数　値　解　析　37－7
（１９９1．７．１７）

マルチカラーＭＩＬＵ前処理反復法の収束性とそのＳＸ－３／１４上での性能

土肥　俊，保志　敦
日本電気株式会社

概要　3 次元楕円型偏微分方程式の離散化により生じる連立一次方程式のベクトル計算機による求解を検討する。多色のマルチカラーオーダリングを用いた ILU 前処理法 (藤野 & 土肥, 1991) に Gustafsson 流の収束加速法 (土肥, 1991) を組み合わせ、収束性に優れたベクトル計算機向き反復解法を構成する。多色マルチカラーオーダリングを用いた前処理反復法は、従来の (少ない色数の) マルチカラーオーダリングによる方法 (Poole & Ortega, 小柳, 1987) に比べて (同精度に至るまでの) 反復数が少なく、一方 (大規模 3 次元問題に対して)1 反復当たりのベクトル計算時間は両者の間に差がない。この方法を NEC SX-3/14 上に実現し、数種の移流拡散方程式により評価を行なった。従来のハイパープレーン法によるベクトル化法と比べて 2 ～ 3 倍の高速化を達成した。また絶対性能では 2GFLOPS を実測した。

LARGE-NUMBERED MULTICOLOR MILU PRECONDITIONING ON SX-3/14

SHUN DOI * and ATSUSHI HOSHI †

Abstract. This paper considers the vector implementation of preconditioned conjugate gradient (PCG) type methods in solving sparse linear systems of the 3D 7-point difference form. The modified incomplete LU (MILU) factorization is sought, where the system is numbered with respect to the multicolor ordering with a large number of colors (e.g. 75). The advantage in the usage of the large-numbered multicolor ordering is that the PCG-type method based on this ordering tends to require fewer iterations (to reach the same accuracy) than the same method based on a small-numbered multicolor ordering, while both orderings spend almost the same computational time per iteration if the problem is sufficiently large. Numerical experiments are carried out on the SX-3/14 supercomputer, using convection-diffusion equations discretized on a $76\times76\times76$ grid. Results of experiments show that the large-numbered multicolor (M)ILU/Bi-CGSTAB method, which records a speed more than 2 GFLOPS, converges faster than both the small-numbered multicolor and the hyperplane (M)ILU/Bi-CGSTAB methods.

LARGE-NUMBERED MULTICOLOR MILU PRECONDITIONING ON SX-3/14

SHUN DOI * and ATSUSHI HOSHI †

Abstract. This paper considers the vector implementation of preconditioned conjugate gradient (PCG) type methods in solving sparse linear systems of the 3D 7-point difference form. The modified incomplete LU (MILU) factorization is sought, where the system is numbered with respect to the multicolor ordering with a large number of colors (e.g. 75). The advantage in the usage of the large-numbered multicolor ordering is that the PCG-type method based on this ordering tends to require fewer iterations (to reach the same accuracy) than the same method based on a small-numbered multicolor ordering, while both orderings spend almost the same computational time per iteration if the problem is sufficiently large. Numerical experiments are carried out on the SX-3/14 supercomputer, using convection-diffusion equations discretized on a $76 \times 76 \times 76$ grid. Results of experiments show that the large-numbered multicolor (M)ILU/Bi-CGSTAB method, which records a speed more than 2 GFLOPS, converges faster than both the small-numbered multicolor and the hyperplane (M)ILU/Bi-CGSTAB methods.

Key Words. Linear system, modified incomplete LU factorization, vectorization, ordering, multicolor, supercomputer.

1. Introduction. Preconditioned conjugate gradient (PCG) type methods are widely used in solving the sparse linear system $Au = b$ that stems from various engineering problems. These PCG-type methods are defined as unpreconditioned CG-type methods for an equivalent system $M^{-1}Au = M^{-1}b$, where M is a preconditioner. Let L, D and U, respectively, be the strictly lower triangle, the diagonal and the strictly upper triangle of $A = L + D + U$. The incomplete factorization, which is a class of commonly used preconditioners, can be written in the form

$$M = (\Delta + L)(I + \Delta^{-1}U), \tag{1.1}$$

where the diagonal matrix Δ is defined, in the (diagonal) incomplete LU factorization [10, 11], by

$$\Delta + \mathrm{diag}(L\Delta^{-1}U) = D, \tag{1.2}$$

and, in its Gustafsson's modification [9], by

$$\Delta + \mathrm{rowsum}(L\Delta^{-1}U) = D, \tag{1.3}$$

apart from relaxation parameters [1, 9, 15, 18]. Here, "diag($L\Delta^{-1}U$)" is a diagonal matrix composed of diagonal elements of $L\Delta^{-1}U$ and "rowsum($L\Delta^{-1}U$)" is a diagonal matrix where each diagonal element is equal to the summation of row elements. In each iteration in PCG-type methods, it is necessary to find v such that $Mv = w$ for a given w. In the incomplete factorization (1.1), this solution is carried out by the forward and backward substitutions

$$v' = \Delta^{-1}(w - Lv'), \tag{1.4}$$

$$v = v' - \Delta^{-1}Uv. \tag{1.5}$$

* C&C Information Technology Research Laboratories, NEC Corporation; 4-1-1, Miyazaki, Miyamae-ku, Kawasaki, 216 Japan; E-mail: doi@ibl.cl.nec.co.jp.

† EDP Manufacturing Industry Division, NEC Corporation; 5-34-2, Shiba, Minato-ku, Tokyo, 108 Japan; E-mail: hoshi@msd.mt.nec.co.jp.

On vector and/or parallel computers, these forward and backward substitutions turn out to be the most time consuming part in the whole PCG computation. A popular way to vectorize (1.4) (and (1.5)) is to reorder the linear system $Au = b$, so as to avoid recurrence relations in the substitution $Lv' \to v'$ (and $Uv \to v$). One well-known technique is to use the multicolor ordering [7, 12, 13]. In the multicolor ordering, A has a block form $A = [A_{ij}]$ $(i, j = 1, \ldots, c)$ where the diagonal blocks A_{ii} $(i = 1, \ldots, c)$ are diagonal matrices. Here, c is the number of colors. If $c = 2$, the multicolor ordering results in the popular red-black ordering. If an $N \times N$ matrix A is numbered with respect to a multicolor ordering with c colors, then the forward substitution (1.4) for elements in each v'_i $(i = 1, \cdots, c)$ can be computed in parallel, where the average vector length is N/c. It is also the same for the backward substitution (1.5).

In early studies conducted by Poole and Ortega [13] and by Oyanagi [12], only small c (e.g. 7) was examined. One of their main conclusions is that the multicolor PCG method converges slowly compared with the popular row PCG method, especially for anisotropic problems, and that this slow convergence may possibly offset the gain achieved by vectorization. Recently, Fujino and Doi examined the multicolor ICCG method with large c (e.g. 49 or 99) on some vectorcomputers [8]. Their conclusion is that the large-numbered multicolor ICCG method converges faster than the small-numbered multicolor ICCG method, especially for anisotropic problems, and that the method is at least as fast as the traditional hyperplane and diagonal ICCG methods.

Now, recall that each vectorcomputer has a specific vector length for which the computational speed is saturated, and that little improvement will be obtained for longer vectors. On current high speed vectorcomputers, this vector length, denoted l_{lim}, is $O(10^3)$. Assume that the size of a linear system is $N = 1000000$ (which may result from discretization on a $100 \times 100 \times 100$ grid). The multicolor PCG method with $c = O(10^2)$ then provides the vector length of $O(10^4)$ that suffices these vectorcomputers. Since larger c is expected to provide better convergence in PCG method [2], and since vector performance for any c less than $O(10^2)$ is almost the same, the multicolor PCG method with $c = O(10^2)$ is expected to have less computational time than the method with c smaller than this. More precisely, the multicolor PCG method with $c \approx N/l_{lim}$ is expected to give "optimal" performance for a given problem size N and a given "saturation point" l_{lim}.

In the rest of this paper, we assume that the finite difference grid, termed (i, j, k), is a rectangle $(1 \le i \le n_x, 1 \le j \le n_y, 1 \le k \le n_z)$, and that coefficients in A corresponding to a node (i, j, k) reside in $\{i + (j-1)n_x + (k-1)n_x n_y\}$th elements in individual arrays. These assumptions are quite natural in "real life".

Section 2 introduces model problems and the multicolor algorithms. Section 3 reports timing results in the multicolor forward and backward substitutions (which dominate the computational time), measured on the SX-3/14 supercomputer. Section 4 reports results of a comparison between the multicolor (M)ILU/Bi-CGSTAB [1] method (where $c = 75, 25, 5$) and the hyperplane (M)ILU/Bi-CGSTAB method. The superiority of the present method will be concluded.

2. Algorithms.

2.1. Model problem. We consider the 3D convection-diffusion equation

$$(k_x u'_x)'_x + (k_y u'_y)'_y + (k_z u'_z)'_z + v_x u'_x + v_y u'_y + v_z u'_z = 0 \quad (k_x > 0, k_y > 0, k_z > 0), \tag{2.1}$$

[1] The Bi-CGSTAB (Bi-Conjugate Gradient STABle) method is a nonsymmetric linear solver recently proposed by Van der Vorst [19]. Numerical experiments, using some convection-diffusion equations, show that the method converges more smoothly and at least as fast as the CGS method [14] (see also [20]).

5	27	48	6	32	53	11	33	58	16	38	59	21	43	64	22
46	4	26	47	9	31	52	10	36	57	15	37	62	20	42	63
24	45	3	25	50	8	30	51	13	35	56	14	40	61	19	41
1	23	44	2	28	49	7	29	54	12	34	55	17	39	60	18
	$k = 1$				$k = 2$				$k = 3$				$k = 4$		

FIG. 1. *Multicolor ordering with 3 colors for a* $4 \times 4 \times 4$ *grid. The three colors correspond, respectively, to three sets of nodes:* $\{1, \ldots, 22\}$, $\{23, \ldots, 43\}$ *and* $\{44, \ldots, 64\}$.

which is given on a rectangular domain Ω. By discretizing (2.1) with the standard 7-point finite difference approximation on an $n_x \times n_y \times n_z$ grid, a linear system $Au = b$ is obtained. Assume that the unknowns are numbered in such a way that described in § 1, then the ith row $(i = 1, \ldots, N(= n_x \times n_y \times n_z))$ in the left-hand-side Au can be written as

$$(Au)_i = a_i u_{i-n_x n_y} + b_i u_{i-n_x} + c_i u_{i-1} + d_i u_i + e_i u_{i+1} + f_i u_{i+n_x} + g_i u_{i+n_x n_y}. \tag{2.2}$$

2.2. Multicolor ordering. Let $p(i,j,k)$ be a node in the grid $(1 \leq i \leq n_x, 1 \leq j \leq n_y, 1 \leq k \leq n_z)$, and P be the set containing all nodes. Define disjoint subsets, $S_1, S_2, \ldots, S_c$ $(P = \cup_{l=1}^{c} S_l)$, by

$$p(i,j,k) \in S_{\{i+j+k-3(mod\ c)\}+1}. \tag{2.3}$$

Here, c is in the range of $2 \leq c \leq n_x + n_y + n_z - 2$. Since coefficients in A corresponding to nodal connections within a single subset S_l are zero, the forward substitution (1.4) corresponding to individual S_l can be computed in parallel. (The same applies for (1.5).) Hence, if A is renumbered in such a way that nodes in a younger subset are numbered earlier than nodes in an older subset, then this numbering provides a multicolor ordering. Note that, if $c = 2$, the multicolor ordering (2.3) results in the popular red-black ordering, while it results in the row ordering if $c = n_x + n_y + n_z - 2$.

In order to vectorize the substitutions corresponding individual subsets effectively, data units corresponding to nodes in individual subsets should have a constant stride in memory. Under the assumption described in § 1, this requirement is achieved if

$$n_x(\text{mod } c) = n_y(\text{mod } c) = 1. \tag{2.4}$$

Figure 1 illustrates a numbering satisfying (2.4).

2.3. Multicolor incomplete factorizations. If (2.4) holds, then the factorization (1.2) for (2.2) based on the ordering (2.3) is written as follows:

```
for $i = 1 : N : c$
    $\Delta_i = d_i$
end
for $k = 2 : c - 1$
    for $i = k : N : c$
        $\Delta_i = d_i - a_i g_{i-n_x n_y}/\Delta_{i-n_x n_y} - b_i f_{i-n_x}/\Delta_{i-n_x} - c_i e_{i-1}/\Delta_{i-1}$
    end
end
for $i = c : N : c$
    $\Delta_i = d_i - a_i g_{i-n_x n_y}/\Delta_{i-n_x n_y} - b_i f_{i-n_x}/\Delta_{i-n_x} - c_i e_{i-1}/\Delta_{i-1}$
```

```
                $-e_ic_{i+1}/\Delta_{i+1} - f_ib_{i+n_x}/\Delta_{i+n_x} - g_ia_{i+n_xn_y}/\Delta_{i+n_xn_y}$
end
```

In the present program, a Gustafsson-type modification is additionally incorporated, in which two relaxation parameters, α and β, are used [3]. This modification results in the unmodified ILU factorization if $\alpha = \beta = 0$, and it results in the standard modification if $\alpha = \beta \neq 0$. In [3], it is shown that a choice $\alpha \neq \beta$ sometimes provides better convergence than the choice $\alpha = \beta$. For more detailed discussion, see [3].

The forward and backward substitutions are written as follows:

MCINV routine.

```
for $i = 1 : N : c$
    $v_i = w_i/\Delta_i$
end
for $k = 2 : c-1$
    for $i = k : N : c$
        $v_i = (w_i - a_iv_{i-n_xn_y} - b_iv_{i-n_x} - c_iv_{i-1})/\Delta_i$
    end
end
for $i = c : N : c$
    $v_i = (w_i - a_iv_{i-n_xn_y} - b_iv_{i-n_x} - c_iv_{i-1}$
            $-e_iv_{i+1} - f_iv_{i+n_x} - g_iv_{i+n_xn_y})/\Delta_i$
end
for $k = c-1 : 2 : -1$
    for $i = k : N : c$
        $v_i = v_i - (e_iv_{i+1} + f_iv_{i+n_x} + g_iv_{i+n_xn_y})/\Delta_i$
    end
end
for $i = 1 : N : c$
    $v_i = v_i - (a_iv_{i-n_xn_y} + b_iv_{i-n_x} + c_iv_{i-1}$
            $+e_iv_{i+1} + f_iv_{i+n_x} + g_iv_{i+n_xn_y})/\Delta_i$
end
```

Here, the loops for i can be vectorized, where the vector length is N/c and the stride in memory access is c. (In the real program, the division by Δ_i is replaced by a multiplication by $1/\Delta_i$; i.e., the factorization program calculates $1/\Delta_i$.)

3. Timing tests. Since the *MCINV* routine dominates the overall computational speed in a multicolor PCG program, we perform timing tests in the *MCINV* routine on the SX-3/14 (single processor with four sets of pipes). The outline of this machine is described as follows:

Clock cycle	Peak performance	Memory size	Memory banks
2.9 nsec	5.5 GFLOPS	1 Gbyte	512

The 5.5 GFLOPS performance is attained if 16 vector pipes (operating in a 2.9 nsec cycle) are filled with vector data. (Each set of vector pipes contains 4 pipes, resulting in 16 pipes.) Measurement will be made on how vector length and stride in memory access affect the computation speed. In this program, the vector length is N/c and the stride is c.

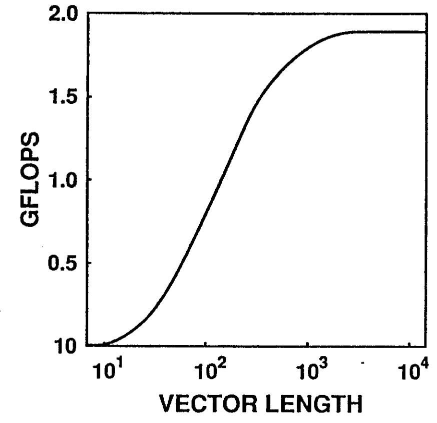

FIG. 2. *Vector speed versus vector length N/c in the MCINV routine.*

Vector speed versus vector length. Figure 2 shows the timing results with 64-bit arithmetic for $4 \le n \le 100$ ($64 \le N \le 1000000$), where $n_x = n_y = n_z = n$ and $c = n - 1$. It shows that the speed is saturated for the vector length $l_{lim} \approx 2500$. The speed achieved then is about 1.89 GFLOPS.

Vector speed versus stride in memory. Another important factor for attaining high vector speed is the stride in memory access. The following table shows how the speed depends on this stride. From this table, it is almost necessary to use odd stride in order to attain the best vector performance.

Stride	Odd	Even			
	3	2	4	8	16
Speed(Mflops)	2035	1496	873	480	253
(Ratio)	(1.00)	(0.74)	(0.43)	(0.24)	(0.12)

4. Numerical experiments.

This section reports results of numerical experiments. Model problems are in the form (2.1). Examples 2 and 3 are borrowed from [4].

Example 1: $\Omega = (0,1) \times (0,1) \times (0,1)$. Dirichlet boundary conditions are imposed on $\partial\Omega$. Model parameters are constants. They are chosen as follows:

Case	k_x	k_y	k_z	v_x	v_y	v_z
1	1	1	1	0	0	0
2	100	1	1	0	0	0
3	1	1	1	100	0	0
4	100	1	1	10000	0	0

Robustness in methods will be examined using anisotropic parameter cases.

Example 2: $\Omega = (-1,1) \times (-1,1) \times (-1,1)$. $k_x = k_y = k_z = 1$.

$v_x = -c_p c_0 yz(1-x^2)^2(1-y^2)(1-z^2)(n_x - 1)$.

$v_y = c_p c_1 xz(1-x^2)(1-y^2)^2(1-z^2)(n_x - 1)$.

$v_z = c_p c_1 xy(1-x^2)(1-y^2)(1-z^2)^2(n_x - 1)$.

TABLE 1
Results of experiments for Example 1.

		Number of Iterations				CPU time (sec)			
α	β	HYP	MC(75)	MC(25)	MC(5)	HYP	MC(75)	MC(25)	MC(5)
Case 1									
0	0	46	54	50	54	2.60	0.88	0.81	0.87
0.98	0	16	30	35	49	1.04	0.49	0.57	0.78
0.98	0.98	–	32	37	52	–	0.52	0.60	0.83
Case 2									
0	0	28	31	38	64	1.66	0.51	0.62	1.02
0.98	0	19	74	52	66	1.19	1.20	0.84	1.05
0.98	0.98	–	29	36	72	–	0.48	0.59	1.15
Case 3									
0	0	29	32	32	36	1.71	0.52	0.52	0.58
0.98	0	13	20	20	31	0.88	0.33	0.33	0.50
0.98	0.98	–	17	19	30	–	0.28	0.31	0.48
Case 4									
0	0	5	9	9	21	0.46	0.15	0.15	0.34
0.98	0	4	8	9	20	0.41	0.14	0.15	0.32
0.98	0.98	–	27	28	38	–	0.44	0.46	0.61

"HYP": Hyperplane ordering. "MC(c)": Multicolor ordering with c colors.

$c_0 = 27/2, \quad c_1 = c_0/2, \quad c_p = 0.5, \quad n_x = n_y = n_z.$
B.C.: $u = 100$ if $z = -1$, $u = 0$ otherwise.
This example is a modeling of heat conduction in a box where a side is heated and the flow is rotating.

Example 3: $\Omega = (0, 20) \times (-1, 1) \times (-1, 1)$. $k_x = k_y = k_z = 1$.
$v_x = -c_{px}c_0(1 - y^4)(1 - z^4)(n_x - 1).$
$v_y = -c_p c_1 z(1 - y^2)^2(1 - z^2)(n_y - 1).$
$v_z = c_p c_1 y(1 - y^2)(1 - z^2)^2(n_y - 1).$
$c_0 = 0.1, \quad c_1 = 3\sqrt{3}/2, \quad (c_{px}, c_p) = (0.5, 1.0), \quad n_y = n_z.$
B.C.: $u = u_{in}$ if $x = 0$, $\partial u/\partial x = 0$ if $x = 20$,
$\partial u/\partial n = -H_c(u - u_0)$ if $y = \pm 1$ and $z = \pm 1$ $(H_c = 1)$.
This example is heat conduction in a pipe where the flow is along the x axis and rotating in the y-z plane. The linear system resulting from this example is anisotropic: matrix coefficients are dominant in the y and z directions.

These model problems are discretized with the standard 7-point finite differences on a $76 \times 76 \times 76$ grid, resulting in 438976 unknowns.

The Bi-CGSTAB method [19] is used as a basic iterative method, where the multicolor MILU and the traditional hyperplane MILU [15, 17] preconditioners are incorporated. In the hyperplane method, the forward and backward substitutions are vectorized (using the gather-scatter operations) for nodes (i, j, k) satisfying $i + j + k =$ constant. The average vector length (indicated with an asterisk in the table below) is $n^3/(3n - 2)$. The stride in memory access is $n - 1$ (mostly). In the multicolor method, the vector length is n^3/c and the stride is c. The number of colors examined are 75, 25 and 5 (which satisfy the condition (2.4)). Note that all colors tested provide vector length longer than $l_{lim} \approx 2500$.

TABLE 2
Results of experiments for Example 2.

		Number of Iterations				CPU time (sec)			
α	β	HYP	MC(75)	MC(25)	MC(5)	HYP	MC(75)	MC(25)	MC(5)
0	0	71	71	82	91	3.91	1.16	1.34	1.46
0.98	0	23	49	49	79	1.40	0.80	0.80	1.27
0.98	0.98	–	46	53	80	–	0.76	0.87	1.28

TABLE 3
Results of experiments for Example 3.

		Number of Iterations				CPU time (sec)			
α	β	HYP	MC(75)	MC(25)	MC(5)	HYP	MC(75)	MC(25)	MC(5)
0	0	102	100	112	124	5.53	1.63	1.82	1.98
0.98	0	83	154	175	200	4.54	2.51	2.85	3.19
0.98	0.98	–	166	200	200	–	2.70	3.25	3.20

Method	Number of Colors	Vector Length	Stride in Memory
Hyperplane	–	1942*	75
Multicolor	75	5853	75
	25	17559	25
	5	87795	5

All the computations were carried out in 64-bit arithmetic with standard FORTRAN 77. The relaxation parameter sets (α, β) tested were (0, 0) (resulting in unmodified ILU), (0.98, 0) and (0.98, 0.98). Iterations were terminated when $||r||_2/||b||_2 \leq 10^{-6}$. The starting vector $u^{(0)}$ was $\text{diag}(A)^{-1}b$.

The total amount of computation is

$$76 \times n_x \times n_y \times n_z \times n_{iter}.$$

Here, the number 76 comes from the number of operations per node per iteration in the (M)ILU/Bi-CGSTAB method for the matrix of the form (2.2). From this equation and timing results in tables, the computational speed can be calculated: it is slightly more than 2 GFLOPS. No significant difference in GFLOPS rate can be observed between the methods with different c, since vector performance is saturated for all the cases.

In Tables 1–3, the number of iterations tends to decrease as the number of colors increases. This tendency agrees with results presented in [2, 8]. This decrease in the number of iterations hence contributes directly to the reduction in computational time.

With respect to the Gustafsson's modification, it more effectively reduces the number of iterations in the hyperplane method than the multicolor method. Within the multicolor method, the modification is more efficient for lager c. This observation agrees with results presented in [3].

As for the total computational time, the multicolor (M)ILU/Bi-CGSTAB method with $c = 75$ is the fastest of all the cases. The method is 2 to 3 times faster than the traditional hyperplane method. The multicolor method with $c = 75$ is additionally robust against anisotropy in model problems.

5. Conclusion. The large-numbered multicolor MILU preconditioner has been introduced for a sparse matrix of the 3D 7-point difference form. The advantage in the usage of the large-numbered multicolor ordering is that the PCG-type method based on this ordering tends to require fewer iterations (to reach the same accuracy) than the same method based on a small-numbered multicolor ordering, while both orderings spend almost the same computational time per iteration if the problem is sufficiently large.

Numerical experiments have been carried out on the SX-3/14, using some convection-diffusion equations discretized on a $76 \times 76 \times 76$ grid. The large-numbered multicolor (M)ILU/Bi-CGSTAB method has achieved a computational speed exceeding 2 GFLOPS (with 64-bit arithmetic). The present method is 2 to 3 times faster than the Bi-CGSTAB method preconditioned by the traditional hyperplane (M)ILU factorization.

Acknowledgment. Discussions with Seiji Fujino (Institute of Computational Fluid Dynamics, Tokyo) proved to be very helpful.

REFERENCES

[1] C. C. Ashcraft and R. G. Grimes, *On Vectorizing Incomplete Factorization and SSOR Preconditioners*, SIAM J. Sci. Stat. Comput., 9(1988), pp. 122–151.

[2] S. Doi, *On Parallelism and Convergence of Incomplete LU Factorizations*, Appl. Numer. Math., to appear.

[3] ———, *A Gustafsson-Type Modification for Parallel Ordered Incomplete LU Factorizations*, in: T. Nodera, Advances in Numerical Methods for Large Sparse Sets of Linear Systems, 7, Keio University, 1991.

[4] S. Doi and A. Lichnewsky, *Some Parallel and Vector Implementations of Preconditioned Iterative Methods on Cray-2*, Int. J. High Speed Comput, 2(1990), pp. 143–179.

[5] ———, *A Graph-Theory Approach for Analyzing the Effects of Ordering on ILU Preconditioning*, Rapport de Recherche INRIA, 1991 (submitted to SIAM J. Sci. Stat. Comput., 1990).

[6] I. S. Duff and G. A. Meurant, *The Effect of Ordering on Preconditioned Conjugate Gradients*, BIT, 29(1989), pp. 635–657.

[7] H. C. Elman and E. Agrón, *Ordering Techniques for the Preconditioned Conjugate Gradient Method on Parallel Computers*, Comput. Phys. Comm., 53(1989), pp. 253–269.

[8] S. Fujino and S. Doi, *Optimizing Multicolor ICCG Methods on Some Vectorcomputers*, in: R. Beauwens, Proc. IMACS Int. Symp. Iterative Methods in Linear Algebra, March 1991.

[9] I. Gustafsson, *A Class of First Order Factorization Methods*, BIT, 18(1978), pp. 142–156.

[10] J. A. Meijerink and H. A. van der Vorst, *An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix*, Math. Comput., 31(1977), pp. 148–162.

[11] ———, *Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems*, J. Comput. Phys., 44(1981), pp. 134–155.

[12] Y. Oyanagi, *Hyperplane vs. Multicolor Vectorization of Incomplete LU Processing for Wilson Fermion on the Lattice*, J. Inf. Process., 11(1987), pp. 32–37.

[13] E. L. Poole and J. M. Ortega, *Multicolor ICCG Methods for Vector Computers*, SIAM J. Numer. Anal., 24(1987), pp. 1394–1418.

[14] P. Sonneveld, *CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comput., 10(1989), pp. 36–52.

[15] Y. Ushiro, *Vector Computer Version of ICCG Method*, in: M. Mori, Parallel Numerical Computational Algorithms and Related Topics, RIMS Report 514, Kyoto University, 1984 (in Japanese).

[16] H. A. van der Vorst, *ICCG and Related Methods for 3D Problems on Vector Computers*, Comput. Phys. Comm., 53(1989), pp. 223–235.

[17] ———, *Large Tridiagonal and Block Tridiagonal Linear Systems on Vector and Parallel Computers*, Parallel Comput., 5(1987), pp. 45–54.

[18] ———, *High Performance Preconditioning*, SIAM J. Sci. Stat. Comput., 10(1989), pp. 1174–1185.

[19] ———, *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, Preprint Nr. 633, University Utrecht, Dept. Math., 1990.

[20] H. Watanabe and S. Doi *A Comparison of Bi-CGSTAB and CGS Methods for Convection-Diffusion Equations*, in: (the same as ref. [3]), 1991.