Preconditioned Iterative Methods for Nonsymmetric Linear Systems

C. den Heijer

ISA-ISC-TIS/CARD, Bldg SAQ-2
Nederlandse Philips Bedrijven B.V.
5600 MD Eindhoven, The Netherlands

Summary

In this paper we consider a class of iterative methods (projection methods) for the solution of linear equations. This class contains among others Conjugate Gradients, Bi-Conjugate Gradients and Orthomin. It appears that a variant of Bi-Conjugate Gradients (CG-Squared), when used with a so-called incomplete line block factorization as preconditioning, is a robust and efficient method for the solution of the nonsymmetric linear systems arising in the numerical solution methods for the coupled semiconductor equations.

1. The Problem

In this paper we shall be concerned with the (approximate) solution of linear equations that arise when solving a discretised semiconductor problem. More specifically the semiconductor equations under consideration is the following set of three coupled equations on $\Omega \subset \mathbb{R}^2$.

$$\text{(1.1a)} \qquad \operatorname{div}(-\varepsilon \operatorname{grad} V) - \rho(V,\phi_p,\phi_n) \equiv F_1(V,\phi_p,\phi_n) = 0$$

$$\text{(1.1b)} \qquad -\operatorname{div}\, \mu_p p(V,\phi_p) \operatorname{grad} \phi_p + R(V,\phi_p,\phi_n) \equiv F_2(V,\phi_p,\phi_n) = 0$$

$$\text{(1.1c)} \qquad -\operatorname{div}\, (\mu_n n(V,\phi_n) \operatorname{grad} \phi_n - R(V,\phi_p,\phi_n) \equiv F_3(V,\phi_p,\phi_n) = 0$$

where

$$\text{(1.1d)} \qquad \rho(V,\phi_p,\phi_n) = q(p(V,\phi_p) - n(V,\phi_n) + D)$$

and

$$\text{(1.1e)} \qquad \begin{aligned} p(V,\phi_p) &= n_i \exp((q/KT)\ (\phi_p - V)) \\ n(V,\phi_n) &= n_i \exp((q/KT)\ (V - \phi_n)) \end{aligned}$$

R is the recombination and may be of SRH or Auger type [10]. With the given boundary values, (1.1) is discretised with the Gummel-Scharfetter scheme on a (distortion of a) rectangular grid on Ω(see [8], [9]).

This result in a system of, say n, equations

(1.2a) $F(w)=0$

where

(1.2b) $F(w)=(F_1(w),F_2(w),.....,F_n(w))$

and

(1.2c) $w=(w_1,w_2,...,w_n)$

the components w_i of the solution of (1.2) are approxations to nodal values of V, ϕ_p and ϕ_n of (1.1).

We solve problem (1.2) by a continuation method (see [7], [9]). Each nonlinear subproblem is solved by a damped Newton method. This means that we have to solve a sequence of problems of the type

(1.3) $J(w).dw=-F(w)$

for the unknown vector dw.
Here J(w) is the Jacobian of F in the (given) point w. The numerical solution of problems of type (1.3) will be the topic of this paper.
(We note however that the methods considered are applicable to a much wider class than (1.3).)

It is easily verified that for most discretisations of Poisson's equation (1.1a) only, (1.3) is of the type

(1.4) $Ax=b$

where A is a positive definite matrix (b, $x\in\mathbb{R}^n$). However, when dealing with two or three coupled equations in (1.1), (1.3) is of the type

(1.5) $Bx=d$

where B is nonsymmetric.

Especially linear problems of this last type will be considered.

Some notational conventions.
The solution of (1.4) and (1.5) will be denoted by x^* (which is assumed to exist and to be unique). Matrix A of (1.4) will always be assumed to be symmetric (not necessarily positive definite, unless explicitly stated).

n: number of equations and unknown in (1.4,5)

(x,y) is the l_2 innerproduct in $\mathbb{R}^n$ $(x,y)=x^Ty$

$(x,y)_H=(x,Hy)$ for any symmetric nxn matrix H. When H is positive definite, $(.,.)_H$ is an innerproduct.

2. Some Iterative Methods (for Linear Equations)

In this section we present a class of iterative methods for (1.4,5) known as projection methods. We first give a general description of such methods.

2.1 The General Projection Method

let $\langle .,. \rangle$ be an innerproduct in $\mathbb{R}^n$. Let the vectors $p_0,p_1,\ldots,p_{k-1}$ satisfy

(2.1.1a) $\langle p_i,p_j \rangle = 0 \quad (i \neq j) \quad ;(p_i \neq 0)$.

Let

(2.1.1b) $K_k=\text{span}\{p_0,p_1,\ldots,p_{k-1}\}$

and let

(2.1.1c) $y_k=\sum_{j=0}^{k-1} \alpha_j p_j$

satisfy

(2.1.1d) $y_k=\arg \min_{y \in K_k} \langle x^*-y,x^*-y \rangle$.

Hence y_k is the projection of x^* onto K_k (w.r.t. $\langle .,. \rangle$).
If $y_k \neq x^*$, choose $p_k \neq 0$ such that

(2.1.1e) $\langle p_k,p_j \rangle = 0 \quad (\forall j<k)$

Let $K_{k+1}=\text{span}\{p_0,p_1,\ldots,p_k\}$ and

(2.1.1f) $y_{k+1}=\arg \min_{y \in K_{k+1}} \langle x^*-y,x^*-y \rangle$.

Then, obviously,

(2.1.1g) $$y_{k+1}=\sum_{j=0}^{k}\alpha_j p_j$$

where

(2.1.1h) $\alpha_k=\langle x^*,p_k\rangle / \langle p_k,p_k\rangle$ $(=\langle x^*-y_k,p_k\rangle / \langle p_k,p_k\rangle$;

because of (2.1.1c) and (2.1.1e)).

This means that

(2.1.1i) $y_{k+1}=y_k+\alpha_k p_k$.

We give some relations that are often used in projection methods. Since $x^*-y_{k+1}\perp K_{k+1}$, we have

(2.1.2) $\langle p_j,x^*-y_{k+1}\rangle=0 \quad (\forall\, j\leq k)$.

For most projection methods, p_0, p_1..... are chosen such that

(2.1.3) $K_j=\{y \mid y=\pi_j(B)d, \pi_j$ any polynomial of degree $\leq j\}$.

More specifically,

(2.1.4a) $p_0=d$

and

(2.1.4b) $$p_k=q_k-\sum_{j=0}^{k-1}\beta_{k,j}p_j$$

where $q_k\in K_k-K_{k-1}$ of (2.1.3) and

(2.1.4c) $\beta_{k,j}=\langle q_k,p_j\rangle/\langle p_j,p_j\rangle$

Hence p_k satisfies (2.1.1e), $p_k\neq 0$, and $p_k\in K_k$ of (2.1.3). In this case, by (2.1.2) and (2.1.4b)

(2.1.5) $\alpha_k=\langle x^*-y_k,p_k\rangle/\langle p_k,p_k\rangle=\langle x^*-y_k,q_k\rangle/\langle p_k\ p_k\rangle$

It is easily verified that whenever $K_k=K_{k-1}$ (of (2.1.3)), then $x^*\in K_{k-1}$ and hence $y_{k-1}=x^*$. So obviously $x^*=y_l$, for some $l\leq n$.

We give two examples. The first choice is

(2.1.6a) $q_k=r_k$

where

(2.1.6b) $r_k=d-By_k$

This only works when $\alpha_{k-1}\neq 0$ (otherwise $r_k=r_{k-1}\notin K_k$ of (2.1.3)). In that case, when B is symmetric w.r.t. $\langle\cdot,\cdot\rangle$

(2.1.7a) $\beta_{k,j}=\langle r_k,p_j\rangle/\langle p_j,p_j\rangle=\langle x^*-y_k,Bp_j\rangle/\langle p_j,p_j\rangle=0 \quad (\forall j\leq k-2)$

since $x^*-y_k\perp K_k$ and $Bp_j\in K_{j+2}$

Hence

(2.1.7b) $p_k=r_k-\beta_{k,k-1}p_{k-1}$

and (see (2.1.5))

(2.1.7c) $\beta_{k,k-1}=\langle x^*-y_k,\alpha_{k-1}Bp_{k-1}\rangle/\langle x^*-y_{k-1},r_{k-1}\rangle$

$=-\langle x^*-y_k,r_k\rangle/\langle x^*-y_{k-1},r_{k-1}\rangle$

since $x^*-y_k\perp K_k$ and $r_k\in K_k$.

Another choice is

(2.1.8) $q_k=Bp_{k-1} \quad (k>0)$.

(This choice works also when $\alpha_{k-1}=0$.)

In this case, when B is symmetric w.r.t. $\langle\cdot,\cdot\rangle$

(2.1.9a) $\beta_{k,j}=\langle Bp_{k-1},p_j\rangle/\langle p_j,p_j\rangle=\langle p_{k-1},Bp_j\rangle/\langle p_j,p_j\rangle$

$= 0 \qquad$ (for all $j\leq k-3$),

since $p_{k-1}\perp K_{k-1}$ and $Bp_j\in K_{j+2}$

Hence

(2.1.9b) $p_k=Bp_{k-1}-\beta_{k,k-1}p_{k-1}-\beta_{k,k-2}p_{k-2}$.

2.2 ORTHOMIN

In [1, 12] method (2.1.1) is proposed for problem (1.5) where $\langle\cdot,\cdot\rangle=(\cdot,\cdot)_{B^TB}$

Then

$$y_k=\arg\min_{y\in K_k}(d-By,d-By) \quad .$$

and $p_0, p_1, \ldots$ should satisfy

(2.2.1) $(Bp_i, Bp_j) = 0 \quad (i \neq j)$,

p_k is determined by (2.1.4,6)

Hence ORTHOMIN can be described as follows:

(2.2.2a) Start: $y_0 = 0$; $r_0 = d$, $p_0 = r_0$; $k = 0$.

while $(\|r_k\| > \varepsilon)$

do

(2.2.2b) $\alpha_k = (r_k, Bp_k)/(Bp_k, Bp_k)$ (cf. (2.1.5))

(2.2.2c) $y_{k+1} = y_k + \alpha_k p_k$

(2.2.2d) $r_{k+1} = r_k - \alpha_k Bp_k$

(2.2.2e) $p_{k+1} = r_{k+1} - \sum_{j=0}^{k} \beta_{k+1,j} p_j$; $\beta_{k+1,j} = (Br_{k+1}, Bp_j)/(Bp_j, Bp_j)$ (cf. (2.1.7a))

$k = k+1$

od

2.3 ORTHOMIN (m)

For k large, the amount of work involved for (2.2.2e) may be prohibitive. Therefore a variant of (2.2.2) is often used, for which (2.2.2e) is replaced by

(2.2.2e') $p_{k+1} = r_{k+1} - \sum_{j=k-m}^{k} \beta_{k+1,j} p_j$

Here $m \geq 1$ is given (generally $1 \leq m \leq 10$).

It is obvious that when B is nonsymmetric in most cases this variant is not a projection method.

2.4 Conjugate Gradients

The method of conjugate gradients for (1.4) can be derived from section 2.1 by putting

$$\langle .,. \rangle = (.,.)_A .$$

Hence, with the notation of 2.1

$$y_k = \arg\min_{y \in K_k} (b-Ay, b-Ay)_{A^{-1}}$$

and

$$(p_i, Ap_j) = 0 \qquad \text{(for all } i \neq j\text{)}.$$

p_k is determined by (2.1.4,6). Since A is symmetric w.r.t. $\langle \cdot,\cdot \rangle$ (2.1.7b) holds.

We give here a version of the method.

(2.4.1a) Start: $y_0=0$; $r_0=b$, $p_0=r_0$; $k=0$.

while $(\|r_k\| > \varepsilon)$

do

(2.4.1b) $\alpha_k = (r_k, r_k)/(p_k, Ap_k)$ (cf. (2.1.5))

(2.4.1c) $y_{k+1} = y_k + \alpha_k p_k$

(2.4.1d) $r_{k+1} = r_k - \alpha_k A p_k$

(2.4.1e) $p_{k+1} = r_{k+1} + \beta_k p_k$; $\beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k)$ (cf. (2.1.7c))

$k=k+1$

od

2.5 Bi-Conjugate Gradients

In [4],[3] the following generalisation of the CG-method was proposed, which may be applied to problems of type (1.5).

Let

(2.5.1) $$A = \begin{pmatrix} B & \emptyset \\ \emptyset & B^T \end{pmatrix}, \quad A \in L(\mathbb{R}^{2n}),$$

and

(2.5.2) $$b = (d,d)^T.$$

Consider

(2.5.3) $Aw=b$

where

$$w=(x,\tilde{x})^T, \qquad x,\tilde{x}\in\mathbb{R}^n,$$

then (2.5.3) is equivalent to

$$Bx=d$$

and

$$B^T\tilde{x}=d.$$

Define the "innerproduct" $[.,.]$ in $\mathbb{R}^{2n}$ by $[z_1,z_2]=z_1^TQz_2$ $(z_1,z_2\in\mathbb{R}^{2n})$, where

$$Q=\begin{pmatrix} \emptyset & I_n \\ I_n & \emptyset \end{pmatrix}$$

I_n is the n-dimensional unity operator.
$[.,.]$ is a symmetric and linear form, but not positive definite ($[z,z]\leqq 0$ for some $z\neq 0, z\in\mathbb{R}^{2n}$).
A is symmetric w.r.t. $[.,.]$.

The method of bi-conjugate gradients is derived by applying method (2.1.1) to problem (2.5.3) where

$$\langle z_1,z_2\rangle=[z_1,A^{-1}z_2].$$

p_k is determined by (2.1.4,6). Since A is symmetric w.r.t. $\langle .,.\rangle$, (2.1.7b) holds. The method is usually given as follows

(2.5.5a) Start $y_0=0$ $(\tilde{y}_0=0)$; $r_0=d$, $\tilde{r}_0=d$, $p_0=r_0$, $\tilde{p}_0=\tilde{r}_0$;

$k=0$.

while $(\|r_k\|>\varepsilon)$

do

(2.5.5b) $$\alpha_k=\frac{(\tilde{r}_k,r_k)}{(\tilde{p}_k,Bp_k)} \qquad \text{(cf. (2.1.5))}$$

(2.5.5c) $$y_{k+1}=y_k+\alpha_k p_k$$

$$(\tilde{y}_{k+1}=\tilde{y}_k+\alpha_k\tilde{p}_k)$$

(2.5.5d) $r_{k+1}=r_k-\alpha_k B p_k$

$\tilde{r}_{k+1}=\tilde{r}_k-\alpha_k B^T \tilde{p}_k$

(2.5.5e) $p_{k+1}=r_{k+1}+\beta_k p_k$; $\beta_k=\dfrac{(\tilde{r}_{k+1},r_{k+1})}{(\tilde{r}_k,r_k)}$ (cf.(2.1.7c))

$\tilde{p}_{k+1}=r_{k+1}+\beta_k \tilde{p}_k$

$k=k+1$

od

Some remarks

2.5.1 It is easily verfied that

$$r_k=\phi_k(B)r_0 \ , \quad \tilde{r}_k=\phi_k(B^T)\tilde{r}_0$$
$$p_k=\theta_k(B)r_0 \ , \quad \tilde{p}_k=\theta_k(B^T)\tilde{r}_0 \qquad (2.5.6)$$

where ϕ_k and θ_k are polynomials of degree k.

2.5.2 When B is symmetric, then method (2.5.5) is equivalent to the CG process.

2.5.3 $[z_1,A^{-1}z_2]$ is not an innerproduct. It is therefore clear that many theoretical aspects of the CG process do not hold for the bi-CG process.

2.5.4 In particular, the method breaks down when

$$(\tilde{p}_k,Bp_k)=0 \text{ or } (r_k,\tilde{r}_k)=0 \qquad (\text{and } r_k\neq 0).$$

See [6] and [3] on these aspects.

2.5.5 However it can be shown that whenever the bi-CG process does not break down, and $x^*\in K_k=\text{span}\{p_0,p_1,\ldots,p_{k-1}\}$ then $y_k=x^*$ (see [13, p. 390]).
Hence bi-CG is a "quasi-projection method".

2.6 CG-Squared (CGS)

The determination of $\tilde{r}_k$ and $\tilde{p}_k$ in (2.5.5) is only needed for the calculation of α_k and β_k. In particular, the matrix-vector product $B^T\tilde{p}_k$ of (2.5.5d) is only needed for an innerproduct. In [11] a variant of bi-CG is derived that does not need such matrix vector products.

Since the derivation of CGS is just a manipulation on the formulae of (2.5.5) we shall only indicate how it is done. By (2.5.6)

$$(r_k,\tilde{r}_k)=(\phi_k(B)r_0,\phi_k(B^T)r_0)=(\phi_k{}^2(B)r_0,r_0)$$

and

$$(\tilde{p}_k,Bp_k)=(B\theta_k{}^2(B)r_0,r_0)\,.$$

From (2.5.5) recursions in $\theta_k{}^2(B)r_0(=\hat{r}_k)$ and $\phi_k{}^2(B)r_0(=\hat{p}_k)$ that use $\theta_k(B)\phi_k(B)r_0(=e_k)$ and $\phi_k(B)\theta_{k-1}(B)r_0(=h_k)$ can be derived. For these last two terms also a recursion can be derived.
With $\hat{y}_k$ satisfying $\hat{r}_k=d-B\hat{y}_k$ the following process can be derived

(2.6.1a) Start: $\hat{y}_0=0$; $\hat{r}_0=d$, $\hat{p}_0=\hat{r}_0$;k=0.

while $(\|r_k\|>\varepsilon)$

do

(2.6.1b) $\alpha_k=(r_0,\hat{r}_k)/(r_0,B\hat{p}_k)$

(2.6.1c) $h_{k+1}=e_k-\alpha_kB\hat{p}_k$

(2.6.1d) $\hat{r}_{k+1}=\hat{r}_k-\alpha_kB(e_k+h_{k+1})$

(2.6.1e) $\hat{y}_{k+1}=\hat{y}_k+\alpha_k(e_k+h_{k+1})$

(2.6.1f) $\beta_k=(r_0,\hat{r}_{k+1})/(r_0,\hat{r}_k)$

(2.6.1g) $e_{k+1}=\hat{r}_{k+1}+\beta_kh_{k+1}$

(2.6.1h) $\hat{p}_{k+1}=e_{k+1}+\beta_k(h_{k+1}+\beta_k\hat{p}_k)$

k=k+1

od

In (2.5.5d) $r_k=\phi_k(B)d$ and in (2.6.7d) $\hat{r}_k=\phi_k{}^2(B)d$ (for the same plynomial ϕ_k).
Consequently, if $y_{k*}=x^*$ for some k^*, then $\hat{y}_k=x^*$.

Furthermore, if $\|\phi_k(B)d\|<<d$, then in many cases $\|\phi_k{}^2(B)d\|$ may be expected to be even smaller (see also [11]). Hence one may expect CGS to converge faster than bi-CG.

3. Preconditionings

In this section we present the two preconditionings we used with the iterative methods of the previous section to solve the linear problems.

3.1 A "block-Gauss-Seidel" Preconditioning

In problems involving the continuity equations the unknowns may be ordered in such a way that

$$B=\begin{pmatrix} D_1 & U_{1,1} & U_{2,2} \\ L_{2,1} & D_2 & U_{2,3} \\ L_{3,1} & L_{3,2} & D_3 \end{pmatrix}$$

where $D_1 \sim V$, $D_2 \sim \phi_p$ and $D_3 \sim \phi_n$.

Hence $B=L+D+U$, and (1.5) is equivalent to the following preconditioned system

$$(3.1.1) \qquad [(L+D)^{-1}B(I+U)^{-1}]\ (I+U)x=(L+D)^{-1}d$$

in short

$$(3.1.2) \qquad \tilde{B}y=\bar{d}.$$

It is easily verified that $(L+D)^{-1}q$ (for some q) can be obtained by a forward substitution process, requiring L-U decompositions of D_i $(i=1,2,3)$.
In section 4 we present some testresults for iterative methods of section 2 that are applied to the preconditioned system (3.1.2).

3.2 An Incomplete Line-block Factorization

Assume that the unknowns are ordered in such a way that

$$B=\begin{bmatrix} D_1 & U_1 & & \\ L_2 & D_2 & & \emptyset \\ & & & \\ \emptyset & & & U_{N-1} \\ & & L_N & D_N \end{bmatrix}$$

in short

$$B=L+D+U.$$

For example on (grids that are distortions of) a rectangular grid, many box-schemes, difference schemes and finite element schemes allow such orderings. In such cases the blocks B_i are associated with mesh-columns. From now on, we assume the grid to be (a distortion of) a rectangular grid.

In many cases Matrix B of type (3.2.1) can be decomposed as follows.

$$(3.2.2) \qquad B=(L+\Delta)\Delta^{-1}(\Delta+U)$$

where $\Delta=\text{diag}\ (\Delta_1,\ldots,\Delta_N)$

satisfies

$$\begin{aligned} &\Delta_1=D_1 \\ (3.2.3) \qquad &\Delta_j=D_j-L_j\Delta_{j-1}^{-1}U_{j-1} \qquad\qquad (j=2,3,..,N) \end{aligned}$$

In [5], [2] a factorization of B is proposed where

$$(3.2.4) \qquad B\cong(L+\tilde{\Delta})\tilde{\Delta}^{-1}(\tilde{\Delta}+U)$$

with

$$\begin{aligned} &\tilde{\Delta}_1=D_1 \\ (3.2.5) \qquad &\tilde{\Delta}_j=D_j-Sp_j(L_1\tilde{\Delta}_{j-1}^{-1}U_{j-1}) \qquad\qquad (j=2,3,..,N) \end{aligned}$$

where $Sp_j(C)_{k,l}\equiv\begin{cases}0 \text{ if } d_{k,l}^{j}=0\\ c_{k,l} \quad \text{otherwise}\end{cases}$

for an m x m matrix $C=(c_{k,l})$

Hence $\tilde{\Delta}_j$ has the same sparsity pattern as D_j.
Since L_j and U_{j-1} are sparse, it is obvious that not all elements of Δ_{j-1}^{-1} need to be calculated. We only need the main diagonal and some co-diagonals. These can be calculated quite easily in many cases.
For most discretisations used on (distortions of) rectangular gids, D_j has a tridiagonal structure, so that

$$(3.2.6)\qquad j=\begin{pmatrix}\theta_1 & \upsilon_1 & \emptyset\\ \lambda_2 & & \upsilon_{M-1}\\ \emptyset & \lambda_M & \theta_M\end{pmatrix}$$

that is: $\Delta_j = \Lambda_j+\Theta_j k+U_j$
where λ_j,θ_j and υ_j are 3 x 3 matrices.
Δ_j may be decomposed into

$$(3.2.7a)\qquad \Delta_j=(\Lambda_j+\Gamma_j)\Gamma_j^{-1}(\Gamma_j+U_j)$$

where

$$(3.2.7b)\qquad \Gamma_j=\mathrm{diag}\ (\gamma_1,..,\gamma_M)$$

and

$$(3.2.7c)\qquad \begin{aligned}\gamma_1 &=\theta_1\\ \gamma_1&=\theta_j-\lambda_j\gamma_{j-1}^{-1}\upsilon_{j-1} \qquad (j=2,3,..M).\end{aligned}$$

Let

$$(3.2.8)\qquad S_j=\Delta_j^{-1}$$

then, with $S_j=(s_{k,l})$,

we have (see [5])

$$(3.2.9a)\qquad s_{M,M}=\gamma_M^{-1}$$

$$(3.2.9b)\qquad s_{k,k}=\gamma_k^{-1}+\gamma_k^{-1}\cdot\upsilon_k s_{k+1\ k+1}\cdot\lambda_{k+1}\cdot\gamma_k^{-1}$$

$$(3.2.9c)\qquad \begin{aligned}s_{k,k-1}&=-s_{k,k-1+1}\cdot\lambda_{k-1+1}\gamma_{k-1}^{-1}\\ s_{k-1,k}&=-\gamma_{k-1}^{-1}\cdot\upsilon_{k-1}s_{k-1+1,k}\end{aligned}$$

$$(l=1,2,..k-1)$$
$$(k=M-1,M-2,..,1).$$

Remark: When dealing with a five-point discretisation, L_j and U_j are diagonal and only 3 diagonals of S_j need to be calculated. For the usual nine-point scheme, 7 diagonals of S_j are needed for (3.2.5).

We resume:

$$B\cong(L+\Delta)\Delta^{-1}(\Delta+U)$$

where

$$\Delta=(\Lambda+\Gamma)\,\Gamma^{-1}(\Gamma+U)$$

is a block-diagonal matrix (cf. (3.2.2), (3.2.4) and (3.2.7)). The following equation is now equivalent to (1.5)

$$[\Gamma(\Gamma+U)^{-1}(L+\Delta)^{-1}B(\Delta+U)^{-1}(\Lambda+\Gamma)](\Lambda+\Gamma)^{-1}(\Delta+U)x= \tag{3.2.10a}$$
$$\Gamma(\Gamma+U)^{-1}(L+\Delta)^{-1}d$$

in short

$$\hat{B}\hat{y}=\hat{d}. \tag{3.2.10b}$$

In the next section we present some results with the preconditioned problem (3.2.10).

3.3 A Simplification of the Line-block Factorization

Instead of (3.2.4,5) we shall also consider the follwing factorization of B,

$$B\cong(L+\tilde{\Delta})\,\tilde{\Delta}^{-1}(\tilde{\Delta}+U) \tag{3.3.1}$$

where

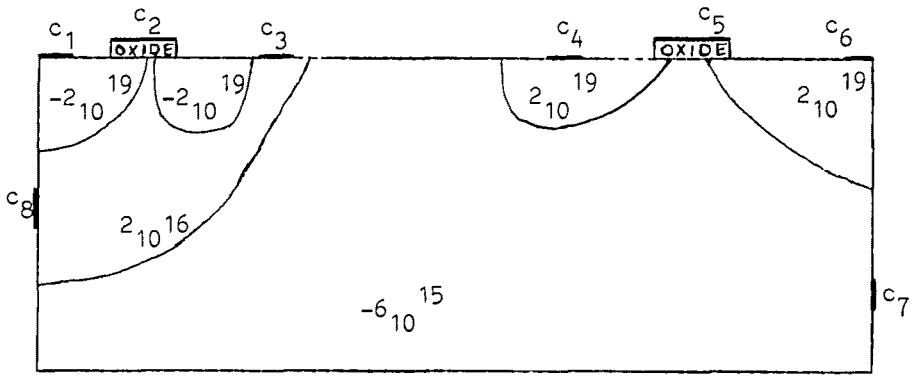
$$\tilde{\Delta}_j=D_j \qquad (j=1,2,..,N). \tag{3.3.2}$$

Obviously, (3.3.1,2) is a cheaper preconditioning than (3.2.4,5).

4.1 Testresults

In this section we present testresults for some combinations of iterative methods of section 2 and preconditionings of section 3, when applied to linear problems arising in the calculations on a CMOS Inverter. The device is described below

Figure 4.1
CMOS INVERTER



c_1 (p-source): $\phi_p=5, \phi_n=5$, charge neutrality (c.n);

c_2 (p-gate): $V=5$;

c_3 (p-drain): $\phi_p=0$, $\phi_n=0$, c.n.;

c_4 (n-drain): $\phi_p=0$, $\phi_n=0$, c.n.;

c_5 (n-gate): $V=-5$;

c_6 (n-source): $\phi_p=-5$, $\phi_n=-5$, c.n.;

c_7 (p-substrate): $\phi_p=-5$, $\phi_n=-5$, c.n.;

c_8 (n-well): $\phi_p=4.275 \longrightarrow 4.2725$, $\phi_n=4.275 \longrightarrow 4.2725$, c.n.

The top-dope values are given per μ^3.

The Gummel-Scharfetter scheme ([8]) was used on a nonuniform 48 x 30 mesh. This resulted in a system of equations

(4.1) $F(w)=0$

with

(4.2) $F:\mathbb{R}^n \longrightarrow \mathbb{R}^n$.

In this case, n=4128.
We obtained solutions of problem (1.1) for different values of ϕ_p and ϕ_n at the contact c_8, by means of a continuation method (see [7]).

Each nonlinear subproblem was solved by Newton's method (with damping), that is, for each subproblem a sequence w^k had to be generated, where

$$w^{k+1}=w^k+\lambda_k dw_k \tag{4.3}$$

λ_k suitably chosen

and

$$J(w^k)dw_k=-F(w^k) \qquad (k=0,1,\ldots). \tag{4.4}$$

Table 4.1 gives results for one such Newton process. In that case w^0 is the solution of (4.1) with $\phi_p=\phi_n=4.275$ at c_8 and F (and J) are associated with $\phi_p=\phi_n=4.2725$ at c_8. It took 4 Newton iterations for the stopping criterion to be satisfied. The tables below give results for several methods to solve the linear problems (4.4). The CPU times given include the time needed for assembling the Jacobian J and righthandside F, etc.

Table 4.1
Testresults for CMOS-inverter (one Newton process)

method	preconditioning	CPU-time
(2.3);m=10	(3.1)	failure*
(2.3);m=10	(3.2)	failure*
(2.6)	(3.1)	445^s
(2.6)	(3.2)	230^s
Direct (MA32AD, Harwell)		720^s

Table 4.2
Testresults CMOS-inverter (first Newton-correction)

method	preconditioning	# iterations	CPU-time
(2.6)	(3.2)	23	88^s
(2.6)	(3.3)	44	120^s
Direct (see above)			188^s

* see next section.

5. Conclusions

Method (2.3) is not a projection method. It is our experience (see e.g. Table 4.1) that this causes the process to "converge" very slowly for several problems. That is, very small correction steps are being taken while the approximations are far away from the solution. (A similar behaviour can be observed in gradient methods for linear problems.) The "quasi-projection method" CGS (2.6) does not have this draw-back. Although there is hardly any theoretical evidence, it appears to work very well, when used with the proper type of preconditioning.

Both preconditionings (3.1,2) appear to work well although the "Block-Gauss-Seidel" preconditioning (which requires 3 L-U decompositions of n/3 x n/3 matrices) is much more expensive.

In conclusion, CGS with line block preconditioning is a very robust combination to solve the linear problems arising in coupled semiconductor problems. It is also much more efficient than Gaussian elimination.

References

[1] O. Axelsson, "Conjugate Gradient Type Methods for Unsymmetric Systems of Linear Equations"
Linear Algebra and Its Appl., vol 29, p.1, 1980

[2] O. Axelsson, et al, "On Some Versions of Incomplete Block-Matrix Factorization Iterative Methods"
Rept. 8322, Cath. University, Dept. of Math., Nijmegen, The Netherlands, 1983

[3] R. Fletcher, "Conjugate Gradient Methods for Indefinite Systems"
Proc. of the Dundee Biennial Conference on Numerical Analysis (G.A. Watsone (ed.)), Springer-Verlag, N.Y., 1975

[4] C. Lanczos, "Solution of Systems of Linear Equations by Minimized Iteration"
J. Res. N.B.S., 49 p.33, 1952

[5] J.A. Meijerink, "Iterative Methods for the Solution of Linear Equations Based on Incomplet Factorization of the Matrix"
Publ. 643, Shell, Rijswijk, The Netherlands, 1983

[6] B.N. Parlett & D. Taylor, "A Look Ahead Lanczos Algorithm for Unsymmetric Matrices"
Rept. PAM-43, University of California, Berkeley, Center for Pure and Applied Math., 1981

[7] S.J. Polak, et. al., "Automatic Problemsize Reduction for On-State Semiconductor Problems"
IEEE Trans. Elect. Dev., Vol ED-30, p.1050, 1983

[8] D.L. Scharfetter & H.K. Gummel, "Large-Signal Analysis of a Silicon Read Diode Oscillator"
IEEE Trans. Elect. Dev., Vol ED-16, p.64, 1969

[9] W. Schilders, et. al., "A Comparison of Subset Solving Algorithms"
Proceedings of NASECODE III (J. Miller (ed.)), Boole Press, Dublin, p.258, 1983

[10] J.W. Slotboom, "Analysis of Bipolar Transistors", Ph.D Dissertation, University Eindhoven, The Netherlands, 1977

[11] P. Sonneveld, "CGS, a Fast Lanczos-type Solver for Nonsymmetric Linear Systems", Rept. 84-16, Delft University, Dept. of Math., Delft, The Netherlands, 1984

[12] P.K.W. Vinsome, "ORTHOMIN- An Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations" Proc. Fourth. SPE Symposium on Reservoir Simulation, L.A., p.149, 1976

[13] J.H. Wilkinson, "The Algebraic Eigenvalue Problem" Clarendon Press, Oxford, 1972