



University of Glasgow
DEPARTMENT OF

**AEROSPACE
ENGINEERING**



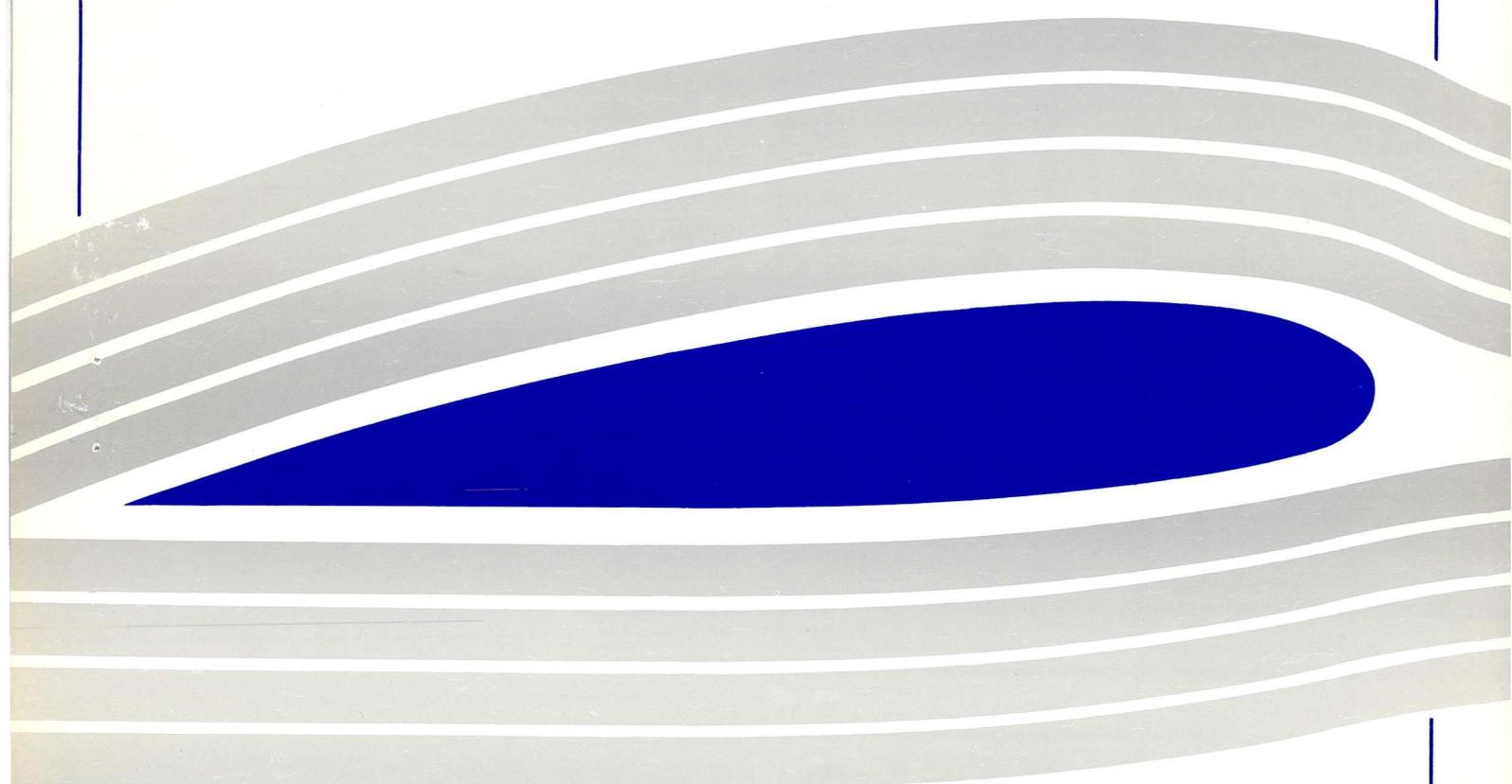
**Newton-like Methods for
Navier-Stokes Solution**

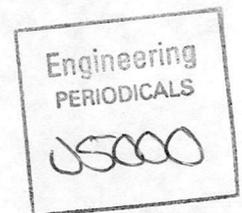
N.QIN, X.XU and Bryan E. RICHARDS

G.U. Aero Report 9229

Engineering
PERIODICALS

J5000





**Newton-like Methods for
Navier-Stokes Solution**

N.QIN, X.XU and Bryan E. RICHARDS

G.U. Aero Report 9229

Department of Aerospace Engineering
University of Glasgow
GLASGOW G12 8QQ

To be published in the Proceedings of the First European Computational Fluid
Dynamics Conference Brussels, Sept. 1992. Elsevier, Amsterdam 1992

June 1992

NEWTON-LIKE METHODS FOR NAVIER-STOKES SOLUTION

N. QIN, X. XU and B. E. RICHARDS

Department of Aerospace Engineering, University of Glasgow, Glasgow, G12 8QQ, UK.

The paper reports on Newton-like methods called SFDN- α -GMRES and SQN- α -GMRES methods that have been devised and proven as powerful schemes for large nonlinear problems typical of viscous compressible Navier-Stokes solutions. They can be applied using a partially converged solution from a conventional explicit or approximate implicit method. Developments have included the efficient parallelisation of the schemes on a distributed memory parallel computer. The methods are illustrated using a RISC workstation and a transputer parallel system respectively to solve a hypersonic vortical flow.

1. INTRODUCTION

The work to be reported contributes to the aim of the CFD research team in the Department of Aerospace Engineering at Glasgow to provide the computational tools to calculate accurately and efficiently steady and unsteady viscous compressible flows over complex aerospace configurations.

Following many studies by the team of adapting current state of the art CFD techniques to predicting hypersonic viscous flows, it is apparent to us and the customers that have taken delivery of our CFD hypersonic codes that, with increasing complexity of case, there are difficulties in convergence to sufficient accuracy using explicit and approximate implicit schemes without using gigantic amounts of time on even the most powerful of national supercomputing facilities. There is a train of thought that with the advent of more powerful computers these difficulties will be easily overcome and that the present algorithms will be sufficient. This view however ignores several general points. The past history of dramatic developments in CFD was achieved by clever new algorithms and attention to detail along with the hardware and software advances. Secondly, the predictions of equally dramatic advances in the future have assumed that opportunities offered by new architectures will be fully grasped. It is with this in mind that the team has meticulously explored, for hypersonic and transonic applications, the behaviour and characteristics of CFD techniques and algebra solvers and from this experience

either chosen those that give the best accuracy or developed new techniques, such as acceleration procedures or algebra solvers, where these are deficient. Both vector processors and parallel architectures have been used in these developments.

The existence of strong shock waves, thin shear layers and their interaction in hypersonic viscous flows requires the use of a high resolution scheme for an accurate numerical simulation. Through an extensive study (ref.1) of different flux formulae on their capabilities of capturing both shock waves and shear layers, the Osher flux difference splitting scheme has been found to be satisfactory. However, high resolution schemes usually involve more complicated formulation and thus longer computation time per iteration as compared to the simpler central differencing scheme. Therefore, the acceleration of the convergence for high resolution schemes becomes an increasingly important issue.

In this paper, we will present a new iterative approach for fast steady state solution of Navier-Stokes equations using a Newton-like approach and an efficient algebra solver. Parallelisation of the approach will also be addressed for its application on distributed memory parallel computers, such as the transputer parallel system available to us. The performance of the approach is illustrated by applying it to the prediction of the hypersonic viscous flow over a cone at high angle of attack in which the high resolution Osher scheme is used.

2. THE HIGH RESOLUTION SCHEME

2.1 The Governing Equations

The governing equations are generally the compressible Navier-Stokes equations. Corresponding to the test case presented in this paper, they are the locally conical Navier-Stokes equations, which can be derived through a general coordinate transformation to the three dimensional Navier-Stokes equations in Cartesian coordinates and then applying the locally conical approximation (ref.1). The no-slip boundary condition is applied at the wall.

2.2 The Osher Flux Difference Splitting Scheme

In the cell centred finite difference or finite volume formulation, the state variables are evaluated at cell centres and represent cell-averaged values. The fluxes are evaluated at cell interfaces. The spatial derivatives are then represented as a flux balance across a cell. The diffusive fluxes are calculated at cell interfaces using a central differencing scheme. The convective interface flux is determined from a local one-dimensional model of wave interactions normal to the cell interfaces. With the flux difference splitting (FDS) model developed by Osher and Chakravarthy (ref.2), the convective interface flux can be written as an integral in the state variable domain carried out along a path piecewise parallel to the eigenvectors.

2.3 The MUSCL Approach for Higher Order Accuracy

State-variable interpolations determine the resulting accuracy of the scheme. A third order upwind-biased scheme is chosen from the family of higher-order schemes (ref.3). Higher-order terms in the interpolation are limited in order to avoid oscillations at discontinuities such as shock waves in the solutions. The limiting is implemented by locally modifying the difference values in the interpolation to ensure monotone interpolation.

3. THE SFDN- α -GMRES AND SQN- α -GMRES METHODS

3.1 Introduction

After the above discretisation and proper treatment at the domain boundaries, a large sparse nonlinear system results, which we denote as

$$\mathbf{R}(\mathbf{Q}) = 0 \quad (3.1)$$

For steady state problems, a time dependent approach is often employed, which can be written as

$$\frac{\partial \mathbf{Q}}{\partial t} + \mathbf{R}(\mathbf{Q}) = 0 \quad (3.2)$$

Using a fully implicit method, e.g. the backward Euler implicit method,

$$\left[\frac{1}{\Delta t} \mathbf{I} + \left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right)^n \right] \Delta \mathbf{Q}^n = -\mathbf{R}(\mathbf{Q}^n) \quad (3.3)$$

unconditional stability can be achieved and as the time step approaches infinity the method approaches the Newton method

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right)^n \Delta \mathbf{Q}^n = -\mathbf{R}(\mathbf{Q}^n) \quad (3.4)$$

for the solution of the nonlinear system (3.1). In practical applications to CFD problems, however, it is very difficult (i) to get the analytical Jacobian of the nonlinear system for a high order high resolution scheme for viscous flows (it is almost impossible if turbulence or chemical reactions are involved) and (ii) to solve the resulting large sparse nonsymmetric linear system efficiently. Previous researchers in CFD have tried to avoid these two difficulties in the following ways respectively: (i) to construct simplified implicit operators, e.g. to use only first order inviscid implicit operators; (ii) to use approximate factorisation for the multidimensional implicit operator so that the resulting linear systems can be solved easily. Both of these naturally negate the advantages of the fully implicit scheme. Therefore the time step size is still limited due to the inconsistency of the implicit operator and the right hand side (the nonlinear system) and the factorisation error which increases with the time step. Simplified implicit methods will thus obviously not approach a Newton iterative method as the time step approaches infinity.

3.2 The SFDN and SQN nonlinear iterative methods

Instead of avoiding the difficulties for a fully implicit method, Qin and Richards (refs. 4, 5) tackled the problem directly in order to achieve fast convergence for the steady state solution. The discretisation of the Navier-Stokes equations results in a large sparse nonlinear system to be solved, which can be considered as a fully implicit scheme with an infinite time step. Viewing the Navier-Stokes solution as the solution of a large sparse nonlinear system, we derived a fast convergence algorithm which is general and robust.

The algorithm is based on the Newton iterative method. Due to the complexity of the nonlinear system, an analytical expression for the Jacobian matrix is usually not

obtainable. Therefore, we then took the following two approaches: (i) the sparse finite difference Newton(SFDN) method(ref. 6); and (ii) the sparse quasi-Newton(SQN) method(ref. 7).

The SFDN method calculates numerically the Jacobian of the nonlinear system. Making use of its structured sparsity, Qin and Richards (ref. 4) devised a practical way of calculating the Jacobian using finite differences. If we take the present 2-D case as an example, the above higher order spatial discretisation will result in a 13-point stencil (Fig. 1).

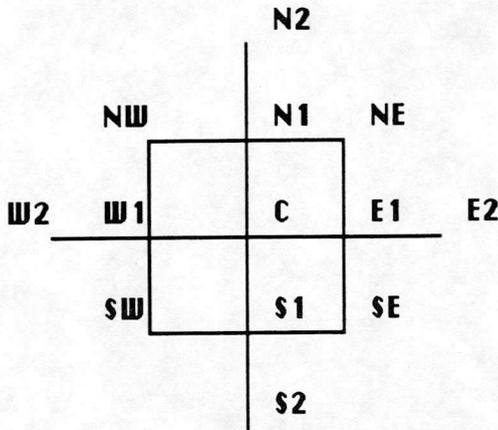


Fig. 1 Discretisation stencil using the high resolution scheme

In the calculation of the Jacobian, we can minimize the number of calculations of $R(Q)$ in the following way. Because the discretisation has a 13-point stencil, we can perturb one of the five state variables by a local increment $h_{i,j}^l$ at every 5 points in both coordinate directions in one evaluation of $R(Q)$, i.e. we calculate

$$R(Q + \sum_{\substack{i=m,I,5 \\ j=n,J,5}} h_{i,j}^l e_{i,j}^l), l=1,5; m=1,5; n=1,5 \quad (3.5)$$

where $e_{i,j}^l$ is the unit vector at point (i,j) for the lth component of the state. Therefore we can get the finite difference approximation of the Jacobian column by column through a total number of 125 evaluations of $R(Q)$. If the increments are properly chosen according to machine zero and the rounding errors in calculating $R(Q)$, the SFDN method can still give a quadratic convergence rate as has been shown by Dennis and Schnabel (ref. 8).

The SQN method updates an approximation to the Jacobian from the solution of the linear system and the value of $R(Q)$ available. It is an extension of the quasi-Newton method to nonlinear systems with sparse Jacobians. To keep the sparsity structure of the Jacobian, only those non-zero elements are updated through a matrix projection operator P_J , which maps a matrix M to a matrix

retaining only those non-zero elements according to the sparsity structure of the Jacobian. The updating procedure can be written as

$$\begin{aligned} A^n \Delta Q^n &= -R(Q^n) \\ Y^n &= R(Q^{n+1}) - R(Q^n) \\ \Delta A^n &= P_J [D^+ (Y^n - A^n \Delta Q^n) (\Delta Q^n)^T] \\ A^{n+1} &= A^n + \Delta A^n \end{aligned} \quad (3.6)$$

where D^+ is a diagonal matrix which is determined from the linear solution ΔQ^n and the sparsity structure of the Jacobian matrix. One can see that there is no extra evaluation of $R(Q)$ involved in updating the approximation. It has been proven that the SQN method has a superlinear convergence rate (ref. 8). Qin and Richards (refs. 4, 5) formulated its application to nonlinear systems with sparse block structured Jacobian matrices arising from Euler and Navier-Stokes solutions.

It is obvious that the SFDN method requires much more computing time in generating the Jacobian approximation as compared to the SQN method in which the computing time for generating the Jacobian approximation is negligible. On the other hand, the difference between quadratic convergence and superlinear convergence can be significant in practical applications because a large amount of computing time has to be spent in solving the large sparse nonsymmetric linear system at each iteration.

3.3 The α -GMRES Linear Solver

After the linearisation of the nonlinear system, a large sparse nonsymmetric linear system results, either (3.4) for the SFDN method or (3.6) for the SQN method, which we denote as

$$Ax = b \quad (3.7)$$

For a 2-D case, A is a block 13-point diagonal structured sparse matrix as shown in Fig.2.

One of the most successful methods for solving large sparse nonsymmetric linear systems is the GMRES (Generalized Minimal RESidual) method (ref. 9), which generally requires preconditioning of the matrix for practical problems. Direct use of the GMRES method to the present problem (3.7) produced nonconvergent results. A simple block diagonal preconditioning improved the results very little in convergence. Based on these observations, Xu *et al.* (ref. 10) proposed a new efficient multilevel iterative method, the α -GMRES method for the solution of the sparse nonsymmetric linear system. The matrix is first preconditioned by the inverse of its block diagonal matrix and a parameter α ($0 < \alpha < 1$) is added to the diagonal to further improve the matrix property enabling a

successful application of GMRES method. Thus a multi-level iterative solver results, which is written as

$$(\alpha I + D^{-1}A) \mathbf{x}^{k+1} = D^{-1}\mathbf{b} + \alpha \mathbf{x}^k \quad (3.8)$$

where D is the block diagonal matrix of A . We have proven the existence of a value of α ($0 < \alpha < 1$) such that the above iterative procedure will converge (ref. 10). In practical application, the parameter α is determined by a balanced convergence of the GMRES inner loop and the outer loop, which is found to be around 0.1 for the test cases. Another promising aspect of the α -GMRES is that it can easily be parallelised for distributed memory parallel computers, which is to be discussed in the following section.

Combining the α -GMRES linear solver with the nonlinear SQN and SFDN methods, we have thus devised fast convergent solvers for Navier-Stokes solutions, which we have named the SFDN- α -GMRES and SQN- α -GMRES methods respectively.

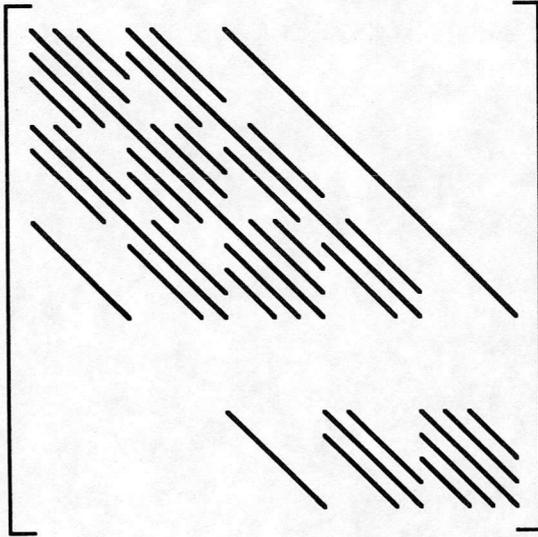


Fig. 2 Sparsity pattern of the Jacobian matrix

4. THE PARALLEL SFDN- α -GMRES METHOD

One of the main factors limiting the practical use of the above approach is their requirement for comparatively large computer memory to store the Jacobian matrix. Recent advances in parallel distributed memory multiprocessor computers offer a practical solution to this memory problem. However, how to develop efficient parallel algorithms suitable to these computer architectures is not a straightforward task. Recently Venkatakrishnan *et al.* (ref. 11) and Braaten (ref. 12) among others have carried out research in parallel N-S solutions. Radicati *et al.* (ref. 13) studied parallel linear system solutions.

The parallelisation of the SFDN- α -GMRES method is presented in the following sections through discussion of data storage and communication between the processors. Further details can be found in ref. 14.

4.1 Parallel Generation of the Sparse Jacobian Matrix and Data Storage

As presented in Sec. 3.2, the SFDN method generates the Jacobian matrix column by column. The task of generating the global Jacobian matrix is divided into balanced subtasks to be processed on each processor. The global matrix is stored according to its columns and distributed to each processor so that the communication required in the calculation is limited to minimum.

If there are P processors available, the Jacobian matrix A can be written in columns as $A = [A^1, A^2, \dots, A^P]$, where A^p are $N \times L$ submatrices stored in processor p . A vector \mathbf{v} can be written as $\mathbf{v} = (\mathbf{v}^1, \mathbf{v}^2, \dots, \mathbf{v}^P)^T$, where \mathbf{v}^p is a vector of dimension L corresponding to A^p and is stored in processor p . The distribution of the matrix data according to its columns does not increase the overall data storage compared to the sequential case.

Matching the subdivision of the global matrix and vector data to the physical domain, we obtain a corresponding domain decomposition.

4.2 The Parallel α -GMRES Method

The calculation in the GMRES method is primarily the calculation of matrix-vector products and the inner products of vectors. Parallelisation of the algorithm means the parallelisation of these basic operations. It is obvious that the inner products can be carried out in each processor using the local data and the result is assembled through scalar data collection. The global matrix-vector product can be processed using P processors as illustrated below:

$$\begin{aligned} A \mathbf{v}_i &= (A^1, A^2, \dots, A^P) \left(\begin{pmatrix} \mathbf{v}_i^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \mathbf{v}_i^2 \\ \vdots \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \mathbf{v}_i^P \end{pmatrix} \right) \\ &= A^1 \mathbf{v}_i^1 + A^2 \mathbf{v}_i^2 + \dots + A^P \mathbf{v}_i^P \\ &= \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} + \dots + \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} \Rightarrow \begin{pmatrix} \bar{\mathbf{v}}_i^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \bar{\mathbf{v}}_i^2 \\ \vdots \\ 0 \end{pmatrix} + \dots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \bar{\mathbf{v}}_i^P \end{pmatrix} \end{aligned}$$

where " \Rightarrow " indicates the communication of data among different processors to form $\bar{\mathbf{v}}_i$. In this way, the task of calculating $A \mathbf{v}_i$ is divided by calculating $A^p \mathbf{v}_i^p$ on processor p . The resulting vector $\bar{\mathbf{v}}_i$ is again distributed to the P processors. The only communication required in the calculation is in the formation of $\bar{\mathbf{v}}_i$. Due to the sparsity of

the matrix A , this communication is only of a limited nature.

The present block diagonal preconditioner can easily be parallelised, which is one of the reasons for choosing such a simple preconditioner. The block diagonal submatrices of A are inverted separately in each processor. The calculation of the matrix-matrix product $D^{-1}A$ can be performed in each processor provided that appropriate communications are arranged. Then the α is added in diagonal elements in each processor. The matrix-vector product $D^{-1}b$ can be performed in each processor without any data communication.

5. NUMERICAL EXAMPLES

The test case chosen is a hypersonic viscous flow around a sharp cone at high angle of attack (ref. 15). The flow is modelled by the Locally Conical Navier-Stokes equations, which is discretised using the Osher flux difference splitting scheme for the inviscid fluxes and a central differencing scheme for the viscous terms. The flow conditions and the cross sectional view of the temperature contours of the converged flowfield is shown in Fig. 3, in which the strong bow shock wave on the windward side and the separated shear layer on the leeward side can clearly be seen.

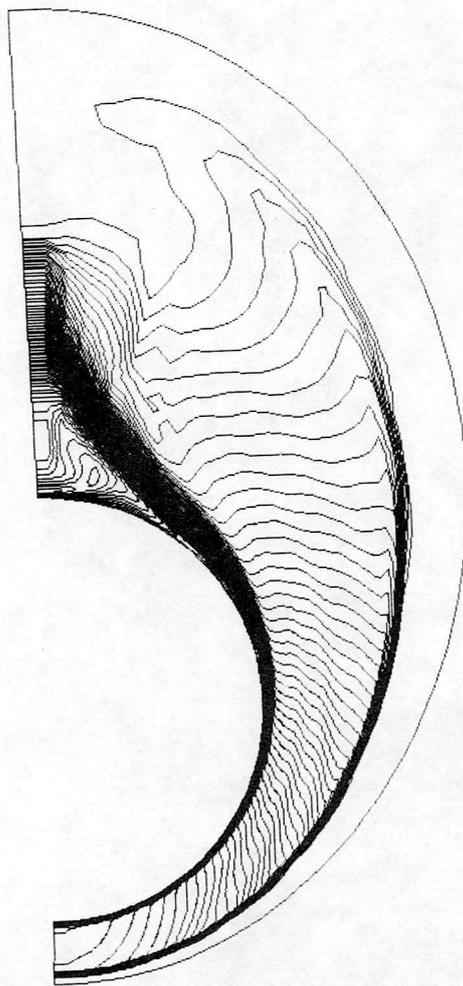
In the following two sections we present some results concerning the convergence and efficiency of the approach on a RISC workstation, the IBM RS/6000 320H, and a transputer parallel system, the Glasgow University Meiko Computing Surface respectively.

5.1 Convergence Tests using RISC Workstation

In the present tests, we choose $\alpha=0.1$ and the Krylov subspace dimension in the GMRES method as 30 and 50 for 33×33 or 66×66 grids respectively. To produce a starting solution suitable for an effective application, we use a time dependent approach for the initial phase, in which a Runge-Kutta method with local time stepping is employed.

Fig. 4 plots the convergence against computing time for calculations using the SQN- α -GMRES method or the SFDN- α -GMRES method on a 33×33 grid. As can be seen, the convergence for the explicit scheme is typically slow even though local time stepping has already been employed for efficiency. After switching to the SFDN- α -GMRES method or the SQN- α -GMRES method, the solutions converge quadratically or superlinearly respectively and the residuals reduce to machine zero in 4 or 8 iterations. For this particular case, the two methods produce similar efficiency but the SQN- α -GMRES method is expected to be more promising for problems involving more complicated physics when the expense in evaluating $R(Q)$ is much higher.

In Fig. 5, we show a test on a 66×66 grid using different convergence criteria for the iterative linear solver. We do not need to solve the linear systems (3.8) using the GMRES method or (3.7) using the α -GMRES method to a high accuracy as long as a reasonable convergence in the nonlinear iteration can be achieved. In Fig. 5, ϵ_1 and ϵ_2 represent the convergence criteria for the solution of (3.8) and (3.7) respectively. As can be seen, a larger convergence criterion can save computing time in the linear solver and it will also degrade the convergence rate of the outer nonlinear iteration. An optimum choice can be made through numerical experiments.



10° Cone
 $\bullet A_oA = 24^\circ$
 $M_\infty = 7.95$
 $T_\infty = 55.4 \text{ K}$
 $T_w = 309.8 \text{ K}$
 $Re_\infty = 4.1 \times 10^6$
 $r = 0.1 \text{ m}$

Fig. 3. Flow conditions and cross flow temperature contours

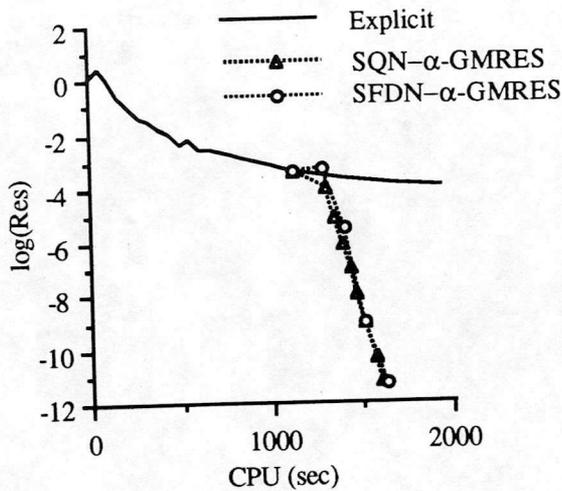


Fig.4 Convergence of the SFDN- α -GMRES and SQN- α -GMRES methods as compared with the Runge-Kutta explicit method (grid 33x33)

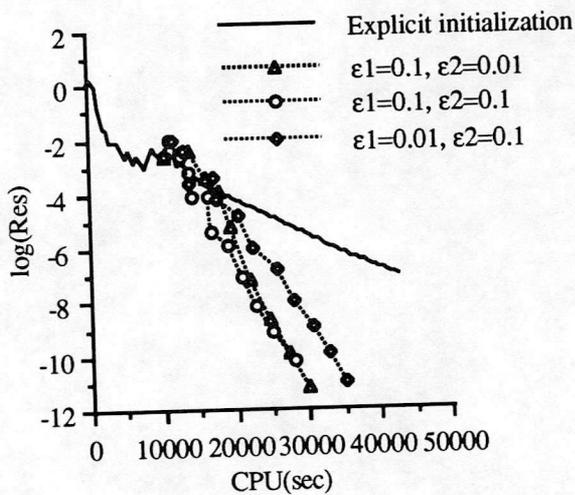


Fig.5 Parameter tests for the SFDN- α -GMRES method (grid 66x66).

5.2 Parallelisation Tests using Transputer Parallel System

The parallel algorithm has been tested on a distributed memory computer, the University of Glasgow Meiko Computing Surface composed of T800 transputers.

Fig.6 shows the speedup achieved using from 1 to 8 processors for solving the linear system using the α -GMRES algorithm.

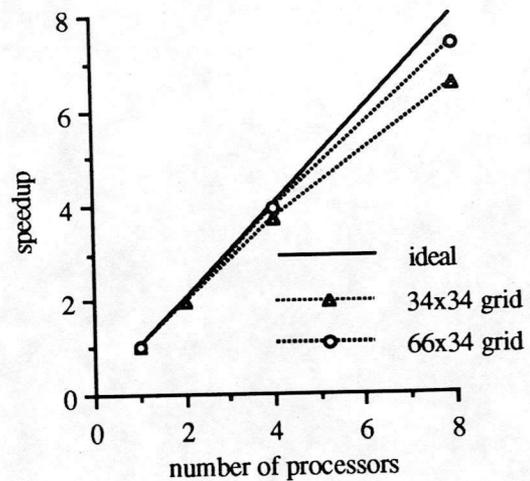


Fig. 6 Speedup with different grids for α -GMRES algorithm

Fig.7 shows the convergence histories for different numbers of processors. The convergence criterion of the inner GMRES algorithm ϵ_1 is 10^{-1} and the convergence criterion of the outer loop of the α -GMRES algorithm ϵ_2 is 10^{-10} .

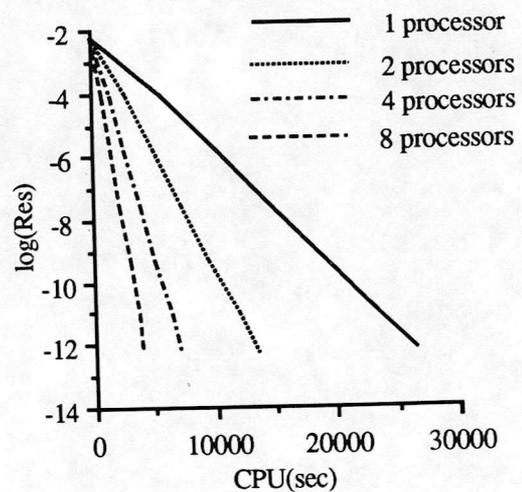


Fig. 7 Convergence of α -GMRES algorithm with different number of processors

Fig.8 shows the speedup achieved using from 1 to 8 processors for solving the Navier-Stokes equation using the Newton-like method.

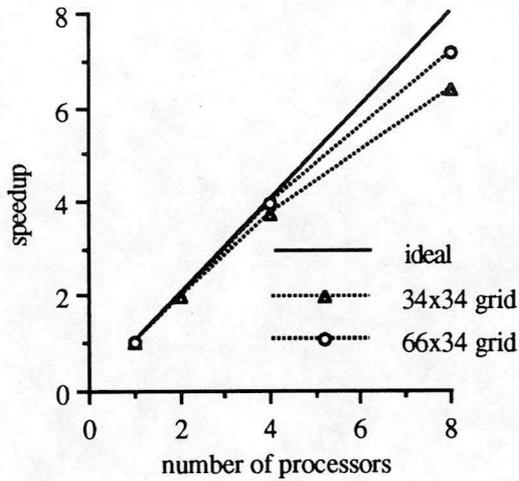


Fig. 8 Speedup for the whole N-S computation using different grids

Fig. 9 shows the convergence histories for different numbers of processors, the convergence criterion of the inner GMRES algorithm ϵ_1 is 10^{-1} , the convergence criterion of the outer loop of α -GMRES algorithm ϵ_2 is 10^{-2} , and the convergence criterion of the whole Navier-Stokes solution ϵ_3 is 10^{-10} .

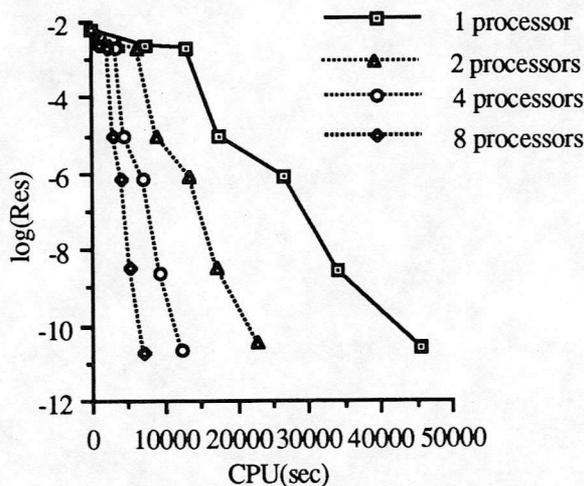


Fig. 9 Convergence of the whole N-S solution with different number of processors

Fig.10 shows the memory required on each processor for solving the Navier-Stokes equation. As can be seen, the requirement on the memory for each processor decreases as the number of the processors increases.

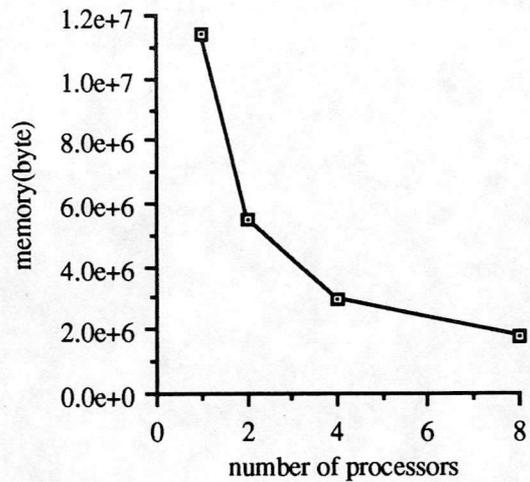


Fig. 10 Memory requirement against processor number

6. CONCLUDING REMARKS

The SFDN- α -GMRES and SQN- α -GMRES methods presented in this paper have provided a new approach for fast steady state Navier-Stokes solutions, when complexity from using high resolution schemes produces slow convergence using conventional time-dependent approach and when the analytical Jacobian is difficult to obtain. In comparison, both of the methods produce similar improvement over the corresponding explicit method in computing time for the test case. They are to be investigated further in parallel when applied to more complicated cases including turbulent modelling and/or real gas effects.

The parallelisation of the approach has been developed on a distributed memory parallel computer. The parallelisation is based on the parallel generation of the Jacobian matrix and the parallel matrix-vector products, inner products and matrix-matrix product in the α -GMRES solver. The parallel scheme maintains the convergence and the accuracy of the original sequential scheme and does not add any inner boundary conditions. In the parallel scheme the elements of Jacobian of the linear system are stored with no overlap in different processors, i.e. the sum of the storage sizes of the matrix elements in each processor is the same as the storage size of the matrix elements in the sequential scheme, which is the main storage of the Newton-like methods. This paves the way for the application of the Newton-like methods to solve the full 3-D Reynold's averaged Navier-Stokes equations.

REFERENCES

1. N. Qin, K.W. Scriba and B.E. Richards, *Aeronautical J.*, 945 (1991) pp. 152-160.
2. S. Osher and S.R. Chakravarthy, *J. Comp. Phys.*, 50 (1983) pp. 447-481.
3. B. van Leer, *Lect. Appl. Math.*, 22 (1985) pp. 327-336.
4. N. Qin and B.E. Richards, in: M. Deville (Ed.), *Notes in Numerical Fluid Mechanics* 20, Vieweg, Braunschweig, 1988, pp. 310-317.
5. N. Qin and B.E. Richards, in: P. Wesseling (Ed.), *Notes in Numerical Fluid Mechanics*, 29, Vieweg, Braunschweig, 1988, pp. 474-483.
6. A. Curtis, M.J.D. Powell and J.K. Reid, *J. Inst. Maths. Applics.*, 13 (1974) pp. 117-120.
7. L.K. Schubert, *Math. Comp.*, 24 (1970) pp. 27-30.
8. J.E. Dennis, Jr. and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, N.Y. 1983.
9. Y. Saad and M.H. Schultz, *SIAM J. Stat. Comp.*, 7 (1986) pp. 856-869.
10. X. Xu, N. Qin and B.E. Richards, *GU Aero Report* 9110 (1991) also to appear in *Int. J. Numer. Meths. Fluids*. 1992.
11. V. Venkatakrishnan, J. H. Saltz and D. J. Mavriplis, in: K.W. Morton (Ed), *Lecture Notes in Physics*, 371, Springer-Verlag, 1990, pp. 233-237.
12. M. E. Braaten, *Int. J. Numer. Methods Fluids*, 10 (1990) pp. 889-905.
13. G. Radicati and Y. Robert, *Parallel Computing*, 11 (1989) pp. 223-239.
14. X. Xu, N. Qin and B.E. Richards, *GU Aero Report* 9210 (1992) also under review in *J. Parallel and Distributed Computing*.
15. R.R. Tracy, California Institute of Technology Aeronautical Laboratory Memorandum, No.69, 1963.

