: Flux residual dependency for an exact derivation of the inviscid terms

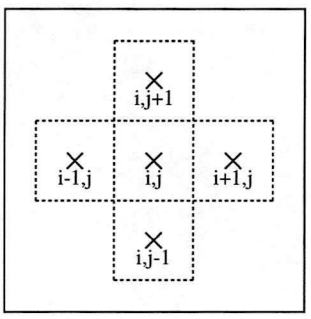


Figure 3: Flux residual dependency for an approximate derivation of the inviscid terms

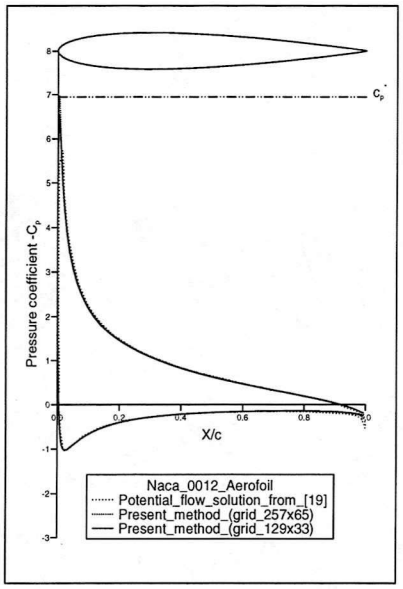


Figure 4: Typical pressure distribution
NACA 0012 aerofoil
$M_\infty = 0.3$, $\alpha = 10^o$

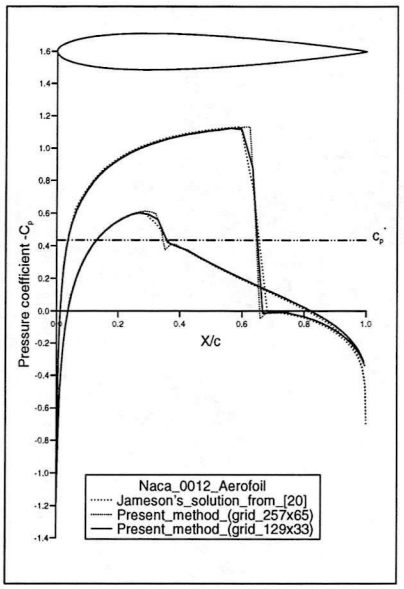


Figure 6: Typical pressure distribution
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 1.25^o$

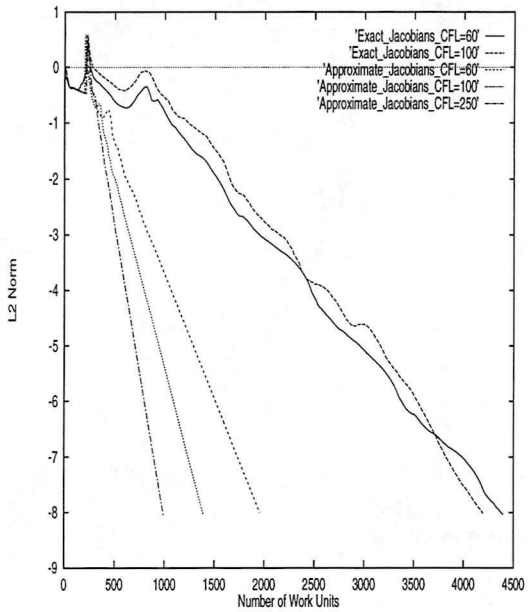


Figure 5: Convergence history
NACA 0012 aerofoil
$M_\infty = 0.3$, $\alpha = 10^o$

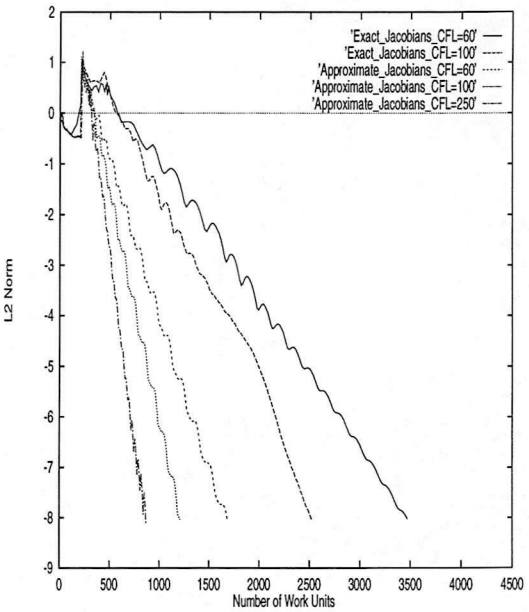


Figure 7: Convergence history
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 1.25^o$

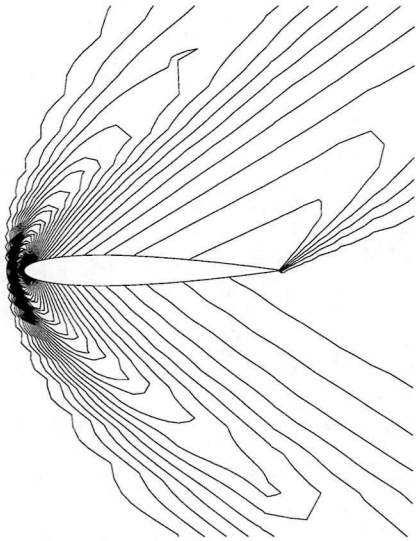


Figure 8: Typical pressure contours
NACA 0012 aerofoil
$M_\infty = 1.8$, $\alpha = 7^o$

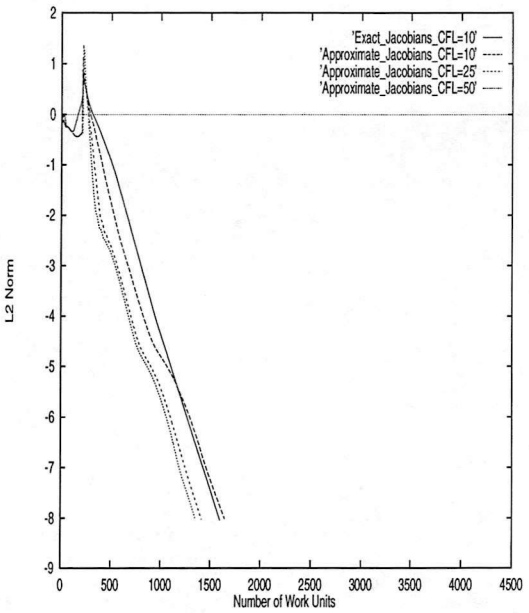


Figure 10: Convergence history
NACA 0012 aerofoil
$M_\infty = 1.8$, $\alpha = 7^o$

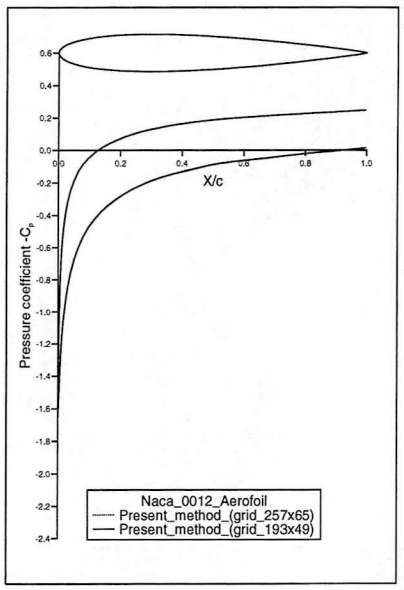


Figure 9: Typical pressure distribution
NACA 0012 aerofoil
$M_\infty = 1.8$, $\alpha = 7^o$

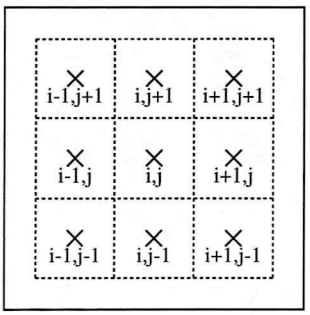


Figure 11: Flux residual dependency for an exact derivation of the viscous terms

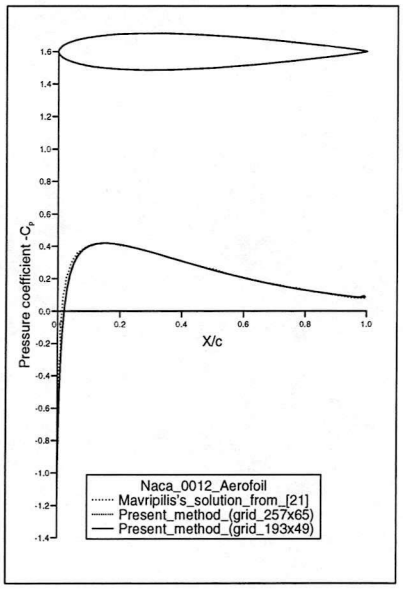


Figure 12: Pressure distribution
NACA 0012 aerofoil
$M_\infty = 0.5$, $\alpha = 0^o$, Re = 5000

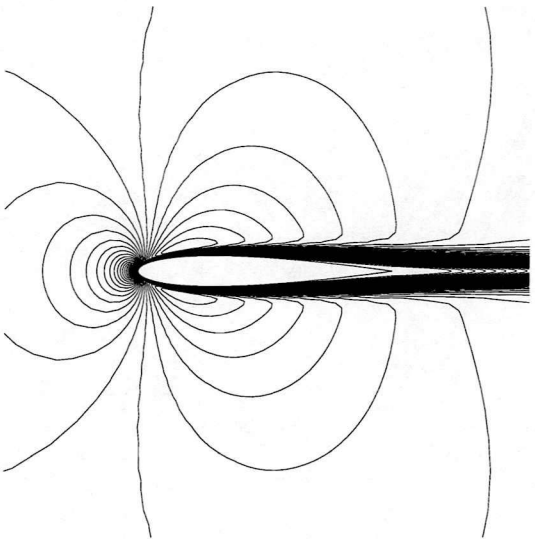


Figure 14: Mach number contours
NACA 0012 aerofoil
$M_\infty = 0.5$, $\alpha = 0^o$, Re = 5000

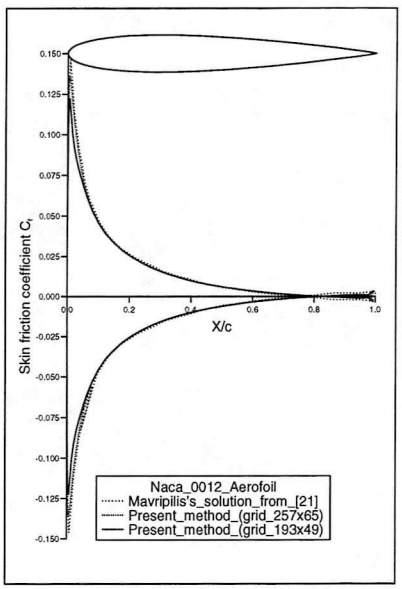


Figure 13: Skin friction distribution
NACA 0012 aerofoil
$M_\infty = 0.5$, $\alpha = 0^o$, Re = 5000

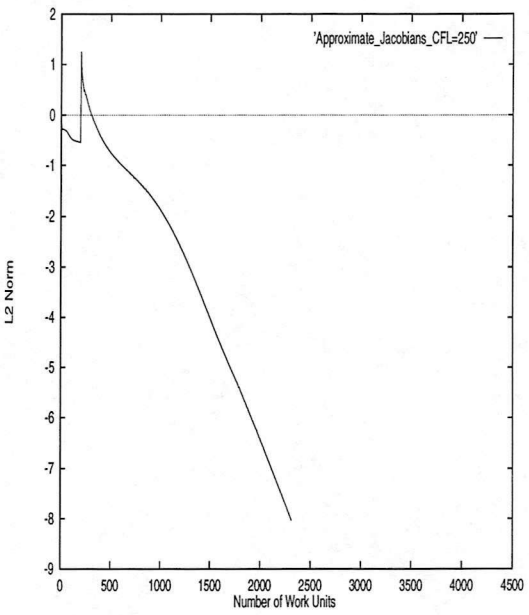


Figure 15: Convergence history
NACA 0012 aerofoil
$M_\infty = 0.5$, $\alpha = 0^o$, Re = 5000

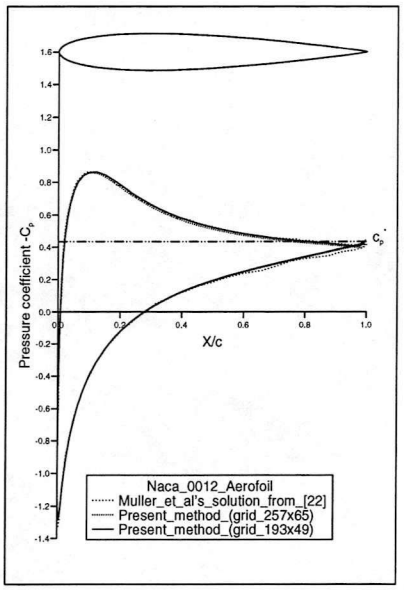


Figure 16: Pressure distribution
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 10^o$, Re = 500

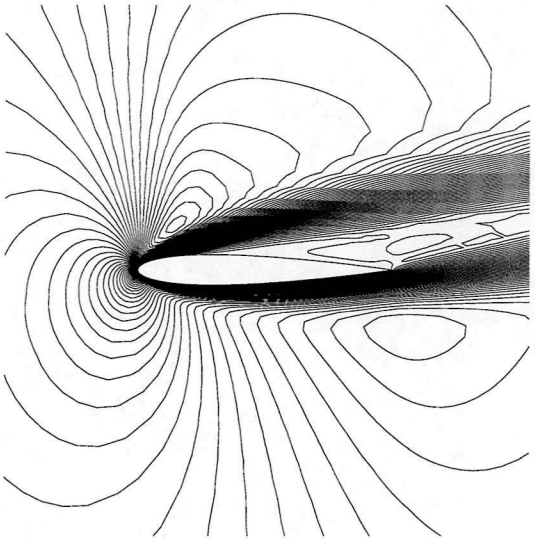


Figure 18: Mach number contours
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 10^o$, Re = 500

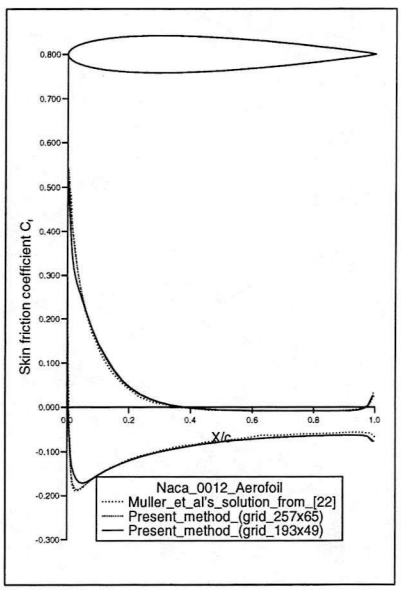


Figure 17: Skin friction distribution
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 10^o$, Re = 500

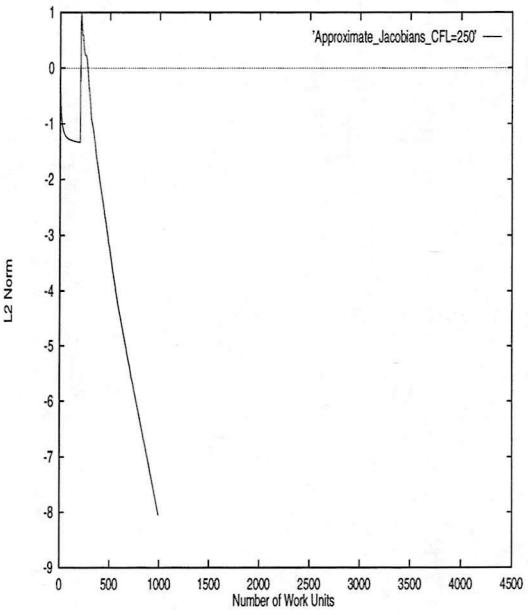


Figure 19: Convergence history
NACA 0012 aerofoil
$M_\infty = 0.8$, $\alpha = 10^o$, Re = 500

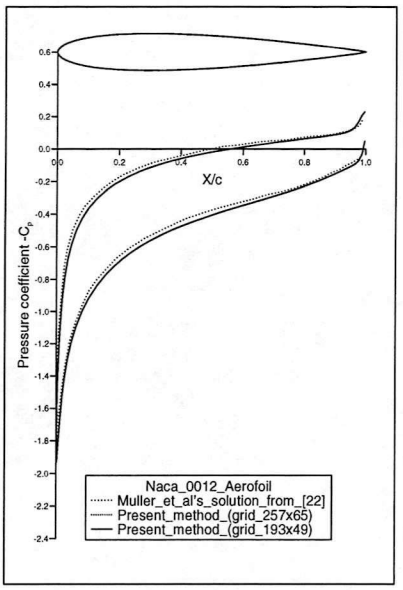


Figure 20: Pressure distribution
NACA 0012 aerofoil
$M_\infty = 2.0$, $\alpha = 10^o$, Re = 106

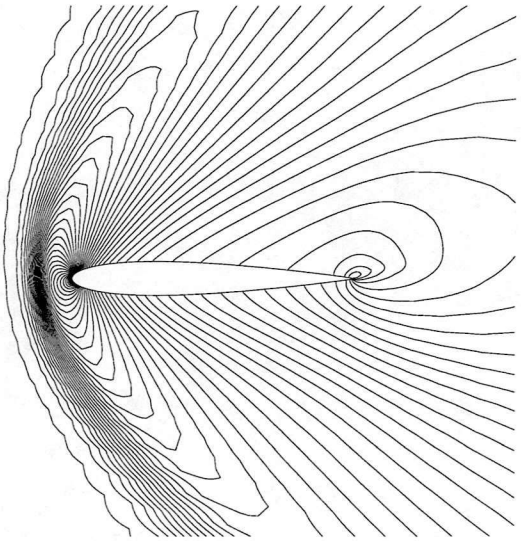


Figure 22: Density contours
NACA 0012 aerofoil
$M_\infty = 2.0$, $\alpha = 10^o$, Re = 106

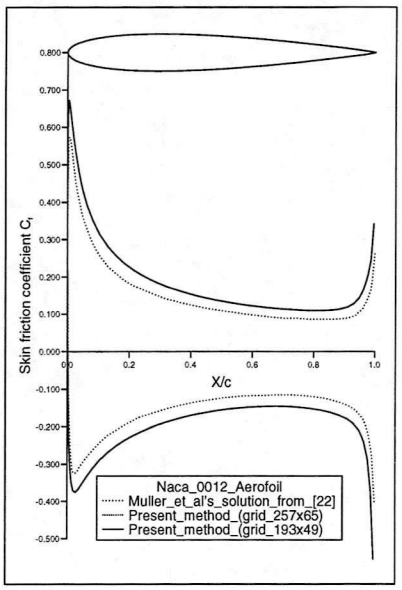


Figure 21: Skin friction distribution
NACA 0012 aerofoil
$M_\infty = 2.0$, $\alpha = 10^o$, Re = 106

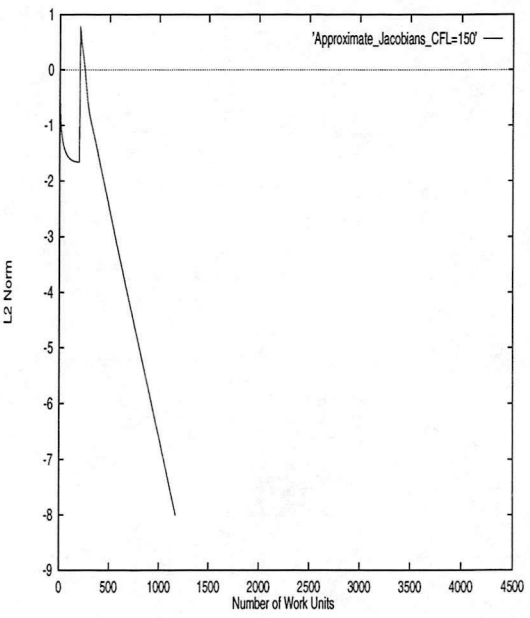


Figure 23: Convergence history
NACA 0012 aerofoil
$M_\infty = 2.0$, $\alpha = 10^o$, Re = 106