CRAY-2에서의 대형희귀행렬 연립방정식의 해법을 위한 벡터준비행렬의 재배열방법

마 상 백†

요 약

이 논문에서 우리는 CRAY-2 에서 편미분방정식에서 발생하는 대형희귀연립방정식의 효과적인 벡터 준비행렬을 만들기 위한 재배열방법을 제시한다. 이 재배열방법은 종래의 빨강/검정 배열의 선형 형태로써, ILU 준비행렬의 변형 형태에 사용될 경우 필인(fill-in)을 크게 하면 종래의 빨강/검정 재배열의 약점이던 수렴율의 감소를 극복할 수 있다. 우리는 CRAY-2 에서 여러 가지 실험을 통해 우리의 주장을 입증한다. 또, 에러 행렬의 후로베니우스 놈을 계산한 결과도 우리의 주장과 일치한다.

A reordering scheme for the vectorizable preconditioner for the large sparse linear systems on the CRAY-2

Sang-back Ma†

ABSTRACT

In this paper we present a reordering scheme that could lead to efficient vectorization of the preconditioners for the large sparse linear systems arising from partial differential equations on the CRAY-2. This reordering scheme is a line version of the conventional red/black ordering. This reordering scheme, coupled with a variant of ILU(Incomplete LU) preconditioning, can overcome the poor rate of convergence of the conventional red/black reordering, if relatively large number of fill-ins were used. We substantiate our claim by conducting various experiments on the CRAY-2 machine. Also, the computation of the Frobenius norm of the error matrices agree with our claim.

1. Introduction

Iterative solutions of large sparse linear systems need the use of preconditioning techniques in order to converge in a reasonable number of iterations. Reordering the equations through multi-coloring provides a parallelism of order N, where N is the dimension of the given matrix. For example, if the matrix has property A[2], as is the case for the standard 5-point matrices obtained from centered Finite Difference(FD) discretizations of elliptic Partial Differential Equations (PDE's), there is a partition of the grid-points in two disjoint subsets such that the unknowns of any one subset are only related to unknowns of the other subset. This enables to produce a reordered matrix having a block-tridiagonal matrix, where the diagonal blocks are diagonal matrices and there are several different ways of exploiting this structure. For example, the unknowns associated with one of the subsets can be easily eliminated, and the resulting reduced system is often well-conditioned. This 'two-coloring' often referred to as a

† 정 회 원 : 한양대학교 공학대학 전자계산학과
논문접수 : 1995년 8월 31일, 심사완료:1995년 11월 2일

red-black or checkerboard ordering, can be generalized to arbitrary sparse matrices by using multi-coloring. But the drawback of this approach is that it often suffers from the deterioration of the rate of convergence with certain iterative methods, such as in the preconditioned CG(Conjugate Gradient) method.[2]

ILU(Incomplete LU) factorization is one of the most powerful preconditioners. It is simple to implement and cost-effective, but it is inherently serial, since it involves forward /backward substitutions. Hence, there has been intensive research on the vectorization/parallelization of this ILU type preconditioner.

Level scheduling technique(or wavefront technique)[11] is an effective algorithm for vectorization of ILU type precoditioners on the vector machine, like CRAY-2. It fully exploits the paralleilism given in the matrix without introducing any change in the adjacency information in the initial ordering, so it preserves the rate of convergence. So, major advantage of level scheduling is that it does not suffer from the deterioration of the rate of convergence as with multi-coloring., but the maximum parallelism is determined by the lengths of the wavefront, which is often nonuniform.

The CRAY-2 is a four-processor machine, each with 8 vector registers of 64 words long. The peak performance of CRAY-2 is two floating-point results per clock period, or 500 Mega Flop/s, giving 2 Giga Flop/s for the four-CPU computer.

Efficient utilization of the CRAY-2 depends heavily on the good vectorization of a given algorithm. In case of preconditioners for the large sparse linear systems we present a combination of ILU type preconditioner and a line red/black reordering. Also, we implemented this algorithm on the CRAY-2 using the level scheduling.

1.1 Discretization matrix

Finite difference method(FDM) or finite element method(FEM) discretizations of the following partial differential equation (PDE)

$$-(K_1(x, y)u_x)_x-(K_2(x, y)u_y)_y+(d(x, y)u)_x+(e(x, y)u)_y+f(x, y)u=g(x, y)$$
$$\Omega=[0, 1]\times[0, 1]$$
$$u=0 \text{ on } \delta\Omega \qquad (1)$$

with meshsize $h=1/(n+1)$ give rise to a linear system

$$Au=b \qquad (2)$$

of order $N=n\times n$, where the the matrix A is a sparse matrix. The matrix A is nonsymmetric due to the presence of the terms of first order derivatives. The problem is called a model problem when the underlying PDE is Laplace equation, i.e, when $K_1=K_2=1$ and $d=e=f=0$. For small h n could become very large. This matrix is very sparse. The number of nonzero entries in a row is fixed independent of the meshsize h. It is 5 for the model problem with the central difference operator, and it is usually less than 10.

Solutions of these sparse linear systems are usually obtained by the iterative methods to exploit the sparsity of the matrix. Krylov subspace methods, such as CG(Conjugate Gradient) method[2] for the positive definite matrices or GMRES(Generalized Minimal Residual)[8] method for the general nonsymmetric matrices are widely used. Since our test problems will be general nonsymmetric, we will use GMRES method as our iterative method in this paper.

1.2 GMRES(m) Method

Given a linear system $Au=b$ of dimension N GMRES(m) method can be described as follows. m is the number of dimensions of the associated Krylov subspace.

Algorithm 1.1 GMRES(m)

1. Choose an initial guess vector u_0 and com-

pute $r_0 = b - Au_0$,

$\beta = \|r_0\|_2$, $v_1 = \frac{r_0}{\beta}$

2. For j=1, , , , , Until Convergence Do
3. $\hat{v}_{j+1} = A v_j$
4. For i =1, ···, j-1 Do
5. $h_{i,j} = (\hat{v}_{j+1}, v_i)$
6. $\hat{v}_{j+1} = \hat{v}_{j+1} - h_{ij} v_i$
7. Endfor
8. $h_{j+1,j} = \|\hat{v}_{j+1}\|_2$. If 0, then goto 11.
9. $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$.
10. Endfor
11. Form the approximate solution:
Let $\bar{H}_m$ be the $(m+1) \times m$ upper Hessenberg matrix whose nonzero entries are the coefficients $h_{i,j}$, $1 \le i \le j+1$, $1 \le j \le m$.
Find y_m, such that $\|\beta e_1 - \bar{H}_m y\|_2$ is minimized and define $u_m = u_0 + V_m y_m$, where $V_m = [v_1, v_2 \cdots v_m]$.
12. Compute $r_m = b - Ax_0$; if satisfied then stop else compute $x_0 = x_m$, $v_1 = r_m / \|r_m\|$ and goto 2

1.3 Preconditioning

Preconditioning is the process of transforming a given linear system into another system, which has the same solution, but with more favorable distribution of eigenvalues.

We will use the right preconditioning[3], i.e, let M be a matrix close to A, and $Mx = y$ is easy to solve. Then $Ax = b$ is equivalent to

$$(AM^{-1})(Mx) = b$$

Practically, it needs 2 steps, namely,

$$(AM^{-1})y = b,$$
$$Mx = y.$$

Preconditioning introduces additional cost per iteration but the iterative solution of the preconditioned equation could converge faster, so that the total cost could be reduced. The following is a description of the Preconditioned GMRES(m) with matrix M.

Algorithm 1.2 Preconditioned GMRES(m)

1. Choose an initial guess vector u_0 and compute $r_0 = b - Au_0$,
$\beta = \|r_0\|_2$, $v_1 = \frac{r_0}{\beta}$
2. For j=1, , ,m, Until Convergence Do
3. $\hat{v}_{j+1} = AM^{-1} v_j$
4. For i =1, ···, j-1 Do
5. $h_{i,j} = (\hat{v}_{j+1}, v_i)$
6. $\hat{v}_{j+1} = \hat{v}_{j+1} - h_{ij} v_i$.
7. Endfor
8. $h_{j+1,j} = \|\hat{v}_{j+1}\|_2$ If 0 then goto 11.
9. $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$.
10. Endfor
11. Form the approximate solution:
Let $\bar{H}_m$ be the(m+1)xm upper Hessenberg matrix whose nonzero entries are the coefficients $h_{i,j}$, $1 \le i \le j+1$, $1 \le j \le m$.
Find y_m, such that $\|\beta e_1 - \bar{H}_m y\|_2$ is minimized and define $u_m = u_0 + M^{-1}(V_m y_m)$, where $V_m = [v_1, v_2 \cdots v_m]$.
12. Compute $r_m = b - Ax_0$; if satisfied then stop else compute $x_0 = x_m$, $v_1 = r_m / \|r_m\|$ and goto 2

1.4 ILU(0) Factorization

Meijerink and Van. der Vorst[6] introduced a so-called Incomplete LU(ILU) preconditioner for symmetric matrices. The following is a modification of the original ILU for nonsymmetric matrices, as described in [3]. Let $A = LU + E$, where $L_{ij} = U_{ij} = 0$ if $A_{ij} = 0$ and $E_{i,j} = 0$ if $A_{i,j} \neq 0$. Let NZ(A), the nonzero pattern of A, denote the set of pairs of {i,j} for which A_{ij}, the (i,j) entry of A, is nonzero. Let N denote the dimension of the matrix A.

Algorithm 1.3 ILU(0) Factorization

1. For i=, ···, Until N Do
2. For j=1, ···, Until N Do
3. If((i,j) belongs to NZ(A)) then
4. $s_{i,j} = A_{i,j} - \sum_{t=1}^{\min(i,j)-1} L_{i,t} U_{t,j}$

5. if($i \geq j$) then $L_{i,j}=s_{i,j}$
6. if($i<j$) then $U_{i,j}=s_{i,j}/L_{i,i}$
7. Endif
8. Endfor
9. Endfor

Then, in particular, $Mx=LUx=y$ will be easy to solve, since $Mx=y$ can be decomposed into two steps, the solutions of $Lz=y$ and $Ux=z$, which are simply forward and backward substitutions. By definition, the L and U matrices in ILU(0) have together the same number of nonzero elements as the original matrix A.

2. Multi-coloring and Level Scheduling

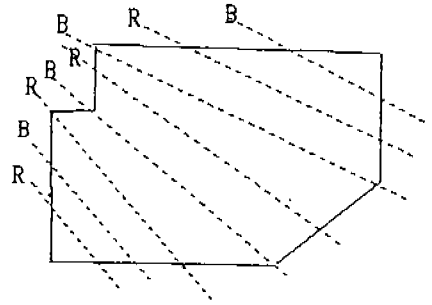
2.1 Multi-color reordering

Given a mesh, multi-coloring consists of assigning a color to each point so that the couplings between two points of the same color are eliminated in the discretization matrix. For example, for the 5-point Laplacian on a square with two colors in a checkerboard fashion we can remove the coupling between any two points of the same color, so that the values at all points of one color can be updated simultaneously. Similarly, four colors are needed to color the grid points of the same color of the 9-point Laplacian. However, it has been known that the convergence rate for the reordered systems often deteriorates.[4] For the model problem SSOR(Symmetric Successive Over Relaxation) and preconditioned-CG with the red/black ordering have a worse convergence rate than with the natural ordering. The following table[6] contains the rates of convergence for SSOR, and ILU(0) preconditioned-CG methods with natural and red/black ordering for the 5-point Laplacian matrix.

(Fig. 1) illustrates the line version of the classical point red/black ordering. This line version can be used for arbitrary shaped grids.

(Table 1) Rate of convergence when reordering is used. h is the meshsize

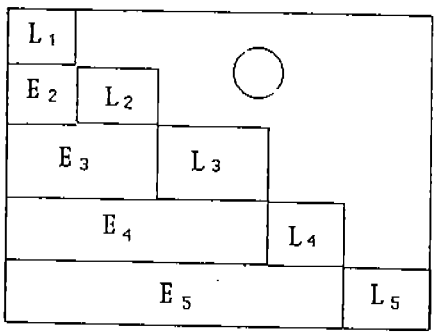
	SSOR	ILU-CG
Natural Ordering	$O(h)$	$O(\sqrt{h})$
red/black Ordering	$O(h^2)$	$O(\sqrt{h})$



(Figure 1) Line red/black ordering for general grid

2.2 Level Scheduling

Rather than pursuing the parallelisms through reordering the level scheduling technique exploits the structure of the given matrix. If the matrix comes from the discretizations of PDEs such as by FDM or FEM the value of a certain node is usually dependent on only the values of its neighbors. Hence, once the values of its neighbors are known, that node can be updated. For example, let us assume that we have an ILU(0) factorization for the 5-point Laplacian for the Poisson equation, and for a



(Figure 2) Block Partitioning for L

preconditioner we need to solve

$$Lz=y,$$

and

$$Ux=z$$

Level scheduling is a process of finding new ordering of the nodes so that the new matrix would look like as in ⟨Fig. 2⟩, where the L_i's are diagonal blocks, and E_i's are the nondiagonal remainder. This technique would work equally well for three dimensional problems as well as two dimensional problems. We will assume that $l_{i,i}=1$, $1\le i\le N$, and the matrix L is stored in Compressed Sparse Row(CSR) format. CSR format is described by the following data structures.

- AL : nonzeros of L, stored by rows.
- JAL : column indices for each element in AL.
- IAL : IAL(i) points to the start of row i in AL, JAL.

Level scheduling is a process of finding new ordering of the nodes so that the new matrix would look like ⟨Fig. 2⟩, where the L_i's are diagonal blocks. Consider the following formula for x_i,

$$x_i=b_i-\sum_{j<i,\ l_{i,j}\neq 0} l_{i,j}\,x_j$$

This means that x_i can be computed after all the components x_j appearing in the sum have been computed. Then, from the adjacency graph of the matrix we could determine the groups of equations that can be solved at the same time. An imaginary node is created and called the root node, and the depth of a node is the maximum distance of that node from the root,i.e,

$$\text{depth(i)}=\begin{cases}1 \text{ if } l_{i,j}\\ 1+\max_{j<i}\{\text{depth(j)},\ l_{i,j}\neq 0\}, \text{otherwise}\end{cases}$$

A level of the graph is, by definition, the set of nodes with the same depth. Thus the depth of a node will be the same as the block number of its row in the block algorithm, and the nodes of a level will define the set of rows that belong to the block. Let q(1:n) defines new ordering after regrouping nodes in the same level and level(i), i = 1, ···, nlev+1 points to the beginning of the i-th level in that array.

Algorithm 2.1 Level-scheduling

```
do lev= 1, nlev
      j1=level(lev)
      j2=level(lev+1)-1
      do k=j1, j2
        i=q(k)
        do j=ial(i), ial(i+1)-1
          x(i)=x(i)- a(j)*x(jal(j))
        enddo
        enddo
enddo
```

The j loop will be suitable for vector processing. This technique would work equally well for three dimensional problems as well as two dimensional problems.

2.3 ILUT preconditioner

The accuracy of the ILU factorization is often insufficient to yield an adequate rate of convergence. One way to enhance the accuracy is to increase the number of nonzero elements in L and U matrices beyond the original sparsity pattern, to obtain an algorithm ILU(p). ILU(0) will be equivalent to ILU defined previously. We could generalize this definition by introducing the concept of level of fill-in. The idea is quite simple. Any nonzero element in A is initially given a level of fill-in equal to 0. When a fill-in element $f_{i,j}$ is created in position (i, j) by a formula such as $f_{i,j}=-L_{ik}U_{kj}$ then its level of fill-in is defined by

$$\text{level}(f_{ij})=\text{level}(L_{ik})+\text{level}(U_{kj})+1$$

This systematic definition allows one to derive a strategy for discarding elements. Typically, for diagonal dominant matrices, the higher the level of fill-in of an element, the smaller its

magnitude, and this suggests dropping any fill-in element whose level is higher than a certain integer. p However, for more general matrices the size of an element is not necessarily related to its level of fill-in. For simplicity, we will describe a variant of ILU(p) which performs the symbolic and the numeric factorization at the same time.

Algorithm 2.2 ILU(p)

1. For all nonzero elements $a_{i,j}$, define $u_{i,j}=a_{i,j}$, $lev(u_{i,j})=0$ $.
2. For i=2, ···, N Do
3. For each k=1, ··· , i-1 and if $u_{ik}\neq 0$ do
4. Compute $l_{ik}=u_{ik}/u_{kk}$ and set lev (l_{ik}) $=lev(u_{ik})$.
5. Compute $ u—{i, *}=u—{i, *} -l —{i, k} u—{k,*} $.
6. Update the levels of $u_{i,*}$ using the definition of level.
7. Replace any element in row i with $lev(u_{ij}) >p$ by zero.
8. EndFor
9. EndFor

There are a number of drawbacks to the above algorithm. First, the amount of fill-in and computational work for obtaining the ILU(p) factorization is not predictable for $p>0$. Second, the cost of updating the levels can be quite high. One drawback of this approach is that in some cases the level of fill-in may not be a good indicator of the size of the elements that are being dropped. Thus, we may drop large elements and obtain an inaccurate incomplete factorization. Saad[10] introduced a variant called ILUT(p,r)(Incomplte LU with Threshold), making provisions both for the the amount of fill-in per row and the size of the element dropped. r is a parameter controlling the threshold of the size of the element dropped.

Algorithm. 2.3 Generic ILU Factorization with threshold(ILUT(p, r)

```
0  row(1:n)=0
1   do i=2, n
2    row(1:n)=a(i, 1:n) ! sparse copy
3     for(k=1, i-1 and where row(k)
      is nonzero) do
4      row(k):=row(k)/a(k, k)
5       apply a dropping rule to row(k)
6        if (row(k). ne. 0) then
7         row(k+1:n)=row(k+1:n)-row
          (k)*u(k, k+1:n) ! sparse update
8        endif
9        apply a dropping rule to row(1:n)
10   enddo
11   l(i, 1:i-1)=row(1:i-1) ! sparse copy
12   u(i,i:n)=row(i:n) ! sparse copy
13   row(1:n)=0 ! sparse set-to-zero
     operation
14   enddo
15 enddo
```

Various strategies could be adopted in line 5 and 9. For example, in line 5 an element in row (k) can be dropped if it is smaller than the relative tolerance r_i which is obtained by multiplying r by the original i-th row (e.g. the 2-norm). In line 9, a dropping rule of different type is applied. First, we drop again any element in the row whose magnitude is below the relative tolerance r_i. Then we keep only the p largest elements in the L part of the row and the p largest elements in the U part of the row in addition to the diagonal element which is always kept. We can view ILU(0) as a particular case of the above algorithm. The dropping rule for ILU(0) is simply to drop elements that are in positions not belonging to the original structure of the matrix.

3. Experiments

3.1 Test problems

- Problem 1. Elman's problem [3]

$$-(bu_x)_x-(cu_y)_y+(du)_x+du_x+(eu)_y+eu_y+fu=g \quad (1)$$

$$\Omega = (0,1)\times(0,1)$$
$$u = on\ \partial\Omega$$

where

$b=\exp(-xy)$, $c=\exp(xy)$, $d=\beta(x+y)$,

$e=\gamma(x+y)$, $f=\dfrac{1}{(1+xy)}$.

- Problem 2. Discontinuous Elman's problem

Same as problem 1, except

$b(x, y)=100$, if $x<0.5$, $y<0.5$

$=0$, elsewhere

Same for $c(x, y)$.

3.2 Results

We tested the above problems on the CRAY-2 machine at the University of Minnesota Supercomputer Center. 〈Table 1-5〉 shows the elapsed times of various steps. For 〈table 1-4〉 N is 128x128, and $\gamma=50$, $\beta=1$, which leads to nonsymmetric matrices. For table 5 the Sherman5 test matrix from the Harwell-Boeing collection has N=3312. In all cases GMRES(4) was used as an iterative method, with $\varepsilon=10^{-6}$ as the final stopping criterion. For problem 1 and 2, both FDM and FEM were tested. For ILUT τ was set to 10^{-3}, hence we used the notation ILUT(p) instead of ILUT(p,τ). The parallel stands for the level scheduling approach, and serial for the usual one without level scheduling. The elapsed time is in seconds on the CRAY-2 machine. Since the elapsed time is done in an non-exclusive mode we took an average of the same computation over several times.

We can think of ILU(0) with level scheduling as the industry standard. By comparing the total CPU time on the tables we can see that ILUT(8) with level scheduling on line red/black ordering has the best performance, except 〈Table 2〉. So, with ILUT(8) line red/black ordering has the better rate of convergence than natural ordering, contrary to the usual worsening of the rate of convergence brought about by a type of red/black ordering. With ILUT(8), since we introduce 8 more fill-ins, the number of iterations becomes smaller than that with ILU(0) but the time for preparing the ILUT factors increases. With level scheduling there is a nonnegligible overhead. Also, for higher number of fill-in, we can expect a potential benefit of more robust convergence than ILU(0). This could be very useful for indefinite problems.

〈Table 6-7〉 show the Frobenius norm of error matrix $E=A-LU$. (Frobenius norm of a matrix is defined by $\|A\|_F=\sqrt{\sum_i\sum_j a_{i,j}^2}$).

When the level of fill-in is 0, as in ILU(0) or ILUT(0), the error norm with line red/black ordering is much larger than that with natural ordering, but as it increases, the gap decreases and finally for level of fill-in greater or equal to 8 the situation reverses. That means that for level of fill-in greater or equal to 8 the ILUT factorization matrix with line red/black order-

〈Table 1〉 Elman problem FDM-GMRES(4)

	Natural ordering				Line Red/Black ordering	
	ILU(0)		ILUT(8)		ILUT(8)	
	Serial	Parallel	Serial	Parallel	Serial	Parallel
ILU	0.28	0.28	2.41	2.40	1.96	2.0
Lev-sched	0.	0.71	0.	0.74	0.	0.68
Iterations	103	103	15	15	10	10
Execution	12.02	3.29	3.71	1.11	2.18	0.69
Total time	12.35	4.33	6.18	4.31	4.58	3.80

〈Table 2〉 Elman problem FDM-GMRES(4)

	Natural ordering				Line Red/Black ordering	
	ILU(0)		ILUT(8)		ILUT(8)	
	Serial	Parallel	Serial	Parallel	Serial	Parallel
ILU	0.38	0.39	2.57	2.58	2.93	2.95
Lev-sched	0	0.65	0	0.76	0	0.78
Iterations	42	42	14	14	9	9
Execution	6.03	1.10	3.69	1.15	2.37	0.70
Total time	6.47	2.80	6.32	4.55	5.83	4.96

〈Table 3〉 Discontinuous Elman problem FDM-GMRES(4)

	Natural ordering				Line Red/Black ordering	
	ILU(0)		ILUT(8)		ILUT(8)	
	Serial	Parallel	Serial	Parallel	Serial	Parallel
ILU	0.28	0.28	2.23	2.18	1.69	1.70
Lev-sched	0	0.72	0	0.66	0	0.63
Iterations	173	173	50	50	24	24
Execution	20.32	5.57	11.61	4.49	5.11	1.90
Total time	20.65	6.62	13.83	7.38	7.23	4.66

〈Table 4〉 Discontinuous Elman problem FDM-GMRES(4)

	Natural ordering				Line Red/Black ordering	
	ILU(0)		ILUT(8)		ILUT(8)	
	Serial	Parallel	Serial	Parallel	Serial	Parallel
ILU	0.39	0.39	2.58	2.56	2.11	3.30
Lev-sched	0.	0.63	0	0.68	0	0.67
Iterations	118	118	40	40	26	26
Execution	16.77	4.49	10.09	4.08	5.32	2.00
Total time	17.21	5.57	12.72	7.38	7.96	5.30

〈Table 5〉 Sherman5-GMRES(4)

	Natural ordering				Line Red/Black ordering	
	ILU(0)		ILUT(8)		ILUT(8)	
	Serial	Parallel	Serial	Parallel	Serial	Parallel
ILU	0.086	0.085	0.76	0.77	0.45	0.45
Lev-sched	0.	0.097	0	0.10	0	0.10
Iterations	90	90	40	40	19	19
Execution	2.32	0.79	1.37	0.80	0.60	0.39
Total time	2.42	0.99	2.14	1.68	1.23	1.10

〈Table 6〉 Elman problem with $N=128\times128$, $\tau=50$, $\beta=1$, FDM

Frobenius norm of E=A-LU						
fill-in	0	2	4	6	8	10
Natural ordering	59.77	8.01	3.27	2.69	2.75	2.75
LineR/B ordering	117.64	62.46	9.76	4.03	1.83	1.76

〈Table 7〉 Elman problem with $N=128\times128$, $\tau=50$, $\beta=1$, FDM

Frobenius norm of E=A-LU						
fill-in	0	2	4	6	8	10
Natural ordering	2847.82	539.62	279.03	159.07	149.50	154.93
LineR/B ordering	6675.54	272.04	622.73	258.18	133.12	95.39

ing becomes closer to A than that with natural ordering. This also confirms our results in 〈Table 1-5〉. For the 〈tables 1-5〉 the times are in seconds.

4. Conclusions

The CRAY-2 is a vector machine whose performance heavily depends on the efficient vectorization of a given algorithm. In the case of ILU type preconditioner for large sparse linear systems that arise from the discretizations of PDE we have shown that for sufficiently large number of fill-ins in ILUT line red/black ordering performed with the level scheduling approach could produce a better rate of convergence than ILU(0) with natural ordering on the CRAY-2 machine. This approach could be useful in the vectorization of ILU type preconditioner.

References

[1] E. Anderson, "Parallel implementation of preconditioned conjugate gradient methods for solving sparse systems of linear equations", Technical Report 805, CSRD, Univeristy of Illinois, Urbana, IL, 1988. MS Thesis

[2] R. Chandra, "Conjugate gradient methods for partial differential equations", Ph. D. Thesis, Dept of Computer Science, Yale University, 1978.

[3] H. Elman, "Iterative methods for large, sparse, nonsymmetric systems of linear equations", Ph. D Thesis, Yale University, 1982.

[4] C. -C. Jay Kuo and Tony Chan, "Two-color Fourier analysis of iterative algorithms for elliptic problems with red/black ordering", SIAM J. Sci. Stat, Vol. 11, No. 4, pp. 767-793, 1990.

[5] S. Ma, "Parallel block preconditioned Krylov subspace methods for partial differential equations," Ph. D Thesis, University of Minnesota, Minneapolis, Aug, 1993.

[6] J. Meijerink and H. Van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix", Math. Comp., Vol. 31, pp. 148-162, 1977.

[7] D. O'Leary, "Ordering schemes for parallel processing of certain mesh problems", SIAM J. Sci. Stat Vol. 5, pp. 620-632, 1984.

[8] Y. Saad and M. Schultz, "GMRES : A Generalized minimal residual algorithm for solving nonsymmetric linear systems," SIAM J. Sci. Stat, Vol. 7, July, 1986.

[9] Y. Saad, "Highly parallel preconditioners for general sparse matrices," Technical Report UMSI 92/45, University of Minnesota, Army High Performance Computing Research Center, Minneapolis, Minnesota, 1992.

[10] Y. Saad, "ILUT : A dual threshold incomplete LU factorization", UMSI report 92/38, University of Minnesota Supercomputer Institute, 1992

[11] Y. Saad, "Krylov subspace methods on supercomputers", SIAM J. Sci. Stat, Vol. 10, pp. 1200-1232, Nov, 1989

[12] H. Van Der Vorst, "A vectorizable variant of some ICCG methods", SIAM J. Sci. Stat, Vol. 3, pp. 1174-1185, 1989

[13] R. Varga, Matrix Iterative Analysis, Prentice-Hall, New York, 1962

[14] D. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971

마 상 백

1978년 서울대학교 계산통계학과 졸업
1978년~83년 국방과학연구소 연구원
1983년~87년 미국 미네소타 주립대학교 수학 석사
1987년~93년 미국 미네소타 주립대학교 전자계산학 박사
1994년~현재 한양대학교 공과대학 전자계산학과 전임강사
관심분야 : 수치해석, 병렬알고리즘