Cranfield
UNIVERSITY

Comparison of the convergence behaviour of three linear solvers for large, sparse unsymmetric matrices

Scott Shaw

COA Report No. 9504
July 1995

Department of Aerospace Science
College of Aeronautics
Cranfield University
Cranfield
Bedford MK43 0AL
England

Cranfield

College of Aeronautics Report No 9504
July 1995

Comparison of the convergence behaviour of three linear solvers for large, sparse unsymmetric matrices.

Scott Shaw

Department of Aerospace Science
College of Aeronautics
Cranfield University
Cranfield, Bedford MK43 0AL. England

ISBN 1871315573

£8

Abstract

Implicit methods for the calculation of unsteady flows require the solution of large, sparse non-symmetric systems of linear equations. The size of such systems makes their solution by direct methods impractical and consequently iterative techniques are often used. A popular class of such methods are those based upon the conjugate gradient method. In this paper we examine three such methods, CGS, restarted GMRES and restarted GMRESR and compare their convergence properties.

1. Introduction

Implicit methods for the calculation of unsteady flows require the solution of large, sparse systems of linear equations. While direct methods exist which can be used to obtain the solution of such systems of equations they become increasingly expensive with increasing grid size and are impractical for all but the most trivial of problems. Iterative techniques based upon the conjugate gradient method are most commonly used for the solution of systems of equations of this type. The basis of such methods is summarised below.

For an initial guess, $\{x_0\}$, to the system of equations,

$$[A]\{x\}=\{b\} \tag{1}$$

a residual $\{r_0\}$ is obtained thus,

$$\{r_0\} = \{b\} - [A]\{x_0\} \tag{2}$$

conjugate gradient type methods begin by assuming that the correction to $\{x_0\}$ required to set the residual to zero lies within a vector space (the Krylov subspace) constructed from a series of direction vectors. Once this subspace has been constructed a unit vector $\{p\}$ is then formed which reduces the residual and the solution is updated by,

$$\{x^{(i+1)}\}= \{x^{(i)}\} + \alpha\{p^{(i)}\} \tag{3}$$

There exist a large number of conjugate gradient type solvers which differ mainly in the way in which the subspace is constructed and the update vector $\{p\}$ is calculated. In this paper the performance of three such schemes, CGS[1], restarted GMRES[2] and restarted GMRESR[3], is compared.

2. The linear solvers.

(i) CGS

The conjugate gradient method is only valid for symmetric matrices, but has been extended for nonsymmetric matrices in Bi-CG. In this method two sets of residuals and two sets of direction vectors are updated at each iteration thus,

$$\begin{aligned} \{r_{i+1}\} &= \{r_i\} - \alpha_i [A]\{p_{i+1}\} \\ \{\tilde{r}_{i+1}\} &= \{\tilde{r}_i\} - \alpha_i [A^T]\{\tilde{p}_{i+1}\} \\ \text{and} & \\ \{p_{i+1}\} &= \{r_i\} + \beta_i \{p_i\} \\ \{\tilde{p}_{i+1}\} &= \{\tilde{r}_i\} + \beta_i \{\tilde{p}_i\} \end{aligned} \tag{4}$$

where α and β are chosen to ensure that the search directions are bi-orthogonal. Sonneveld [1] noted that the residual vectors {r} and { $\tilde{r}$ } could be expressed by the same polynomial in [A], that is,

$$\{r_i\} = P_i[A]\{r_o\} \quad \text{and} \quad \{\tilde{r}_i\} = P_i[A]\{\tilde{r}_o\} \tag{5}$$

an it follows that { $\tilde{r}$ } can be rewritten as a function of both {r} and [A],

$$\{\tilde{r}_i\} = P_i^2[A]\{r_o\} \tag{6}$$

Sonneveld suggested that if the contraction operator P[A] reduced the residual to some smaller vector (Equation (5)) then carrying out the contraction operator twice (Equation (6)) should be advantageous. Such an approach forms the basis of the CGS algorithm.

Unlike GMRES, CGS does not minimise the residual over the span of the subspace, this has two important consequences firstly the scheme is not very robust and secondly CGS can exhibit highly irregular convergence behaviour. Experience has shown that despite these weaknesses CGS can be quicker than other conjugate gradient type methods with the additional advantage that memory requirements are low as only one search direction need be stored at each iteration. Pseudo code for CGS is given in Listing (1).

(ii) Restarted GMRES

In GMRES the initial direction in the Krylov subspace is obtained as,

$$\{v_1\} = \frac{\{r_o\}}{\|r_o\|} \tag{7}$$

The remaining search directions are then computed using expressions of the form,

$$\{w\} = [A]\{v_i\} - \sum_{k=1}^{i}([A]\{v_i\},\{v_k\})\{v_k\}$$

$$\{v_{i+1}\} = \frac{\{w\}}{\|\{w\}\|} \tag{8}$$

The inner product coefficients are stored in the Hessenberg matrix, [H], which is used to solve the minimisation problem $\|r_o\{e_1\} - [H]\{y\}\|$. The coefficients of this minimisation problem, $\{y\}$, may then be used to update the solution using,

$$\{x_i\} = \{x_o\} + \{y\}[v] \tag{9}$$

In exact arithmetic GMRES will reduce the residual to zero over the full Krylov subspace, in practice it is not necessary to iterate for the full subspace and a truncation strategy is employed. The most common truncation strategy limits the subspace to the first m vectors, if the residual has not been reduced to an acceptable limit over this subspace then the solution vector is updated and the new system of equations which results is solved. This process is known as restarting.

While restarted GMRES(m) is more robust than CGS and has the desirable property that convergence is monotonic it requires more storage than CGS because of the need to store the Krylov subspace, m vectors each of length N (where N is the number of unknowns). The choice of the parameter m in restarted GMRES is problem dependant and is influenced heavily by the 'stiffness' of the problem and the overall size of the system which must be solved, when a good preconditioner is employed the number of vectors which must be stored can be relatively small (between 10 and 25). For some values of the parameter m it is possible that the solution process may stall. Pseudo code for GMRES is given in Listing (2).

(iii) Restarted GMRESR

GMRESR is a variant of GMRES proposed recently by Van der Vorst and Vuik[3] in which GMRES is used to improve the condition of the matrix [A]. This preconditioning is achieved by using a few iterations of GMRES to obtain an approximate solution ,$\{y\}$, of the system of equations $[A]\{y\}=\{r_i\}$. This is equivalent to stating that $[A^{-1}]\{r_i\}$ can be approximated by the polynomial in [A] which is implicit in the GMRES method.

The application of this method results in a two level scheme in which GMRES is used for both the inner and outer iterations, although any alternate linear solver, including GMRESR itself, could be used for the inner iteration. The outer iteration of GMRESR can be easily truncated which coupled with the requirement for very few vectors for the inner iteration offers the potential for large reductions in storage over more conventional GMRES schemes. Pseudo code for GMRESR is given in Listing (3).

3. Numerical scheme.

The flow field around moving aerofoils can be calculated using the unsteady, two dimensional thin layer Navier-Stokes equations which are written in generalised moving co-ordinates as,

$$\frac{\partial Q}{\partial t}+\frac{\partial F}{\partial \xi}+\frac{\partial G}{\partial \eta}=\frac{\partial S}{\partial \eta} \qquad (10)$$

Here Q is the vector of conservative variables, F and G are the inviscid flux vectors in the streamwise and normal directions and S is the viscous flux vector in the normal direction. Turbulent viscosity is calculated using the Baldwin-Lomax turbulence model.

Equation (10) is solved using a Beam-Warming type implicit scheme in which the inviscid flux vectors are discretised using Oshers method, with third order accuracy provided by MUSCL interpolation and a flux limiter, while the viscous flux vector is discretised using central differences. This discretisation results in a large, sparse system of linear equations which can be written in the form of Equation (1). The condition of this system of linear equations is improved using ADI preconditioning, see for example Wigton et al [4] and Badcock[5], and solved using a CG type linear solver.

4. Test Cases.

For the purposes of investigating and comparing the convergence behaviour of the linear solvers under consideration calculations are performed for a symmetric NACA 0012 aerofoil at zero degrees incidence undergoing inplane oscillations described by,

$$M_\infty = 0.536 + 0.61\sin(0.185\tau) \qquad (11)$$

where τ is the non-dimensional time. Calculations were performed for both the Euler equations and the turbulent Navier-Stokes equations (Re = 1.75 million). Euler solutions were obtained for the grid shown in Figure (11a) which has 153 nodes around the aerofoil (100 nodes on the aerofoil surface) and 48 nodes normal to the surface. Navier-Stokes solutions were obtained for the stretched grids shown in Figures (11b) and (11c), which have 153 x 48 and 259 x 96 nodes respectively.

Calculations were performed with 250, 600 and 3000 iterations per cycle of the aerofoil motion and the solution obtained by the linear solver was judged to be converged when the ratio of the residual norm to the norm of the right hand side was less than 0.005.

In Figures (12), (13) and (14) surface pressure coefficients, pressure contours and Mach number contours are presented for the current Navier-Stokes calculation at azimuth angles of 60, 90, 120 and 180 degrees. These Figures clearly illustrate the unsteady nature of the flow (compare the surface pressure distributions obtained for azimuth angles of 60 and 120 degrees which have the same incident Mach number) and the wide variety of flow conditions which are encountered.

5. Numerical experiments.

In Figure (1) the convergence behaviour of CGS, GMRES (10) and GMRESR(5,2) are compared for the first iteration of the inviscid calculation. This Figure shows quite clearly that initially both GMRES(10) and GMRESR(5,2) converge to the required tolerance in a shorter period of time than that taken by CGS. Experience has shown, however, that over the course of a full cycle of the aerofoil motion fewer iterations of all of the linear solvers are required. For the present calculation three iterations of CGS are required compared to a single iteration of GMRES(10) and one or two iterations of GMRESR(5,2). Consequently over the course of the full calculation CGS performs more efficiently than either of the alternatives considered, although this may not be the case for any individual timestep. This behaviour is demonstrated in Figure (2) which shows a comparison of the total time required by both CGS and GMRES(10) to calculate a full cycle of the aerofoil motion. The irregular convergence behaviour associated with CGS is demonstrated quite clearly by the results presented in Figure (1) and is in stark contrast to the smooth convergence of both GMRES(10) and GMRESR(5,2).

When the average residual over each time step is considered it is generally the case that GMRES is found to be more efficient than CGS. For the modest tolerances which are imposed on the convergence of the linear solvers in the current calculations the erratic convergence of CGS is of some benefit, allowing the required tolerance to be achieved in a shorter time than that required by GMRES. When more stringent conditions are placed upon the convergence of the linear solver the irregular convergence of CGS becomes disadvantageous and causes the solution process to become stalled at a much larger residual than is the case for GMRES.

For implicit calculations the size of the time step which is taken has important consequences for the condition of the Jacobian matrix (because the time step appears in the denominator term for the leading diagonal). For this reason the data presented in Figure (2), which shows calculation time against number of iterations per cycle of the aerofoil motion, also illustrates the comparative robustness of GMRES(10) and CGS for the current Euler calculation. GMRES was found to be significantly more robust than CGS for which convergence was unobtainable below 500 time steps per cycle.

In Figure (3) the convergence behaviour of CGS, GMRES(10) and GMRESR(5,5) is shown for the first iteration of the Navier-Stokes calculation performed using 600 time steps per iteration on the coarse, stretched grid shown in Figure (11b). It was found that CGS did not converge which is thought to be as a consequence of the poor condition of the Jacobian matrix. Converged results were obtained using both GMRESR(10) and GMRESR(5,2) demonstrating again the robustness of these two schemes.

The sensitivity of the convergence behaviour of GMRESR to the number of inner and outer iterations performed was investigated and results are compared with GMRES(10) in Figures (4) and (5) respectively for the first iteration of the Navier-

Stokes calculation (on the coarser grid). As the number of inner iterations is increased the ability of GMRES to act as a preconditioner to [A] increases leading to improvements in the efficiency of the outer iterations of GMRESR. This behaviour is demonstrated by the results shown in Figure (4) which indicates that a modest increase in the number of inner iterations from 2 to 5 has a dramatic impact on the rate of convergence of GMRESR (reducing by about 20% the required amount of CPU time) for this problem, while the increase from 5 to 10 inner iterations has a more limited impact. For the present calculation GMRESR(5,10) is computationally more efficient than GMRES(10), although this improvement is at the expense of an additional storage requirement. The results presented in Figure (5) show that increasing the number of outer iterations performed by GMRESR can lead to some small gains in performance. GMRESR(5,5) was found to have very similar convergence properties to GMRES(10), while for the current calculation GMRESR(10,5) was marginally quicker than GMRES(10).

The size of the time step taken in the viscous calculation was reduced until the condition of the Jacobian matrix, [A] became sufficiently good to allow CGS to converge to the required tolerance. In Figure (6) the convergence behaviour of CGS, GMRES(10) and GMRESR(5,5) are compared for a calculation performed with 3000 time steps per cycle. While CGS is now marginally quicker than both GMRES and GMRESR the calculation of a full cycle is likely to be much less efficient than the GMRES(10) calculation performed for 600 iterations per cycle.

These results suggest that the overall efficiency of the method depends very heavily on the choice of a suitable time step, unfortunately this information is often not available a priori. For this reason the size of time step is varied during the course of the calculation based upon the convergence behaviour of the previous iteration. If for a particular time step the residual does not meet the required convergence tolerance within N iterations (where N is a user defined integer) of the linear solver then the time step is reduced and that iteration is repeated. Conversely if the time step is considered to be too small, i.e. if the method converges one order below that required in a single iteration of the linear solver, then the time step is increased. In this way the timestep can be altered until a more computationally efficient value is obtained. While such an approach will eventually lead to a suitable size of time step the computation involved can be expensive. For the Navier-Stokes calculation performed using CGS in the previous paragraph fifty iterations of AF-CGS (each requiring 20 iterations of CGS) were necessary for the calculation of a single time step before the most suitable time step size was found.

The performance of GMRESR for the calculations detailed above is disappointing, in Reference (3) Van der Vorst and Vuik suggest that GMRESR will become increasingly competitive as the size of the GMRES subspace required to solve the system of equations increases. For the surprisingly small GMRES subspace required for the calculations detailed above it is therefore unlikely that GMRESR can ever be competitive. In order to investigate the benefits of GMRESR for more difficult, ‘stiffer’, systems of equations calculations were performed for the 259x96 node grid presented in

Figure (11c) with only 250 time steps per cycle. These changes have the combined effect of increasing the size of the system of equations which must be solved , reducing the effectiveness of the preconditioning and worsening the condition of the Jacobian matrix. In Figure (8) the convergence behaviour of GMRES is presented for Krylov subspace sizes of 10, 30 and 50 vectors. The results indicate that the convergence of both GMRES(10) and GMRES(30) becomes stalled above the required tolerance indicating that the present calculation requires a large subspace.

The convergence behaviours of several variants of GMRESR, all of which require the same amount of memory, are presented in Figure (9). It is of interest to note that all of these variants show very similar behaviour and become stalled close to the required convergence tolerance. Further investigations have shown that it is only when the memory requirement of GMRESR approaches that of GMRES(50) that the required convergence tolerance is reached. In Figure (10) the convergence of GMRES(50) is compared with GMRESR(10,26). Both of these schemes require the same amount of memory and the results shown in Figure (10) show that their respective convergence histories are very similar, both converging to the required tolerances in around the same amount of time.

The amount of memory required by each of the three linear solvers considered has been estimated by counting the occurrence of vectors and matrices which are of the same order as the solution vector and is shown in Table (1) below. As expected the storage cost of CGS is much smaller than both GMRES and GMRESR. It is apparent that the storage requirement of GMRESR will only be smaller than that of GMRES when the GMRES subspace is considerably larger than those used in GMRESR.

Linear Solver	Memory required
CGS	11
GMRES(m)	5+m
GMRESR(n,m)	9+2n+m

Table (1) Memory requirements of CGS, GMRES and GMRESR.

6. Conclusions.

An investigation has been carried out to determine the relative merits of three linear solvers, CGS, GMRES and GMRESR, for the solution of the large, sparse, non-symmetric system of equations which arise in the calculation of unsteady flows. No clear 'best' linear solver has emerged from the calculations which have been performed. It has been found that when convergence can be obtained using CGS it is generally faster than both GMRES and GMRESR, however CGS exhibits irregular convergence and does not appear to be very robust. It is thought that the utility of the CGS method , when compared with GMRES and GMRESR, will decrease as more stringent conditions are placed on the convergence of the linear solver due to its erratic convergence behaviour. For the current class of problems it has been found that the required solver tolerance is modest and consequently CGS has emerged as the most efficient of the methods when a suitable time step is selected.

Surprisingly the condition of the Jacobian for this class of problems appears, for reasonable sized time steps, to be well behaved and only very small subspace sizes are required for GMRES like methods. Both GMRES and GMRESR exhibit smooth monotonic convergence and are more robust than CGS (as demonstrated by their ability to obtain a solution for the viscous test case) with a considerably larger time step than CGS. For the calculations which require a small GMRES subspace, for example the current Navier-Stokes calculations performed with the grid shown in Figure (11b), there is no clear benefit in using GMRESR and to do so would incur a storage penalty. For problems which require larger GMRES subspace sizes, such as the current ill conditioned Navier-Stokes calculation performed with the finer grid shown in Figure (11c), the use of GMRESR becomes much more competitive, as was suggested by Van der Vorst and Vuik. Despite the increased difficulty of this calculation we have still not been able to demonstrate a clear case for using GMRESR instead of GMRES.

At the present time the selection of a suitable time step seems to be of great importance in determining the overall efficiency of the method. Currently the most efficient time step is found by considering the convergence behaviour of linear solver over each time step, if the convergence behaviour does not meet the required criteria then the size of the time step is altered and that iteration is repeated. For the Euler calculations which have been presented the convergence behaviour of all three linear solvers generally satisfied the required criteria and the time step was unchanged from that which had been specified. Similar behaviour was noted for Navier-Stokes calculations performed using GMRES and GMRESR, however for CGS it was found that the most suitable time step varied dramatically (from 500 to 3000 steps per cycle). Without an a priori knowledge of a suitable time step CGS was found to be inefficient because of the additional computation which was necessary to find a time step to satisfy all of the convergence criteria and consequently it is suggested that restarted GMRES should be used instead of CGS for viscous calculations.

References

1. Sonneveld, P. **CGS: A fast Lanczos type solver for non-symmetric linear systems.** SIAM Journal of Sci. Stat. Comp. 10 pp. 36-52 (1989).

2. Saad, Y. **GMRES: A generalised minimal residual algorithm for solving non-symmetric linear systems.** SIAM Journal of Sci. Stat. Comp. 7 pp. 856-69 (1986).

3. Vand der Vorst, H. Vuik, C. **GMRESR: A family of nested GMRES methods.** Numerical Linear Algebra with Applications Vol. 1(1) (1993).

4. Wigton, L.B. Yu, N.J. Yang, O.P. **GMRES acceleration of computational fluid dynamic codes.** AIAA Paper 85-1494 CP (1985).

5. Badcock, K.J. **The AF-CGS aerofoil code: Theory and User guide.** Glasgow University Report (1993).

```
For k=1,2,.........until converged.

    $\rho_1 = (r_o,r_{k-1})$
    $\beta = \rho_1/\rho_o$
    $u_k = r_{k-1} + \beta p_{k-1}$
    $p_k = u_k + \beta(q_{k-1} + \beta p_{k-1})$
    $v_k = [A]p_k$
    $\alpha = \rho_1/(r_o,v_k)$
    $q_k = u_k - \alpha v_k$
    $r_k = r_{k-1} - \alpha[A](u_k+q_k)$
    $x_k = x_{k-1} + \alpha(u_k+q_k)$

next k

{x}={xk}
```

Listing (1) : Pseudo code for CGS.

```
For j=1,2,........,until converged.

    $w = Av_j$

    For i=1,2.........,j

        $h_{i,j} = (w,v_i)$
        $w = w - h_{i,j} v_i$

    next i

    $h_{j+1,j} = \sqrt{}(w,w)$
    $v_{j+1} = w/h_{j+1,j}$

    Solve $H_m y_m = \sqrt{}(r_o,r_o)e_1$

next j

$x= x_o + Vy_m$
```

Listing (2) : Pseudo code for GMRES.

```
For $i = 0,1,2,$...........until converged.

    Use GMRES to solve for $z_m$ in $Az_m = r_i$

    $c = Az_m$

    For $k = 0,1,2,$.........,$i - 1$

        $\alpha = (c_k, c)$
        $c = c - \alpha c_k$
        $z_m = z_m - \alpha u_k$

    next k

    $\beta = \sqrt{(c,c)}$
    $c_i = c/\beta$
    $u_i = z_m/\beta$
    $x_{i+1} = x_i + (c_i, r_i) u_i$
    $r_{i+1} = r_i - (c_i, r_i) c_i$
```

Listing (3) : Pseudo code for GMRESR.

Figure (1) Comparison of the convergence behaviour of three linear solvers.

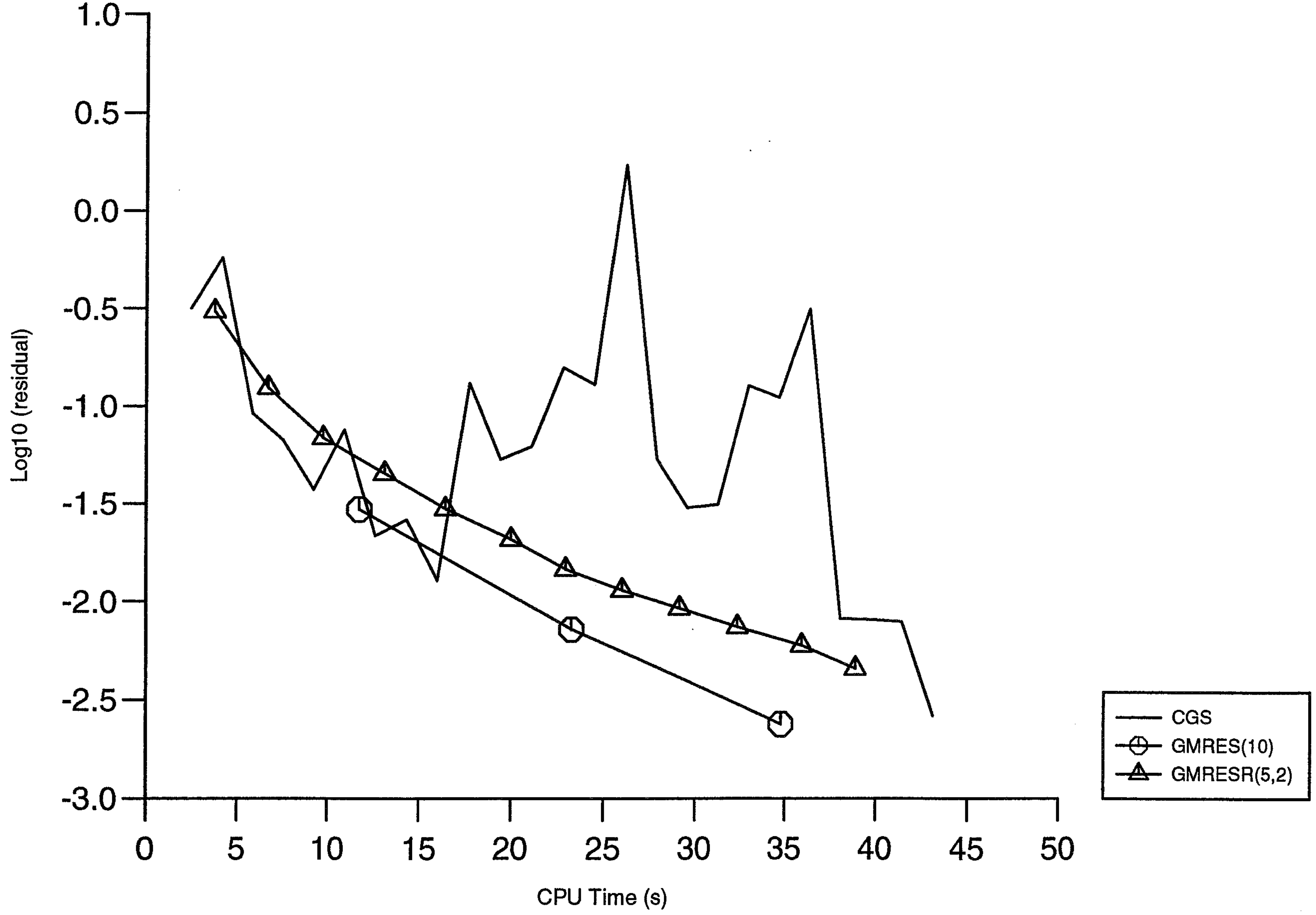


Figure (2) CPU time required for 1 full cycle of the aerofoil motion.

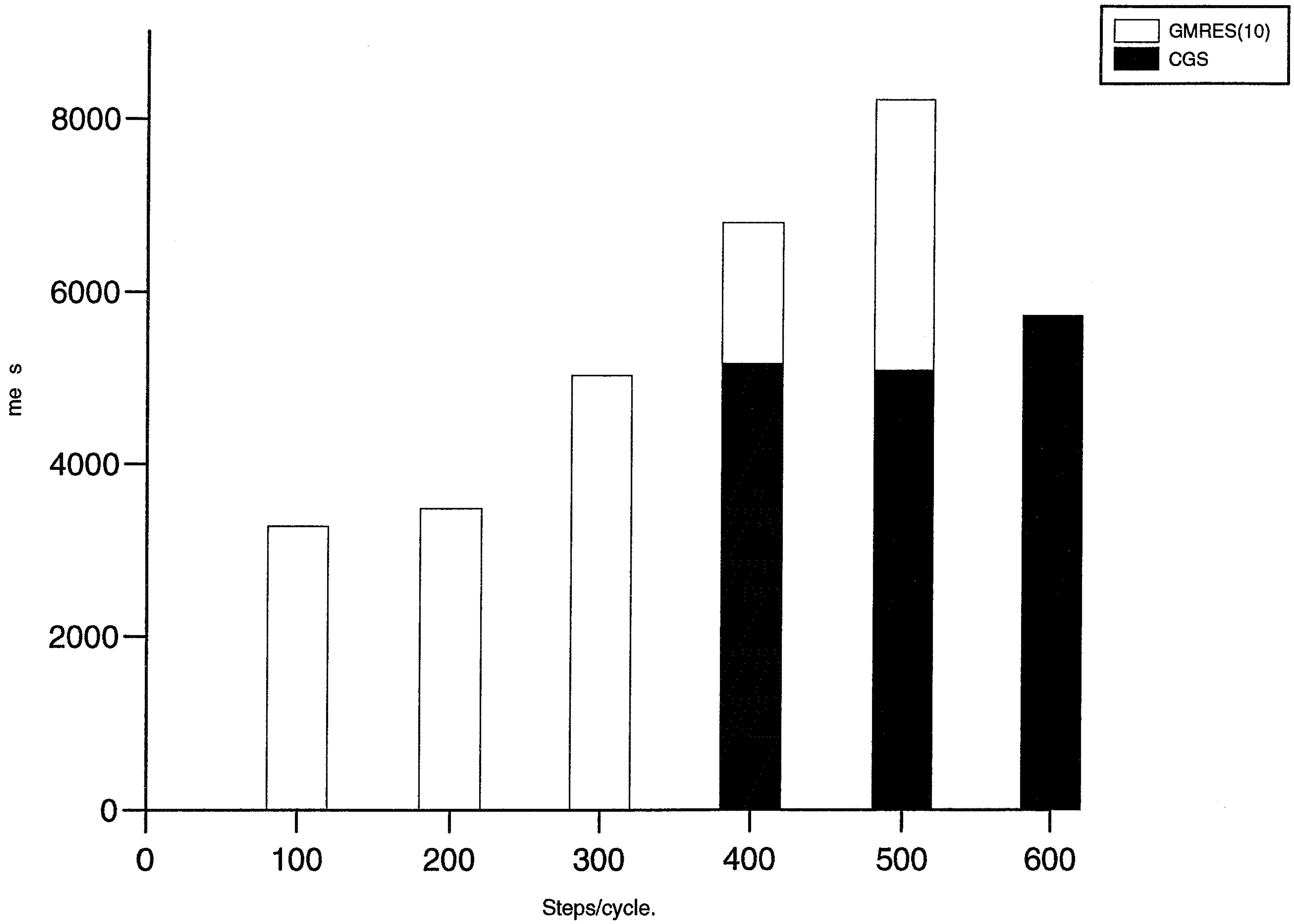


Figure (3) Comparison of the convergence behaviour of three linear solvers.

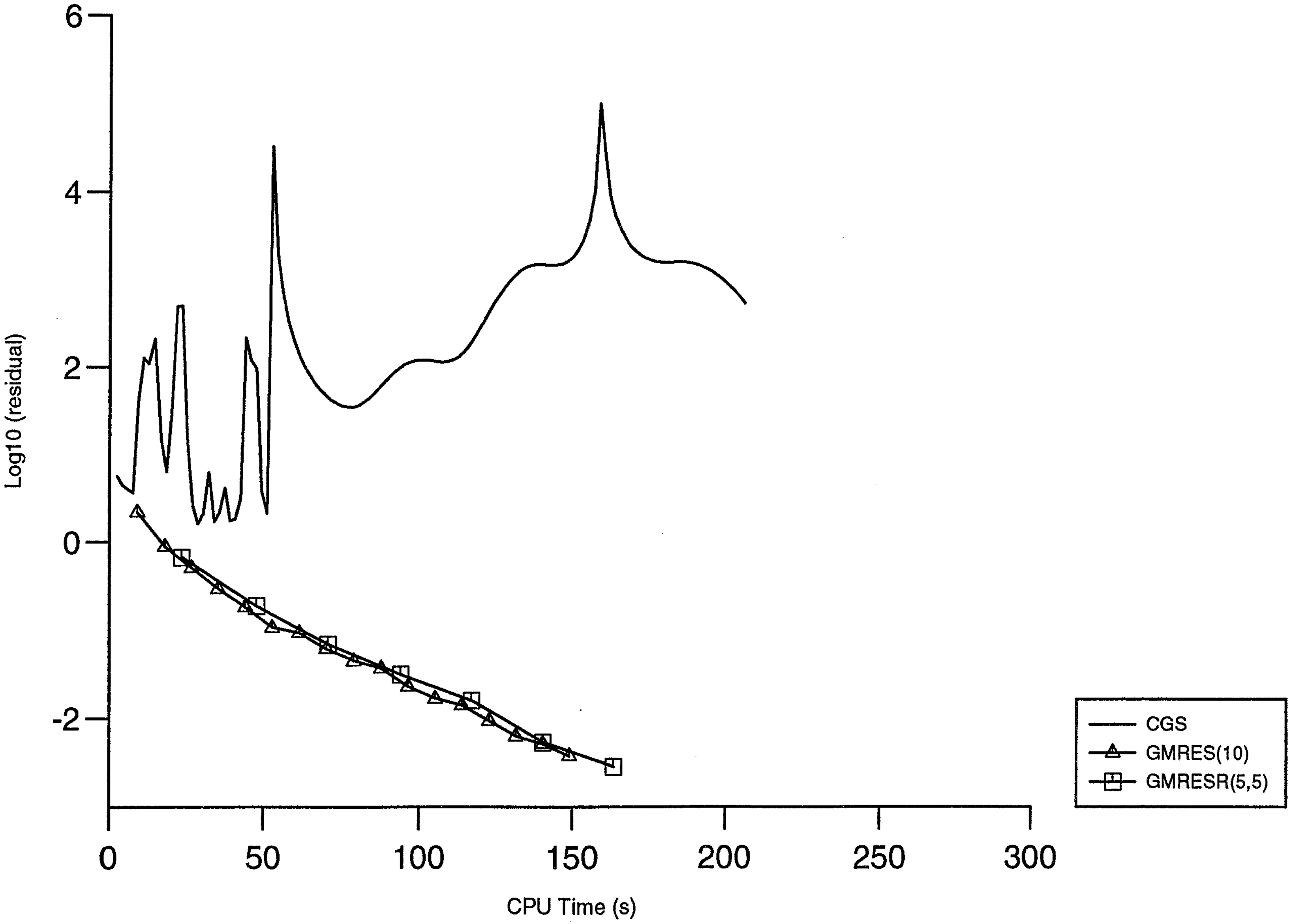


Figure (4) Effect of number of inner iterations on GMRESR.

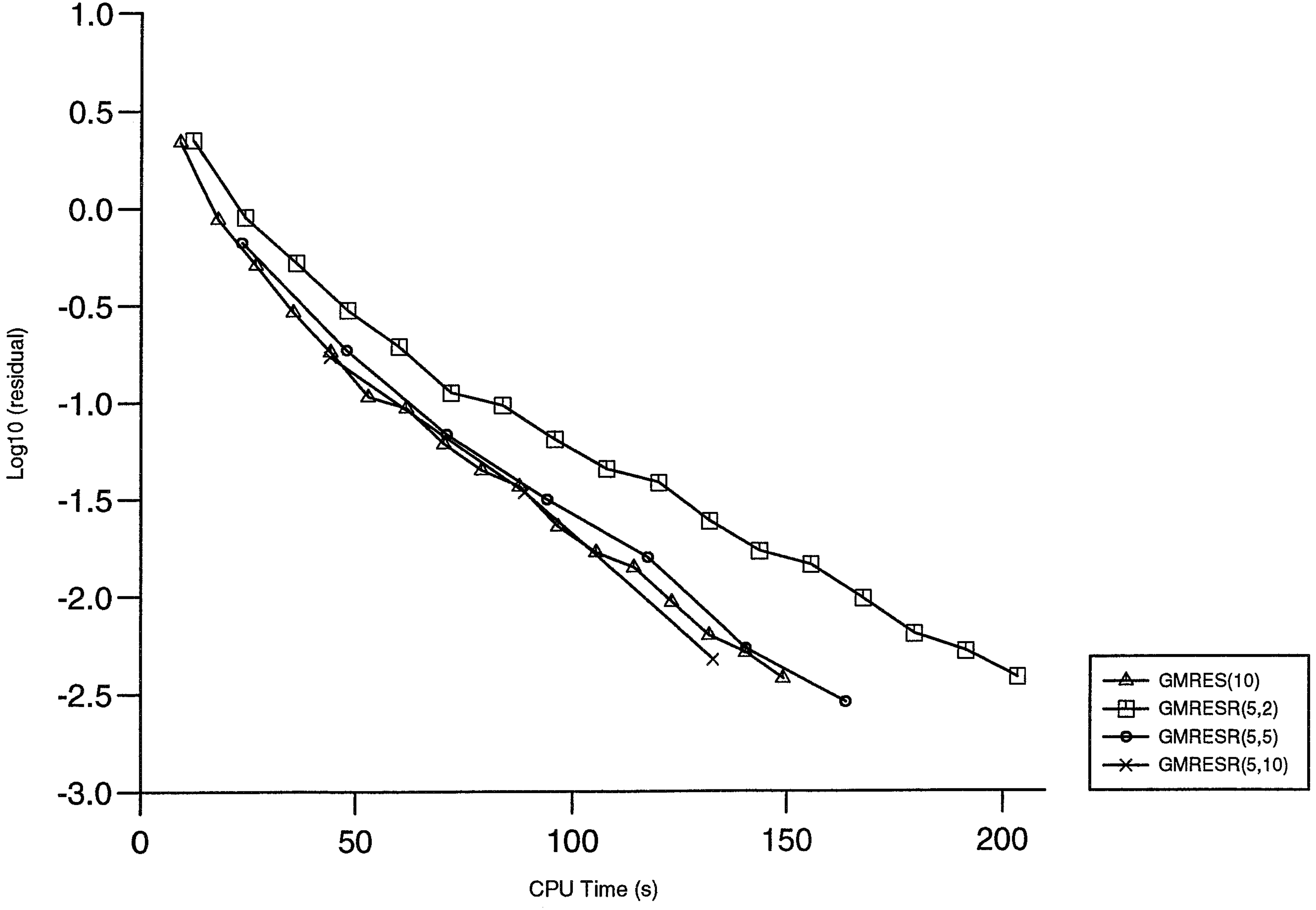


Figure (5) Effect of number of outer iterations on GMRESR.

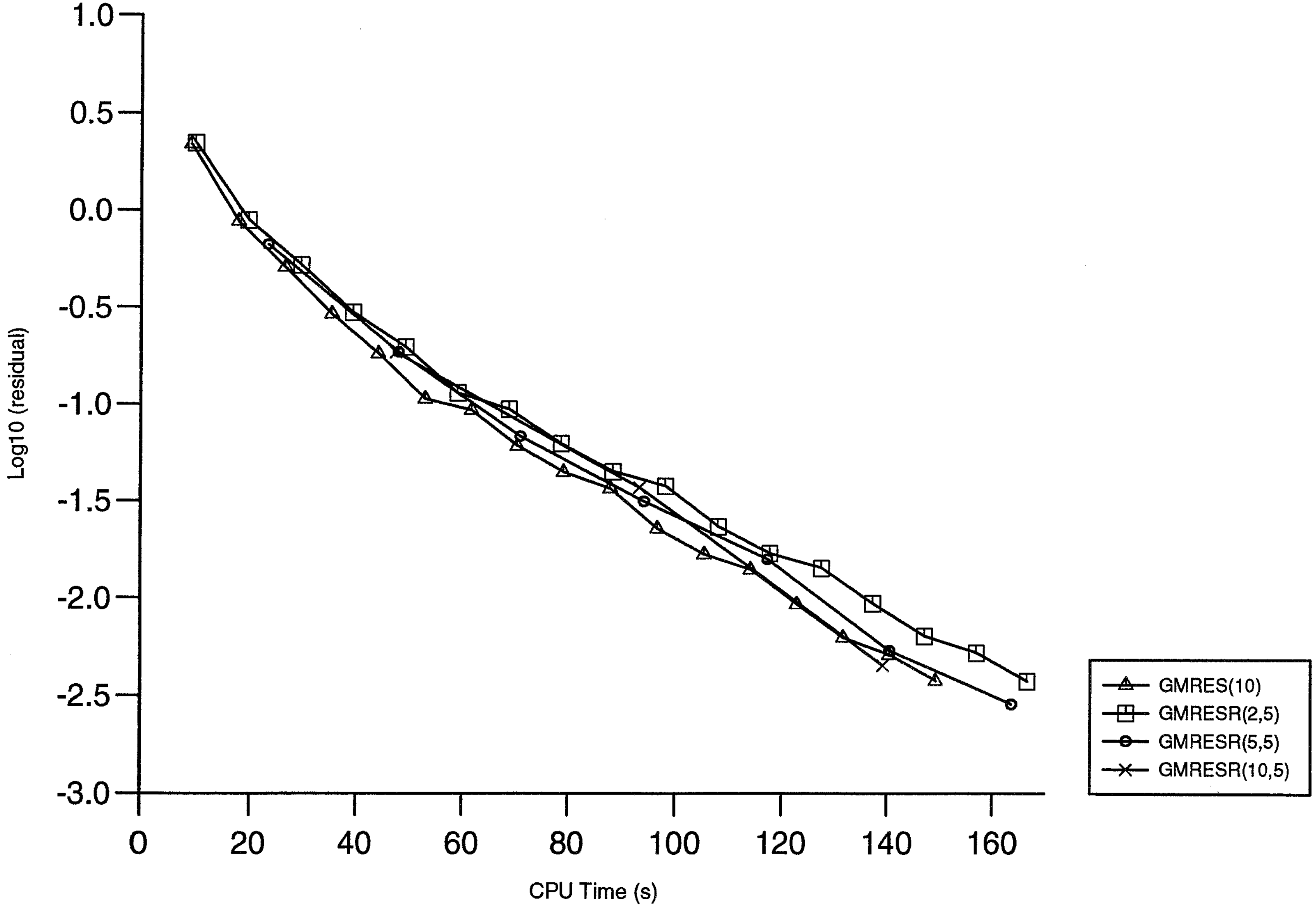
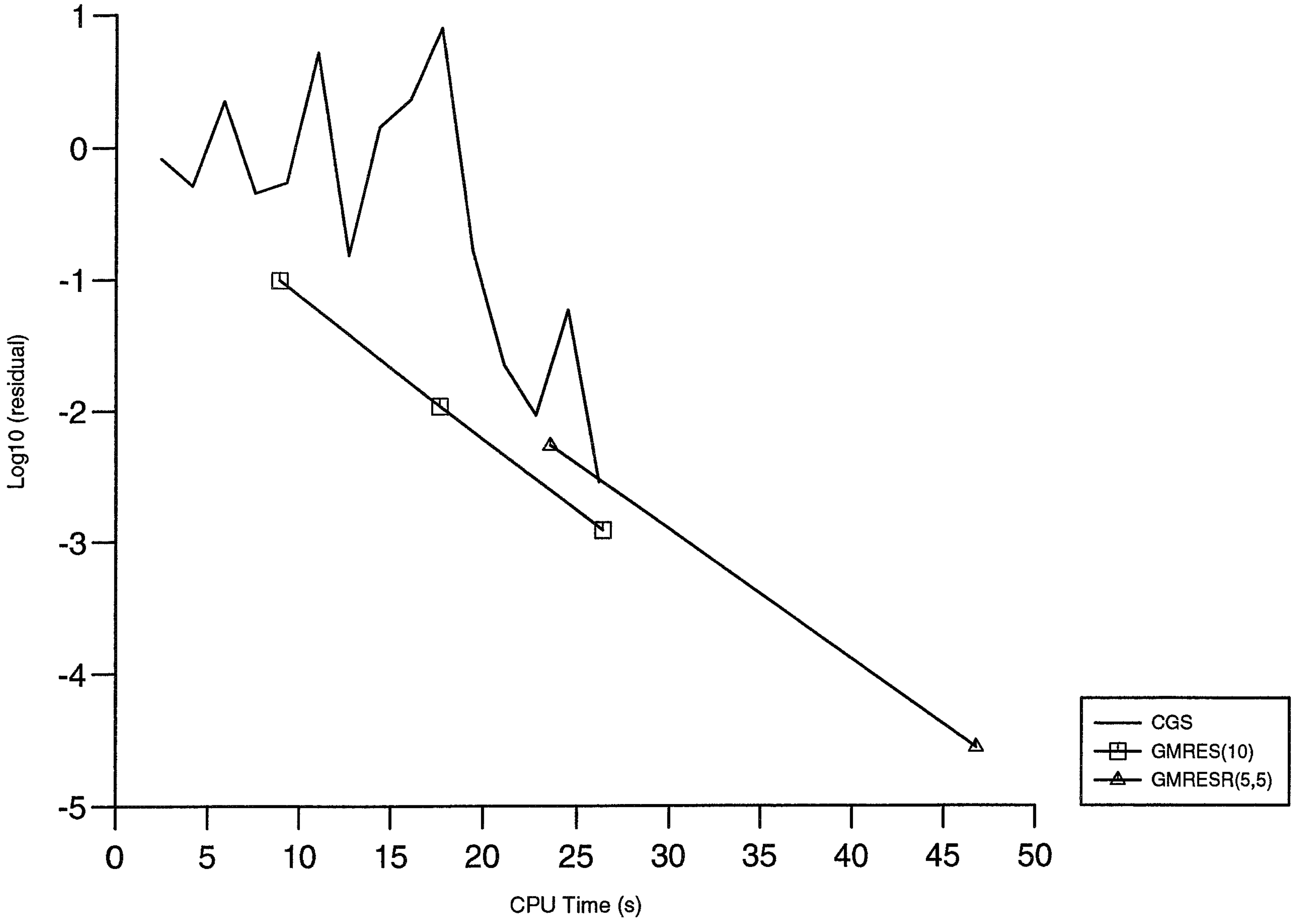
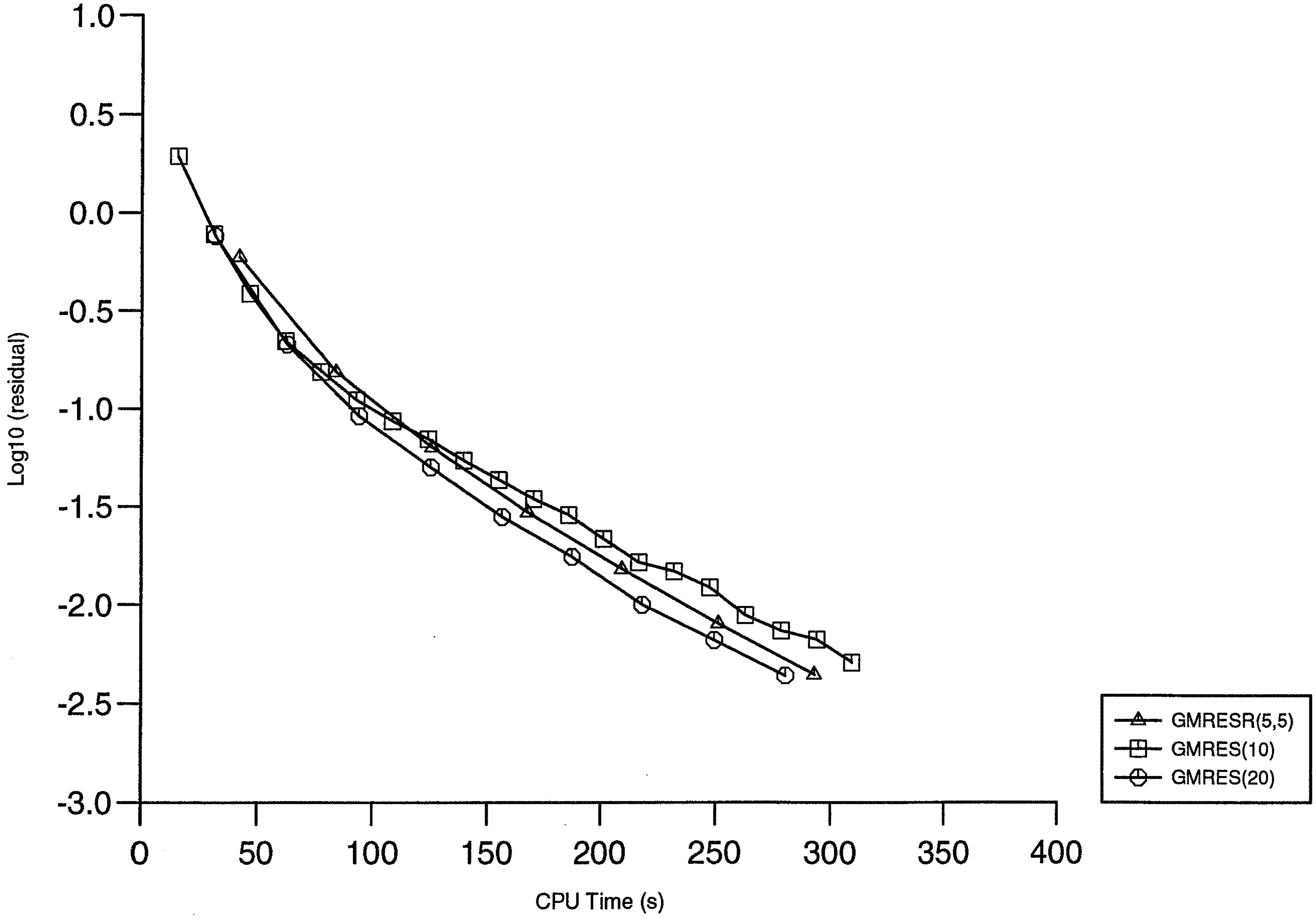


Figure (6) Comparison of the convergence behaviour of three linear solvers.



Figure(7) Comparison of convergence behaviour of GMRES and GMRESR.



Figure(8) Influence of Krylov subspace size on convergence of GMRES(m).

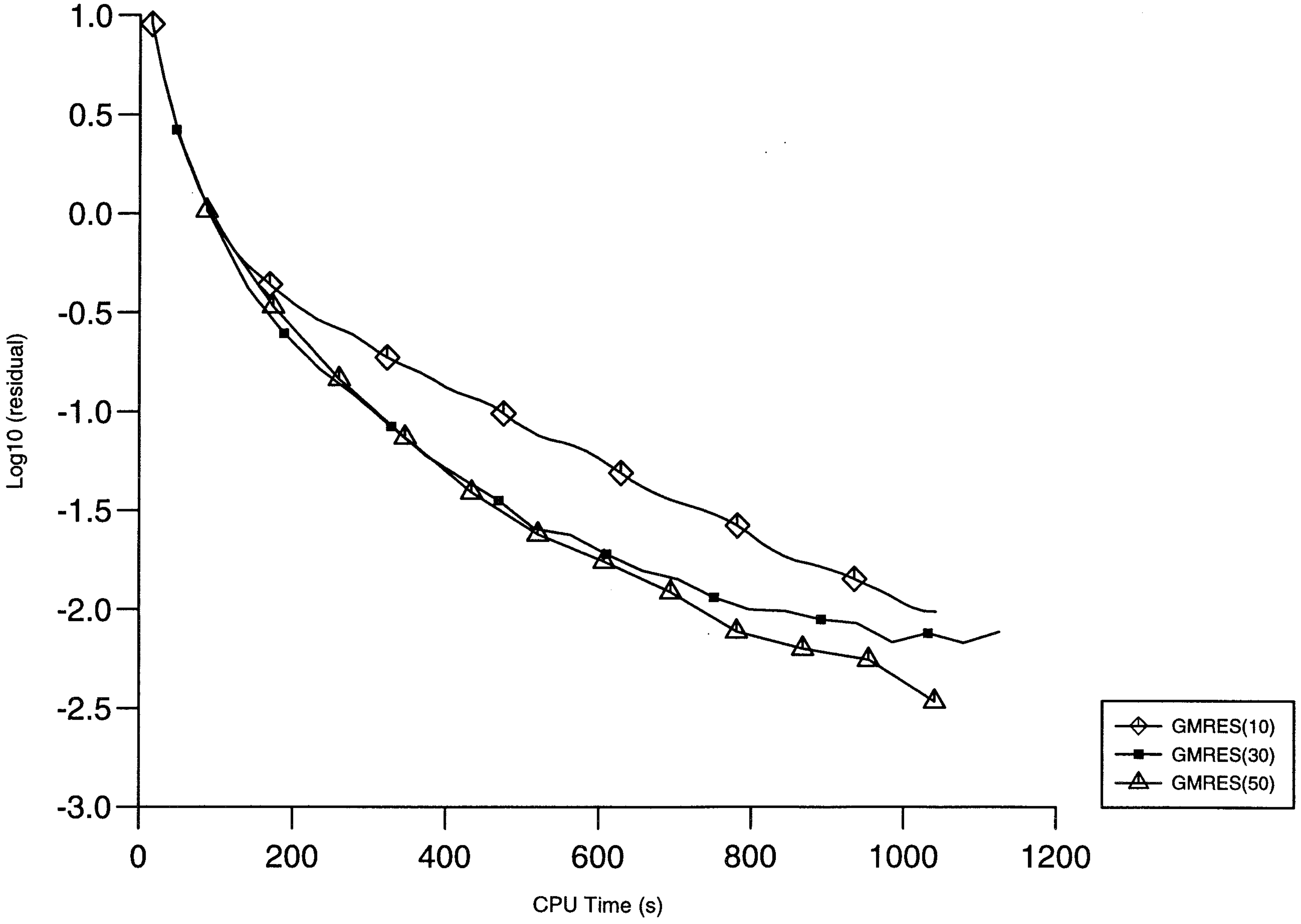


Figure (9) Comparison of GMRESR variants with a constant memory requirement.

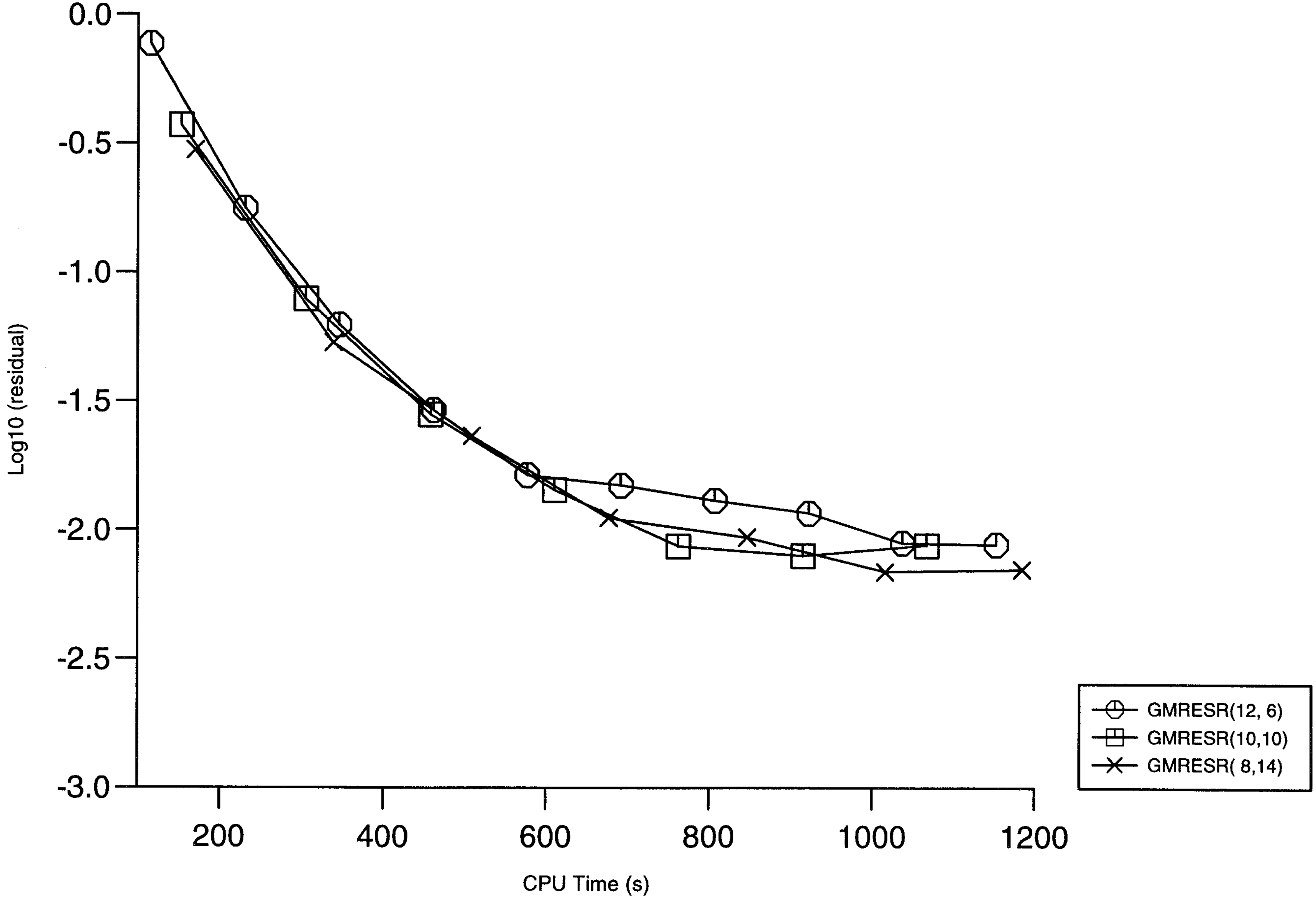
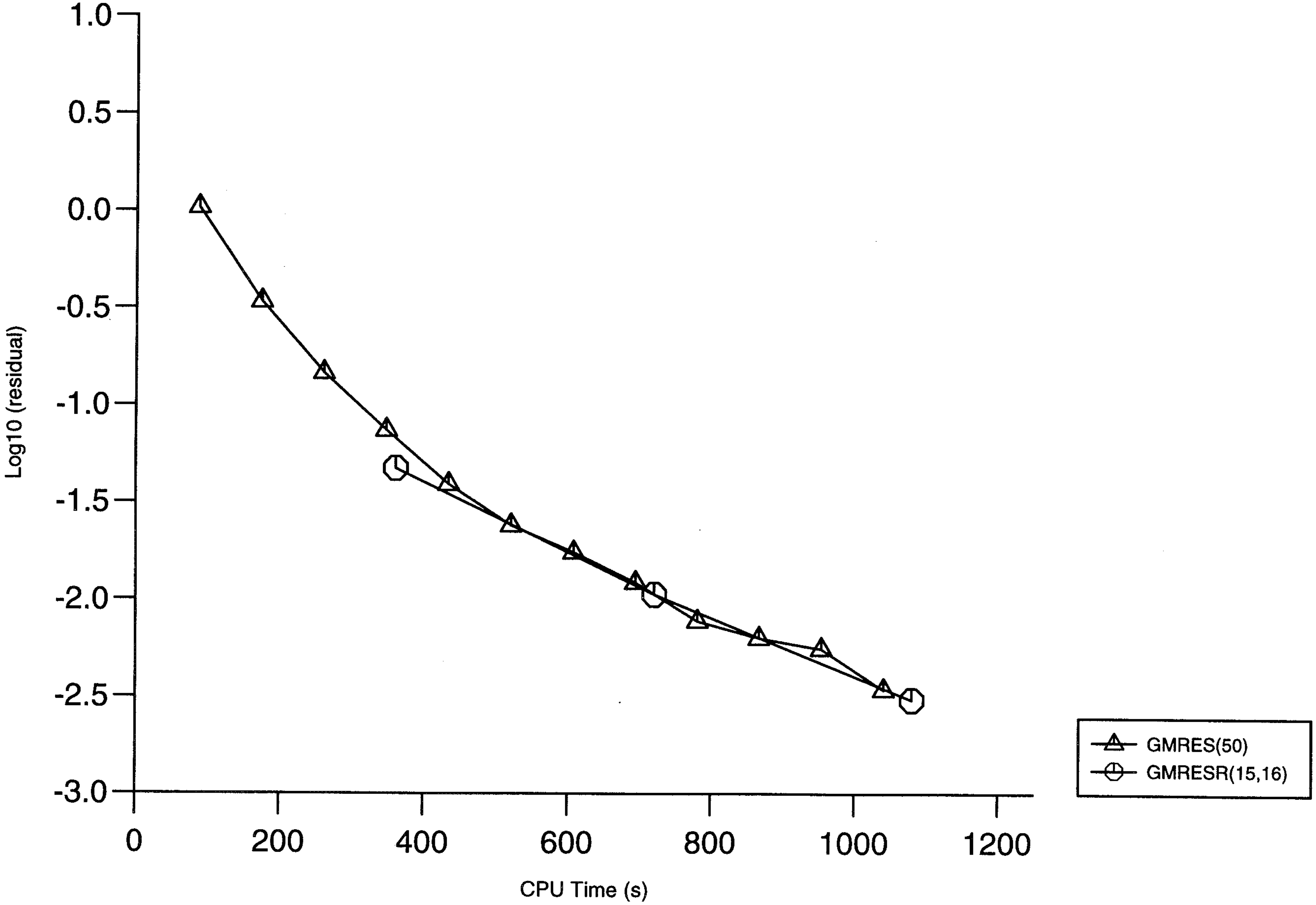
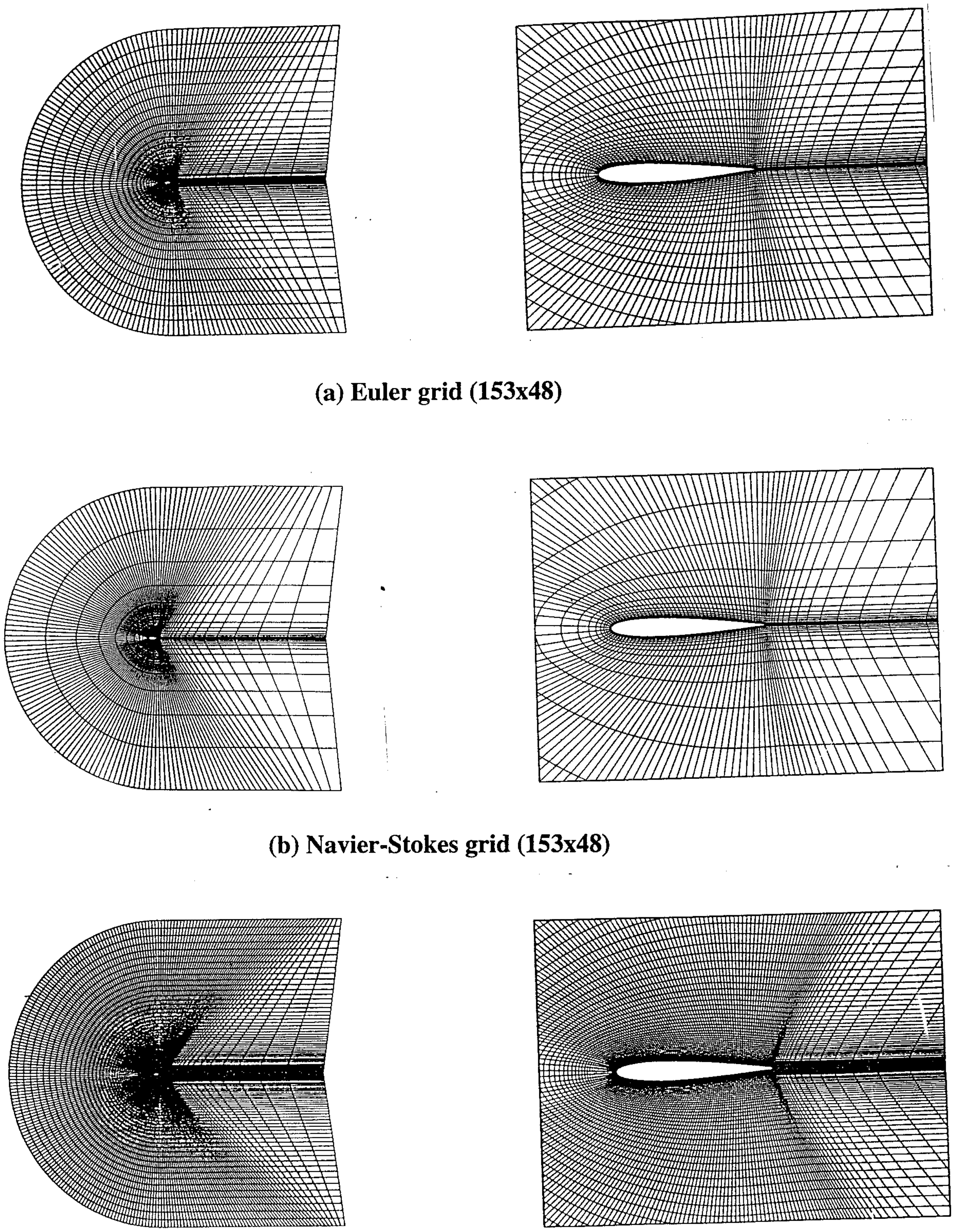


Figure (10) Comparison of convergence behaviour of GMRES and GMRESR.



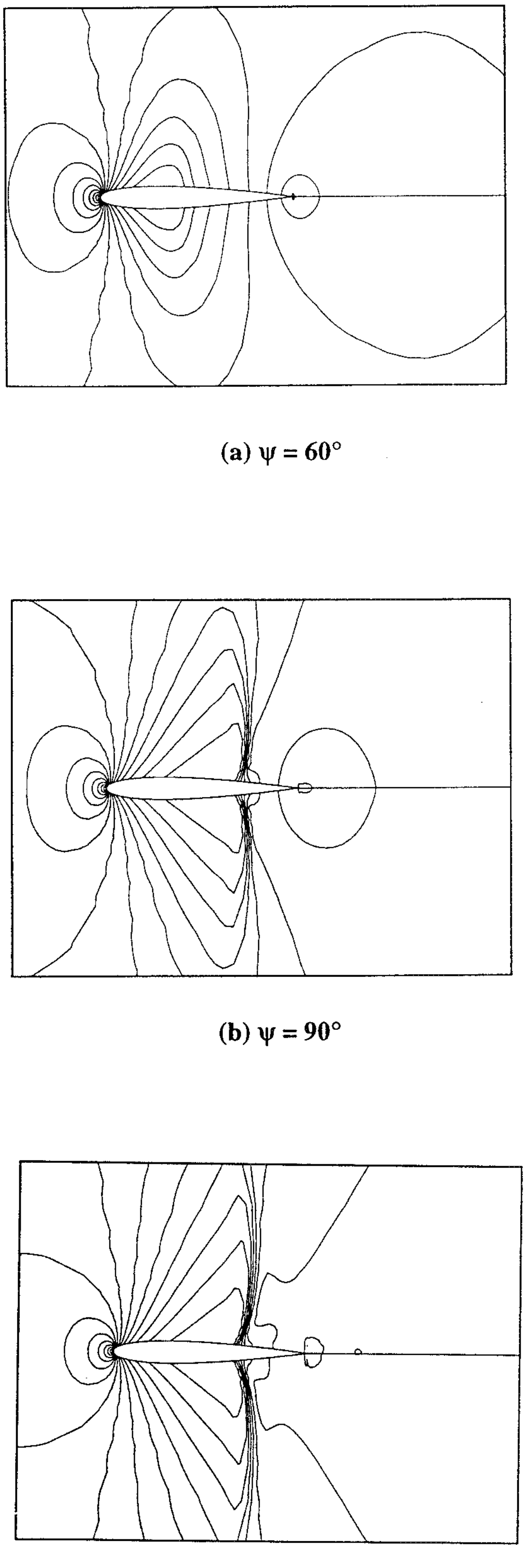


(a) Euler grid (153x48)

(b) Navier-Stokes grid (153x48)

(c) Navier-Stokes grid (259x96)

Figure (11) Computational Grids.

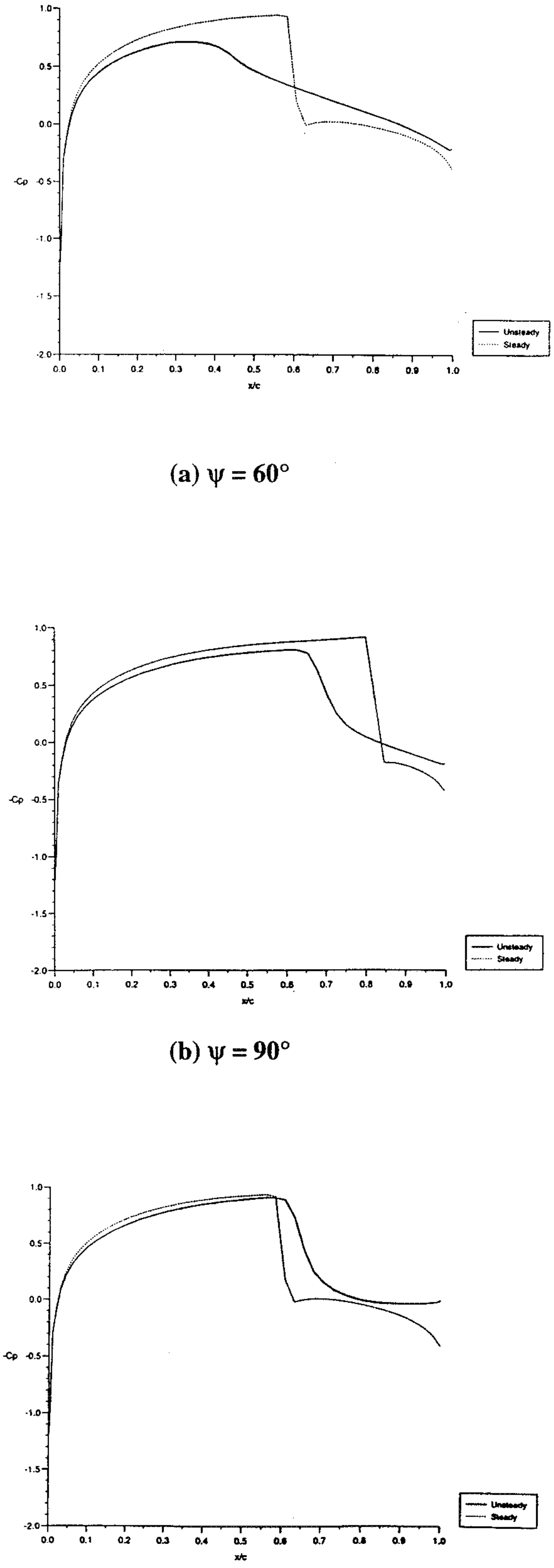


(a) ψ = 60°

(b) ψ = 90°

(c) ψ = 120°

Figure (13) Instantaneous local pressure contours.



(a) ψ = 60°

(b) ψ = 90°

(c) ψ = 120°

Figure (12) Surface pressure distributions.