SIMULATION OF SEMICONDUCTOR DEVICES AND PROCESSES Vol. 4
Edited by W. Fichtner, D. Aemmer - Zurich (Switzerland) September 12-14, 1991 - Hartung-Gorre

BI-CGSTAB in semiconductor modelling

Marjan Driessen
Nederlandse Philips Bedrijven B.V., Philips Research Laboratories, Eindhoven, The Netherlands

Henk A. Van der Vorst
Mathematical Institute, University of Utrecht, Budapestlaan 6, Utrecht, the Netherlands

Abstract

We investigate the use of the Bi-CGSTAB method [17] for solving the linear systems that typically occur when solving the coupled semiconductor equations. The investigations have been performed with the device modelling package Curry [9]. The Bi-CGSTAB method is compared with the CGS method [13], which was standard in Curry.

Over the last few years the linear solvers used in the software package Curry [9] have evolved to reasonably robust and satisfying modules. The package includes several linear solver modules, because the two nonlinear solution methods implemented (Newton, Gummel) as well as the various types of analysis (AC, DC, transient) give rise to different linear systems, requiring different linear solvers.
We will restrict ourselves to the linear solver for the systems which arise from using Newton's method on the coupled equations. This choice was made because most of the steady-state and transient calculations can be done using this solver, so we expect the test results for these problems to give a fair appraisal of the method.
The CGS [13] method works in general very well for this type of semiconductor problems. It converges much faster than BI-CG [5] and does not have the disadvantage of having to store extra vectors like in GMRES [12]. These three methods have been compared in many studies (see, e.g., [2],[1], [10], [8]).
But CGS usually shows a very irregular convergence behaviour. This behaviour can even lead to cancellation and a spoiled solution, which can prevent the nonlinear solver from converging. These problems are largely avoided in Bi-CGSTAB [17], a recently proposed conjugate gradient like iterative method with some very promising properties. Not only does it exhibit a more smooth convergence behaviour, so that cancellation usually does not play a role, but it often also takes less iterations than CGS. In the figures 1 and 2 we see two typical convergence histories for CGS and Bi-CGSTAB that we have obtained in our 2D semiconductor computations.
For the Bi-CGSTAB algorithm there are several interesting issues to consider, like the current conservation, choice of starting vectors, computation of parameters, and preconditioning. We will address these issues to some extent in this paper.

1 Bi-CGSTAB

The preconditioned Bi-CGSTAB algorithm for solving the linear system $Ax = b$, with preconditioning K reads as follows [17]:

x_0 is an initial guess;$r_0 = b - Ax_0$;
$\bar{r}$ arbitrary, such that $(\bar{r}, r_0) \neq 0$;
$\omega_0 = \rho_0 = \alpha = 1$; $v_0 = 0$; $i = 1$;
while x_i is not acceptable **do**
begin
  $\rho_i = (\bar{r}, r_i)$; $\beta = \left(\frac{\rho_i}{\rho_{i-1}}\right)\left(\frac{\alpha}{\omega_{i-1}}\right)$;
  $p_i = r_{i-1} + \beta(p_{i-1} - \omega_{i-1} v_{i-1})$;
  solve $\hat{p}$ from $K\hat{p} = p_i$
  $v_i = A\hat{p}$;
  $\alpha = \rho_i/(\bar{r}, v_i)$;
  $s = r_{i-1} - \alpha v_i$;
  solve $\hat{s}$ from $K\hat{s} = s$;
  $t = A\hat{s}$;
  $\omega_i = (K_L^{-1}t, K_L^{-1}s)/(K_L^{-1}t, K_L^{-1}t)$;
  $x_i = x_{i-1} + \alpha\hat{p} + \omega_i\hat{s}$;
  $r_i = s - \omega_i t$;
  $i = i + 1$;
end

The matrix K $(= K_L K_R)$ in this scheme represents the preconditioning matrix and the way of preconditioning [17]. The above scheme in fact carries out the Bi-CGSTAB procedure for the explicitly preconditioned linear system

$$K_L^{-1} A K_R^{-1} y = K_L^{-1} b,$$

but the vectors y_i and the residual have been backtransformed to the vectors x_i and r_i corresponding to the original system $Ax = b$.

Compared to CGS two extra innerproducts need to be calculated. Furthermore the calculation of $K_L^{-1}t$ does not appear in the CGS algorithm, but we will return to this subject in section 3.1.

There is an interesting relation between the residuals generated by BI-CG, CGS and BI-CGSTAB. In order to simplify notations we assume that these schemes are applied with $K = I$ (no preconditioning). Then it is well-known that the residual r_i^{BI-CG} can be written as

$$r_i^{BI-CG} = P_i(A) r_0,$$

where r_0 is the starting residual and $P_i(A)$ is an i-th degree polynomial with $P_i(0) = 1$. In CGS a solution is constructed for which the residual r_i^{CGS} satisfies

$$r_i^{CGS} = P_i^2(A) r_0,$$

where P_i is the same polynomial as in BI-CG. Bi-CGSTAB finally, leads to residuals $r_i^{Bi-CGSTAB}$ that can be written as

$$r_i^{Bi-CGSTAB} = Q_i(A)P_i(A)r_0,$$

with P_i again as in BI-CG, and Q_i is chosen as a polynomial of the form

$$Q_i(A) = (I - \omega_1 A)(I - \omega_2 A)\cdots(I - \omega_i A).$$

The value of ω_j is determined in the j-th iteration step such that $r_j^{Bi-CGSTAB}$ is minimal as a function of ω_j. Gutknecht [6] has recently proposed a variant of Bi-CGSTAB in which the minimization constants ω_j are chosen pairwise so as to minimize over two-dimensional spaces rather than in only one direction at a time. This variant has been named BICGSTAB2.

The above described polynomial relations will be used in section 3.3 in order to understand some effects in the convergence behaviour of the BI-CG family.

2 Current conservation in semiconductor modelling

In the package Curry [9] the semiconductor equations can be solved in a coupled way. The coupled equations are highly non-linear and they are solved using Newton's method. It is well known that Newton's method converges quadratically when close enough to the solution.
In each iteration of the Newton method a linear system needs to be solved. The convergence criteria for the linear solver should be so that the errors that are made in solving the linear system do not disturb the (quadratic) convergence behaviour of Newton. On the other hand they should not be so strict that the linear systems are solved more accurately than necessary for Newton. Therefore, in Curry the convergence criteria for the linear and the non-linear solver are coupled as described in [11]:

$$\|\mathrm{dx}_i\|_\infty \leq C_i\varepsilon_r\|\mathrm{x}_i\|_\infty + C_N\varepsilon_N\|\mathrm{y}_k\|_\infty + \varepsilon_a$$

where dx_i is the increment to x_i in the i-th iteration, x_i is the approximation of the solution in the i-th iteration, y_k is the solution of the Newton process in the previous Newton iteration, C_i is a constant which becomes smaller when Newton starts converging quadratically. C_N is a constant smaller than 1, which has been added to avoid identical convergence criteria for the linear process and the Newton process, which could cause a non-convergent Newton process.
The parameters ε_r, ε_N and ε_a are the tolerances. They are given a default value, but they can be changed by the user.

The determining factor in semiconductor modelling for the choice of the convergence criteria is the current conservation. With default accuracy we want to have current conservation for currents that are of order 10^{-8}.

In comparing the Bi-CGSTAB method with the CGS method it was found that the parameter ε_r should be much smaller for Bi-CGSTAB than for CGS if we want current conservation for currents of order 10^{-8}.
This is not as strange as it might seem. In semiconductor modelling, as in many other applications, CGS exhibits sometimes a very irregular convergence behaviour. In case of a rather

uneven distribution of the errorcomponents with respect to an eigenvector basis, squaring the BI-CG polynomial, as is done in CGS, leads to an immediate increase of the residual. For that reason we require the residual to be small during a successive, small number of iteration steps. As a result of this the actual error is often noticeably less than the required one.
For Bi-CGSTAB such a procedure is not necessary since it shows a much less irregular convergence behaviour, but now a smaller ε_r is needed to ensure a similar current conservation as in CGS.
Hence, in the examples that are shown further on in this paper, the parameters in the convergence criteria are not identical for both methods, but they are such that the current conservation was satisfied to about the same degree.

3 Choice of parameters in Bi-CGSTAB

In [17] a number of options are given for the algorithm. The options are here examined on their merit in semiconductor problems.

3.1 Determination of ω

In the algorithm the expression for ω contains $K_L^{-1}t$. Our preconditioning is based on an incomplete decomposition of the matrix A. Results for CGS have shown us that the preconditioning can best be done from both sides [4]. Hence in our case K_L would be the lower triangular factor and this means one extra forward substitution in each iteration step for computing ω. This is not a very attractive prospect. Furthermore, an extra auxiliary vector is necessary to store the vector $K_L^{-1}s$ (which is obtained as an intermediate result in the computation of $\hat{s}$).
The alternative presented in [17] is to replace the expression simply by $\omega_i = (t,s)/(t,t)$. In our examples this worked fine. This approach requires sometimes a few more iterations, but since the time per iteration is less, the total time decreases. In [17] it is shown that this choice effectively comes down to postconditioning for another choice of $\bar{r}$. This may well explain the differences in convergence behaviour, see section 3.3.
Note that with the above choice for ω we minimize the current residual for the original system, which might be the most desirable thing to do anyway.

3.2 The parameter ρ

The schemes for CGS and Bi-CGSTAB are esentially based upon the reconstruction of the BI-CG iteration coefficients α_j and β_j from certain innerproducts (these parameters define the polynomial P_i, see section 1). It is obvious that the precision with which these parameters can be determined depends quite critically upon the precision in the parameter ρ_j. This value is computed with the vector r_j, which has ben updated in the previous iteration step as $r_j = s - \omega_j t$. In that expression s represents the residual after the BI-CG part of the iteration procedure and $\omega_j t$ is the minimum residual correction in the j-th iteration step. It is clear that when Bi-CGSTAB is very successful, i.e., when $\|r_j\| \ll s$, then r_j may be expected to be less significant than either s or t. This suggests to compute ρ_j more accurately as

$$\rho_j = (\bar{r}, r_j) = (\bar{r}, s - \omega_j t) = -\omega_j(\bar{r}, t).$$

This has been suggested in [17] and it is stated there that the improved formula migth lead to a reduction in iteration steps. However, this is not in agreement with what others have observed in more realistic situations, in particular we have seen degrading effects in semiconductor modelling problems.
As might be expected, the values of the α_j and β_j that one would have obtained by carrying out the BI-CG scheme are now better recovered by the changed Bi-CGSTAB scheme in comparison with the standard scheme (as in section 2). But, surprisingly this does not necessarily lead to an improved convergence behaviour. Even on the contrary, it sometimes helps to compute ρ_j as $\rho_j = (\bar{r}, r_j)$ in reduced precision (e.g., in single precision, while all other computation is done in double precision). This is nicely illustrated by figure 3, in which we see the iteration results for the three different cases:

- $\rho_j = (\bar{r}, r_j)$ in full precision
- $\rho_j = (\bar{r}, r_j)$ in single precision
- $\rho_j = -\omega_j(\bar{r}, t)$ in full precision

We have seen many problems where the difference with the standard algorithm was minimal (see figure 2), but in the problems where a different convergence behaviour occurred, Bi-CGSTAB with the "improved" ρ_j computation required more iterations. At present we have the impression that the changes in the convergence behaviour are correlated with a loss of biorthogonality in the underlying BI-CG scheme, but we do not know how that could explain the observed differences.

3.3 The choice of $\bar{r}$

In our experiments we have seen that the choice of $\bar{r}$ may have a dramatic influence on all the algorithms in the BI-CG family: BI-CG, CGS and Bi-CGSTAB. This is well-known for BI-CG, since one has to avoid that the innerproducts that occur in the nominators for the iteration coefficients in that algorithm are not equal to zero (the so-called serious breakdown conditions). Because of the relation of CGS and Bi-CGSTAB with BI-CG we encounter, at least in exact arithmetic, the same problems at exactly the same iterationsteps as in BI-CG.
In practice one tries to avoid such situations by taking $\bar{r} = r_0$, so that at least no breakdown will occur in the very first step. For symmetric matrices one can prove that this choice does not lead to serious breakdown situations. Though in practice for more general situations this choice does not lead to breakdown problems, it may be responsible for the notorious wild convergence behaviour of CGS and even for some local fluctuations in Bi-CGSTAB.

In order to understand the effects that may take place we consider the case where A is symmetric positive definite, and we take first $\bar{r} = r_0$. BI-CG now delivers the same results as the Conjugate Gradient algorithm. Let us denote the eigenvalues and normalized eigenvectors of A by λ_j and z_j, and write $r_0 = \sum \gamma_j z_j$.
Then the residual r_i^{BI-CG} in BI-CG can be written as (cf. section 1)

$$r_i^{BI-CG} = \sum_j P_i(\lambda_j)\gamma_j z_j$$

and BI-CG (or CG) leads now to the polynomial for which $(r_i^{BI-CG}, A^{-1}r_i^{BI-CG})$ is minimal, i.e.,

$$(r_i^{BI-CG}, A^{-1}r_i^{BI-CG}) = \sum_j P_i^2(\lambda_j)\frac{\gamma_j^2}{\lambda_j}$$

is minimal. Furthermore,

$$\|r_i^{BI-CG}\|_2^2 = \sum_j P_i^2(\lambda_j)\gamma_j^2$$

and

$$\|r_i^{CGS}\|_2^2 = \sum_j P_i^4(\lambda_j)\gamma_j^2.$$

When a certain γ_k is small, i.e., the starting residual is deficient in the k-th eigenvector, then $P_i(\lambda_k)$ may be rather large (as long as $P_i^2(\lambda_k)\gamma_k^2/\lambda_k$ is small this does not contribute much to the norm to be minimized), and it may be even so large that the term $P_i^4(\lambda_k)\gamma_k^2$ is large compared with $\|r_i^{BI-CG}\|_2^2$. This helps to explain why $\|r_i^{CGS}\|_2$ may have large peaks even in cases where the $\|r_i^{BI-CG}\|_2^2$ converge quite smoothly.

Now it is easy to prove that with $\bar{r} = \sum \delta_j z_j$ a polynomial P_i is constructed such that

$$\sum_j P_i^2(\lambda_j)\frac{\gamma_j\delta_j}{\lambda_j}$$

is minimal, provided that $\gamma_j\delta_j > 0$ for all $\gamma_j \neq 0$.
With the choice $\delta_j = \frac{1}{\gamma_j}$, when $\gamma_j \neq 0$, P_i minimizes the expression $\sum P_i^2(\lambda_j)/\lambda_j$ and now it cannot happen that for some k the value of $P_i(\lambda_k)$ is large compared to the other values. Hence we may expect in that situation a rather smooth convergence behaviour for CGS. This is exactly what we see in experiments with this rather special choice for $\bar{r}$.

Unfortunately, in practice we never know the spectral distribution of r_0 and therefore this approach is not very practical. But these experiments tell us that, especially in situations where the starting residual is rather deficient in some eigenvector directions (i.e., the starting vector x_0 is already rich in some eigenvector directions, which may happen in the final stages of the Newton iteration), then it may not be a very good idea to take $\bar{r} = r_0$. Indeed, in the final stages of the newton iteration we have often seen that the inner iteration process, especially when carried out with CGS, takes unexpectedly many iterations and converges quite wild. To a lesser extent this has also been observed for Bi-CGSTAB.

Sometimes it helps to "enrich" $\bar{r}$ with the deficient directions by choosing $\bar{r}$ as

$$\bar{r} = r_0 + \frac{\|r_0\|}{\|y\|}y \quad ,$$

for some suitable vector $y \neq 0$.
For instance, when solving the nonlinear system $F(y) = 0$ with Newton's method:

$$y_i = y_{i-1} - \left(\frac{\partial F(y)}{\partial y}\right)^{-1}_{y=y_{i-1}} F(y_{i-1}) = y_{i-1} + \delta_i,$$

then the Newton correction δ_i is determined by solving the linear system

$$\left(\frac{\partial F(y)}{\partial y}\right)_{y=y_{i-1}} \delta_i = F(y_{i-1}).$$

In this case we suggest to take

$$\bar{r} = r_0 + \frac{\|r_0\|}{\|y_{i-1}\|} y_{i-1}$$

for any member of the BI-CG family. Our limited experience with this choice, when solving the inner iterations in the Newton process for semiconductor problems is promising sofar.
Others suggest to take $\bar{r}$ as a random vector, which in view of the foregoing analysis also might help to alleviate the problem of deficient eigenvector directions.
Note that for both choices most likely the restriction $\delta_j \gamma_j$ will not be satisfied, but we do not know how serious that condition is with respect to the convergence behaviour of the BI-CG type methods.

4 Vector and Parallel aspects

Major part of the iteration schemes for CGS as well as Bi-CGSTAB are trivially vectorizable and parallelizable. The only part that really needs attention is the preconditioner. Sofar we have quite a lot of experience with incomplete choleski preconditioning [7] for the discretized and linearized semiconductor equations in each gummel or newton step. These preconditioners can be vectorized on regular grids as is shown in [15], without any change in the convergence behaviour nor the computational complexity.
By constructing so-called nested twisted incomplete decompositions ([14], [15], [3]) we can introduce a modest amount of parallelism in the preconditioner (e.g., 8 parallel parts for regular 3D grids), while retaining most of the vectorizing properties.

For semiconductor device modelling we have build up some experience with these twisted incomplete factorizations in 2D models (4 parallel parts obtained by numbering the unknowns from 4 corners on inwardly in the domain). In our experiments this has also reduced the total computational work, sothat this approach has proven to be whorthwhile even on nonparallel computers.
Experiments carried out on a CRAY X-MP/2 and a Convex C240 have led to reductions in wall clock time (on dedicated systems) by factors of about 2 for the 2-processor CRAY and 3.3 for 4-processor Convex , without significant increases in the total CPU-times [16]. Also on loaded systems this approach is reported to have advantages.
Further increases in the amount of parallelism are suggested in [3]. The idea is to combine the nested twisted factorization with incomplete decompositions obtained over slightly overlapping diagonal blocks in the matrix. The latter technique is proposed in [2].

References

[1] G. Brussino and V. Sonnad, *A comparison of direct and preconditioned iterative techniques for sparse unsymmetric systems of linear equations*, Int. J. for Num. Methods in Eng., 28 (1989), pp. 801–815.

[2] G. R. di Brozolo and Y. Robert, *Parallel conjugate gradient-like algorithms for solving sparse non-symmetric systems on a vector multiprocessor*, Parallel Computing, 11 (1989), pp. 223–239.

[3] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, PA, 1991.

[4] M. Driessen, *Some practical problems in the numerical solution of elliptic partial differential equations*, master's thesis, Catholic University of Nijmegen, 1986.

[5] R. Fletcher, *Conjugate gradient methods for indefinite systems*, vol. 506 of Lecture Notes Math., Springer-Verlag, Berlin, 1976, pp. 73–89.

[6] M. H. Gutknecht, *Variants of bicgstab for matrices with complex spectrum*, preliminary report.

[7] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix*, Math.of Comp., 31 (1977), pp. 148–162.

[8] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, Tech. Report 90-2, MIT, Cambridge, MA, 1990.

[9] S. J. Polak, C. den Heijer, W. H. A. Schilders, and P. Markowich, *Semiconductor device modelling from the numerical point of view*, Int. J. for Num. Methods in Eng., 24 (1987), pp. 763–838.

[10] C. Pommerell and W. Fichtner, *Pils: An iterative linear solver package for ill-conditioned systems*, Tech. Report 91/5, ETH Zürich, 1991.

[11] J. Rutten, *Preconditioned iterative methods for solving linear systems with a non-symmetric coefficient matrix*, master's thesis, Catholic University of Nijmegen, 1987.

[12] Y. Saad and M. H. Schultz, *Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J.Sci.Statist.Comput., 7 (1986), pp. 856–869.

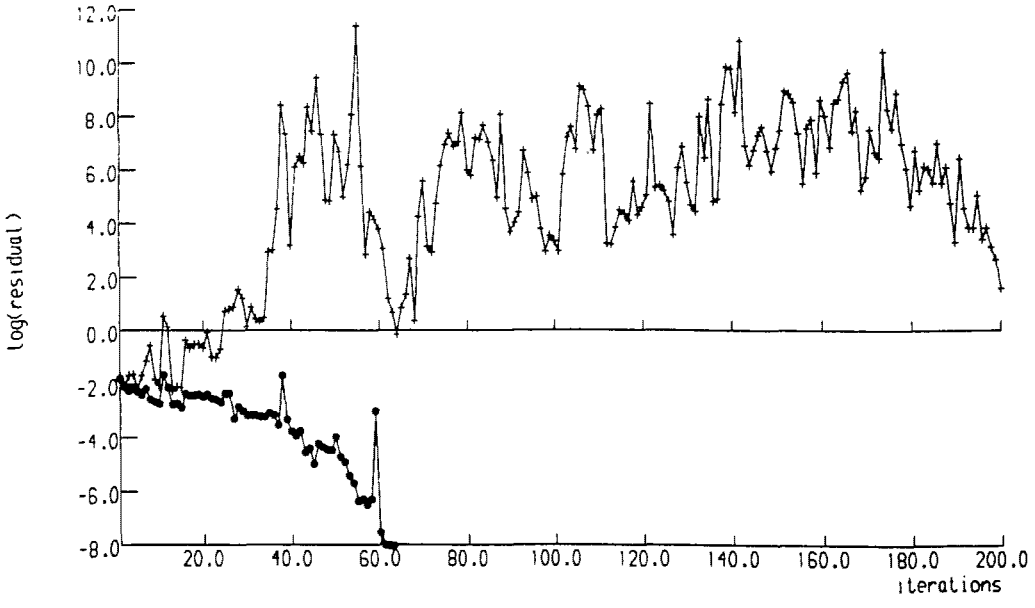
[13] P. Sonneveld, *Cgs: a fast lanczos-type solver for nonsymmetric linear systems*, SIAM J.Sci.Statist.Comput., 10 (1989), pp. 36–52.

[14] H. A. van der Vorst, *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45–54.

[15] ——, *High performance preconditioning*, SIAM J.Sci.Statist.Comput., 10 (1989), pp. 1174–1989.

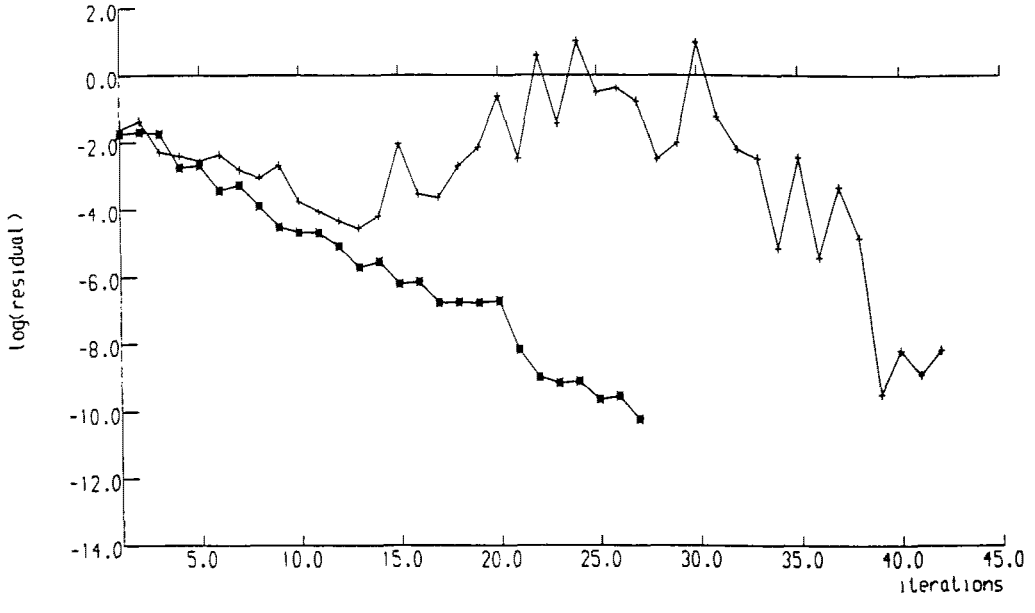
[16] ——, *Experiences with parallel vector computers for sparse linear systems*, Supercomputer, 37 (1990).

[17] ——, *Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of non-symmetric linear systems*, SIAM J. Sci. Statist. Comput., (1992). to appear.



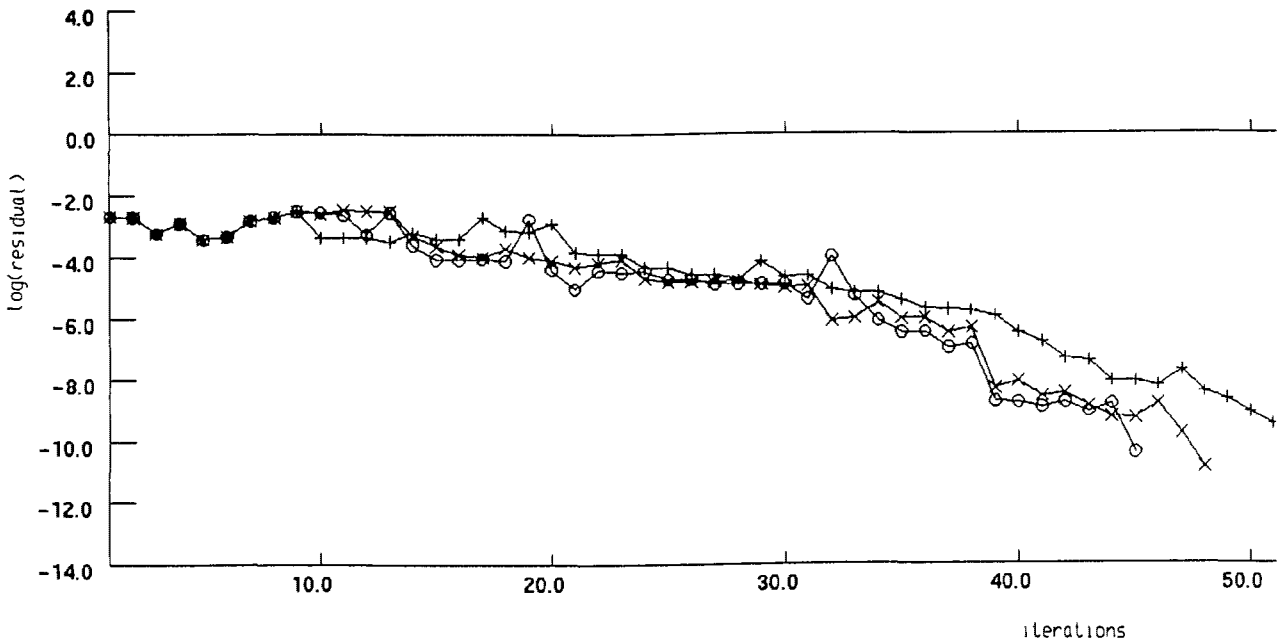
+ : CGS; • : Bi-CGSTAB

Figure 1: Residuals for an example where CGS didn't converge



× : Bi-CGSTAB; • : Bi-CGSTAB ($\rho_j = -\omega_j(\bar{r}, t)$); + : CGS

Figure 2: Residuals for an example where CGS converged



$\times$: standard ρ_j; $\circ$: half precision ρ_j; $+$: $\rho_j = -\omega_j(\bar{r}, t)$

Figure 3: Residuals for Bi-CGSTAB with different ρ_j