University of Glasgow

DEPARTMENT OF

AEROSPACE ENGINEERING

BILUF: a Preconditioner of Linear Solver in Newton's Method for Solving Steady State Laminar Locally Conical Navier-Stokes Equations

X. Xu

GU Aero Report 9321, August, 1993

BILUF: a Preconditioner of Linear Solver in Newton's Method for Solving Steady State Laminar Locally Conical Navier-Stokes Equations

X. Xu

GU Aero Report 9321, August, 1993

Department of Aerospace Engineering
University of Glasgow
Glasgow G12 8QQ

BILUF: a Preconditioner of Linear Solver in Newton's Method for Solving Steady State Laminar Locally Conical Navier-Stokes Equations

Xiao Xu

August, 1993

Department of Aerospace Engineering, University of Glasgow, U.K.

SUMMARY

Linearization of the non-linear system arising from Newton's method in solving steady state laminar locally conical Navier-Stokes equations results in a linear system with a large sparse non-symmetric Jacobian matrix, which will be a block 13-point diagonal stencil since high order spatial discretization scheme and structured grid are used. A new suitable arrangement of the matrix elements makes a certain BILU factorization become a very robust preconditioner in GMRES and CGS solvers. The structure of the matrix is employed in the procedure of generation of the incomplete lower and upper matrices, which greatly reduces the CPU time. These linear solvers significantly accelerate the convergence of the Newton's solver for the hypersonic viscous flows over a cone at high angle of attack, in which the Osher flux difference splitting high resolution scheme is used for capturing both shock waves and shear layers in the flowfield.

CONTENTS

1. Introduction

High order accuracy high resolution numerical schemes are most desirable when solving increasingly complex physical problems in CFD. However this involves more complicated computational formulations and increased CPU times. For solving steady state Navier-Stokes equations, the explicit method is robust in the sense that non-physical states can easily be avoided as long as the initial flow field is physically defined, but it can be extremely slow to converge due to the stability restrictions on time steps even if some acceleration techniques are employed. The Newton's method has been proved to be one of the fastest convergence algorithms for solving the Navier-Stokes equations when coupled with a robust and efficient linear solver [1-14]. However it is generally not easy to obtain the real Jacobian of the non-linear system and to solve the resulting large sparse non-symmetric linear system. As a result, much research is currently tackling this problem. In order to avoid the difficulty in generating the real Jacobian matrix, the sparse quasi-Newton method (SQN) and sparse finite difference Newton method (SFDN) were proposed by Qin and Richards [5-6] for getting the numerically approximate Jacobian matrix. A simplified procedure for generating the Jacobian matrix which reduces the CPU time was proposed by Xu [3], and is used in this paper. After the linearization of the non-linear system is achieved, a large sparse non-symmetric linear system results. In this paper, a new BILUF (Block Incomplete Lower Upper Factorization) preconditioner will be presented, which proves to be very efficient after a suitable arrangement of the matrix elements. The GMRES (Generalized Minimal Residual) [15] and CGS (Conjugate Gradient Square) [16] linear solvers are used in the Newton's method for fast steady state solution of Navier-Stokes equations.

2. High Resolution Discretization and Structured Grid

2.1 Locally conical Navier-Stokes equations and structure grid

Examination of many experimental studies [17-21] of supersonic or hypersonic laminar flows around conical shapes reveals that these flows exhibit a locally conical behaviour downstream of the nose region even though relatively large viscous regions exist. Based on this observation, McRae [22] introduced a locally conical approximation to the full Navier-Stokes equations for the solution of supersonic/hypersonic viscous flows around cones. This approximation has also been used for numerical solutions of supersonic/hypersonic viscous flows around other conical shapes [23-25]. The validity of this approximation has been well established through these experiments and

computations and the comparison between them. Therefore we have the general locally conical Navier-Stokes equations

$$\frac{\partial \mathbf{F}}{\partial \eta} + \frac{\partial \mathbf{G}}{\partial \zeta} + \mathbf{H} = 0 \qquad (2.1)$$

where **H** is the source term resulting from the locally conical approximation, η, ζ are the curvilinear coordinates. The sharp cone and structured grid in the computational section are illustrated in Fig.2.1. Because the yaw angle is 0^o just the half side of the flow is considered, where the grid overlaps by one point on the line of symmetry (Fig.2.2). We call this type of grid the primary grid, which has I+2 points in the η direction (from solid wall to inflow boundary) and J+2 points in the ζ direction (from lower symmetrical boundary to upper symmetrical boundary). A secondary grid can be obtained by determining the centres of the primary cells and connecting them across cell faces. This has I cells in the η direction and J cells in the ζ direction. We will choose the cells in the secondary grid as the control volumes in the cell centred finite volume method as illustrated in Fig.2.3. Each cell contains the state variables similar to the 3-dimensional flow, i.e. 5 conservative components ρ, ρv_1, ρv_2, ρv_3, ρE or primitive components ρ, v_1, v_2, v_3, p. The unknown variables are set in all the I×J cells of the secondary grid for η direction i from 2 to I+1 and ζ direction j from 2 to J+1, and the number of grid nodes is (I+2)×(J+2).

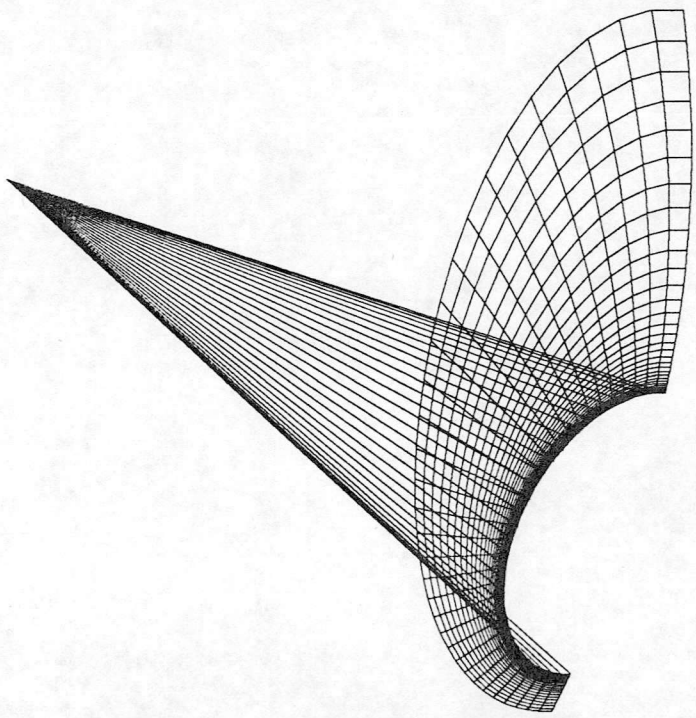


Fig.2.1 The sharp cone and location of the grid

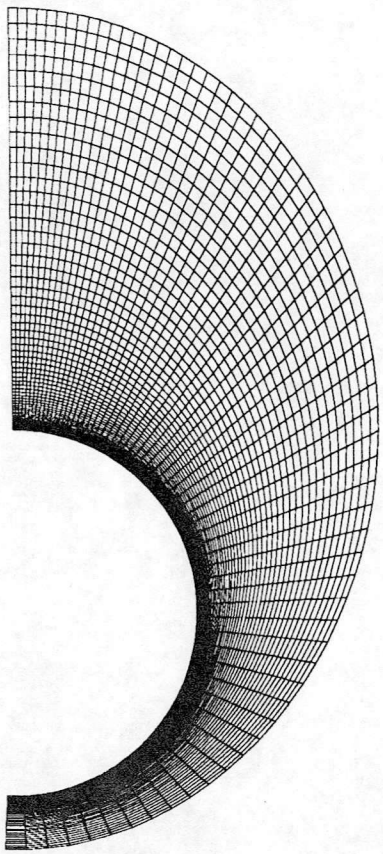
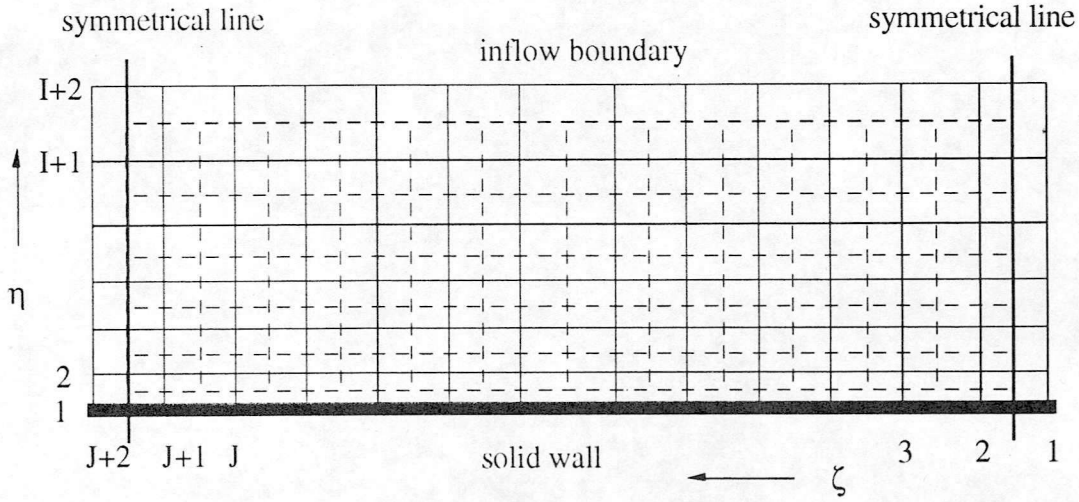


Fig.2.2 The 2-dimensional grid



The plain line is the primary grid. The dashed line is the secondary grid

Fig.2.3 The primary and secondary grids

2.2 High order finite volume discretization

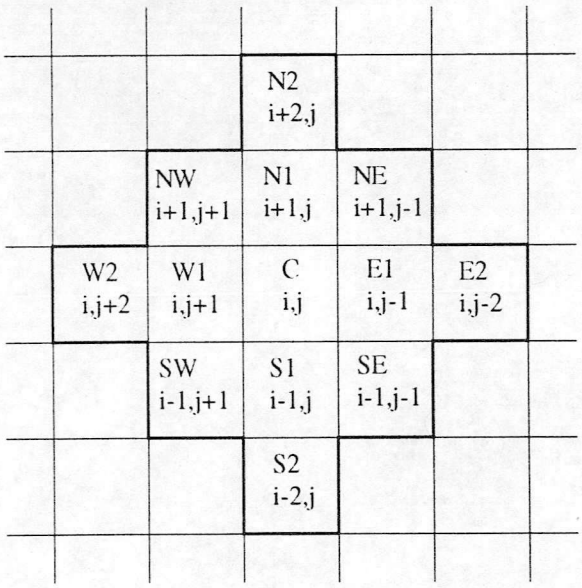
Using the cell centred finite volume method for each control volume (i,j) we have the discretised equation

$$\sum_{L} (\mathbf{F}, \mathbf{G}) \cdot \Delta\vec{l} + \mathbf{H}S_{ij} = 0 \tag{2.2}$$

where L is the bounding line of the cell area S_{ij}. Therefore we obtain:

$$\begin{aligned} R_{cell} = \frac{1}{S_{cell}} \{ & FI_{i+1/2,j} - FI_{i-1/2,j} + FI_{i,j+1/2} - FI_{i,j-1/2} \\ & + FV_{i+1/2,j} - FV_{i-1/2,j} + FV_{i,j+1/2} - FV_{i,j-1/2} \} + H_{i,j} = 0 \end{aligned} \tag{2.3}$$

where FI presents the inviscid flux and FV presents the viscous flux. Eq. (2.3) is a compact form and includes five sub-equations. When a high order Osher flux difference scheme used, the calculation of R_{cell} employs the physical variables in the 13 cells around cell (i,j) as in Fig.2.4.



The residuals calculated for the cell (i,j) and
the discretised *physical* state variables used lie in the 13 cells within the bold line.

Fig.2.4 13 cell stencils

Consolidating all the discretised Navier-Stokes equations in every cell we have an N-dimensional non-linear *algebraic* system as follows:

$$R(V) = 0 \tag{2.4}$$

where $N = I \times J \times 5$ is the total number of unknown variables, $R^T = (R_1, R_2, ..., R_N)$ is the residual vector and $V^T = (v_1, v_2, ..., v_N)$ is the discretised physical state variables vector. We note that $\{v_n, v_{n+1}, v_{n+2}, v_{n+3}, v_{n+4}\} \in V$ are the discretised physical state variables in the cell (i,j), and $\{R_n, R_{n+1}, R_{n+2}, R_{n+3}, R_{n+4}\} \in R$ are the residuals in the same cell, where we can choose

$$n = 5 \times (J \times (i-2) + j - 2) + 1 \tag{2.5}$$

or

$$n = 5 \times (I \times (j-2) + i - 2) + 1 \tag{2.6}$$

respectively corresponding to the orders of grid cells (((i,j), j = 2, 3 ,..., J+1), i = 2, 3, ..., I+1), i.e. cells are given the order from one symmetrical line to another symmetrical line first and then from solid wall to inflow boundary, and (((i,j), i = 2, 3, ..., I+1), j = 2, 3, ..., J+1), i.e. cells are given the order from solid wall to inflow boundary first and then from one symmetrical line to another symmetrical line.

3. The Newton's Method

3.1 The Newton's method

For the non-linear *algebraic* system (2.4) we have the Newton's method

$$\left(\frac{\partial R}{\partial V}\right)^k \Delta V^k = - R (V^k)$$
$$\Delta V^k = V^{k+1} - V^k \tag{3.1}$$

where V is chosen as the primitive variables. For each cell (i,j) Eq. (3.1) is

$$\left(\partial R_{cell} / \partial V\right)^k \Delta V^k = - R_{cell} (V^k), \qquad cell = (2,2), ..., (I+1, J+1) \tag{3.2}$$

3.2 Two different orders of the matrix elements

For any v_{cell}, $\partial R_{cell} / \partial v_{cell}$ is a 5×5 matrix, where R_{cell} is in a cell (i,j) and v_{cell} is in a cell (l,m), where i, l = 2, ..., I+1, and j, m = 2, ..., J+1. From Fig.2.4 we know that $\partial R_{cell} / \partial v_{cell}$ is not equal to zero only for the cell (l,m) within the thirteen cells around cell (i,j). Thus corresponding to each cell (i,j) the elements of the Jacobian matrix, in

five rows, have the form of thirteen 5×5 sub-matrices

$$\begin{aligned}
&\partial R_{cell(i,j)}/\partial v_{cell(i-2,j)},\ \partial R_{cell(i,j)}/\partial v_{cell(i-1,j-1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i-1,j)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i-1,j+1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i,j-2)},\ \partial R_{cell(i,j)}/\partial v_{cell(i,j-1)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i,j)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i,j+1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i,j+2)},\ \partial R_{cell(i,j)}/\partial v_{cell(i+1,j-1)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i+1,j)},\ \partial R_{cell(i,j)}/\partial v_{cell(i+1,j+1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i+2,j)},
\end{aligned} \tag{3.3}$$

or

$$\begin{aligned}
&\partial R_{cell(i,j)}/\partial v_{cell(i,j-2)},\ \partial R_{cell(i,j)}/\partial v_{cell(i-1,j-1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i,j-1)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i+1,j-1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i-2,j)},\ \partial R_{cell(i,j)}/\partial v_{cell(i-1,j)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i,j)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i+1,j)},\ \partial R_{cell(i,j)}/\partial v_{cell(i+2,j)},\ \partial R_{cell(i,j)}/\partial v_{cell(i-1,j+1)},\\
&\partial R_{cell(i,j)}/\partial v_{cell(i,j+1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i+1,j+1)},\ \partial R_{cell(i,j)}/\partial v_{cell(i,j+2)},
\end{aligned} \tag{3.4}$$

which correspond to the grid order (2.5) and the grid order (2.6) respectively.

3.3 The structure of the Jacobian matrix

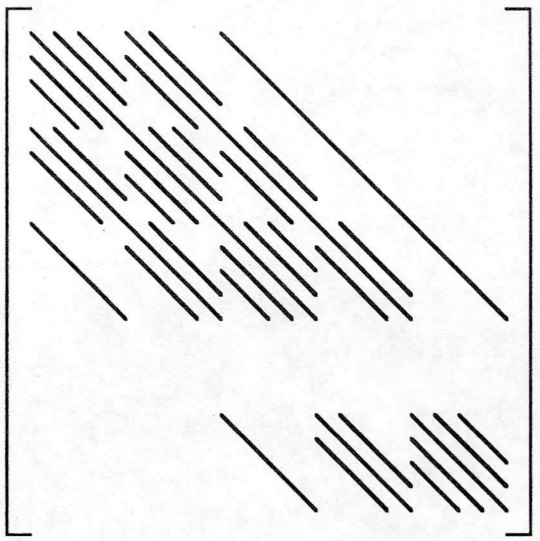
From Fig.2.4 we can illustrate the block elements in a row by using the name of the cell, that is for (3.3) we have

$$S2 \ldots SE\ S1\ SW \ldots E2\ E1\ C\ W1\ W2 \ldots NE\ N1\ NW \ldots N2 \tag{3.5}$$

but for (3.4) we have

$$E2 \ldots SE\ E1\ NE \ldots S2\ S1\ C\ N1\ N2 \ldots SW\ W1\ NW \ldots W2 \tag{3.6}$$

Therefore we obtain the Jacobian matrix of Eq. (3.1) which is an order N, block 13-point diagonal matrix and each block is a 5×5 submatrix, which can be denoted:



We denote the linear system in the Newton's formulation by

$$\mathcal{A}x=b \tag{3.7}$$

Different arrangements of the Jacobian matrix elements (3.4) and (3.5) give significant differences in the characteristic of the Jacobian matrix of the linear system to be solved, which will be seen in Section 6.

4. The Block Incomplete LU Factorization

For the block 13-point diagonal matrix, in which the block row elements are arranged as in (3.6), we can respectively construct lower and upper matrixes $\mathcal{L}_I$, $\mathcal{U}_I$ as follows:

$$\text{E2L ... SEL E1L NEL ... S2L S1L CL} \tag{4.1}$$

$$\text{CU N1U N2U ... SWU W1U NWU ... W2U} \tag{4.2}$$

where the lower and upper matrices retain the sparsity of the Jacobian matrix. Therefore we obtain a block incomplete LU factorization (BILUF), where each of the block elements will be generated using the following formulations:

```
C(i,j)  =  E2L(i,j)*W2U(i,j-2)   +  SEL(i,j)*NWU(i-1,j-1)
        +  E1L(i,j)*W1U(i,j-1)   +  NEL(i,j)*SWU(i+1,j-1)
        +  S2L(i,j)*N2U(i-2,j)   +  S1L(i,j)*N1U(i-1,j)
        +  CL(i,j)*CU(i,j)
N1(i,j) =  E1L(i,j)*NWU(i,j-1)   +  NEL(i,j)*W1U(i+1,j-1)
```

	+	S1L(i,j)*N2U(i-1,j)	+	CL(i,j)*N1U(i,j)
N2(i,j)	=	NEL(i,j)*NWU(i+1,j-1)	+	CL(i,j)*N2U(i,j)
SW(i,j)	=	SEL(i,j)*W2U(i-1,j-1)	+	S2L(i,j)*NWU(i-2,j)
	+	S1L(i,j)*W1U(i-1,j)	+	CL(i,j)*SWU(i,j)
W1(i,j)	=	E1L(i,j)*W2U(i,j-1)	+	S1L(i,j)*NWU(i-1,j)
	+	CL(i,j)*W1U(i,j)		
NW(i,j)	=	NEL(i,j)*W2U(i+1,j-1)	+	CL(i,j)*NWU(i,j)
W2(i,j)	=	CL(i,j)*W2U(i,j)		
S1(i+1,j)	=	SEL(i+1,j)*W1U(i,j-1)	+	E1L(i+1,j)*SWU(i+1,j-1)
	+	S2L(i+1,j)*N1U(i-1,j)	+	S1L(i+1,j)*CU(i,j)
S2(i+2,j)	=	SEL(i+2,j)*SWU(i+1,j-1)	+	S2L(i+2,j)*CU(i,j)
NE(i-1,j+1)	=	E2L(i-1,j+1)*NWU(i-1,j-1)	+	SEL(i-1,j+1)*N2U(i-2,j)
	+	E1L(i-1,j+1)*N1U(i-1,j)	+	NEL(i-1,j+1)*CU(i,j)
E1(i,j+1)	=	E2L(i,j+1)*W1U(i,j-1)	+	SEL(i,j+1)*N1U(i-1,j)
	+	E1L(i,j+1)*CU(i,j)		
SE(i+1,j+1)	=	E2L(i+1,j+1)*SWU(i+1,j-1)	+	SEL(i+1,j+1)*CU(i,j)
E2(i,j+2)	=	E2L(i,j+2)*CU(i,j)		

First we calculate CL(i,j) and CU(i,j) using a rank 5 matrix LU factorization, then calculate N1U(i,j), N2U(i,j), SWU(i,j), W1U(i,j), NWU(i,j), W2U(i,j) successively by using CL(i,j) to generate the block row elements of the upper matrix, finally calculating S1L(i+1,j), S2L(i+2,j), NEL(i-1,j+1), E1L(i,j+1), SEL(i+1,j+1), E2L(i,j+2) successively by using CU(i,j) to generate the block column elements of the lower matrix.

For (3.5) we can process similar BILUF. On the other hand, we can implement a BILUF for the 5-point diagonal matrix of the Jacobian matrix, and present this incomplete LU factorization of the Jacobian matrix as 5BILUF.

5. Preconditioned GMRES and CGS Linear Solvers

Since the block incomplete lower matrix $\mathcal{L}_I$ and upper matrix $\mathcal{U}_I$ have been generated we can change Eq. (3.7) as follows:

$$\begin{aligned} \mathcal{L}_I^{-1}\mathcal{A}\mathcal{U}_I^{-1}y &= \mathcal{L}_I^{-1}b \\ \mathcal{U}_I x &= y \end{aligned} \qquad (5.1)$$

Therefore the procedure for solving Eq. (3.7) is divided into three steps.

Step 1: Set new right hand side vector

$$B=\mathcal{L}_I^{-1}b, \quad \text{or} \quad \mathcal{L}_I B=b$$

Step 2: Solve for y

Step 3: Obtain solution of Eq. (3.7)

$$\mathcal{U}_I x=y$$

Step 1 and 3 correspond to solving the lower and upper linear equations respectively. For solving y in step 2 we use the GMRES and CGS linear solvers in which the new matrix, $C = \mathcal{L}_I^{-1}\mathcal{A}\mathcal{U}_I^{-1}$, vector manipulation can be implemented by solving an upper linear equation, matrix-vector manipulation, and then by solving a lower linear equation.

6. Numerical Tests and Discussion

The foregoing numerical tests have been carried out to solve the locally conical Navier-Stokes equations for compressible flow. The spatial discretization scheme used is the Osher flux difference splitting scheme, and the high order scheme is achieved by using MUSCL method for variable extrapolation. The formal accuracy is third order for the convective fluxes in which the MUSCL method for variable extrapolation is used, and second order for the diffusive fluxes. The case is a laminar Mach 7.95 flow around a sharp cone at an angle of attack of 24° and with a cold wall. This case produces a flow which has a large separated flow region with embedded shock wave in the leeward side of the cone and strong gradient in the thin boundary layer on the windward side. Accurate validation with experiment was achieved in flow field and heat transfer distribution. In the numerical tests for any real data, double precision is used. The computer used is an IBM RISC System/6000 320H workstation in the Department of Aerospace Engineering, University of Glasgow.

Two computational grids in the cross section are 34×34 and 66×66 respectively for the numerical test. Thus the resulting matrixes to be solved are the block 13-point structured matrixes of order 32×32×5 and 64×64×5. For solving the linear system the diagonal preconditioner is also used to enhance the characteristic of the matrix in Eq.(3.7) even though it is not necessary, i.e.

$$\mathcal{D}^{-1}\mathcal{A}x=\mathcal{D}^{-1}b$$

where $\mathcal{D}$ is the block diagonal matrix composed of the block diagonal elements of $\mathcal{A}$.

Fig.6.1 shows the convergence histories for the 34×34 grid, which includes the result of 1000 four-step Runge-Kutta explicit iterations as the initial guess and Newton's method with GMRES linear solver, in which the Jacobian matrix elements are arranged as in (3.6) and BILUF preconditioning is used. Fig.6.2 shows the convergence histories for the 66×66 grid, which includes the result of 2000 four-step Runge-Kutta explicit iterations as the initial guess and Newton's method with the GMRES linear solver, in which the Jacobian matrix elements are arranged as in (3.6) with BILUF and 5BILUF preconditioning. From this figure we can see that the 5BILUF preconditioner, although efficient, is much slower than the BILUF preconditioner.

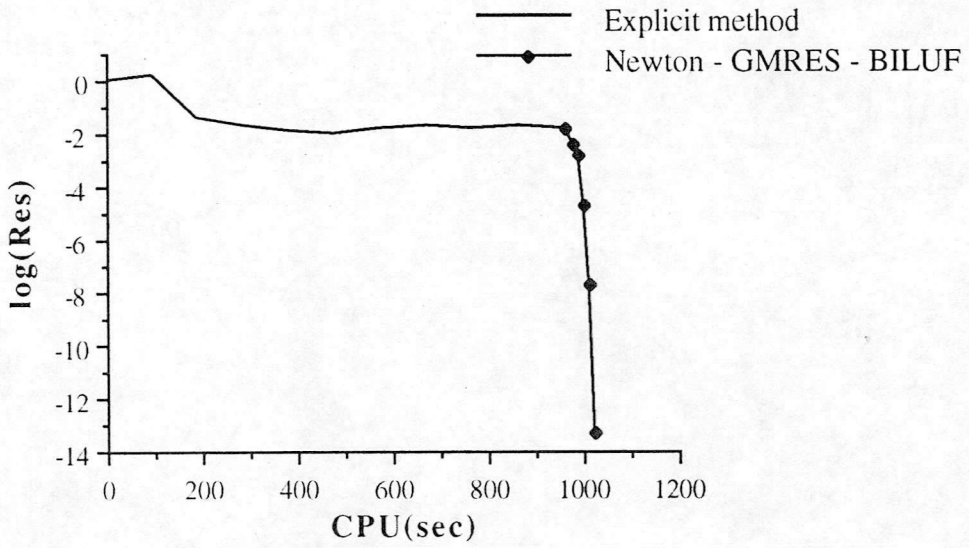


Fig.6.1 Convergence history for N-S solution for 34×34 grid

Table 1 shows the CPU time, in seconds, for different computation procedures, in which the Jacobian matrix elements are arranged as in (3.6) in the Newton's method. From this table we can see that the fastest algorithms have been obtained for (1) the Jacobian matrix generation, (2) the BILUF generation (for 66×66 grid the 5BILUF takes only 0.96 seconds), and (3) for solving the lower and upper linear systems involved in GMRES or CGS solvers. Tables 2 and 3 show the convergence details of GMRES and CGS solvers for the two differing grids above for the Jacobian matrix elements arranged as in (3.6), in which Res is the relative residual. Memory requirements for different schemes and different grids are shown in Table 4. The flow is a demanding case including massive flow separation, bow and flow embedded shocks and very high temperature gradient in the windward boundary layer as illustrated in Fig.6.3.

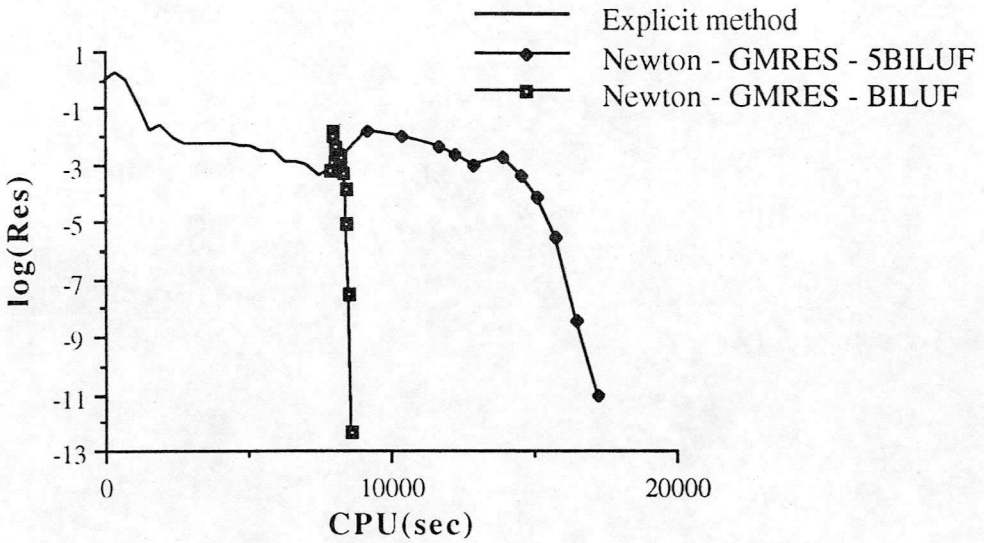


Fig.6.2 Convergence history for N-S solution for 66×66 grid

Table 1 CPU time (second) for one step implementation

	Explicit iteration	Matrix Generate	Diagonal Precon.	BILUF	GMRES (k=30)	GMRES (k=50)	CGS
34×34	0.963	4.790	0.39	0.85	5.63	–	0.44
66×66	3.622	18.63	1.62	3.36	–	40.00	1.96

Table 2 Convergence history for 66×66 grid

	Explicit		Newton-GMRES		Newton-CGS	
Num. Iter	Time	Res up to	Time	Res up to	Time	Res up to
1-2000	7866.48	6.16×10^{-4}	-	-	-	-
2000-2001	-	-	67.11	1.69×10^{-2}	78.00	1.69×10^{-2}
2000-2002	-	-	132.60	1.01×10^{-2}	147.07	1.01×10^{-2}
2000-2003	-	-	198.53	4.62×10^{-3}	209.66	4.62×10^{-3}
2000-2004	-	-	263.89	2.52×10^{-3}	277.36	2.52×10^{-3}
2000-2005	-	-	329.32	1.07×10^{-3}	346.80	1.07×10^{-3}
2000-2006	-	-	395.01	1.76×10^{-3}	409.24	1.75×10^{-3}
2000-2007	-	-	460.73	4.70×10^{-4}	470.15	4.69×10^{-4}
2000-2008	-	-	525.83	1.42×10^{-4}	530.77	1.41×10^{-4}
2000-2009	-	-	590.75	7.97×10^{-6}	597.46	7.94×10^{-6}
2000-2010	-	-	655.75	3.03×10^{-8}	655.66	3.01×10^{-8}
2000-2011	-	-	720.44	4.58×10^{-13}	CGS fail	

Table 3 Convergence history for 34×34 grid

	Explicit		Newton-GMRES		Newton-CGS	
Num. Iter	Time	Res up to	Time	Res up to	Time	Res up to
1-1000	963.18	1.68×10^{-2}	-	-	-	-
1000-1001	-	-	12.18	3.89×10^{-3}	10.17	3.89×10^{-3}
1000-1002	-	-	24.15	1.47×10^{-3}	21.10	1.47×10^{-3}
1000-1003	-	-	36.03	1.86×10^{-5}	35.93	1.86×10^{-5}
1000-1004	-	-	47.94	2.06×10^{-8}	41.85	2.06×10^{-8}
1000-1005	-	-	59.85	5.13×10^{-14}	51.83	5.66×10^{-14}

Table 4 Memory requirements

	Explicit method	Newton-GMRES	Newton-CGS
34×34	574597 bytes	8330833 bytes	7297833 bytes
66×66	2035589 bytes	34765617 bytes	27370537 bytes

10^{o} Cone
AoA = 24^{o}
$M_{\infty} = 7.95$
$T_{\infty} = 55.4$ K
$T_{w} = 309.8$ K
$Re_{\infty} = 4.1\times10^{6}$
r = 0.1 m
66×66 grid

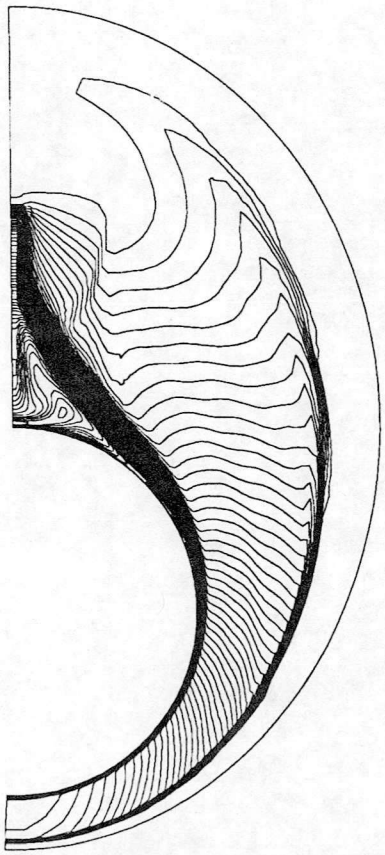


Fig.6.3 Flow conditions and cross flow temperature contours

Table 5 shows the divergence phenomenon of the GMRES linear solver with BILUF preconditioning when the matrix block elements are arranged as in (3.5).

Table 5 The residual of the $b-Ax$

N. I.	0	1	2	3	4	5	6	7	8	9	10
Res	3.268	0.772	0.685	0.659	0.654	0.652	0.648	0.642	0.635	0.628	0.621

7. Concluding Remarks

A very fast convergent Newton's solver has been developed for the steady state laminar locally conical Navier-Stokes equations. Fast convergence is obtained as a result of (1) suitable arrangement of the block elements of the Jacobian matrix. In each block row we arrange these block elements, which correspond to the cells that the physical phenomenon will change most rapidly along, to positions near the diagonal. (2) A fast BILUF preconditioner generation. In both grid cases the residual of the linear solution decrease by an order of 10 in one step of the GMRES(k) iteration, in which k=30 for the 34×34 grid and k=50 for the 66×66 grid. Therefore it is hopeful that this algorithm may be extended to even large grid cases.

From the numerical tests we see that the other arrangement of the block elements of the Jacobian matrix does not support a convergent linear solver. A special case is that when doing block incomplete LU factorization we consider only those block elements, near the diagonal, which correspond to the cells that the physical phenomenon change most rapidly along, i.e. the test of 5BILUF discussed above, in which we found that the convergence result can be obtained but requires increased CPU time.

The BILUF technique and solving lower and upper linear systems involve sequential bottle-necks. However, it is possible for the parallel calculation of the whole Navier-Stokes solver developed in this paper, since the proportion of the CPU time has been reduced for these sequential steps compared with other parallelizable steps [2-3]. That is, the calculation, which is not parallelizable, will have little effect on the efficiency of the parallelization of the complete Newton's method. Using parallel techniques we can divide the storage to individual processors, therefore solving the storage problem for the Newton's method.

REFERENCES

1. Xu, X., Qin, N., and Richards, B.E. "α-GMRES: A New Parallelizable Iterative Solver for Large Sparse Non-symmetric Linear System Arising from CFD", *Int. J. Numer. Methods Fluids.* Vol. 15, pp. 615-625, 1992.

2. Xu, X., Qin, N., and Richards, B.E. "Parallelization of Discretized Newton Method for Navier-Stokes Equations", *Proc. on Parallel CFD '92 Conf.*, Rutgers University, New Jersey, May 1992. Elsevier, Amsterdam, 1993.

3. Xu, X. "A Parallel Implementation of the Newton's Method in Solving Steady State Navier-Stokes Equations for Hypersonic Viscous Flows", Ph.D. Thesis, University of Glasgow, May 1993.

4. Qin, N., Xu, X., and Richards, B.E. "SFDN–α-GMRES and SQN–α-GMRES Methods for Fast High Resolution NS Simulations", *Proc. Conference on Numerical Methods for Fluid Dynamics.*, Reading, April 1992. Oxford University Press 1992.

5. Qin, N. and Richards, B.E. "Sparse Quasi-Newton Method for High Resolution Schemes", *Notes in Numerical Fluid Mechanics* Vol. 20, pp. 310-317, 1988.

6. Qin, N. and Richards B. E. "Sparse Quasi-Newton Method for Navier-Stokes Solutions", *Notes in Numerical Fluid Mechanics*, Vol. 29, pp. 474-483, 1990.

7. Whitfield, D.L. and Taylor, L.K. "Discretized Newton-Relaxation Solution of High Resolution Flux--Difference Split Schemes", AIAA Paper 91-1539, 1991.

8. Orkwis, P.D. and McRae, D.S. "A Newton's Method Solver for the Navier-Stokes Equations", AIAA Paper 90-1524, 1990.

9. Orkwis, P.D. and McRae, D.S. "A Newton's Method Solver for the Axisymmetric Navier-Stokes Equations", AIAA Paper 91-1554, 1991.

10. Orkwis, P.D. and McRae, D.S. "Newton's Method Solver for High-Speed Viscous Separated Flowfields", *AIAA J.* Vol. 30, (1), pp. 78-85, 1992.

11. Mallet, M. Periaux J. and Stoufflet, B. "Convergence Acceleration of Finite Element Methods for the Solution of the Euler and Navier-Stokes Equations of Compressible Flows", *Notes in Numerical Fluid Mechanics*, Vol. 20, pp. 199-210, 1988.

12. Wigton, L.B., Yu, N.J., and Young, D.P. "GMRES Acceleration of Compressible Fluid Dynamics Codes", AIAA Paper 85-1494, 1985.

13. Venkatakrishnan, V., "Newton Solution of Inviscid and Viscous Problems", *AIAA J.* Vol. 27, (7), pp. 885-891, 1989.

14. Venkatakrishnan, V., "Preconditioned Conjugate Gradient Methods for the Compressible Navier-Stokes Equations", *AIAA J.* Vol. 29, (7), pp. 1092-1100, 1991.

15. Saad, Y and Schultz, M.H. "GMRES: A General Minimal Residual Algorithm for Solving Nonsymmetric Linear System", *SIAM J. Stat. Comp.*, Vol. 7, No. 3, pp.

856-869, 1986.

16. Sonneweld, P., Wesseling, P. and de Zeeuw, P.M. "Multigrid and Conjugate Gradient methods as Convergence Acceleration Techniques", in: D.J. Paddon and M. Holstein, eds., *Multigrid Methods for Integral and Differential Equations* (Claredon Press, Oxford), pp. 117-167, 1985.

17. Mollenstadt, W. "Experimentelle Untersuchngen an Langsangestromten, Gepfeilten Echenkonfigurationen im Hyperschallbereich", Dissertation TU Braunschweig, 1984.

18. Tracy, R.R. "Hypersonic Flow over a Yawed Circular Cone", California Institute of Technology Aeronautical Laboratory Memorandum, No. 69, 1962.

19. Cross, E.J., Jr. and Hankey, W.L. "Investigation of the Leeward Side of a Delta Wing at Hypersonic Speeds", *AIAA J.* Vol. 6, (2), pp. 185-190, 1968.

20. Feldhun, R.F., Winkelman, A.E. and Pasiuk, L. "An Experimental Investigation of the Flowfield Around a Yawed Cone", *AIAA J.* Vol. 9, (6), pp. 1074-1081, 1971.

21. Stetson, K.F. Experimental Results of a Laminar Boundary Layer Separation on a Slender Cone at Angle of Attack at M=14.2", ARL 71-0127, Aerospace Research Laboratories, Wright-Patterson AFB, Ohio, 1971.

22. MacRae, D.S. "A Numerical Study of Supersonic Viscous Cone Flow at High Angle of Attack", AIAA Paper 76-97, 1976.

23. Bluford, G.S., Jr. "Numerical Simulation of the Supersonic and Hypersonic Viscous Flow around Thin Delta Wings", *AIAA J.* Vol. 17, (9), pp. 942-949, 1979.

24. Qin, N. and Richards, B.E. "Numerical Experiments with Hypersonic Flows beneath a Cone-Delta-Wing Combination", AGARD-CP-428, Paper 20, 1987.

25. Newsome, R.W. "A Comparison of Euler and Navier-Stokes Solutions for Supersonic Flow Over a Conical Delta Wing", AIAA Paper 85-0111, 1985.