NONLINEAR ITERATION METHODS FOR HIGH SPEED LAMINAR COMPRESSIBLE NAVIER-STOKES EQUATIONS

P.A. FORSYTH* AND H. JIANG†

Abstract. Full Newton nonlinear iteration is compared to use of a defect correction approach (first order Jacobian, second order residual) for solving the steady state compressible flow equations. The Jacobian is constructed numerically, and solved using a PCG type method with block $ILU(k)$ preconditioning. Numerical tests are carried out using the NACA 0012 airfoil, at various free stream Mach numbers and Reynolds numbers. The full Newton approximation is generally more robust and takes less CPU time than the defect correction approach. No particular difficulty was observed in solving the full Newton Jacobian using an $ILU(2)$ ($\simeq 100,000$ unknowns) with CGSTAB acceleration.

Keywords: Compressible Navier-Stokes, Newton iteration, PCG methods
Running Title: Compressible Navier-Stokes
AMS Subject Classification: 65N20, 76H05
Acknowledgment: This work was supported by a grant from the National Sciences and Engineering Research Council of Canada and by the Information Technology Research Center funded by the Province of Ontario, and by Trans Computing Inc.
Date Submitted: July, 1995.
Corresponding Author: P.A. Forsyth, FAX: (519) 885-1208, e-mail: paforsyt@yoho.uwaterloo.ca

* Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

1. Introduction. Recently, there has been considerable interest in implicit methods for the solution of the Euler equations [1, 2, 3, 4, 5, 6] and for the compressible Navier Stokes equations [7, 8, 9, 10, 11, 3, 12, 13, 14]. Implicit methods allow use of large timesteps and rapid convergence to steady-state solutions.

Generally, these methods use a pseudo-timestepping approach to obtain the steady state solution. At each pseudo timestep, a backward Euler timestepping method is used, which gives rise to system of nonlinear equations. These nonlinear equations are solved by (usually) doing one iteration with an approximate Jacobian [7, 8, 9, 3, 12, 14]. The approximate Jacobian is typically based on first order upwinding, while the residual (i.e. the right hand side) is computed using a second order upwind biased scheme [15, 16]. An extensive discussion of this defect-correction approach can be found in [17].

Comparatively few authors use full Newton iteration [10, 11, 13]. Some work comparing the full Newton approach and the defect-correction technique on some model problems was reported in [11]. For problems with weak shocks, there did not seem to be much advantage to using full Newton methods. However, for tests with very strong shocks, the full Newton method was an order of magnitude faster than the quasi-Newton method. For strong shocks [11], convergence difficulties were also observed for the linear solver (a PCG method). For the Euler equations, similar conclusions were reached in [18, 19]. For problems with subsonic free stream conditions, the defect-correction approach converged, but was generally slower than the full Newton method. However, for supersonic far field conditions, convergence problems were observed with the defect-correction method [18, 19].

Preconditioned conjugate gradient (PCG) type methods are popular for solution of the linearized equations [7, 9, 5, 11, 13, 12, 18, 19]. GMRES [20] and CGS [21] acceleration methods have been used, but preconditioning has been restricted to mainly incomplete LU factorizations of level zero ($ILU(0)$) [22].

Full Newton iteration converges quadratically if the initial guess is close enough to the solution, in contrast to the linear convergence of defect-correction methods. Consequently, full Newton iteration will be an effective strategy if small residual solutions are desired.

The objective of this article is to carry out a study of full Newton methods for high speed, laminar, viscous flow. The complete laminar Navier-Stokes equations will be used. The full Jacobian matrix is constructed using numerical differentiation. Particular attention is directed towards methods for obtaining a good starting guess for the Newton iteration, and for effective PCG methods for solving the full Newton Jacobian. Comparisons will also be made with the usual defect-correction (first order Jacobian) approach. The focus throughout will be on on developing robust methods for use in a workstation environment.

Numerical tests are reported for the test cases used in the GAMM workshop [23]. These test cases are two dimensional external flows over an airfoil. Although we use a semi-structured grid, none of our algorithms make use of this fact, and hence we believe that our results should be valid for unstructured meshes as well.

2. Navier-Stokes Equations. In dimensionless form, the full Navier-Stokes equations are

$$\frac{\partial Q}{\partial t} + \frac{\partial f_c}{\partial x} + \frac{\partial g_c}{\partial y} = \frac{\partial f_v}{\partial x} + \frac{\partial g_v}{\partial y}, \tag{1}$$

where the Q is the vector of conservative variables, and f_c, g_c are the convective fluxes:

$$Q = \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{Bmatrix}, \; f_c = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(e+p) \end{Bmatrix}, \; g_c = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(e+p) \end{Bmatrix}. \tag{2}$$

The viscous fluxes are:

$$f_v = \begin{Bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xy} v + q_x \end{Bmatrix}, \; g_v = \begin{Bmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{xy} u + q_y \end{Bmatrix}, \tag{3}$$

where

$$\begin{aligned} \tau_{xx} &= \frac{2M_\infty \mu}{3Re_\infty}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \qquad \tau_{xy} = \frac{M_\infty \mu}{Re_\infty}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \\ \tau_{yy} &= \frac{2M_\infty \mu}{3Re_\infty}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right) \qquad (q_x, q_y) = \frac{M_\infty \mu}{(\gamma-1)P_r Re_\infty}\nabla T, \end{aligned} \tag{4}$$

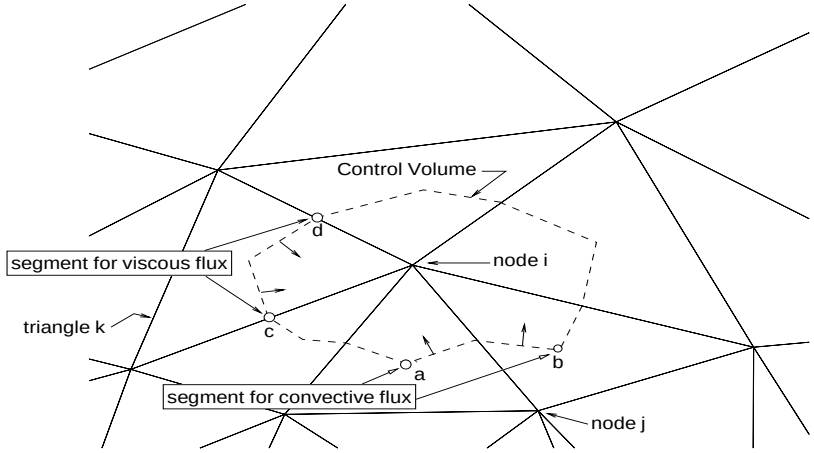
with the following variable definitions

$$\begin{aligned} \rho &= \text{density} \\ u &= \text{x-direction velocity} \\ v &= \text{y-direction velocity} \\ e &= \text{total energy per unit volume} \\ p &= \text{pressure} \\ \mu &= \text{viscosity} \\ M_\infty &= \text{free stream Mach number} \\ Re_\infty &= \text{free stream Reynolds number} \\ P_r &= \text{Prandtl number} \\ \gamma &= \text{ratio of specific heats} \\ T &= \text{temperature.} \end{aligned} \tag{5}$$

The viscosity is given by Sutherland's law [14], and the pressure and temperature are given by:

$$\begin{aligned} \mu &= \frac{(1+C^*)T^{3/2}}{T+C^*} \\ p &= (\gamma-1)\left[e - \frac{\rho(u^2+v^2)}{2}\right] \\ T &= \frac{\gamma p}{\rho} \end{aligned} \tag{6}$$

FIG. 1. *Control volume and integration line segments*



where $C^* = 198.6/460.0$, assuming the free stream temperature is $460^0 R$.

3. Discretization. In the following, we will discretize the Navier-Stokes equations using a finite volume method [4] with Van Leer flux splitting [15] of the first order convective terms.

If the computational domain is tiled with triangles, with the primary variables stored at the triangle vertices, then finite volumes can be constructed surrounding each vertex (see Figure 1).

There are various ways to construct the finite volumes surrounding each node. A common method is join the midpoints of the sides of each triangle to a point (x_m, y_m) in the interior of the triangle. There are several possibilities for selection of the point (x_m, y_m). A popular choice is to use the centroid of the triangle for (x_m, y_m). However, for the semi-structured grids used in this work, we have found that good results can be obtained with the Euler equations [18] if (x_m, y_m) is selected as the intersection of the perpendicular bisectors of each side of the triangle (assuming that the intersection point is within the triangle). If the intersection point of the perpendicular bisectors is outside the triangle, then (x_m, y_m) is the midpoint of the edge nearest the intersection point. We will use this method of selection of (x_m, y_m) in the following (both choices are options in our code).

If fully implicit timestepping is used, then the discrete finite volume Navier-Stokes equations at node i are:

$$\begin{aligned} A_i \frac{\left[Q_i^{N+1} - Q_i^N\right]}{\Delta t} &= \int_c \left(f_c \hat{i} + g_c \hat{j}\right)^{N+1} \cdot \hat{n}\ ds \\ &\quad - \int_c \left(f_v \hat{i} + g_v \hat{j}\right)^{N+1} \cdot \hat{n}\ ds \\ &\quad + S_i^{N+1}, \end{aligned} \tag{7}$$

where

$$
\begin{aligned}
\hat{n} &= \text{inward pointing unit normal} \\
c &= \text{finite volume contour} \\
S_i &= \text{source/sink term} \\
\Delta t &= \text{timestep size} \\
A_i &= \text{area of i'th finite volume} \\
N &= \text{time level.}
\end{aligned} \tag{8}
$$

The source/sink term S_i^{N+1} will be used to enforce boundary conditions.

The convective and viscous flux terms in equation (7) are handled differently. The convective flux terms are approximated by

$$
\begin{aligned}
\int_c \left(f_c \hat{i} + g_c \hat{j} \right)^{N+1} \cdot \hat{n} \; ds &\simeq \sum_{j \in \eta_i} L_{ij} F_{ij+1/2}^{N+1} \\
F &= \left\{ \begin{array}{c} \rho U \\ \rho U u + \hat{n}_x p \\ \rho U v + \hat{n}_y p \\ U(e+p) \end{array} \right\}
\end{aligned} \tag{9}
$$

where

$$
\begin{aligned}
U &= \text{ velocity component in direction}(\hat{n}_x, \hat{n}_y) \\
&= \hat{n}_x u + \hat{n}_y v \\
(\hat{n}_x, \hat{n}_y) &= \text{average inward pointing unit normal to cell face} \\
&\quad \text{between node } i \text{ and node } j \\
L_{ij} &= \text{length of face between node } i \text{ and } j \\
\eta_i &= \text{set of neighbor nodes of node } i.
\end{aligned} \tag{10}
$$

The average normal $(\hat{n}_x, \hat{n}_y)$ and interface area L_{ij} between node i and node j is given by integrating along the path a to b in Figure 1:

$$
\int_a^b \hat{n} \; ds = L_{ij} \left(\hat{n}_x + \hat{n}_y \right). \tag{11}
$$

Van Leer flux splitting [15] is used to evaluate the flux vector $F_{ij+1/2}$ at the interface between node i and node j. The flux vector is split into two contributions

$$
F_{ij+1/2} = F_{ij+1/2}^{+} + F_{ij+1/2}^{-}, \tag{12}
$$

such that the Jacobian of $F_{ij+1/2}^{+}$ has nonnegative eigenvalues and the Jacobian of $F_{ij+1/2}^{-}$ has nonpositive eigenvalues.

Pure first order flux splitting (upwind weighting) uses:

$$
\begin{aligned}
F_{ij+1/2}^{+} &= F^{+}(Q_j), \\
F_{ij+1/2}^{-} &= F^{-}(Q_i).
\end{aligned} \tag{13}
$$

Higher resolution can be obtained near shocks if the upstream value is extrapolated to the face using a MUSCL type method [15]. For example

$$F^-_{ij+1/2} = F^-(Q^-_{ij+1/2}), \tag{14}$$

where

$$\begin{aligned} Q^-_{ij+1/2} &= Q_i + \frac{s}{4}\left((1-s\kappa)\Delta_- + (1+s\kappa)\Delta_+\right), \\ \Delta_- &= (Q_i - Q_{i2up})\,\frac{\|\mathbf{x}_j - \mathbf{x}_i\|_2}{\|\mathbf{x}_i - \mathbf{x}_{i2up}\|_2}, \\ \Delta_+ &= (Q_j - Q_i), \\ s &= \frac{2\Delta_+\Delta_- + \epsilon}{(\Delta_+)^2 + (\Delta_-)^2 + \epsilon}. \end{aligned} \tag{15}$$

In equation (15), $\mathbf{x}_i$ is the location vector of node i, and ϵ is a small number which prevents zero divide.

Equation (15) effectively carries out an extrapolation to the face $ij+1/2$ using the nodes i and $i2up$. Node $i2up$ is the second upstream point to the face $ij+1/2$ in the direction j to i. There are various methods for determining node $i2up$ for unstructured grids as described in [16]. In this work, we simply choose node $i2up$ as the neighbor node of node i which is most nearly in the correct direction. The extrapolation of $Q^-_{ij+1/2}$ in equation (15) is limited so that undershoot and overshoot are avoided. The definition of s in equation (15) carries out this limiting in a smooth manner. We found that this was very helpful in promoting smooth convergence of the Newton iteration. A value of $\kappa = 1/3$ was used in equation (15), which on a regular mesh is an upwind biased third order scheme. Further details about MUSCL methods can be found in [15].

The viscous integral in equation (7) is approximated by

$$\int_c \left(f_v\hat{i} + g_v\hat{j}\right)^{N+1} \cdot \hat{n}\; ds \;\simeq\; \sum_{k\in\beta_i} \left(f_v^k \hat{a}_x^k + g_v^k \hat{a}_y^k\right)^{N+1} \tag{16}$$

where

$$\begin{aligned} (\hat{a}_x^k, \hat{a}_y^k) &= \text{inward pointing vector area in triangle } k \\ \beta_i &= \text{set of triangles having node } i \text{ as a vertex.} \end{aligned} \tag{17}$$

The inward pointing vector area $(\hat{a}_x^k, \hat{a}_y^k)$ to node i in triangle k is computed using the path from c to d in Figure 1:

$$\int_c^d \hat{n}\; ds = \hat{a}_x^k + \hat{a}_y^k. \tag{18}$$

The viscous fluxes in each triangle are estimated by

$$f_v^k = \begin{Bmatrix} 0 \\ \bar{\tau}_{xx} \\ \bar{\tau}_{xy} \\ \bar{\tau}_{xy}\bar{v} + \bar{q}_x \end{Bmatrix},\; g_v^k = \begin{Bmatrix} 0 \\ \bar{\tau}_{xy} \\ \bar{\tau}_{yy} \\ \bar{\tau}_{xy}\bar{u} + \bar{q}_y \end{Bmatrix}, \tag{19}$$

where

$$\bar{\tau}_{xx} = \frac{2M_\infty \bar{\mu}}{3Re_\infty}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \qquad \bar{\tau}_{xy} = \frac{M_\infty \bar{\mu}}{Re_\infty}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$$

$$\bar{\tau}_{yy} = \frac{2M_\infty \bar{\mu}}{3Re_\infty}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right) \qquad (\bar{q}_x, \bar{q}_y) = \frac{M_\infty \bar{\mu}}{(\gamma-1)P_r Re_\infty}\nabla T. \tag{20}$$

In equations (19 - 20), $\bar{u}, \bar{v}, \bar{\mu}$ are the average values of u, v, μ over the triangle. The derivative terms are estimated using linear interpolation with each triangle. If N_i are the usual linear Lagrange polynomial C^0 basis functions where

$$\begin{aligned} N_i &= 1 \text{ at node i} \\ &= 0 \text{ at all other nodes} \\ \sum_j N_j &= 1 \text{ everywhere in the solution domain,} \end{aligned} \tag{21}$$

then, for example, the derivatives of the temperature T are approximated by

$$\begin{aligned} \frac{\partial T}{\partial x} &= \sum_j T_j \frac{\partial N_j}{\partial x} \\ \frac{\partial T}{\partial y} &= \sum_j T_j \frac{\partial N_j}{\partial y} \end{aligned} \tag{22}$$

with similar expressions for the other derivatives.

4. Boundary Conditions. All boundary conditions are enforced by suitable definition of the source/sink terms in equation (7). Physically, we can imagine that the boundary conditions are imposed by injecting an appropriate amount of mass, energy and momentum into nodes which are on the boundary (see Figure 2). In all cases, the boundary conditions are handled fully implicitly.

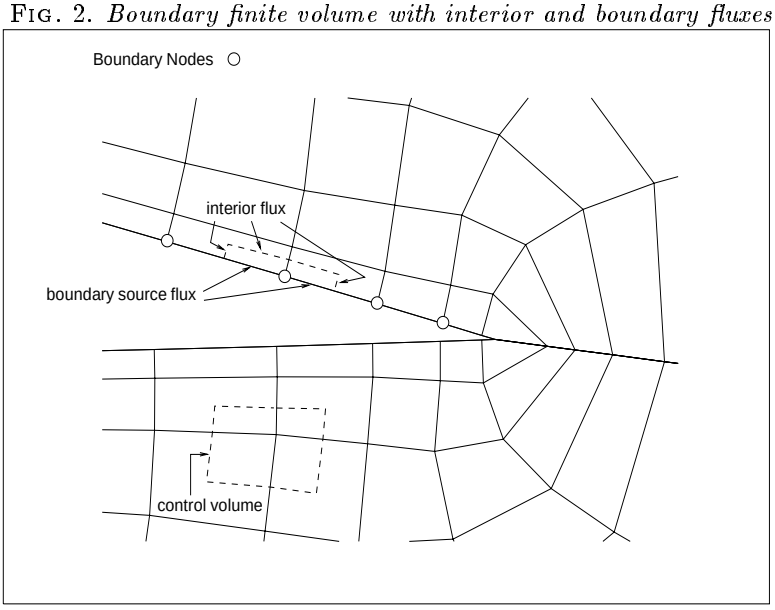
In the far field, the free stream values for the primary variables are, in dimensionless variables:

$$\begin{aligned} \rho = 1 \qquad & u = M_\infty \\ v = 0 \qquad & p = \frac{1}{\gamma} \\ e &= \frac{M_\infty^2}{2} + \frac{1}{\gamma(\gamma-1)}. \end{aligned} \tag{23}$$

The source term for a node on the far field boundary is then (assuming that the flow is inviscid far away from the airfoil)

$$\begin{aligned} S_i &= L_b F_b \\ F_b &= \begin{Bmatrix} \rho_b U_b \\ \rho_b U_b u_b + \hat{n}_x p_b \\ \rho_b U_b v_b + \hat{n}_y p_b \\ U_b(e_b + p_b) \end{Bmatrix} \end{aligned} \tag{24}$$

FIG. 2. *Boundary finite volume with interior and boundary fluxes*



where L_b is the length of the line segment common to the boundary finite volume and the boundary, and $(\hat{n}_x, \hat{n}_y)$ is the inward pointing normal to the boundary segment. $U_b, \rho_b, u_b, v_b, e_b$ are the boundary values of the density, velocity components, and energy.

The boundary values of the normal velocity U_b and other boundary variables which appear in equation (24) are determined by taking appropriate combinations of the Riemann invariants, as described in [14], which are generally a function of both free stream values (23) and interior values.

On the airfoil surface, the boundary conditions are

$$\begin{aligned} u_s &= 0 \\ v_s &= 0 \\ T_s &= 1 + \frac{(\gamma-1)M_\infty^2}{2} \end{aligned} \tag{25}$$

with no mass flow into the airfoil. The source term for nodes on the airfoil surface is then

$$S_i^{N+1} = A_i \begin{Bmatrix} 0 \\ \Omega(u_s - u_i^{N+1}) \\ \Omega(v_s - v_i^{N+1}) \\ \Omega(T_s - T_i^{N+1}) \end{Bmatrix}, \tag{26}$$

where Ω is a large number so that $u_i = v_i \simeq 0$ and $T_i \simeq T_s$. We have found that $\Omega = 10^5$ is sufficient to force $u = v = 0$ and $T = T_s$ to five figures on the boundary without causing any problems for the nonlinear iteration. If Ω is selected too large, then poor convergence of the nonlinear iteration may result [24, 18]. Note that the form of the boundary condition (26) does not require any type of extrapolation from interior points, as is commonly used [25], and is therefore easy to compute fully implicitly. Also, it is

easy to use expression (26) to compute auxiliary quantities such as the surface heat flux and skin friction.

5. Nonlinear Strategies. At each timestep, the nonlinear iteration can be written as

$$\mathbf{A}^k(x^{k+1} - x^k) = -r^k, \tag{27}$$

where x^k is the vector of unknowns, r^k is the residual vector, superscript k is the iteration number, and $\mathbf{A}$ is the linearized equation matrix. Typically, only one nonlinear iteration is carried out per timestep, so that $k = n$ in equation (27). Tests on the Euler equations in [18, 19] confirmed that solving the nonlinear equations to convergence at each timestep was unnecessary. Note that this is in contrast to incompressible flows [26], where divergence sometimes occurred if only one nonlinear iteration per timestep was used.

There are various possibilities for evaluating $\mathbf{A}$ and r^k. Initially, if we are starting from a poor guess for the solution, it is sometimes desirable to use first order upstream evaluation of cell interface values, as in equation (13). In this case, the linearized matrix $\mathbf{A}$ is the actual Jacobian of the first order discretization. The residual is also evaluated using first order upstream weighing. In our experience, the steady state first order solution can be rapidly obtained using any initial guess. The Jacobian of this system is not difficult to solve, and there are no difficulties with the Newton iteration, since the shocks are smeared due to the low order approximation. This first order solution can then be used as the initial state for more accurate MUSCL type discretizations (equation (15)). We shall refer to this method as ALLFO.

If a high order method is used for the cell interface values, then the Jacobian matrix contains more nonzeros than the first order Jacobian, and, as will be seen in the following, the high order Jacobian matrix is more difficult to solve than the first order Jacobian. For this reason, a standard technique is to evaluate the residual using a high order method, but use a low order method to evaluate the Jacobian. This method is then not a full Newton iteration. We shall refer to this method as MUSRES.

Finally, the residual and Jacobian can be evaluated using a high order method. This results in a full Newton iteration. Since we are using a MUSCL type method as our high order technique, we shall refer to this method as ALLMUS.

In all cases, we use numerical differentiation to evaluate the linearized matrices $\mathbf{A}$. The convective and viscous residuals and contributions to the linearized matrix are computed separately. The residual can be written as (see equation (7)):

$$\begin{aligned} r^k &= r_c + r_v + S \\ & (r_c)_i = \int_{c_i} \left(f_c \hat{i} + g_c \hat{j} \right)^{N+1} \cdot \hat{n} \; ds \\ & (r_v)_i = -\int_{c_i} \left(f_v \hat{i} + g_v \hat{j} \right)^{N+1} \cdot \hat{n} \; ds \\ & S = \text{ vector of source terms} \end{aligned} \tag{28}$$

where r_c is the convective residual and r_v is the viscous residual.

TABLE 1
Nonlinear strategies

Method	Jacobian Type	Residual Type
ALLFO	1st order	1st order
MUSRES	1st order	2nd order
ALLMUS	2nd order	2nd order

The first order convective contribution to the Jacobian can be evaluated very efficiently using numerical differentiation. If the computational cost is measured in terms of number of residual evaluations required to construct the Jacobian, then the first order convective Jacobian can be constructed in a cost of four evaluations of r_c, regardless of the type of grid used. This method is described in [27, 28]. If we assume that Q_{i2up} in equation (15) is evaluated using the nearest grid node, then the cost of numerically determining the high order convective contribution to the MUSCL Jacobian is 16 evaluations of r_c [29, 18, 19].

In all cases, a very straightforward approach is used to compute the viscous contributions to the Jacobian. Each triangle in the list is scanned, and the primary variables at each node are perturbed, and the result added into the Jacobian. This requires the work of 12 evaluations of r_v.

Consequently, the ALLFO and MUSRES linearized equations require 4 evaluations of r_c and 12 evaluations of r_v, while the ALLMUS Jacobian requires 16 evaluations of r_c and 12 evaluations of r_v.

To recapitulate the terminology introduced in this Section

- **ALLFO**: Jacobian and right hand side evaluated using first order upstream weighting (13).
- **MUSRES**: Jacobian evaluated using first order upstream weighting, right hand side evaluated using high order MUSCL scheme (14).
- **ALLMUS**: Jacobian and right hand side evaluated using high order MUSCL scheme.

This is also summarized in Table 1.

6. Linear Solution Methods. The linearized equations (27) are solved using a block incomplete LU factorization [30] with CGSTAB acceleration [31]. A level based incomplete factorization ($ILU(k)$) [32] with various levels k was tested.

The incomplete factorization was carried out in a block sense, with the blocks being of size 4×4. Pivoting was using when inverting the diagonal block to enhance stability. The nodes in the mesh were ordered using RCM (Reverse Cuthill McKee) ordering [33].

The convergence tolerance was based on a reduction of the initial residual. If r_L^k is the linear residual, at the k'th inner iteration, then the convergence criteria was

$$\frac{\|r_L^k\|_2}{\|r_L^0\|_2} \quad < \quad tol_{res}. \tag{29}$$

Typically, $tol_{res} = 10^{-3}$. In principal, it is necessary to solve the inner iteration to

smaller tolerances as the Newton iteration converges in order to retain quadratic convergence [34]. However, in finite precision arithmetic, it is only necessary to solve the inner iteration sufficiently accurately, so that quadratic convergence is observed until the desired precision (for the outer iteration) is obtained. Consequently, as long as residual reduction criteria is used for the inner iteration (as in equation (29)), with tol_{res} sufficiently small, then the nonlinear convergence will be rapid. In practice, tol_{res} does not have to be that small. In [18], various experiments were conducted with different values of tol_{res} for the Euler equations. For example, use of $tol_{res} = 10^{-6}$ compared to $tol_{res} = 10^{-3}$ resulted in faster convergence (in terms of number of nonlinear iterations) only at very small nonlinear residuals. However, in terms of CPU time, use of the tighter tolerance was not beneficial.

7. Timestepping. The timestep selector was based on the changes observed in the primary variables (ρ, u, v, e) over a timestep. If superscript n refers to time level, and

$$
\begin{aligned}
\Delta\rho_{max} &= \max_i |\rho_i^{n+1} - \rho_i^n|, \\
\Delta u_{max} &= \max_i |u_i^{n+1} - u_i^n|, \\
\Delta v_{max} &= \max_i |v_i^{n+1} - v_i^n|, \\
\Delta e_{max} &= \max_i |e_i^{n+1} - e_i^n|, \\
\alpha &= \min\left(\frac{\Delta\rho_{targ}}{\Delta\rho_{max}}, \frac{\Delta u_{targ}}{\Delta u_{max}}, \frac{\Delta v_{targ}}{\Delta v_{max}}, \frac{\Delta e_{targ}}{\Delta e_{max}}\right), \qquad (30)
\end{aligned}
$$

where $\Delta\rho_{targ}, \Delta u_{targ}, \Delta v_{targ}, \Delta e_{targ}$ are user specified timestep targets for the primary variables, then the new timestep Δt^{n+1} is given by

$$\Delta t^{n+1} = \min(5, \alpha\Delta t^n). \qquad (31)$$

Essentially, equations (30 - 31) will select timesteps so that the changes observed over a timestep are approximately the target values. Initially, if large changes are observed over a timestep, then the timesteps will be small. After a few timesteps, the observed changes usually decrease, and then the timestep increases very rapidly.

Timesteps were cut only if a negative density or pressure was obtained as a result of the latest nonlinear iteration, or if the inner iteration failed to converge in 100 iterations. No underrelaxation was used.

8. Computational Details.

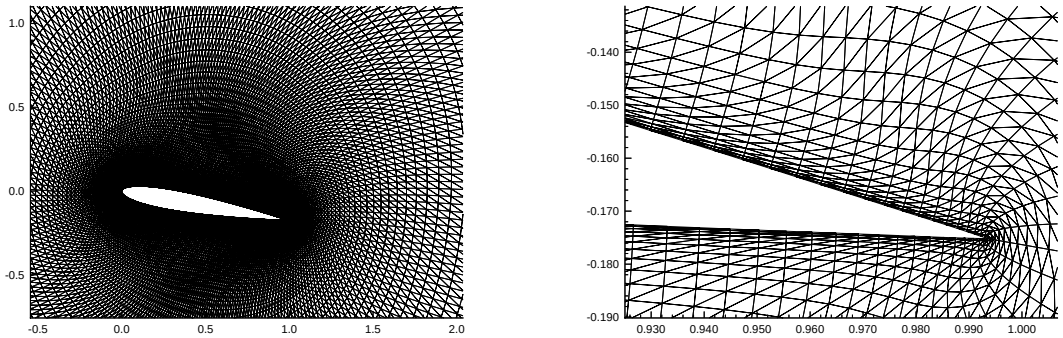
8.1. Test Problems. The test problems are the external flow over the NACA 0012 airfoil. The flow is considered to be laminar (no turbulence model), and would thus correspond to flow in a rarefied gas. Seven cases were used in the GAMM workshop [23] at various angles of attack (AOA), freestream Mach number (M_∞) and Reynolds number (Re). The viscosity and Prandtl numbers (taken to be constants) used in these

TABLE 2

Test cases, NACA 0012 airfoil, Mach Number (M_∞), angle of attack (AOA), Reynolds Number (Re).

Case	M_∞	AOA	Re
A1	0.80	10^o	73
A2	0.80	10^o	500
A3	2.00	10^o	106
A4	2.00	10^o	1000
A5	.85	0^o	500
A6	.85	0^o	2000
A7	.85	0^o	10000

FIG. 3. *Grid used in computation. Left: overview, right: closeup. of trailing edge*



tests were:

$$\begin{aligned} \mu &= 1.0 \\ P_r &= .72 \end{aligned} \tag{32}$$

The seven cases are listed in Table 2.

The grid system used in this work is a body-fitted orthogonal O-grid described in [35]. Two meshes, 192×64 and 244×108, (192 and 244 nodes on the airfoil surface respectively) have been used in computations. Each quadrilateral is divided into two triangles in the obvious manner (see Figure 3). The dense grid thus has $26,352$ nodes and $105,408$ total unknowns. The free stream boundary conditions are imposed at distance of about 25 chord lengths from the airfoil. The average grid spacing normal to the airfoil, at the airfoil surface (for the fine grid) is $< 10^{-4}$ in units of chord length.

In [23], the results are given as Mach number contour plots, and plots of pressure coefficient (C_p), skin friction coefficient (C_f), and heat flux coefficient (C_h). The pressure coefficient (at node i on the airfoil surface) is defined as:

$$C_{pi} = \frac{2(p_i - 1/\gamma)}{M_\infty^2} \tag{33}$$

The total force acting on the fluid, from the airfoil wall, at node i is given by (see equation (26)):

$$\begin{aligned} F_{xi} &= A_i\Omega(0 - u_i^{N+1}) \\ F_{yi} &= A_i\Omega(0 - v_i^{N+1}) \end{aligned} \tag{34}$$

If (n_x^t, n_y^t) is a unit vector tangent to airfoil surface, then the wall stress τ_w at node i on the surface is

$$\tau_{wi} = \frac{-\left(F_{xi}n_x^t + F_{yi}n_y^t\right)}{L_{si}} \tag{35}$$

where L_{si} is the length of the line segment common to the surface finite volume and the airfoil surface. The skin friction at node i is then defined as

$$C_{fi} = \frac{2\tau_{wi}}{M_\infty^2}. \tag{36}$$

The total heat flux from the airfoil to the fluid at a node i on the airfoil surface is

$$q_{ti} = \frac{A_i\Omega(T_s - T_i^{N+1})}{L_{si}} \tag{37}$$

The heat flux coefficient is then

$$C_{hi} = \frac{2q_{ti}}{M_\infty^3}. \tag{38}$$

The lift (C_l) and drag (C_d)coefficients are easily computed

$$C_l = \frac{\sum\limits_{i\in B} F_{yi}}{L_c} \qquad C_d = \frac{\sum\limits_{i\in B} F_{xi}}{L_c} \tag{39}$$

where L_c is the chord length, and B is the set of nodes on the airfoil surface.

8.2. Nonlinear Solution Parameters. All the nonlinear iteration strategies start with the initial guess of the free stream values for the primary variables. Next, the ALLFO method (first order Jacobian and right hand side) is used, and continues until $\|r^N\|_2 < 10^{-3}$, where r^N is defined in equation (28). This usually only takes $\simeq 10 - 15$ nonlinear iterations, and provides a good initial guess for the next stages of the nonlinear iteration. Note that in the ALLFO stage the residual is evaluated using first order upwinding. The initial timestep is set at 10^{-3}, and the timestep control parameters for this stage are (see equation (30)).

$$\Delta\rho_{targ} = \Delta u_{targ} = \Delta v_{targ} = \Delta e_{targ} = 2.0 \tag{40}$$

After many experiments, and bearing mind our experience with the Euler equations [18], we settled on the following two strategies

- ALLFO+MUSRES: After the ALLFO stage described above, the residual is recomputed using the MUSCL residual. The MUSRES iteration (first order Jacobian, MUSCL residual) then continues until convergence. The nonlinear convergence criteria is

$$\frac{\|r^N\|_2}{\|r^A\|_2} < 10^{-6} \tag{41}$$

where $\|r^A\|_2$ is the residual computed after the ALLFO stage. Typically, the initial timestep in the MUSRES stage was set at $\Delta t = 10^{-3}$, and the timestep control parameters were as in equation (40).

- ALLFO+MUSRES+ALLMUS: The first two stages of this method were the same as for ALLFO+MUSRES, except that the MUSRES stage was terminated when

$$\frac{\|r^N\|_2}{\|r^A\|_2} < 10^{-1}. \tag{42}$$

The iteration was then switched to ALLMUS (high order Jacobian and residual) until convergence condition (41) is satisfied. Timestep parameters for all stages were as in equation (40). The initial timestep for the ALLMUS step was essentially infinite ($\Delta t = 10^{+20}$).

Note that the nonlinear convergence condition (41) is relative to the starting guess from the ALLFO stage, which is quite a good initial estimate. Usually, the actual nonlinear residual at convergence $\simeq 10^{-8}$. The changes in the primary variables over the last timestep (where the maximum CFL number is often $\simeq 10^{10}$) are typically $\simeq 10^{-7}$, so that the primary variables (for the full Newton methods) are likely to be correct to $5-6$ figures.

In all cases, the inner iteration was terminated using criterion (29), with $tol_{res} = 10^{-3}$.

8.3. Linear Solution Parameters. Problems A2 and A6 (see Table 2) were used to determine the best parameters for the linear solver, for both the ALLFO+MUSRES and the ALLFO+MUSRES+ALLMUS nonlinear methods. The 192×64 grid was used for these tests. Table 3 shows the results using various levels of fill for the incomplete factorization ($ILU(k)$) [22].

On the basis of the results in Table 3, $ILU(2)$ was selected for all tests in the following. It is interesting to observe that for ALLMUS type nonlinear iterations for the Euler equations [18], at least $ILU(2)$ was required to obtain convergence of the inner iteration, while Table 3 indicates that $ILU(1)$ converges for the full Newton Jacobians in ALLMUS iterations for the Navier-Stokes equations.

9. Numerical Experiments.

9.1. Nonlinear Iteration Tests. Table 4 shows the CPU time for the test problems using both nonlinear iteration methods. The 192×64 grid was used for these tests.

TABLE 3
Comparison of ILU methods, Cases A2, A6, 192 × 64 grid, CPU hrs, SUN Sparc-10.

Nonlinear Method	$ILU(0)$	$ILU(1)$	$ILU(2)$	$ILU(3)$
	Case A2			
ALLFO + MUSRES	1.77	1.40	1.32	1.38
ALLFO + MUSRES +ALLMUS	***	.96	.74	.80
	Case A6			
ALLFO + MUSRES	2.43	2.07	2.01	2.08
ALLFO + MUSRES +ALLMUS	***	.71	.74	.83

****Run aborted due to too many inner iteration failures*

Note that in [23], none of the participants obtained a steady solution for case A7. This will be discussed further in the next Section.

Table 4 shows that the ALLFO+MUSRES nonlinear iteration strategy fails to converge for cases A3 and A4 using the timestepping and convergence tolerance parameters discussed in Section 8.2. We attempted various different timestepping strategies, and inner iteration tolerances, but we were unsuccessful in getting the MUSRES type methods to converge for cases A3 and A4. Similar problems with MUSRES type iteration methods were observed for the Euler equations with supersonic freestream boundary conditions [18]. In [11], MUSRES nonlinear iterations compared poorly to full Newton iteration when used on a wedge problem with supersonic free stream flow (Navier-Stokes solution). However, note that in references [23] and [25], problems A3 and A4 did not appear to cause great difficulty. As discussed by [25], this may be because most of the flowfield is supersonic, and hence an explicit time marching method can be efficient at moving information from upstream to downstream nodes. Note that by using $\kappa = 1/3$ in equation (15), we are following one of the recommended strategies for defect correction iteration as discussed in [17]. Because of the failure of the MUSRES step in the ALLFO+MUSRES+ALLMUS iteration, it was necessary to alter the nonlinear iteration parameters discussed in Section 8.2. During the MUSRES step, the switch to the ALLMUS step was triggered when

$$\frac{\|r^N\|_2}{\|r^A\|_2} < 2 \times 10^{-1} \tag{43}$$

with $\|r^A\|_2$ being the initial residual after the ALLFO step. However, if the ALLMUS step was started with a large timestep, the Newton iteration failed to converge. It was necessary to start the ALLMUS step (for problems A3 and A4) with $\Delta t = 10^{-3}$. The timestep control parameters for this ALLMUS stage are (see equation (30))

$$\Delta\rho_{targ} = \Delta u_{targ} = \Delta v_{targ} = \Delta e_{targ} = .05 \quad . \tag{44}$$

Consequently, choice of these parameters resulted in significantly more CPU time for problems A3 and A4 compared to the other test problems. This should be viewed as

TABLE 4
Comparison of nonlinear iteration methods, 192 × 64 grid, CPU hrs., SUN Sparc-10

	CPU hrs	
Test Case	ALLFO+MUSRES	ALLFO+MUSRES +ALLMUS
A1	.89	.64
A2	1.32	.74
A3	***	1.91†
A4	***	1.70†
A5	1.78	.74
A6	2.01	.74
A7	2.10	.89

****Failed to converge in 500 timesteps*
† *Different timestepping strategy*

a result of the fact the MUSRES step did not produce a good enough estimate for the ALLMUS step (see Table 4).

Since the use of a full Newton iteration for the final nonlinear stage is always faster and more reliable than the defect correction approach, we will use the ALLFO+MUSRES+ALLMUS strategy for the fine grid tests.

9.2. Inner Iteration Tolerance. Figure 4 compares the use of different inner iteration tolerances (tol_{res} in equation (29)) during the ALLMUS stage for Test Case A6 (192×64 grid, ALLFO+MUSRES+ALLMUS iteration used). Values of $tol_{res} = 10^{-6}$ and $tol_{res} = 10^{-3}$ were tested. The use of $tol_{res} = 10^{-6}$ does improve the rate of convergence slightly (in terms of number of iterations) as the limit of double precision accuracy is reached. However, in terms of CPU time, it is not beneficial to use the tighter inner iteration tolerance. Consequently, the the inner iteration tolerance $tol_{res} = 10^{-3}$ will be used in the following.

9.3. Fine Grid Results. Figures 5 and 6 show the results for C_p, C_f, C_h and the Mach number contours for problems A4 and A6 (244×192 grid). These results, as well as the results for the other test cases, are in good agreement with those in [25].

Table 5 shows the lift and drag coefficients for all the test cases, which are also in good agreement with the results in [23].

Note that in [23] none of the participants obtained a steady state solution to Test Case A7. However, no particular difficulty was seen in obtaining a small residual solution for this problem on the fine grid (244×108) using full Newton iteration. Figure 7 shows C_p and C_h for Case A7. Figures 8 and 9 show the skin friction and Mach number contours for Case A7 in detail. In reference [25], unsteady solutions were found at coarse grids, but on the densest grid used, the solution was steady. However, the Mach number contours (on the finest grid in [25]) do not agree with the results computed in this work (see Figure 9).

It is difficult to draw firm conclusions about the existence of a steady solution for

FIG. 4. *Comparison of inner iteration tolerance, Test A6,* 192×64 *grid.*

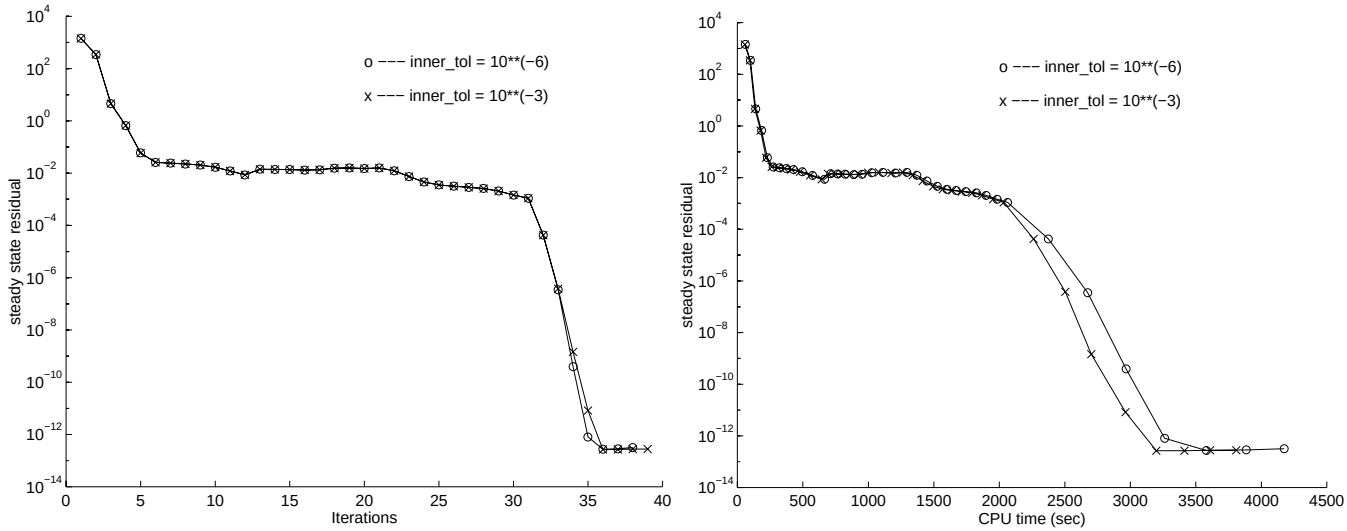


TABLE 5
Lift coefficients C_l *and drag coefficients* C_d *for the test cases (*244×192 *grid).*

Case	C_l	C_d
A1	.5580	.6506
A2	.4692	.2817
A3	.3297	.4850
A4	.3407	.2548
A5	5×10^{-5}	.2326
A6	2×10^{-4}	.1319
A7	4×10^{-4}	.0851

problem A7. Note that the grid used in this work is quite dense in the direction normal to the airfoil (normal spacing $\simeq 10^{-4}$ in units of chord length), and that we are using the full Navier-Stokes equations, not the thin layer approximation that is commonly used. On the other hand, it may be that the use of Newton iteration can produce a steady solution which is difficult to obtain using more explicit timestepping approaches.

Table 6 shows the run statistics for the fine grid computations. As for the coarse grid, it was necessary to use the modified timestepping parameters (see equation (43)) in order to get convergence for problems A3 and A4. Consequently, these two problems required about double the CPU time compared to the other test cases.

Examination of Table 6 shows that for test cases A1, A2, A5, A6, A7 (subsonic free stream conditions), the number of nonlinear iterations required to meet the convergence criterion (41) is in the range $30 - 45$. It is also interesting to observe that for these transonic problems, the average number of inner iterations per outer iteration was < 10. Since the number of unknowns here was $\simeq 100,000$, this indicates that these Jacobians

FIG. 5. *Results for problem A4: C_p, C_f, C_h and Mach number contours.*

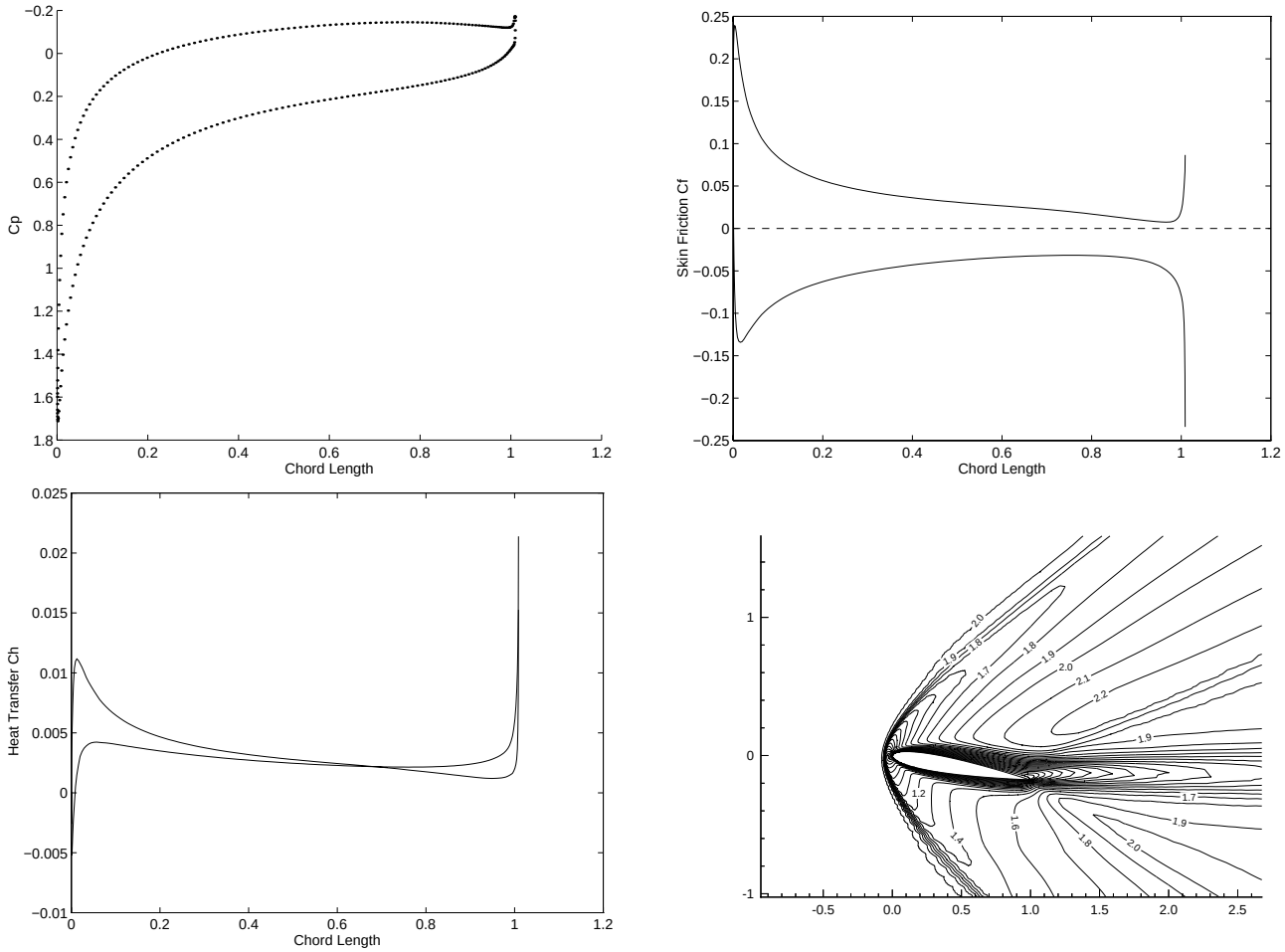


FIG. 6. *Results for problem A6: C_p, C_f, C_h and Mach number contours (244×192 grid).*

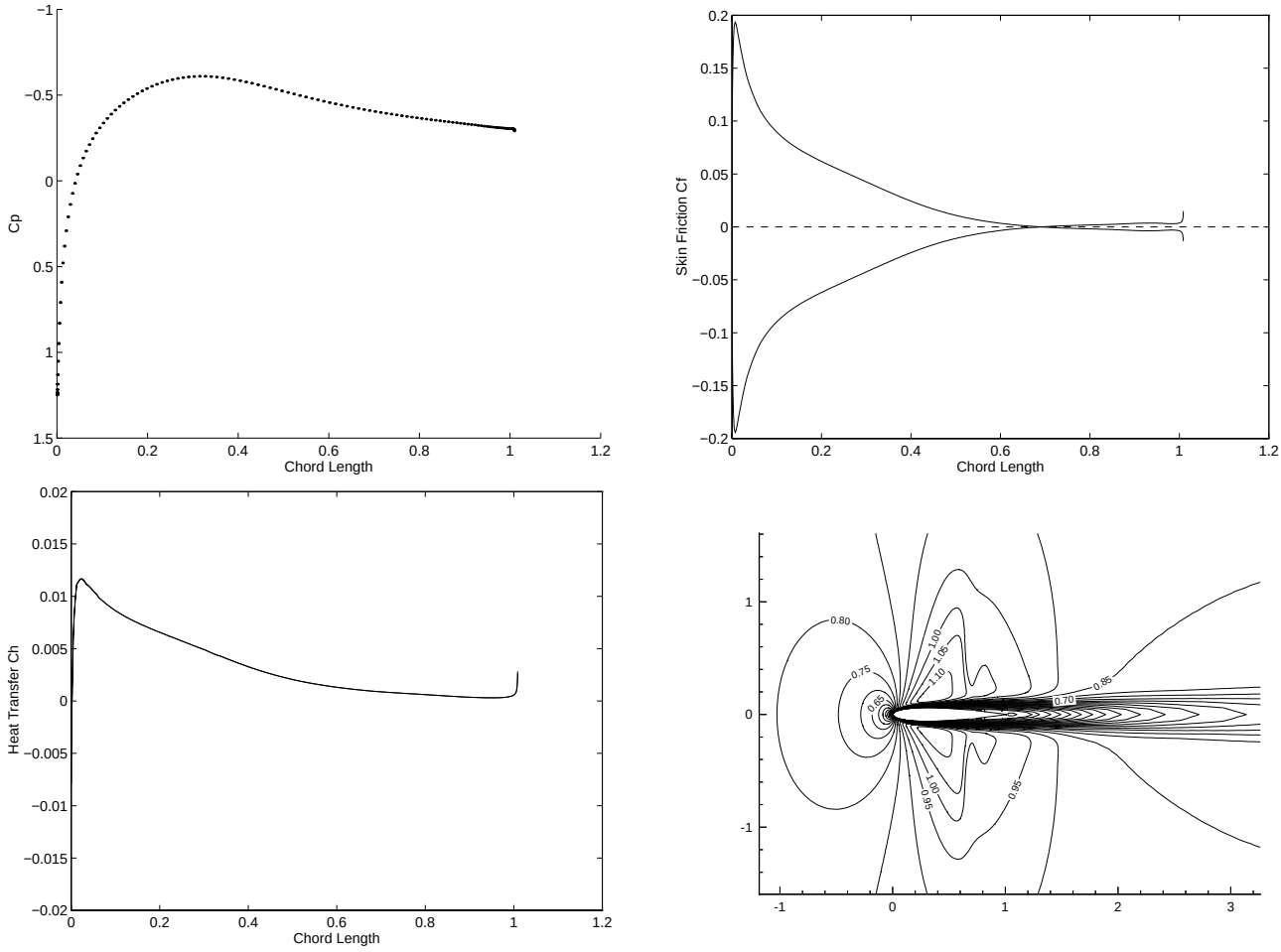


FIG. 7. *Results for problem A7: C_p and C_h (244×192 grid).*

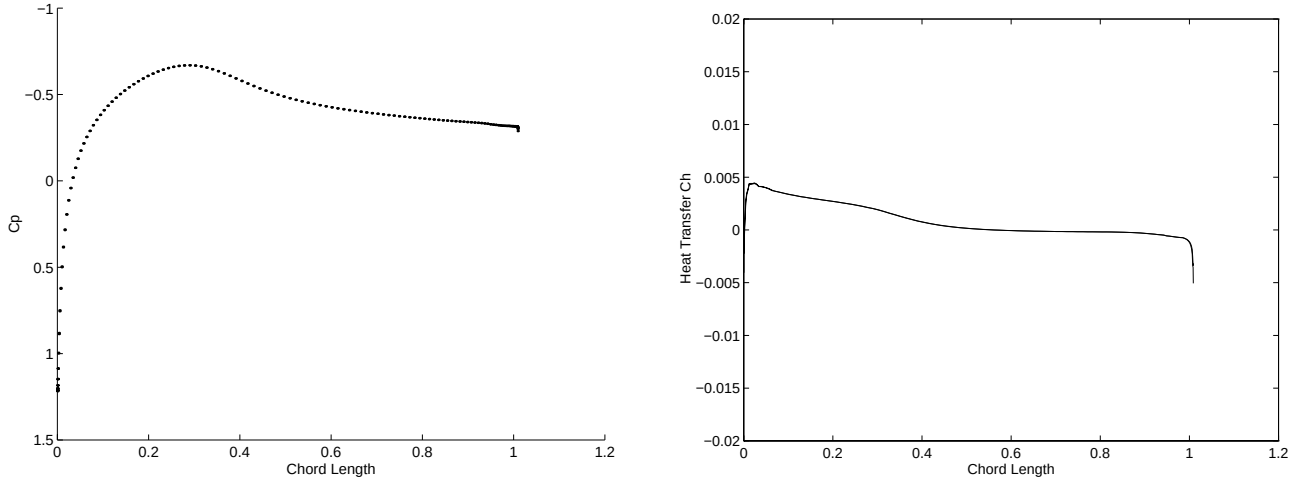


FIG. 8. *Results for problem A7: skin friction (244 × 192 grid).*

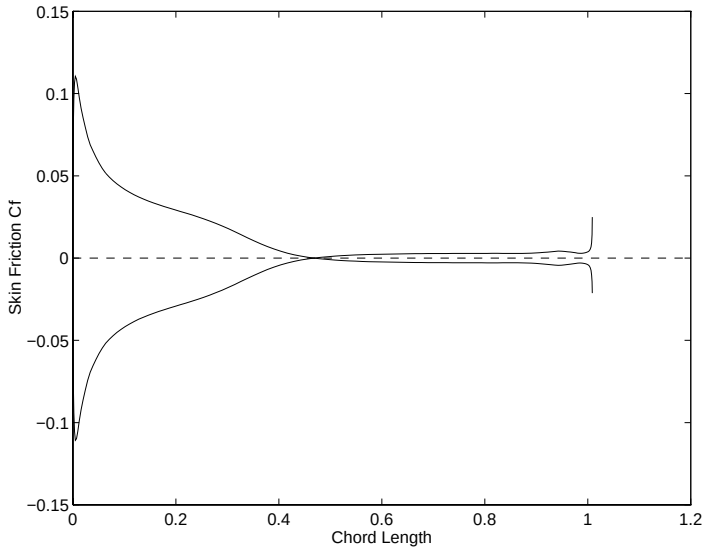


FIG. 9. *Mach number contours for A7 (244 × 192 grid).*

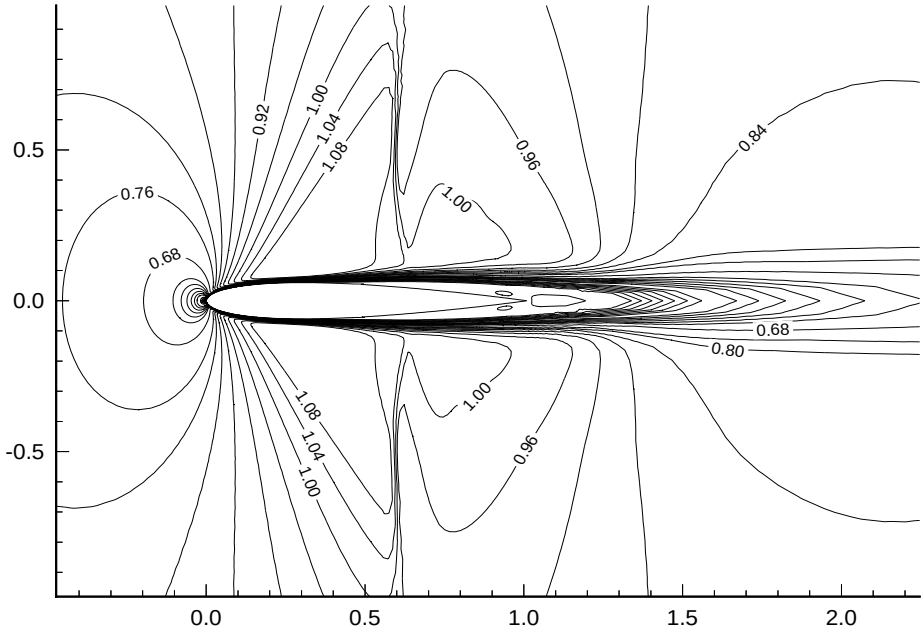


TABLE 6

Run statistics for the ALLFO+MUSRES+ALLMUS method, 244×192 *grid (105,408 unknowns), SUN Sparc-10.*

Case	CPU time (hrs)	Fraction of total time in linear solver	Total Inner iterations	Total Nonlinear iterations
A1	1.7	.61	257	31
A2	1.8	.63	295	35
A3	5.1†	.55	277	91
A4	4.4†	.53	259	105
A5	1.73	.63	275	33
A6	1.85	.64	297	36
A7	2.6	.66	403	45

† *Different timestepping strategy*

TABLE 7

Comparison of Storage (KWords in single precision) for various ILU Levels (244×108 *grid).*

	$ILU(0)$	$ILU(1)$	$ILU(2)$	$ILU(3)$
ALLFO or MUSRES	2,923	3,759	5,392	7,048
ALLMUS	5,383	10,315	16,768	23,192

are comparatively easy to solve. A more detailed perusal of the history of these runs shows that the ALLMUS type Jacobians (second order) were somewhat harder for the iterative solver compared to the MUSRES (first order) Jacobians. In general, the ALLMUS Jacobians required $\simeq 15 - 20$ inner iterations, but this is again quite good considering the number of unknowns.

For problems A3 and A4 (supersonic free stream conditions), only $3 - 4$ inner iterations were required (on average) in order to meet the inner iteration tolerance ($tol_{res} = 10^{-3}$, see equation (29)). Since the flow field is largely supersonic, then there exists an ordering of the nodes which results in the Jacobian having a *mostly* lower triangular structure. Although we made no attempt to determine such an ordering in this study (RCM ordering was used), the Jacobian is obviously very easy to solve.

Table 7 shows the storage required to store the ILU factors for the first order and second order Jacobians, for various levels of $ILU(k)$. Clearly, the robustness of using an $ILU(2)$ with the ALLMUS type iteration comes at a price: the storage required for this method is much greater than that required for an $ILU(0)$ with the MUSRES (first order Jacobian) method.

10. Conclusions. Full Newton iteration (second order Jacobian, second order residual) was always faster and more reliable than the commonly used defect correction approach (first order Jacobian, second order residual) for the test cases from the GAMM workshop [23]. The use of an ALLFO method (first order Jacobian, first order residual), followed by the defect correction approach (MUSRES) generally provided a good initial guess for the full Newton iteration stage (ALLMUS).

The test cases [23] can be divided into two categories: subsonic free stream conditions and supersonic free stream conditions. For the subsonic free stream cases, both the full Newton and defect correction methods converged, although the full Newton approach was more efficient. For the supersonic free stream cases, we were unable to get the defect correction method to converge (a similar situation was observed with the Euler equations [18]). This meant that a good initial guess was not available for the full Newton portion of the ALLFO+MUSRES+ALLMUS iteration. This problem was overcome for the full Newton approach by simply using a small timestep at the start of the ALLMUS iteration.

The first order Jacobians were easily solved using an $ILU(k)$ preconditioner with CGSTAB acceleration. For the first order Jacobians, even an $ILU(0)$ preconditioner converged, but level $ILU(2)$ was slightly more efficient than other levels tested. For the test cases with subsonic free stream conditions, the full Newton Jacobian required at least $ILU(1)$ preconditioner for convergence, but again, $ILU(2)$ was generally more efficient. Convergence of the inner iteration (for $ILU(2)$)was achieved in $10-20$ iterations for problems with $\simeq 100,000$ unknowns. For the test cases with supersonic free stream conditions, the Jacobians were very easy to solve ($3-4$ iterations), although no particular care was taken with the ordering of the unknowns,

To summarize, no particular difficulty was observed in solving all the Jacobian matrices (even the full Newton Jacobian) using an incomplete factorization of at least level (1).

The robustness of the ALLFO+MUSRES+ALLMUS nonlinear iteration method (low order methods used to obtain an initial guess, followed by full Newton iteration) comes at a price. The full Newton Jacobian and the ILU factors require considerably more storage than the low order approximations to the Jacobian.

Finally, it is interesting to note that we have obtained a steady solution to test Case A7 which was not obtained by any of the participants in the original GAMM workshop [23]. A steady solution was obtained by [25] using a dense grid, but our solution does no agree with that in [25]. Our other results for problems A1-A6 are in good agreement with those in [23, 25].

REFERENCES

[1] J. Batina. Unsteady Euler Airfoil solutions using an unstructured dynamic meshes. *AIAA J.*, 28:1381–1388, 1990.

[2] J. Batina. Implicit upwind solution algorithms for three dimensional unstructured meshes. *AIAA J.*, 31:801–805, 1993.

[3] C. B. Jenssen. Implicit multiblock Euler and Navier-Stokes calculations. *AIAA J.*, 32:1808–1814, 1994.

[4] W.K. Anderson. A grid generation and flow solution method for the Euler equations on unstructured meshes. *J. Comp. Phys.*, 110:23–38, 1994.

[5] V. Venkatakrishnan. Parallel implicit unstructured grid Euler solvers. *AIAA J.*, 32:1985–1991, 1993.

[6] K.J. Vanden and D.L. Whitfield. Direct and iterative algorithms for the three dimensional Euler equations. *AIAA J.*, 33:851–857, 1995.

[7] V. Venkatakrishnan. Preconditioned conjugate gradient methods for the compressible Navier-Stokes equations. *AIAA J.*, 29:1092–1100, 1991.

[8] F. F. Felker. Direct solution of the two dimensional Navier-Stokes equations for static aeroelasticity problems. *AIAA J.*, 31:148–153, 1993.

[9] V. Venkatakrishnan and D. Mavriplis. Implicit solvers for unstructured meshes. *J. Comp. Phys.*, 105:83–91, 1993.

[10] P.D. Orkwis and D.S. McRae. Newton's method solver for high speed viscous separated flowfields. *AIAA J.*, 30:78–85, 1992.

[11] P.D. Orkwis. Comparision of Newton's and Quasi-Newton's method solvers for the Navier-Stokes equations. *AIAA J.*, 31:832–836, 1993.

[12] K. Ajmani and W.-F. Ng. Preconditioned conjugate gradient methods for the Navier-Stokes equations. *J. Comp. Phys.*, 110:68–81, 1994.

[13] L.C. Dutto, W.G. Habashi, M.P. Robichaud, and M. Fortin. A method for finite element parallel viscous compressible flow calculations. *Int. J. Num. Meth. Fluids*, 19:275–294, 1994.

[14] W.K. Anderson and D.L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers Fluids*, 23:1–25, 1994.

[15] W.K. Anderson, J.L. Thomas, and B. Van Leer. Comparison of finite volume flux vector splittings for the Euler equations. *AIAA J.*, 24:1453–1460, 1986.

[16] C.J. Hwang and S.J. Wu. Adaptive finite volume upwind approach on mixed quadrilateral-triangular meshes. *AIAA J.*, 31:61–67, 1993.

[17] J.A. Desideri and P.W. Hemker. Convergence analysis of defect correction iteration for hyperbolic problems. *SIAM J. Sci. Comp.*, 16:88–118, 1994.

[18] H. Jiang and P.A. Forsyth. Robust linear and nonlinear strategies for solution of the transonic Euler equations. *Comp. Fluids.* accepted, 1995.

[19] P.A. Forsyth and H. Jiang. Full Newton solution of the Euler equations: An object oriented approach. CFD95, Proceedings third annual meeting of the Canadian CFD Society, Banff, 1995.

[20] Y. Saad and M.H. Schultz. GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7:856–859, 1986.

[21] P. Sonneveld. CGS: A fast lanczos-type solver for non-symmetric systems. *SIAM J. Sci. Stat. Comp.*, 10:36–52, 1989.

[22] E.F. D'Azevedo, P.A. Forsyth, and W.P. Tang. Towards a cost effective ILU preconditioner with high level fill. *BIT*, 32:442–463, 1992.

[23] M.O. Bristeau, R. Glowinski, J. Periaux, and H. Viviand (Eds.). *Numerical Simulation of compressible Navier-Stokes flows.* Vieweg & Sonh, Braunschweig/Wiesbaden, 1987.

[24] Y.S. Wu, P.A. Forsyth, and H. Jiang. A consistent approach for applying numerical boundary conditions for multiphase subsurface flow. *J. Cont. Hyd.* submitted, 1995.

[25] T. Siikonen and J. Hoffren. Solution of the thin layer Navier-Stokes equations for laminar transonic flow. Helsinki University of Technology, Laboratory of Aerodynamics, Report A-11, 1989.

[26] S.S. Clift and P.A. Forsyth. Linear and nonlinear iterative methods for the incompressible Navier-Stokes equations. *Int. J. Num. Meth. Fluids*, 18:229–256, 1994.

[27] P.A. Forsyth and R.B. Simpson. A two phase, two component model for natural convection in a porous medium. *Int. J. Num. Meth. Fluids*, 12:655–682, 1991.

[28] P.A. Forsyth, Y.S. Wu, and K. Pruess. Robust numerical methods for saturated-unsaturated flow with dry initial conditions in heterogeneous media. *Adv. Water Res.*, 18:25–38, 1995.

[29] K.J. Badcock. A partially implicit method for simulating viscous aerofoil problems. *Int. J. Num. Meth. Fluids*, 19:259–268, 1994.

[30] G.A. Behie and P.A. Forsyth. Incomplete factorization methods for fully implicit simulation of enhanced oil recovery. *SIAM J. Sci. Stat. Comp.*, 5:543–561, 1984.

[31] H.A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 13:631–645, 1992.

[32] E.F. D'Azevedo, P.A. Forsyth, and W.P. Tang. Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM J. Matrix Anal. Applic.*,

13:944–961, 1992.

[33] A. George and J.W. Liu. *Computer Solutions of large sparse positive definite systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[34] R. Dembo, S.C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[35] H. Jiang and P.A. Forsyth. Robust numerical methods for transonic flows. *Technical Report CS-94-34, Dept. Comput. Sci., Univ. of Waterloo*, 1994.