INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 15, 613–623 (1992)

α-GMRES: A NEW PARALLELIZABLE ITERATIVE SOLVER FOR LARGE SPARSE NON-SYMMETRIC LINEAR SYSTEMS ARISING FROM CFD

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

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

SUMMARY

Linearization of the non-linear systems arising from fully implicit schemes in computational fluid dynamics often result in a large sparse non-symmetric linear system. Practical experience shows that these linear systems are ill-conditioned if a higher than first-order spatial discretization scheme is used. To solve these linear systems, an efficient multilevel iterative method, the α-GMRES method, is proposed which incorporates a diagonal preconditioning with a damping factor α so that a balanced fast convergence of the inner GMRES iteration and the outer damping loop can be achieved. With this simple and efficient preconditioning and damping of the matrix, the resulting method can be effectively parallelized. The parallelization maintains the effectiveness of the original scheme due to the algorithm equivalence of the sequential and the parallel versions.

KEY WORDS Large sparse non-symmetric linear system Multilevel iteration Generalized minimal residual method Parallel computing Distributed memory Computational fluid dynamics

1. INTRODUCTION

In the numerical solution of Euler and Navier–Stokes equations, there are two major classes of problems, steady and unsteady. For steady-state solutions, a time-dependent approach is usually followed using the unsteady governing equations. There are two advantages of doing so. Firstly, the starting of the solution is robust in the sense that non-physical states can easily be avoided as long as the initial flow field is physically defined and the time step is small enough so that a physical path can be followed during the process of the solution. Secondly, the same code can be used for both steady and unsteady problems if accuracy is maintained. However, this approach also brings out some problems. As an iterative procedure for steady-state solution, the physical path is not necessarily a fast convergence path. Acceleration techniques based on the time-dependent approach, such as local time stepping, multigrid and the use of approximate implicit operators, destroy the time accuracy and, therefore, the second advantage cannot normally be achieved.

In the time-dependent approach, the unsteady governing equations can be discretized in time by an explicit or an implicit method. Using an explicit method, the convergence for a steady-state problem can be extremely slow due to the stability restrictions on time steps even if some acceleration techniques were employed. Using an implicit method, unconditional stability can be achieved and as the time step approaches infinity the method approaches the Newton iterative method for the solution of the non-linear system corresponding to the steady-state problem.

0271–2091/92/170613–11$10.50

Received August 1991
Revised March 1992

However, it is generally not easy (1) to get the real Jacobian of the non-linear system and (2) to solve the resulting large sparse non-symmetric linear system. Previous researchers in CFD have tried to avoid these two difficulties in the following ways, respectively: (1) to construct simplified implicit operators, e.g. to use only first-order inviscid implicit operators; (2) to use approximate factorization for the multidimensional implicit operator so that the resulting linear system can be solved easily. Both of these naturally negate the advantages of the implicit scheme. The time step size for a simplified implicit method is still limited due to the inconsistency of the implicit operator and the right-hand side (the non-linear system) and the factorization 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.

Instead of avoiding the difficulties for a fully implicit method, Qin and Richards[1] tried to tackle the problem directly in order to achieve fast convergence for the steady-state solution. The sparse quasi-Newton method (SQN) and the sparse finite difference Newton method (SFDN) were proposed so that the difficulty in getting the Jacobian of the non-linear system is tackled.

After the linearization of the non-linear system is achieved, a large sparse non-symmetric linear system results. For one-dimensional problems, a block pentadiagonal matrix solver was devised to obtain a direct solution of the resulting linear system. For multidimensional problems the block line Gauss–Seidel iterative method was used. As pointed out by Qin and Richards,[1] the convergence of the method for the linear system is still not satisfactory if higher than first-order spatial discretization is used. A similar problem resulting from the use of high-order schemes was also found by Hemker and his colleagues[2,3,4] to achieve an effective application of the multigrid method. They introduced a defect correction technique to tackle the problem.

In this paper, we propose a new efficient multilevel iterative method for the solution of the sparse non-symmetric linear system arising from the application of the fully implicit method for steady-state solutions or from the SQN method and the SFDN method for the non-linear system corresponding to the steady governing equations. We denote the linear system by

$$Ax=b, \tag{1}$$

where the structure of A depends on the spatial discretization scheme used. Typically we consider the following system resulting from a second- or third-order high-resolution scheme using a structure grid for a two-dimensional Navier–Stokes solution. The linear system will be a block 13-point diagonal matrix which can be denoted as shown in Figure 1.

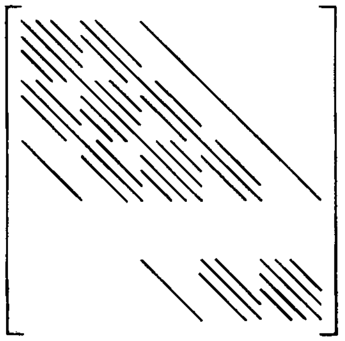

Figure 1. A block 13-point diagonal matrix

2. THE α-GMRES METHOD

2.1. *The GMRES method*

The generalized minimal residual (GMRES) algorithm was proposed by Saad and Schultz[5] for solving non-symmetric linear systems. It seeks a solution x in the form $x=x_0+z$, where x_0 is the initial guess and z belongs to the Krylov subspace $K=\langle r_0, Ar_0, \ldots, A^{k-1}r_0\rangle$ ($r_0=b-Ax_0$). The solution x is chosen such that $\|b-Ax\|$ is minimum.

First we find an orthonormal basis of space K via Gramm–Schmidt orthonormalization. In this process, a $(k+1)\times k$ Hessenberg matrix H_k is formed. The following calculations are performed.

Initially, we set

$$\hat{v}_1=r_0, \qquad v_1=\frac{r_0}{\|r_0\|},$$

and for $i=1$ to k

$$\hat{v}_{i+1}=Av_i-\sum_{j=1}^{i}\beta_{i+1,j}v_j, \quad \text{where } \beta_{i+1,j}=(Av_i, v_j)$$

$$v_{i+1}=\frac{\hat{v}_{i+1}}{\|\hat{v}_{i+1}\|}.$$

After k steps, the Hessenberg matrix is formed as

$$H_k=\begin{pmatrix} \beta_{2,1} & \beta_{3,1} & \cdots & \beta_{k+1,1} \\ \|\hat{v}_2\| & \beta_{3,2} & \cdots & \beta_{k+1,2} \\ 0 & \|\hat{v}_3\| & \ddots & \vdots \\ \vdots & \vdots & \ddots & \beta_{k+1,k} \\ 0 & 0 & \cdots & \|\hat{v}_{k+1}\| \end{pmatrix}_{(k+1)\times k}.$$

We then have[5]

$$\min_{z\in K}\|b-Ax\|=\min_{y\in R^k}\|\delta e_1-H_k y\|,$$

where $z=x-x_0$, $\delta=\|r_0\|$, $e_1=(1, 0, \ldots, 0)^{\mathrm{T}}_{k+1}$ and $y=(y_1, y_2, \ldots, y_k)^{\mathrm{T}}_k$.

The problem is now reduced to the solution of a smaller least-squares problem. Due to the special structure of the Hessenberg matrix H_k, a QR factorization algorithm can easily be applied.

For an efficient practical calculation, the dimension of the Krylov subspace k, is very small as compared to the order of the matrix A because storing all the previous directions is very costly. In application, the algorithm is restarted every k steps until the required accuracy is achieved. In the numerical tests given below, we choose $k=30$.

2.2. *Preconditioning and damping of the matrix*

The linear GMRES method has been applied to finite element solutions of CFD problems by Mallet *et al.*[6] and in its non-linear version by Wigton *et al.*[7] All successful applications required an efficient preconditioning. Bearing in mind the possible parallelization of the preconditioner, we devise a simple preconditioner. Its effectiveness is further enhanced by the introduction of a damping factor, which we describe as follows:

Let $D = \mathrm{diag}(A)$, such that if A is a block-structured matrix, D represents a block diagonal matrix. For equation (1), the following diagonal preconditioning is applied:

$$D^{-1}Ax = D^{-1}b. \tag{2}$$

This diagonal preconditioning has the following advantages: (1) it is simple to programme; (2) the operation is localized so that parallelization can be implemented effectively. However, it has been found from numerical tests of current problems that this simple preconditioning alone is not able to overcome the non-convergence using the GMRES method as illustrated in Figure 2.

We now introduce a damping factor α into equation (2):

$$(\alpha I + D^{-1}A)x = D^{-1}b \qquad (\alpha > 0). \tag{3}$$

It was found through numerical tests that equation (3) can now be solved very efficiently by the GMRES method. Figure 3 shows the convergences of the GMRES method as applied to equation (3) with different values of the damping factor α. The figure illustrates that the larger the α the faster the convergence. However, it should also be noted that with a very small α the non-convergence mentioned does still appear.

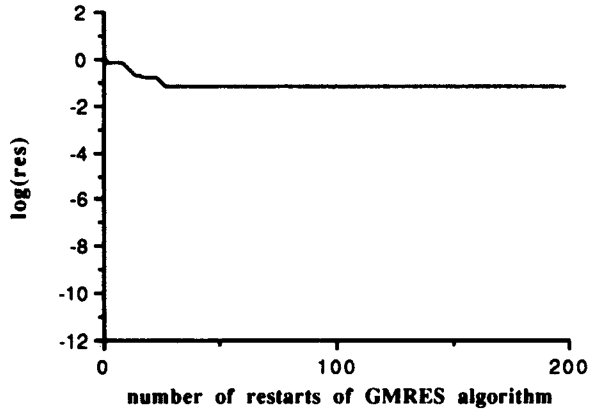

Figure 2. Convergence of GMRES algorithm for $(D^{-1}A)x = D^{-1}b$

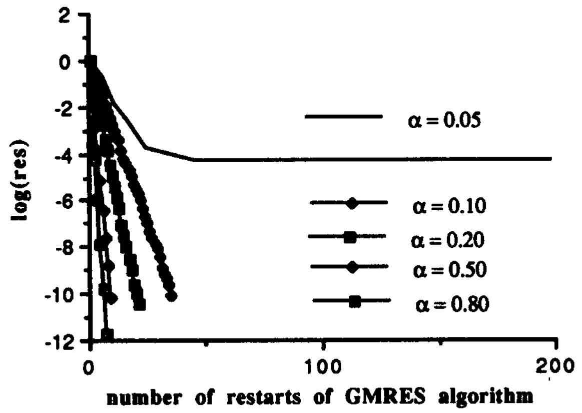

Figure 3. Convergence of GMRES algorithm for $(\alpha I + D^{-1}A)x = D^{-1}b$

2.3. *The α-GMRES multilevel iterative method*

It is clear that equation (3) is not equivalent to equation (2). To solve equation (2), an outer loop has to be introduced. This is done through a multilevel iterative scheme and written as

$$(\alpha I + D^{-1}A)x^{n+1} = D^{-1}b + \alpha x^n. \tag{4}$$

Given x^n, equation (4) is solved for x^{n+1} using GMRES method. This procedure is continued until the sequence x^n is converged. We have proved the following convergence theorem for the iterative procedure (4) as follows:

Theorem

(1) *If x^n converges to x^*, x^* will be the solution of Equation* (2).
(2) *There exists a positive number $\beta > 0$ such that if $0 < \alpha < \beta$, the iterative procedure* (4) *converges.*

Proof.

(1) This is an obvious result of equation (4).
(2) From equation (4), we have

$$\begin{aligned} x^{n+1} - x^n &= (\alpha I + D^{-1}A)^{-1}[(D^{-1}b + \alpha x^n) - (D^{-1}b + \alpha x^{n-1})] \\ &= \alpha(\alpha I + D^{-1}A)^{-1}(x^n - x^{n-1}) \\ &= \alpha^n(\alpha I + D^{-1}A)^{-n}(x^1 - x^0). \end{aligned}$$

Thus

$$\|x^{n+1} - x^n\| \leqslant \alpha^n \|(\alpha I + D^{-1}A)^{-n}\| \, \|x^1 - x^0\| \leqslant [\alpha \|(\alpha I + D^{-1}A)^{-1}\|]^n \|x^1 - x^0\|.$$

Let us define a positive function f, $f(\alpha) = \|(\alpha I + D^{-1}A)^{-1}\|$. The function f is obviously a continuous function of α and $f(0) = \|D^{-1}A\|$ is a constant. From the continuity of f, given a constant $c > 0$, we can find $\alpha_1 > 0$ such that when $0 < \alpha < \alpha_1$, we have

$$0 < f(\alpha) < f(0) + c.$$

On the other hand, for a given constant ε, $0 < \varepsilon < 1$, we can find $\alpha_2 > 0$ such that

$$\alpha_2[f(0) + c] < 1 - \varepsilon.$$

Let $\beta = \min\{\alpha_1, \alpha_2\}$ and choose α, $0 < \alpha < \beta$, we have

$$\alpha \|(\alpha I + D^{-1}A)^{-1}\| = \alpha f(\alpha) < \alpha_2[f(0) + c] < 1 - \varepsilon.$$

Thus,

$$\|x^{n+1} - x^n\| < (1 - \varepsilon)^n \|x^1 - x^0\|.$$

Therefore we have proved the convergence of the iterative procedure (4).

In practical application, a value of α has to be selected to balance the convergence of the outer iterative procedure (4) and that of the inner GMRES algorithm.

2.4. *Numerical tests and discussion*

The foregoing numerical tests have been carried out on a typical matrix resulting from the use of the SFDN method to solve the locally conical Navier–Stokes equations for compressible flow. The spatial discretization scheme used is the Osher flux difference splitting scheme. The formal accuracy is third order for the convective fluxes and second order for the diffusive fluxes. The case is a laminar Mach 7·95 flow around a sharp cone with a cold wall and at an angle of attack of 24°. 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. The grid in the cross-section is 33 × 33. Thus, the resulting matrix to be solved is a block 13-point structured matrix of order 31 × 31 × 5. Figure 4 shows the convergence histories for different values of damping factor α. Let ε_1 be the convergence criterion of the inner GMRES algorithm and ε_2 be the convergence criterion of the outer loop of α-GMRES algorithm. In the calculation plotted in Figure 4, we choose $\varepsilon_1 = 10^{-1}$ and $\varepsilon_2 = 10^{-10}$. Since the main calculation time is spent in the inner GMRES algorithm, we use the total number of restarts of the GMRES algorithm as a unit to measure the progress of the calculation.

Table I shows the details of the calculation for different α. It should be noted that for the case of $\alpha = 0{\cdot}05$ the GMRES algorithm cannot converge to the machine zero but this does not influence the convergence of the α-GMRES algorithm because the full convergence of the inner iteration is not required. From this table we can also see that the performance of the multi-iterative method is not sensitive to the choice of α tested.

Table II shows the CPU time required using different computers for solving the linear system of the test case where $\alpha = 0{\cdot}1$, $\varepsilon_1 = 0{\cdot}5$ and $\varepsilon_2 = 10^{-10}$.

Figure 5 shows the overall convergence of the solution of the NS equations using the SFDN and SQN methods. As for the Newton method, a good initial guess is an important aspect for

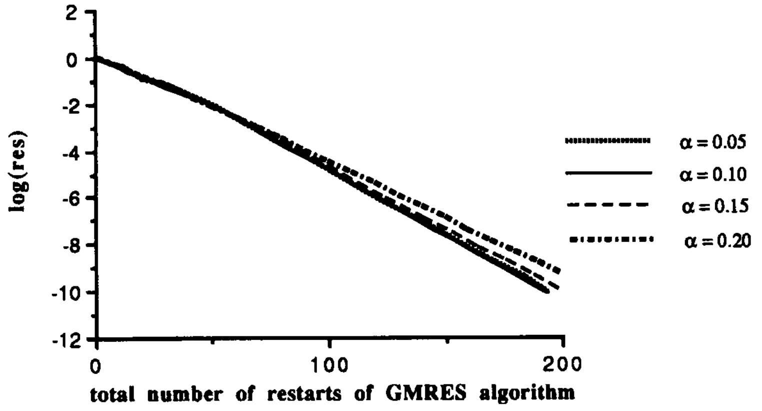

Figure 4. Convergence of α-GMRES algorithm for $Ax = b$

Table I. Effects of different α

α	Iterative number of outer loop of α-GMRES algorithm	Total restart number of GMRES algorithm
0·05	62	193
0·10	110	192
0·15	159	201
0·20	206	214

Table II. CPU time on different machines

Computer	CPU time (s)
IBM 3090 with vectorization	330
IBM 3090 without vectorization	776
IBM RS/6000	777
MEIKO CS with 1 T800 transputer	20852

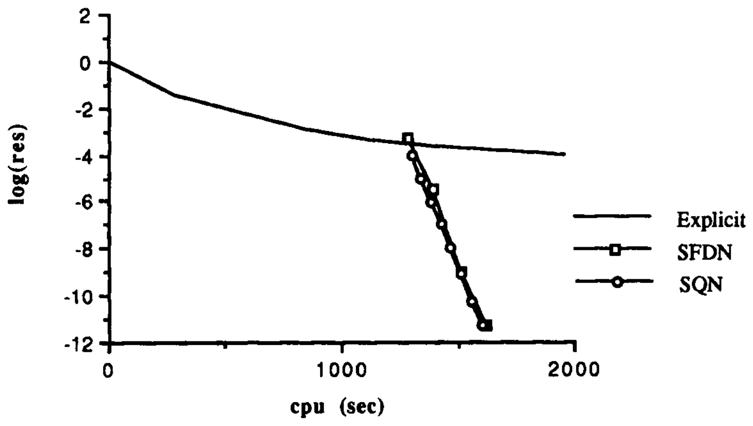

Figure 5. Convergence of SFDN and SQN methods using α-GMRES solver for NS solution

a successful application of the SFDN, SQN or other Newton-like methods. The initial guess used here was provided by an explicit time-dependent approach using the Runge–Kutta method with local time stepping, which is robust when starting the solution from free stream conditions but slow in convergence. Let ε_3 be the convergence criterion of the solution of the NS equations. In Figure 5 we have chosen $\varepsilon_1 = 10^{-1}$, $\varepsilon_2 = 10^{-2}$ and $\varepsilon_3 = 10^{-10}$ in the SFDN method and $\varepsilon_1 = 10^{-1}$, $\varepsilon_2 = 10^{-1}$ and $\varepsilon_3 = 10^{-10}$ in the SQN method.

3. PARALLELIZATION OF THE α-GMRES METHOD ON A DISTRIBUTED MEMORY PARALLEL COMPUTER

Parallelization of the GMRES method on a shared-memory parallel computer is straightforward. But on distributed-memory machines, which are becoming popular because of their low cost and ability to employ large overall memory, communication between processors has to be considered. Furthermore, preconditioning needs more serious consideration in the parallel environment. Incomplete LU (ILU) factorization as a preconditioner for the GMRES algorithm appears effective for many applications using a sequential computer. The full parallelization of ILU, however, is difficult to achieve. To apply ILU on vectorized shared-memory multiprocessors, Radicati and Robert[8] and Venkatakrishnan *et al.*[9] used local ILU techniques. Although it still serves as a useful preconditioner, its effectiveness is degraded as compared with a global ILU preconditioner on a sequential computer. The α-GMRES method presented above which combines a diagonal preconditioner with a damping procedure to provide an effective GMRES algorithm is fully parallelizable as described below. The parallelization maintains the effectiveness of the original scheme due to the algorithm equivalence of the sequential and the parallel versions.

3.1. *Matrix and vector storage*

Assume there are M processors available and the matrix A is of order N ($N > M$). We can write the matrix A in columns as

$$A = [A^1, A^2, \ldots, A^M],$$

where A^m is an $N \times L$ matrix, $m = 1, 2, \ldots, M$ and $L = N/M$ if $N/M - [N/M] = 0$; $L = [N/M]$ for some A^m and $L = [N/M] + 1$ for the other A^m if $N/M - [N/M] > 0$.

The vectors x and b can be written as

$$x = \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^M \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} b^1 \\ b^2 \\ \vdots \\ b^M \end{pmatrix},$$

where x^m and b^m are vectors of order L corresponding to A^m, $m = 1, 2, \ldots, M$.

With the splitting of matrix A and of vectors x and b described above, we store A^m, x^m and b^m in processor m.

3.2. *Parallelization of the multilevel iterative method*

From the GMRES algorithm described in Section 2.1, the main tasks of parallelization are (i) the parallelization of the block diagonal preconditioner, (ii) the product of a matrix and a vector and (iii) the inner product of two vectors. We describe these aspects in the following subsections before dealing with the overall scheme.

3.2.1. Parallelization of the block diagonal preconditioner. As described in Section 3.1, the matrix A is stored in the processors according to columns. Thus D^m is stored in processor m and $D^{-1}A$ results in a row transformation to A. In this way, some elements of D^m are required for the neighbouring processors and corresponding communication needs to be arranged.

3.2.2. Parallelization of a matrix–vector product. Let $y = Ax$, i.e.

$$\begin{pmatrix} y^1 \\ y^2 \\ \vdots \\ y^M \end{pmatrix} = (A^1, A^2, \ldots, A^M) \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^M \end{pmatrix};$$

we have

$$Ax = (A^1, A^2, \ldots, A^M) \left(\begin{pmatrix} x^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ x^2 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ x^M \end{pmatrix} \right)$$

$$= A^1x^1 + A^2x^2 + \cdots + A^Mx^M$$

$$= \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} + \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix} + \cdots + \begin{pmatrix} * \\ * \\ \vdots \\ * \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} y^1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ y^2 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ y^M \end{pmatrix},$$

where $\Rightarrow$ indicates the communication of data among different processors to form y. In this way, we divide the task of calculating Ax to M processors by calculating $A^m x^m$ on processor m and the resulting vector y is again distributed to the M processors. The only communication required in the calculation is in the formation of y. Due to the sparsity of the matrix A, this communication is only of a limited nature. The distribution of the matrix data in columns can be mapped to that carried out in the geometric domain decomposition approach to parallelization.

3.2.3. Parallelization of inner product of vectors. The calculation of the inner product of two vectors a and b is equal to the sum of the inner products of their corresponding components and, therefore, can easily be parallelized as illustrated below.

$$(a, b) = \sum_{m=1}^{M} (a^m, b^m).$$

3.2.4. Parallelization of the GMRES method. The GMRES method as outlined in Section 2.1 is implemented in parallel as follows. In processor m, we perform the following calculations and communications. Initially, we set

$$\hat{v}_1^m = r_0^m,$$

$$\|r_0\| = \sqrt{\left[\sum_{m=1}^{M} (r_0^m, r_0^m)\right]},$$

where the calculation of $\|r_0\|$ requires the collection of the partial inner products carried out on each processor, and obtain

$$v_1^m = \frac{r_0^m}{\|r_0\|}.$$

For $i = 1$ to k

$$A^m v_i^m = \bar{v}_i^m,$$

where the matrix–vector product operation and its parallelization have been discussed in Section 3.2.2. The elements of the Hessenberg matrix are calculated using

$$\beta_{i+1,j} = \sum_{m=1}^{M} (\bar{v}_i^m, v_j^m),$$

which also requires the collection of the partial inner products carried out on each processor. We then calculate

$$\hat{v}^m_{i+1} = \bar{v}^m_i - \sum_{j=1}^{i} \beta_{i+1,j} v^m_j,$$

and

$$\|\hat{v}_{i+1}\| = \sqrt{\left[\sum_{m=1}^{M} (\hat{v}^m_{i+1}, \hat{v}^m_{i+1})\right]},$$

which is again a collection procedure. Then we normalize the base vector as follows:

$$v^m_{i+1} = \frac{\hat{v}^m_{i+1}}{\|\hat{v}_{i+1}\|}.$$

After k steps, the Hessenberg matrix is

$$H_k = \begin{pmatrix} \beta_{2,1} & \beta_{3,1} & \cdots & \beta_{k+1,1} \\ \|\hat{v}_2\| & \beta_{3,2} & \cdots & \beta_{k+1,2} \\ 0 & \|\hat{v}_3\| & \ddots & \vdots \\ \vdots & \vdots & \ddots & \beta_{k+1,k} \\ 0 & 0 & \cdots & \|\hat{v}_{k+1}\| \end{pmatrix}_{(k+1)\times k}.$$

From

$$\min_{z\in K} \|b - Ax\| = \min_{y\in R^k} \|\delta e_1 - H_k y\|,$$

we solve the same least-squares problem on all the processors.

3.4. *Numerical tests and discussion*

The parallel α-GMRES algorithm has been tested on the University of Glasgow Meiko Computing Surface, which consists of 32 T800 transputers. The speed-up achieved using 1 to 4 processors is illustrated in Figure 6.

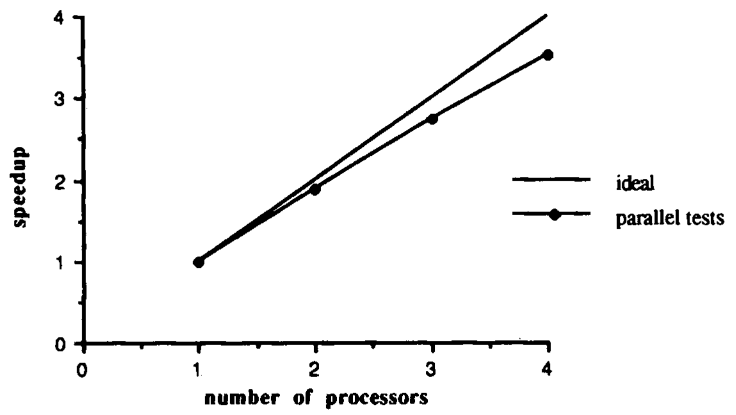

Figure 6. Speed-up using parallel computer

The parallel efficiency for 2, 3 and 4 processors are 93·8%, 91·7% and 88·1%, respectively. The parallel procedure produces the same results as those produced by the sequential procedure. Therefore, the accuracy and the convergence of the sequential procedure are maintained by the parallelization.

4. CONCLUDING REMARKS

An efficient multilevel iterative solver has been developed for the large sparse non-symmetric linear systems, which result from fully implicit or Newton-like solutions of the steady Navier–Stokes equations. Fast convergence, which is insensitive to the choice of α tested, has been achieved in solving a practical matrix problem. Parallelization of this new linear solver has also been presented showing promising results. In a fashion similar to a geometric domain decomposition approach to parallelize a code, the data of the matrix are distributed according to columns. The parallelization maintains the effectiveness of the original scheme due to the algorithm equivalence of the sequential and the parallel versions.

ACKNOWLEDGEMENTS

The first author thanks the University of Glasgow for a research scholarship and his friends who generously supported him. The second author gratefully acknowledges the support by U.K. Science and Engineering Research Council Grant GR/E 89056.

REFERENCES

1. N. Qin and B. E. Richards, 'Sparse quasi-Newton method for Navier–Stokes solutions', *Notes in Numerical Fluid Mechanics*, **29**, 474–483 (1990).
2. P. W. Hemker, 'Defect correction and high order schemes for multigrid solution of the steady Euler equations', *Lecture Notes in Maths*, **1128**, 150–165 (1986).
3. S. P. Spekreijse, 'Multigrid solution of the steady Euler equations', *Ph.D. Thesis*, CWI, Amsterdam, 1987.
4. B. Koren, 'Defect correction and multigrid for an efficient and accurate computation of airfoil flows', *J. Comput. Phys.*, **77**, 183–206 (1986).
5. Y. Saad and M. H. Schultz, 'GMRES: A general minimal residual algorithm for solving nonsymmetric linear system', *SIAM J. Stat. Comput.* **7**, 856–869 (1986)
6. M. Mallet, J. Periaux and B. Stoufflet, 'Convergence acceleration of finite element methods for the solution of the Euler and Navier–Stokes equations of compressible flows', *Notes in Numerical Fluid Mechanics*, **20**, 199–210 (1988).
7. L. B. Wigton, N. J. Yu and D. P. Young, 'GMRES acceleration of compressible fluid dynamics codes', *AIAA paper 85-1494*, 1985.
8. G. Radicati and Y. Robert, 'Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric linear systems on a vector multiprocessor', *Parallel Computing*, **11**, 223–239 (1989).
9. V. Venkatakrishnan, J. H. Saltz and D. J. Mavriplis, 'Parallel preconditioned iterative methods for the compressible Navier–Stokes equations', *Lecture Notes in Physics*, **371**, 233–237 (1990).