Invited Paper

Tridiagonal Factorization Algorithm: A Preconditioner for Nonsymmetric System Solving on Vectorcomputers

SHUN DOI* and NORIO HARADA*

The Tridiagonal Factorization (TF) algorithm was originally introduced by the authors as a matrix splitting type preconditioner for regular sparse nonsymmetric system solving on vector and/or parallel computers. It has been re-introduced using a Preconditioner Introduction Process, which also gives an alternative introduction process to the ILU preconditioner. The approximation properties of the TF and ILU preconditioners against the coefficient matrix of linear systems have been analyzed. It is shown that they have similar approximation properties versus the change in diffusion anisotropy as well as advection intensity of an advection diffusion equation. Convergence and CPU-time of both preconditioners have been compared for some practical 2D and 3D device simulation problems on the NEC SX-2 supercomputers. It is observed that the iterative procedures with the TF preconditioner are up to 3 times faster than those with vectorized ILU preconditioners.

1. Introduction

This paper discusses the solution for regular sparse nonsymmetric linear systems, suitable for vector and/or parallel computation, arising from the finite difference approximation for advection diffusion equations. The most promising class of the solution methods at present includes preconditioned iterative methods, such as the preconditioned Bi-Conjugate gradient (BCG)[1], preconditioned Conjugate Gradient Squared (CGS)[2] and preconditioned Conjugate Residual (CR) [3] methods. Especially for the ILUBCG and ILUCGS methods, the combination of the Incomplate LU factorization preconditioning technique[4] with the BCG and CGS basic iterative procedure, are thought to be the most efficient and the most popular methods for use on scalar computers. However, these methods have sequential features when conventionally programmed. Among several vectorization techniques reported[6],[7], [8], the most efficient and robust one, at the moment, is to change the order of computation, called the diagonalwise and hyperplane vectorization techniques[8],[9].

Algorithms inheriting more 'natural' parallelism are investigated because it is confidently expected that they will lead to more efficient processes. However, additional mathematical problems arise which must be analyzed. The Tridiagonal Factorization (TF) algorithm is proposed as a preconditioning algorithm for the solution of regular sparse nonsymmetric linear systems, suitable for vector and/or parallel computation[14]. The TF preconditioning matrix, termed the TF preconditioner, is defined by matrix splitting to tridiagonal matrices, each of which corresponds to an independent variable of P.D.E.s.

Section 2 presents a Preconditioner Introduction Process, which gives an alternative way of introducing both the TF and ILU preconditioners. Section 3 serves to show that the approximation properties of both the TF and ILU preconditioners against the original coefficient matrix of linear systems have quite similar features versus changes in diffusion anisotropy as well as advection intensity in P.D.E.s. Section 4 discusses parallelism and vectorization of both the TF and ILU preconditioners for 2-dimensional 5-point and 3-dimensional 7-point difference problems. In section 5, comparison with 2- and 3-dimensional practical device simulation on the NEC SX-2 supercomputer has shown the effectiveness of the TF preconditioner.

2. The Tridiagonal Factorization Preconditioner

The original TF preconditioner is in the class of matrix splitting algorithms. In this section, the Preconditioner Introduction Process is shown to be an alternative definition process to both the TF and ILU preconditioners. This process leads to a better understanding of the TF preconditioners as concluded in this section. Although discussions here are restricted to the 2-dimensional 5-point finite difference approximation of P.D.E.s defined over rectangular regions, they can easily be expanded to other cases, including higher order difference cases as well as 3-dimensional cases.

Consider the linear system,

*M.B.31-3610
Information Basic Research Laboratory C & C Information Technology Research Laboratories NEC Corporation, 4-1-1, Miyazaki, Miyamae-ku, Kawasaki, 213 JAPAN.

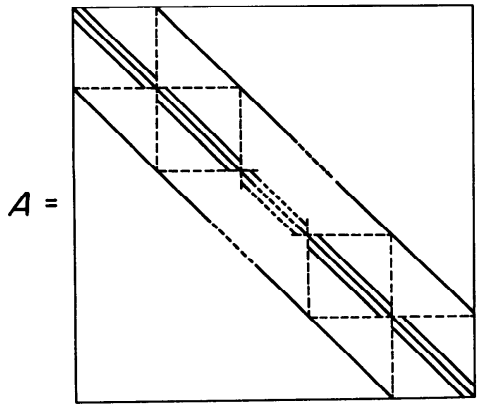

Fig. 1 An Illustrative matrix structure for 2D 5-point finite difference approximation.

$$Au=f \tag{1}$$

where A is a given regular sparse nonsymmetric matrix, such as shown in Fig. 1. Let M be a non-singular matrix, Eq. (1) is equivalent to

$$(AM^{-1})(Mu)=f. \tag{2}$$

Equation (2) is generally known as the preconditioned system, and M is the preconditioning matrix or preconditioner. The preconditioned iterative method is, namely, a combination of a basic iterative procedure, such as BCG[1], CGS[2] or CR[3], and a preconditioning technique, e.g. ILU[4], SSOR[5] or TF[14].

Note that the following three requirements should be satisfied on any successful preconditioners:

1) The computational work for preconditioning matrix inversion, usually forward and backward substitutions, is $O(N)$, where N is the matrix size. This is because the computational work should be consistent with that for a basic iterative procedure.

2) M is an appropriate approximation to the original matrix A. This comes from the fact that coefficient matrix eigenvalues concentration generally improves convergence.

3) The M inversion process has parallelism suitable for the computer architecture.

On computers with higher vector and/or parallel processing capabilities, the third requirement for parallelism is a key for a more efficient preconditioner. This is discussed in more detail in section 4.

Define a 2-dimensional 5-point difference matrix as

$$A[i]=[\underset{i-n_x}{b_i}\ \underset{i-1}{c_i}\ \underset{i}{d_i}\ \underset{i+1}{e_i}\ \underset{i+n_x}{f_i}] \tag{3}$$

which indicates non-zero row elements in i-th column. The term n_x is the number of gridpoints along the x axis.

With this notation, the original definition of the TF preconditioner, M_{TF}, is written as,

$$M_{TF}=XDY,$$

Table 1 The number of operations for M^{-1} (or M^{-T}) for one grid-point per one iteration step.

	addition	multipl.
2D 5-point	4	5
3D 7-point	6	7

where

$$X=[0\ c_i\ \ d_i\ \ e_i\ 0],$$
$$D=[0\ 0\ 1/d_i\ 0\ 0],$$
$$Y=[b_i\ 0\ \ d_i\ \ 0\ f_i]. \tag{4}$$

D is a diagonal matrix and X is a tridiagonal matrix. A simple permutation also makes Y a tridiagonal matrix.

X and Y, respectively, are to be factored to multiples of upper and lower triangular matrices, for M_{TF} inversion in every iteration step. Note that these LU factorizations are complete and cause no filling-in, and that the computational work for M_{TF}^{-1} is the same as that for M_{ILU}^{-1} (Table 1).

A Preconditioner Introduction Process is introduced:

STEP 1. Introduce the preconditioner M as a multiple of component matrices, whose non-zero elements are unknown variables. Each component matrix must be easily inverted and have a specific zero and non-zero structure.

STEP 2. Compute the multiple of component matrices and express the non-zero elements in M as functions of the unknown variables in the component matrices.

STEP 3. Determine the unknown variables of the component matrices so as to satisfy the following equivalence condition: Each non-zero element in A has the same value as the corresponding element in M.

First, the TF preconditioner is introduced using this process.

STEP 1. Let $M_{TF}=XDY$, where

$$X=[0\ C_i\ 1/D_i\ E_i\ 0],$$
$$D=[0\ 0\ \ D_i\ \ 0\ 0],$$
$$Y=[B_i\ 0\ 1/D_i\ 0\ F_i]. \tag{5}$$

B_i, C_i, D_i, E_i and F_i are unknown variables. X and Y (after permutation) are tridiagonal matrices. The difference from Eq. (4) is that elements are unknown.

STEP 2. Computing the multiple XDY results in,

$$\{XDY\}[i]=[\ \begin{array}{ll} & \text{row} \\ C_iD_{i-1}B_{i-1} & i-n_x-1 \\ B_i & i-n_x \\ B_iD_{i+1}E_{i+1} & i-n_x+1 \\ C_i & i-1 \\ 1/D_i & i \\ E_i & i+1 \\ C_iD_{i-1}F_{i-1} & i+n_x-1 \end{array}$$

$$\begin{matrix} F_i & i+n_x \\ E_iD_{i+1}F_{i+1}]. & i+n_x+1 \end{matrix} \qquad (6)$$

STEP 3. From the equivalence condition; i.e., i, $i\pm1$ and $i\pm n_x$ rows in Eq. (6) have the same values as the corresponding rows in Eq. (3),

$$B_i=b_i,\ C_i=c_i,\ E_i=e_i,\ F_i=f_i,\ D_i=1/d_i. \qquad (7)$$

Substituting Eq. (7) into Eq. (5) yields Eq. (4), the original definition of the TF preconditioner.

Next, the ILU preconditioner is introduced.

STEP 1. Let $M_{\mathrm{ILU}}=LDU$, where

$$L=[B_i\ C_i\ 1/D_i\ 0\ 0],$$
$$D=[0\ 0\ D_i\ 0\ 0],$$
$$U=[0\ 0\ 1/D_i\ E_i\ F_i]. \qquad (8)$$

Here, B_i, C_i, D_i, E_i and F_i are unknown variables. D is a diagonal matrix, and L and U, respectively, gives the lower and upper triangular matrices which have the same non-zero structure as A. It should also be noted here that the difference from Eq. (5) is only in the position of two unknown elements; B_i and E_i.

STEP 2. Computing the multiple LDU results in,

$$\{LDU\}[i]=[\ \begin{matrix} B_i & i-n_x \\ B_iD_{i-n_x}E_{i-n_x} & i-n_x+1 \\ C_i & i-1 \\ B_iD_{i-n_x}F_{i-n_x}+C_iD_{i-1}E_{i-1}+1/D_i & i \\ E_i & i+1 \\ C_iD_{i-1}F_{i-1} & i+n_x-1 \\ F_i\ ]. & i+n_x \end{matrix} \qquad (9)$$

(row)

STEP 3. Comparing Eq. (9) with Eq. (3) yield the following equivalences,

$$B_i=b_i,\ C_i=c_i,\ E_i=e_i,\ F_i=f_i,$$
$$b_if_{i-n_x}D_{i-n_x}+c_ie_{i-1}D_{i-1}+1/D_i=d_i. \qquad (10)$$

Equation (10) is rewritten as

$$D_i=1/(d_i-b_if_{i-n_x}D_{i-n_x}-c_ie_{i-1}D_{i-1}). \qquad (11)$$

It can be easily shown that Eq. (11) gives the recursive equation for the ILU preconditioner.

From the above discussions, the following are observed:

1) The TF preconditioner and the ILU preconditioner can be introduced from the same process, which we call the Preconditioner Introduction Process.

2) Their major difference is in their structure; i.e., triangular matrices for the ILU and tridiagonal matrices for the TF.

3) They both satisfy the equivalence condition: "Each non-zero element in A has the same value as the corresponding element in M."

3. Evaluating the Approximation Properties

Equations (6) and (9) indicate that the TF preconditioner has 4 error terms in the $i-n_x-1$, $i-n_x+1$, $i+n_x-1$ and $i+n_x+1$ columns for each i-th row, whereas the ILU has 2 in the $i-n_x+1$ and $i+n_x-1$ columns. Figure 2 gives an alternative expression of these error structures. In this section, these error terms for both the TF and ILU preconditioners are evaluated and their variation is compared against the change of anisotropy in diffusion parameters and the advection intensity.

Define an approximation error matrix R by $M-A$ and express the i-th row as

$$R[i]=[\ \underset{i-n_x-1}{r^1}\quad \underset{i-n_x+1}{r^2}\quad \underset{i+n_x-1}{r^3}\quad \underset{i+n_x+1}{r^4}\ ]$$

In the following, discussions are restricted to the case where $b_i=b$, $c_i=c$, $d_i=d$, $e_i=e$ and $f_i=f$, and hence subscript i will be omitted.

The TF preconditioner error is easily obtained from Eq. (6) and (7) as follows:

$$r^1_{\mathrm{TF}}(=r^1_{\mathrm{TF}\,i}=c_ib_{i-1}/d_{i-1})=bc/d,$$
$$r^2_{\mathrm{TF}}(=r^2_{\mathrm{TF}\,i}=e_ib_{i+1}/d_{i+1})=be/d,$$
$$r^3_{\mathrm{TF}}(=r^3_{\mathrm{TF}\,i}=c_if_{i-1}/d_{i-1})=cf/d,$$
$$r^4_{\mathrm{TF}}(=r^4_{\mathrm{TF}\,i}=e_if_{i+1}/d_{i+1})=ef/d. \qquad (12)$$

Although the recursive equation (11) prevents the ILU preconditioner error from being expressed as an explicit function of given elements such as b_i and c_i, the following approximate equations are obtained on an assumption that the series $\{D_i\}$ converge to a definite value D. This assumption is actually satisfied on some simple numerical examples. From Eq. (9) and (10), in which D_i, D_{i-1} and D_{i-nx} are replaced by D, it follows that

$$r^1_{\mathrm{ILU}}=r^4_{\mathrm{ILU}}=0,$$
$$r^2_{\mathrm{ILU}}=0.5be\{d-(d^2-4(bf+ce))^{0.5}\}/(bf+ce),$$
$$r^3_{\mathrm{ILU}}=0.5cf\{d-(d^2-4(bf+ce))^{0.5}\}/(bf+ce). \qquad (13)$$

An advection diffusion equation,

$$(k_xu_x)_x+(k_yu_y)_y-v_xu_x=0,$$

is now used for evaluating Eq. (12) and (13) against the

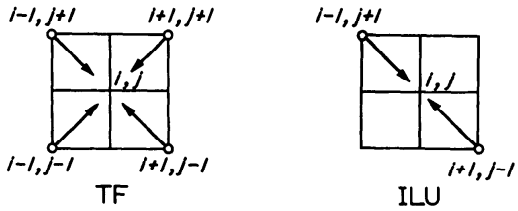

Fig. 2 Spatial expression of preconditioner errors.

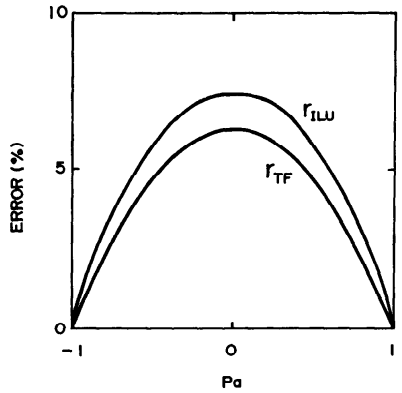

Fig. 3 Approximation errors for preconditioners vs. anisotropy in the diffusion factors in P.D.E.s.

change in the diffusion factors k_x and k_y, and the advection term intensity v_x. Terms u_x and u_y are partial derivatives. Adapting the 1st-order centered difference approximation to the diffusion term u_{xx} and u_{yy}, and 1st-order up-wind difference approximation to the advection term u_x results in 5-point finite difference equations, whose matrix has the form shown in Fig. 1. The grid size Δx and Δy are, for simplicity, fixed as 1.

First, in order to analyze an approximation property of r_{TF} and r_{ILU} against the diffusion anisotropy, let v_x be 0 and express b, c, d, e and f as functions of k_x and k_y:

$$b=f=-k_y,\ c=e=-k_x,\ d=2(k_x+k_y).$$

Substituting these into Eq. (12) and (13), it follows that:

$$r^1_{TF}=r^2_{TF}=r^3_{TF}=r^4_{TF}=k_xk_y/2(k_x+k_y),$$
$$r^2_{ILU}=r^3_{ILU}=k_xk_y\{k_x+k_y-(2k_xk_y)^{0.5}\}/(k_x^2+k_y^2) \quad (14)$$

Equation (14) indicates that all non-zero elements in R_{TF} become equal. It also indicates that all non-zero elements in R_{ILU} have approximately the same magnitude. Define an anisotropy parameter P_a as

$$P_a=(k_x-k_y)/(k_x+k_y),$$

which has value in the range -1 to 1. Figure 3 shows the change in r_{TF} and r_{ILU} (both normalized by A diagonal elements) against P_a.

In the same manner, an approximation property of r_{TF} and r_{ILU} against the advection intensity can be ana-

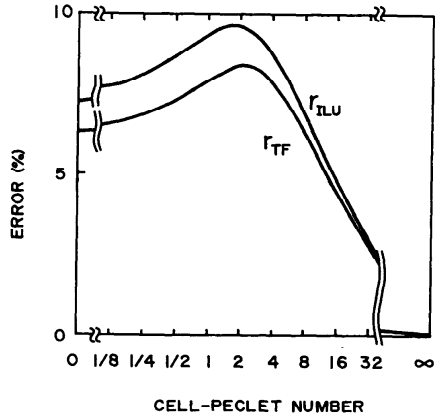

Fig. 4 Approximation errors for preconditioners vs. advection intensity in P.D.E.s.

lyzed. Let k_x and k_y be unity and express b, c, d, e and f using v_x:

$$b=c=f=-1,\ d=4+v_x,\ e=-(1+v_x).$$

Substituting these to Eq. (12) and (13), it follows that:

$$r^1_{TF}=r^3_{TF}=1/(4+v_x),$$
$$r^2_{TF}=r^4_{TF}=(1+v_x)/(4+v_x),$$
$$r^2_{ILU}=0.5(1+v_x)\{4+v_x-(8+4v_x+v_x^2)^{0.5}\}/(2+v_x),$$
$$r^3_{ILU}=0.5\{4+v_x-(8+4v_x+v_x^2)^{0.5}\}/(2+v_x), \quad (15)$$

Equation (15) indicates that r^2_{TF} ($=r^4_{TF}$) and r^2_{ILU} are major parts of the approximation error, because others simply decrease as v_x increase. The Cell-Peclet number P_c, an advection term intensity parameter, is defined:

$$P_c=\Delta x*v_x/k_x=v_x.$$

Figure 4 shows the change in r^2_{TF} and r^2_{ILU} (both normalized by A diagonal elements) against P_c.

From these two figures, the following are concluded:

1) Both r_{TF} and r_{ILU} change against P_a or P_c with almost the same behavior.

2) r_{TF} is always about 20% smaller than r_{ILU}. Note that this does not mean the TF preconditioner is superior to the ILU preconditioner, because the TF preconditioner has 4 error terms whereas the ILU has 2.

3) Both r_{TF} and r_{ILU} become smaller as either the diffusion anisotropy or the advection intensity become larger. Both r_{TF} and r_{ILU} have a peak around $P_c=2$.

These analyses indicate that the TF preconditioner convergence performance is not significantly inferior to the ILU preconditioner. Therefore, on computers with high vector and/or parallel processing capabilities, the requirement for parallelism is a crucial factor for estimating the efficiency of the preconditioners. The next section discusses parallelism and vectorization of both preconditioners.

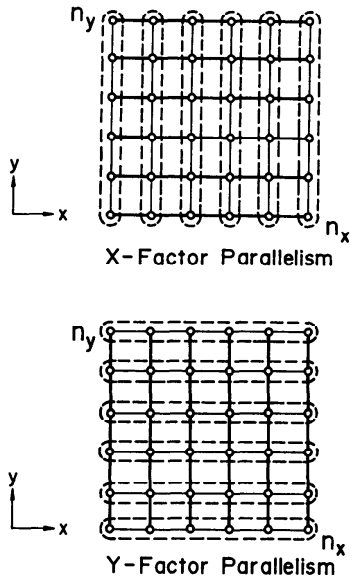

Fig. 5 Parallelism for TF preconditioner.

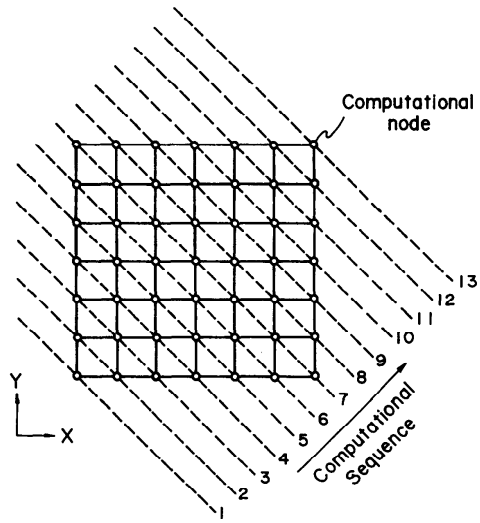

Fig. 6 Parallelism for ILU preconditioner.

4. Parallelism and Vectorization

4.1 2-Dimensional 5-Point Difference Case

First, the TF Preconditioner vectorization is discussed. The x-direction factor, X, of the 2D TF preconditioner defined by Eq. (4) has no y-direction connection term. Therefore, inverting X involves n_y independent tasks, where n_y is the number of gridpoints along the y-axis. Inverting Y also involves n_x independent tasks, where n_x is the number of gridpoints along the x-axis. Figure 5 shows the parallelism for the TF preconditioner. Gridpoints encircled by dotted lines are computed in parallel. These parallel tasks can also be executed in a vectorizable manner. For the inverse of the X factor, the vector length is n_y, and the stride is n_x. For the inverse of the Y factor, the vector length is n_x, and the stride is 1. Note that this TF preconditioner inverse requires no special vectorization techniques, and that a simple FORTRAN program yields a vectorizable code.

Next, consider the forward substitution process, $v=L^{-1}g$, of the ILU preconditioner for the 2D 5-point case; i.e.,

$$v_{ij}=(g_{ij}-l_{ij-1}v_{ij-1}-l_{i-1j}v_{i-1j})/l_{ij}. \tag{16}$$

Gridpoints with $i+j=$constant can be executed in parallel. In Fig. 6, dotted lines indicate this parallelism. This results in a vectorizable do-loop with a stride of n_x-1. This is called the diagonalwise vectorization technique. Note that the vector length for this technique ranges from 1 to min (n_x, n_y), and that the average vector length for a typical case with $n_x=n_y=n$ is $n/2$, half of that for the TF preconditioning technique. This variable short vector length is the major drawback of ILU preconditioner vector processing.

4.2 3-Dimensional 7-Point Difference Case

Next, consider a 3D 7-point difference case. First, the TF preconditioner for a 3D 7-point difference matrix is introduced. Denote the 3D 7-point difference matrix A:

$$A[i]=[\underset{i-n_xn_y}{a_i}\ \underset{i-n_x}{b_i}\ \underset{i-1}{c_i}\ \underset{i}{d_i}\ \underset{i+1}{e_i}\ \underset{i+n_x}{f_i}\ \underset{i+n_xn_y}{g_i}],$$

where n_x and n_y are the number of gridpoints along the x and y axis, respectively. The TF preconditioner, M_{TF}, for this matrix is defined as follows:

$$M_{\mathrm{TF}}=XDYDZ$$

where

$$D[i]=[0\ 0\ 0\ 1/d_i\ 0\ 0\ 0],$$
$$X[i]=[0\ 0\ c_i\ \ d_i\ \ e_i\ 0\ 0],$$
$$Y[i]=[0\ b_i\ 0\ \ d_i\ \ 0\ f_i\ 0],$$
$$Z[i]=[a_i\ 0\ 0\ \ d_i\ \ 0\ 0\ g_i].$$

X is a tridiagonal matrix. Y and Z can also be transformed to tridiagonal matrices by a simple permutation as in the 2D case. Note that this definition can also be introduced by the Preconditioner Introduction Process.

The inversion of M_{TF} is executed factor by factor. X is, for example, obtained by omitting the y- and z-direction connection terms in A. Hence, X^{-1} has $n_y^*n_z$ independent tasks. This is also the same for the y- and z-direction terms. The computational work to invert M_{TF} and M_{ILU} is equal (Table 1).

Next, the vectorization of this 3D TF preconditioner

is discussed. Suppose that the 3D gridpoints (i, j, k) are assigned on a memory unit so as to be continuous first in the x (or i) direction, then continuous in the y (or j) direction and finally continuous in the z (or k) direction. This is practically the simplest and the most popular mapping when they are coded by FORTRAN. Then, X factor inversion vector processing has $n_y^* n_z$ vector length with a stride of n_x. Z factor inversion vector processing has $n_x^* n_y$ vector length with stride 1. Although Y factor inversion has $n_x^* n_z$ independent tasks, it does not have a constant stride. The inversion consists of n_z sets of n_x continuous data units, where each set has a stride of $n_x^* n_y$. Therefore, it can not be executed by a single vector operation on the state-of-the-art vector computers. As a result, for the Y factor inversion, the vector length is n_x with stride 1. Sophisticated compilers, which can vectorize douple do-loops, do not require any special programming techniques to vectorize the inversion of this TF preconditioner.

Two implementations, the diagonal and the hyperplane vectorizations, are introduced for the ILU preconditioner inversion for the 3D 7-point difference case. Consider the forward substitution process, $v=L^{-1}g$, of the ILU preconditioner for the 3D 7-point case; i.e.,

$$v_{ijk}=(g_{ijk}-l_{ijk-1}v_{ijk-1}-l_{ij-1k}v_{ij-1k}-l_{i-1jk}v_{i-1jk})/l_{ijk}. \quad (17)$$

Gridpoints (i, j, k) lying in a hyperplane, defined by $i+j+k=$constant, are executed in parallel. In this case, the stride is not a constant, as in the 2D case. The most popular and the simplest way to achieve a long vector length is to use indirect list-vectors. This is called the hyperplane technique. Equation (17) can be divided into the following two steps:

$$h_{ijk}=g_{ijk}-l_{ijk-1}v_{ijk-1}. \quad (18)$$

$$v_{ijk}=(h_{ijk}-l_{ij-1k}v_{ij-1k}-l_{i-1jk}v_{i-1jk})/l_{ijk}. \quad (19)$$

Equation (18) for $k=$constant has $n_x^* n_y$ independent tasks. Therefore, they are computed in a vectorizable manner. The vector length is $n_x^* n_y$ with stride 1. Equation (19), on the other hand, is equivalent to Eq. (16). Therefore, the diagonalwise vectorization technique can be used to vectorize Eq. (19). The average vector length for Eq. (19) for a typical case with $n_x=n_y=n$ is $n/2$, as in the 2D case.

In both vectorization techniques for the ILU preconditioner, artificial modifications in programs are necessary.

Several drawbacks for the ILU or TF preconditioner vectorizations are listed below:

1) In the diagonalwise vectorization technique (for both 2D and 3D ILUs), the vector length changes from 1 to min (n_x, n_y), and the average vector length is about half of that for the 2D TF preconditioning technique.

2) In the hyperplane vectorization technique for the 3D ILU, the indirect memory access, which generally degrades memory access performance, is required.

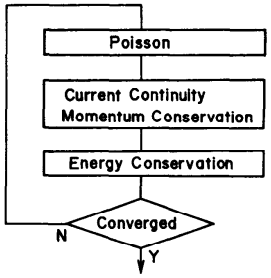

Fig. 7 Decoupled iteration for device simulation.

3) In the 3D TF preconditioner vectorization technique, y factor inversion has vector length of a single dimensional order. (orthers have two dimensional order.)

The TF preconditioners have several advantages when used on multi-processing systems. Each factor inversion can be divided into independent sub-tasks. Hence, they will be well handled on multi-processing systems, including SIMD machines with common memory units. On the other hand, it is impossible to divide the ILU preconditioner inversion task to independent sub-tasks, as shown in Fig. 6.

5. Numerical Examples

Convergence and computational speed of the TF preconditioner and the ILU preconditioner have been compared on 2- and 3-dimensional device simulation problems.

5.1 2-Dimensional Examples

Numerical examples are 2-dimensional nonsymmetric linear systems arising from the carrier continuity equations and the energy conservation equations for 2-dimensional MOSFET device simulation with energy transport phenomena[12]. Equations of the simulation model are listed in Appendix A. The decoupled method is used, in which each equation is solved independently and iteratively until the solution reaches a self-consistent state (Fig. 7). Equations arising at an initial step of the Decoupled iteration is, in general, ill-conditioned, and hence requires more iterations for linear equation solving.

The Bi-conjugate Gradient (BCG) method and the Conjugate Gradient Squared (CGS) method are used as the basic iterative procedure. The BCG basic iterative procedure for Eq. (2), where M is replaced by M_{TF}, for example, results in the TFBCG algorithm. The most time consuming and hence important part in vector and/or parallel computation is the preconditioner inversion process. The TF preconditioner inversion vectorization is explained in section 4. The diagonalwise vectorization technique, which is also explained in section 4, is used for the ILU preconditioner inversion vectorization.

Table 2 Numerical results for 2D device simulation with high gate-voltage. (Problem 1; $n_x*n_y=50*50$)
(a) The number of iterations.

Equation	Iter. Step	BCG		CGS	
		TFBCG	ILUBCG	TFCGS	ILUCGS
Carrier Continu. Eq.	Initial	170	133	114	82
	Final	105	91	59	49
Energy Conserv. Eq.	Initial	8	8	5	5
	Final	9	5	6	3

(b) CPU-time. ($\times 10^{-3}$ sec)

Equation	Iter. Step	BCG		CGS	
		TFBCG	ILUBCG	TFCGS	ILUCGS
Carrier Continu. Eq.	Initial	87.7	151.2	56.1	93.2
	Final	50.4	103.8	28.2	56.1
Energy Conserv. Eq.	Initial	4.1	9.6	2.8	6.1
	Final	4.5	6.1	3.2	3.8
Total		146.7	270.7	90.3	159.2
Ratio		1.62	3.00	1.00	1.76

Table 3 Numerical results for 2D device simulation with low gate-voltage. (Problem 2; $n_x*n_y=50*50$).
(a) The number of iterations.

Equation	Iter. Step	BCG		CGS	
		TFBCG	ILUBCG	TFCGS	ILUCGS
Carrier Continu. Eq.	Initial	158	102	68	53
	Final	103	73	70	46
Energy Conserv. Eq.	Initial	12	8	9	6
	Final	12	8	7	4

(b) CPU-time. ($\times 10^{-3}$ sec)

Equation	Iter. Step	BCG		CGS	
		TFBCG	ILUBCG	TFCGS	ILUCGS
Carrier Continu. Eq.	Initial	76.3	88.7	32.6	47.0
	Final	47.2	64.0	33.4	40.7
Energy Conserv. Eq.	Initial	6.0	7.3	4.6	5.6
	Final	5.9	7.3	3.7	3.9
Total		135.4	167.3	74.3	97.2
Ratio		1.82	2.25	1.00	1.31

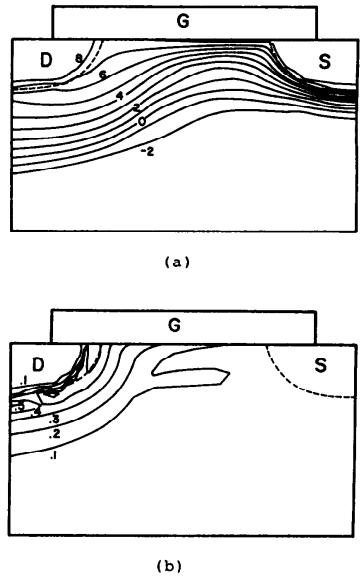

Fig. 8 Typical simulation results. (a) 2D potential distribution. (b) 2D energy distribution. nMOS, $V_D=V_G=2$ V, $L_{eff}=0.25$ μm, $T_{ox}=100$ A, $X_j=0.1$ μm.

Table 2 and 3 show numerical test results. Each corresponds to different test conditions (see Appendix A, Problem 1). The higher gate-voltage condition in Problem 1 leads to greater ill-conditioning than in Problem 2. This reflects in the number of iterations. The number of gridpoints is 50*50. Figure 8 displays a typical example of the solution. Table 2.a and 3.a show the number of iterations required to achieve

$$\|Au-f\|_2/\|f\|_2<10^{-12} \tag{20}$$

where $\|\cdot\|_2$ denotes the L_2-norm of a vector. Table 2.b and 3.b show the CPU-time measured on the SX-2 supercomputer. The following are concluded from these test results:

1) The CGS procedure requires less iterations than the BCG procedure for both preconditioners.

2) The TF preconditioner requires 20 to 50% more iterations for the carrier continuity equation, as well as up to 100% more iterations for the energy conservation equation than the ILU preconditioner.

3) The average CPU-time for one iteration step is about 0.5 msec for the TFBCG and TFCGS programs and about 1.1 msec for the ILUBCG and ILUCGS programs. Iteration procedures with the TF proconditioner are about twice as fast as those with the ILU. (The reasons are discussed in section 4.)

4) As a result, the TF preconditioner requires 20 to 50% less CPU-time for the carrier equation, and up to 60% less for the energy equation than the ILU preconditioner.

5.2 3-Dimensional Examples

Numerical examples are 3-dimensional nonsymmetric

Table 4 Numerical results for 3D device simulation. (Problem 3; $n_x * n_y * n_z = 51 * 51 * 21$)

METHOD	Drive Vortage (V)				
	0	25	50	75	100
ILUBCG Number of Iteration	12	12	12	12	12
CPU-time (ratio) Diagonal-wise	.422 (2.65)	.422 (2.51)	.422 (2.37)	.422 (2.37)	.422 (2.26)
Hyperplane	.320 (2.01)	.320 (1.90)	.320 (1.80)	.320 (1.80)	.320 (1.71)
TFBCG Number of Iteration	15	16	17	17	18
CPU-time (ratio)	.159 (1.00)	.168 (1.00)	.178 (1.00)	.178 (1.00)	.187 (1.00)

Table 5 Numerical results for 3D device simulation. (Problem 4; $n_x * n_y * n_z = 201 * 21 * 21$)

METHOD	Drive Vortage (V)				
	0	25	50	75	100
ILUBCG Number of Iteration	9	9	9	9	9
CPU-time (ratio) Diagonal-wise	.517 (3.29)	.517 (3.04)	.517 (2.84)	.517 (2.64)	.517 (2.64)
Hyperplane	.501 (3.19)	.501 (2.95)	.501 (2.75)	.501 (2.58)	.501 (2.58)
TFBCG Number of Iteration	11	12	13	14	14
CPU-time (ratio)	.157 (1.00)	.170 (1.00)	.182 (1.00)	.194 (1.00)	.194 (1.00)

linear systems arising from a carrier diffusion equation for opto-electronics device simulation[13]. The equation of the simulation model is given in Appendix *B*. Two different numbers of gridpoints corresponding to two different device shapes, are used as examples; one is 201*21*21 (Problem 3) and the other is 51*51*21 (Problem 4). Convergence and speed were evaluated against the change in the advection intensity, which corresponds to the drive potential for the devices. The BCG method is used as a basic iterative procedure.

The ILUBCG algorithm was coded in two ways, as introduced in section 4. One uses the diagonalwise vectorization technique, and the other uses the hyperplane vectorization technique with list-vectors. Iterations are terminated when a relative L_2-norm of the residual vector, defined by Eq. (20), reaches 10^{-12}.

Numerical results are shown in Table 4 and 5. From these tables, it can be seen that:

1) According to the ILU preconditioner vectorization, the hyperplane technique is faster than the diagonalwise vectorization. However, the difference is not great.

2) The TF preconditioner requires about 20 to 50% more iterations than the ILU preconditioner.

3) CPU-time required for one iteration step for the TFBCG program is 1/3 to 1/4 of that for the vectorized ILUBCG programs.

4) As a result, the TFBCG program is about 2 or 3 times faster than the vectorized ILUBCG programs.

5) In Problem 4, 450 MFLOPS were obtained by the TFBCG program.

It is a well-known fact that the Gustafsson's modification version of the ILU preconditioner for a certain class of problems reduces the number of iterations. However, it does not improve convergence for problems used here[14],[15]. (The same tendencies are often reported in other device simulation problems.)

6. Conclusion

This paper introduced the Preconditioner Introduction Process, which gives an alternative way of introducing both the TF and ILU preconditioners. This process also gives a way of introducing the Gustafsson's modification version of the ILU[10] and TF[16] preconditioners. The Gustafsson's modification for the TF preconditioner should be studied in more detail.

The approximation properties of the TF and ILU preconditioners against the coefficient matrix of linear systems were compared to show that they have similar approximation properties versus the change in diffusion anisotropy and advection intensity of P.D.E.s Convergence and CPU-time were compared for both preconditioners for practical 2D and 3D device simulation problems on the NEC SX-2 supercomputer. Although the TFBCG and TFCGS programs, respectively, require more iterations than the ILUBCG and ILUCGS programs, they are 20% to 3 times faster than the vectorized ILUBCG and ILUCGS programs. The TF preconditioner has several advantages when used on multi-processing systems. Performance comparison of both preconditioners for these systems is an interesting and important task.

The ADI (Alternate Direction Implicit) method includes parallelism similar to the TF preconditioner. It is also possible to apply the ADI method as a preconditioner[11]. Performance comparison of these two preconditioners is also important and is the subject of further study by the authors.

The BCG and CGS iterative procedures with both the TF and diagonalwise ILU preconditioners for the 2D 5-point and 3D 7-point differencing are installed in the ASL/SX (A Scientific Library for SX supercomputers).

Acknowledgment

The authors would like to thank Mr. M. Fukuma of Micro Electronics Research Labs., NEC Corp., for providing the Appendix A problem, and Mr. N. Oda of Material Development Center, NEC Corp., for providing the Appendix B problem. The authors also would like to thank Prof. B. Parlett of University of California, Berkeley, and Prof. G. Golub of Stanford University for suggesting some studies on the ADI methods used as preconditioners.

References

1. Fletcher, R. Conjugate Gradient Methods for Indefinite Systems, *Proc. of the Dundee Biennial Conf. on Num. Anal., Springer-verlag* (1975), 73–89.
2. Heijer, C. den Iterative Methods for Nonsymmetric Linear Systems, *Proc. of Int. Conf. on Simulation of Semiconductor Devices and Processes*, SWANSEA (1984), 267–285.
3. Sadd, Y. and Schults, M. H. Conjugate Gradient-Like Algorithms for Solving Nonsymmetric Linear Systems, *Math. Comp.* **44**, 170 (1985) 417–424.
4. Kershaw, D. S. The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear-Equations, *J. of Comp. Phys.* **26** (1978), 43–65.
5. Hageman, A. and Young, D. Applied Iterative Methods, *Academic Press*, New York (1981).
6. Johnson, O. G. and Paul, G. Vector Algorithms for Elliptic Partial Differential Equations Based on JACOBI Method, *Elliptic Problems Solvers, Academic Press* (1981), 345–351.
7. Vorst, H. A. van der A Vectorizable Variant of Some ICCG Methods, *SIAM J. of Sci. Stat. Comp* **3**, 3 (1982), 350–356.
8. Ushiro, Y. Vector Computer Version of ICCG Method, *Trans. of the Research Institute for Mathematical Science, Kyoto Univ.* **514** (1984), 110–134.
9. Vorst, H. A. van der (M)ICCG for 2D Problems on Vector-computers, Report A-17, Data Processing Center, Kyoto Univ. (1986).
10. Gustafsson, I. A Class of First Order Factorization Methods, *BIT* **18** (1978) 142–156.
11. Chin, R. C. Y., Manteuffel, T. A. and Pillis, J. de ADI as a Preconditioning for Solving the Convection-Diffusion Equation, *SIAM J. of Sci. Stat. Comp.* **5**, 2 (1984), 281–299.
12. Fukuma, M. and Uebbing, R. H. Two-Dimensional MOS-FET Simulation with Energy Transport Phenomena, *Proc. of the IEEE Int. Electron Devices Meeting, IEDM* **84 (1984), 621–624.**
13. Oda, N., Miyamoto, K. and Yamagata, T. High Spatial Resolution HgCdTe Photoconductors with Optical Masks, *Proc. of the 31st SPIE Conf.*, Infrared Technology XIII, 819 (1988), in press.
14. Doi, S. and Harada, N. A Preconditioning Algorithm for Solving Nonsymmetric Linear Systems Suitable for Supercomputers, *Proc. of the 2nd Int. Conf. on Supercomputing* **2** (1987) 503–509.
15. Doi, S. and Harada, N. Comparing Performance of the TF and ILU Preconditioners for Device Simulation Application, *Proc. of the Numerical Analysis Symposium, IPSJ* **21**-1 (1987) (in Japanese).
16. Doi, S. and Harada, N. Preconditioners for Nonsymmetric Systems Suitable for Supercomputers—Comparing TF and ILU Preconditioners—, Advances in Numerical Methods for Large Sparse Sets of Linear Equations, 3, Keio Univ. (1987) 1–8. (in Japanese).

Appendix

Appendix A: 2-dimensional device simulation problem.

Poisson Eq.; $\operatorname{div}(\varepsilon \operatorname{grad} \varphi) = q(n - p + N_A - N_D)$

Carrier Continuity Eq.; $\operatorname{div}(nv) = 0$

Momentum Conservation Eq.; $v = (\tau_p/m)\{qE - (2/3)\operatorname{grad} w - (2w/3n)\operatorname{grad} n\}$

Energy Conservation Eq.; $(5/3)(v \cdot \operatorname{grad} w) = qE \cdot v - (w - w_0)/\tau_w$.

φ: potential, n: electron density, p: positive-hole density, N_A, N_D: the accepter and donner density, v; carrier drift velocity, E: electric field, m: effective mass, w: average energy, w_0: equilibrium state of energy, τ_P, τ_W: relaxation time for momentum and energy.

Model conditions:

n MOS, $L_{\text{eff}} = 0.25\ \mu\text{m}$, $T_{\text{ox}} = 100$ A, $X_i = 0.1\ \mu\text{m}$, $V_D = 2$ V, (for Problem 1 and 2),

$V_G = 2$ V (for Problem 1),

$V_G = 1$ V (for Problem 2).

Discretization conditions:

The Decoupled method is used (see Fig. 7).

The centered and up-wind difference approximations are used for the diffusion and advection terms, respectively.

The Cartesian non-uniform coordinate system is used.

The ratio between the largest mesh and the smallest mesh is about 20 for both directions.

The number of gridpoints is 50*50.

Appendix B: 3-dimensional device simulation problem.

$$\{(1/\tau) - D \operatorname{div} \operatorname{grad} + \mu E \operatorname{grad}\} P = \eta \phi_s / d, \quad (0 < x < l,\ 0 < y < w,\ 0 < z < d),$$

P: carrier density, τ: carrier life time, D: diffusion coefficient, μ: mobility, $E = [E_x, 0, 0]^T$: bias electric field, η: quantum efficiency, ϕ_s: photon flux, d: device thickness.

Boundary conditions:

$$D\{\operatorname{grad} P\}_x = (\mu E_x + S_x')P \text{ at } x = 0,$$

$$D\{\operatorname{grad} P\}_x = (\mu E_x - S_x)P \text{ at } x = l, \text{ etc.}$$

(Received August 21, 1987)