ISE-TR-86-57

AN INCOMPLETE LDU DECOMPOSITION OF LATTICE FERMIONS AND ITS APPLICATION TO CONJUGATE RESIDUAL METHODS

by

Yoshio Oyanagi

June 7, 1986

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

ISE-TR-86-57

An Incomplete LDU Decomposition of Lattice Fermions and its Application to Conjugate Residual Methods

Yoshio OYANAGI
Institute of Information Sciences, University of Tsukuba
Sakura-mura, Niihari-gun, Ibaraki, 305 JAPAN

A class of incomplete LDU decomposition of the Wilson fermion with arbitrary r ($|r| \leq 1$) as well as the Kogut-Susskind fermion is proposed. This decompostion is combined with the conjugate residual method to provide a fast iterative solver. The method is implemented on a vector processor (HITAC S810) in terms of a hyperplane method.

1. Introduction

A time-consuming part of the numerical simulation of the lattice gauge theory with fermions in both quenched and unquenched formulations is the solution of large sets of linear equations,

$$A\,x = b, \tag{1.1}$$

where A is a large sparse complex non-hermitian matrix. For example, in the case of the Wilson fermion on a $16^3 \times 32$ lattice, the order of A is 1572964. The quark propagator on a given gauge configuration is given by the solution of (1.1) where b is the point-like source term. In the Langevin formulation[1] of dynamical quarks, it is necessary to solve (1.1) twice in each iteration where the elements of b are independent gaussian random variables. It is therefore urgent to develop an efficient solver for such very large systems of linear equations.

Since the size of the coefficient matrix A is very large, the amount of work and the storage required in direct methods such as Gauss elimination is nearly prohibitive. One should therefore adopt interative algorithms to solve (1.1). The methods used so far were relaxation-type ones such as Gauss-Seidel, SOR[2] or time-relaxation[3] methods. The speed of convergence, however, is not fast enough for large scale lattices.

The conjugate gradient(CG) method, first proposed by Hestenes and Stiefel[4] is widely used to solve large sparse systems of linear equations where the coefficient matrix is real

symmetric (or complex hermitian) and positive definite. Although A for the lattice fermions is not hermitian, variants of the CG method are applicable to (1.1) where the coefficient matrix is A^+A[4] or AA^+[5] instead of A. Possible algorithms are described in the Appendix. An important feature of the CG method that makes it particularly suitable for the lattice fermions is that all reference to A is in the form of the multiplication of a vector by A or A^+, so that no explicit form of A is necessary.

The CG method gives the exact solution in finite steps if the round-off error is absent. In the actual application, however, it is regarded as an iterative decent procedure based on the conjugate directions and the iteration is terminated in relatively small number of steps. Recently, a class of different decent methods are proposed for nonsymmetric systems[6-8], which are based on minimizing $||b-Ax||$ over the affine space $x_i + \langle p_i, p_{i+1}, \cdots, p_{i+k} \rangle$. We refer to them as conjugate residual (CR) methods in general. These methods can be considerably improved by the use of preconditioning and are in general faster than the CG method.

In this paper we present a fast solver based on the CR method with preconditioning. In the next section we introduce an incomplete LU (or LDU) factorization originally proposed by Meijerink and van der Vorst[9] adapted for the lattice fermions. The CR methods and their acceleration are discussed in section 3. Section 4 contains the comparison of algorithms and the details for implementing the algorithm on a vector processor.

2. Incomplete LDU Decomposition

We will describe the fermion matrix A as a block matrix $\{A_{ij}\}$, where i and j represent generic lattice sites in a four-dimentional hypercubic lattice with the periodic boundary condition (Antiperiodic boundary conditions for the fermions can be easily incorporated). If the gauge group is SU(3), each block A_{ij} is a matrix of size 12x12 for the Wilson fermion and 3x3 for the Kogut-Susskind fermion. The generalization to other groups is straightforward. The striking feature of the fermion matrix is, like the matrices generated by discretization of elliptic or parabolic differential equations, that an off-diagonal block A_{ij} is nonzero only when i and j are adjacent.

2.1 Definition

An incomplete block LDU decomposition of A is written as

$$A = L\,D\,R - N \tag{2.1}$$

where $L=\{L_{ij}\}$ is a block lower (or left) triangular matrix, $D=\{D_i\}$ is a block diagonal matrix and $R=\{R_{ij}\}$ is a block upper (or right) triangular matrix. We use R instead of U, since we will reserve the symbol U for the gauge field.

In order to define "lower" or "upper" matrix, we have to introduce an order in the set of lattice sites. The ordering of the lattice sites is arbitrary in principle. We will adopt a natural numbering, i.e.,

$$i = (((i_t-1)n_z+i_z-1)n_y+i_y-1)n_x+i_x\ , \tag{2.2}$$

where n_x, n_y, n_z and n_t are the linear extensions of the lattice in the four directions and i_x, i_y, i_z and i_t are the coordinates in the lattice (e.g. $i_x=1,2, .. n_x$). The site number i runs from 1 to n, where $n=n_x n_y n_z n_t$ is the total number of the lattice sites.

The LDU decomposition (2.1) has some arbitrariness, which is reduced by imposing the relation

$$L_{ii} = R_{ii} = D_i^{-1}. \tag{2.3}$$

It will be clear later that this choice is more appropriate than the usual convention $L_{ii} = R_{ii} = I$.

The matrix N in (2.1) is the error of decomposition. In applying the decomposition to iterative methods as a preconditioner, the more the LDR resembles A, the faster the method will converge. On the other hand, we have to solve the equation

$$L\ D\ R\ s = t \tag{2.4}$$

during every iteration, so that the number of non-zero blocks in L and R should not be too large.

Various incomplete LDU decompositions arise by choosing the the set P of the off-diagonal blocks (i,j) for which $L_{ij} = R_{ij} = 0$. The usual incomplete LU decomposition[9] requires

$$N_{ij} = 0 \quad \text{if } (i,j) \in P, \tag{2.5}$$

which imposes n^2-p matrix equations, where p is the number of elements in P. Since the number of independent unknown blocks in L, D and R is also n^2-p, the decomposition is obtained via Gauss elimination provided the diagonal blocks do not become singular during the process. The existence and uniqueness of the incomplete decomposition are proved[9] for real M-matrix. The fermion matrix we are considering, however, is complex, so that the general proof is not applicable. We will show the existence of the decomposition case by case.

Since the fermion matrix A connects only adjacent lattice sites, the most natural choice of P would be

$$P = \{(i,j) | i \neq j, \text{ and } i \text{ and } j \text{ are } \underline{\text{not}} \text{ adjacent}\}, \tag{2.6}$$

which requires that the relation

$$(L\ D\ R)_{ij} = A_{ij} \tag{2.7}$$

should hold for i=j and adjacent (i,j) pairs.

When the linear extensions of the lattice are all greater than three, the set P has a special property that

$$(i,j) \notin P \text{ and } (i,k) \notin P \longrightarrow (k,j) \in P \tag{2.8}$$

for any three sites i, j and k different with each other. In other words any three sites cannot be mutually adjacent at the

same time. This property holds only for n_x, n_y, n_z, $n_t > 3$. In case $n_x = 3$ for example, three sites with the same i_y, i_z and i_t are mutually adjacent due to the periodic boundary condition.

2.2 Wilson Fermions (n_x, n_y, n_z, $n_t \geq 4$)

The Wilson fermion matrix[10] is given in terms of the Dirac matrices γ_μ (4x4 complex) and the gauge field U_{ij} (a unitary unimodular matrix of order 3),

$$\begin{aligned} A_{ii} &= I \\ A_{ij} &= -K\,(r-\gamma_\mu)\,U_{ij} \qquad \text{if } j=i+\hat{\mu} \\ A_{ij} &= -K\,(r+\gamma_\mu)\,U_{ij} \qquad \text{if } j=i-\hat{\mu} \\ A_{ij} &= 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{aligned} \tag{2.9}$$

Here $j=i\pm\hat{\mu}$ means that the site j lies next to the site i in the positive (negative) μ-direction. K is the hopping parameter. The parameter r, which satisfies $|r| \leq 1$, is called the Wilson parameter. In the usual formulation r is fixed to unity, but sometimes r may be different site by site.

For adjacent (i,j) with $i < j$, (2.7) reads

$$A_{ij} = \sum_{k=1}^{i-1} L_{ik}\, D_k\, R_{kj} + L_{ii}\, D_i\, R_{ij} \tag{2.10}$$

Due to the property (2.8) the first term in (2.10) vanishes. Using (2.3) we simply have,

$$A_{ij} = R_{ij}. \tag{2.11}$$

In the same manner, we have for adjacent (i,j) with $i > j$,

$$A_{ij} = L_{ij} \tag{2.12}$$

For $i = j$, eq.(2.7) requires

$$A_{ii} = \sum_{k=1}^{i-1} L_{ik} D_k R_{ki} + L_{ii} D_i R_{ii}, \tag{2.13}$$

that is

$$L_{ii} = R_{ii} = D_i^{-1} = I - \sum_{j=1}^{i-1} A_{ij} D_j A_{ji}. \tag{2.14}$$

Theorem 2.1

L_{ii} $(=R_{ii})$ is a constant multiple of I (unit matrix) and the constant is equal to or greater than unity.

Proof

The proof is given by induction on i. For i=1, (2.14) reads

$$L_{ii} = I \tag{2.15}$$

Assume $L_{jj} = c_j I$ $(c_j \geq 1)$ holds for $j < i$. Then we have

$$L_{ii} = I - \sum_{j=1}^{i-1}{}' K^2 (r_j^2 - 1)/c_j \; I, \tag{2.16}$$

using the property

$$U_{ij}\, U_{ji} = I. \tag{2.17}$$

The summation with prime means to sum only over j's which are adjacent to i.

Since $|r_j| \leq 1$,

$$c_i = 1 + K^2 \sum{}' (1 - r_j{}^2)/c_j \geq 1 \tag{2.18}$$

Hense the theorem holds for i. **Q.E.D.**

This theorem implies that the Gauss elimination procedure (2.16) is numerically stable. To implement the LDU decomposition, one has to calculate c_i in terms of (2.18) once before the iteration. If r=1, however, the decomposition is quite simple due to the following collorary.

Collorary 2.2

For the usual Wilson fermion with r = 1,

$$L_{ii} = R_{ii} = D_i = I. \tag{2.19}$$

We will consider here the error of the LDU decomposition N defined by (2.1). The matrix product LDR has nonzero blocks $(LDR)_{ij}$ of order $O(K^2)$ for

$$j = i + \hat{\mu} - \hat{\nu} \quad (\mu \neq \nu). \tag{2.20}$$

Since the two sites i and j in (2.20) are not adjacent, N_{ij} is nonzero.

2.3 Wilson fermions ($n_\mu = 3$)

If the linear extensions in some of the directions are equal to three, the property (2.6) no longer holds, so that eqs. (2.11) and (2.12) are not valid. One could in principle perform a Gauss elimination procedure using (2.3), (2.10) and (2.13). The resultant blocks L_{ij} and R_{ij} would not be equal to A_{ij} and the diagonal blocks D_i would neither be proportional to I. The beautiful features of the LDU decomposition would be lost.

Instead, we propose to apply formally (2.11), (2.12) and (2.16) to this case. Since the property (2.6) does not hold, the incomplete decomposition (2.1) has some error even for adjacent (i,j) pairs. For example if $n_x=3$ and n_y, n_z, $n_t > 3$,

$$\begin{aligned}(L\ D\ R)_{23} &= A_{21}D_1A_{13} + A_{23} \\ &= A_{23} + K^2(r^2-1)U_{21}U_{13}c_1 \end{aligned} \tag{2.21}$$

that is

$$N_{23} = O(K^2), \tag{2.22}$$

where 1,2 and 3 denote the three sites with $i_x=1,2$ and 3 and the same i_y, i_z and i_t. We note that the error N is of order K^2 as a whole.

In the usual Wilson fermion with r=1, the second term in (2.21) vanishes, so that (2.5) and (2.7) also holds in this case.

2.4 Kogut-Susskind fermion

We will define the Kogut-Susskind fermion [11] in terms of the following blocks of size 3x3,

$$
\begin{aligned}
A_{ii} &= m\, I \\
A_{ij} &= \pm\tfrac{1}{2}U_{ij} && \text{if } j=i\pm\hat{x} \\
A_{ij} &= \pm\tfrac{1}{2}(-1)^{i_x} U_{ij} && \text{if } j=i\pm\hat{y} \\
A_{ij} &= \pm\tfrac{1}{2}(-1)^{i_x+i_y} U_{ij} && \text{if } j=i\pm\hat{z} \\
A_{ij} &= \pm\tfrac{1}{2}(-1)^{i_x+i_y+i_z} U_{ij} && \text{if } j=i\pm\hat{t} \\
A_{ij} &= 0 && \text{otherwise}
\end{aligned}
\tag{2.23}
$$

where m is the quark mass. We assume the linear extentions of the lattice are greater than three. Due to the property (2.8), we have for the off-diagonal blocks,

$$
\begin{aligned}
R_{ij} &= A_{ij} && (i < j) \\
L_{ij} &= A_{ij} && (i > j)
\end{aligned}
\tag{2.24}
$$

The diagonal blocks are obtained iteratively as

$$
L_{ii} = R_{ii} = D_i^{-1} = mI - \sum_{j=1}^{i-1} A_{ij} D_j A_{ji} \tag{2.25}
$$

We will prove the following theorem.

Theorem 2.3

$L_{ii}(=R_{ii})$ is a constant multiple of I and the constant is equal to or greater than m.

Proof

The proof is given by induction on i. For i=1, (2.25)

implies

$$L_{11} = m\, I. \tag{2.26}$$

If $L_{jj}=c_j I$ $(c_j \geq m)$ holds for $j < i$, then we have from (2.23)

$$L_{ii} = m\, I - \sum_{j=1}^{i-1}{}' \left(-\frac{1}{4}\right) c_j^{-1}\, I, \tag{2.27}$$

that is

$$c_i = m + \sum_{j=1}^{i-1}{}' \frac{1}{4} c_j^{-1} > m. \tag{2.28}$$

This completes the induction. **Q.E.D.**

3. Conjugate Residual Method

A class of iterative methods for solving the system of linear equations (1.1) by decreasing the residual has been proposed [6-8]. They have the following general form

$$\begin{aligned} &r = b - A\,x; \quad p = r \\ &\underline{\text{repeat until convergence}} \\ &\quad \alpha = (r,\ A\,p)/(A\,p,\ A\,p) \\ &\quad x = x + \alpha p \\ &\quad r = r - \alpha\, A\,p \\ &\quad \text{update } p \end{aligned} \tag{3.1}$$

The coefficient α is so determined as to minimize the Euclidean

norm of the residual $|| r - \alpha A p ||_2$ as a function of α.

3.1 CR(k) methods

The variants differ in the way to compute the new direction vector p. A good choice would not only decrease the residual significantly but also require only a reasonalble amount of storage and computation.

If the sequence of the direction vectors p is made fully orthogonal with respect to A^+A, the algorithm (3.1) give the exact solution to (1.1) in finite steps in the absense of round-off errors. The amount of storage and computation, however, to keep {p} orthogonal is enormous for large scale problems we consider here.

More practical algorithm is to make the new direction vector p orthogonal to only last k (≥ 0) direction vectors[7], namely

$$p = r + \beta_1 p_1 + \beta_2 p_2 + \ldots + \beta_k p_k \tag{3.2}$$

with

$$\beta_j = -(A r, A p_j)/(A p_j, A p_j) \quad (j=1,2,..,k) \tag{3.3}$$

where p_1, p_2, .., p_k are the last k direction vectors. Instead of multiplying p by A in (3.1), we may update Ap by

$$Ap = Ar + \beta_1 Ap_1 + \beta_2 Ap_2 + \ldots + \beta_k Ap_k. \tag{3.4}$$

The storage required to implement this algorithm is the space for

the (3+2k) vectors: x, r, Ar, $\{p_j\}$ and $\{Ap_j\}$. We excluded the storage for A since it can be constructed in terms of the gauge field {U}.

For the special case k=0, (3.2) is reduced to

$$p = r \tag{3.5}$$

and only three vectors x, r and Ar have to be stored. This method is also called minimal residual (MR) method.

3.2 Convergence of the CR(k) methods

The algorithm (3.1),(3.2) converges to the solution for (1.1), if $H = (A^{+}+A)/2$, the hermitian part of A, is positive definite, due to the theorem of Eisenstat et al.[8]

Theorem 3.1

If $\{r_i\}$ is the sequence of residuals generated by CR(k) ($k \geq 0$) and H is positive definite, then we have

$$\frac{\|r_{i+1}\|^2}{\|r_i\|^2} \leq 1 - \frac{\lambda_{min}(H)^2}{\lambda_{max}(A^{+}A)} \tag{3.6}$$

where λ_{min} and λ_{max} denote the minimum and maximum eigenvalues.

The proof is given in ref.[8]. This bound (3.6) is not very stringent since it does not incorporate the effect of orthogonalization (3.2), but it certifies the convergence of the algorithm.

The positivity of H is crucial in the theorem. If the positivity is lost, the coefficient α becomes zero or very small

and the residual no longer decreases.

3.3 Preconditioning

In practice, the convergence has to be accelerated by the use of preconditioning. We will apply the incomplete LU or LDU decomposition described in the preceeding section as a preconditioner. The original equation (1.1) is now replaced by

$$(L\ D\ R)^{-1}\ A\ x = (L\ D\ R)^{-1}\ b \tag{3.7}$$

which is solved by (3.1). We will refer to the CR(k) method with the incomplete LDU preconditioning as ILUCR(k) or ILUMR (for k=0) [12].

For example the algorithm of ILUCR(1) will be as follows:

$$\begin{aligned}
&r = (L\ D\ R)^{-1}(b - A\ x)\\
&p = r\\
&q = (L\ D\ R)^{-1}A\ p\\
&\underline{\text{repeat until convergence}}\\
&\quad \alpha = (q,\ r)/(q,\ q)\\
&\quad x = x + \alpha\ p\\
&\quad r = r - \alpha\ q\\
&\quad s = (L\ D\ R)^{-1}A\ r\\
&\quad \beta = -(q,\ s)/(q,\ q)\\
&\quad p = r + \beta\ p\\
&\quad q = s + \beta\ q
\end{aligned} \tag{3.8}$$

We note the storage requirement is the same for ILUCR(k) as that

for CR(k), since the operation such as

$$s = (L\ D\ R)^{-1}\ A\ r \tag{3.9}$$

can be performed without any extra working vectors.

3.3 Acceleration

The ILUCR(k) method has no adjustable parameter such as the ω-parameter in the succesive overrelaxation (SOR) method. We found, however, considerable acceleration using a kind of freedom to replace the hopping parameter K by cK in the incomplete LDU decomposition, where c is a constant factor. Since the preconditioner is in principle arbitrary, the hopping parameter in LDU is not necessarily the same as that in the original fermion matrix A. If c is greater than unity, the off-diagonal elements are multiplied by c. The effect of c and its tuning is discussed in the following section.

4. Implementation

In the previous sections, we have presented a new iterative solver, which is based on an incomplete LDU decomposition and conjugate residual method. In this section we will discuss the details for implementing the algorithm on vector computers. We are mainly concerned about the usual Wilson fermion with r=1, where D is a unit matrix. Generalization to other cases will be

straightforward.

4.1 Convergence Criterion

In implementing the algorithm (3.1) and (3.8), we have to put a convergence criterion. Two kinds of criterion are adopted. One is the norm of the residual $||r|| = ||b-Ax||$. The iteration is terminated if $||r|| < \varepsilon$ or $||r|| < \varepsilon||b||$. Since the theorem 3.1 assures that $||r||$ decreases monotonically, it is an unambiguous way to detect the convergence. In the case of the CR methods with preconditioning, however, r is not the residual b-Ax, but a modifined one $(LR)^{-1}(b-Ax)$. If LR is a good approximation to A, r is approximately equal to the deviation of x from the true solution $A^{-1}b$, since $r=(LR)^{-1}b - (LR)^{-1}Ax \simeq A^{-1}b - x$.

The other criterion is the change Δx of the elements of x between two consecutive iterations. We terminate the iteration if $||\Delta x|| < \varepsilon'||x||$. In spite of the desirable property that the change in x is closely related to the error of x, the problem remains which elements to monitor, since it is not practical to monitor all elements of x.

In practical applications one of the two or both is adopted to determine when to terminate the iteration.

4.2 Acceleration

The acceleration similar to SOR introduced in 3.3 was found very effective in both ILUCR(1) and ILUMR. As an example, the dependency of the number of iterations on the acceleration parameter c is shown in Fig. 1. The gauge configurations are taken from the Langevin simulation of QCD on a $9^3 \times 18$ lattice with

dynamical Wilson fermions (r=1) at β =5.5 and K=0.162[13]. The number of elements in x is 157461. The right hand side is a complex gaussian random noise with unit variance and the convergence criterion is $||r|| < 1$. The initial value of x is set to be equal to the right hand side.

We find the choice c=1.2 is the optimum. The average reduction at c=1.2 is about 25% as compared to c=1. Although the reduction is less significant for smaller hopping parameter K, the optimal value of c does not depend on K. Unlike the SOR method, the number of iterations is not very sensitive to c.

We note that the convergence criterion depends on c, because r is a "modified" residual. In our algorithm, the true residual (b-Ax) cannot be obtained. Since the off-diagonal elements of L and R are multiplied by c, the modified residual $r=(LR)^{-1}(b-Ax)$ increases as c becomes large for a fixed x. The criteiron requires more iterations for larger c.

The theoretical foundation for the acceleration is yet unknown. It may effectively reduce the error N of the incomplete LU decomposition (2.1), namely the non-zero blocks N_{ij} may be in a sense cancelled by increasing the off-diagonal blocks. We expected that such cancellation will become more strinking for smoother gauge. As an example, we tested a gauge fixing to A_4 = 0. After the gauge fixing, however, the number of iterations for the optimal c does not decrease or even increases as compared to the unfixed gauge.

4.3 Hyperplane Method

The computer-time consuming step in the ILUCR method is the

calculation of $(LR)^{-1}Ar$. Since the number of elements are quite large, the amount of computations makes it worth-while to investigate the ways for vectorization of the program. A necessary condition for vectorizability is that in a given loop the computations pertaining to different lattice sites are independent of each other. In other words, the computations belonging to a site should not refer to the results of the computation belonging to another sites in the same loop.

The multiplication of a vector by a matrix t=Ar, namely

$$t_i = \sum_{j=1}^{n}{}' A_{ij} r_j , \tag{4.1}$$

can be easily vectorized, since the calculation for different i's is independent of each other. The vector length in (4.1) is the number of total lattice sites, n.

On the other hand, $s=(LR)^{-1}t$ is not vectorizable as it is, since it is obtained by applying the forward and backward substitutions:

$$\underline{\text{do } i=1,n}$$
$$z_i = t_i - \sum_{j=1}^{i-1}{}' L_{ij} z_j \tag{4.2}$$

$$\underline{\text{do } i=n,1,-1}$$
$$s_i = z_i - \sum_{j=i+1}^{n}{}' R_{ij} s_j , \tag{4.3}$$

In this case, the calculation for different i's is not independent with each other, since new values of z or s are referred to in (4.2) and (4.3). The checkerboard or red-black ordering, which is often used in the vectorization of SOR or

Monte Carlo, does not work in this case, since the substitution is not an iterative updating.

The forward and backward substitutions are vectorized in terms of a hyperplane method proposed by Ushiro et al. in the case of finite difference method for partial differential equation [14]. We find it can be generalized to the lattice fermions. We define the α-th hyperplance h_α as the set of lattice sites whose coordinates satisfy

$$i_x + i_y + i_z + i_t = \alpha \tag{4.4}$$

where α runs from 4 to $n_h = n_x + n_y + n_z + n_t$. All the lattice sites belong to a hyperplane. Since these hyperplanes are oblique, any two sites i and j ($i \neq j$) on a given hyperplane are not adjacent with each other, so that

$$L_{ij} = 0 \quad \text{and} \quad R_{ij} = 0. \tag{4.5}$$

We can therefore vectorize the calculation of z_i and s_i on a single hyperplane. Except at the boundary, the adjacent sites to a site on h_α belong to $h_{\alpha+1}$ or $h_{\alpha-1}$. The average vector length is equal to the average number of sites on a hyperplane, $n_x n_y n_z n_t/(n_h-3)$, e.g. 1700 for $16^3 \times 32$ lattice.

4.4 Fine tuning

The algorithm was mainly tested on HITAC S810/10 at KEK. A standard FORTRAN77 with several compiler directives (*VOPTION) is supported and all the DO-loops in our program have been

vectorized without difficulty. The ratio of the vectorized part in the whole computation is more than 99.9%.

A further speed-up was achieved by a fine tuning of the program. Main improvement was as follows:

a) The IF-statements in the innermost DO-loops are removed. Although they are vectorizable in terms of the masked operation and eight consecutive mask bits are skipped in one machine cycle, they are not fast enough. In the original program the judgement whether the nearest neighbor site $j=i\pm\hat{\mu}$ ($\mu=1,2,3,4$) is forward to i or not was implemented by an IF-statement. We replace it by two index lists in (4.2), which tabulate the site i for which $j=i\pm\hat{\mu} < i$. Similar lists are also used for (4.3).

b) Complex multiplication by 1 or i is removed. Since the nonzero elements of the γ matrices are ± 1 or $\pm i$, the complex arithmetic like GAMMA(ALPHA, MU) * X(BETA, I,...) can be simplified as $\pm$ X(....) or $\pm$CMPLX(-IMAG(X(..),REAL(X(..)) according to the value of GAMMA(ALPHA, MU). Here ALPHA and BETA are Dirac indices (1-4). We can thus reduce four (real) multiplications and two additions.

c) Loop unrolling. Since HITAC S810/10 has two multiplication and four addition pipes, the performance is improved if the six pipes are always active. In addition to this, the store operations from the vector register to the main memory should be minimized. For this purpose we unrolled twofold outer loops of length 3 with respect to the color indices. The number of operations in the inner DO-loop became nine times

larger.

By these improvements the execution time for $s=(LR)^{-1}Ar$ on a 6^3x9 lattice was reduced from 0.375 sec to 0.150 sec (Table 1).

4.5 Comparison of algorithms

We show in Fig. 2 how the error $||x - A^{-1}b||$ decreases in various algorithms. The gauge configuration was taken from a quenched simulation at $\beta=5.5$ on 9^3x18 lattice. The critical value K_c, for which the pion mass vanishes, is 0.1844 ± 0.0009 [13]. The right hand side b is a point source. The CPU time for one iteration is 1.23 sec for ILUCR(1), 1.21 sec for ILUMR, 0.39 sec for CR and 0.66 sec for CG on HITAC S810/10. We present in Table 2 the number of iterations needed to reach $||x - A^{-1}b|| < 10^{-4}$ for ILUCR(1), ILUMR, CR and CG methods. We note that, although ILUCR(1) is faster than ILUMR for c=1.0, the number of iterations for c=1.2 is almost the same in the two algorithms. The CG methods are not practical when K is close to K_c.

5. Conclusion

We have shown that the conjugate residual method proconditioned by the incomplete LDU decomposition is suited for the solution of large sets of linear equations appearing in the lattice gauge theory. The convergence is accelerated by a multiplicative factor for the hopping parameter in the preconditioner $(L\ D\ R)^{-1}$. The program was fully vectorized on HITAC S810/10 vector processor located at KEK and a fine tuning

of the code further increased the performance more than twice.

The algorithm was applied to the Langevin simulation of the lattice QCD on a $9^3 \times 18$ lattice [13].

Acknowledgement

We would like to thank M. Fukugita, A. Ukawa, T. Kaneko, Y. Iwasaki, T. Yoshié and S. Itoh for valuable discussions. We are specially indebted to K. Murata for introducing us the conjugate residual method and to Y. Ushiro for suggesting us the hyperplane method

Appendix Variants of the CG Method

The conjugate gradient method is used to solve a linear equation with positive-definite hermitian coefficient matrix. It is now modified to apply to a linear equation

$$A\,x = b \tag{A.1}$$

where A is a nonhermitian matrix. The algorithms are based on the fact that for arbitrary matrix A, A^+A and AA^+ are always hermitian and positive-(semi)definite. The trick is that we never calculate A^+A or AA^+ explicitly, since it would require a large computer time and memory. In many cases A^+A or AA^+ is less sparse. There are two variants of the CG method.

1. Least Squares-type method

This method is to solve the normal equation

$$A^+ A\,x = A^+ b \tag{A.2}$$

instead of (A.1). If (A.1) has a solution x_0, x_0 also satisfies (A.2). The algorithm is as follows:

$r = b - A\,x$; $p = 0$

<u>repeat until converge</u>

$s = A^+ r$

$\beta = (s, s)^{-1}$

$p = p + \beta\, s$ (A.3)

$$s = A\,p$$

$$\alpha = (s,\ s)^{-1}$$

$$x = x + \alpha\,p$$

$$r = r - \alpha\,s$$

Four working vectors x, r, p and s are needed to implement the algorithm. Three vectors x, r and p are updated recursively, while s is a temporary one. During every iteration, two matrix multiplications A p and A^+ r are performed, which are the most time consuming part of the algorithm.

This method minimizes

$$f(x) = ((x-x_0),\ A^+A(x-x_0)) = (r,\ r) \qquad (A.4)$$

where $x_0 = A^{-1}b$, $r = b - Ax$. If the equation (A.1) is inconsistent, it gives one of the least squares solutions.

2. Least Norm-type method

The other method is based on the solution of

$$A\ A^+\ u = b. \qquad (A.5)$$

When the solution u_0 is obtained, the solution x_0 for (A.1) is given as

$$x_0 = A^+\ u_0. \qquad (A.6)$$

The algorithm is as follows:

$r = b - A x; \quad p = 0$

repeat until converge

$$\begin{aligned} \beta &= (r, r)^{-1} \\ p &= p + \beta A^{+} r \\ \alpha &= (p, p)^{-1} \\ x &= x + \alpha p \\ r &= r - \alpha A p \end{aligned} \tag{A.7}$$

We note the vector u in (A.6) does not appear in (A.7) at all. This algorithm has two advantages over the least-squares-type method. Firstly, only three working vectors x, r and p are necessary in (A.7), while four are needed in (A.3). Secondly, this algorithm minimizes the norm of the <u>error</u>, i.e. the difference between x and x_0:

$$f(x) = ((u-u_0), AA^{+}(u-u_0)) = (x-x_0, x-x_0), \tag{A.8}$$

while (A.3) minimizes the norm of the residual r. The amount of computation in one iteration is almost the same in the two algorithms. The ingeneous trick [15] to calculate Ap and $A^{+}r$ at the same time in (A.3) may also be applicable to (A.7).

References

1. A. Ukawa and M. Fukugita, Phys. Rev. Letters 55 (1985) 1854.
   G. Batrouni et al., Phys. Rev. D32 (1985) 2736.
2. E. Martinari, G. Parisi and C. Rebbi, Phys. Rev. Letters 47 (1981) 1795.
   D. H. Weingarten and D. N. Petcher, Phys. Letters 99B (1981) 333.
3. M. Fukugita, T. Kaneko and A. Ukawa, Nucl. Phys. B230 FS10 (1984) 62.
4. M. R. Hestenes and E. Stiefel, J. Res. Nat. Bur. Standards 49(1952) 409.
5. H. Takahashi and T. Nodera, in Numerical Method for Engineering, eds. E. Absi and R. Glowinski (1980) 209.
6. P. Concus and G. H. Golub, in Lecture Notes in Economics and Mathematical Systems 134, eds. R. Glowinski and J. L. Lions (Springer-Verlag, Berlin 1976) 56.
7. P. K. W. Winsome, in Proc. Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME (1976) 149.
8. S. L. Eisenstat, H. C. Elman and M. H. Schults, SIAM J. Numer. Anal. 20 (1983) 345.
9. J. A. Meijerink and H. Z. van der Vorst, Math. Comp., 31 (1977) 148.
10. K. G. Wilson, in New Phenomena in Subnuclear Physics (Erice 1975), ed. A. Zichichi (Plenum, New York 1977).
11. L. Susskind, Phys. Rev. D16 (1977) 3031.
12. S. Itoh, Y. Iwasaki, Y. Oyanagi and T. Yoshiė, Tsukuba preprint UTHEP-150 (1986).

13. M. Fukugita, Y. Oyanagi and A. Ukawa, Tsukuba preprint UTHEP-152 (1986).
14. Y. Ushiro, M. Nishikata and F. Nagahori, Hitachi Hyoron (in Japanese) 65 (1983) 557.
15. D. Barkai, K. J. M. Moriarty and C. Rebbi, Comput. Phys. Commun. 36 (1985) 1.

Figure Captions

Fig. 1

The number of iterations as a function of the acceleration parameter c for 7 gauge configurations.

Fig. 2

The error $||x-A^{-1}b||$ as a function of the number of iterations k for various algorithms on a quenched gauge configuration at $\beta=5.5$ and K=0.18. The symbols denote: A for ILUMR with c=1.2; B for ILUCR(1) with c=1.2; C for ILUCR(1) with c=1.0; D for ILUMR with c=1.0; E for CR; F for CG (least norm type); G for CG (least square type).

Table Captions

Table 1

Execution time in 64 bit arithmetic for $s=(LR)^{-1}Ar$ on 6^3x9 lattice. The letters a, b and c denote the three kinds of improvement described in the text. MULT and ILU denote the subroutines to compute $t=Ar$ and $s=(LR)^{-1}t$, respectively.

Table 2

The number of iterations to attain $||x-A^{-1}b|| < 10^{-4}$ for various algorithms. The right hand side b is a point source. The initial value of x is set to be equal to b. The number in the parentheses is a rough estimate.

Table 1

version	improvements in ILU	improvements in MULT	time(sec)
original	----	----	0.375
1	a	----	0.220
2	a,b	----	0.192
3	a,c	----	0.190
4	a,b,c	----	0.174
5	a,b,c	c	0.150

Table 2

K=	0.17	0.180	0.181	0.182	0.183	0.184
$m_\pi a$=	0.91	0.49	0.43	0.37	0.27	0.15
ILUCR(1)						
c=1.0	26	90	115	165	275	---
c=1.1	23	74	96	136	224	658
c=1.2	22	70	88	121	200	575
c=1.3	25	88	113	157	236	661
ILUMR						
c=1.0	31	113	149	217	386	---
c=1.1	25	84	109	155	263	749
c=1.2	22	70	89	125	212	581
c=1.3	26	93	119	164	265	786
CR	196	(5000)	(8000)	≫1000	≫1000	
CG(least squares type)	305	(1400)				
CG(least norm type)	299	(1300)				

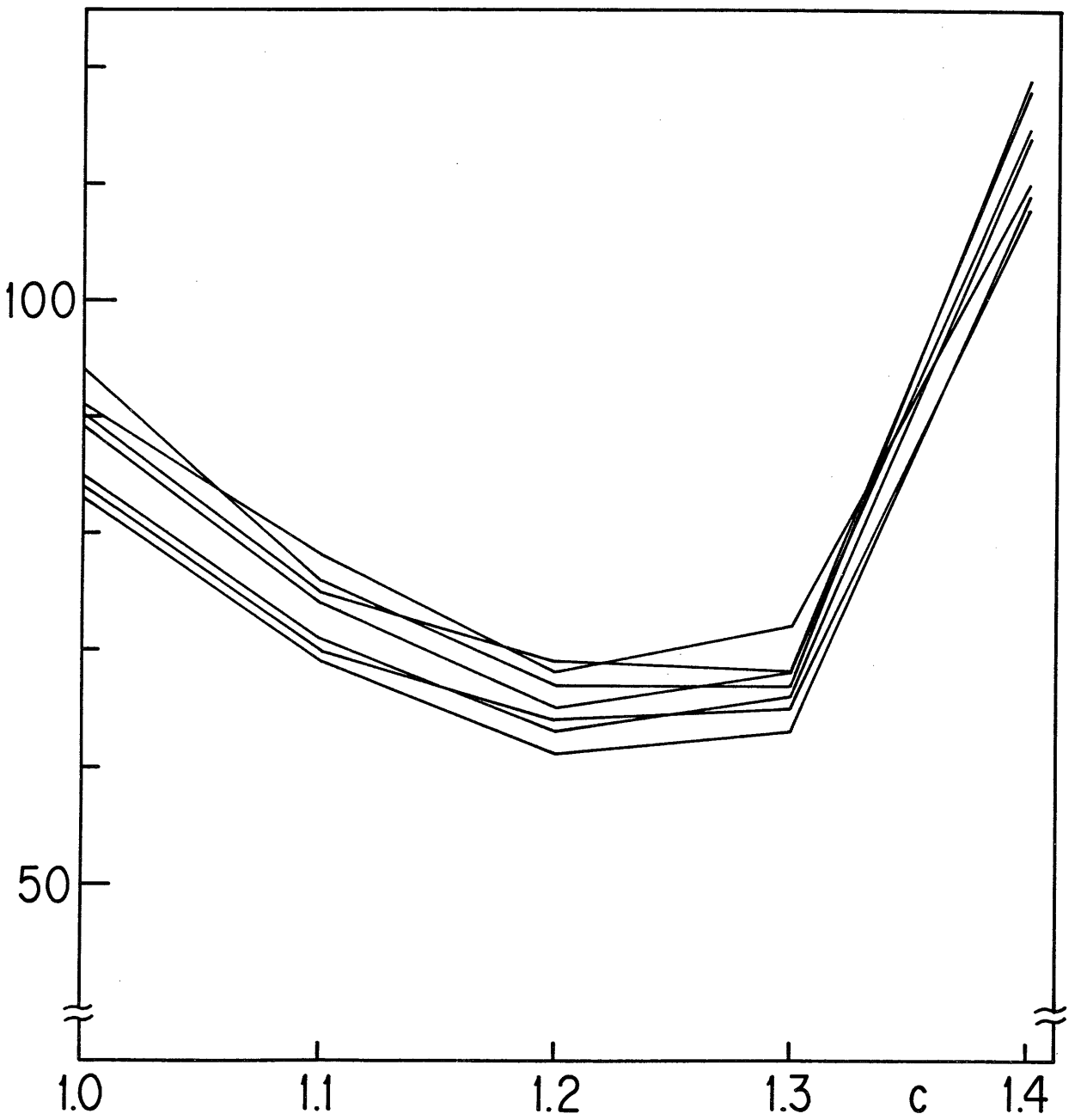

Fig. 1

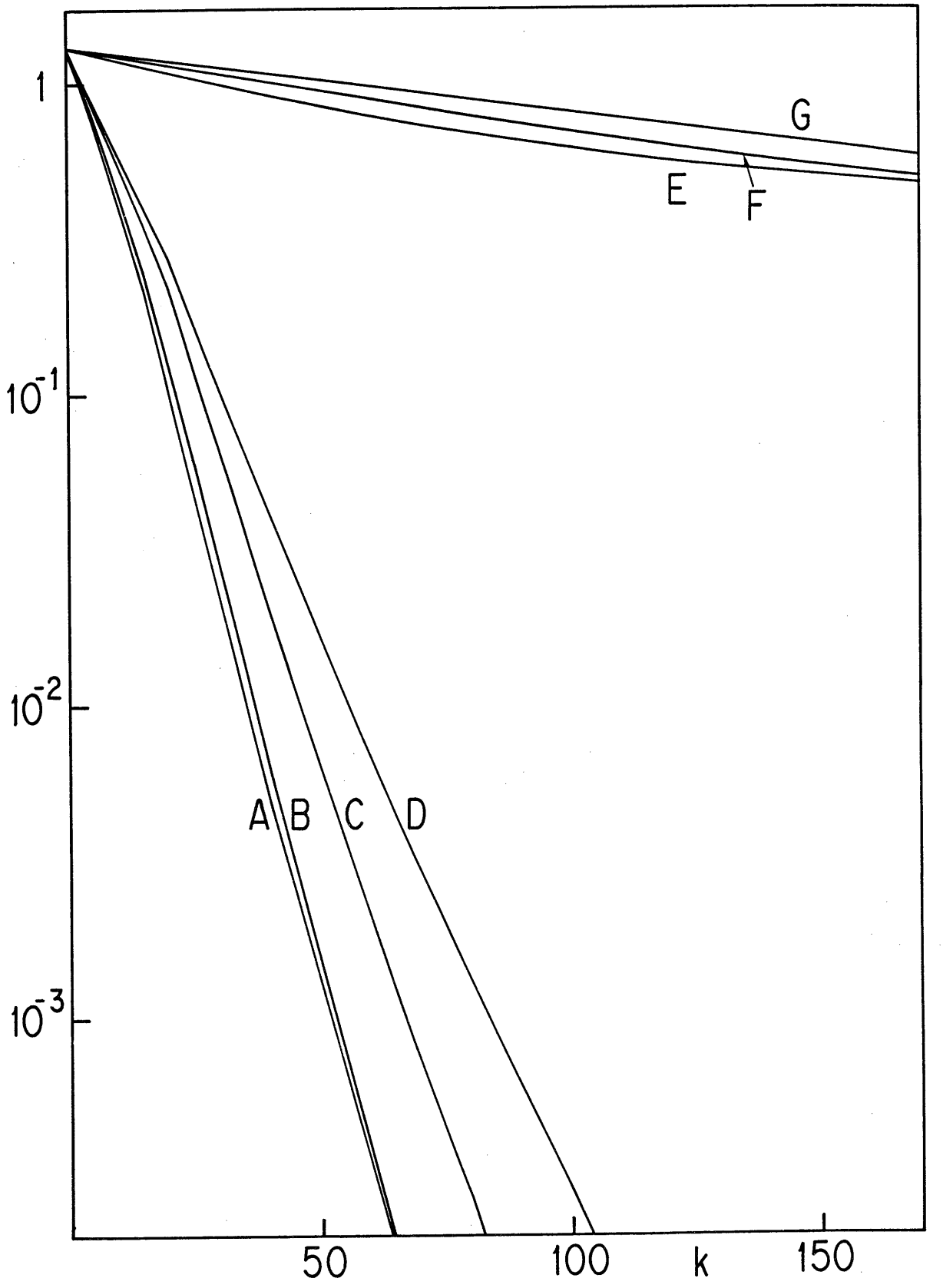

Fig. 2

INSTITUTE OF INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA
SAKURA-MURA, NIIHARI-GUN, IBARAKI 305 JAPAN

REPORT DOCUMENTATION PAGE

REPORT NUMBER
ISE-TR-86-57

TITLE

An Incomplete LDU Decomposition of Lattice Fermions and its Application to Conjugate Residual Methods

AUTHOR(S)

Yoshio OYANAGI

REPORT DATE
June 7, 1986

NUMBER OF PAGES
32

MAIN CATEGORY
linear algebra

CR CATEGORIES
5.14

KEY WORDS
Incomplete LU decomposition, Lattice gauge theory, Wilson fermion, Kogut-Susskind fermion, Conjugate residual method, Hyperplane method

ABSTRACT

A class of incomplete LDU decomposition of the Wilson fermion with arbitrary r ($|r| \leq 1$) as well as the Kogut-Susskind fermion is proposed. This decomposition is combined with the conjugate residual method to provide a fast iterative solver. The method is implemented on a vector processor (HITAC S810) in terms of a hyperplane method.

SUPPLEMENTARY NOTES