RAL-92-061

Science and Engineering Research Council
Rutherford Appleton Laboratory
Chilton DIDCOT Oxon OX11 0QX

RAL-92-061

An Efficient Implementation of Bi-CGSTAB Applied to Three-Dimensional Semiconductor Device Problems

C J Fitzsimons

September 1992

An Efficient Implementation of Bi-CGSTAB Applied to Three-Dimensional Semiconductor Device Problems

C.J. Fitzsimons

September 1992

Mathematical Software Group
Computational Modelling Division
Rutherford Appleton Laboratory
Chilton, Didcot
Oxfordshire OX11 0QX

Contents

1 INTRODUCTION

The three-dimensional analysis of semiconductor devices, which is increasingly important for industrial design, requires the solution of the following system of coupled nonlinear partial differential equations, subject to suitable boundary and initial conditions.

$$\nabla \cdot (\varepsilon \nabla \psi) = n - p - N \quad (1)$$

$$\frac{\partial n}{\partial t} = -\nabla \cdot J_n - R \quad (2)$$

$$\frac{\partial p}{\partial t} = \nabla \cdot J_p - R \quad (3)$$

$$J_n = -\mu_n (\nabla n - n \nabla \psi) \quad (4)$$

$$J_p = \mu_p (\nabla p + p \nabla \psi) \quad (5)$$

where ε denotes the permittivity, ψ denotes the electrostatic potential, n and p the electron and hole concentrations respectively, J_n and J_p the electron and hole currents respectively, R the net recombination rate, N the impurity concentration, and μ_n and μ_p the electron and hole mobility rates respectively. The form of N, which is a function of position, determines the type of semiconductor device under consideration. For a detailed discussion of these equations and their solution we refer the reader to [18]. The numerical solution of these equations is very time-consuming due to their nonlinearity and the ill-conditioned nonsymmetric linear systems which arise during the solution process. A further problem is due to the extreme range which some of the dependent variables exhibit, e.g. $10^{-4} < n < 10^{20}$. The equations also provide a good testbed for testing linear solutions algorithms because different types of semiconductor devices and sets of operating conditions give rise to distinct solution difficulties.

There are two nonlinear algorithms used in semiconductor device modelling and they give rise to linear systems of different types. In the first, Gummel's method, each of the partial differential equations (1–3) is solved in turn, using the most up to date values of the other dependent variables until a self-consistent solution of the system is obtained. Nonsymmetric linear systems arise in the upwind finite element or finite volume discretisation of the electron and hole continuity equations (2–3). In the second, Newton's method, the fully coupled system of nonlinear Poisson (1) and electron and hole continuity equations (2–3) is solved simultaneously. The resultant linear system is typically very poorly conditioned, nonsymmetric and not diagonally-dominant [17].

In both approaches the majority of the computational effort is invested in solving the linear systems which arise. Therefore, over the years considerable research has gone into developing efficient iterative solution techniques for such systems. We refer to [5, 8, 18] for surveys of the different methods used in practice.

The Conjugate Gradient Squared (CGS) method [19], which is a variant of the Bi-Conjugate Gradient (Bi-CG) method [10] has tended to be used in the solution of three-dimensional problems, despite its irregular convergence pattern, because when it works — which is almost all the time — it works quite well. Recently, Van der Vorst [21] has presented a new variant of Bi-CG, called Bi-CGSTAB because of its stable behaviour, which combines the efficiency of CGS with the more regular convergence pattern of Bi-CG.

In this report we describe the implementation of Bi-CGSTAB in EVEREST [1, 12, 13] and compare CGS and Bi-CGSTAB for a variety of three-dimensional device modelling problems. In Section 2 we outline how iterative methods work and report on different methods used previously to solve linear systems in semiconductor device modelling.

In Section 4 we explain the motivation behind methods of conjugate gradient (CG) type while in Section 5 we discuss an efficient implementation of Bi-CGSTAB. In [7] Eisenstat has shown how it is possible to implement the CG method efficiently and replace a matrix-vector product by $2N$ multiply-adds. In this report we show how the same idea can be implemented to improve the efficiency of Bi-CGSTAB.
We look at the effect of the choice of preconditioning matrix on the method in Section 6 and in Section 7 we present some numerical results. We give separate results for the methods applied to linear systems arising when a Gummel algorithm is used to solve the nonlinear problem and when a Newton algorithm is used since the improvement of Bi-CGSTAB over CGS depends on which overall solution algorithm is used.

2 Some Background

We wish to solve a linear system of the form

$$A\mathbf{x} = \mathbf{b} \tag{6}$$

using an iterative method, where A is an $N \times N$ sparse matrix and $\mathbf{x}$ and $\mathbf{b}$ are vectors of order N. Most iterative methods are based on the following idea: let K be an arbitrary, non-singular, preferably easily invertible matrix, and let $A = K - R$ represent a splitting of A. Then the basic iteration is

$$K\mathbf{x}_i = R\mathbf{x}_{i-1} + \mathbf{b} \tag{7}$$

where $\mathbf{x}_0$ is a user-specified starting vector, and the iteration proceeds until convergence to the solution to (6) is achieved. Many choices for K are possible. For example, if K is chosen to be a diagonal matrix equal to the main diagonal of A, the Jacobi method results.
We define the residual at iteration i to be

$$\mathbf{r}_i = A\mathbf{x}_i - \mathbf{b} \tag{8}$$

Using (8) we can rewrite (7) in the form

$$\mathbf{x}_i = \mathbf{x}_{i-1} + K^{-1}\mathbf{r}_i = \mathbf{x}_0 + \alpha_0 K^{-1}\mathbf{r}_0 + \ldots + \alpha_{i-1}(K^{-1}A)^{i-1}K^{-1}\mathbf{r}_0 \tag{9}$$

It is clear that $\mathbf{x}_i$ is $\mathbf{x}_0$ plus a vector from the i-dimensional Krylov subspace, denoted $K_i(K^{-1}A; K^{-1}\mathbf{r}_0)$, which is spanned by the vectors $\{K^{-1}\mathbf{r}_0, \ldots, (K^{-1}A)^{i-1}K^{-1}\mathbf{r}_0\}$. From (9) it is clear that the more closely K resembles A, the fewer iterations are required to attain convergence.
In CG-type methods we construct a suitable basis for $K_i(A; \mathbf{r}_0)$ and solve (6) projected onto this subspace, i.e. we choose $\mathbf{x}_i$ to minimise an appropriate functional over the subspace. Obtaining the solution of (6) when A is symmetric and positive definite corresponds to minimising the functional [1]

$$h^2 = [\mathbf{r}, A^{-1}\mathbf{r}] \tag{10}$$

$$= \mathbf{r}^T A^{-1}\mathbf{r} \tag{11}$$

where the bilinear form is $[\mathbf{u}, \mathbf{v}] = \mathbf{u}^T\mathbf{v}$. CG has the property that it converges in at most N iterations and the minimisation of h^2 is monotone, i.e.

$$\mathbf{r}_{i+1}^T A^{-1}\mathbf{r}_{i+1} = h_{i+1}^2 \leq h_i^2 = \mathbf{r}_i^T A^{-1}\mathbf{r}_i \tag{12}$$

[1] Since A is symmetric and positive definite, the minimum of h^2 is 0 and this only occurs when $\mathbf{r}$ is zero, i.e. $A\mathbf{x} - \mathbf{b} = \mathbf{0}$.

In CG this leads to an iteration of the form

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha \mathbf{p}_{i-1} \tag{13}$$

where α is a suitably chosen iteration parameter and $\mathbf{p}_{i-1}$ is called the search direction, chosen so that it is orthogonal to the previous search directions, i.e. $(\mathbf{p}_i, \mathbf{p}_j) = \mathbf{p}_i^T \mathbf{p}_j = 0$ for $i \neq j$.
The rate of convergence of methods of CG-type is proportional to the condition number of A, denoted $\kappa(A)$ which is defined as the ratio of largest and smallest eigenvalues of A. This rate of convergence can be improved by choosing a preconditioning matrix K with the properties that $\kappa(K^{-1}A) < \kappa(A)$, K is easily factored and the linear system $K\hat{\mathbf{p}} = \mathbf{p}_k$ is solved efficiently for vectors $\hat{\mathbf{p}}, \mathbf{p}_k$.

3 A Brief Survey of Solution Methods for the Device Equations

The CG method has been used to solve the nonsymmetric linear systems arising from the discretisation of the continuity equations. Wada and Dang [22, 23] introduced a modification to CG where the incomplete Cholesky preconditioner of A is replaced by an incomplete LDU decomposition and where the method is used in conjunction with the re-starting technique, i.e. after every k^{th} iteration of the CG method the present estimate $\mathbf{x}^*$ is taken as the initial estimate $\mathbf{x}_0$ and the iteration procedure is restarted. In their work Wada and Dang take $k = 10$ and usually obtain a solution after two restarts, i.e. within thirty iterations. Chang and Wagner [2] used the CG method to solve the normal equations

$$A^T A\mathbf{x} = A^T \mathbf{b} \tag{14}$$

CG may be applied directly to (14). However, the convergence rate of the CG is adversely affected because of the large increase in the condition number of $A^T A$. They indicated that the method of Wada and Dang fails if A is highly non-symmetric. (Their counter-example did not arise from device modelling.) Toyabe [20] also reported that the method of Wada and Dang fails to converge when the ratio of non-zero non-symmetric elements is in excess of 50%.
Den Heijer [3] applied ORTHOMIN, Bi-conjugate gradients (Bi-CG) and CGS to the two-dimensional device problem. He also developed two preconditioners for use with CGS when solving a coupled system when the grid used is a distortion of a rectangular grid. Edwards [6] reported on a preconditioner for the case when the system matrix arises from the discretisation of the coupled system on a triangular grid and found CGS to be more robust and efficient than ORTHOMIN(10), i.e. ORTHOMIN where the ten most recent search directions are stored.
Fitzsimons [8, 9] introduced a diagonal matrix which can be used to premultiply both sides of (6) when A arises from the appropriate discretisation of (2–3). The resultant system is solved using CG.
Recently, Van der Vorst [5] has presented a new variant of Bi-CG, called Bi-CGSTAB because of its stable behaviour, which combines the efficiency of CGS with the more regular convergence pattern of Bi-CG. In [21] results for the method applied to two-dimensional device modelling problems are presented. Heinreichsberger [15] compared CG, symmetrised CG, the generalised minimum residual (GMRES) and CGS methods for the three-dimensional device equations and studied the effect of the choice of preconditioner on the methods. In [16] the implementation of CG-type methods on distributed memory parallel processors is discussed.

4 Conjugate Gradient-Type Methods

For a detailed description of CG-type methods, e.g. Bi-CG, CGS, and Bi-CGSTAB, we refer to [4, 5, 21]. The purpose of this section is to outline Bi-CG and show its relationship to CGS and Bi-CGSTAB.
In Bi-CG we solve, instead of (6), an augmented system

$$C\mathbf{w} = \begin{pmatrix} A & 0 \\ 0 & A^T \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{b} \end{pmatrix} = \mathbf{d} \tag{15}$$

in order to compute $\mathbf{x}$. Using the definition (8) for the residual and

$$\tilde{\mathbf{r}}_i = A^T\tilde{\mathbf{x}}_i - \mathbf{b} \tag{16}$$

for the bi-residual, $\mathbf{R} = (\mathbf{r}, \tilde{\mathbf{r}})^T$ and the bilinear form

$$[\mathbf{u}, \mathbf{v}] = \mathbf{u}^T Q \mathbf{v} \quad \text{where } Q = \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \tag{17}$$

Bi-CG corresponds to the minimisation of the functional

$$h^2 = [\mathbf{R}, C^{-1}\mathbf{R}] \tag{18}$$

$$= \mathbf{r}^T A^{-T}\tilde{\mathbf{r}} + \tilde{\mathbf{r}}^T A^{-1}\mathbf{r} \tag{19}$$

However, the minimisation of h^2 is not monotone. Its name derives from the fact that $\mathbf{r}_i$ and $\tilde{\mathbf{r}}_i$ satisfy the relation

$$(\mathbf{r}_i, \tilde{\mathbf{r}}_j) = 0 \qquad \text{for} \quad i \neq j \tag{20}$$

and the search directions $\mathbf{p}_i, \tilde{\mathbf{p}}_i$ satisfy the bi-conjugacy condition

$$(\tilde{\mathbf{p}}_i, A\mathbf{p}_j) = 0 \qquad \text{for} i \neq j \tag{21}$$

The residuals can be written as polynomials of A and A^T, i.e. $\mathbf{r}_i = P_i(A)\mathbf{r}_0$ and $\tilde{\mathbf{r}}_i = P_i(A^T)\mathbf{r}_0$. Because of the bi-orthogonality property (20) we have

$$(\mathbf{r}_j, \tilde{\mathbf{r}}_i) = (P_j(A)\mathbf{r}_0, P_i(A^T)\tilde{\mathbf{r}}_0) = (P_i(A)P_j(A)\mathbf{r}_0, \tilde{\mathbf{r}}_0) = 0 \tag{22}$$

for $i < j$. This indicates that $P_j(A)\mathbf{r}_0$ is perpendicular to the subspace $K_i(A^T; \tilde{\mathbf{r}}_0)$. We are free to choose any linearly independent set of vectors from the subspace as its basis. In Bi-CG we choose

$$\tilde{\mathbf{r}}_i = P_i(A^T)\tilde{\mathbf{r}}_0 \tag{23}$$

In CGS we avoid computing (23) directly. Instead we make use of (22) and compute

$$\mathbf{r}_i = P_i^2(A)\mathbf{r}_0 \tag{24}$$

In most cases this leads to a method which converges more quickly than Bi-CG. There are some examples [5] where this is not the case. Indeed, in some examples the value of $(\mathbf{r}_i, \mathbf{r}_i)$ oscillates wildly, leading to updates which swamp the true solution to (6).
In Bi-CGSTAB a different basis for $K_i(A^T; \tilde{\mathbf{r}}_0)$ is chosen and in this case the residual can be written in the form

$$\mathbf{r}_i = Q_i(A)P_i(A)\mathbf{r}_0 \tag{25}$$

$$Q_i(A) = (I - \omega_1 A)(I - \omega_2 A)\ldots(I - \omega_i A) \tag{26}$$

and ω_i is selected to minimise $\mathbf{r}_i$ for residuals written in this form. he full Bi-CGSTAB algorithm is

$\mathbf{x}_0$ is an initial guess; $\mathbf{r}_0 = b - A\mathbf{x}_0$;
$\tilde{\mathbf{r}}_0 = \mathbf{r}_0$;
$\rho_0 = \alpha_0 = \omega_0 = k = 1$;
$\mathbf{v}_0 = \mathbf{p}_0 = 0$;
While $\mathbf{x}_k$ is not accurate enough **do**
$\rho_k = (\tilde{\mathbf{r}}_0, \mathbf{r}_{k-1})$; $\beta = (\rho_k/\rho_{k-1})(\alpha_{k-1}/\omega_{k-1})$;
$\mathbf{p}_k = \mathbf{r}_{k-1} + \beta(\mathbf{p}_{k-1} - \omega_{k-1}\mathbf{v}_{k-1})$;
Solve $\hat{\mathbf{p}}$ from $K\hat{\mathbf{p}} = \mathbf{p}_k$;
$\mathbf{v}_k = A\hat{\mathbf{p}}$;
$\alpha_k = \rho_k/(\tilde{\mathbf{r}}_0, \mathbf{v}_k)$;
$\mathbf{s} = \mathbf{r}_{k-1} - \alpha_k\mathbf{v}_k$;
Solve $\hat{\mathbf{s}}$ from $K\hat{\mathbf{s}} = \mathbf{s}$;
$\mathbf{t} = A\hat{\mathbf{s}}$;
$\omega_k = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$;
$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k\hat{\mathbf{p}} + \omega_k\hat{\mathbf{s}}$;
$\mathbf{r}_k = \mathbf{s} - \omega_k\mathbf{t}$;
$k = k + 1$;
end

5 Eisenstat Bi-CGSTAB

Eisenstat [7] noticed that when using a preconditioned CG to solve (6), when A is symmetric positive definite, it is possible to reduce the computational effort per iteration significantly when using a preconditioning matrix of the form

$$K = (L + \tilde{D})\tilde{D}^{-1}(\tilde{D} + U) \tag{27}$$

where

$$A = L + D + U \tag{28}$$

The same reduction in computational effort can be achieved with Bi-CGSTAB. If we restate (6) in the form

$$[(L + \tilde{D})^{-1}A(\tilde{D} + U)^{-1}][(\tilde{D} + U)\mathbf{x}] = (L + \tilde{D})^{-1}b \tag{29}$$

$$\text{or} \qquad \hat{A}\hat{\mathbf{x}} = \hat{b} \tag{30}$$

applying Bi-CGSTAB to (6) with (27) as preconditioner is equivalent to applying Bi-CGSTAB to (30) with $K = \tilde{D}^{-1}$ as preconditioner.
In the original form of the algorithm we compute

$$\text{Solve } \hat{\mathbf{p}} \text{ from } K\hat{\mathbf{p}} = \mathbf{p}_i \tag{31}$$

$$\mathbf{v}_i = A\mathbf{y} \tag{32}$$

where K is given by (27), which requires approximately $2\text{NZ}(A)$ multiply-adds, where $\text{NZ}(A)$ denotes the number of non-zero entries in A. In our Eisenstat implementation we replace (31–32) by

$$\hat{\mathbf{y}} = \tilde{D}\mathbf{p}_i \tag{33}$$

$$\mathbf{y} = (\tilde{D} + U)^{-1}\hat{\mathbf{y}} \tag{34}$$

$$\mathbf{v}_i = \mathbf{y} + (L + \tilde{D})^{-1}(\hat{\mathbf{y}} - (D - 2\tilde{D})\mathbf{y}) \tag{35}$$

which requires approximately $\mathrm{NZ}(A) + 2N$ multiply adds.
Analogously,

$$\text{Solve } \hat{\mathbf{s}} \text{ from } K\hat{\mathbf{s}} = \mathbf{s} \tag{36}$$

$$\mathbf{t} = A\hat{\mathbf{s}} \tag{37}$$

is replaced by

$$\hat{\mathbf{z}} = \tilde{D}\mathbf{s} \tag{38}$$

$$\mathbf{z} = (\tilde{D} + U)^{-1}\hat{\mathbf{z}} \tag{39}$$

$$\mathbf{t} = \mathbf{z} + (L + \tilde{D})^{-1}(\hat{\mathbf{z}} - (D - 2\tilde{D})\mathbf{z}) \tag{40}$$

A difficulty with solving (30) directly is that we obtain a solution of the specified accuracy to (30) rather than to (6) since

$$\hat{\mathbf{x}} = (\tilde{D} + U)\mathbf{x} \tag{41}$$

Therefore, we modify the Eisenstat form of Bi-CGSTAB so that we compute $\mathbf{x}$ directly. The full Eisenstat Bi-CGSTAB algorithm, where we compute the solution to (6), is:

$\mathbf{x}_0$ is an initial guess; $(L + \tilde{D})^{-1}(\mathbf{r}_0 = b - A\mathbf{x}_0)$;
$\hat{\mathbf{r}}_0 = \tilde{D}\mathbf{r}_0$;
$\rho_0 = \alpha_0 = \omega_0 = i = 1$;
$\mathbf{v}_0 = \mathbf{p}_0 = 0$;
While $\mathbf{x}_{i-1}$ is not accurate enough **do**
  $\rho_i = (\hat{\mathbf{r}}_0, \mathbf{r}_{i-1}); \beta = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$;
  $\mathbf{p}_i = \mathbf{r}_{i-1} + \beta(\mathbf{p}_{i-1} - \omega_{i-1}\mathbf{v}_{i-1})$;
  $\hat{\mathbf{y}} = \tilde{D}\mathbf{p}_i$;
  $\mathbf{y} = (\tilde{D} + U)^{-1}\hat{\mathbf{y}}$;
  $\mathbf{v}_i = \mathbf{y} + (L + \tilde{D})^{-1}(\hat{\mathbf{y}} - (D - 2\tilde{D})\mathbf{y})$;
  $\alpha_i = \rho_i/(\hat{\mathbf{r}}_0, \mathbf{v}_i)$;
  $\mathbf{s} = \mathbf{r}_{i-1} - \alpha_i\mathbf{v}_i$;
  $\hat{\mathbf{z}} = \tilde{D}\mathbf{s}$;
  $\mathbf{z} = (\tilde{D} + U)^{-1}\hat{\mathbf{z}}$;
  $\mathbf{t} = \mathbf{z} + (L + \tilde{D})^{-1}(\hat{\mathbf{z}} - (D - 2\tilde{D})\mathbf{z})$;
  $\omega_i = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$;
  $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i\mathbf{y} + \omega_i\mathbf{z}$;
  $\mathbf{r}_i = \mathbf{s} - \omega_i\mathbf{t}$;
  $i = i + 1$;
end

6 Choice of Preconditioner

EVEREST offers a choice of two preconditioners for use with the already implemented CGS algorithm. The first is an incomplete LDU decomposition with the same sparsity pattern as A in which the off-diagonal elements are modified. We call this K_1. The second is of the form (27) in which the off-diagonal elements of A are not modified. We will call this K_2.

Bi-CGSTAB experienced difficulty in converging when using K_1 as preconditioner for problems which were ill-conditioned. In order to investigate this problem we took a test program from the BECAUSE Benchmark Set (BBS) [11] which replicates the performance of EVEREST on a simple test example. One of the features of the linear system generated in this test is that in one portion of the matrix the main diagonal has entries $s = 2.36159 \times 10^{-7}$ while some of the off-diagonal entries on these rows are $O(1)$. We considered $s = 2.36159 \times 10^{-k}$ where $k = 0, 1, \ldots, 7$. We then computed $\mathbf{y}_i = K_i^{-1}A\mathbf{1}$ and evaluated $\max \mid \mathbf{y}_i \mid$. The results of this and the number of iterations required to solve the linear system of order 343 using Bi-CGSTAB is shown in Table 1. In this table, the maximum value of $\mathbf{y}$ provides

k	$\max \mid \mathbf{y}_1 \mid$	Number of Iterations (K_1)	$\max \mid \mathbf{y}_2 \mid$	Number of Iteration (K_2)
0	9.978(-1)	11	9.903(-1)	9
1	6.285(0)	13	4.230(0)	10
2	6.973(+2)	15	4.230(+1)	10
3	7.122(+4)	20	4.230(+2)	12
4	7.139(+6)	200	4.231(+3)	14
5	7.156(+8)	* * *	4.238(+4)	15
6	7.182(+10)	* * *	4.305(+5)	16
7	7.194(+12)	* * *	5.096(+6)	17

Table 1: Summary of Results for Different Preconditioners Applied to the modified BECAUSE Benchmark Set Test Problem

one measure of how well each preconditioner approximates the original system matrix A. The number of iterations required to obtain a solution increases only moderately with problem difficulty when K_2 is used, while the * * * in the final three cases indicates that a solution was not obtained within 250 iterations when K_1 was used. Therefore we have decided to always use K_2 as the preconditioner for Bi-CGSTAB in EVEREST.

7 Numerical Results

We have implemented both the Eisenstat and regular Bi-CGSTAB algorithms in EVEREST. The comparisons obtained so far with the existing Eisenstat and regular CGS solvers used in the package are very encouraging. The new solvers use the same preconditioning and vector operation routines as the existing CGS solvers where possible. In the tables given below we present the amount of CPU time spent in solving nonsymmetric linear systems in the overall solution process. Two types of linear system arise: from the continuity equations and from the fully coupled system of equations. The total number of linear iterations required to solve the systems of each type is given. Preconditioner K_1 is used by CGS, the others use K_2. All computations were performed on a Stardent GS2000.

We present results for four devices which are described in [1, 14]. The first device is an *npn*-transistor which we solve for two sets of boundary conditions. In the first case we solve a sequence of six nonlinear problems of increasing difficulty in which the solution of one problem provides the starting point for the solution of the next. In the second case we solve the final problem in the sequence starting from scratch, rather than the solution to the fifth problem. A grid of 12163 nodes and 16108 elements is used to discretise the problem. The linear systems arising from the continuity equations are of order 11833 and those arising from the fully coupled equations are of order 35499. In Case 2 we solve 12 continuity

equation systems and 4 coupled systems. A summary of the CPU time and the total number of CGS or Bi-CGSTAB iterations required to solve the linear systems is given in Table 2 and Table 3 for Cases 1 and 2 respectively.

Method	Time (Minutes) Gummel	Time (Minutes) Newton	Total Iterations Gummel	Total Iterations Newton
Eisenstat Bi-CGSTAB	42.31	35.14	1479	120
Bi-CGSTAB	50.93	39.33	1475	105
Eisenstat CGS	58.45	54.4	1809	218
CGS	59.60	73.29	1577	209

Table 2: Summary of results for npn-Transistor, Case 1

Method	Time (Minutes)	Speed-up over CGS	Total Iterations Gummel	Total Iterations Newton
Method	Time (Minutes) Gummel	Time (Minutes) Newton	Total Iterations Gummel	Total Iterations Newton
Eisenstat Bi-CGSTAB	18.58	18.15	656	75
Bi-CGSTAB	24.71	28.62	721	89
Eisenstat CGS	29.41	43.03	924	189
CGS	30.44	52.09	815	169

Table 3: Summary of results for npn-Transistor, Case 2

The second device is a MOS transistor for which we solve a sequence of nine problems. In this case the overall solution procedure does not require the solution of any coupled systems. Each system arising from the continuity equations is of order 21019. The results for this problem are summarised in Table 4.

Method	Time (Minutes) Gummel	Total Iterations Gummel
Eisenstat Bi-CGSTAB	85.89	1740
Bi-CGSTAB	109.02	1859
Eisenstat CGS	178.96	3242
CGS	184.95	2680

Table 4: Summary of Results for the MOS Transistor

The third device is a charge coupled device (CCD) for which we solve a sequence of eight problems. The relative difficulty of the problems is seen in the increased CPU time and increased iteration count for the CGS and Bi-CGSTAB solvers in this case. Each system arising during the Gummel method is of order 9449, while each system arising during the Newton method is of order 28347. The results for this problem are summarised in Table 5.

Method	Time (Minutes) Gummel	Time (Minutes) Newton	Total Iterations Gummel	Total Iterations Newton
Eisenstat Bi-CGSTAB	225.62	190.46	11046	1196
Bi-CGSTAB	272.93	268.38	10872	1207
Eisenstat CGS	281.55	356.11	11818	2234
CGS	317.52	361.95	11373	1741

Table 5: Summary of Results for the CCD

The final example is a simple two-dimensional diode which is being solved using an adaptive meshing technique. The times given are for the total amount of time spent on all the meshes during the adaptive procedure. The final mesh gives rises to systems of order 1682 for the Gummel method and of order 5046 for the Newton method. The results for this problem are summarised in Table 6.

Method	Time (Minutes) Gummel	Time (Minutes) Newton	Total Iterations Gummel	Total Iterations Newton
Eisenstat Bi-CGSTAB	2.78	17.83	756	596
Bi-CGSTAB	3.39	24.29	792	607
Eisenstat CGS	3.82	27.71	918	928
CGS	3.87	32.74	783	796

Table 6: Summary of Results for the p-n Diode Solved Adaptively

In all five tables we observe the improvement in solution time when Bi-CGSTAB is used rather than CGS. This is due to the smoother convergence pattern of Bi-CGSTAB. An example of this is shown in Figure 1 which gives the convergence patterns for both CGS and Bi-CGSTAB for a linear system which arose during Case 2 for the npn-transistor. A log plot of the same data is given in Figure 2.
The implemented convergence check is that two successive solution vectors, $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$, satisfy the error tolerance. This protects against the oscillatory behaviour of the residual in CGS. The dramatic reduction in solution time is due entirely to the reduction in the number of iterations, since each Bi-CGSTAB iteration is slightly more expensive than a CGS iteration: it requires two additional inner-products. The reduction is more pronounced for the systems arising from the coupled equations, which are typically more ill-conditioned. The Eisenstat implementation of Bi-CGSTAB described here further enhances the performance of the Bi-CGSTAB algorithm.

8 Summary

In this report we have described in detail an Eisenstat implementation of the Bi-CGSTAB algorithm and applied it to realistic problems arising in three-dimensional semiconductor device analysis. We have compared standard and Eisenstat implementations of the algorithm with and standard and Eisenstat implementations of the CGS algorithm.
The impact of the choice of preconditioner on the method has been investigated. We have also surveyed other work on iterative solution techniques applied to linear systems which arise in device modelling.

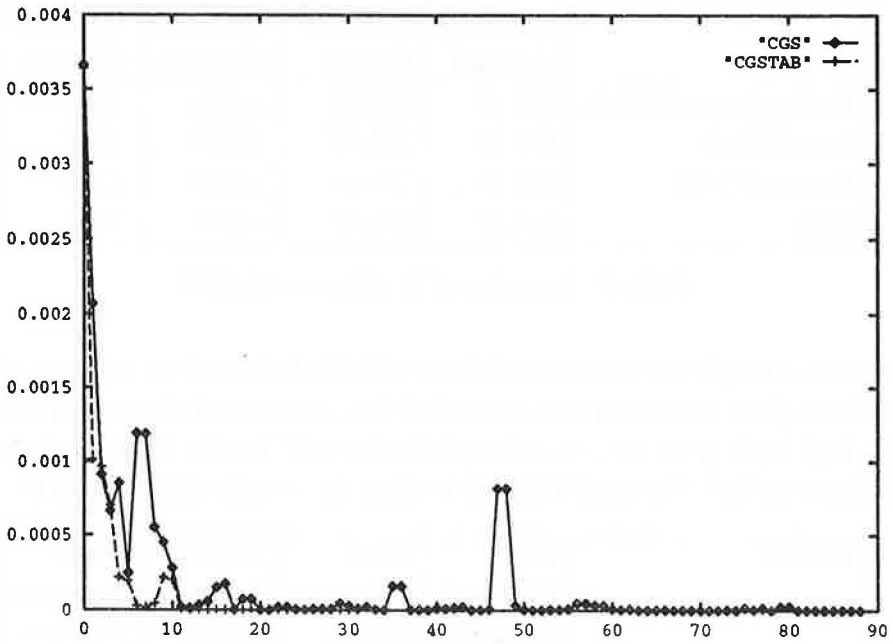

Figure 1: Typical Convergence Pattern for CGS and Bi-CGSTAB

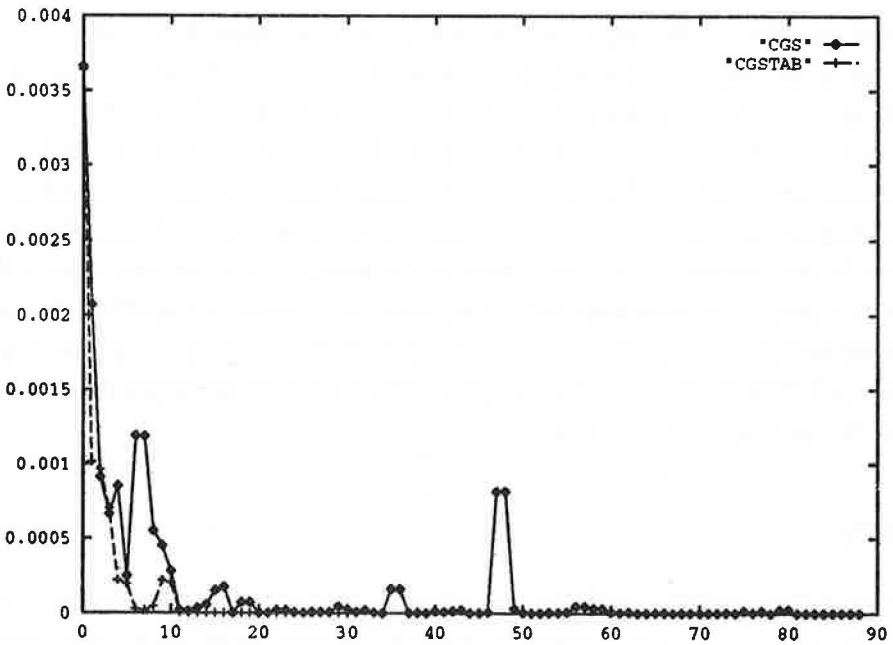

Figure 2: Typical Convergence Pattern for CGS and Bi-CGSTAB (Log Plot)

In the tests we have performed we have shown that Bi-CGSTAB provides useful efficiency gains over CGS for a wide range of problems. These can sometimes reach a factor of two.

Acknowledgements

I would like to thank Professor Henk van der Vorst for his patience and helpful discussions during this research. I would also like to thank my colleagues Dr Chris Greenough, Dr John Ashby and Dr Ron Fowler for their comments and suggestions.

References

[1] **J.V. Ashby, C.J. Fitzsimons, R.F. Fowler, C. Greenough, P.J. Mole, W.H.A. Schilders** "Three-Dimensional Adaptive Semiconductor Device Simulation" from *Simulation of Semiconductor Devices and Processes Vol. 4* (ed. W. Fichtner, D. Aemmer) Hartung-Gorre (1991)

[2] **F.Y. Chang, L.F. Wagner** "Generalised ICCG Method for Solution of Asymmetric, Sparse, Linear Systems of Discretized Semiconductor Device Equations" *Electron. Lett.* **18**, No. 15, 658–660 (1985)

[3] **C. Den Heijer** "Preconditioned Iterative Methods for Non-symmetric Linear Systems" from *Proceedings of the International Conference on Simulation of Semiconductor Devices and Processes* (eds. K. Board, D.R.J. Owen) Pineridge Press, Swansea (1984)

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

[5] **M. Driessen, H.A. Van der Vorst** "BI-CGSTAB in Semiconductor Modelling" from *Simulation of Semiconductor Devices and Processes Vol. 4* (ed. W. Fichtner, D. Aemmer) Hartung-Gorre (1991)

[6] **S.P. Edwards, A.M. Howland, P.J. Mole** "Initial Guess Strategy and Linear Algebra Techniques for a Coupled Two-Dimensional Semiconductor Equation Solver" from *Proceedings of the Fourth International Conference on the Numerical Analysis of Semiconductor Devices and Integrated Circuits* (ed. J.J.H. Miller) Boole Press, Dublin (1985)

[7] **S.C. Eisenstat** "Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods" *SIAM J. Sci. Stat. Comput.*

[8] **C.J. Fitzsimons** *Methods for the Numerical Solution of Singular Perturbation Problems in One and Two Dimensions* Ph.D. Thesis, University of Dublin, Trinity College (1986)

[9] **C.J. Fitzsimons** "A Novel Method for Solving the Continuity Equations" *COMPEL* **6** No. 2, 71-76 (1987)

[10] **R. Fletcher** "Conjugate Gradient Methods for Indefinite Systems" *Lecture Notes in Mathematics* **506** 773–89 Springer Verlag (1976)

[11] **R.F. Fowler, C. Greenough** "Benchmark Problems Proposed by RAL" *ESPRIT Project No. 5417, BECAUSE RAL 91.7* (1991)

[12] **C. Greenough, D. Gunasekera, C.J. Fitzsimons, P.A. Mawby, M.S. Towers** "Modelling Semiconductor Devices in Three Dimensions" from *Software Tools for Process, Device and Circuit Modelling* (Ed. W. Crans) Boole Press, Dublin (1989)

[13] **C. Greenough, C.J. Fitzsimons, R.F. Fowler** "Software for Modelling Semiconductor Devices in Three Dimensions" *RAL–91–042* Rutherford Appleton Laboratory Report (1991)

[14] **D. Gunasekera, C. Greenough** "Comprehensive Testing of the Solver Module" EVEREST ESPRIT Project 962E, Workpackage 5 Report (1988)

[15] **O. Heinreichsberger, S. Selberherr, M. Stiftinger, K.P. Traar** "Fast Iterative Solution of Carrier Continuity Equations for Three-Dimensional Device Simulation" *SIAM J. Sci. Stat. Comput.* **13** No. 1, 289–306, (1992)

[16] **C. Pommerell, M. Annaratone, W. Fichtner** "A Set of New Mapping and Colouring Heuristics for Distributed-Memory Parallel Processors" (1992)

[17] **C.S. Rafferty, M.R. Pinto, Dutton R.W.** "Iterative Methods in Semiconductor Device Simulation" *IEEE Trans. Elect. Dev.* **ED–32** No. 10, 2018–2027 (1985)

[18] **S. Selberherr** *Analysis and Simulation of Semiconductor Devices* Springer Verlag Wien-New York (1984)

[19] **P. Sonneveld** "CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear Systems" *Rep.s of the Dept. of Mathematics and Informatics* **#84–16** Delft (1984)

[20] **T. Toyabe, H. Masuda, Y. Aoki, H. Shukuri, T. Hagiwara** "Three-Dimensional Device Simulator CADDETH With Highly Convergent Matrix Solution Algorithms" *IEEE Trans. Elect. Dev.* **ED–32** No. 10, 2038–2043 (1985)

[21] **H.A. Van der Vorst** "Bi-CGSTSAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Systems" *SIAM J. Sci. Stat. Comput.* March (1992)

[22] **T. Wada, R.L.M. Dang** "Modification of ICCG Method for Application to Semiconductor Device Simulators" *Sol. St. Elect.* **SSE–25** No. 11, 1083–1087 (1982)

[23] **T. Wada, R. Dang** "Development and Application of a High-Speed 2-Dimensional Time Dependent Device Simulator (MOS2C)" from *Proceedings of the Fourth International Conference on the Numerical Analysis of Semiconductor Devices and Integrated Circuits* (ed. J.J.H. Miller) Boole Press, Dublin (1985)