数理解析研究所講究録
第1957巻 2015年 45-54

Cache-Cache(カシュカシュ) Elements法による反復法の並列化

Parallelism of iterative methods by Cache-Cache [kaʃkaʃ] Elements technique

九州大学情報基盤研究開発センター　藤野清次 (Seiji Fujino)
Research Institute for Information Technology, Kyushu University
九州大学工学部電気情報工学科　伊東千晶 (Chiaki Itoh)
Faculty of Engineering, Kyushu University
九州大学大学院システム情報科学府　岩里洸介 (Kousuke Iwasato)
Graduate School of Information Science and Electrical Engineering, Kyushu University

Abstract: Eisenstat SSOR preconditioning for Krylov subspace methods is known to be very efficient compared with other preconditionings. However, Eisenstat SSOR preconditioning is not suited to parallel computation because it includes sequential process of computations as L^{-1} and U^{-1}. Moreover, we cannot avoid these substantial computations on parallel computers. In this paper, we propose a Cache-Cache Elements technique of Eisenstat SSOR preconditioning for parallelism, and demonstrate its effectiveness through numerical experiments.

1　はじめに

解くべき連立一次方程式を

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

とする．ただし，$A \in \mathbb{R}^{N\times N}$ を実非対称行列，$\boldsymbol{x} \in \mathbb{R}^N$ を解ベクトル，$\boldsymbol{b} \in \mathbb{R}^N$ を右辺ベクトルとする．この方程式を数値的にかつ高速で解きたいとき，一般に前処理行列を M とおくとき，

$$M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b} \tag{2}$$

と変換し，実際にはこの変換された方程式を元の方程式に戻して解くことが多い．本研究では，前処理の中でも特に高速な前処理としてよく知られたEisenstat-S(Symmetric)SOR (以下，E-SSORと呼ぶ) 前処理を扱うことにする [9] [7] [20].

E-SSOR前処理には，前進・後退代入計算が含まれ，この計算は計算順序に依存関係があるため，逐次計算では非常に高速であるが，並列計算では工夫が必要である．並列化手法としてよく知られている解法にはBlock分割法，Multi-Color ordering法 [3][4][22] や，岩下らの提案したAlgebraic Block Red-Black ordering(以下ABRBと略す)法，およびABRB法における色数を任意としたAlgebraic Block Multi Color (以下，ABMCと略す)法 [18] がある．また，染原らによるAlgebraic Multi Block(以下，AMBと略す)法 [23] は，ABMC法にブロックサイズの上限を設けることで負荷分散のバランスを改善した手法などもある．

AMB法などの並列化手法では，前進・後退代入の計算順序の逐次性を，行列の非零要素を前処理行列から棄却して，解消させる．非零要素の棄却は，一般に前処理行列の効果を弱めるが，計算結果そのものには何ら影響を与えない．しかし，E-SSOR前処理の場合では，(1) 式の等号関係が破綻し正しい解を求められない．したがって，E-SSOR前処理の並列化では，元のEisenstat版の算法を修正し，(1) 式の関係が保存される形にする必要がある．

本論文の目的は，E-SSOR 前処理を並列化が可能な形に修正したアルゴリズムを示すことと，並列版 E-SSOR 前処理を反復解法に適用し，その並列性能を明らかにすることである．本論文の構成は以下のとおりである．第 2 節で，E-SSOR 前処理のアルゴリズムについて記述する．第 3 節で，E-SSOR 前処理の並列化の準備を行う．第 4 節で，Cache-Cache Elements 法による並列化について記述する．第 5 節で，数値実験を通して，E-SSOR 前処理が並列効率が他の前処理に比べて非常に優れていることを明らかにする．第 6 節でまとめを行う．

2 Eisenstat-SSOR 前処理

Eisenstat 版の SSOR 前処理つき IDR(s)-R2 法について記述する [1][5][19]．(1) 式において，行列 A を

$$A = L + D + U \tag{3}$$

と分離する．ここで，L，D，U は，行列 A の狭義下三角行列，対角行列，狭義上三角行列を各々意味する．SSOR 前処理では，前処理行列 M を

$$M = (L + \frac{D}{\omega})(\frac{D}{\omega})^{-1}(U + \frac{D}{\omega}) \tag{4}$$

とする．ω は加速パラメータである．さらに，

$$\begin{aligned} M &= K_1 K_2, \\ K_1 &= (L + \frac{D}{\omega})(\frac{D}{\omega})^{-1},\ K_2 = (U + \frac{D}{\omega}) \end{aligned}$$

とおくと，

$$K_1{}^{-1} A K_2{}^{-1} = (\frac{D}{\omega})(L + \frac{D}{\omega})^{-1} A (U + \frac{D}{\omega})^{-1}, \tag{5}$$

$$K_2 \boldsymbol{x} = (U + \frac{D}{\omega})\boldsymbol{x}, \tag{6}$$

$$K_1 \boldsymbol{b} = (\frac{D}{\omega})(L + \frac{D}{\omega})^{-1}\boldsymbol{b}, \tag{7}$$

$$K_1 \boldsymbol{r} = (\frac{D}{\omega})(L + \frac{D}{\omega})^{-1}\boldsymbol{r} \tag{8}$$

となり，(1) 式は次のように変換される．

$$(\frac{D}{\omega})(L + \frac{D}{\omega})^{-1} A (U + \frac{D}{\omega})^{-1}(U + \frac{D}{\omega})\boldsymbol{x} = (\frac{D}{\omega})(L + \frac{D}{\omega})^{-1}\boldsymbol{b}. \tag{9}$$

(9) 式の両辺に左から $(\frac{D}{\omega})^{-1}$ をかけ次式を得る．

$$(L + \frac{D}{\omega})^{-1} A (U + \frac{D}{\omega})^{-1}(U + \frac{D}{\omega})\boldsymbol{x} = (L + \frac{D}{\omega})^{-1}\boldsymbol{b}. \tag{10}$$

(10) 式より，前処理後の係数行列 $\tilde{A}$，解ベクトル $\tilde{x}$，右辺項ベクトル $\tilde{b}$，および残差ベクトルは，

$$\tilde{A} = (L + \frac{D}{\omega})^{-1} A (U + \frac{D}{\omega})^{-1}, \tag{11}$$

$$\tilde{\boldsymbol{x}} = (U + \frac{D}{\omega})\boldsymbol{x}, \tag{12}$$

$$\tilde{\boldsymbol{b}} = (L + \frac{D}{\omega})^{-1}\boldsymbol{b}, \tag{13}$$

$$\tilde{\boldsymbol{r}} = (L+\frac{D}{\omega})^{-1}\boldsymbol{r} \tag{14}$$

と各々表される．このとき，前処理後の係数行列 $\tilde{A}$ を

$$\begin{aligned}\tilde{A} &= (L+\frac{D}{\omega})^{-1}A(U+\frac{D}{\omega})^{-1} \\ &= (U+\frac{D}{\omega})^{-1}+ \\ &\quad (L+\frac{D}{\omega})^{-1}(I+(1-2/\omega)D(U+\frac{D}{\omega})^{-1})\end{aligned} \tag{15}$$

と式変形すると，$\tilde{A}$ とベクトルの積を次の手順で計算することで計算量を削減することができる [2][5][27].

1. $\boldsymbol{y} = (U+\frac{D}{\omega})^{-1}\boldsymbol{v}$
2. $\boldsymbol{z} = \boldsymbol{v} + (1-\frac{2}{\omega})D\boldsymbol{y}$
3. $\boldsymbol{w} = (L+\frac{D}{\omega})^{-1}\boldsymbol{z}$
4. $\tilde{A}\boldsymbol{v} = \boldsymbol{y}+\boldsymbol{w}$

3 Eisenstat-SSOR前処理の並列化の準備

ここでは，Eisenstat trick による $\tilde{A}$ とベクトルの積計算の並列化を考える．係数行列 A を

$$A = \hat{L}+D+\hat{U}+R \tag{16}$$

と分離する．$\hat{L}$, $\hat{U}$ は係数行列の狭義下三角部分，狭義上三角部分のうちで Block 分割法等により並列化した場合に前処理に用いられる部分を各々意味する．残余行列 R (Remainder の頭文字) は $A=L+D+U$ と分離した場合の狭義下三角行列 L，狭義上三角行列 U の要素のうち，$\hat{L}$, $\hat{U}$ に含まれない要素の行列を意味する．

図 1 に，2 並列のときの行列 A(大きさ：8×8) に対する CCE 法の手順の概略を示す．また，残余行列 R について

$$R = R_U - R_L \tag{17}$$

の関係があり，$\hat{L}=L+R_L$, $\hat{U}=U-R_U$ とする．

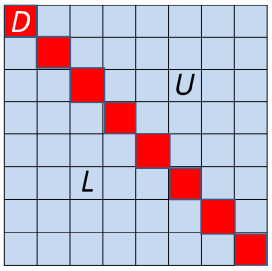
前処理行列は $\hat{L}$, D, $\hat{U}$ を用いて次のように構成する．

$$M = (\hat{L}+\frac{D}{\omega})(\frac{D}{\omega})^{-1}(\hat{U}+\frac{D}{\omega}) \tag{18}$$

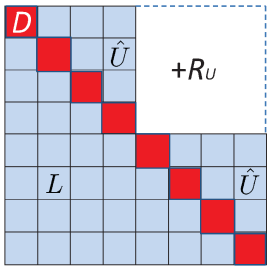
(5) 式から (14) 式までの式変形と同様に，前処理後の係数行列 $\tilde{A}$，解ベクトル $\tilde{x}$，右辺項ベクトル $\tilde{b}$，および残差ベクトルを各々，

$$\begin{aligned}\tilde{A} &= (\hat{L}+\frac{D}{\omega})^{-1}A(\hat{U}+\frac{D}{\omega})^{-1}, \\ \tilde{\boldsymbol{x}} &= (\hat{U}+\frac{D}{\omega})\boldsymbol{x}, \\ \tilde{\boldsymbol{b}} &= (\hat{L}+\frac{D}{\omega})^{-1}\boldsymbol{b}, \\ \tilde{\boldsymbol{r}} &= (\hat{L}+\frac{D}{\omega})^{-1}\boldsymbol{r}\end{aligned}$$

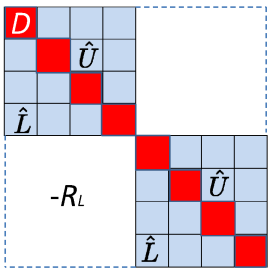
とおく．E-SSOR 前処理を逐次計算で用いる場合は，残余行列 R は不要である．



(a) 元の行列 $A = L + D + U$



(b) 上三角行列中の R_U 部分と上三角行列の残り $\hat{U}$



(c) 下三角行列中の R_L 部分と下三角行列の残り $\hat{L}$

図 1: 2 並列のときの行列 A(大きさ：8×8) に対する CCE 法の手順の概略.

4 Cache-Cache Elements法

ここでは，前節の導入に基づきE-SSOR前処理の並列化を考える．このとき，(18)式の前処理行列を用いると(15)式における式変形が成り立たない．そこで，残余行列Rを用いると，(15)式の式の変形を(18)式の前処理行列を用いることができる．このとき，(15)式における前処理後の係数行列などの式は次のように代わる．

$$\begin{aligned}\tilde{A} &= (\hat{L}+\frac{D}{\omega})^{-1}A(\hat{U}+\frac{D}{\omega})^{-1},\\ &= (\hat{L}+\frac{D}{\omega})^{-1}(\hat{L}+\frac{D}{\omega}+\hat{U}+\frac{D}{\omega}+D-\frac{2D}{\omega}+R)\times(\hat{U}+\frac{D}{\omega})^{-1}\\ &= (\hat{U}+\frac{D}{\omega})^{-1}+(\hat{L}+\frac{D}{\omega})^{-1}(I+(1-2/\omega)D\times(\hat{U}+\frac{D}{\omega})^{-1}+R(\hat{U}+D/\omega)) \qquad (19)\end{aligned}$$

(19)式より，$\tilde{A}$とベクトルの積は次の手順で計算できる．

$$\begin{aligned}\text{手順 1.}\quad \boldsymbol{y} &= (\hat{U}+\frac{D}{\omega})^{-1}\boldsymbol{v}\\ &= (U-R_U+\frac{D}{\omega})^{-1}\boldsymbol{v}\\ \text{手順 2.}\quad \boldsymbol{z} &= \boldsymbol{v}+(1-\frac{2}{\omega})D\boldsymbol{y}\\ \text{手順 3.}\quad \boldsymbol{u} &= R\boldsymbol{y}=(R_U-R_L)\boldsymbol{y}\\ \text{手順 4.}\quad \boldsymbol{w} &= (\hat{L}+\frac{D}{\omega})^{-1}(\boldsymbol{z}+\boldsymbol{u})\\ &= (L+R_L+\frac{D}{\omega})^{-1}(\boldsymbol{z}+\boldsymbol{u})\\ \text{手順 5.}\quad \tilde{A}\boldsymbol{v} &= \boldsymbol{y}+\boldsymbol{w}\end{aligned}$$

図1に対応して，以下ではCache-Cache Elements法の手順の概略を記述する．

Cache-Cache Elements法の手順の概略

元の行列A中の要素のうち，残余行列R_Uに含まれる要素(elements)を隠し，上記の手順1の$(\hat{U}+\frac{D}{\omega})^{-1}\boldsymbol{v}$の計算を行う．そして，隠していた残余行列$R_U$の要素は手順3で復元する．次に，同じ手順3で残余行列$-R_L$の要素を隠し，手順4で復元する．

このような「隠して見つける，また隠して見つける」という手順が，子供の遊び「かくれんぼ」(フランス語のCache-Cache[16]，カシュカシュと発音)に似ていることから，E-SSOR前処理に対する並列化手法を**Cache-Cache Elements**(以下では，CCEと略す)法と本論文では呼ぶことにする[7][12][13][14][15].

5 数値実験

5.1 計算機環境と計算条件

計算機環境と計算条件を以下に示す．CPU：Intel Xeon X5570（2.93GHz，8MB L3 Cache．4cores）×2，メモリ：24GB，OS：RedHat Enterprize Linux 5.2を使用した．プログラム：Fortran90，コンパイラ：Portland Group Fortran 90 compiler ver.10.5. 最適化オプションは"-O3"を使用した．計算はすべて倍精度浮動小数点演算で行い，時間計測には時間計測関数gettimeofdayを用いた．E-SSOR前処理の並列化は

OpenMP を用いて行った．反復解法には，収束性に優れた IDR(s)-SOR 法 [6] を用いたが，BiCGSafe 法などの Krylov 部分空間法であれば，E-SSOR 前処理はすべての解法に適用可能である．ここで使用した IDR(s)-SOR 法では，計算を効率よく行うために，2 段階の収束判定が行われる．すなわち，連立 1 次方程式に対する ϵ_d による収束判定は，通常のように，相対残差の 2 ノルム：$\|\boldsymbol{r}_{k+1}\|_2/\|\boldsymbol{r}_0\|_2 \leq 10^{-8}$ で行い，その内部計算では，$\tilde{\boldsymbol{r}}$ と ϵ による判定を変換残差の 2 ノルム：$\|\tilde{\boldsymbol{r}}_{k+1}\|_2/\|\tilde{\boldsymbol{r}}_0\|_2 \leq 10^{-6}$ として，残差の変換に要する時間を節約した．また，内側の収束判定が満された後の収束判定は反復 5 回毎に行った．方程式の右辺項には物理条件から得られる値を用いた．初期近似解 $\boldsymbol{x}_0$ はすべて 0，最大反復回数は 10000 回とした．行列は予め対角スケーリングによって対角項をすべて 1 に正規化した．次数 s は 2，4，8 の 3 通り，加速係数 ω は 0.4 から 1.6 まで 0.2 刻みで 7 通り調べた．スレッド (Threads) 数は 1，2，4，8 の 4 通りとした．

5.2 テスト行列

表 1 にテスト行列 10 個の主な特徴を示す．表中の “平均非零要素数” は係数行列の 1 行当りの平均非零要素数を意味する．行列 air-cfl5 はマンチェスター大学 F. Costen 研究室から提供を受けた．他の行列はフロリダ大学の疎行列データベース [28] から選出した．

表 1: テスト行列 (10 個) の主な特徴．

No.	行列	次元数	総非零要素数	平均非零要素
1	air-cfl5	1,536,000	19,435,428	12.65
2	matrix_9	103,430	1,205,518	11.66
3	epb1	14,743	95,053	6.45
4	epb2	25,228	175,027	6.94
5	epb3	84,617	463,625	5.48
6	language	399,130	1,216,334	3.05
7	trans4	116,835	749,800	6.42
8	wang4	26,068	177,196	6.80
9	xenon1	48,600	1,181,120	24.30
10	xenon2	157,464	3,866,688	24.56

5.3 実験結果

表 3 と表 4 に逐次の場合，および CCE 法，ABMC 法，AMB 法を用いて並列化した IDR(s)-R2 法の各スレッド数に対する収束性を示す．表中の時間単位はすべて秒とする．表中の “th.” はスレッド (threads) 数，“並替時間” は行列の並べ替えに要した時間，“台数効果” は 1 スレッドの場合の合計時間に対する各合計時間の比を各々意味する．“棄却率” は，係数行列の非零要素数 nnz，前処理に用いられなかった非零要素数を nnz_R とするとき，nnz_R/nnz で与えられる値を意味する．棄却率の値が 0 に近い程並列化による安定性の低下が小さくなると考えられる．“負荷分散” は i 番目のスレッドに割り当てられた非零要素数を tnz_i とするとき，$\min_i(tnz_i)/\max_i(tnz_i)$ で与えられる値を意味する．“負荷分散” の値が 1 に近い程負荷分散のバランスがとれていることを意味し，逆に値が 0 の場合は計算をしないスレッドが存在することを意味する，“負荷分散” の値が 0 の場合には，“負荷分散” の欄に括弧つきで実際に要素が割り当てられたスレッド数を記載した．“**TRR**” は近似解 $\boldsymbol{x}_{n+1}$ に対する真の相対残差 (True Relative Residual)，すなわち，$||\boldsymbol{b}-A\boldsymbol{x}_{k+1}||_2/||\boldsymbol{b}-A\boldsymbol{x}_0||_2$ の常用対数: $\log_{10}(\cdot)$ の値を意味する．太字の数字は，8 並列のとき，合計時間が最も少なかったことを表す．

表 3 と表 4 から，以下の観察が得られる．

1. 10 ケース中で CCE 法が最速結果を示した行列は，air-cfl5，epb1，epb2，epb3，language，trans4，wang4，xenon1，xenon2 の合計 9 ケースであった．
2. AMB 法が最速結果を示した行列は，行列 matrix_9 の僅か 1 ケースのみであった．ABMC が最速のケースはなかった．

並列化による効果について，以下の観察が得られる．

1. 並列化の効果が大きかった行列は，matrix_9，epb3，wang4，xenon1，xenon2 の 6 個であった．これらの行列はバンド行列であった．
2. 行列 air-cfl5 については，棄却率や負荷分散の値が良いにも関わらず ABMC 法，AMB 法による高速化の効果が少なかった．これは要素の並替に時間がかかったためである．
3. 特に，行列 matrix_9 や epb3 などの行列では，どの並列化手法においても並列化効果が大きかった．

解法ごとの特徴として，以下の知見が得られる．

1. CCE 法は ABMC や AMB と比較して棄却率の値が大きい場合が多く反復回数が多いが，高速であるケースが多く，行列の並替時間がないことなどが寄与していると考えられる．
2. ABMC 法は，CCE 法や AMB 法と比べて棄却率の値が小さく，反復回数が少なかった．しかし，ABMC 法の負荷分散のバランスが悪いケースが多く，収束までの時間は CCE 法や AMB 法の方が速く収束した．

全体的に，CCE 法による並列化は，ABMC 法や AMB 法による並列化よりも収束が速いケースが多かった．この理由として，ABMC 法や AMB 法は，元々ILU (tolerance：棄却用閾値) のようなフィルインを考慮した前処理向きの手法であったと考えられる．また，CCE 法は同期や行列要素の並替が不要であることも CCE 法が速いのに寄与したと思われる．

表 2 に，CCE 法，ABMC 法，AMB 法による並列化における台数効果の総括を示す．表中の最上行の番号は，表 1 に示した行列の番号である．表の最右欄は 10 個のテスト行列に対する各法の平均台数効果を示す．CCE 法の平均台数効果が **3.27** 倍と調べた並列化技法の中で最も高いことがことがわかる．

表 2: 8 並列のときの CCE 法，ABMC 法，AMB 法による並列化における台数効果の総括．

方法	表 1 中の行列番号										平均台
	1	2	3	4	5	6	7	8	9	10	数効果
CCE	2.08	4.81	2.68	2.44	5.57	1.79	2.24	3.36	3.85	3.84	**3.27**
ABMC	1.22	4.03	1.93	1.83	2.85	0.74	1.49	2.36	2.44	3.03	2.19
AMB	1.34	5.79	2.38	1.83	3.08	0.80	1.39	2.87	2.23	3.15	2.49

6 まとめ

Eisenstat-SSOR 前処理の並列化に Cache-Cache Elements 法を提案した．数値実験において，CCE 法，ABMC 法，AMB 法の 3 種類の並列化手法を用いて，非対称行列用の反復法の並列性能の評価を行った．その結果，提案した CCE 法は，他の並列化手法並列性能に非常に優れていることがわかった．

参考文献

[1] Axelsson, O.: Iterative Solution Methods, Cambridge University Press, 1994.

[2] Chan, T. F., van der Vorst, H. A.: Approximate and Incomplete Factorizations, in David E. Keyes, Ahmed Samed and V. Venkatakrishnan(eds), Parallel Numerical Algorithms, 1997.

[3] Duff, I. S., van der Vorst, H. A.: Preconditioning and Parallel Preconditioning, RAL Technical Reports, RAL-TR-1998-052, Rutherford Appleton Laboratory, 1998.

[4] Duff, I. S., van der Vorst, H. A.: Developments and trends in the parallel solution of linear systems, Parallel Computing 25, pp.1931-1970, 1999.

[5] Eisenstat, S. C.: Efficient implementation of a class of preconditioned conjugate gradient methods, SIAM J. Sci. Stat. Compute., Vol.2, No.1, pp.1-4, 1981.

[6] 藤野清次, P. Sonneveld, 尾上勇介, M. van Gijzen：IDR(s)-SOR の提案, A proposal of IDR(s)-SOR method, 日本応用数理学会論文誌, Vol.20, No.4, pp.289-308, 2010.

[7] 藤野清次, 阿部邦美, 中嶋徳正, 杉原正顯：計算力学レクチャーコース 線形方程式の反復解法, 丸善出版, 9 月, 2013.

[8] 藤野清次, 伊東千晶：Cache-Cache(カシュカシュ)Balancing による IDR(s)-SOR 法の並列化, 石垣島 CME ワークショップ 予稿集, pp.42-47, 5 月, 2014.

[9] 東慶幸, 藤野清次, 尾上勇介：Eisenstat 版 GS 型前処理付き MRTR 法の収束性について, 日本計算工学会論文集, No.20110006, 2011.

[10] 井上明彦：オーダリングによってブロック数を増した ABRB 手法に基づく VRIC(ω)-CG 法の並列化, 九州大学大学院システム情報科学府修士論文, 3 月, 2005.

[11] 井上明彦, 藤野清次：フィルインの選択に基づく改良版 ABRB 順序付け法による ICCG 法の並列化, 情報処理学会論文誌:コンピューティングシステム, Vol.46, No.SIG16(ACS12), pp.119-128, 2005.

[12] 伊東千晶, 藤野清次：Cache(カシュ) に関する小文, 石垣島 CME ワークショップ 予稿集, pp.23-28, 5 月, 2014.

[13] 伊東千晶, 藤野清次：日本計算工学会論文集： 2014 年 10 月 7 日 Web 公開 https://www.jstage.jst.go.jp /browse/jsces/-char/ja/

[14] 伊東千晶, 東慶幸, 藤野清次：Cache-Cache(カシュカシュ)Balance による拡張セカント法に基づく IDR(s) 法の並列化と性能評価, 日本シミュレーション学会, 2014. (印刷中)

[15] 伊東千晶, 藤野清次：A Review on Cache in French in view of Engineering, Transaction of Information, No.10, 2014. (印刷中)

[16] Petit Dictionaire Francais-Japonais Royal：プチ・ロワイヤル仏和辞典, 旺文社, 1986.

[17] Iwashita, T., Shimasaki, M.: Algebraic Multicolor Ordering for Parallelized ICCG Solver in Finite-Element Analyses, IEEE Trans. Magn., Vol.38-2, pp.429-432, 2002.

[18] Iwashita, T., Shimasaki, M.: Algebraic Block Red-Black Ordering Method for Parallelized ICCG Solver with Fast Convergence and Low Communication Costs, IEEE Trans. Magn., Vol.39-3, pp.1713-1716, 2003.

[19] 日下部雄三, 藤野清次, 春松正敏：Sonnevelt 型 SOR 法 vs. 古典的 SOR 法, 九州大学大学院システム情報科学紀要, Vol.14, No.2, pp.71-76, 2009.

[20] 村上啓一, 藤野清次, 尾上勇介, 平良賢剛：メモリアクセスの視点からの Eisenstat 版前処理の考察, 日本シミュレーション学会論文誌, Vol.3, No.2, pp.36-47, 2011.

[21] 野寺 隆：大型疎行列に対する PCG 法, SEMINER ON MATHEMATICAL SCIENCES, No.7, 1983.

[22] Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd Edition, SIAM, Philadelphia, 2003.

[23] 染原一仁, 藤野清次：代数マルチブロック技法による ICCG 法の並列性能の向上, 情報処理学会論文誌: コンピューティングシステム, Vol.47, No.SIG18(ACS16), pp.21-30, 2006.

[24] 染原一仁：実対称正定値行列に対する固有値密集化前処理技法, 九州大学大学院システム情報科学府修士論文, 2008.3.

[25] Sonneveld, P., van Gijzen, M. B., IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems, SIAM J. Sci. Stat. Compute., Vol.31, No.2, pp.1035-1062, 2008.

[26] 杉原正顯, 室田一雄：線形計算の数理, 岩波書店, 東京, 2009.

[27] 高橋秀俊, 野寺 隆:PCG アルゴリズムの効果的な実現の一考察, 情報処理学会第 24 回全国大会講演集, pp.901-902, 1982.

[28] University of Florida Spares Matrix Collection: http://www.cise.ufl.edu/research/sparse/matrices/ index.html

表 3: CCE 法と他の並列化手法との性能比較.

行列	並列化手法	th.	色数	s	ω	反復回数	並替時間	前処理時間	反復時間	合計時間	台数効果	棄却率 [%]	負荷分散	TRR
air-cfl5	逐次	1	-	2	0.8	35	-	0.071	4.945	5.016	1.00	-	-	-8.31
	CCE	2	-	2	0.8	60	-	0.188	3.167	3.355	1.50	40.9	0.99	-8.01
		4	-	2	0.8	80	-	0.158	2.877	3.035	1.65	61.4	0.99	-8.46
		8	-	2	0.8	85	-	0.157	2.249	**2.406**	2.08	61.6	0.97	-8.84
	ABMC	2	8	2	0.8	45	1.629	0.167	2.811	4.607	1.09	2.7	0.78	-8.00
		4	8	2	0.8	50	1.720	0.150	2.290	4.160	1.21	3.6	0.79	-8.72
		8	8	2	0.8	50	1.936	0.144	2.040	4.120	1.22	1.3	0.79	-8.56
	AMB	2	8	2	0.8	50	1.620	0.175	3.048	4.842	1.04	4.0	0.99	-8.35
		4	8	2	0.8	55	1.735	0.150	2.460	4.344	1.15	4.0	0.98	-9.21
		8	8	2	0.8	40	1.935	0.147	1.660	3.742	1.34	0.6	0.97	-8.48
matrix_9	-	1	-	4	0.6	1,015	-	0.007	8.917	8.924	1.00	-	-	-8.76
	CCE	2	-	4	0.6	865	-	0.017	3.708	3.725	2.40	1.3	0.99	-8.13
		4	-	4	0.6	820	-	0.014	2.288	2.301	3.88	3.8	0.98	-8.01
		8	-	2	0.6	1,110	-	0.015	1.839	1.854	4.81	8.5	0.95	-8.55
	ABMC	2	4	4	0.6	810	0.081	0.017	3.650	3.748	2.38	4.6	0.99	-8.03
		4	4	4	0.6	590	0.106	0.015	1.903	2.024	4.41	10.0	0.71	-8.00
		8	4	4	0.6	590	0.176	0.022	2.015	2.213	4.03	16.1	0.0(6)	-7.92
	AMB	2	4	4	0.6	810	0.083	0.018	3.676	3.777	2.36	4.6	0.99	-8.03
		4	4	4	0.6	645	0.101	0.015	1.939	2.055	4.34	9.9	0.98	-8.23
		8	4	4	0.6	575	0.122	0.014	1.405	**1.541**	5.79	17.3	0.95	-8.44
epb1	逐次	1	-	4	1.2	185	-	0.000	0.114	0.114	1.00	-	-	-8.66
	CCE	2	-	4	1.2	185	-	0.001	0.061	0.063	1.81	0.4	1.00	-8.44
		4	-	4	1.2	185	-	0.001	0.050	0.052	2.21	1.1	0.99	-8.45
		8	-	4	1.2	190	-	0.001	0.041	**0.043**	2.68	2.6	0.98	-8.39
	ABMC	2	2	4	1.2	180	0.003	0.001	0.064	0.068	1.68	0.3	1.00	-8.19
		4	2	4	1.2	205	0.004	0.001	0.054	0.059	1.93	2.3	0.99	-8.35
		8	2	4	1.2	245	0.004	0.001	0.054	0.059	1.93	6.3	0.98	-8.90
	AMB	2	2	4	1.2	180	0.003	0.001	0.075	0.079	1.44	0.3	1.00	-8.19
		4	2	4	1.2	200	0.003	0.001	0.051	0.055	2.07	2.3	0.99	-8.15
		8	2	4	1.2	225	0.004	0.001	0.043	0.048	2.38	6.3	0.98	-8.09
epb2	逐次	1	-	4	1.2	70	-	0.000	0.088	0.088	1.00	-	-	-8.50
	CCE	2	-	2	1.2	100	-	0.003	0.064	0.067	1.31	0.4	0.97	-9.53
		4	-	2	1.2	85	-	0.002	0.034	0.037	2.38	1.4	0.88	-8.19
		8	-	2	1.2	95	-	0.002	0.034	**0.036**	2.44	3.0	0.76	-8.16
	ABMC	2	2	4	1.2	90	0.005	0.002	0.059	0.066	1.33	0.5	0.97	-8.28
		4	2	4	1.2	100	0.006	0.002	0.042	0.050	1.76	2.5	0.88	-8.12
		8	2	4	1.2	140	0.007	0.002	0.040	0.048	1.83	10.6	0.72	-8.21
	AMB	2	2	4	1.2	90	0.006	0.002	0.058	0.066	1.33	0.5	0.97	-8.28
		4	2	4	1.2	100	0.005	0.002	0.043	0.050	1.76	2.5	0.88	-8.12
		8	2	4	1.2	140	0.006	0.002	0.039	0.048	1.83	10.6	0.72	-8.21
epb3	逐次	1	-	4	1.0	375	-	0.002	1.797	1.799	1.00	-	-	-8.41
	CCE	2	-	8	1.0	230	-	0.009	0.700	0.709	2.54	0.1	1.00	-8.14
		4	-	4	1.0	360	-	0.007	0.413	0.421	4.27	0.3	1.00	-8.13
		8	-	4	1.0	360	-	0.008	0.315	**0.323**	5.57	0.6	1.00	-8.06
	ABMC	2	2	4	1.0	395	0.017	0.006	0.802	0.825	2.18	0.1	1.00	-8.02
		4	2	4	1.0	390	0.017	0.005	0.484	0.506	3.56	0.6	1.00	-8.03
		8	2	4	1.0	540	0.026	0.006	0.600	0.632	2.85	1.5	1.00	-8.10
	AMB	2	2	4	1.0	395	0.016	0.007	0.799	0.822	2.19	0.1	1.00	-8.02
		4	2	4	1.0	415	0.016	0.005	0.514	0.536	3.36	0.6	1.00	-8.15
		8	2	4	1.0	540	0.018	0.005	0.562	0.585	3.08	1.5	1.00	-8.12
language	逐次	1	-	2	1.0	15	-	0.011	0.325	0.336	1.00	-	-	-13.17
	CCE	2	-	2	1.0	20	-	0.020	0.241	0.261	1.29	15.6	0.63	-11.25
		4	-	2	0.8	25	-	0.019	0.192	0.210	1.60	29.7	0.48	-8.62
		8	-	2	0.8	30	-	0.018	0.170	**0.188**	1.79	48.3	0.40	-9.01
	ABMC	2	2	2	1.0	20	0.132	0.025	0.259	0.415	0.81	13.8	0.69	-9.18
		4	2	2	1.0	25	0.146	0.027	0.226	0.400	0.84	24.5	0.65	-8.78
		8	2	2	1.0	30	0.160	0.029	0.269	0.457	0.74	26.8	0.00	-12.60
	AMB	2	2	2	1.0	20	0.131	0.024	0.257	0.412	0.82	13.8	0.69	-9.18
		4	2	2	1.0	25	0.146	0.027	0.223	0.396	0.85	24.5	0.65	-8.78
		8	2	2	1.0	25	0.207	0.025	0.189	0.421	0.80	30.0	0.61	-8.01

表 4: CCE 法と他の並列化手法との性能比較 (続き).

行列	並列化手法	th.	色数	s	ω	反復回数	並替時間	前処理時間	反復時間	合計時間	台数効果	棄却率 [%]	負荷分散	TRR
trans4	逐次	1	-	4	1.2	145	-	0.003	0.933	0.936	1.00	-	-	-8.91
	CCE	2	-	4	1.0	205	-	0.010	0.783	0.794	1.18	22.6	0.64	-8.44
		4	-	4	1.0	205	-	0.008	0.481	0.489	1.91	33.2	0.47	-8.04
		8	-	4	1.0	215	-	0.009	0.409	**0.418**	2.24	39.5	0.30	-8.02
	ABMC	2	4	4	1.2	150	0.033	0.012	0.641	0.687	1.36	0.0	0.01	-8.10
		4	4	4	1.2	150	0.037	0.012	0.574	0.623	1.50	0.0	0.01	-8.15
		8	4	4	1.2	155	0.034	0.012	0.582	0.629	1.49	0.0	0.01	-8.32
	AMB	2	2	4	1.0	225	0.035	0.010	0.782	0.827	1.13	11.3	0.64	-8.05
		4	2	4	1.0	250	0.041	0.010	0.622	0.673	1.39	18.6	0.46	-8.32
		8	2	4	1.0	260	0.052	0.010	0.613	0.675	1.39	23.3	0.30	-8.25
wang4	逐次	1	-	4	1.0	120	-	0.001	0.157	0.158	1.00	-	-	-8.74
	CCE	2	-	4	1.2	115	-	0.002	0.081	0.084	1.88	1.1	1.00	-8.26
		4	-	4	1.0	125	-	0.002	0.053	0.055	2.87	3.2	0.98	-8.57
		8	-	4	1.0	145	-	0.002	0.045	**0.047**	3.36	7.4	0.96	-8.44
	ABMC	2	4	4	1.0	110	0.005	0.002	0.074	0.081	1.95	0.0	1.00	-8.24
		4	4	4	1.0	110	0.007	0.002	0.045	0.054	2.93	0.0	0.98	-8.03
		8	4	4	1.0	185	0.013	0.003	0.051	0.067	2.36	15.4	0.29	-8.12
	AMB	2	2	4	1.0	115	0.005	0.002	0.071	0.078	2.03	0.0	1.00	-8.08
		4	2	4	1.0	130	0.005	0.002	0.053	0.060	2.63	4.2	0.98	-8.54
		8	2	4	1.0	180	0.007	0.002	0.046	0.055	2.87	12.6	0.96	-8.37
xenon1	逐次	1	-	8	1.0	1,495	-	0.004	8.138	8.142	1.00	-	-	-8.06
	CCE	2	-	8	1.0	1,570	-	0.012	4.428	4.441	1.83	1.8	0.99	-8.03
		4	-	8	1.0	1,795	-	0.009	3.363	3.372	2.41	5.4	0.97	-8.15
		8	-	8	1.0	1,510	-	0.008	2.105	**2.114**	3.85	8.3	0.96	-8.02
	ABMC	2	4	8	1.0	1,645	0.056	0.010	4.474	4.540	1.79	3.1	1.00	-8.08
		4	4	8	1.0	1,720	0.060	0.008	3.136	3.205	2.54	11.2	0.98	-8.10
		8	4	8	1.0	2,075	0.065	0.009	3.258	3.331	2.44	22.6	0.15	-8.14
	AMB	2	4	8	1.0	1,645	0.055	0.010	4.529	4.594	1.77	3.1	1.00	-8.08
		4	4	8	1.0	1,665	0.060	0.008	3.052	3.121	2.61	11.2	0.98	-8.04
		8	4	8	1.0	2,355	0.066	0.009	3.579	3.653	2.23	27.0	0.98	-8.23
xenon2	逐次	1	-	8	1.0	2,510	-	0.013	46.047	46.060	1.00	-	-	-8.03
	CCE	2	-	8	1.0	2,275	-	0.028	18.577	18.604	2.48	0.9	1.00	-8.01
		4	-	8	1.0	2,455	-	0.025	13.154	13.179	3.49	2.6	0.98	-8.08
		8	-	8	1.0	2,755	-	0.023	11.987	**12.010**	3.84	6.2	0.97	-8.02
	ABMC	2	4	8	1.0	2,420	0.183	0.040	23.137	23.360	1.97	0.8	1.00	-8.08
		4	4	8	1.0	2,430	0.196	0.028	15.122	15.346	3.00	4.0	0.98	-8.17
		8	4	8	1.0	2,930	0.213	0.027	14.950	15.190	3.03	10.6	0.99	-8.21
	AMB	2	4	8	1.0	2,420	0.181	0.040	23.135	23.355	1.97	0.8	1.00	-8.08
		4	4	8	1.0	2,365	0.195	0.028	14.457	14.680	3.14	4.0	0.98	-8.06
		8	4	8	1.0	2,830	0.214	0.027	14.392	14.633	3.15	10.6	0.99	-8.26