計算機アーキテクチャ　142－10
ハイパフォーマンス　85－10
コンピューティング
（２００１．３．８）

多色順序付けを用いた並列化ICCGソルバに関する検討

－ブロック化による性能向上と工学的応用－

[1]岩下　武史　　[2]島崎　眞昭
[1]京都大学大型計算機センター　[2]京都大学大学院工学研究科電気工学専攻

ICCG法の並列化手法の一つである多色順序付けに関して、構造型の差分格子を対象とした場合と非構造メッシュを対象とした場合の各々に関して報告する。構造型の差分格子を対象とした場合では、従来の大きな色数を用いた場合に問題となる代入計算中の同期点を削減する方法として、ブロック化による手法を提案する。ブロック化を進めることにより色数を最小値である２色にまで下げ、同期点を最小にしたブロック化赤－黒順序付けを提案し、同手法においてICCG法の収束性を保ちながら実行環境に合わせた並列度が得られることを示す。次に、非構造メッシュを対象とした有限要素解析において、多色順序付けを係数行列ベースで適用するための代数学的多色順序付けの一手法を提案し、並列電磁界辺要素有限要素解析において良好な結果が得られること示す。

A Study of Parallelized ICCG Solvers Based on Multi-Color Ordering

－Improvement in Performance by Blocking Approach and Application to Engineering Problems－

Takeshi Iwashita and Masaaki Shimasaki
Kyoto University

The present paper treats parallel processing of the ICCG method using multi-color ordering. First, we discuss a structured mesh in a finite difference analysis. In order to decrease synchronization points in the large-numbered multi-color ordering method, we propose that nodes belonging in several colors are assembled and are assigned into a color. This method provides a high convergence rate and suitable parallelism for computation environment in a parallelelized ICCG solver due to small synchronization (communication) costs. Next, we examine an application of the large-numbered multi-color ordering to a finite edge-element analysis with an unstructured mesh. It is shown that an algebraic approach based on the multi-color ordering attains a high parallel performance in an electromagnetic field analysis.

1. はじめに

偏微分方程式の境界値問題を有限要素法や差分法で解く際に生ずる正値対称な係数行列を持つ連立一次方程式の一般的な解法としてICCG（Incomplete Cholesky Conjugate Gradient）法がある。ICCG法は共役勾配ソルバと、不完全分解に基づいた前処理部により構成される。このうち、共役勾配ソルバ部分については、係数行列や残差ベクトルを行方向に分割することで容易に並列化することができる。一方、不完全分解と前処理として行われる代入計算は逐次型計算であるためにその並列化が困難である。なかでも、代入計算は反復の度に実行されるので、全体の計算コストに対する影響が大きく、その並列化は課題である。これまでにも、代入計算を含むICCG法の並列化に関して、様々な研究がなされている。H. A. Ven der Vorst、T. F. Chanは文献[1]において、不完全分解前処理の代表的な並列化手法として、１．打ち切りノイマン系列によるもの　２．リオーダ

リングによるもの　3．領域分割法によるものを挙げている。本稿ではこれらのなかで、リオーダリング(並列順序付け)による手法に注目する。

本稿では、まず構造型の差分格子を対象とした場合について考える。差分格子を対象とした並列番号付けとしてよく知られているものには、赤黒順序付け、dissection ordering などがある。I. S. Duff、G. A. Meurant は文献[2]において様々なオーダリングの優劣に関して論じているが、一般に並列順序付けには、並列化の代償として反復数が増加するというトレードオフが生ずる。このトレードオフを解決するために、土肥らはオーダリンググラフ理論によるオーダリングの定量的評価から、大きな色数を使用した多色順序付けを提案した[3][4]。従来、多色順序付けでは、2色（この場合は赤黒順序付けとなる)、ないし4色といった少数の色数を用いていたが、同手法では30～100色の色数を用いる。他の並列順序付けと比較した場合、同手法は少ない反復数の増加で高い並列度を実現している。そこで、著者らも同手法に注目し、工学分野における有限要素解析などへの応用を目指し検討を行ってきた。しかし、それらの研究過程において、同手法により得られる高い並列度が十分に活用されない場合（スカラ並列計算機上など）があることが分かった。これは、m 色を用いた多色順序付けでは、代入計算において、$2m$ 回の同期（通信）が必要となるからである。同手法では、例えば500×500の差分格子に対し、数万程度の並列度が得られる。しかし、計算コストと同期（通信）コストのバランスから、通常は多くても数十台のプロセッサで速度向上は飽和してしまう。そこで本稿では、多くの色数による多色順序付けにおいて、収束性を維持しながら同期コストを少なくする手法として、ブロック化による手法を提案する。節点のいくつかをブロック化することにより色数を削減し、収束性を維持しながら代入計算における同期点を少なくする。ブロック化を進めることにより、最終的には最小の色数である2色にまで色数を下げた、ブロック化赤黒順序付けを提案する。同手法では、使用プロセッサ数に応じて最適なブロック内節点数を選択することにより、最も少ない色数で高い収束性を得ることができる。

本稿では、次に有限要素解析などに用いられる非構造メッシュへの並列順序付けの応用のために、係数行列ベースでのリオーダリングの適用について検討する。このような研究においては、襲田らも RCM 順序付けと多色順序付けを組み合わせた手法を提案し、高い実効性能を得ている[5]。但し、同手法において、高い収束性を得るための戦略は多色順序付けにおいて色数を大きくするのではなくむしろ RCM 順序付けをベースにしている。本稿では、大きな色数による多色順序付けの考え方をより直接的に係数行列ベースで利用する手法について検討を行う。本手法では、係数行列ベースでの各未知数の色分けが最も重要である。各未知数の色分け方には様々な手法が考えられるが、本稿ではその一手法を提案し、同手法を辺要素有限要素解析に適用した場合についてその結果を述べる。

2．構造型差分格子を対象とした並列オーダリングに関する検討

2．1　オーダリンググラフ理論

ICCG 法に代表される係数行列の不完全 LU 分解を前処理とする反復解法の並列化に際しては、前進・後退代入計算の並列化が最も問題となる。その解決策の一つに並列オーダリングの利用があり、特に構造型格子を対象として、詳細な解析が行われている[2]。しかし、これらの研究において、多くの並列オーダリングが、逐次型のオーダリングと比較して反復法の収束性を低下させることが分かってきた。そこで、土肥らはオーダリングを定量的に扱う手法として、図1に示されるようなオーダリンググラフによる解析を試みている[3]。オーダリンググラフ中の矢印は節点間のデータ依存関係を表す。即ち、矢印の終点は始点よりも後に番号付けされている。土肥らは、このオーダリンググラフの中で、図1に示されるような incompatible node の割合が（反復法の）収束性に影響することを数値実験で示している。つまり、incompatible node の割合が高ければ高いほど収

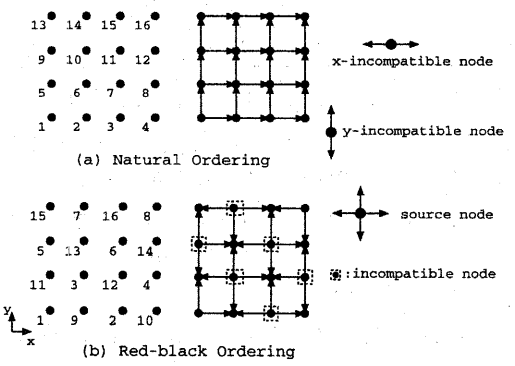

図1　オーダリンググラフの例

束性は悪化する。土肥らはこのオーダリンググラフ理論に基づいて、incompatible node の割合を適度に保ちながら、高い並列度を実現するオーダリングとして、多色順序付けにおいて多くの色数を用いることを提案した[3][4]。多色順序付けでは、同色に塗られた節点はお互いにデータ依存関係がなく、並列に扱うことができる。図2に4色による多色順序付けを示す。図中の C1～C4 は色を表し、黒丸は incompatible node を表す（図 3、4 とも）。図中において、色数を増やすと、全節点に対して incompatible node の割合が下がり、収束性が増すことが分かる。多色順序付けで得られる並列度は、一色に割り当てられた節点数となるので、色数を多くしすぎると並列度が下がる。これまでの研究においては、並列度と収束性のバランスをとる適当な値として、30～100 色程度が用いられている（ベクトル計算機上）[4]。

2．2　ブロック化による多色順序付けの性能向上

前節で述べた大きな色数による多色順序付けは、高い並列度と収束性を持つ優れた並列順序付けであるが、実際に並列計算環境で実行した場合、以下のような問題が生ずる。多色順序付けでは、各色毎に代入計算が並列実行されるので、色数に等しい同期点が前進・後退代入計算の各々に生ずる。この同期（通信）コストは、使用プロセッサ数が増加した場合に並列化による速度向上の妨げとなる。即ち、同期コストのためにオーダリングが持つ高い並列度を十分に活用することができなくなる。そこで、本稿では、大きな色数による多

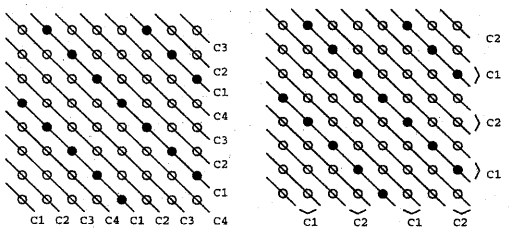

図2　4色順序付け　図3　4色から2色へのブロック化

色順序付けにより得られる収束性を維持しながら、色数を少なくして同期コストを削減する手法として、いくつかの節点をブロック化する手法（ブロック化多色順序付け）を提案する。

図3に前節で述べた4色順序付けのブロック化の例を示す。図3においては、4色順序付けにおける2色を1色にまとめることにより色数を削減している。この場合、オーダリンググラフは4色順序付けと全く同じであり、収束性も等しい。但し、ブロック内の節点はまとめて扱う必要があり、並列度は1色に割り当てられたブロック数となる。

図3のようなブロック化の場合、ブロック化された節点数にばらつきがでるため、プロセッサ間の負荷バランスがとりにくい。そこで、図4のようなブロック化を行う。図 4 のブロック化では、全てのブロック内の節点数が等しいので、プロセッサ間の負荷バランスを均等に保つことができる。図4のブロック化においても、ブロック化により多色順序付けの色数を少なくすることができる。ここで、このようなブロック化による色数の削減を進めていくと、最終的には最も少ない色数である2色（赤－黒）の場合が最も有効であることが分かる。即ち、本手法では色数の増加により収束性を高めるのではなく、ブロック化により収束性を高めることにより、代入計算中の同期点を最小

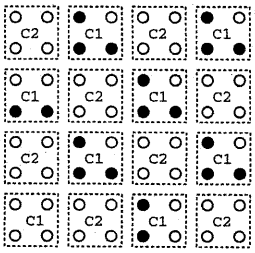

図4　ブロック化多色順序付け（2色）

に保ちながら、大きな色数での多色順序付けと同等の収束性が得られる。ここでは、本手法をブロック化赤－黒順序付けと呼ぶことにする。また、大きな色数による多色順序付けでは最適な色数は、計算環境や問題サイズに依存するのでその決定が難しい。しかし、ブロック化赤－黒順序付けでは、最適なブロック数も以下のように簡単に決定することができる。ブロック内のx方向、y方向の節点数は異なっていても良いが、簡単のため $n \times n$ のメッシュで、ブロック内の節点数を $nb \times nb$ とし、使用プロセッサ数をプロセッサ数 Np とする。ここで最適なブロック数は1色あたりのブロック数が Np となる場合である。このときオーダリングの持つ並列度はプロセッサ数と一致し、一方でincompatible nodeの数は最小となり、最も高い収束性が得られる。従って、ブロック数は、

$nb = n\sqrt{1/2Np}$ とすればよい。但し、ベクトルプロセッサ上で計算する場合で、代入計算を並列・ベクトル実行する場合には、ベクトルレジスタ長を考慮にいれて並列度をある程度高く設定しなければならない。

2．3　解析対象

本節では、解析対象として1998年並列ソフトウェアコンテストの予選問題4を用いる。但し、メッシュ分割数については若干変更している。本問題は次式で与えられるポアソン方程式の境界値問題である。

$$-\nabla \cdot (k\nabla u(x,y)) = f \ \ in\ \Omega(0,1)\times(0,1) \quad (1)$$
$$u(x,y) = 0 \ on\ \partial\Omega$$
$$if\ (1/4 \le x \le 3/4 \ \ and \ \ 1/4 \le y \le 3/4) \ \ then$$
$$k = 100 \quad else \quad k = 1$$

ここで、f は格子点を辞書式順序付けで並べた場合の格子番号を i として、$0.5\ sin(i+1)$ である。本稿では、未知数の個数を512×512（513×513分割）とする。

本解析では、式(1)を5点差分定式化により離散化し、これにより生ずる連立一次方程式をICCG法により解く。係数行列はCRS形式で格納されているものとし、ICCG法の収束判定基準として、右辺ベクトルと残差ベクトルが 10^{-7} 以下となる条件を用いる。解析は京都大学大型計算機センターの富士通製VPP-800を用いて行い、通信ライブラリとしてMPIライブラリを用いる。

表1　多色順序付けの収束性

色数	1	2	4	8	16	32
反復数	425	831	685	606	560	537

2．4　解析結果

本解析では、ICCGソルバの反復開始から収束までを計時した。

表1に多色順序付けにおいて色数を変化させた場合の結果を示す。表中の1色は辞書式順序付けを示す。表1に示されるとおり、色数が増加するにつれ反復回数が減少していることが分かる。図5に辞書式順序付けを用いた逐次型のICCGソルバの計算時間を基準とした多色順序付けの台数効果を示す。色数を増加させることにより反復数が少なくなるために、収束性の観点からは大きな色数を用いた方が有利であることが分かる。しかし、32色を使用した場合、2、8色に比べて通信（同期）回数が多いため、少ないプロセッサ数で台数効果が飽和する。このような速度向上の飽和は全計算時間における通信コストの割合が増加するために生ずるもので、解析対象の規模や並列実行環境に依存するものの、プロセッサ数が増加するに従い必ず発現する。そこで、少ない色数で高い収束性を得る手段として、2.2節で提案したブロック化赤－黒順序付けを用いる。図6に赤－黒順序付けにおいてブロック化を進めた場合の解析結果を示す。ここでは、簡単のためブロック内のx方向、y方向の節点数が同じ場合を示す。1ブロック内の節点数を、2×2、8×8、64×64と増やすことにより収束性を増し、高い台数効果が得られることがわかる。ブロック化した多色順序付けでは1色内のブロック数がオーダリングの並列度となるが、ブロック内節点数8×8とした場合、並列度は2048である。これは512×512の規模の問題に対しては十分な値といえる。図5、6を比較すると、代入計算中の同期点の数を最小にできるため、従来の多色順序付けと比較して

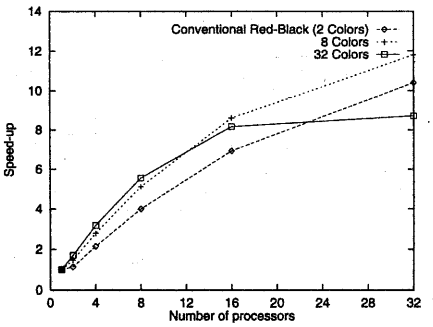

図5　多色順序付けによる台数効果

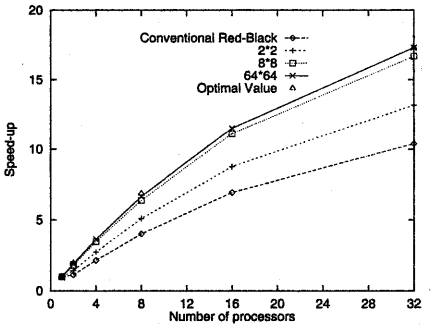

図6　ブロック化赤ー黒順序付けによる台数効果

高い並列台数効果を実現している。また、前節で述べた最適なブロック数を用いた結果を 2、8、32CPU 時について図 6 内に△印で示す。同手法においていずれのプロセッサ数についても最良の台数効果が得られていることが分かる。

3．非構造メッシュへの多色順序付けの適用

本節では、差分解析においてその有効性が確かめられている大きな色数による多色順序付けを非構造メッシュによる有限要素解析に適用する手法について検討する。

著者らは、以前の研究において、2. 1 節で述べたオーダリンググラフ理論を有限辺要素解析に適用する手法について検討を行い、仮想節点による手法を提案した[6]。同手法により多色順序付けの持つ高い並列度と収束性を有限要素解析においても利用することができた。しかし、本手法は要素間の幾何的情報を利用しており、ソルバ部の前に多色順序付けのためのプリ処理が必要となる。ソルバはその可搬性を考えると、連立一次方程式の構成部と独立であるほうが望ましい。そこで、本稿では多色順序付けをソルバ内で自動的に行う手法(代数学的多色順序付け)に関して検討を行う。

3．1　代数学的多色順序付け

ソルバ内で自動的に多色順序付けに基づくリオーダリングを行う場合、最も重要なことは未知変数の色分けである。即ち、同色の未知変数が互いに依存関係がないように塗り分けることができれば、差分解析における多色順序付けと同様に代入計算を並列化できる。そこで、本論では未知変数の色分けについてのみ述べる。

手法の説明のため、以下の表記を導入する。解くべき連立一次方程式を $\boldsymbol{Ax=b}$ とし、その次元数を n とする。係数行列 $\boldsymbol{A}$ の狭義下三角行列を $\boldsymbol{L}$ とし、その i 行の非ゼロ要素の列番号を **lnzc(i, j) (j=1, … , lnz(i))** とする。ここで、**lnz(i)** は i 行の非ゼロ要素数である。用いる色数を *ncolor* とし、各未知変数の色を保持する配列を **color(i) (i=1, … , n)** とすると、この配列は図 7 のような手順で決定される。各未知変数には 1～*ncolor* までの色がほぼサイクリックに割り当てられる。i 番目の未知変数の色は、**lnzc(i, j) (j=1, … , lnz(i))** の未知変数の色を調べ、それらの全ての色と異なるように

```
ncolor=some value      ! Set number of used colors
color(1:n)=0           ! Initialize the array color
icolor=1
do i=1,n
  j=1
  do while(j <= lnz(i))
    if (color(lnzc(i,j))==icolor) then
    icolor=mod(icolor+1,ncolor)+1    ! To next color
    j=0
    endif
    j=j+1
  enddo
  color(i)=icolor                    ! Assignment of color
  icolor=mod(icolor+1,ncolor)+1      ! To next color
  enddo
```

図7　代数学的多色順序付けにおける色の決定

決定される。全ての未知変数の色が決定された後、未知変数を色に従ってリオーダリングする。同色の未知変数は並列に扱うことができるので、1色あたりの未知変数にばらつきがなければ、並列度は *n/ncolor* で与えられる。

3．2　解析結果

代数学的多色順序付けを電磁界辺要素有限要素解析に適用し、その効果について調べる。解析対象は電気学会3次元渦電流解析検証モデルとする。基礎方程式は、変位電流を無視したマクスウェル方程式より導かれるベクトルポテンシャルを未知変数とする curl curl 方程式である。これを、一次直方体辺要素を用いたガラーキン法により解く。未知数の総数は 1011920 である。使用計算機、ICCG 法の収束判定基準は 2.3 節の場合と同じとする。

多色順序付けでは、収束性はプロセッサ数ではなく色数に依存する。本解析では、同色の未知変数が互いに依存関係を持たないための色数の下限は 33 色である。そこで、33～100 色までの色数を用いて解析を行った。その結果、60 色の際に最もよい結果を得た。表2、図8にソルバ独立型の代表的な並列化 ICCG ソルバであるブロック ICCG 法との比較を示す。差分解析の場合と同様に、少ない反復数の増加で並列化を実現しているので、高い台数効果が得られる。本解析で得られた並列度は約 16800 であり、十分な値といえるが、図8において速度向上は飽和しつつある。これは、本論の第2節で述べたように通信コストによるものである。差分解析において本論で提案したブロック化による通信コストの削減については現在検討中であり、今後の課題である。

4．結論

ICCG 法の並列化手法の一つである多色順序付けに関して大別して二つの検討を行った。

差分格子を対象とした場合について、同期コストの削減のために複数の色の節点を一色にブロック化することを提案した。さらに、このようなブロック化を進めることにより、最小の色数である2色にまで色数を下げたブロック化赤－黒順序付けを提案した。同手法では、同期コストを最小に保ちながら、与えられた並列計算環境に適した並列度を選択し、最も高い収束性を得ることができる。本手法を約25万自由度の差分解析に適用し、従来の多色順序付けよりも高い台数効果を得た。本手法は、同期コストの影響が大きい並列スカラ計算機やクラスタシステムにおいては特にその優位性が高いと考えられる。

次に多色順序付けの非構造メッシュへの応用について検討を行い、色分けに関する一手法を提案した。同手法を約 100 万自由度の電磁界辺要素有限要素解析に適用し、ブロック ICCG ソルバと比べて高い速度向上を得た。

表2　反復回数の比較

	ICCG on 1P	60-color ordering	Block ICCG	Block ICCG
Np	1	Any	4	8
反復数	366	390	779	834

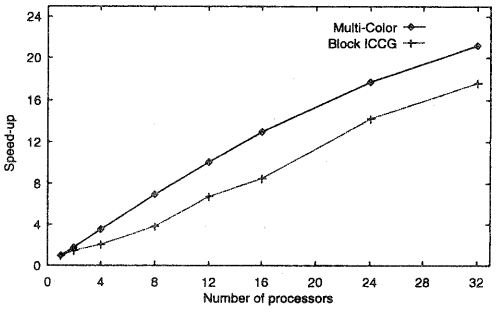

図8　代数学的多色順序付けによる台数効果

参考文献

[1] H. A. van der Vorst and T. F. Chan, *ZAMM.Z. angew. Math. Mech.*, 76, (1996), pp.167-170.

[2] I. S. Duff and G. A. Meurant, *BIT*, 29, (1989), pp. 635-657.

[3] S. Doi and A. Lichnewsky, INRIA report 1452, (1991).

[4] S. Doi and T. Washio, Parallel Computing, 25, (1999), pp. 1995-2014.

[5] 襲田, 丸山, 鷲尾, 土肥, 山田, 情報処理学会論文誌, Vol.41 No. SIG 8 (HPS 2), (2000), pp. 92-99.

[6] T. Iwashita and M Shimasaki, Digests of IEEE-CEFC, (2000), p. 175.