科学技術計算のための マルチコアプログラミング入門

第Ⅱ部：オーダリング

2011年6月30日・7月1日

中島研吾

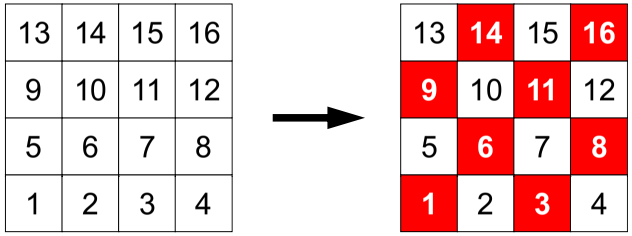
- データ依存性の解決策は?
- オーダリング（Ordering）について
  - Red-Black，Multicolor（MC）
  - Cuthill-McKee（CM），Reverse-CM（RCM）
  - オーダリングと収束の関係
- オーダリングの実装
- オーダリング付ICCG法の実装

ICCG法の並列化

- 内積：OK
- DAXPY：OK
- 行列ベクトル積：OK
- 前処理：なんとかしなければならない
  - 単純にOpenMPなどの指示行（directive）を挿入しただけでは「並列化」できない。

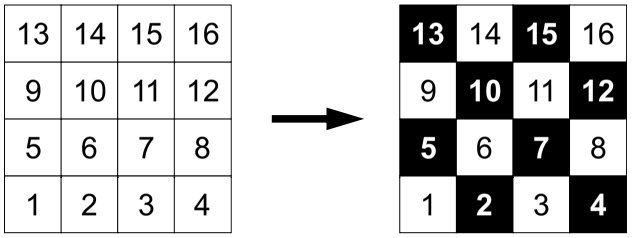
データ依存性の解決策＝ 色分け，色づけ（coloring）

- 依存性を持たない（互いに独立な）要素を同時に処理するようにすれば良い



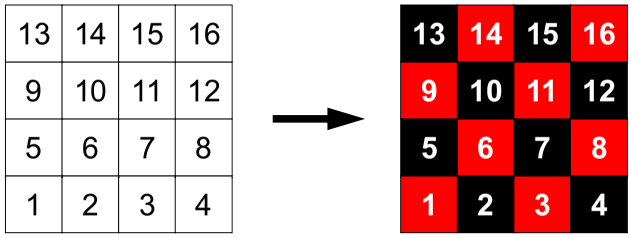
データ依存性の解決策＝色分け，色づけ(coloring)

- 依存性を持たない(互いに独立な)要素を同時に処理するようにすれば良い

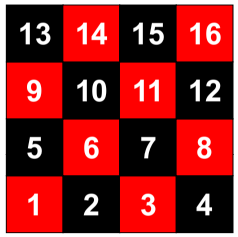


データ依存性の解決策＝色分け，色づけ(coloring)

- 依存性を持たない要素群⇒同じ「色」に色づけ(coloring)する
- 最も単純な色づけ：Red-Black Coloring(2色)

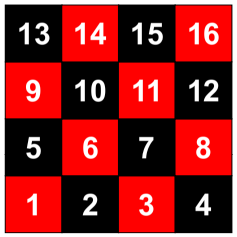


Red-Black(1/3)




```
!$omp parallel do private (ip,i,k,VAL)
  do ip= 1, 4
  do   i= INDEX(ip-1)+1, INDEX(ip)
    if (COLOR(i).eq.RED) then
      WVAL= W(i,Z)
      do k= indexL(i-1)+1, indexL(i)
        WVAL= WVAL -  AL(k) * W(itemL(k),Z)
      enddo
      W(i,Z)= WVAL * W(i,DD)
    endif
  enddo
  enddo
!$omp parallel enddo

!$omp parallel do private (ip,i,k,VAL)
  do ip= 1, 4
  do   i= INDEX(ip-1)+1, INDEX(ip)
    if (COLOR(i).eq.BLACK) then
      WVAL= W(i,Z)
      do k= indexL(i-1)+1, indexL(i)
        WVAL= WVAL -  AL(k) * W(itemL(k), Z)
      enddo
      W(i,Z)= WVAL * W(i,DD)
    endif
  enddo
  enddo
!$omp parallel enddo
```

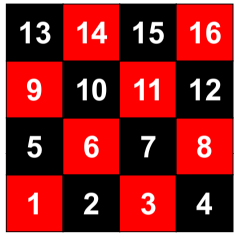

Red-Black(2/3)




```
!$omp parallel do private (ip,i,k,VAL)
  do ip= 1, 4
  do   i= INDEX(ip-1)+1, INDEX(ip)
    if (COLOR(i).eq.RED) then
      WVAL= W(i,Z)
      do k= indexL(i-1)+1, indexL(i)
        WVAL= WVAL -  AL(k) * W(itemL(k), Z)
      enddo
      W(i,Z)= WVAL * W(i,DD)
    endif
  enddo
  enddo
!$omp parallel enddo
```

- 「Red」要素を処理する間，右辺に来るのは必ず「Black」要素
  - RED：書き出し，BLACK：読み込み
- 「Red」要素の処理をする間，「Black」要素の内容が変わることは無い
- データ依存性が回避される

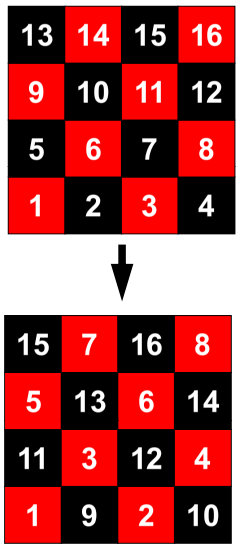
Red-Black（3/3）



- 「Black」要素を処理する間，右辺に来るのは必ず「Red」要素
  - RED：読み込み，BLACK：書き出し
- 「Black」要素の処理をする間，「Red」要素の内容が変わることは無い
- データ依存性が回避される

```
!$omp parallel do private (ip,i,k,VAL)
  do ip= 1, 4
  do  i= INDEX(ip-1)+1, INDEX(ip)
    if (COLOR(i).eq.BLACK) then
      WVAL= W(i,Z)
      do k= indexL(i-1)+1, indexL(i)
        WVAL= WVAL -  AL(k) * W(itemL(k), Z)
      enddo
      W(i,Z)= WVAL * W(i,DD)
    endif
  enddo
  enddo
!$omp parallel enddo
```


Red-Black Ordering/Reordering




```
  do icol= 1, 2
!$omp parallel do private (ip,i,j,VAL)
    do ip= 1, 4
      do  i= INDEX(ip-1,icol)+1, INDEX(ip,icol)
        WVAL= W(i,Z)
        do k= indexL(i-1)+1, indexL(i)
          WVAL= WVAL -  AL(k) * W(itemL(k), Z)
        enddo
        W(i,Z)= WVAL * W(i,DD)
      enddo
    enddo
!$omp parallel enddo
  enddo
```

```
INDEX(0,1)= 0
INDEX(1,1)= 2
INDEX(2,1)= 4
INDEX(3,1)= 6
INDEX(4,1)= 8
```

```
INDEX(0,2)= 8
INDEX(1,2)=10
INDEX(2,2)=12
INDEX(3,2)=14
INDEX(4,2)=16
```

- 要素番号を「Red」⇒「Black」の順にふり直す（reordering, ordering）とプログラムが簡単になる（計算効率も高い）

- データ依存性の解決策は?
- **オーダリング（Ordering）について**
  - **Red-Black, Multicolor（MC）**
  - **Cuthill-McKee（CM），Reverse-CM（RCM）**
  - オーダリングと収束の関係
- オーダリングの実装
- オーダリング付ICCG法の実装
- マルチコアへの実装（OpenMP）へ向けて

オーダリング（ordering）の効用

- **並列性を得る：依存性の排除**
- Fill-inを減らす
- バンド幅を減らす，プロフィルを減らす
- ブロック化

- もともとは，4色問題，一筆書き（巡回セールスマン問題）等と関連
  - 数値計算への適用

- 小国他「行列計算ソフトウェア」，丸善（1991）

並列計算のためのオーダリング法

- マルチカラーオーダリング
  - 並列性
  - Red-Black オーダリング（2色）等
- CM法（Cuthill-McKee），RCM法（Reverse Cuthill-McKee）
  - fill-inを減らす
  - マトリクスのバンド幅を減らす，プロフィルを減らす
  - 並列性

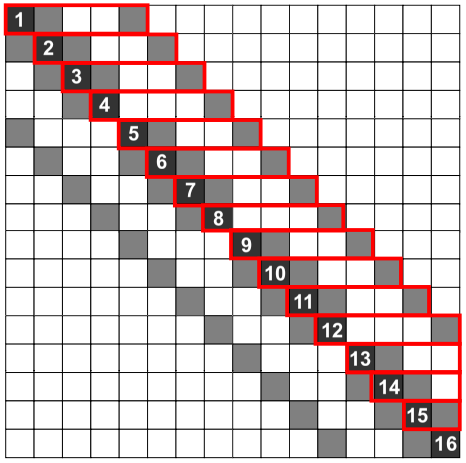
用語の定義

- β_i: i 行における非ゼロ成分の列番号の最大値を k とするとき，$\beta_i = k - i$
- バンド幅: β_iの最大値
- プロフィル: β_iの和

- バンド幅，プロフィル，Fill-inともに少ない方が都合が良い
- 特にバンド幅，プロフィルは収束に影響

β_i の定義

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4



マルチカラーオーダリング

15	11	16	12
9	13	10	14
7	3	8	4
1	5	2	6

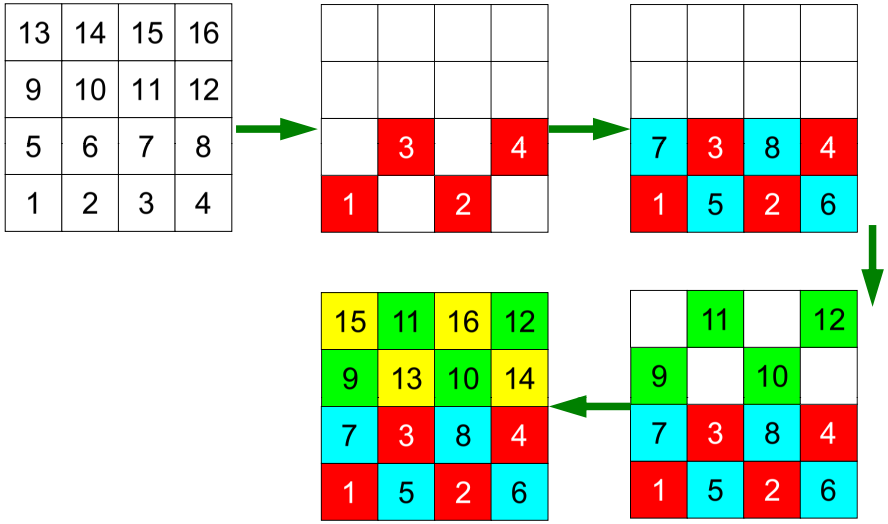
15	7	16	8
5	13	6	14
11	3	12	4
1	9	2	10

- Multicolor Ordering
  - 「MC法」と呼ぶ
- マルチカラーオーダリングは，互いに独立で依存性のない要素同士を同じ「色」に分類し，その分類に従って要素や節点の番号を振りなおす手法である。
  - Red-Blackは2色の場合
  - 複雑形状の場合，より多い「色」が必要
- 同じ「色」に分類された要素に関する計算は並列に実施できる。
- ある要素と，その要素に接続する周囲の要素は違う「色」に属している。

MC法の基本的なアルゴリズム

① 「要素数÷色数」を「m」とする。
② 初期要素番号が若い順に互いに独立な要素を「m」個選び出して彩色し，次の色へ進む。
③ 指定した色数に達して，全ての要素が彩色されるまで②を繰り返す。
④ 色番号の若い順番に要素を再番号づけする（各色内では初期要素番号が若い順）。

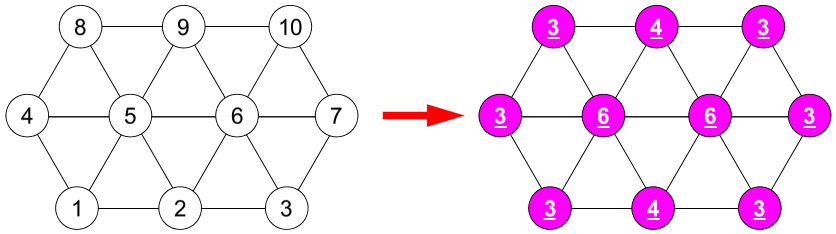
MC法：4色



修正されたMC法

① 「次数」最小の要素を「新要素番号＝1」，「第1色」とし，「色数のカウンタ＝1」とする。
② ITEMcou=ICELTOT/NCOLORtotに相当する値を「各色に含まれる要素数」とする。
③ ITEMcou個の独立な要素を初期要素番号が若い順に選び出す。
④ ITEMcou個の要素が選ばれるか，あるいは独立な要素が無くなったら「色数のカウンタ＝2」として第2色へ進む。
⑤ 全ての要素が彩色されるまで③，④を繰り返す。
⑥ 最終的な色数カウンタの値をNCOLORtotとし，色番号の若い順番に要素を再番号づけする（各色内では初期要素番号の順番）。

次数（degree）：各点に隣接する点の数



修正されたMC法

① 「次数」最小の要素を「新要素番号＝1」，「第1色」とし，「色数のカウンタ＝1」とする。

② ITEMcou=ICELTOT/NCOLORtotに相当する値を「各色に含まれる要素数」とする。

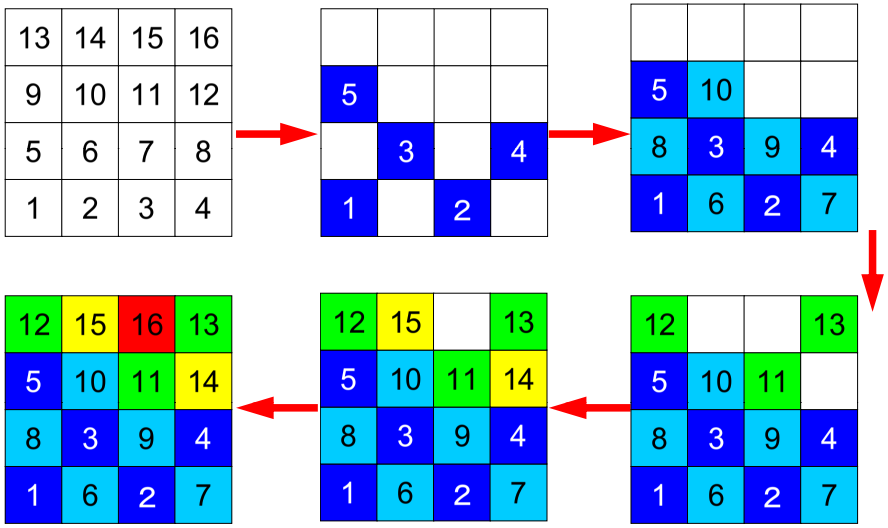
③ ITEMcou個の独立な要素を初期要素番号が若い順に選び出す。

④ ITEMcou個の要素が選ばれるか，あるいは独立な要素が無くなったら「色数のカウンタ＝2」として第2色へ進む。

⑤ 全ての要素が彩色されるまで③，④を繰り返す。

⑥ 最終的な色数カウンタの値をNCOLORtotとし，色番号の若い順番に要素を再番号づけする（各色内では初期要素番号の順番）。同じ色：連続した「新」要素番号

MC法：3色に設定，実は5色



並列計算のためのオーダリング法

- マルチカラーオーダリング
  - 並列性
  - Red-Black オーダリング（2色）等
- CM法（Cuthill-McKee），RCM法（Reverse Cuthill-McKee）
  - fill-inを減らす
  - マトリクスのバンド幅を減らす，プロフィルを減らす
  - 並列性

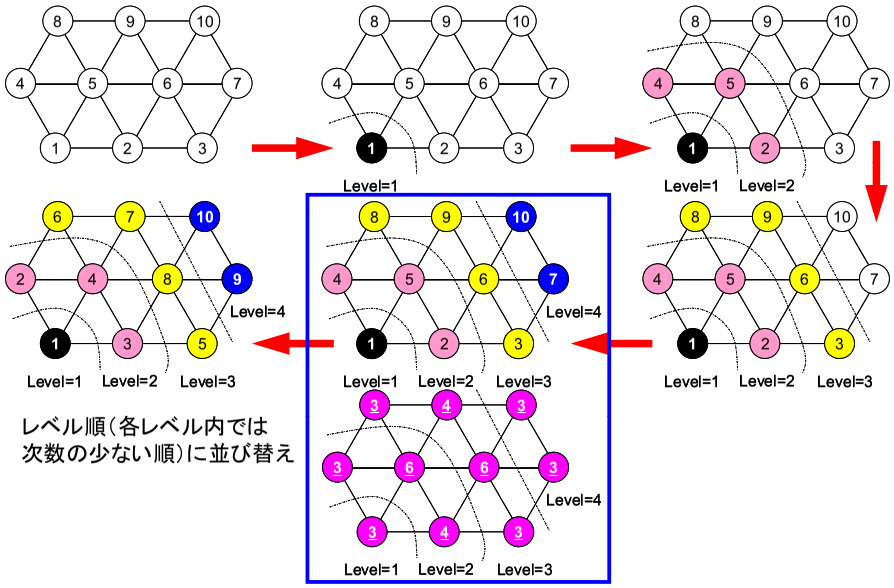
CM法の基本的なアルゴリズム

① 各点に隣接する点の数を「次数（degree）」，最小次数の点を「レベル=1」の点とする。

② 「レベル=k-1」に属する点に隣接する点を「レベル=k」とする。全ての点にレベルづけがされるまで「k」を1つずつ増やして繰り返す。

③ 全ての点がレベルづけされたら，レベルの順番に再番号づけする（各レベル内では「次数」の番号が少ない順）。

Cuthill-Mckee Ordering（CM法）の例



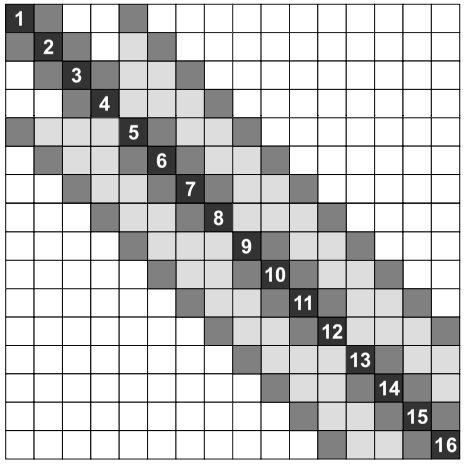
RCM（Reverse CM）

- まずCMの手順を実行
  - 次数（degree）の計算
  - 「レベル（k（k≧2））」の要素の選択
  - 繰り返し，再番号付け
- 再々番号付け
  - CMの番号付けを更に逆順にふり直す
  - Fill-inがCMの場合より少なくなる

初期状態

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

バンド幅　4
プロフィル　51
Fill-in　54

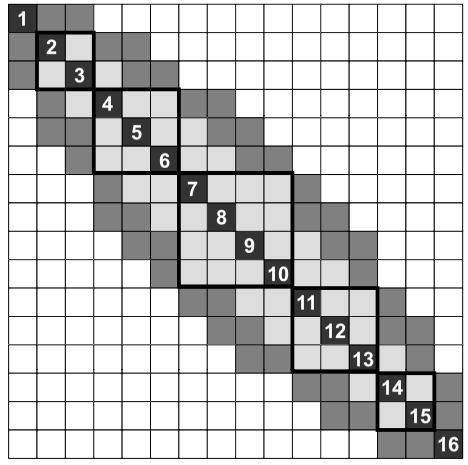


■非ゼロ成分，■Fill-in

CMオーダリング

10	13	15	16
6	9	12	14
3	5	8	11
1	2	4	7

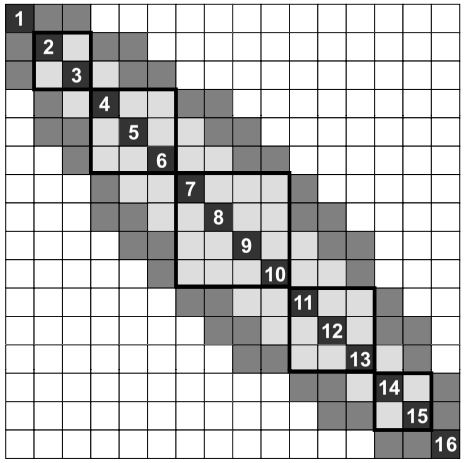
バンド幅　4
プロフィル　46
Fill-in　44



■非ゼロ成分，■Fill-in

RCMオーダリング

7	4	2	1
11	8	5	3
14	12	9	6
16	15	13	10



バンド幅　4
プロフィル　46
Fill-in　44

■非ゼロ成分，■Fill-in

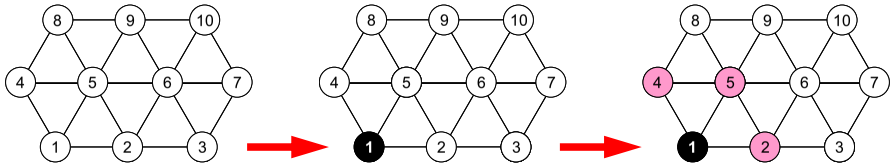
修正されたCM法
並列計算向け

① 各要素に隣接する要素数を「次数」とし，最小次数の要素を「レベル=1」の要素とする。

② 「レベル=k-1」の要素に隣接する要素を「レベル=k」とする。同じレベルに属する要素はデータ依存性が発生しないように，隣接している要素同士が同じレベルに入る場合は一方を除外する（現状では先に見つかった要素を優先している）。全ての点要素にレベルづけがされるまで「k」を1つずつ増やして繰り返す。

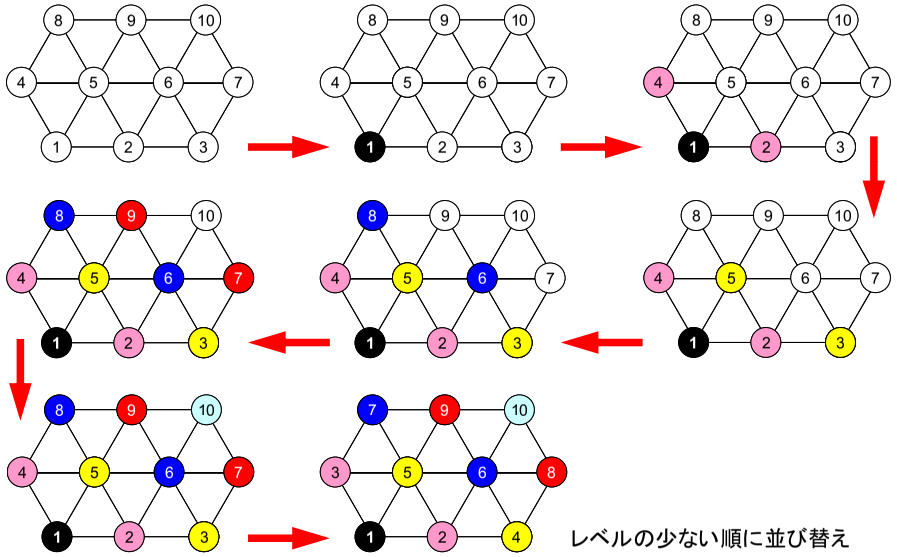
③ 全ての要素がレベルづけされたら，レベルの順番に再番号づけする。同じレベル：連続した「新」要素番号

修正されたCM法



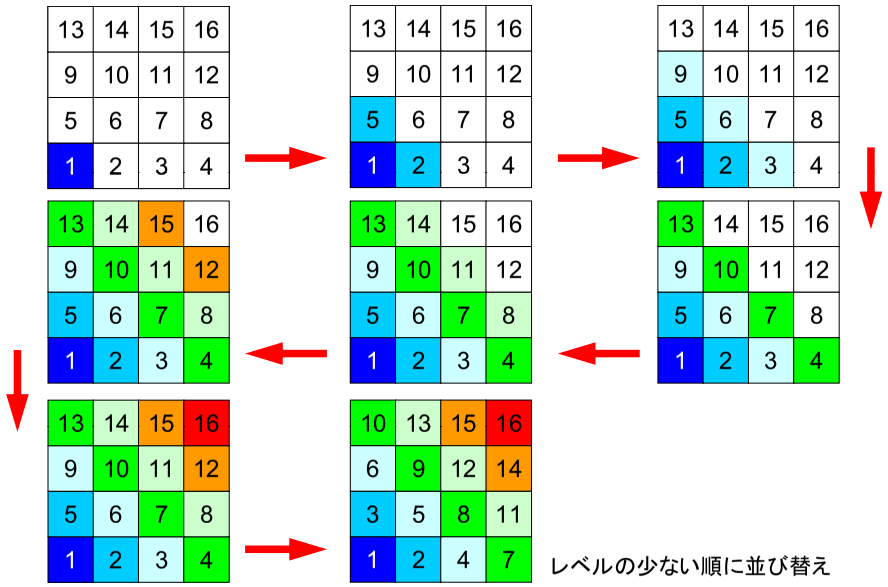
同じレベルに属する要素は
データ依存性が発生しない

修正されたCM法



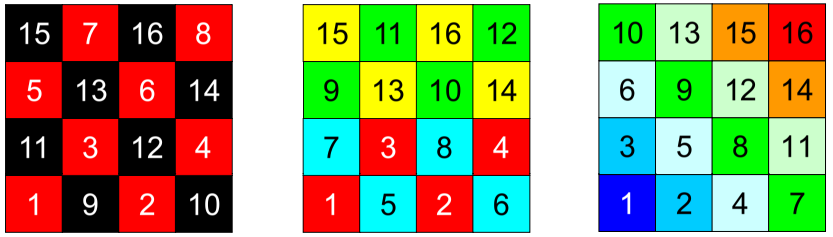
レベルの少ない順に並び替え

修正されたCM法



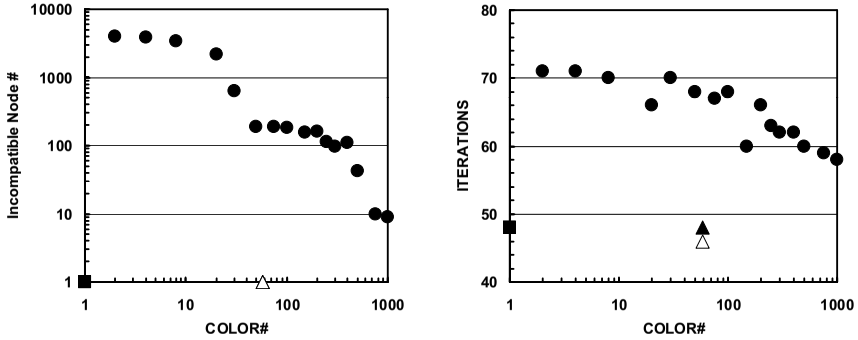
MC法，CM/RCM法の違い

- CM法では，同一レベル（色）における各要素の独立性だけでなく，計算順序を考慮して，レベル間の依存性を考慮している点



- データ依存性の解決策は?
- **オーダリング（Ordering）について**
  - Red-Black，Multicolor（MC）
  - Cuthill-McKee（CM），Reverse-CM（RCM）
  - **オーダリングと収束の関係**
- オーダリングの実装
- オーダリング付ICCG法の実装
- マルチコアへの実装（OpenMP）へ向けて

ICCG法の収束（後述）



（20^3=8,000要素，EPSICCG=10^{-8}）
（■：ICCG(L1)，●：ICCG-MC，▲：ICCG-CM，△：ICCG-RCM）

収束とオーダリングの関係（後述）

- 要素数＝20^3
- Red-Black ～ 4色 ＜ 初期状態 ～ CM, RCM

初期状態	
バンド幅	4
プロフィル	51
Fill-in	54

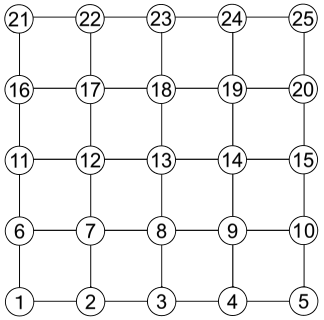
Red-Black	
バンド幅	10
プロフィル	77
Fill-in	44

4色	
バンド幅	10
プロフィル	57
Fill-in	46

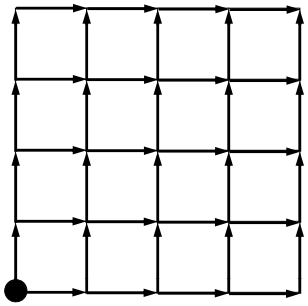
CM，RCM	
バンド幅	4
プロフィル	46
Fill-in	44

反復回数と色数の関係
Incompatible Nodesとは?

Doi, S. (NEC) et al.

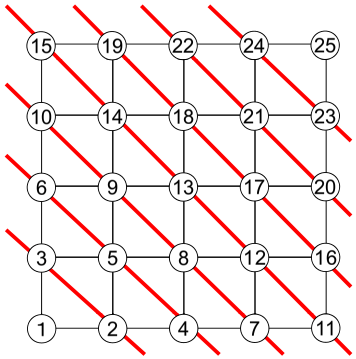


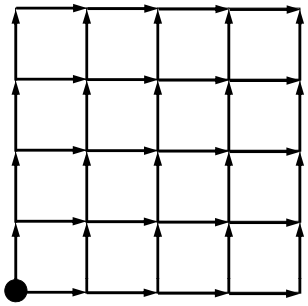
影響の伝わり方



他から影響を受けない点が「Incompatible Node」
周囲の全ての点よりも番号が若い，ということ
少ない方が収束が早い

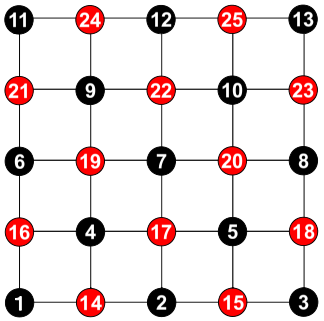
CM（Cuthill-McKee）の場合

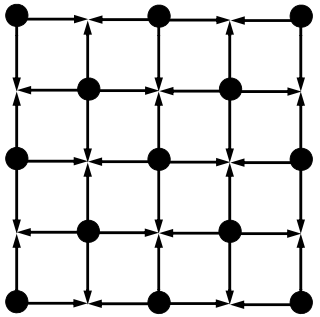




Red-Blackの場合

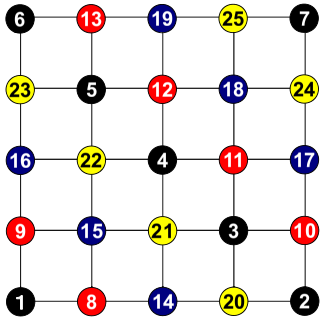
並列性は高いがincompatible node多い
ILU系前処理，Gauss-Seidelで反復回数増加

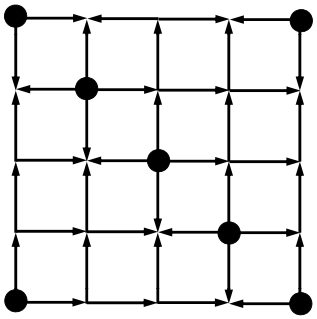




4色の場合

並列性は弱まるがincompatible nodeは減少
ILU系前処理，Gauss-Seidelで反復回数減少





収束とオーダリング

- これ以外にも境界条件の影響などもあり，一概には言えない
- たとえばCMとRCMは本問題の場合全く同じになるはずであるが，RCMの方が若干収束が良い

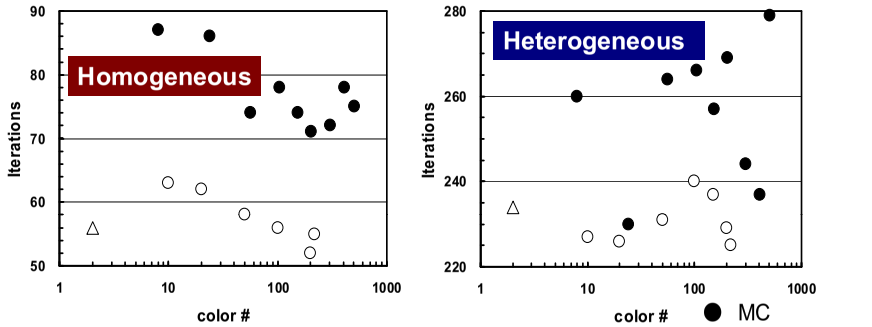
オーダリングの効果

- オーダリングによって，行列の処理の順番が変わり，何らかの「改善」が得られる。
  - 並列性の付与：並列計算，ベクトル計算への適合性
  - 収束が早くなる場合もある。
- 例に示したような単純な形状ですら，オーダリングをしなければ，並列化，ベクトル化できない。

- 注意点
  - オーダリングによって答えが変わることもある。
  - 対象としている物理，数学に関する深い知識と洞察を要する。

オーダリング手法の比較
三次元弾性問題

- MCは収束遅い，不安定（特に不均質（悪条件）問題）
- Cyclic-Mulricoloring + RCM（CM-RCM）が有効（後述）
〔Washio et al. 2000〕



3D Linear-Elastic Problems with 32,768 DOF

- データ依存性の解決策は?
- オーダリング(Ordering)について
  - Red-Black, Multicolor(MC)
  - Cuthill-McKee(CM), Reverse-CM(RCM)
  - オーダリングと収束の関係
- **オーダリングの実装**
- オーダリング付ICCG法の実装
- マルチコアへの実装(OpenMP)へ向けて

オーダリング実装:L2-color

実習

- 三次元形状(ここでは二次元)の色づけのプログラム
  - マルチカラーオーダリング, CM法, RCM法(CMRCMについてはあとで)

```
$ cd <$L2>/color/src
$ make
$ cd ../run
$ ./L2-color

You have       16 elements.
How many colors do you need ?
  #COLOR must be more than 2 and
  #COLOR must not be more than      16
   CM if #COLOR .eq. 0
  RCM if #COLOR .eq.-1
CMRCM if #COLOR .le.-2
=>

$ ls color.log colo.inp
```

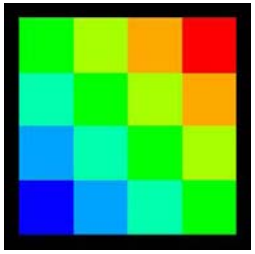
この2次元形状を接続関係に基づき色づけする。

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

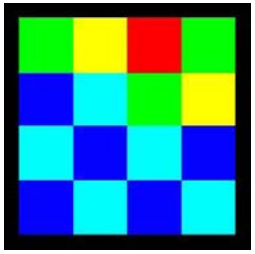
実施内容(2/2)

実習

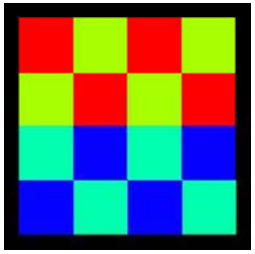
- 2つのファイルが生成される
  - color.log　　新旧メッシュ番号の対応表
行列関連情報
  - color.inp　　メッシュの色分けファイル(MicroAVS用)



入力: 0
(CM, 7 colors)



入力: 3
(5 colors)



入力: 4
(4 colors)

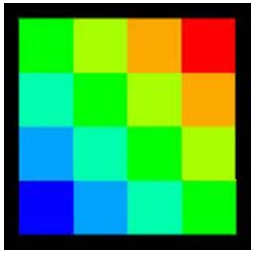
入力=0: CM(7色)

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

10	13	15	16
6	9	12	14
3	5	8	11
1	2	4	7

```
#new    1   #old    1   color    1
#new    2   #old    2   color    2
#new    3   #old    5   color    2
#new    4   #old    3   color    3
#new    5   #old    6   color    3
#new    6   #old    9   color    3
#new    7   #old    4   color    4
#new    8   #old    7   color    4
#new    9   #old   10   color    4
#new   10   #old   13   color    4
#new   11   #old    8   color    5
#new   12   #old   11   color    5
#new   13   #old   14   color    5
#new   14   #old   12   color    6
#new   15   #old   15   color    6
#new   16   #old   16   color    7
```



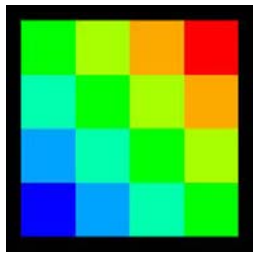
入力=0：CM（7色）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

10	13	15	16
6	9	12	14
3	5	8	11
1	2	4	7

```
#new    1   #old    1   color   1
#new    2   #old    2   color   2
#new    3   #old    5   color   2
#new    4   #old    3   color   3
#new    5   #old    6   color   3
#new    6   #old    9   color   3
#new    7   #old    4   color   4
#new    8   #old    7   color   4
#new    9   #old   10   color   4
#new   10   #old   13   color   4
#new   11   #old    8   color   5
#new   12   #old   11   color   5
#new   13   #old   14   color   5
#new   14   #old   12   color   6
#new   15   #old   15   color   6
#new   16   #old   16   color   7
```



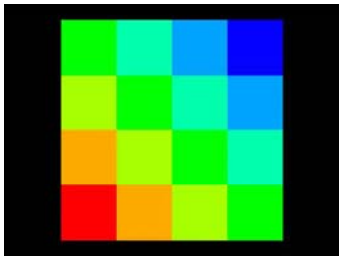
入力=-1：RCM（7色）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

7	4	2	1
11	8	5	3
14	12	9	6
16	15	13	10

```
#new    1   #old   16   color   1
#new    2   #old   15   color   2
#new    3   #old   12   color   2
#new    4   #old   14   color   3
#new    5   #old   11   color   3
#new    6   #old    8   color   3
#new    7   #old   13   color   4
#new    8   #old   10   color   4
#new    9   #old    7   color   4
#new   10   #old    4   color   4
#new   11   #old    9   color   5
#new   12   #old    6   color   5
#new   13   #old    3   color   5
#new   14   #old    5   color   6
#new   15   #old    2   color   6
#new   16   #old    1   color   7
```



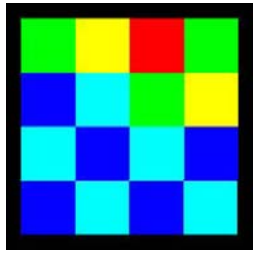
入力=3：実際は5色（マルチカラー）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

```
#new    1   #old    1   color   1
#new    2   #old    3   color   1
#new    3   #old    6   color   1
#new    4   #old    8   color   1
#new    5   #old    9   color   1
#new    6   #old    2   color   2
#new    7   #old    4   color   2
#new    8   #old    5   color   2
#new    9   #old    7   color   2
#new   10   #old   10   color   2
#new   11   #old   11   color   3
#new   12   #old   13   color   3
#new   13   #old   16   color   3
#new   14   #old   12   color   4
#new   15   #old   14   color   4
#new   16   #old   15   color   5
```



入力=3：実際は5色（マルチカラー）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

5			
	3		4
1		2	

```
#new    1   #old    1   color   1
#new    2   #old    3   color   1
#new    3   #old    6   color   1
#new    4   #old    8   color   1
#new    5   #old    9   color   1
#new    6   #old    2   color   2
#new    7   #old    4   color   2
#new    8   #old    5   color   2
#new    9   #old    7   color   2
#new   10   #old   10   color   2
#new   11   #old   11   color   3
#new   12   #old   13   color   3
#new   13   #old   16   color   3
#new   14   #old   12   color   4
#new   15   #old   14   color   4
#new   16   #old   15   color   5
```

16/3=5

「5」個ずつ独立な要素を元の番号順に選択

入力=3：実際は5色（multicolor）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

5	10		
8	3	9	4
1	6	2	7

```
#new    1   #old    1   color    1
#new    2   #old    3   color    1
#new    3   #old    6   color    1
#new    4   #old    8   color    1
#new    5   #old    9   color    1
#new    6   #old    2   color    2
#new    7   #old    4   color    2
#new    8   #old    5   color    2
#new    9   #old    7   color    2
#new   10   #old   10   color    2
#new   11   #old   11   color    3
#new   12   #old   13   color    3
#new   13   #old   16   color    3
#new   14   #old   12   color    4
#new   15   #old   14   color    4
#new   16   #old   15   color    5
```

16/3=5

「5」個ずつ独立な要素を元の番号順に選択

入力=3：実際は5色（マルチカラー）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

12			13
5	10	11	
8	3	9	4
1	6	2	7

```
#new    1   #old    1   color    1
#new    2   #old    3   color    1
#new    3   #old    6   color    1
#new    4   #old    8   color    1
#new    5   #old    9   color    1
#new    6   #old    2   color    2
#new    7   #old    4   color    2
#new    8   #old    5   color    2
#new    9   #old    7   color    2
#new   10   #old   10   color    2
#new   11   #old   11   color    3
#new   12   #old   13   color    3
#new   13   #old   16   color    3
#new   14   #old   12   color    4
#new   15   #old   14   color    4
#new   16   #old   15   color    5
```

16/3=5

独立な要素が無くなったら次の色へ

入力=3：実際は5色（マルチカラー）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

12	15		13
5	10	11	14
8	3	9	4
1	6	2	7

```
#new    1   #old    1   color    1
#new    2   #old    3   color    1
#new    3   #old    6   color    1
#new    4   #old    8   color    1
#new    5   #old    9   color    1
#new    6   #old    2   color    2
#new    7   #old    4   color    2
#new    8   #old    5   color    2
#new    9   #old    7   color    2
#new   10   #old   10   color    2
#new   11   #old   11   color    3
#new   12   #old   13   color    3
#new   13   #old   16   color    3
#new   14   #old   12   color    4
#new   15   #old   14   color    4
#new   16   #old   15   color    5
```

16/3=5

独立な要素が無くなったら次の色へ

入力=3：実際は5色（マルチカラー）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

```
#new    1   #old    1   color    1
#new    2   #old    3   color    1
#new    3   #old    6   color    1
#new    4   #old    8   color    1
#new    5   #old    9   color    1
#new    6   #old    2   color    2
#new    7   #old    4   color    2
#new    8   #old    5   color    2
#new    9   #old    7   color    2
#new   10   #old   10   color    2
#new   11   #old   11   color    3
#new   12   #old   13   color    3
#new   13   #old   16   color    3
#new   14   #old   12   color    4
#new   15   #old   14   color    4
#new   16   #old   15   color    5
```

16/3=5

最終的に5色必要であった

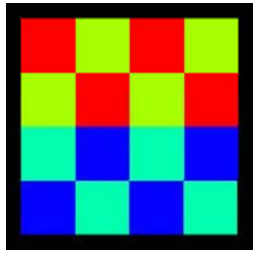
入力=4：4色（マルチカラー）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

15	11	16	12
9	13	10	14
7	3	8	4
1	5	2	6

```
#new     1   #old     1  color     1
#new     2   #old     3  color     1
#new     3   #old     6  color     1
#new     4   #old     8  color     1
#new     5   #old     2  color     2
#new     6   #old     4  color     2
#new     7   #old     5  color     2
#new     8   #old     7  color     2
#new     9   #old     9  color     3
#new    10   #old    11  color     3
#new    11   #old    14  color     3
#new    12   #old    16  color     3
#new    13   #old    10  color     4
#new    14   #old    12  color     4
#new    15   #old    13  color     4
#new    16   #old    15  color     4
```



入力=3：実際は5色（マルチカラー）

color.log：行列関連情報出力

13	14	15	16
9	10	11	12
5	**6**	**7**	8
1	2	3	4

↓

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

```
### INITIAL connectivity
I=    1     INL(i)=    0     INU(i)=    2
  IAL:
  IAU:    2    5
I=    2     INL(i)=    1     INU(i)=    2
  IAL:    1
  IAU:    3    6
I=    3     INL(i)=    1     INU(i)=    2
  IAL:    2
  IAU:    4    7
I=    4     INL(i)=    1     INU(i)=    1
  IAL:    3
  IAU:    8
I=    5     INL(i)=    1     INU(i)=    2
  IAL:    1
  IAU:    6    9
I=    6     INL(i)=    2     INU(i)=    2
  IAL:    2    5
  IAU:    7   10
I=    7     INL(i)=    2     INU(i)=    2
  IAL:    3    6
  IAU:    8   11
I=    8     INL(i)=    2     INU(i)=    1
  IAL:    4    7
  IAU:   12
I=    9     INL(i)=    1     INU(i)=    2
  IAL:    5
  IAU:   10   13
I=   10     INL(i)=    2     INU(i)=    2
  IAL:    6    9
  IAU:   11   14
I=   11     INL(i)=    2     INU(i)=    2
  IAL:    7   10
  IAU:   12   15
I=   12     INL(i)=    2     INU(i)=    1
  IAL:    8   11
  IAU:   16
I=   13     INL(i)=    1     INU(i)=    1
  IAL:    9
  IAU:   14
```

```
I=   14     INL(i)=    2     INU(i)=    1
  IAL:   10   13
  IAU:   15
I=   15     INL(i)=    2     INU(i)=    1
  IAL:   11   14
  IAU:   16
I=   16     INL(i)=    2     INU(i)=    0
  IAL:   12   15
  IAU:

COLOR number       5

  #new         1   #old         1  color         1
  #new         2   #old         3  color         1
  #new         3   #old         6  color         1
  #new         4   #old         8  color         1
  #new         5   #old         9  color         1
  #new         6   #old         2  color         2
  #new         7   #old         4  color         2
  #new         8   #old         5  color         2
  #new         9   #old         7  color         2
  #new        10   #old        10  color         2
  #new        11   #old        11  color         3
  #new        12   #old        13  color         3
  #new        13   #old        16  color         3
  #new        14   #old        12  color         4
  #new        15   #old        14  color         4
  #new        16   #old        15  color         5
```


入力=3：実際は5色（マルチカラー）

color.log：行列関連情報出力

13	14	15	16
9	10	11	12
5	**6**	**7**	8
1	2	3	4

↓

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

```
### FINAL connectivity
I=    1     INL(i)=    0     INU(i)=    2
  IAL:
  IAU:    6    8
I=    2     INL(i)=    0     INU(i)=    3
  IAL:
  IAU:    6    7    9
I=    3     INL(i)=    0     INU(i)=    4
  IAL:
  IAU:    6    8    9   10
I=    4     INL(i)=    0     INU(i)=    3
  IAL:
  IAU:    7    9   14
I=    5     INL(i)=    0     INU(i)=    3
  IAL:
  IAU:    8   10   12
I=    6     INL(i)=    3     INU(i)=    0
  IAL:    1    2    3
  IAU:
I=    7     INL(i)=    2     INU(i)=    0
  IAL:    2    4
  IAU:
I=    8     INL(i)=    3     INU(i)=    0
  IAL:    1    3    5
  IAU:
I=    9     INL(i)=    3     INU(i)=    1
  IAL:    2    3    4
  IAU:   11
I=   10     INL(i)=    2     INU(i)=    2
  IAL:    3    5
  IAU:   11   15
I=   11     INL(i)=    2     INU(i)=    2
  IAL:    9   10
  IAU:   14   16
I=   12     INL(i)=    1     INU(i)=    1
  IAL:    5
  IAU:   15
I=   13     INL(i)=    0     INU(i)=    2
  IAL:
  IAU:   14   16
```

```
I=   14     INL(i)=    3     INU(i)=    0
  IAL:    4   11   13
  IAU:
I=   15     INL(i)=    2     INU(i)=    1
  IAL:   10   12
  IAU:   16
I=   16     INL(i)=    3     INU(i)=    0
  IAL:   11   15   13
  IAU:
```


「L2-color」のソースファイル

```
$ cd <$L2>/coloring/src
$ ls
```

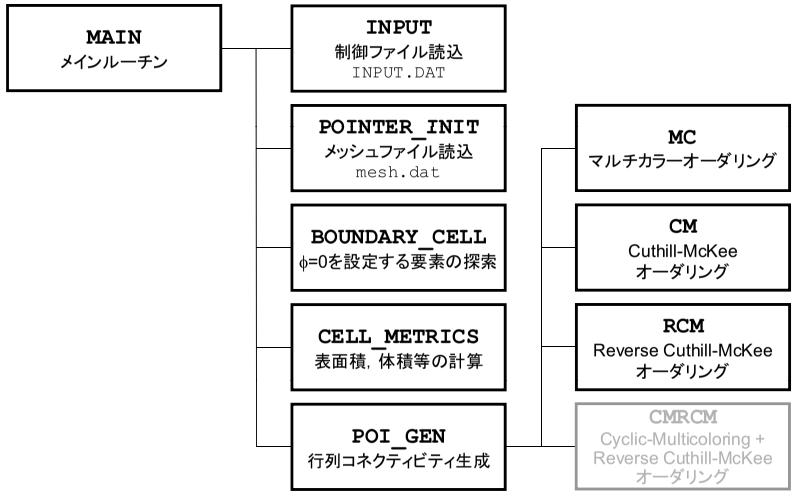
13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

この2次元形状を色づけする。

プログラムの構成

```
      program MAIN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

      call POINTER_INIT
      call POI_GEN

      call OUTUCD

      open (21, file='color.log', status='unknown')
        write (21,'(//,a,i8,/)') 'COLOR number', NCOLORtot
        do ic= 1, NCOLORtot
          do i= COLORindex(ic-1)+1, COLORindex(ic)
            write (21,'(3(a,i8))') '  #new', i, '    #old', NEWtoOLD(i), &
     &                             '  color', ic
          enddo
        enddo
      close (21)

      stop
      end
```


プログラムの構成図



プログラムの構成

```
      program MAIN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

      call POINTER_INIT
      call POI_GEN

      call OUTUCD

      open (21, file='color.log', status='unknown')
        write (21,'(//,a,i8,/)') 'COLOR number', NCOLORtot
        do ic= 1, NCOLORtot
          do i= COLORindex(ic-1)+1, COLORindex(ic)
            write (21,'(3(a,i8))') '  #new', i, '    #old', NEWtoOLD(i), &
     &                             '  color', ic
          enddo
        enddo
      close (21)

      stop
      end
```


module STRUCT

```
      module STRUCT

      include 'precision.inc'

!C
!C-- METRICs & FLUX
      integer (kind=kint) :: ICELTOT, ICELTOTp, N
      integer (kind=kint) :: NX, NY, NZ, NXP1, NYP1, NZP1, IBNODTOT
      integer (kind=kint) :: NXc, NYc, NZc

      real (kind=kreal) ::                                              &
     &      DX, DY, DZ, XAREA, YAREA, ZAREA, RDX, RDY, RDZ,             &
     &      RDX2, RDY2, RDZ2, R2DX, R2DY, R2DZ

      real (kind=kreal), dimension(:), allocatable ::                   &
     &     VOLCEL, VOLNOD, RVC, RVN

      integer (kind=kint), dimension(:,:), allocatable ::               &
     &        XYZ, NEIBcell

!C
!C-- BOUNDARYs
      integer (kind=kint) :: ZmaxCELtot
      integer (kind=kint), dimension(:), allocatable :: BC_INDEX, BC_NOD
      integer (kind=kint), dimension(:), allocatable :: ZmaxCEL

!C
!C-- WORK
      integer (kind=kint), dimension(:,:), allocatable :: IWKX
      real(kind=kreal),    dimension(:,:), allocatable :: FCV

      end module STRUCT
```

ICELTOT： 要素数
N： 節点数

NX,NY,NZ： x,y,z方向要素数
NXP1,NYP1,NZP1：
x,y,z方向節点数
IBNODTOT： NXP1×NYP1

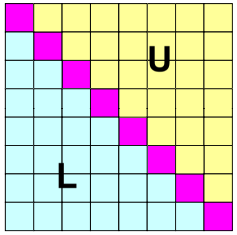
XYZ(ICELTOT,3)： 要素座標（後述）
NEIBcell(ICELTOT,6)：
隣接要素（後述）

module PCG

```
      module PCG

      integer, parameter :: N2= 256
      integer :: NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL

      real(kind=8) :: EPSICCG

      real(kind=8), dimension(:  ), allocatable :: D, PHI, BFORCE
      real(kind=8), dimension(:,:), allocatable :: AL, AU

      integer, dimension(:), allocatable :: INL, INU, COLORindex
      integer, dimension(:), allocatable :: OLDtoNEW, NEWtoOLD

      integer, dimension(:,:), allocatable :: IAL, IAU

      end module PCG
```

扱う行列：疎行列
（自分の周辺のみと接続）
⇒ 非ゼロ成分のみを記憶する

上下三角成分を別々に記憶



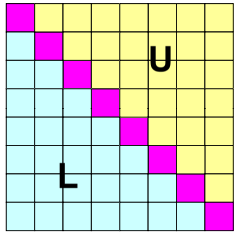
module PCG

```
      module PCG

      integer, parameter :: N2= 256
      integer :: NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL

      real(kind=8) :: EPSICCG

      real(kind=8), dimension(:  ), allocatable :: D, PHI, BFORCE
      real(kind=8), dimension(:,:), allocatable :: AL, AU

      integer, dimension(:), allocatable :: INL, INU, COLORindex
      integer, dimension(:), allocatable :: OLDtoNEW, NEWtoOLD

      integer, dimension(:,:), allocatable :: IAL, IAU

      end module PCG
```

下三角成分(列番号)：
非対角成分で自分より要素番号が小さい。
IAL(icou,i) < i

上三角成分(列番号)：
非対角成分で自分より要素番号が大きい。
IAU(icou,i) > i

INL(ICELTOT)	下三角成分の数
IAL(NL, ICELTOT)	下三角成分（列番号）
INU(ICELTOT)	上三角成分の数
IAU(NU, ICELTOT)	上三角成分（列番号）
NU,NL	上下三角成分の最大数（ここでは6）
NUmax,NLmax	未使用
NCOLORtot	色数，レベル数
COLORindex(0:NCOLORtot)	各色（レベル）に含まれる要素数の インデックス （COLORindex(icol)-COLORindex(icol-1)）
OLDtoNEW, NEWtoOLD	Coloring前後の要素番号対照表



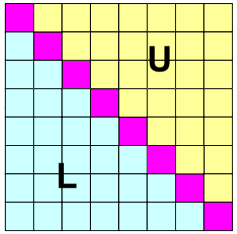
module PCG

```
      module PCG

      integer, parameter :: N2= 256
      integer :: NUmax, NLmax, NCOLORtot, NCOLORk, NU, NL

      real(kind=8) :: EPSICCG

      real(kind=8), dimension(:  ), allocatable :: D, PHI, BFORCE
      real(kind=8), dimension(:,:), allocatable :: AL, AU

      integer, dimension(:), allocatable :: INL, INU, COLORindex
      integer, dimension(:), allocatable :: OLDtoNEW, NEWtoOLD

      integer, dimension(:,:), allocatable :: IAL, IAU

      end module PCG
```

下三角成分(列番号)：
IAL(icou,i) < i
その個数が INL(i)

上三角成分(列番号)：
IAU(icou,i) > i
その個数がINU(i)

INL(ICELTOT)	下三角成分の数
IAL(NL, ICELTOT)	下三角成分（列番号）
INU(ICELTOT)	上三角成分の数
IAU(NU, ICELTOT)	上三角成分（列番号）
NU,NL	上下三角成分の最大数（ここでは6）
NUmax,NLmax	未使用
NCOLORtot	色数，レベル数
COLORindex(0:NCOLORtot)	各色（レベル）に含まれる要素数の インデックス （COLORindex(icol)-COLORindex(icol-1)）
OLDtoNEW, NEWtoOLD	Coloring前後の要素番号対照表



入力=3：実際は5色（multicolor）

color.logに行列関連情報出力

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

↓

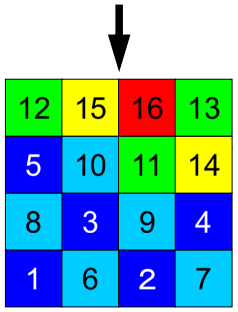
12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

```
### INITIAL connectivity
I=   1     INL(i)=   0     INU(i)=   2
  IAL:
  IAU:   2   5
I=   2     INL(i)=   1     INU(i)=   2
  IAL:   1
  IAU:   3   6
I=   3     INL(i)=   1     INU(i)=   2
  IAL:   2
  IAU:   4   7
I=   4     INL(i)=   1     INU(i)=   1
  IAL:   3
  IAU:   8
I=   5     INL(i)=   1     INU(i)=   2
  IAL:   1
  IAU:   6   9
I=   6     INL(i)=   2     INU(i)=   2
  IAL:   2   5
  IAU:   7  10
I=   7     INL(i)=   2     INU(i)=   2
  IAL:   3   6
  IAU:   8  11
I=   8     INL(i)=   2     INU(i)=   1
  IAL:   4   7
  IAU:  12
I=   9     INL(i)=   1     INU(i)=   2
  IAL:   5
  IAU:  10  13
I=  10     INL(i)=   2     INU(i)=   2
  IAL:   6   9
  IAU:  11  14
I=  11     INL(i)=   2     INU(i)=   2
  IAL:   7  10
  IAU:  12  15
I=  12     INL(i)=   2     INU(i)=   1
  IAL:   8  11
  IAU:  16
I=  13     INL(i)=   1     INU(i)=   1
  IAL:   9
  IAU:  14
I=  14     INL(i)=   2     INU(i)=   1
  IAL:  10  13
  IAU:  15
I=  15     INL(i)=   2     INU(i)=   1
  IAL:  11  14
  IAU:  16
I=  16     INL(i)=   2     INU(i)=   0
  IAL:  12  15
  IAU:


COLOR number       5

  #new       1   #old       1  color       1
  #new       2   #old       3  color       1
  #new       3   #old       6  color       1
  #new       4   #old       8  color       1
  #new       5   #old       9  color       1
  #new       6   #old       2  color       2
  #new       7   #old       4  color       2
  #new       8   #old       5  color       2
  #new       9   #old       7  color       2
  #new      10   #old      10  color       2
  #new      11   #old      11  color       3
  #new      12   #old      13  color       3
  #new      13   #old      16  color       3
  #new      14   #old      12  color       4
  #new      15   #old      14  color       4
  #new      16   #old      15  color       5
```

入力=3：実際は5色（multicolor）

color.logに行列関連情報出力

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4




```
### FINAL connectivity
I=    1      INL(i)=    0     INU(i)=    2
  IAL:
  IAU:    6    8
I=    2      INL(i)=    0     INU(i)=    3
  IAL:
  IAU:    6    7    9
I=    3      INL(i)=    0     INU(i)=    4
  IAL:
  IAU:    6    8    9   10
I=    4      INL(i)=    0     INU(i)=    3
  IAL:
  IAU:    7    9   14
I=    5      INL(i)=    0     INU(i)=    3
  IAL:
  IAU:    8   10   12
I=    6      INL(i)=    3     INU(i)=    0
  IAL:    1    2    3
  IAU:
I=    7      INL(i)=    2     INU(i)=    0
  IAL:    2    4
  IAU:
I=    8      INL(i)=    3     INU(i)=    0
  IAL:    1    3    5
  IAU:
I=    9      INL(i)=    3     INU(i)=    1
  IAL:    2    3    4
  IAU:   11
I=   10      INL(i)=    2     INU(i)=    2
  IAL:    3    5
  IAU:   11   15
I=   11      INL(i)=    2     INU(i)=    2
  IAL:    9   10
  IAU:   14   16
I=   12      INL(i)=    1     INU(i)=    1
  IAL:    5
  IAU:   15
I=   13      INL(i)=    0     INU(i)=    2
  IAL:
  IAU:   14   16
```

```
I=   14      INL(i)=    3     INU(i)=    0
  IAL:    4   11   13
  IAU:
I=   15      INL(i)=    2     INU(i)=    1
  IAL:   10   12
  IAU:   16
I=   16      INL(i)=    3     INU(i)=    0
  IAL:   11   15   13
  IAU:
```

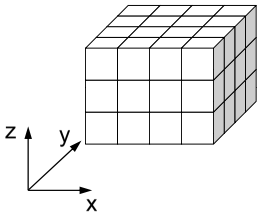

プログラムの構成

```
      program MAIN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

      call POINTER_INIT
      call POI_GEN

      call OUTUCD

      open (21, file='color.log', status='unknown')
        write (21,'(//,a,i8,/)') 'COLOR number', NCOLORtot
        do ic= 1, NCOLORtot
          do i= COLORindex(ic-1)+1, COLORindex(ic)
            write (21,'(3(a,i8))') '   #new', i, '    #old', NEWtoOLD(i), &
     &                             '   color', ic
          enddo
        enddo
      close (21)

      stop
      end
```


pointer_init（1/2）

```
!C
!C***
!C*** POINTER_INIT
!C***
!C
      subroutine POINTER_INIT

      use STRUCT
      use PCG
      implicit REAL*8 (A-H,O-Z)
      character*9 fname

      open (21, file="mesh.dat", status='unknown')

      read (21,'(10i10)') NX, NY, NZ
      read (21,'(10i10)') ICELTOT

      allocate (NEIBcell(ICELTOT,6), XYZ(ICELTOT,3))

      do i= 1, ICELTOT
        read (21,'(10i10)') ii, (NEIBcell(i,k), k= 1, 6),      &
     &                      (XYZ     (i,j), j= 1, 3)
      enddo
      close (21)

      NXP1= NX + 1
      NYP1= NY + 1
      NZP1= NZ + 1

      IBNODTOT= NXP1 * NYP1
      N       = NXP1 * NYP1 * NZP1

      return
      end
```

「mesh.dat」を読み込む



pointer_init（2/2）

```
!C
!C***
!C*** POINTER_INIT
!C***
!C
      subroutine POINTER_INIT

      use STRUCT
      use PCG
      implicit REAL*8 (A-H,O-Z)
      character*9 fname

      open (21, file="mesh.dat", status='unknown')

      read (21,'(10i10)') NX, NY, NZ
      read (21,'(10i10)') ICELTOT

      allocate (NEIBcell(ICELTOT,6), XYZ(ICELTOT,3))

      do i= 1, ICELTOT
        read (21,'(10i10)') ii, (NEIBcell(i,k), k= 1, 6),      &
     &                      (XYZ     (i,j), j= 1, 3)
      enddo
      close (21)

      NXP1= NX + 1
      NYP1= NY + 1
      NZP1= NZ + 1

      IBNODTOT= NXP1 * NYP1
      N       = NXP1 * NYP1 * NZP1

      return
      end
```

NXP1, NYP1, NZP1などはUCD
ファイル出力に必要

プログラムの構成

```
      program MAIN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

      call POINTER_INIT
      call POI_GEN

      call OUTUCD

      open (21, file='color.log', status='unknown')
        write (21,'(//,a,i8,/)') 'COLOR number', NCOLORtot
        do ic= 1, NCOLORtot
          do i= COLORindex(ic-1)+1, COLORindex(ic)
            write (21,'(3(a,i8))') '  #new', i, '    #old', NEWtoOLD(i), &
     &                             '  color', ic
          enddo
        enddo
      close (21)

      stop
      end
```

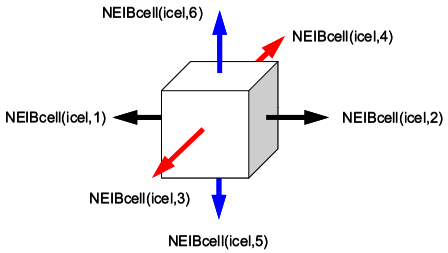

poi_gen（1/4）

```
      subroutine POI_GEN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

!C
!C-- INIT.
      nn = ICELTOT

      NU= 6
      NL= 6

      allocate (INL(nn), INU(nn), IAL(NL,nn), IAU(NU,nn))

      INL= 0
      INU= 0
      IAL= 0
      IAU= 0
```

配列の宣言


```
!C
!C +------------+
!C | CONNECTIVITY |
!C +------------+
!C==
      do icel= 1, ICELTOT
        icN1= NEIBcell(icel,1)
        icN2= NEIBcell(icel,2)
        icN3= NEIBcell(icel,3)
        icN4= NEIBcell(icel,4)
        icN5= NEIBcell(icel,5)
        icN6= NEIBcell(icel,6)

        icouG= 0
        if (icN5.ne.0.and.icN5.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN5
          INL(    icel)= icou
        endif

        if (icN3.ne.0.and.icN3.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN3
          INL(    icel)= icou
        endif

        if (icN1.ne.0.and.icN1.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN1
          INL(    icel)= icou
        endif
```

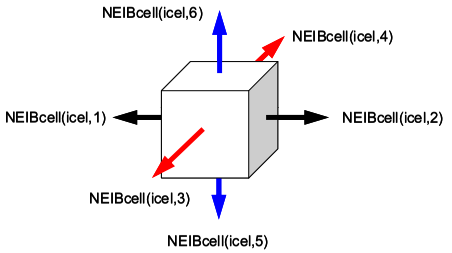
poi_gen（2/4）



下三角成分
NEIBcell(icel,5)= icel – NX*NY
NEIBcell(icel,3)= icel – NX
NEIBcell(icel,1)= icel – 1


```
!C
!C +------------+
!C | CONNECTIVITY |
!C +------------+
!C==
      do icel= 1, ICELTOT
        icN1= NEIBcell(icel,1)
        icN2= NEIBcell(icel,2)
        icN3= NEIBcell(icel,3)
        icN4= NEIBcell(icel,4)
        icN5= NEIBcell(icel,5)
        icN6= NEIBcell(icel,6)

...

        if (icN2.ne.0.and.icN2.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN2
          INU(    icel)= icou
        endif

        if (icN4.ne.0.and.icN4.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN4
          INU(    icel)= icou
        endif

        if (icN6.ne.0.and.icN6.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN6
          INU(    icel)= icou
        endif
      enddo
!C==
```

poi_gen（3/4）



上三角成分
NEIBcell(icel,2)= icel + 1
NEIBcell(icel,4)= icel + NX
NEIBcell(icel,6)= icel + NX*NY

poi_gen（4/4）

```
!C
!C +-------------+
!C | MULTICOLORING |
!C +-------------+
!C==
 111  continue
      write (*,'(//a,i8,a)') 'You have', ICELTOT, ' elements.'
      write (*,'( a     )') 'How many colors do you need ?'
      write (*,'( a     )') '  #COLOR must be more than 2 and'
      write (*,'( a,i8  )') '  #COLOR must not be more than', ICELTOT
      write (*,'( a     )') '  if #COLOR=0 :  CM ordering'
      write (*,'( a     )') '  if #COLOR<0 : RCM ordering'
      write (*,'( a     )') '=>'
      read  (*,*)             NCOLORtot
      if (NCOLORtot.eq.1.or.NCOLORtot.gt.ICELTOT) goto 111

      allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))
      allocate (COLORindex(0:ICELTOT))

      if (NCOLORtot.gt.0) then
        call MC (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.eq.0) then
        call  CM (ICELTOT, NL, NU, INL, IAL, INU, IAU,               &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.lt.0) then
        call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,               &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
!C==

      write (*,'(/a, i8)') '# TOTAL COLOR number', NCOLORtot
```

色数を読み込む

poi_gen（4/4）

```
!C
!C +-------------+
!C | MULTICOLORING |
!C +-------------+
!C==
 111  continue
      write (*,'(//a,i8,a)') 'You have', ICELTOT, ' elements.'
      write (*,'( a     )') 'How many colors do you need ?'
      write (*,'( a     )') '  #COLOR must be more than 2 and'
      write (*,'( a,i8  )') '  #COLOR must not be more than', ICELTOT
      write (*,'( a     )') '  if #COLOR=0 :  CM ordering'
      write (*,'( a     )') '  if #COLOR<0 : RCM ordering'
      write (*,'( a     )') '=>'
      read  (*,*)             NCOLORtot
      if (NCOLORtot.eq.1.or.NCOLORtot.gt.ICELTOT) goto 111

      allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))
      allocate (COLORindex(0:ICELTOT))

      if (NCOLORtot.gt.0) then
        call MC (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.eq.0) then
        call  CM (ICELTOT, NL, NU, INL, IAL, INU, IAU,               &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.lt.0) then
        call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,               &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
!C==

      write (*,'(/a, i8)') '# TOTAL COLOR number', NCOLORtot
```

配列宣言を実施する。

poi_gen（4/4）

```
!C
!C +-------------+
!C | MULTICOLORING |
!C +-------------+
!C==

...
      if (NCOLORtot.gt.0) then
        call MC (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.eq.0) then
        call  CM (ICELTOT, NL, NU, INL, IAL, INU, IAU,               &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.lt.0) then
        call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,               &
     &           NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
!C==

      write (*,'(/a, i8)') '# TOTAL COLOR number', NCOLORtot
```

INL, INU, IAL, IAU	再番号付け後の情報が入る
OLDtoNEW, NEWtoOLD	新旧要素番号対象表
NCOLORtot	色数（入力値と同じかそれより大きくなる）
COLORindex(0:NCOLORtot)	COLORindex(ic-1)+1からCOLORindex(ic)までの要素（再番号付け後）が「ic」色になる。 同じ色の要素は互いに独立：並列計算可能

COLORindex

COLORindex(0:NCOLORtot)	COLORindex(ic-1)+1からCOLORindex(ic)までの要素（再番号付け後）が「ic」色になる。 同じ色の要素は互いに独立：並列計算可能

```
do ic= 1, NCOLORtot
  do i= COLORindex(ic-1)+1, COLORindex(ic)
    write (21,*) i, NEWtoOLD(i), ic
  enddo
enddo
```

```
COLOR number       5
  #new       1   #old       1  color       1
  #new       2   #old       3  color       1
  #new       3   #old       6  color       1
  #new       4   #old       8  color       1
  #new       5   #old       9  color       1
  #new       6   #old       2  color       2
  #new       7   #old       4  color       2
  #new       8   #old       5  color       2
  #new       9   #old       7  color       2
  #new      10   #old      10  color       2
  #new      11   #old      11  color       3
  #new      12   #old      13  color       3
  #new      13   #old      16  color       3
  #new      14   #old      12  color       4
  #new      15   #old      14  color       4
  #new      16   #old      15  color       5
```

mc(1/8)

```
!C
!C***
!C*** MC
!C***
!C
!C    Multicolor Ordering Method
!C

      subroutine MC (N, NL, NU, INL, IAL, INU, IAU,             &
     &               NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)

      implicit REAL*8(A-H,O-Z)

      integer, dimension(N)   :: INL, INU, NEWtoOLD, OLDtoNEW
      integer, dimension(0:N) :: COLORindex
      integer, dimension(NL,N):: IAL
      integer, dimension(NU,N):: IAU

      integer, dimension(:) , allocatable :: IW, INLw, INUw
      integer, dimension(:,:), allocatable :: IALw, IAUw
```


mc(2/8)

```
!C
!C +------+
!C | INIT. |
!C +------+
!C===
      allocate (IW(N))
      IW= 0

      NCOLORk  = NCOLORtot

      do i= 1, N
        NEWtoOLD  (i)= i
        OLDtoNEW  (i)= i
      enddo

       INmin= N
      NODmin= 0
      do i= 1, N
        icon= INL(i) + INU(i)
        if (icon.lt.INmin) then
           INmin= icon
          NODmin= i
        endif
      enddo

      OLDtoNEW(NODmin)= 1
      NEWtoOLD(     1)= NODmin
      IW        = 0
      IW(NODmin)= 1

      ITEMcou= N/NCOLORk
!C===
```

作業配列「IW」に「0」を入れる
IWには各要素の色番号が入る

接続要素数(次数)が最小の要素を探索
NODmin

mc(2/8)

```
!C
!C +------+
!C | INIT. |
!C +------+
!C===
      allocate (IW(N))
      IW= 0

      NCOLORk  = NCOLORtot

      do i= 1, N
        NEWtoOLD  (i)= i
        OLDtoNEW  (i)= i
      enddo

       INmin= N
      NODmin= 0
      do i= 1, N
        icon= INL(i) + INU(i)
        if (icon.lt.INmin) then
           INmin= icon
          NODmin= i
        endif
      enddo

      OLDtoNEW(NODmin)= 1
      NEWtoOLD(     1)= NODmin
      IW        = 0
      IW(NODmin)= 1

      ITEMcou= N/NCOLORk
!C===
```

接続要素数が最小の要素をオーダリング後の「1」番とする。対照表を更新。

作業配列「IW(1)」に「1(色番号)」を入れる。

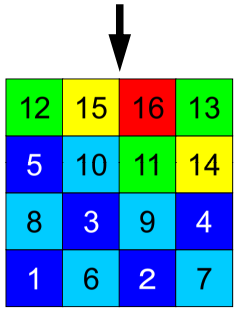
mc(2/8)

```
!C
!C +------+
!C | INIT. |
!C +------+
!C===
      allocate (IW(N))
      IW= 0

      NCOLORk  = NCOLORtot

      do i= 1, N
        NEWtoOLD  (i)= i
        OLDtoNEW  (i)= i
      enddo

       INmin= N
      NODmin= 0
      do i= 1, N
        icon= INL(i) + INU(i)
        if (icon.lt.INmin) then
           INmin= icon
          NODmin= i
        endif
      enddo

      OLDtoNEW(NODmin)= 1
      NEWtoOLD(     1)= NODmin
      IW        = 0
      IW(NODmin)= 1

      ITEMcou= N/NCOLORk
!C===
```

各色に含まれる要素数の目安

入力=3：実際は5色（multicolor）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4




```
#new     1   #old     1  color     1
#new     2   #old     3  color     1
#new     3   #old     6  color     1
#new     4   #old     8  color     1
#new     5   #old     9  color     1
#new     6   #old     2  color     2
#new     7   #old     4  color     2
#new     8   #old     5  color     2
#new     9   #old     7  color     2
#new    10   #old    10  color     2
#new    11   #old    11  color     3
#new    12   #old    13  color     3
#new    13   #old    16  color     3
#new    14   #old    12  color     4
#new    15   #old    14  color     4
#new    16   #old    15  color     5
```

16/3=5 = **ITEMcou**

最終的に5色必要であった


```
!C
!C +-------------+
!C | MULTIcoloring |
!C +-------------+
!C===
      icou = 1
      icouK= 1

      do icol= 1, N
        NCOLORk= icol
        do i= 1, N
          if (IW(i).le.0) IW(i)= 0
        enddo

        do i= 1, N
!C
!C-- already COLORED
          if (IW(i).eq.icol) then
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
!C
!C-- not COLORED
          if (IW(i).eq.0) then
            icou = icou  + 1
            icouK= icouK + 1
            IW(i)= icol
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
```

mc（3/8）

カウンタ初期化

`icou` ： 全体のカウンタ
`icouK` ： 色内のカウンタ


```
!C
!C +-------------+
!C | MULTIcoloring |
!C +-------------+
!C===
      icou = 1
      icouK= 1

      do icol= 1, N
        NCOLORk= icol
        do i= 1, N
          if (IW(i).le.0) IW(i)= 0
        enddo

        do i= 1, N
!C
!C-- already COLORED
          if (IW(i).eq.icol) then
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
!C
!C-- not COLORED
          if (IW(i).eq.0) then
            icou = icou  + 1
            icouK= icouK + 1
            IW(i)= icol
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
```

mc（3/5）

色数に関するループ


```
!C
!C +-------------+
!C | MULTIcoloring |
!C +-------------+
!C===
      icou = 1
      icouK= 1

      do icol= 1, N
        NCOLORk= icol
        do i= 1, N
          if (IW(i).le.0) IW(i)= 0
        enddo

        do i= 1, N
!C
!C-- already COLORED
          if (IW(i).eq.icol) then
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
!C
!C-- not COLORED
          if (IW(i).eq.0) then
            icou = icou  + 1
            icouK= icouK + 1
            IW(i)= icol
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
```

mc（3/8）

現在の色数を「NCOLORk」とする。

色づけされていない要素の「IW」の値を0とする。

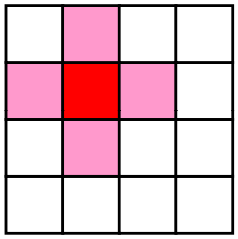
mc(3/8)

```
!C
!C +-------------+
!C | MULTIcoloring |
!C +-------------+
!C===
      icou = 1
      icouK= 1

      do icol= 1, N
        NCOLORk= icol
        do i= 1, N
          if (IW(i).le.0) IW(i)= 0
        enddo

        do i= 1, N
!C
!C-- already COLORED
          if (IW(i).eq.icol) then
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
!C
!C-- not COLORED
          if (IW(i).eq.0) then
            icou = icou  + 1
            icouK= icouK + 1
            IW(i)= icol
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
```

全要素に関するループ。

すでに現在の色に色づけされている場合は，隣接する要素のIWの値を「-1」とする（実際にこの部分を通る可能性は最初の一回のみであるが，念のため）。

すでに「現在の色」に色づけされている要素の隣接要素は「現在の色」に入る可能性が無いため除外。



mc(3/8)

```
!C
!C +-------------+
!C | MULTIcoloring |
!C +-------------+
!C===
      icou = 1
      icouK= 1

      do icol= 1, N
        NCOLORk= icol
        do i= 1, N
          if (IW(i).le.0) IW(i)= 0
        enddo

        do i= 1, N
!C
!C-- already COLORED
          if (IW(i).eq.icol) then
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
!C
!C-- not COLORED
          if (IW(i).eq.0) then
            icou = icou  + 1
            icouK= icouK + 1
            IW(i)= icol
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
```

「IW(i)=0」の場合，カウンタを1つずつ増やし，自分の色をicolとする（IW(i)= icol）。

隣接する要素のIWの値を「-1」とする。

mc(4/8)

```
      do icol= 1, N
        NCOLORk= icol
        do i= 1, N
          if (IW(i).le.0) IW(i)= 0
        enddo

        do i= 1, N
!C
!C-- already COLORED
          if (IW(i).eq.icol) then
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
!C
!C-- not COLORED
          if (IW(i).eq.0) then
            icou = icou  + 1
            icouK= icouK + 1
            IW(i)= icol
            do k= 1, INL(i)
              ik= IAL(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
            do k= 1, INU(i)
              ik= IAU(k,i)
              if (IW(ik).le.0) IW(ik)= -1
            enddo
          endif
          if (icou .eq.N)       goto 200
          if (icouK.eq.ITEMcou) goto 100
        enddo

 100    continue
        icouK= 0
      enddo
 200  continue
```

icou ： 全体のカウンタ
icouK ： 色内のカウンタ

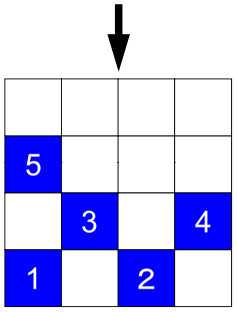
要素数のカウンタ(icou)が「N(ICELTOT)」を超えたらループを抜ける（全要素の色付け完了）。

色内のカウンタ(icouK)が「ITEMcou」を超えたら「icouK=0」として次の色へ移る。

「icouK < ITEMcou」でも「i」が「N」に到達したら（独立な要素がもうないので）次の色へ移る。

入力=3：実際は5色（multicolor）

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

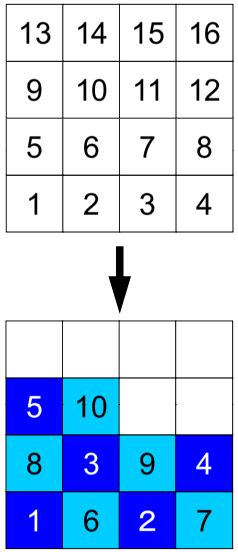



```
#new     1   #old     1   color     1
#new     2   #old     3   color     1
#new     3   #old     6   color     1
#new     4   #old     8   color     1
#new     5   #old     9   color     1
#new     6   #old     2   color     2
#new     7   #old     4   color     2
#new     8   #old     5   color     2
#new     9   #old     7   color     2
#new    10   #old    10   color     2
#new    11   #old    11   color     3
#new    12   #old    13   color     3
#new    13   #old    16   color     3
#new    14   #old    12   color     4
#new    15   #old    14   color     4
#new    16   #old    15   color     5
```

16/3=5

「5」個ずつ独立な要素を元の番号順に選択

入力=3：実際は5色（multicolor）

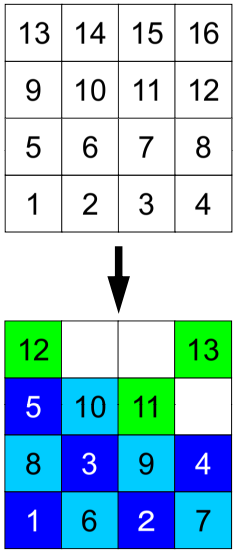



```
#new     1   #old     1  color     1
#new     2   #old     3  color     1
#new     3   #old     6  color     1
#new     4   #old     8  color     1
#new     5   #old     9  color     1
#new     6   #old     2  color     2
#new     7   #old     4  color     2
#new     8   #old     5  color     2
#new     9   #old     7  color     2
#new    10   #old    10  color     2
#new    11   #old    11  color     3
#new    12   #old    13  color     3
#new    13   #old    16  color     3
#new    14   #old    12  color     4
#new    15   #old    14  color     4
#new    16   #old    15  color     5
```

16/3=5

「5」個ずつ独立な要素を元の番号順に選択

入力=3：実際は5色（multicolor）

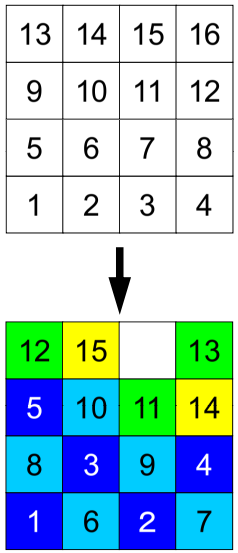



```
#new     1   #old     1  color     1
#new     2   #old     3  color     1
#new     3   #old     6  color     1
#new     4   #old     8  color     1
#new     5   #old     9  color     1
#new     6   #old     2  color     2
#new     7   #old     4  color     2
#new     8   #old     5  color     2
#new     9   #old     7  color     2
#new    10   #old    10  color     2
#new    11   #old    11  color     3
#new    12   #old    13  color     3
#new    13   #old    16  color     3
#new    14   #old    12  color     4
#new    15   #old    14  color     4
#new    16   #old    15  color     5
```

16/3=5

独立な要素が無くなったら次の色へ

入力=3：実際は5色（multicolor）




```
#new     1   #old     1  color     1
#new     2   #old     3  color     1
#new     3   #old     6  color     1
#new     4   #old     8  color     1
#new     5   #old     9  color     1
#new     6   #old     2  color     2
#new     7   #old     4  color     2
#new     8   #old     5  color     2
#new     9   #old     7  color     2
#new    10   #old    10  color     2
#new    11   #old    11  color     3
#new    12   #old    13  color     3
#new    13   #old    16  color     3
#new    14   #old    12  color     4
#new    15   #old    14  color     4
#new    16   #old    15  color     5
```

16/3=5

独立な要素が無くなったら次の色へ


```
!C
!C +-------------+
!C | FINAL COLORING |
!C +-------------+
!C===
      NCOLORtot= NCOLORk
      COLORindex= 0
      icoug= 0
      do ic= 1, NCOLORtot
        icou= 0
        do i= 1, N
          if (IW(i).eq.ic) then
            icou = icou + 1
            icoug= icoug + 1
            NEWtoOLD(icoug)= i
            OLDtoNEW(i    )= icoug
          endif
        enddo
        COLORindex(ic)= icou
      enddo

      do ic= 1, NCOLORtot
        COLORindex(ic)= COLORindex(ic-1) + COLORindex(ic)
      enddo
!C===
```

mc（5/8）

色づけを終了した時点での色数「NCOLORk」を最終的な色数とする。
ユーザーが最初に設定した色数と同じかそれより多くなっている。

mc(5/8)

```
!C
!C +--------------+
!C | FINAL COLORING |
!C +--------------+
!C===
      NCOLORtot= NCOLORk
      COLORindex= 0
      icoug= 0
      do ic= 1, NCOLORtot
        icou= 0
        do i= 1, N
          if (IW(i).eq.ic) then
            icou = icou + 1
            icoug= icoug + 1
            NEWtoOLD(icoug)= i
            OLDtoNEW(i    )= icoug
          endif
        enddo
        COLORindex(ic)= icou
      enddo

      do ic= 1, NCOLORtot
        COLORindex(ic)= COLORindex(ic-1) + COLORindex(ic)
      enddo
!C===
```

各要素の色に従って，色番号の少ない方から，要素の再番号付けを行なう。

```
OLDtoNEW(old_ID)= new_ID
NEWtoOLD(new_ID)= old_ID
```

この時点では「COLORindex」には各色の要素数が入っている。

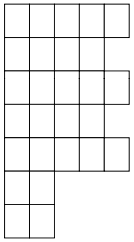
mc(6/8)

```
!C
!C +--------------+
!C | FINAL COLORING |
!C +--------------+
!C===
      NCOLORtot= NCOLORk
      COLORindex= 0
      icoug= 0
      do ic= 1, NCOLORtot
        icou= 0
        do i= 1, N
          if (IW(i).eq.ic) then
            icou = icou + 1
            icoug= icoug + 1
            NEWtoOLD(icoug)= i
            OLDtoNEW(i    )= icoug
          endif
        enddo
        COLORindex(ic)= icou
      enddo

      do ic= 1, NCOLORtot
        COLORindex(ic)= COLORindex(ic-1) + COLORindex(ic)
      enddo
!C===
```

「COLORindex」を一次元インデックスに置き換える。

mc(6/8)

```
!C
!C +----------------+
!C | MATRIX transfer |
!C +----------------+
!C===
      allocate (INLw(N), INUw(N), IALw(NL,N), IAUw(NU,N))

      do j= 1, NL
        do i= 1, N
          IW(i) = IAL(j,NEWtoOLD(i))
        enddo
        do i= 1, N
          IAL(j,i) = IW(i)
        enddo
      enddo

      do j= 1, NU
        do i= 1, N
          IW(i) = IAU(j,NEWtoOLD(i))
        enddo
        do i= 1, N
          IAU(j,i) = IW(i)
        enddo
      enddo
```



ワーク配列の定義
- INLw(N)
- INUw(N)
- IALw(NL,N)
- IAUw(NU,N)

上下三角成分を，新しい番号付けに従って入れ替える。上下三角成分そのものの番号はそのまま。

mc(7/8)

```
      do i= 1, N
        INLw(i) = INL(NEWtoOLD(i))
      enddo

      do i= 1, N
        INUw(i) = INU(NEWtoOLD(i))
      enddo

      do j= 1, NL
        do i= 1, N
          if (IAL(j,i).eq.0) then
            IALw(j,i) = 0
          else
            IALw(j,i) = OLDtoNEW(IAL(j,i))
          endif
        enddo
      enddo

      do j= 1, NU
        do i= 1, N
          if ( IAU(j,i).eq.0) then
            IAUw(j,i) = 0
          else
            IAUw(j,i) = OLDtoNEW(IAU(j,i))
          endif
        enddo
      enddo
```

上下三角成分の数を，新しい番号付けに従って入れ替える。

INLを並び替えてIWに格納し，更にINLw に格納する。

INUを並び替えてIWに格納し，更にINUw に格納する。

mc(7/8)

```
      do i= 1, N
        INLw(i) = INL(NEWtoOLD(i))
      enddo

      do i= 1, N
        INUw(i) = INU(NEWtoOLD(i))
      enddo

      do j= 1, NL
        do i= 1, N
          if (IAL(j,i).eq.0) then
            IALw(j,i) = 0
          else
            IALw(j,i) = OLDtoNEW(IAL(j,i))
          endif
        enddo
      enddo

      do j= 1, NU
        do i= 1, N
          if ( IAU(j,i).eq.0) then
            IAUw(j,i) = 0
          else
            IAUw(j,i) = OLDtoNEW(IAU(j,i))
          endif
        enddo
      enddo
```

上下三角成分を，新しい番号付けに
従って新しい番号に付け替える。

IALw，IAUw に格納する

mc(8/8)

```
      INL= 0
      INU= 0
      IAL= 0
      IAU= 0

      do i= 1, N
        jL= 0
        jU= 0
        do j= 1, INLw(i)
          if (IALw(j,i).gt.i) then
            jU= jU + 1
            IAU(jU,i)= IALw(j,i)
          else
            jL= jL + 1
            IAL(jL,i)= IALw(j,i)
          endif
        enddo

        do j= 1, INUw(i)
          if (IAUw(j,i).gt.i) then
            jU= jU + 1
            IAU(jU,i)= IAUw(j,i)
          else
            jL= jL + 1
            IAL(jL,i)= IAUw(j,i)
          endif
        enddo

        INL(i)= jL
        INU(i)= jU
      enddo
!C===
      deallocate (IW, INLw, INUw, IALw, IAUw)

      return
      end
```

もともとの下三角成分にたいする処理。

```
do i= 1, N
  jL= 0
  jU= 0
  do j= 1, INLw(i)
    if (IALw(j,i).gt.i) then
      jU= jU + 1
      IAU(jU,i)= IALw(j,i)
    else
      jL= jL + 1
      IAL(jL,i)= IALw(j,i)
    endif
  enddo

enddo
```

新しく番号がついた IALw(j,i)
がiより大きいか小さいかによって
上三角成分と下三角成分に
振り分ける。

この操作が必要な理由

Original

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

INL(7)= 2
IAL(1,7)= 3, IAL(2,7)= 6

INU(7)= 2
IAU(1,7)= 8, IAU(2,7)=11

5 Color

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

INL(9)= 3
IAL(1,9)= 2, IAL(2,9)= 3
IAL(3,9)= 4

INU(9)= 1
IAU(1,9)=11

再番号付けによって隣接要素との大小関係も変わってしまうため

mc(8/8)

```
      INL= 0
      INU= 0
      IAL= 0
      IAU= 0

      do i= 1, N
        jL= 0
        jU= 0
        do j= 1, INLw(i)
          if (IALw(j,i).gt.i) then
            jU= jU + 1
            IAU(jU,i)= IALw(j,i)
          else
            jL= jL + 1
            IAL(jL,i)= IALw(j,i)
          endif
        enddo

        do j= 1, INUw(i)
          if (IAUw(j,i).gt.i) then
            jU= jU + 1
            IAU(jU,i)= IAUw(j,i)
          else
            jL= jL + 1
            IAL(jL,i)= IAUw(j,i)
          endif
        enddo

        INL(i)= jL
        INU(i)= jU
      enddo
!C===
      deallocate (IW, INLw, INUw, IALw, IAUw)

      return
      end
```

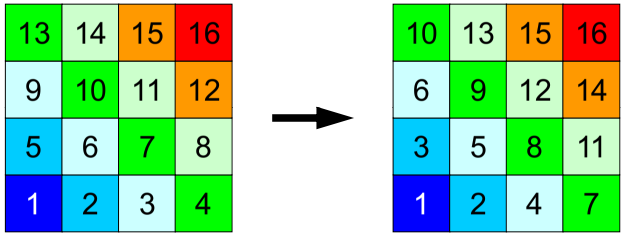
もともとの上三角成分にたいする処理。

mc（8/8）

```
      INL= 0
      INU= 0
      IAL= 0
      IAU= 0

      do i= 1, N
        jL= 0
        jU= 0
        do j= 1, INLw(i)
          if (IALw(j,i).gt.i) then
            jU= jU + 1
            IAU(jU,i)= IALw(j,i)
          else
            jL= jL + 1
            IAL(jL,i)= IALw(j,i)
          endif
        enddo

        do j= 1, INUw(i)
          if (IAUw(j,i).gt.i) then
            jU= jU + 1
            IAU(jU,i)= IAUw(j,i)
          else
            jL= jL + 1
            IAL(jL,i)= IAUw(j,i)
          endif
        enddo

        INL(i)= jL
        INU(i)= jU
      enddo
!C===
      deallocate (IW, INLw, INUw, IALw, IAUw)

      return
      end
```

上下三角成分の数。

CM法の手順

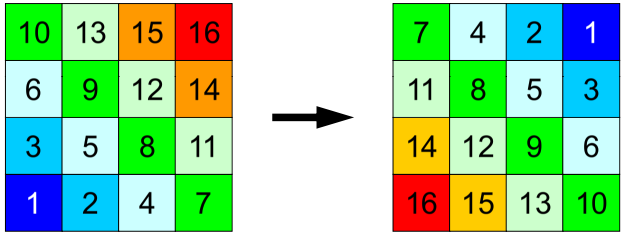


- 手順3：繰り返し，再番号付け
  - 「レベル（k）」に属する条件を満たす要素が無くなったら，k=k+1 として，手順2を繰り返し，全要素の「レベル」が決定したら終了
  - 「レベル」の若い順に要素番号をふり直す

RCM（Reverse CM）

- まずCMの手順を実行
  - 手順1：次数（degree）の計算
  - 手順2：「レベル（k（k≧2））」の要素の選択
  - 手順3：繰り返し，再番号付け

- 手順4：再々番号付け
  - CMの番号付けを更に逆順にふり直す
  - Fill-inがCMの場合より少なくなる



cm（1/5）

```
!C
!C***
!C*** CM
!C***
!C
      subroutine CM  (N, NL, NU, INL, IAL, INU, IAU,
     &                NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)

      implicit REAL*8(A-H,O-Z)
      integer, dimension(N)    :: INL, INU, NEWtoOLD, OLDtoNEW
      integer, dimension(0:N)  :: COLORindex
      integer, dimension(NL,N) :: IAL
      integer, dimension(NU,N) :: IAU

      integer, dimension(:,:), allocatable :: IW
      integer, dimension(:)  , allocatable :: INLw, INUw
      integer, dimension(:,:), allocatable :: IALw, IAUw
```

cm(2/5)

```
!C
!C +------+
!C | INIT. |
!C +------+
!C==
      allocate (IW(N,2))

      IW = 0

       INmin= N
      NODmin= 0

      do i= 1, N
        icon= 0
        do k= 1, INL(i)
          icon= icon + 1
        enddo
        do k= 1, INU(i)
          icon= icon + 1
        enddo

        if (icon.lt.INmin) then
          INmin = icon
          NODmin= i
        endif
      enddo
 200  continue


      if (NODmin.eq.0) NODmin= 1

      IW(NODmin,2)= 1

      NEWtoOLD(1     )= NODmin
      OLDtoNEW(NODmin)= 1

      icol= 1
!C==
```

補助配列

IW(i,1):ワーク配列
IW(i,2):要素iの所属レベル

cm(2/5)

```
!C
!C +------+
!C | INIT. |
!C +------+
!C==
      allocate (IW(N,2))

      IW = 0

       INmin= N
      NODmin= 0

      do i= 1, N
        icon= 0
        do k= 1, INL(i)
          icon= icon + 1
        enddo
        do k= 1, INU(i)
          icon= icon + 1
        enddo

        if (icon.lt.INmin) then
          INmin = icon
          NODmin= i
        endif
      enddo
 200  continue


      if (NODmin.eq.0) NODmin= 1

      IW(NODmin,2)= 1

      NEWtoOLD(1     )= NODmin
      OLDtoNEW(NODmin)= 1

      icol= 1
!C==
```

接続要素数(次数)が最小の要素を探索
NODmin

cm(2/5)

```
!C
!C +------+
!C | INIT. |
!C +------+
!C==
      allocate (IW(N,2))

      IW = 0

       INmin= N
      NODmin= 0

      do i= 1, N
        icon= 0
        do k= 1, INL(i)
          icon= icon + 1
        enddo
        do k= 1, INU(i)
          icon= icon + 1
        enddo

        if (icon.lt.INmin) then
          INmin = icon
          NODmin= i
        endif
      enddo
 200  continue


      if (NODmin.eq.0) NODmin= 1

      IW(NODmin,2)= 1

      NEWtoOLD(1     )= NODmin
      OLDtoNEW(NODmin)= 1

      icol= 1
!C==
```

NODminの所属レベル「IW(NODmin,2」を「1」とする。

NODminを新しい番号付けにおける「1」番の要素とする。

cm(3/5)

```
!C
!C +--------+
!C | COLORING |
!C +--------+
!C==
      icouG= 1
      do icol= 2, N
        icou = 0
        do i= 1, N
          if (IW(i,2).eq.icol-1) then
            do k= 1, INL(i)
              in= IAL(k,i)
              if (IW(in,2).eq.0) then
                IW(in  ,2)= -icol
                icou      = icou + 1
                IW(icou,1)= in
              endif
            enddo
            do k= 1, INU(i)
              in= IAU(k,i)
              if (IW(in,2).eq.0) then
                IW(in  ,2)= -icol
                icou      = icou + 1
                IW(icou,1)= in
              endif
            enddo
          endif
        enddo

        if (icou.eq.0) then
          do i= 1, N
            if (IW(i,2).eq.0) then
              icou= icou + 1
              IW(i   ,2)= -icol
              IW(icou,1)= i
              goto 850
            endif
          enddo
        endif
 850    continue
```

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

「レベル」に関するループ

```
!C
!C +---------+
!C | COLORING |
!C +---------+
!C===
      icouG= 1
      do icol= 2, N
        icou = 0
        do i= 1, N
          if (IW(i,2).eq.icol-1) then
            do k= 1, INL(i)
              in= IAL(k, i)
              if (IW(in,2).eq.0) then
                IW(in  ,2)= -icol
                icou      = icou + 1
                IW(icou,1)= in
              endif
            enddo
            do k= 1, INU(i)
              in= IAU(k, i)
              if (IW(in,2).eq.0) then
                IW(in  ,2)= -icol
                icou      = icou + 1
                IW(icou,1)= in
              endif
            enddo
          endif
        enddo

        if (icou.eq.0) then
          do i= 1, N
            if (IW(i,2).eq.0) then
              icou= icou + 1
              IW(i   ,2)= -icol
              IW(icou,1)= i
              goto 850
            endif
          enddo
        endif
  850   continue
```

cm(3/5)

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

「レベル」内での，各要素に関するループ

IW(i,2)=icol-1
(所属レベル番号：icol-1)である要素i
に接続しており，かつ，

所属レベルが決定していない要素inに
関して，所属レベル番号「icol」の要素の
候補として，IW(in,2)= -icol とする。

レベル内カウンタ icou=icou+1 とする。

また，「見つかった」順番に：
IW(ic,1)= in (ic= 1~icou)
とする。

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

icol=4
IW(i,2)= icol-1=3: i=3,6,9

どういうことかというと

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

「レベル」内での，各要素に関するループ

IW(i,2)=icol-1
(所属レベル番号：icol-1)である要素i
に接続しており，かつ，

所属レベルが決定していない要素inに
関して，所属レベル番号「icol」の要素の
候補として，IW(in,2)= -icol とする。

レベル内カウンタ icou=icou+1 とする。

また，「見つかった」順番に：
IW(ic,1)= in (ic= 1~icou)
とする。

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

icol=4
IW(i,2)= icol-1=3: i=3,6,9

```
IW( 4,2)= -4
IW( 7,2)= -4
IW(10,2)= -4
IW(13,2)= -4

IW(1,1)=  4
IW(2,1)=  7
IW(3,1)= 10
IW(4,1)= 13
```

どういうことかというと

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

「レベル」内での，各要素に関するループ

IW(i,2)=icol-1
(所属レベル番号：icol-1)である要素i
に接続しており，かつ，

所属レベルが決定していない要素inに
関して，所属レベル番号「icol」の要素の
候補として，IW(in,2)= -icol とする。

レベル内カウンタ icou=icou+1 とする。

また，「見つかった」順番に：
IW(ic,1)= in (ic= 1~icou)
とする。


```
!C
!C +---------+
!C | COLORING |
!C +---------+
!C===
      icouG= 1
      do icol= 2, N
        icou = 0
        do i= 1, N
          if (IW(i,2).eq.icol-1) then
            do k= 1, INL(i)
              in= IAL(k, i)
              if (IW(in,2).eq.0) then
                IW(in  ,2)= -icol
                icou      = icou + 1
                IW(icou,1)= in
              endif
            enddo
            do k= 1, INU(i)
              in= IAU(k, i)
              if (IW(in,2).eq.0) then
                IW(in  ,2)= -icol
                icou      = icou + 1
                IW(icou,1)= in
              endif
            enddo
          endif
        enddo

        if (icou.eq.0) then
          do i= 1, N
            if (IW(i,2).eq.0) then
              icou= icou + 1
              IW(i   ,2)= -icol
              IW(icou,1)= i
              goto 850
            endif
          enddo
        endif
  850   continue
```

cm(3/5)

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

「レベル」内での，各要素に関するループ

もし icou=0 となったら，まだ所属レベル
が決定しない要素の中で最も要素番号が
小さいものを選び出す（通常はこのループ
は通らない）

cm（4/5）

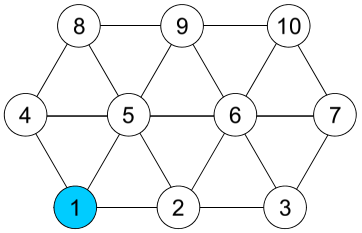
```
!C
!C +----------+
!C | COLORING |
!C +----------+
!C===

...

        do ic= 1, icou
          inC= IW(ic,1)
          if (IW(inC,2).ne.0) then
            do k= 1, INL(inC)
              in= IAL(k,inC)
              if (IW(in,2).le.0) IW(in,2)= 0
            enddo
            do k= 1, INU(inC)
              in= IAU(k,inC)
              if (IW(in,2).le.0) IW(in,2)= 0
            enddo
          endif
        enddo

        do ic= 1, icou
          inC= IW(ic,1)
          if (IW(inC,2).ne.0) then
            icouG      = icouG + 1
            IW(inC,2)= icol
          endif
        enddo

        if (icouG.eq.N) exit
      enddo
!C===
```

所属レベル「icol」の候補となっている要素の番号が，IW(ic,1)(ic=1~icou)に格納されている。

各要素inC=IW(ic,1)(ic=1~icou)に隣接する要素inのうちで，所属レベル「icol」の候補となっているものがある場合，要素inを候補の中からはずす。
（隣接する要素同士は同じレベルには属することができない）

そのような要素に対して：
IW(in,2)=0
とする

どういうことかというと



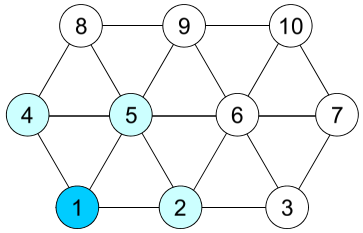
例えば①の所属レベル=icol-1とする

所属レベル「icol」の候補となっている要素の番号が，IW(ic,1)(ic=1~icou)に格納されている。

各要素inC=IW(ic,1)(ic=1~icou)に隣接する要素inのうちで，所属レベル「icol」の候補となっているものがある場合，要素inを候補の中からはずす。
（隣接する要素同士は同じレベルには属することができない）

そのような要素に対して：
IW(in,2)=0
とする

どういうことかというと



所属レベル「icol」の候補となる節点は②，④，⑤の3点，したがって：

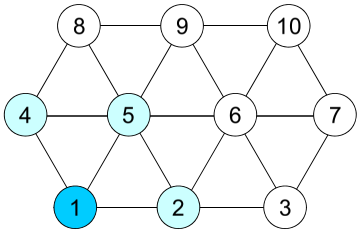
```
IW(2,2)= -icol
IW(4,2)= -icol
IW(5,2)= -icol

IW(1,1)= 2
IW(2,1)= 4
IW(3,1)= 5
```

所属レベル「icol」の候補となっている要素の番号が，IW(ic,1)(ic=1~icou)に格納されている。

各要素inC=IW(ic,1)(ic=1~icou)に隣接する要素inのうちで，所属レベル「icol」の候補となっているものがある場合，要素inを候補の中からはずす。
（隣接する要素同士は同じレベルには属することができない）

そのような要素に対して：
IW(in,2)=0
とする

どういうことかというと



所属レベル⑤は②と隣接しているため，②と同じレベルに属することはできない：

```
IW(2,2)= -icol
IW(4,2)= -icol
IW(5,2)= 0
```

所属レベル「icol」の候補となっている要素の番号が，IW(ic,1)(ic=1~icou)に格納されている。

各要素inC=IW(ic,1)(ic=1~icou)に隣接する要素inのうちで，所属レベル「icol」の候補となっているものがある場合，要素inを候補の中からはずす。
（隣接する要素同士は同じレベルには属することができない）

そのような要素に対して：
IW(in,2)=0
とする

```
!C
!C +---------+
!C | COLORING |
!C +---------+
!C===
...

      do ic= 1, icou
        inC= IW(ic,1)
        if (IW(inC,2).ne.0) then
          do k= 1, INL(inC)
            in= IAL(k, inC)
            if (IW(in,2).le.0) IW(in,2)= 0
          enddo
          do k= 1, INU(inC)
            in= IAU(k, inC)
            if (IW(in,2).le.0) IW(in,2)= 0
          enddo
        endif
      enddo

      do ic= 1, icou
        inC= IW(ic,1)
        if (IW(inC,2).ne.0) then
          icouG      = icouG + 1
          IW(inC,2)= icol
        endif
      enddo

      if (icouG.eq.N) exit
    enddo
!C===
```

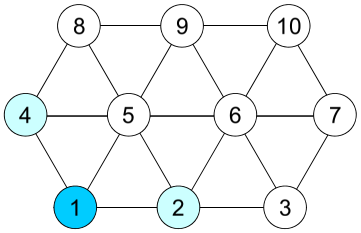
cm(4/5)

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

IW(inC,2)=-icolの要素inCが最終的にレベル番号「icol」に所属する。

このような要素に対して：
IW(inC,2)= icol
とする。

レベル番号が決定した要素数のカウンタ「icouG」を1つ増やす。



所属レベル⑤は②と隣接しているため，②と同じレベルに属することはできない：

```
IW(2,2)= -icol
IW(4,2)= -icol
IW(5,2)= 0
```

②と④のレベル番号を「icol」とする：

```
IW(2,2)= icol
IW(4,2)= icol
```

どういうことかというと

IW(inC,2)=-icolの要素inCが最終的にレベル番号「icol」に所属する。

このような要素に対して：
IW(inC,2)= icol
とする。

レベル番号が決定した要素数のカウンタ「icouG」を1つ増やす。


```
!C
!C +---------+
!C | COLORING |
!C +---------+
!C===
...

      do ic= 1, icou
        inC= IW(ic,1)
        if (IW(inC,2).ne.0) then
          do k= 1, INL(inC)
            in= IAL(k, inC)
            if (IW(in,2).le.0) IW(in,2)= 0
          enddo
          do k= 1, INU(inC)
            in= IAU(k, inC)
            if (IW(in,2).le.0) IW(in,2)= 0
          enddo
        endif
      enddo

      do ic= 1, icou
        inC= IW(ic,1)
        if (IW(inC,2).ne.0) then
          icouG      = icouG + 1
          IW(inC,2)= icol
        endif
      enddo

      if (icouG.eq.N) exit
    enddo
!C===
```

cm(4/5)

icouG ： 全体のカウンタ
icou ： レベル内のカウンタ

icouG=Nとなっていたら，全要素の所属レベルが決まったことになるので，終了。

そうでない場合は，レベル数を一つ増やして，探索を継続する。


```
!C
!C +---------------+
!C | FINAL COLORING |
!C +---------------+
!C===
3000 continue
      NCOLORtot= icol
      icoug= 0
      do ic= 1, NCOLORtot
        icou= 0
        do i= 1, N
          if (IW(i,2).eq.ic) then
            icou = icou + 1
            icoug= icoug + 1
            NEWtoOLD(icoug)= i
            OLDtoNEW(i    )= icoug
          endif
        enddo
        COLORindex(ic)= icou
      enddo

      COLORindex(0)= 0
      do ic= 1, NCOLORtot
        COLORindex(ic)= COLORindex(ic-1) + COLORindex(ic)
      enddo
!C===

!C
!C +----------------+
!C | MATRIX transfer |
!C +----------------+
!C===
      ...
!C===
      return
      end
```

cm(5/5)

icouG=Nとなった時点でのレベル数icolを総レベル数 NCOLORtot とする。

各要素所属レベルに従って，レベル番号の少ない方から，要素の再番号付けを行なう。

```
OLDtoNEW(old_ID)= new_ID
NEWtoOLD(new_ID)= old_ID
```

この時点では「COLORindex」には各色の要素数が入っている。

cm(5/5)

```
!C
!C +---------------+
!C | FINAL COLORING |
!C +---------------+
!C===
 3000 continue
      NCOLORtot= icol
      icoug= 0
      do ic= 1, NCOLORtot
        icou= 0
        do i= 1, N
          if (IW(i,2).eq.ic) then
            icou = icou + 1
            icoug= icoug + 1
            NEWtoOLD(icoug)= i
            OLDtoNEW(i    )= icoug
          endif
        enddo
        COLORindex(ic)= icou
      enddo

      COLORindex(0)= 0
      do ic= 1, NCOLORtot
        COLORindex(ic)= COLORindex(ic-1) + COLORindex(ic)
      enddo
!C===

!C
!C +---------------+
!C | MATRIX transfer |
!C +---------------+
!C===
     ...
!C===
      return
      end
```

「COLORindex」を一次元インデックスに置き換える。

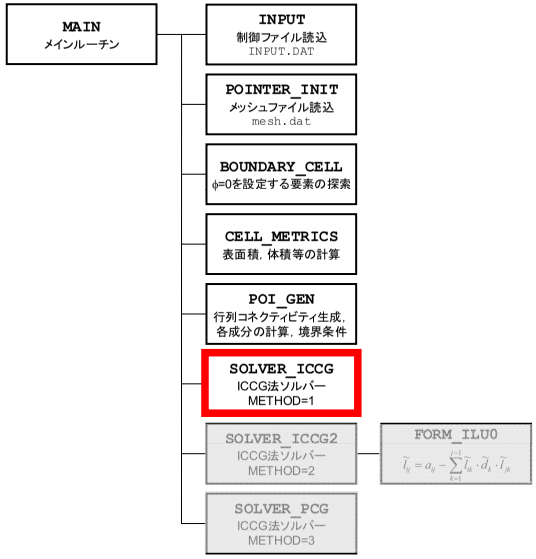
- データ依存性の解決策は?
- オーダリング(Ordering)について
  - Red-Black，Multicolor(MC)
  - Cuthill-McKee(CM)，Reverse-CM(RCM)
  - オーダリングと収束の関係
- オーダリングの実装
- **オーダリング付ICCG法の実装**
- マルチコアへの実装(OpenMP)へ向けて

オーダリング付きICCG法の実装

- 「L2-color」の機能を「L1-sol」へ組み込む

- 「POI_GEN」において，INU,INL,IAL,IAUを求めた時点で，「mc」，「cm」，「rcm」を呼ぶ。
- AL，AUを新しいオーダリングに準じて計算する。
- 境界条件，右辺(体積フラックス項)を新しいオーダリングに準じて計算する。
- ソルバーを呼ぶ。
- 結果(PHI)を古いオーダリングに戻す。
- UCDファイルを書き出す(OUPUT_UCDを呼ぶ)

L1-sol



Minv{r}={z}（1/2）

Forward Substitution $(L)\{z\}=\{r\}$ $(M)\{z\}=(LDL^T)\{z\}=\{r\}$

```
do i= 1, N
  WVAL= R(i)
  do k= indexL(i-1)+1, indexL(i)
    WVAL= WVAL -  MAL(k) * Z(itemL(k))
  enddo
  Z(i)= WVAL / D(i)
enddo
```

Backward Substitution $(DL^T)\{z\}=\{z\}$

```
do i= N, 1, -1
  SW  = 0.0d0
  do k= indexU(i-1)+1, indexU(i)
    SW= SW + MAU(k) * Z(itemU(k))
  enddo
  Z(i)= Z(i) - SW / MD(i)
enddo
```

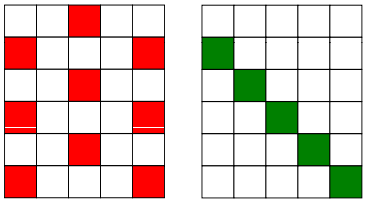
データ依存性あり。ある点におけるZを求めるために，他の点におけるZが右辺に現れる：
⇒ 並列化困難

右辺に自身に依存しないZが現れるようにすればよい
⇒ オーダリング

Minv{r}={z}（2/2）

Forward Substitution $(L)\{z\}=\{r\}$ $(M)\{z\}=(LDL^T)\{z\}=\{r\}$

```
do icol= 1, NCOLORtot
  do i= COLORindex(icol-1)+1, COLORindex(icol)
    WVAL= R(i)
    do k= indexL(i-1)+1, indexL(i)
      WVAL= WVAL -  MAL(k) * Z(itemL(k))
    enddo
    Z(i)= WVAL / D(i)
  enddo
enddo
```



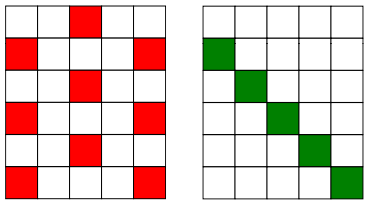
右辺に現れる「Z」は，必ず，現在の「色」（icol）以外の色に属している。
同じ色に属している要素はお互いに依存性，関連性を持たない。

⇒データ依存性回避可能

Minv{r}={z}（2/2）

Forward Substitution $(L)\{z\}=\{r\}$ $(M)\{z\}=(LDL^T)\{z\}=\{r\}$

```
do icol= 1, NCOLORtot
  do i= COLORindex(icol-1)+1, COLORindex(icol)
    WVAL= R(i)
    do k= indexL(i-1)+1, indexL(i)
      WVAL= WVAL -  MAL(k) * Z(itemL(k))
    enddo
    Z(i)= WVAL / D(i)
  enddo
enddo
```



このループ（各色内のループ）は並列，独立に計算可能である。

ファイルのありかと実行方法

```
$> cd <$L2>/solver/run
$> f90 -O3 mg.f -o mg (or cc -O3 mg.c -o mg)
$> ls mg
     mg

$> cd ../src
$> make
$> ls ../run/L2-sol
   L2-sol
```

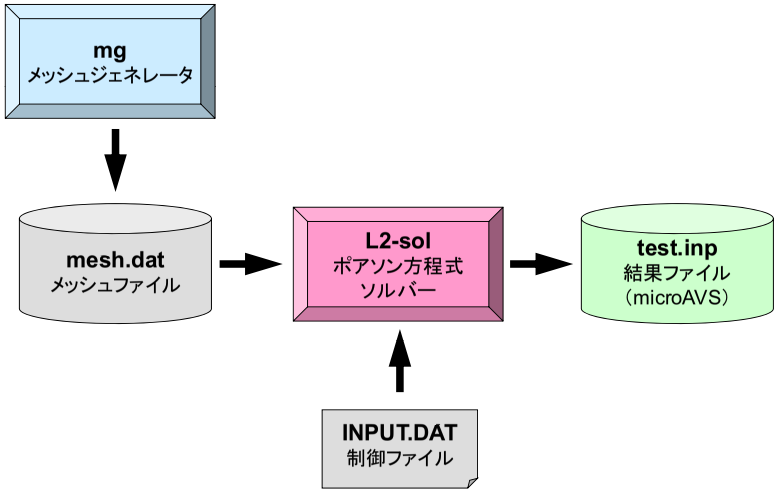
メッシュジェネレータ：**mg**

ポアソン方程式ソルバー：**L2-sol**

プログラムの実行

プログラム，必要なファイル等

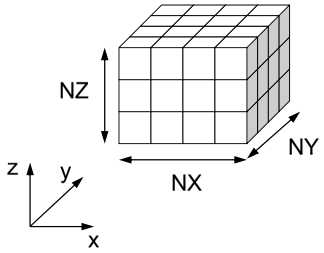
実行ディレクトリ：<$L2>/solver/run



プログラムの実行

メッシュ生成

```
$> cd ../run
$> ./mg
   20 20 20
$> ls mesh.dat
     mesh.dat
```



プログラムの実行

制御データ「<$L2>/run/INPUT.DAT」の作成

```
1.00e-00 1.00e-00 1.00e-00       DX/DY/DZ
1.0e-08                          EPSICCG
```

- DX, DY, DZ
  - 各要素のX,Y,Z方向辺長さ
- EPSICCG
  - ICCG法の収束判定値

プログラムの実行

制御データ「<$L2>/solver/run/INPUT.DAT」の作成

```
$ cd <$L2>/solver/run
$ ./L2-sol

You have    8000 elements.
How many colors do you need ?
  #COLOR must be more than 2 and
  #COLOR must not be more than    8000
   CM if #COLOR .eq. 0
  RCM if #COLOR .eq.-1
CMRCM if #COLOR .le.-2

=> XXX

$ ls test.inp
```

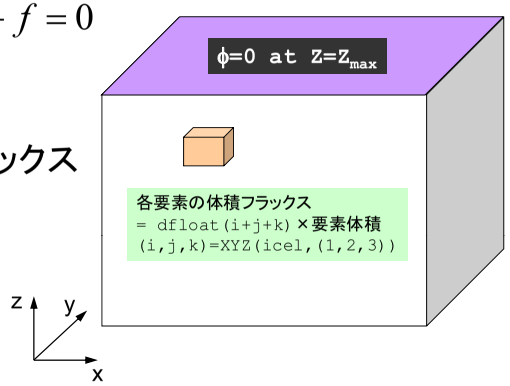
解いている問題：三次元ポアソン方程式

ポアソン方程式

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}+f=0$$

境界条件

- 各要素で体積フラックス
- $Z=Z_{max}$面で$\phi=0$

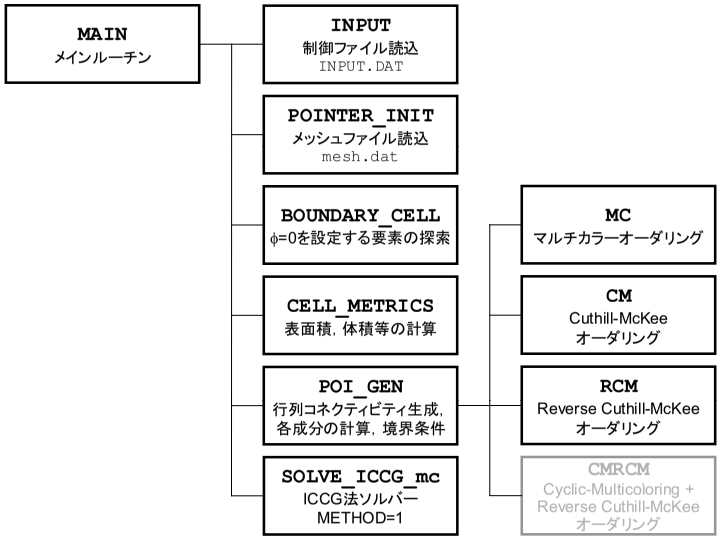


プログラムの構成

```
program MAIN

use STRUCT
use PCG
use solver_ICCG_mc

implicit REAL*8 (A-H,O-Z)
real(kind=8), dimension(:), allocatable :: WK

call INPUT
call POINTER_INIT
call BOUNDARY_CELL
call CELL_METRICS
call POI_GEN

PHI= 0.d0
call solve_ICCG_mc                                              &
&      ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,     &
&        BFORCE,  PHI, AL, AU, NCOLORtot, COLORindex,            &
&        EPSICCG, ITR, IER)

allocate (WK(ICELTOT))

do ic0= 1, ICELTOT
  icel= NEWtoOLD(ic0)
  WK(icel)= PHI(ic0)
enddo

do icel= 1, ICELTOT
  PHI(icel)= WK(icel)
enddo

call OUTUCD

stop
end
```

結果(PHI)をもとの番号付けにもどす

プログラムの構成図



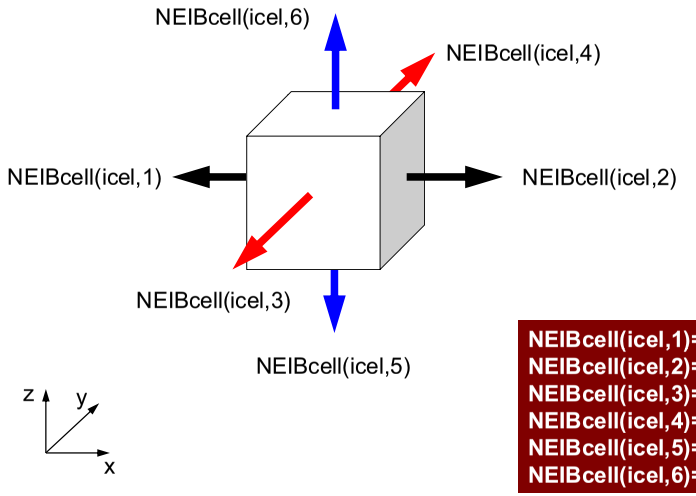
変数表(1/2)

配列・変数名	型	内　容
D(N)	R	対角成分，(N:全メッシュ数)
BFORCE(N)	R	右辺ベクトル
PHI(N)	R	未知数ベクトル
indexL(0:N)	I	各行の非零下三角成分数(CRS)
indexU(0:N)	I	各行の非零上三角成分数(CRS)
NPL	I	非零下三角成分総数(CRS)
NPU	I	非零上三角成分総数(CRS)
itemL(NPL)	I	非零下三角成分(列番号)(CRS)
itemU(NPU)	I	非零下三角成分(列番号)(CRS)
AL(NPL)	R	非零下三角成分(係数)(CRS)
AU(NPL)	R	非零上三角成分(係数)(CRS)
NL,NU	I	各行の非零上下三角成分の最大数（ここでは6)
INL(N)	I	各行の非零下三角成分数
INU(N)	I	各行の非零上三角成分数
IAL(NL,N)	I	各行の非零下三角成分に対応する列番号
IAU(NU,N)	I	各行の非零上三角成分に対応する列番号

変数表(2/2)

配列・変数名	型	内 容
NCOLORtot	**I**	入力時にはOrdering手法（>2：MC，=0：CM，=-1：RCM,<-1：CMRCM），最終的には色数，レベル数が入る
COLORindex (0:NCOLORtot)	**I**	各色，レベルに含まれる要素数の一次元圧縮配列，COLORindex(icol-1)+1からCOLORindex(icol)までの要素がicol番目の色（レベル）に含まれる。
NEWtoOLD(N)	**I**	新番号⇒旧番号への参照配列
OLDtoNEW(N)	**I**	旧番号⇒新番号への参照配列

NEIBcell：隣接している要素番号 境界面の場合は0



NEIBcell(icel,1)= icel – 1
NEIBcell(icel,2)= icel + 1
NEIBcell(icel,3)= icel – NX
NEIBcell(icel,4)= icel + NX
NEIBcell(icel,5)= icel – NX*NY
NEIBcell(icel,6)= icel + NX*NY

プログラムの構成

```
      program MAIN

      use STRUCT
      use PCG
      use solver_ICCG_mc

      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:), allocatable :: WK

      call INPUT
      call POINTER_INIT
      call BOUNDARY_CELL
      call CELL_METRICS
      call POI_GEN

      PHI= 0.d0
      call solve_ICCG_mc                                          &
     &     ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,  &
     &       BFORCE,  PHI, AL, AU, NCOLORtot, COLORindex,         &
     &       EPSICCG, ITR, IER)

      allocate (WK(ICELTOT))
      do ic0= 1, ICELTOT
        icel= NEWtoOLD(ic0)
        WK(icel)= PHI(ic0)
      enddo

      do icel= 1, ICELTOT
        PHI(icel)= WK(icel)
      enddo

      call OUTUCD

      stop
      end
```

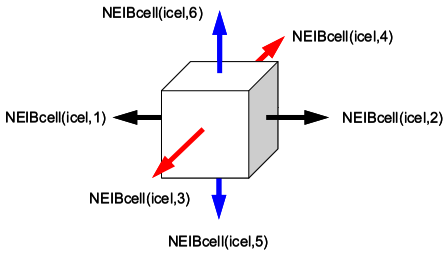

poi_gen(1/8)

```
      subroutine POI_GEN

      use STRUCT
      use PCG

      implicit REAL*8 (A-H,O-Z)

!C
!C-- INIT.
      nn = ICELTOT

      NU= 6
      NL= 6

      allocate (BFORCE(nn), D(nn), PHI(nn))
      allocate (INL(nn), INU(nn), IAL(NL,nn), IAU(NU,nn))

         PHI= 0.d0
           D= 0.d0
      BFORCE= 0.d0

      INL= 0
      INU= 0
      IAL= 0
      IAU= 0
```

配列の宣言

```
!C
!C +-------------+
!C | CONNECTIVITY |
!C +-------------+
!C==
      do icel= 1, ICELTOT
        icN1= NEIBcell(icel,1)
        icN2= NEIBcell(icel,2)
        icN3= NEIBcell(icel,3)
        icN4= NEIBcell(icel,4)
        icN5= NEIBcell(icel,5)
        icN6= NEIBcell(icel,6)

        icouG= 0
        if (icN5.ne.0.and.icN5.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN5
          INL(     icel)= icou
        endif

        if (icN3.ne.0.and.icN3.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN3
          INL(     icel)= icou
        endif

        if (icN1.ne.0.and.icN1.le.ICELTOT) then
          icou= INL(icel) + 1
          IAL(icou,icel)= icN1
          INL(     icel)= icou
        endif
```

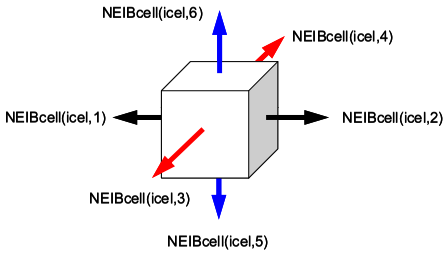
poi_gen(2/8)



下三角成分
NEIBcell(icel,5)= icel – NX*NY
NEIBcell(icel,3)= icel – NX
NEIBcell(icel,1)= icel – 1


```
!C
!C +-------------+
!C | CONNECTIVITY |
!C +-------------+
!C==
      do icel= 1, ICELTOT
        icN1= NEIBcell(icel,1)
        icN2= NEIBcell(icel,2)
        icN3= NEIBcell(icel,3)
        icN4= NEIBcell(icel,4)
        icN5= NEIBcell(icel,5)
        icN6= NEIBcell(icel,6)

        icouG= 0
...

        if (icN2.ne.0.and.icN2.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN2
          INU(     icel)= icou
        endif

        if (icN4.ne.0.and.icN4.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN4
          INU(     icel)= icou
        endif

        if (icN6.ne.0.and.icN6.le.ICELTOT) then
          icou= INU(icel) + 1
          IAU(icou,icel)= icN6
          INU(     icel)= icou
        endif
      enddo
!C==
```

poi_gen(3/8)



上三角成分
NEIBcell(icel,2)= icel + 1
NEIBcell(icel,4)= icel + NX
NEIBcell(icel,6)= icel + NX*NY

poi_gen (4/8)

```
!C
!C +-------------+
!C | MULTICOLORING |
!C +-------------+
!C==
      allocate (OLDtoNEW(ICELTOT), NEWtoOLD(ICELTOT))
      allocate (COLORindex(0:ICELTOT))

 111    continue
        write (*,'(//a,i8,a)') 'You have', ICELTOT, ' elements.'
        write (*,'(  a     )') 'How many colors do you need ?'
        write (*,'(  a     )') '   #COLOR must be more than 2 and'
        write (*,'(  a,i8  )') '   #COLOR must not be more than', ICELTOT
        write (*,'(  a     )') '    CM if #COLOR .eq. 0'
        write (*,'(  a     )') '    RCM if #COLOR .eq.-1'
        write (*,'(  a     )') 'CMRCM if #COLOR .le.-2'
        write (*,'(  a     )') '=>'
        read  (*,*)             NCOLORtot
        if (NCOLORtot.eq.1.or.NCOLORtot.gt.ICELTOT) goto 111

      if (NCOLORtot.gt.0) then
        call MC  (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &            NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.eq.0) then
        call  CM (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &            NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.eq.-1) then
        call RCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,                &
     &            NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif
      if (NCOLORtot.lt.-1) then
        call CMRCM (ICELTOT, NL, NU, INL, IAL, INU, IAU,              &
     &            NCOLORtot, COLORindex, NEWtoOLD, OLDtoNEW)
      endif

      write (*,'(//a,i8,//)') '### FINAL COLOR NUMBER', NCOLORtot
!C==
```



```
!C
!C-- 1D array

      allocate (indexL(0:nn), indexU(0:nn))
      indexL= 0
      indexU= 0

      do icel= 1, ICELTOT
        indexL(icel)= INL(icel)
        indexU(icel)= INU(icel)
      enddo

      do icel= 1, ICELTOT
        indexL(icel)= indexL(icel) + indexL(icel-1)
        indexU(icel)= indexU(icel) + indexU(icel-1)
      enddo

      NPL= indexL(ICELTOT)
      NPU= indexU(ICELTOT)

      allocate (itemL(NPL), AL(NPL))
      allocate (itemU(NPU), AU(NPU))

      itemL= 0
      itemU= 0

        AL= 0.d0
        AU= 0.d0
!C==
```

poi_gen(5/8)

配列の宣言

配列・変数名	型	内 容
D(N)	R	対角成分．(N:全メッシュ数)
BFORCE(N)	R	右辺ベクトル
PHI(N)	R	未知数ベクトル
indexL(0:N)	I	各行の非零下三角成分数(CRS)
indexU(0:N)	I	各行の非零上三角成分数(CRS)
NPL	I	非零下三角成分総数(CRS)
NPU	I	非零上三角成分総数(CRS)
itemL(NPL)	I	非零下三角成分(列番号)(CRS)
itemU(NPU)	I	非零下三角成分(列番号)(CRS)
AL(NPL)	R	非零下三角成分(係数)(CRS)
AU(NPL)	R	非零上三角成分(係数)(CRS)

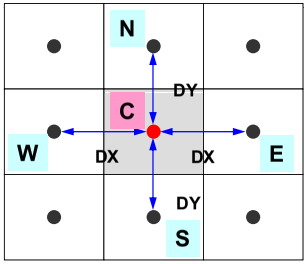
```
do i= 1, N

  VAL= D(i)*p(i)

  do k= indexL(i-1)+1, indexL(i)
    VAL= VAL + AL(k)*p(itemL(k))
  enddo

  do k= indexU(i-1)+1, indexU(i)
    VAL= VAL + AU(k)*p(itemU(k))
  enddo

  q(i)= VAL

enddo
```

```
!C
!C +-------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-------------------------------+
!C===
      icouG= 0
      do icol= 1, NCOLORtot
      do icel= COLORindex(icol-1)+1, COLORindex(icol)
        ic0 = NEWtoOLD(icel)
        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)
        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                  AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
          else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                  AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

poi_gen(6/8)

これ以降は新しい番号付けを使用

係数の計算(境界面以外)



$$\frac{\phi_E-\phi_i}{\Delta x}\Delta y+\frac{\phi_W-\phi_i}{\Delta x}\Delta y+$$
$$\frac{\phi_N-\phi_i}{\Delta y}\Delta x+\frac{\phi_S-\phi_i}{\Delta y}\Delta x=f_c\Delta x\Delta y$$

「新しい番号付け」とは?

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

→

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

- RCMまたはマルチカラーによるオーダリングを施す。
- **色番号順**に要素番号が並んでいる。上記の例では:
  - 第1色(濃い青):1,2,3,4,5(旧:1,3,6,8,9)
  - 第2色(薄い青):6,7,8,9,10(旧:2,4,5,7,10)
  - 第3色(黄緑) :11,12,13(旧:11,13,16)
  - 第4色(黄) :14,15(旧:12,14), 第5色(赤):16(旧:15)

「新しい番号付け」とは?(続き)

13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

→

12	15	16	13
5	10	11	14
8	3	9	4
1	6	2	7

```
NCOLORtot= 5
COLORindex(0)=  0, COLORindex(1)=  5, COLORindex(2)= 10
COLORindex(3)= 13, COLORindex(4)= 15, COLORindex(5)= 16
```

- NEWtoOLD, OLDtoNEW という配列
- `OLDtoNEW(6)=3, NEWtoOLD(3)= 6`


```
!C
!C +-------------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-------------------------------+
!C===
      icouG= 0
      do icol= 1, NCOLORtot
      do icel= COLORindex(icol-1)+1, COLORindex(icol)
        ic0 = NEWtoOLD(icel)
        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)
        VOL0= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                  AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
          else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                  AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

poi_gen(6/8)

これ以降は新しい番号付けを使用

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$
$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$
$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$
$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$
$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$
$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y=f_{icel}\Delta x\Delta y\Delta z$$

```
!C
!C +-----------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-----------------------------+
!C===
      icouG= 0
      do icol= 1, NCOLORtot
      do icel= COLORindex(icol-1)+1, COLORindex(icol)
        ic0 = NEWtoOLD(icel)
        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)
        VOLO= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                  AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
          else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                  AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

icN5がicelより小さければ下三角成分

poi_gen（6/8）

これ以降は新しい番号付けを使用

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$

$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y=f_{icel}\Delta x\Delta y\Delta z$$


```
!C
!C +-----------------------------+
!C | INTERIOR & NEUMANN BOUNDARY CELLs |
!C +-----------------------------+
!C===
      icouG= 0
      do icol= 1, NCOLORtot
      do icel= COLORindex(icol-1)+1, COLORindex(icol)
        ic0 = NEWtoOLD(icel)
        icN1= NEIBcell(ic0,1)
        icN2= NEIBcell(ic0,2)
        icN3= NEIBcell(ic0,3)
        icN4= NEIBcell(ic0,4)
        icN5= NEIBcell(ic0,5)
        icN6= NEIBcell(ic0,6)
        VOLO= VOLCEL  (ic0)

        if (icN5.ne.0) then
          icN5= OLDtoNEW(icN5)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN5.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN5) then
                itemL(j+indexL(icel-1))= icN5
                  AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
          else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN5) then
                itemU(j+indexU(icel-1))= icN5
                  AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif
```

icN5がicelより大きければ上三角成分

poi_gen（6/8）

これ以降は新しい番号付けを使用

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$

$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y=f_{icel}\Delta x\Delta y\Delta z$$


```
        if (icN6.ne.0) then
          icN6= OLDtoNEW(icN6)
          coef= RDZ * ZAREA
          D(icel)= D(icel) - coef

          if (icN6.lt.icel) then
            do j= 1, INL(icel)
              if (IAL(j,icel).eq.icN6) then
                itemL(j+indexL(icel-1))= icN6
                  AL(j+indexL(icel-1))= coef
                exit
              endif
            enddo
          else
            do j= 1, INU(icel)
              if (IAU(j,icel).eq.icN6) then
                itemU(j+indexU(icel-1))= icN6
                  AU(j+indexU(icel-1))= coef
                exit
              endif
            enddo
          endif
        endif

        ii= XYZ(ic0,1)
        jj= XYZ(ic0,2)
        kk= XYZ(ic0,3)

        BFORCE(icel)= -dfloat(ii+jj+kk) * VOLO

      enddo
      enddo
!C===
```

もとの座標に従ってBFORCE計算

poi_gen（7/8）

係数の計算（境界面以外）

$$\frac{\phi_{neib(icel,1)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,2)}-\phi_{icel}}{\Delta x}\Delta y\Delta z+$$

$$\frac{\phi_{neib(icel,3)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

$$\frac{\phi_{neib(icel,4)}-\phi_{icel}}{\Delta y}\Delta z\Delta x+$$

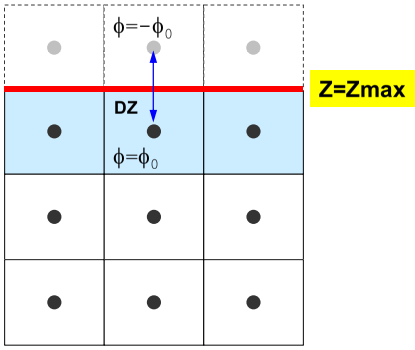
$$\frac{\phi_{neib(icel,5)}-\phi_{icel}}{\Delta z}\Delta x\Delta y+$$

$$\frac{\phi_{neib(icel,6)}-\phi_{icel}}{\Delta z}\Delta x\Delta y=f_{icel}\Delta x\Delta y\Delta z$$


```
!C
!C +-----------------------+
!C | DIRICHLET BOUNDARY CELLs |
!C +-----------------------+
!C   TOP SURFACE
!C===
      do ib= 1, ZmaxCELtot
        ic0= ZmaxCEL(ib)
        coef= 2.d0 * RDZ * ZAREA
        icel= OLDtoNEW(ic0)
        D(icel)= D(icel) - coef
      enddo
!C===

      return
      end
```

poi_gen（8/8）

係数の計算（境界面以外）
係数の計算（境界面）



境界面の外側に，大きさが同じで，値が$\phi=-\phi_0$となるような要素があると仮定（境界面で丁度$\phi=0$となる）。

プログラムの構成

```
      program MAIN

      use STRUCT
      use PCG
      use solver_ICCG_mc

      implicit REAL*8 (A-H,O-Z)
      real(kind=8), dimension(:), allocatable :: WK

      call INPUT
      call POINTER_INIT
      call BOUNDARY_CELL
      call CELL_METRICS
      call POI_GEN

      PHI= 0.d0
      call solve_ICCG_mc                                        &
     &     ( ICELTOT, NPL, NPU, indexL, itemL, indexU, itemU, D,      &
     &       BFORCE, PHI, AL, AU, NCOLORtot, COLORindex,              &
     &       EPSICCG, ITR, IER)
      allocate (WK(ICELTOT))

      do ic0= 1, ICELTOT
        icel= NEWtoOLD(ic0)
        WK(icel)= PHI(ic0)
      enddo

      do icel= 1, ICELTOT
        PHI(icel)= WK(icel)
      enddo

      call OUTUCD

      stop
      end
```

この時点で，係数，右辺ベクトルともに，「新しい」番号にしたがって計算，記憶されている。

solve_ICCG_mc（1/7）

```
!C***
!C*** module solver_ICCG_mc
!C***
!
      module solver_ICCG_mc
      contains
!C
!C*** solve_ICCG
!C
 subroutine solve_ICCG_mc                                          &
     &         ( N, NPL, NPU, indexL, itemL, indexU, itemU, D, B, X,    &
     &           AL, AU, NCOLORtot, COLORindex, EPS, ITR, IER)

      implicit REAL*8 (A-H,O-Z)

      integer :: N, NL, NU, NCOLOR

      real(kind=8), dimension(N)   :: D
      real(kind=8), dimension(N)   :: B
      real(kind=8), dimension(N)   :: X
      real(kind=8), dimension(NPL) :: AL
      real(kind=8), dimension(NPU) :: AU

      integer, dimension(0:N)   :: indexL, indexU
      integer, dimension(NPL):: itemL
      integer, dimension(NPU):: itemU

      integer, dimension(0:NCOLORtot) :: COLORindex

      real(kind=8), dimension(:,:), allocatable :: W

      integer, parameter ::  R= 1
      integer, parameter ::  Z= 2
      integer, parameter ::  Q= 2
      integer, parameter ::  P= 3
      integer, parameter :: DD= 4
```


solve_ICCG_mc（2/7）

```
!C
!C +-----+
!C | INIT |
!C +-----+
!C===
      allocate (W(N,4))

      do i= 1, N
        X(i)  = 0.d0
        W(i,2)= 0.0D0
        W(i,3)= 0.0D0
        W(i,4)= 0.0D0
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          VAL= D(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL - (AL(k)**2) * W(itemL(k),DD)
          enddo
          W(i,DD)= 1.d0/VAL
        enddo
      enddo
!C===
```

不完全修正
コレスキー分解

Compute $r^{(0)}= b-[A]x^{(0)}$

for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)} z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

solve_ICCG_mc（2/7）

```
!C
!C +-----+
!C | INIT |
!C +-----+
!C===
      allocate (W(N,4))

      do i= 1, N
        X(i)  = 0.d0
        W(i,2)= 0.0D0
        W(i,3)= 0.0D0
        W(i,4)= 0.0D0
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          VAL= D(i)
          do k= indexL(i-1)+1, indexL(i)
            VAL= VAL - (AL(k)**2) * W(itemL(k),DD)
          enddo
          W(i,DD)= 1.d0/VAL
        enddo
      enddo
!C===
```

不完全修正
コレスキー分解

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^{\ 2} \cdot d_k \right)^{-1} = l_{ii}^{-1}$$

$$d_i = \left(a_{ii} - \sum_{k=1}^{i-1} a_{ik}^{\ 2} \cdot d_k \right)^{-1} = l_{ii}^{-1}$$

W(i,DD):	d_i
D(i):	a_{ii}
IAL(j,i):	k
AL(j,i):	a_{ik}

不完全修正コレスキー分解

```
do i= 1, N
  VAL= D(i)
  do k= indexL(i-1)+1, indexL(i)
    VAL= VAL - (AL(k)**2) * W(itemL(k),DD)
  enddo
  W(i,DD)= 1.d0/VAL
enddo
```

```
do ic= 1, NCOLORtot
  do i= COLORindex(ic-1)+1, COLORindex(ic)
    VAL= D(i)
    do k= indexL(i-1)+1, indexL(i)
      VAL= VAL - (AL(k)**2) * W(itemL(k),DD)
    enddo
    W(i,DD)= 1.d0/VAL
  enddo
enddo
```

右辺に現れる要素「itemL(k)」は
要素「i」とは異なる「色」に
属している ⇒ データ依存性排除

同じ「色」に属する要素はお互いに
依存性を持たない（接続しない）

solve_ICCG_mc（3/7）

```
!C
!C +-------------------+
!C | {r0}= {b} - [A]{xini} |
!C +-------------------+
!C===
      do i= 1, N
        VAL= D(i)*X(i)
        do k= indexL(i-1)+1, indexL(i)
          VAL= VAL + AL(k)*X(itemL(k))
        enddo
        do k= indexU(i-1)+1, indexU(i)
          VAL= VAL + AU(k)*X(itemU(k))
        enddo
        W(i,R)= B(i) - VAL
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2 + B(i)  **2
      enddo
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

solve_ICCG_mc（4/7）

```
!C
!C********************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +-------------+
!C | {z}= [Minv]{r} |
!C +-------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
      enddo

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i-1)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

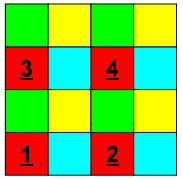
solve_ICCG_mc（4/7）

```
!C
!C********************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +-------------+
!C | {z}= [Minv]{r} |
!C +-------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
      enddo

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C===
```

$(M)\{z\}=(LDL^T)\{z\}=\{r\}$

$(L)\{z\}=\{r\}$

前進代入
Forward Substitution



ic=1

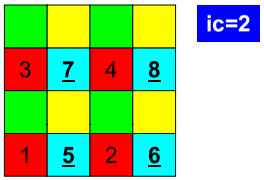
solve_ICCG_mc(4/7)

```
!C
!C*************************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +--------------+
!C | {z}= [Minv]{r} |
!C +--------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
      enddo

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C===
```

$$(M)\{z\}=(LDL^T)\{z\}=\{r\}$$

$$(L)\{z\}=\{r\}$$

前進代入
Forward Substitution



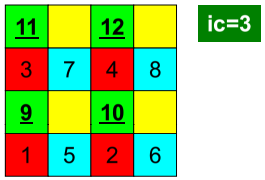
solve_ICCG_mc(4/7)

```
!C
!C*************************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +--------------+
!C | {z}= [Minv]{r} |
!C +--------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
      enddo

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C===
```

$$(M)\{z\}=(LDL^T)\{z\}=\{r\}$$

$$(L)\{z\}=\{r\}$$

前進代入
Forward Substitution



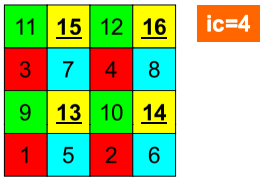
solve_ICCG_mc(4/7)

```
!C
!C*************************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +--------------+
!C | {z}= [Minv]{r} |
!C +--------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
      enddo

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C===
```

$$(M)\{z\}=(LDL^T)\{z\}=\{r\}$$

$$(L)\{z\}=\{r\}$$

前進代入
Forward Substitution



solve_ICCG_mc(4/7)

```
!C
!C*************************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +--------------+
!C | {z}= [Minv]{r} |
!C +--------------+
!C===
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

      do ic= 1, NCOLORtot
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          WVAL= W(i,Z)
          do k= indexL(i-1)+1, indexL(i)
            WVAL= WVAL -  AL(k) * W(itemL(k),Z)
          enddo
          W(i,Z)= WVAL * W(i,DD)
        enddo
      enddo

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C===
```

$$(M)\{z\}=(LDL^T)\{z\}=\{r\}$$

$$(L)\{z\}=\{r\}$$

前進代入
Forward Substitution

$$(DL^T)\{z\}=\{z\}$$

後退代入
Backward Substitution

solve_ICCG_mc(4/7)

```
!C
!C*************************************************** ITERATION
      ITR= N

      do L= 1, ITR
!C
!C +-----------------+
!C | {z}= [Minv]{r} |
!C +-----------------+
!C==
      do i= 1, N
        W(i,Z)= W(i,R)
      enddo

おなじ色の中で計算順序を変えても
結果は同じ:
 i= COLOR(ic-1)+1, COLOR(i)
 i= COLOR(i), COLOR(ic-1)+1, -1

既に計算した結果(上三角成分)
のみ使用

      do ic= NCOLORtot, 1, -1
        do i= COLORindex(ic-1)+1, COLORindex(ic)
          SW  = 0.0d0
          do k= indexU(i-1)+1, indexU(i)
            SW= SW + AU(k) * W(itemU(k),Z)
          enddo
          W(i,Z)= W(i,Z) - W(i,DD) * SW
        enddo
      enddo
!C==
```

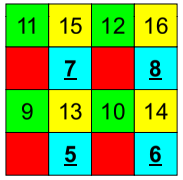
$(M)\{z\}=(LDL^T)\{z\}=\{r\}$

$(L)\{z\}=\{r\}$

$(DL^T)\{z\}=\{z\}$

前進代入
Forward Substitution

後退代入
Backward Substitution



前進後退代入

```
do i= 1, N
  WVAL= W(i,Z)
  do k= indexL(i-1)+1, indexL(i)
    WVAL= WVAL -  AL(k) * W(itemL(k),Z)
  enddo
  W(i,Z)= WVAL * W(i,DD)
enddo


do i= N, 1, -1
  SW= 0.0d0
  do k= indexU(i-1)+1, indexU(i)
    SW= SW + AU(k) * W(itemU(k),Z)
  enddo
  W(i,Z)= W(i,Z) - W(i,DD) * SW
enddo
```

→

```
do ic= 1, NCOLORtot
  do i= COLORindex(ic-1)+1, COLORindex(ic)
    WVAL= W(i,Z)
    do k= indexL(i-1)+1, indexL(i)
      WVAL= WVAL -  AL(k) * W(itemL(k),Z)
    enddo
    W(i,Z)= WVAL * W(i,DD)
  enddo
enddo

do ic= NCOLORtot, 1, -1
  do i= COLORindex(ic-1)+1, COLORindex(ic)
    SW= 0.0d0
    do k= indexL(i-1)+1, indexL(i)
      SW= SW + AU(k) * W(itemU(k),Z)
    enddo
    W(i,Z)= W(i,Z) - W(i,DD) * SW
  enddo
enddo
```


solve_ICCG_mc(5/7)

```
!C
!C +-----------+
!C | RHO= {r}{z} |
!C +-----------+
!C==
      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo
!C==

!C
!C +-------------------------+
!C | {p} = {z} if      ITER=1    |
!C | BETA= RHO / RHO1  otherwise |
!C +-------------------------+
!C==
      if ( L.eq.1 ) then
        do i= 1, N
          W(i,P)= W(i,Z)
        enddo
      else
        BETA= RHO / RHO1
        do i= 1, N
          W(i,P)= W(i,Z) + BETA*W(i,P)
        enddo
      endif
!C==
```

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
    else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```


solve_ICCG_mc(6/7)

```
!C
!C +-----------+
!C | {q}= [A]{p} |
!C +-----------+
!C==
      do i= 1, N
        VAL= D(i)*W(i,P)
        do k= indexL(i-1)+1, indexL(i)
          VAL= VAL + AL(k)*W(itemL(k),P)
        enddo
        do k= indexU(i-1)+1, indexU(i)
          VAL= VAL + AU(k)*W(itemU(k),P)
        enddo
        W(i,Q)= VAL
      enddo
!C==
```

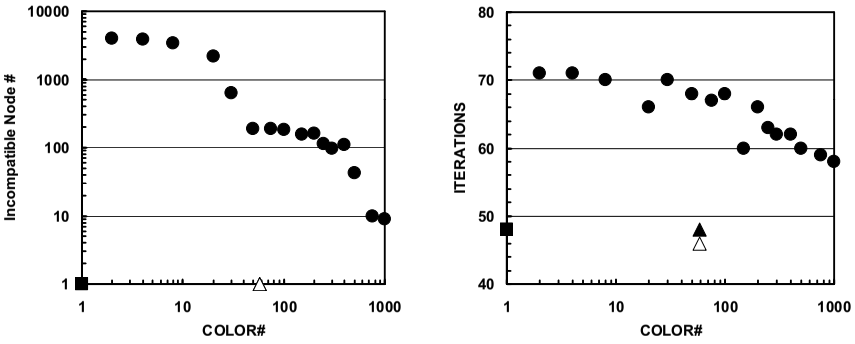
```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
    else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

solve_ICCG_mc(7/7)

```
!C
!C +---------------------+
!C | ALPHA= RHO / {p}{q} |
!C +---------------------+
!C===
      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1
!C===
!C
!C +---------------------+
!C | {x}= {x} + ALPHA*{p} |
!C | {r}= {r} - ALPHA*{q} |
!C +---------------------+
!C===
      do i= 1, N
        X(i)   = X(i)   + ALPHA * W(i,P)
        W(i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
      DNRM2= 0.d0
      do i= 1, N
        DNRM2= DNRM2 + W(i,R)**2
      enddo
!C===
      ERR = dsqrt(DNRM2/BNRM2)
      if (ERR .lt. EPS) then
          IER = 0
          goto 900
        else
          RHO1 = RHO
        endif
      enddo
      IER = 1

  900 continue
```

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```


ICCG法の収束



(20^3=8,000要素, EPSICCG=10^{-8})
(■:ICCG(L1), ●:ICCG-MC, ▲:ICCG-CM, △:ICCG-RCM)

- データ依存性の解決策は?
- オーダリング(Ordering)について
  - Red-Black, Multicolor(MC)
  - Cuthill-McKee(CM), Reverse-CM(RCM)
  - オーダリングと収束の関係
- オーダリングの実装
- オーダリング付ICCG法の実装
- **マルチコアへの実装(OpenMP)へ向けて**
  - **L2-solにOpenMPを適用するだけ・・・**