Vol.2009-HPC-122 No.2
2009/10/9

Localized IC 分解と多色順序付けを併用したハイブリッド型並列 ICCG 法に関する検討

福原敏行[†]　高橋康人[†]　岩下武史[††]　中島浩[††]

本論文では，分散メモリ形式に対応した並列化 ICCG ソルバを開発した．局所 IC 分解前処理と多色順序付け法を併用し，OpenMP と MPI の両者を用いるハイブリッド並列化を可能とした．3 種の数値実験例において，Flat MPI による並列化を行った場合とハイブリッド並列処理を用いた場合の比較検討を実施した．数値実験の結果，いずれの数値例においてもハイブリッド並列方式が Flat MPI より良好な結果を示し，最大で２倍以上の性能差があったことを報告する．

Analysis of Hybrid-type Parallel ICCG Method Based on Localized IC Factorization and Multi-color Ordering

TOSHIYUKI FUKUHARA[†]　YASUHITO TAKAHASHI[†]
TAKESHI IWASHITA [††]　HIROSHI NAKASHIMA [††]

This paper reports our development of a parallel ICCG solver for distributed memory parallel computers. We utilize localized IC preconditioning and multi-color ordering for hybrid parallel processing approach in which both OpenMP and MPI are used. Three numerical tests were conducted to examine the performance of hybrid parallel programming model against that of Flat MPI. It is shown that the hybrid type solver attains computational performance up to more than twice that of Flat MPI.

[†] 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University
[††] 京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

1. はじめに

電磁場解析や構造解析等の数値シミュレーションでは，解析対象が複雑・大規模化しており，シミュレーションの高速化が要望されている．一方，今後，プロセッサのクロック数が大きく伸びることは期待できず，プロセッサのマルチコア化が進むと予測されている．また，複数のプロセッサを搭載した PC やそれらをクラスタ化した計算環境が広く普及しており，数値シミュレーションにおいて並列処理技術を有効に活用することが必要不可欠となってきている．

本論文では，数値シミュレーション分野でよく用いられる線形反復ソルバの一種である ICCG 法 [1]の並列化について報告する．ICCG 法は不完全コレスキー分解前処理付き共役勾配法の略称で，正値・対称な係数行列を持つ連立一次方程式の求解法として広く利用されている．しかしながら，ICCG 法は並列化において阻害要因となる前進・後退代入計算を含んでおり，並列化に際してなんらかの工夫が必要となる．本論文では，係数行列のデータが分散メモリ上に保持されている場合を対象とし，分散メモリ形式に対応したブラックボックス型の並列化 ICCG ソルバについて検討する．まず，分散メモリに対応した並列処理として MPI によるマルチプロセス並列処理を行う．ここで，代入計算をマルチプロセス並列処理するために，局所 IC 分解前処理を用いる．同前処理はプロセス数が増加するにつれてその前処理効果が低下する問題点があるが，代入計算に通信を必要とすることなく並列化できる長所がある．

次に，マルチプロセス並列処理とマルチスレッド並列処理を併用するハイブリッド並列処理方式を活用することで，ソルバを高性能化する手法を検討する．MPI 通信によるマルチプロセス並列処理のみを使用する Flat MPI 方式と比較した場合，ハイブリッド並列処理方式が優位となるかどうかは対象とする問題や計算環境に依存して一概には言えないが，ICCG 法の並列化に際しては共有メモリ並列方式のスレッド並列では分散メモリ方式と比べてより幅広い並列化手法の選択が可能となる [2]．そこで，一般的に局所 IC 分解よりも高い収束性が期待できる多色順序付け法をスレッド並列化手法として導入し，ハイブリッド並列処理によるソルバの性能改善を試みる．開発した並列化 ICCG ソルバを UF Matrix Collection DB[3] 上の 2 種の行列データと 3 次元差分解析により得られる行列データにより性能評価し，Flat MPI 方式とハイブリッド並列処理方式の比較を行う．

2. ICCG 法と並列化

ICCG 法は，n 元連立一次方程式

$$Ax = b \tag{1}$$

の係数行列 A が，対称正定値性を持ち，かつ疎行列である場合の解法として，一般的なものである．ICCG 法は，前処理付き共役勾配法の一種で，前処理行列として，係数行列 A の不完全コレスキー分解を用いるものである．

ICCG 法の計算は，主に，次の要素で構成される．

（1）行列・ベクトル積
（2）ベクトルの内積
（3）ベクトル（およびその実数倍）の加減
（4）前進・後退代入計算

このうち（1）～（3）の計算は，並列化が比較的容易である．分散メモリ環境での MPI によるプロセス並列を行う場合，後述する一般的なデータ分散方法では（1）の並列処理において MPI_ALLGATHER あるいは MPI_ALLGATHERV の通信が必要となり，（2）の並列処理に伴い MPI_ALLREDUCE の通信が必要となる．

一方，（4）の前進・後退代入計算は，計算で導出される要素間に依存関係が存在するため，並列化は一般的に困難で，なんらかの工夫が必要となる．本研究においては，プロセス並列/スレッド並列併用のハイブリッド並列プログラミングを導入している．つまり，前進・後退代入計算の並列化にあたっては，スレッド並列処理を行うために多色順序付け法を用い，プロセス並列処理を行うために前処理の局所化を行うことで，並列計算を可能にした．以下にこれらの前処理に関する並列処理の詳細について述べる．

3. IC 分解前処理の並列化（局所 IC 分解と多色順序付け法）

3.1 データ構造

係数行列 A は，分散メモリに対応した CRS 形式（Compressed Row Storage）[4]によりメモリ上に保持する．CRS 形式では，3 つの一元配列を利用し，それぞれに非零要素の列番号（整数），値（倍精度実数），列番号リストにおける各行の開始位置（整数）を格納する．ICCG 法で扱う係数行列は対称であるが，本研究では係数行列全体を格納するものとする．図 1 のように係数行列を行方向に分割し，各行ブロックを各プロセスが保持，担当するものとする．なお，係数行列データの分割はソルバを利用するユーザが行うものとしており，ソルバ内ではプロセス間通信を伴うデータ再配置は行わないこととする．疎行列・ベクトル積における負荷バランスの点では，各行ブロック内の非零要素数がなるべく均等化されるような分割が望ましい．

ソルバ内では係数行列に加えて，数種の一次元ベクトルが用いられるが，係数行列のブロック分割に対応する形で分割し，メモリ上に格納する．なお，CG 法における勾配ベクトルは各プロセスがその全体を使用するため，n 個の要素をもつ一元配列として格納する．

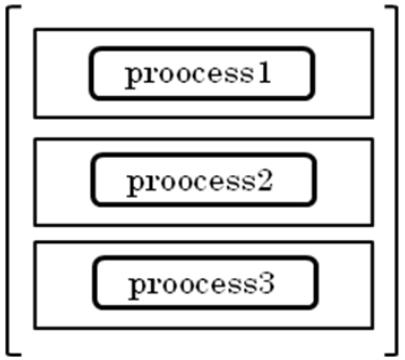

図 1　3 プロセス並列の場合
Fig 1 Case of 3 process Parallelization

3.2 局所不完全コレスキー分解 [4) 5)]

本研究では，IC 分解前処理の分散並列処理（プロセス並列処理）を可能にする手段として，局所不完全コレスキー分解を用いる．同手法は加法シュワルツ型の前処理の一種で，各部分領域と各プロセッサが保持する係数行列データを対応させる．また，部分領域内の解法として通常の不完全コレスキー分解（前処理）を用いる．以下にその手法について簡単に述べる．

プロセス ip が担当する未知変数(対応する行ブロックと考えてよい)を s_{ip}~e_{ip} とし，$n_{ip}=e_{ip}-s_{ip}+1$（担当行数）する．このとき，s_{ip}~e_{ip} 行に位置する対角要素が 1 でその他の全ての要素を 0 とする n 次元正方行列 P_{ip} を考える．局所不完全コレスキー分解前処理では，プロセッサ ip は $P_{ip}AP_{ip}$ を次のように不完全コレスキー分解する．

$$P_{ip}AP_{ip} \cong L_{ip}L_{ip}^{T} \tag{2}$$

ここで，L_{ip} は n 次元の下三角行列で，s_{ip}~e_{ip} の行・列の範囲内にのみ非零要素を持つ．このとき，当該前処理行列 M は

$$M = \sum_{ip} L_{ip}L_{ip}^{T} \tag{3}$$

のように書ける．反復内の前処理手順は

$$z = \left(\sum_{ip} L_{ip}^{T\ -1} L_{ip}^{\ -1} \right) r \qquad (4)$$

で与えられるが，各プロセスが行う処理（総和の各項の計算）は互いに独立であるために通信を必要とせず，並列に処理することができる．また実際には，残差ベクトル $\boldsymbol{r}$ も各プロセスが全体を持つ必要はなく s_{ip}~e_{ip} の行部分のみを保持するのみで計算が可能となる．図 2 に，局所コレスキー分解前処理で用いられる前処理行列の形状を示す．上述したように，同前処理の並列化は容易で通信も必要とされないため，並列処理への適合性が極めて高い．一方，図 2 に示すように，通常の逐次不完全コレスキー分解前処理と比較した場合，一部の非対角要素の影響を無視している．影響を無視した要素の数はプロセス数に比例して増加するため，一般的にプロセス数が増加すると前処理効果が低下する問題がある．

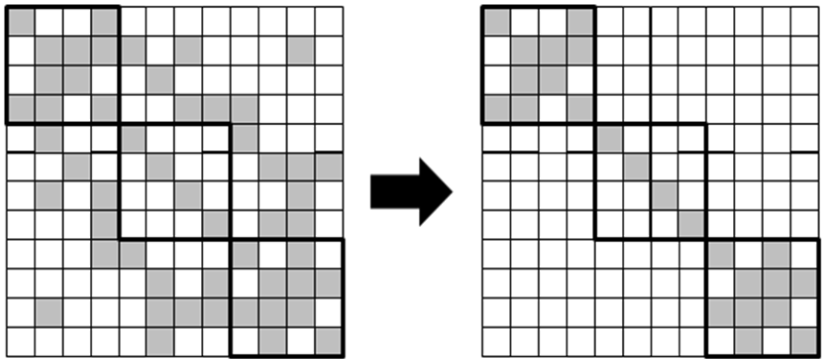

図 2　局所前処理イメージ
Fig 2 Image of Localized Preconditioning

3.3 多色順序付け

本研究では，前節で述べた局所不完全 IC 分解によるプロセス並列処理のみを行うソルバ(Flat MPI 方式)に加えて，ハイブリッド並列処理と呼ばれるマルチプロセス/マルチスレッド併用型の並列処理に対応したソルバもあわせて開発する．ハイブリッド並列処理を行う場合，各プロセスでマルチスレッド並列処理を行う必要があり，代入計算の並列化のために何らかの並列化手法を適用することが必要となる．ここで，最も簡便な方法としては,プロセス並列処理で使用した局所 IC 分解前処理をマルチスレッド並列処理にも使用することが考えられる．しかしこの簡易な手法では，前小節で述べたように，スレッド数の総和が増大すると前処理効果が低下する問題がある．そこで，より高い前処理効果を得るために，マルチスレッド並列処理手法として（代数）多色順序付け法を用いる．

多色順序付けとは，未知変数の依存関係を解析し，未知変数に色分けを行う手法である．具体的には，互いに依存関係のないものは同じ色を，あるものには異なる色を付けていく．すべての未知変数に色付けを終えると，次に同じ色同士が連続した位置になるように並び替えを行う．ここで，本研究では，各プロセスが保持している行ブロックに対応する未知変数に対して，多色順序付けを適用する．局所 IC 分解前処理に加えて各プロセス内で多色順序付けを実行した場合の前処理行列の非零要素パターンは，図 3 に示すような形になる．各プロセスにおいて同色の未知変数は互いに独立となり，前進・後退代入計算のマルチスレッド並列処理が可能となる．ただし，各プロセス内において各代入計算中に色数－1 回のスレッドの同期処理が必要となる．なお，多色順序付けに必要な未知変数間の依存関係の解析には Iwashita らが提案している代数多色順序付け法（AMC 法）を用いた．AMC 法の詳細は文献 6),7) に示されている．

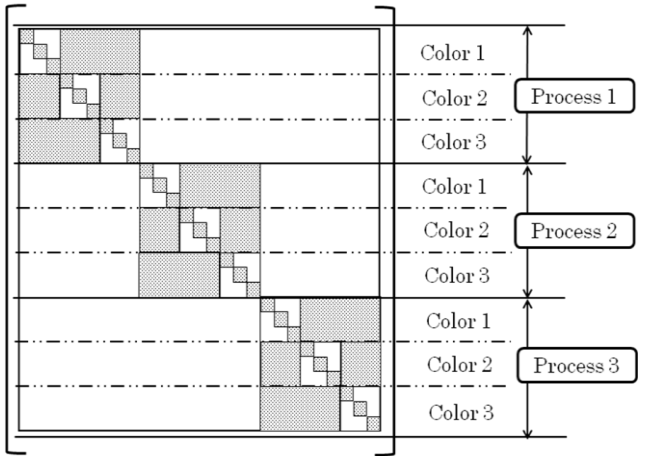

図 3　多色順序付けを併用した局所前処理
Fig 3 Localized Preconditioning with multi-color ordering

4. 数値実験結果

4.1 使用計算環境とテスト問題

開発したハイブリッド対応並列 ICCG 法の有効性を評価するために，数値実験を行

った．数値実験に使用する計算機は，京都大学学術情報メディアセンターの富士通 HX600[8] である．HX600 の各ノードは，4 個の AMD 社製クアッドコア Opteron プロセッサと 32GB（DDR2-667）のメモリを有している．当該計算機を最大 4 ノード使用して数値実験を行った．プログラミング言語として，C 言語を使用した．ICCG 法の収束判定基準は相対残差ノルムが 10^{-7} 以下となる条件とし，反復開始から残差ノルムが本基準を満たすまでの経過時間を計測した．

本数値実験の問題対象として，The University of Florida Sparse Matrix Collection (http://www.cise.ufl.edu/research/sparse/matrices/)より入手した連立一次方程式 2 種を用いる．表 1 に使用した行列データの諸元を示す．

また，ハイブリッド並列方式（以降，Hybrid と表記する）については，次の 3 種類のスレッド／プロセスの組み合わせを適用した．表記法については，文献 2) にならった．

- Hybrid 4×4 ：ノード内の各プロセッサに OpenMP スレッドを 4 つ，ノードあたり 4 つの MPI プロセス
- Hybrid 8×2 ：ノード内の 2 つのプロセッサに OpenMP スレッドを 8 つ，ノードあたり 2 つの MPI プロセス
- Hybrid 16×1：ノード内全体に OpenMP スレッドを 16，ノードあたり 1 つの MPI プロセス

4.2 UF matrix collection テスト問題による数値実験

表 2，3 に，parabolic_fem，及び thermal2 による数値実験結果を示す．ここで，Hybrid の結果は，多色順序付けの色数を 10，20，30 色として実験を行い，その中で最も計算時間が短かったものを示している．

表 2，表 3 を見ると，16 並列（単一ノード）の場合は，Flat MPI の方が優位となる場合が多いが，並列度が 32，64 と増すにつれ，Hybrid 方式が優位となっている．これは，MPI 通信コストによる影響が並列度の増加とともに顕著となっていることが原因だと考えられ，表内の一反復あたりの MPI 通信時間の列を見るとその様子が見て取れる．開発した並列 ICCG ソルバの MPI 通信において扱う送受信データ量が最も大きいのは MPI_ALLGATHERV による集合通信であるが，表 2，3 によると，反復あたりの MPI による通信時間はほぼプロセス数に依存し，プロセス数が増加するに従い増大している．その結果，複数ノードを使用した場合，上記のように相対的にプロセス数の少ない Hybrid が優位となったと考えられる．また，収束性については，両実験においてプロセス数が増加するに従い反復回数が増加する傾向がみられ，局所 IC 分解前処理で問題となる収束性の劣化が観測される．したがって，多色順序付けを併用することによりプロセス数の削減と収束性の改善を図ることができ，開発手法の有効性がわかる．

次に Hybrid 方式同士を比較すると，16 並列の場合は，4（スレッド）×4（プロセス）が一番速い結果となっている．これは，8×2，16×1 がプロセッサ間で同期をとらなければならないのに対し，4×4 の場合は，同期がプロセッサ内で完結しており，同期コストが他に比べ，低いためと考えられる．しかし，複数ノードを使用し，並列度が増すにつれ，16（スレッド）×1（プロセス）の実行形式が，最も計算が速い結果となっている．これは，先述した MPI 通信の影響によるものと考えられる．

4.3 3 次元ポアソン方程式の差分解析による数値実験

ポアソン方程式の 3 次元差分解析において生ずる連立一次方程式を対象として，開発手法の有効性を検証する．本実験では，ソルバの Weak scalability の観点からコアあたりの問題サイズを（$(200^3)/16$）に固定し，使用コア数に応じて問題サイズを調整した．表 4 に Hybrid 16×1 と Flat MPI による数値実験の結果を示す．ここで，表 4 内の準演算時間は，総計算時間から通信時間を引いたものとしている．

反復回数については，同一の問題サイズの場合（使用コア数が同一の場合）Flat MPI よりも Hybrid のほうが多い結果となっている．これは，応用分野において一般的に知られている傾向とは異なっているが，問題が比較的良条件であり，局所 IC 分解前処理による収束性の劣化がそれほど大きくなかったためと考えられる．なお，同様の傾向はポアソン方程式による 2 次元差分解析においても報告されている[9]．次に，経過時間から MPI による通信時間を差し引いた準演算時間を見比べると，Flat MPI のほうが相対的に速いという結果となっている．これは，Flat MPI の場合，プロセス数が多く，局所化により前進・後退代入計算時に無視する領域が Hybrid に比べ多いため，計算量が少なくなっていることが原因である．また，Hybrid におけるスレッド間の同期コストも原因として考えられる．

次に，一反復あたりの通信時間を見ると，Flat MPI の結果では，通信時間はプロセス数（問題サイズ）にほぼ比例して増加している．ここで，表 4 の(c)において問題を 200^3 に固定した場合の Flat MPI の結果を示すが，一反復あたりの通信時間はほとんどプロセス数に依存していない．このように，4.2 節で扱った問題と異なり，本テスト問題において Flat MPI 形式を用いたとき，通信時間はプロセス数ではなく問題サイズに依存している．

最後に，ソルバ全体の計算時間を比較すると，Flat MPI における，前述の前進・後退代入における計算量と反復回数の少なさの優位性を打ち消してしまうほど，MPI 通信の影響が大きく，結果として，全体の計算時間では Hybrid 方式の方が速いという結果となっている．

5. おわりに

本論文では，線形ソルバの一つである ICCG 法の並列化に，ハイブリッド並列プロ

グラミングを適用し，その並列化効果について Flat MPI と比較を行った．プロセス並列を実施するために前処理の局所化を，スレッド並列を実施するために多色順序付けを導入することで Hybrid 並列化を実現した．3 種の数値計算例で比較を行った結果，16 並列の場合は，Flat MPI の方が高い計算性能を示す場合が見られた．しかし，使用ノード数が増えるにつれ，Hybrid の方が高い計算性能を示した．Hybrid 方式同士を比べると，16 並列のときは，4（スレッド）×4（プロセス）が最も速いという結果が得られたが，64 並列の場合では，16×1 がいずれの計算例においても最も高い計算性能を示した．

謝辞　本研究の一部は，日本学術振興会 科学研究費補助金(基礎研究(B)，課題番号 20300011)の助成を受けている．

表 1　係数行列データ諸元
Table 1 Properties of coefficient problem

係数行列名	問題領域	次元数	非ゼロ要素数
parabolic_fem	計算流体力学	525,825	3,674,625
thermal2	定常熱問題	1,228,045	8,580,313

表 2　係数行列データ Parabolic_fem による実験結果
Table 2　Numerical results of parabolic_fem test problem

Programming model (thread×process)	並列数	反復回数	計算時間(sec)	一反復あたり計算時間(sec)	一反復あたり MPI 通信時間(sec)	色数
Flat MPI	16	1287	29.005	0.02254	0.0131	---
	32	1290	37.447	0.02903	0.0226	---
	64	1361	61.261	0.04501	0.0417	---
4×4	16	1360	26.132	0.01921	0.0046	30
	32	1277	21.574	0.01689	0.0090	30
	64	1287	22.916	0.01781	0.0136	30
8×2	16	1117	30.983	0.02774	0.0039	30
	32	1360	21.980	0.01616	0.0060	30
	64	1278	18.579	0.01454	0.0089	10
16×1	16	647	27.906	0.04313	--------	10
	32	1069	21.644	0.02025	0.0032	10
	64	1360	17.252	0.01269	0.0055	30

表 3　係数行列データ thermal2 による数値実験結果
Table 3　Numerical results of thermal2 test problem

Programming model (thread ×process)	並列数	反復回数	計算時間(sec)	一反復あたり計算時間(sec)	一反復あたり MPI 通信時間(sec)	色数
Flat MPI	16	2291	117.795	0.05142	0.0203	---
	32	2441	108.971	0.04464	0.0330	---
	64	2610	157.190	0.06023	0.0520	---
4×4	16	2114	126.506	0.05984	0.0107	10
	32	2258	90.543	0.04010	0.0152	30
	64	2318	81.699	0.03525	0.0209	30
8×2	16	2022	138.020	0.06826	0.0098	10
	32	2118	86.961	0.04106	0.0110	10
	64	2260	67.734	0.02997	0.0149	30
16×1	16	1474	119.389	0.08100	----------	10
	32	2023	90.518	0.04474	0.0064	10
	64	2115	58.243	0.02754	0.0093	10

表 4　ポアソン方程式による実験結果
Table 4　Numerical results of Poisson equation test problem
(a)Hybrid(16×1)による実験結果

問題サイズ	thread×process	色数	反復回数	計算時間 (sec)	一反復あたり計算時間 (sec)	MPI 通信時間 (sec)	一反復あたり通信時間 (sec)	準演算時間 (sec)	一反復あたり準演算時間 (sec)
200^3	16×1	8	173	45.817	0.265	---	---	45.817	0.265
		16	183	50.939	0.278	---	---	50.939	0.278
		32	188	52.660	0.280	---	---	52.660	0.280
252^3	16×2	8	293	99.087	0.338	16.562	0.0565	82.525	0.282
		16	271	92.283	0.341	15.391	0.0568	76.892	0.284
		32	277	97.536	0.352	15.723	0.0568	81.813	0.295
288^3	16×3	8	349	128.493	0.368	33.142	0.0950	95.351	0.273
		16	329	117.438	0.357	31.213	0.0949	86.225	0.262
		32	318	114.671	0.361	29.757	0.0936	84.914	0.267
317^3	16×4	8	414	178.811	0.432	54.170	0.1308	124.641	0.301
		16	410	174.636	0.426	48.880	0.1192	125.756	0.306
		32	387	169.302	0.437	50.765	0.1312	118.537	0.306

(b)Flat MPI による実験結果

問題サイズ	プロセス数	反復回数	計算時間 (sec)	一反復あたり計算演算時間(sec)	MPI 通信時間 (sec)	一反復あたり通信時間 (sec)	準演算時間 (sec)	一反復あたり準演算時間 (sec)
200^3	16 並列	173	63.336	0.366	36.174	0.209	27.162	0.1570
252^3	32 並列	235	141.397	0.602	103.870	0.442	37.527	0.1597
288^3	48 並列	250	204.669	0.819	164.625	0.659	40.044	0.1602
317^3	64 並列	323	339.373	1.051	284.369	0.880	55.004	0.1703

(c)問題サイズを 200^3 としたときの Flat MPI による実験結果

問題サイズ	プロセス数	反復回数	計算時間(sec)	一反復あたり計算時間(sec)	一反復あたり通信時間(sec)
200^3	16 並列	173	63.336	0.3661	0.209
	32 並列	191	58.305	0.3053	0.221
	48 並列	199	56.387	0.2834	0.221
	64 並列	206	53.961	0.2619	0.234

参考文献

1) Meijerink, J. and van der Vorst, H. A.: An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix Is a Symmetric M-matrix, Mathematics of Computation, Vol.31, pp.148-162, (1977).
2) 中島研吾;「並列反復法と自動チューニング ーマルチコア時代の並列プログラミングモデル」, 情報処理 Vol.50 No.6,pp.517-522,(2009)
3) http://www.cise.ufl.edu/research/sparse/matrices/
4) R. Barrett, et al.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, (1994).
5) Dongarra, J.J., Duff, I.S., Sorensen, D.C. and van der Vorst, H.A.: Solving Linear Systems on Vector and Shared Memory Computers, SIAM, (1991).
6) 岩下武史, 島崎眞昭;「多色順序付けを用いた並列化 ICCG ソルバに関する検討 ーブロック化による性能向上と工学的応用ー 」, 情報処理学会研究会報告集 ハイパフォーマンスコンピューティング, HPC-85, pp. 55-60, (2001).
7) Iwashita, T. and Shimasaki, M.: Algebraic Multi-Color Ordering for Parallelized ICCG Solver in Finite Element Analyses", IEEE Transaction on Magnetics, Vol. 38-2, pp. 429-432, (2002).
8) Nakashima, H.: T2K Open Supercomputer: Inter University and Inter-Disciplinary: Collaboration on the New Generation Supercomputer, in Proc. Intl. Conf. Informatics Education and Research for Knowledge-Circulating Society, pp.137-142, (2008).
9) 岩下武史, 島崎眞昭;「同期点の少ない並列化 ICCG 法のためのブロック化赤ー黒順序付け」, 情報処理学会論文誌, Vol. 43, pp. 893-904, (2002).