線形ソルバー
課題S3出題

2007年5月16日

中島研吾

並列計算プログラミング（616-2057）・先端計算機演習I（616-4009）

本日の話題

- 線形ソルバーの概要
  - 直接法
  - 反復法
  - 共役勾配法（Conjugate Gradient）
  - 前処理
- 接触問題の例（前処理）
  - Selective Blocking Preconditioning
- 課題S3

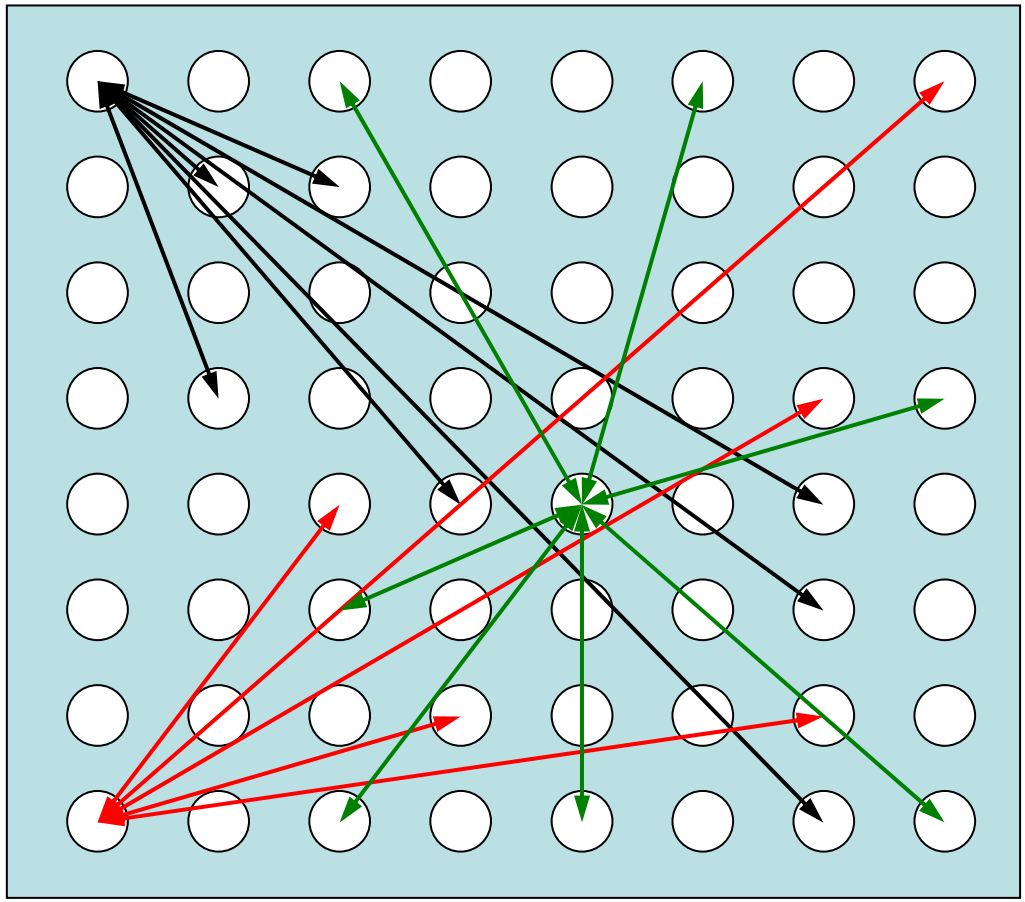
科学技術計算における大規模線形方程式の解法

- 多くの科学技術計算は，最終的に大規模線形方程式Ax=bを解くことに帰着される。
  - important, expensive
- アプリケーションに応じて様々な手法が提案されている
  - 疎行列（sparse），密行列（dense）
  - 直接法（direct），反復法（iterative）
    - 本日は，疎行列，反復法について主に扱う。
- 密行列（dense）
  - グローバルな相互作用あり：BEM，スペクトル法，MO，MD（気液）
- 疎行列（sparse）
  - ローカルな相互作用：FEM，FDM，MD（固），高速多重極展開付BEM

グループ通信，1対1通信（密行列）

遠隔PE（領域）も含め，多数のPE（領域）との相互作用あり

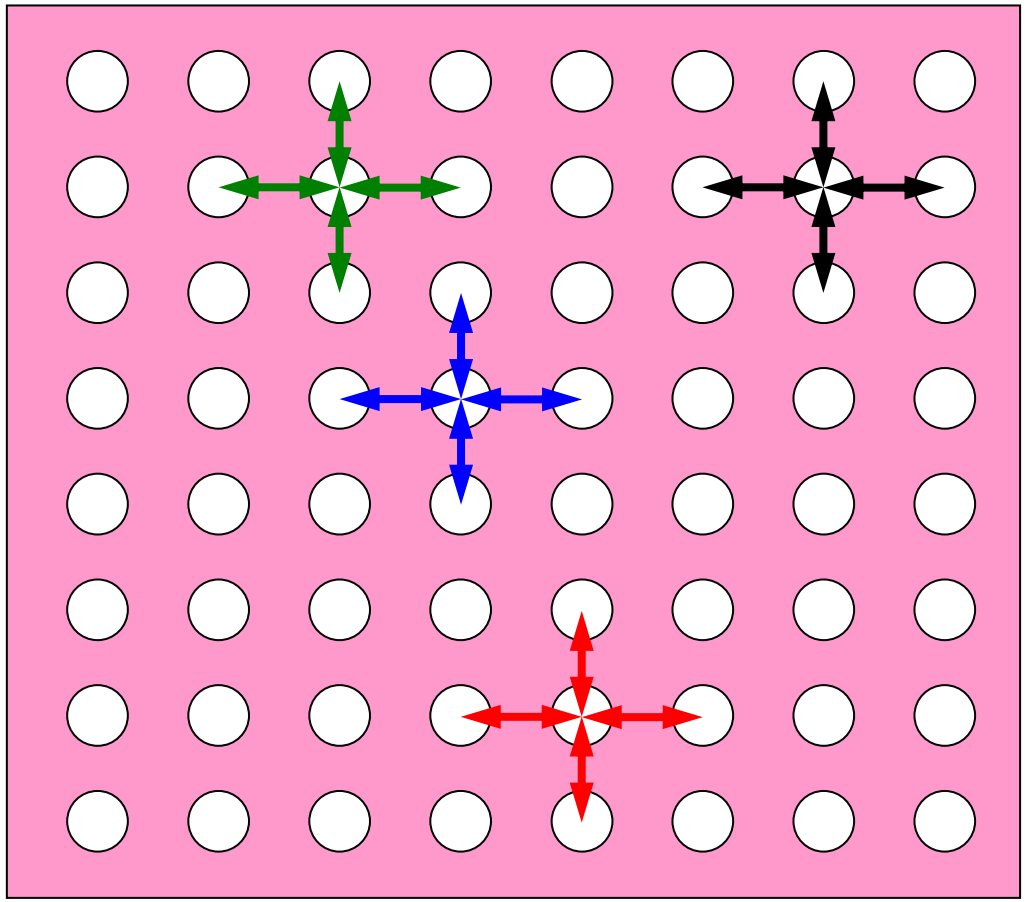
境界要素法，スペクトル法，MD法



グループ通信，1対1通信（疎行列）

近接PE（領域）のみとの相互作用

差分法，有限要素法



「線形ソルバー」にどう向き合うか？

- 一言で，「線形ソルバー」というが，一つの大きな学問体系を構成しており，短時間で学ぶことは不可能である。
  - とは言え，ある程度理解しておく必要はある
- MPIのときと同じであるが，各自の研究に応じた方法を選択する必要がある。
  - 選択ができる程度の知識は必要⇒科学者としてのたしなみ
  - 公開ソフトの利用，改良
  - 前処理付き反復法であればある程度相談に乗れます。
- 自分も実は1995年頃までは，アプリケーションサイドの「一般ユーザー」であった。
  - 何とか，安定に解けないか，速く解けないか，ということをやっているうちに，この分野が専門になってしまった。
  - はまると果てしない。

適切な解法の選択が最も重要

- そのためには，自分の解いている問題の特性（数学的特性，物理的特性）を知ることが重要

- 係数行列の性質
  - 正方行列，MxN行列
  - 疎行列，密行列
  - 対称，非対称
  - 対角成分に0を含む？
  - 対角成分の絶対値は非対角成分と比較して大きい？

公開ソフト

- ACTS（Advanced CompuTational Software）Collection
  - http://acts.nersc.gov/
  - US-DOEのプロジェクトで開発された様々なライブラリ
  - SuperLU，PETSc，Aztec
- HPC-MW
  - http://hpcmw.tokyo.rist.or.jp/
  - 並列反復法ライブラリ

参考書

- J.J.Dongarra et al. “Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods”, SIAM, 1994.（邦訳：長谷川他「反復法Templates」朝倉書店, 1995）

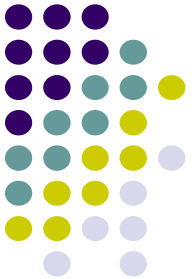
- http://www.netlib.org/templates/index.html
  - template.pdf/ps/html：本そのもの
  - サンプルプログラム等も上記URLから取れる。

直接法（Direct Method）

- Gaussの消去法，完全LU分解
  - 逆行列A^{-1}を直接求める

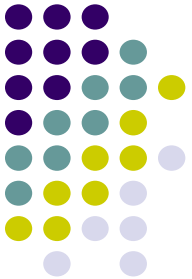
- 利点
  - 安定，幅広いアプリケーションに適用可能
    - Partial Pivoting
  - 疎行列，密行列いずれにも適用可能
- 欠点
  - 反復法よりもメモリ，計算時間を必要とする
    - 密行列の場合，$O(N^3)$の計算量
  - 大規模な計算向けではない
    - $O(N^2)$の記憶容量，$O(N^3)$の計算量

LU分解法：完全LU分解法



- 直接法の一種
  - 逆行列を直接求める手法
  - 「逆行列」に相当するものを保存しておけるので，右辺が変わったときに計算時間を節約できる
  - 逆行列を求める際にFill-in（もとの行列では0であったところに値が入る）が生じる

LU分解による連立一次方程式の解法



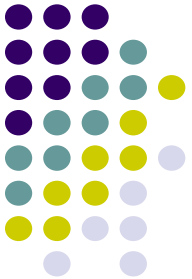
Aがn×n行列のとき、Aを次式のように表すことを
（あるいは、そのようなLとUそのものを）AのLU分解という.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

$$\mathbf{A = LU}$$

L : Lower triangular part of matrix A
U : Upper triangular part of matrix A

連立一次方程式の行列表現



n元の連立一次方程式の一般形

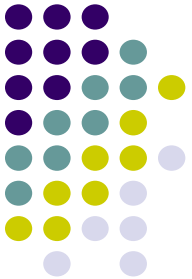
$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
$$

行列表現

$$
\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \Leftrightarrow \mathbf{Ax} = \mathbf{b}
$$

$\mathbf{A}$ $\mathbf{x}$ $\mathbf{b}$

LU分解を用いたAx=bの解法



1 $\mathbf{A = LU}$ となるAのLU分解LとUを求める.

2 $\mathbf{Ly = b}$ の解yを求める.（簡単！）

3 $\mathbf{Ux = y}$ の解xを求める.（簡単！）

このxが $\mathbf{Ax = b}$ の解となる

$$\because \mathbf{Ax = LUx = Ly = b}$$

Ly=bの解法：前進代入

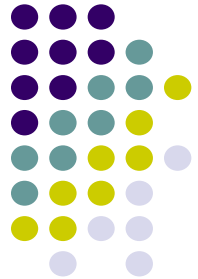
$$\mathbf{Ly} = \mathbf{b} \quad \Longleftrightarrow \quad \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

$$\Longleftrightarrow$$

$$\begin{aligned} & y_1 = b_1 \\ & l_{21} y_1 + y_2 = b_2 \\ & \vdots \\ & l_{n1} y_1 + l_{n2} y_2 + \cdots + y_n = b_n \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} & y_1 = b_1 \\ & y_2 = b_2 - l_{21} y_1 \\ & \vdots \\ & y_n = b_n - l_{n1} y_1 - l_{n2} y_2 = b_n - \sum_{i=1}^{n-1} l_{ni} y_i \end{aligned}$$

芋づる式に（one after another）解が求まる.

Ux=yの解法：後退代入



$$\mathbf{Ux} = \mathbf{y} \quad \Longleftrightarrow \quad \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$\Longleftrightarrow$$

$$u_{nn} x_n = y_n$$

$$u_{n-1,n-1} x_{n-1} + u_{n-1,n} x_n = y_{n-1}$$

$$\vdots$$

$$u_{11} x_1 + u_{12} x_2 + \cdots + u_{1n} x_n = y_1$$

$$\Longleftrightarrow$$

$$x_n = y_n / u_{nn}$$

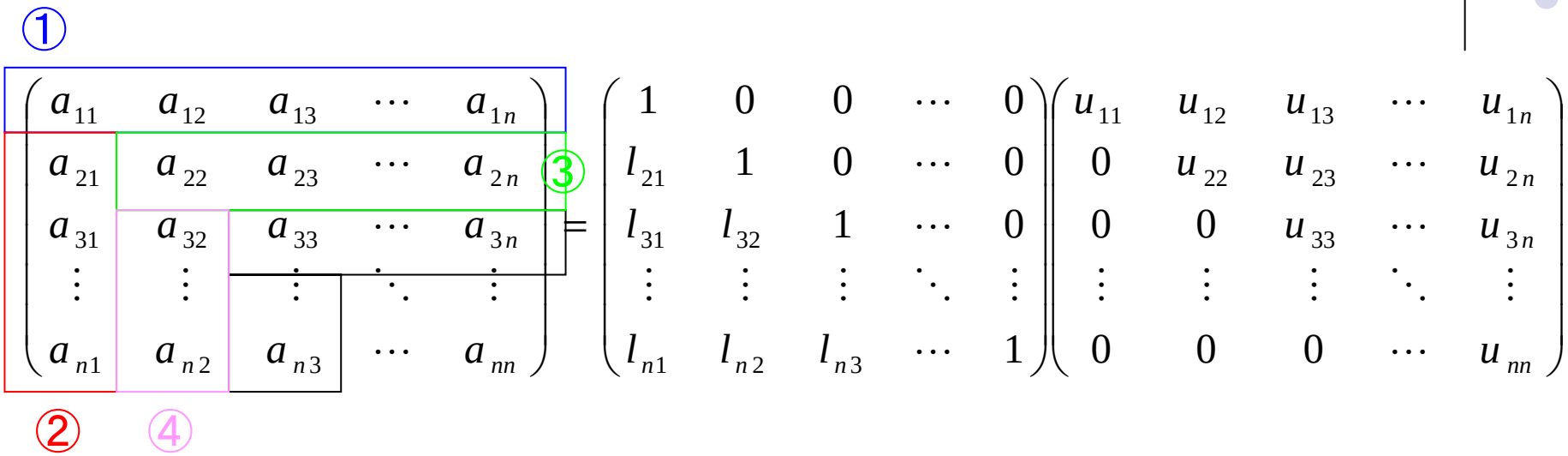
$$x_{n-1} = (y_{n-1} - u_{n-1,n} x_n) / u_{n-1,n-1}$$

$$\vdots$$

$$x_1 = \left(y_1 - \sum_{i=2}^{n} u_{1j} x_j \right) / u_{11}$$

芋づる式に（one after another）解が求まる.

LU分解の求め方



$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix}$$

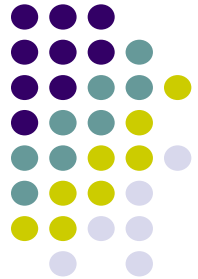
① ➡ $a_{11} = u_{11}, a_{12} = u_{12}, \cdots, a_{1n} = u_{1n} \Rightarrow u_{11}, u_{12}, \cdots, u_{1n}$

② ➡ $a_{21} = l_{21}u_{11}, a_{31} = l_{31}u_{11}, \cdots, a_{n1} = l_{n1}u_{11} \Rightarrow l_{21}, l_{31}, \cdots, l_{n1}$

③ ➡ $a_{22} = l_{21}u_{12} + u_{22}, \cdots, a_{2n} = l_{21}u_{1n} + u_{2n} \Rightarrow u_{22}, u_{23}, \cdots, u_{2n}$

④ ➡ $a_{32} = l_{31}u_{12} + l_{32}u_{22}, \cdots \Rightarrow l_{32}, l_{42}, \cdots, l_{n2}$

数値例



$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 10 \\ 2 & 2 & 8 & 7 \\ 0 & -4 & 7 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

第1行 ➡ $1 = u_{11}, 2 = u_{12}, 3 = u_{13}, 4 = u_{14}$

第1列 ➡ $2 = l_{21}u_{11} \Rightarrow l_{21} = 2/u_{11} = 2, \quad 2 = l_{31}u_{11} \Rightarrow l_{31} = 2/u_{11} = 2$

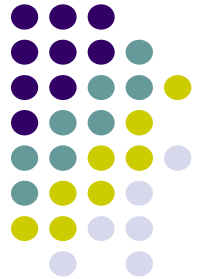
$0 = l_{41}u_{11} \Rightarrow l_{41} = 0/u_{11} = 0$

第2行 ➡ $6 = l_{21}u_{12} + u_{22} \Rightarrow u_{22} = 2, \quad 7 = l_{21}u_{13} + u_{23} \Rightarrow u_{23} = 1$

$10 = l_{21}u_{14} + u_{24} \Rightarrow u_{24} = 2$

第2列 ➡ $2 = l_{31}u_{12} + l_{32}u_{22} \Rightarrow l_{32} = -1, \quad -4 = l_{41}u_{12} + l_{42}u_{22} \Rightarrow l_{42} = -2$

数値例(続き)



$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 10 \\ 2 & 2 & 8 & 7 \\ 0 & -4 & 7 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix}$$

第3行 ➡ $8 = l_{31}u_{13} + l_{32}u_{23} + u_{33} \Rightarrow u_{33} = 3,$

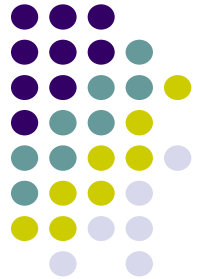
$7 = l_{31}u_{14} + l_{32}u_{24} + u_{34} \Rightarrow u_{34} = 1$

第3列 ➡ $7 = l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} \Rightarrow u_{43} = 3$

第4行(第4列) ➡ $1 = l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{44} \Rightarrow u_{44} = 2$

1行、1列、2行、2列、・・・の順に求める式を作っていく.

数値例(続き)



結局

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 6 & 7 & 10 \\ 2 & 2 & 8 & 7 \\ 0 & -4 & 7 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & -1 & 1 & 0 \\ 0 & -2 & 3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$\mathbf{L}$ $\mathbf{U}$

Partial Pivoting（ピボットの部分選択）

- LU分解を実施する場合，各段階で適用される対角成分A_{kk}をピボット（pivot）という。
- 消去のある段階でピボットが0となると，ゼロ割が生じ，計算続行が不可能となる。
  - ピボットの絶対値が非常に小さい場合も同様

```
do i= 2, n
  do k= 1,  i-1
    a_ik := a_ik/a_kk
    do j= k+1, n
      a_ij := a_ij - a_ik*a_kj
    enddo
  enddo
enddo
```

- ピボットの部分選択
  - 絶対値最大の成分がピボットの位置に来るように行を入れ替える。
  - もとの連立一次方程式におけるK行とL行の入れ替えに相当する。

並列直接法ライブラリ

- ScaLAPACK
  - http://www.netlib.org/scalapack/
  - ACTSの一部でもある。
  - 密行列，一般，幅広い応用分野
    - LAPACKの並列版
  - TOP500 List

- SuperLU
  - http://acts.nersc.gov/superlu/
  - Lawrence Berkeley National Laboratory
  - 疎行列に対応，FORTRAN/Cインタフェース

- いずれも「partial pivoting」に対応

反復法（Iterative Method）

- 定常（stationary）法
  - 反復計算中，解ベクトル以外の変数は変化せず
  - SOR，Gauss-Seidel，Jacobiなど（課題S2）

- 非定常（nonstationary）法
  - 拘束，最適化条件が加わる
  - Krylov部分空間（subspace）への写像を基底として使用するため，Krylov部分空間法とも呼ばれる
  - CG（Conjugate Gradient：共役勾配法）
  - BiCGSTAB（Bi-Conjugate Gradient Stabilized）
  - GMRES（Generalized Minimal Residual）

反復法（Iterative Method）（続き）

- 利点
  - 直接法と比較して，メモリ使用量，計算量が少ない。
  - 並列計算には適している。

- 欠点
  - 収束性が，アプリケーション，境界条件の影響を受けやすい。
  - 前処理（preconditioning）が重要。

並列反復法ライブラリ

- PETSc
  - http://acts.nersc.gov/petsc/
  - Portable, Extensible Toolkit for Scientific Computing
  - MPICHを開発したアルゴンヌのグループ
- Aztec
  - http://acts.nersc.gov/aztec/
  - Sandia National Laboratories
  - PETScより玄人向け，と言われている。
- HPC-MW
  - http://hpcmw.tokyo.rist.or.jp/
- 一般的な前処理手法をサポート

代表的な反復法：共役勾配法

- Conjugate Gradient法，略して「CG」法
  - 最も代表的な「非定常」反復法
- 対称正定値行列（Symmetric Positive Definite：SPD）向け
  - 任意のベクトル$\{x\}$に対して$\{x\}^T[A]\{x\}>0$
  - 全対角成分>0，全固有値>0，全部分行列式>0と同値
- アルゴリズム
  - 最急降下法（Steepest Descent Method）の変種
  - **$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$**
    - **$x^{(i)}$：反復解，$p^{(i)}$：探索ベクトル，α_i：定数）**
  - 厳密解をyとするとき　$\{x-y\}^T[A]\{x-y\}$を最小とするような$\{x\}$を求める。
  - 詳細は参考文献〔長谷川ら〕参照

共役勾配法のアルゴリズム

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  $z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$:ベクトル
α_i :スカラー

共役勾配法のアルゴリズム(1/5)

yを厳密解($Ay=b$)とするとき，下式を最小にするxを求める:

$$(x-y)^T[A](x-y)$$

$$(x-y)^T[A](x-y)=(x,Ax)-(y,Ax)-(x,Ay)+(y,Ay)$$
$$=(x,Ax)-2(x,Ay)+(y,Ay)=(x,Ax)-2(x,b)+\underline{(y,b)}$$ 定数

従って，下記 $f(x)$ を最小にするxを求めればよい:

$$f(x)=\frac{1}{2}(x,Ax)-(x,b)$$

$$f(x+h)=f(x+h)+(h,Ax-b)+\frac{1}{2}(h,Ah)$$ 任意のベクトルh

共役勾配法のアルゴリズム(2/5)

CG法は任意の $x^{(0)}$ から始めて，$f(x)$の最小値を逐次探索する。
今，k番目の近似値 $x^{(k)}$と探索方向 $p^{(k)}$が決まったとすると:

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$$

$f(x^{(k+1)})$ を最小にするためには:

$$f\left(x^{(k)} + \alpha_k p^{(k)}\right) = \frac{1}{2}\alpha_k{}^2\left(p^{(k)}, Ap^{(k)}\right) - \alpha_k\left(p^{(k)}, b - Ax^{(k)}\right) + f\left(x^{(k)}\right)$$

$$\frac{\partial f\left(x^{(k)} + \alpha_k p^{(k)}\right)}{\partial \alpha_k} = 0 \Rightarrow \alpha_k = \frac{\left(p^{(k)}, b - Ax^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)} = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

$r^{(k)} = b - Ax^{(k)}$ は第k近似に対する残差

共役勾配法のアルゴリズム（3/5）

残差 $r^{(k)}$も以下の式によって計算できる:

$$r^{(k+1)} = r^{(k)} - \alpha_k A p^{(k)}$$

探索方向を以下の漸化式によって求める:

$$p^{(k+1)} = r^{(k+1)} + \beta_k p^{(k)}, r^{(0)} = p^{(0)}$$

本当のところは下記のように（k+1）回目に厳密解 y が求まれば良いのであるが，解がわかっていない場合は困難・・・

$$y = x^{(k+1)} + \alpha_{k+1} p^{(k+1)}$$

共役勾配法のアルゴリズム(4/5)

ところで，下式のような都合の良い直交関係がある:

$$\left(Ap^{(k)}, y - x^{(k+1)}\right) = 0$$

$$\left(Ap^{(k)}, y - x^{(k+1)}\right) = \left(p^{(k)}, Ay - Ax^{(k+1)}\right) = \left(p^{(k)}, b - Ax^{(k+1)}\right)$$
$$= \left(p^{(k)}, b - A\left[x^{(k)} + \alpha_k p^{(k)}\right]\right) = \left(p^{(k)}, b - Ax^{(k)} - \alpha_k Ap^{(k)}\right)$$
$$= \left(p^{(k)}, r^{(k)} - \alpha_k Ap^{(k)}\right) = \left(p^{(k)}, r^{(k)}\right) - \alpha_k \left(p^{(k)}, Ap^{(k)}\right) = 0$$

$$\alpha_k = \frac{\left(p^{(k)}, r^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

従って以下が成立する

$$\left(Ap^{(k)}, y - x^{(k+1)}\right) = \left(Ap^{(k)}, \alpha_{k+1} p^{(k+1)}\right) = 0 \Rightarrow \left(p^{(k+1)}, Ap^{(k)}\right) = 0$$

共役勾配法のアルゴリズム(5/5)

$$\left(p^{(k+1)}, Ap^{(k)}\right) = \left(r^{(k+1)} + \beta_k p^{(k)}, Ap^{(k)}\right) = \left(r^{(k+1)}, Ap^{(k)}\right) + \beta_k \left(p^{(k)}, Ap^{(k)}\right) = 0$$

$$\Rightarrow \beta_k = \frac{-\left(r^{(k+1)}, Ap^{(k)}\right)}{\left(p^{(k)}, Ap^{(k)}\right)}$$

$\left(p^{(k+1)}, Ap^{(k)}\right) = 0$ 勾配ベクトル $p^{(k)}$がAに関して共役である

勾配ベクトル $p^{(k)}$, 残差ベクトル $r^{(k)}$ について，以下の関係が成立

$$\left(p^{(i)}, Ap^{(j)}\right) = 0 \ (i \neq j), \quad \left(r^{(i)}, r^{(j)}\right) = 0 \ (i \neq j)$$

N次元空間で互いに直交で一時独立な残差ベクトル $r^{(k)}$ はN個しか存在しない，従って共役勾配法は未知数がN個のときにN回以内に収束する
⇒ 実際は丸め誤差の影響がある

共役勾配法のアルゴリズム

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

for i= 1, 2, …

$z^{(i-1)} = r^{(i-1)}$

$\rho_{i-1} = r^{(i-1)} z^{(i-1)}$

if i=1

$p^{(1)} = z^{(0)}$

else

$\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$

$p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i)}$

endif

$\mathbf{q^{(i)} = [A]p^{(i)}}$

$\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$

$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$

check convergence |r|

end

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$: ベクトル

α_i : スカラー

共役勾配法のアルゴリズム

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  $z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$: ベクトル
α_i : スカラー

共役勾配法のアルゴリズム

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

- 行列ベクトル積
- ベクトル内積
- ベクトル定数倍の加減

$x^{(i)}$: ベクトル
α_i : スカラー

前処理（preconditioning）とは?

- 反復法の収束は係数行列の固有値分布に依存
  - 固有値分布が少なく，かつ1に近いほど収束が早い（単位行列）
  - 条件数（condition number）（対称正定）＝最大最小固有値の比
    - 条件数が1に近いほど収束しやすい
- もとの係数行列**A**に良く似た前処理行列**M**を適用することによって固有値分布を改善する。
  - 前処理行列 **M** によって元の方程式 **Ax=b** を **A‘x=b’** へと変換する。ここで $\mathbf{A'=M^{-1}A}$, $\mathbf{b'=M^{-1}b}$ である。
  - $\mathbf{A'=M^{-1}A}$が単位行列に近ければ良い，ということになる。
- 「前処理」は密行列，疎行列ともに使用するが，普通は疎行列を対象にすることが多い。

前処理付き共役勾配法のアルゴリズム

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
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i)}$
  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

x

ILU(0), IC(0)

- 最もよく使用されている前処理（疎行列用）
  - 不完全LU分解
    - Incomplete LU Factorization
  - 不完全コレスキー分解
    - Incomplete Cholesky Factorization（対称行列）

- 不完全な直接法
  - もとの行列が疎でも，逆行列は疎とは限らない。
  - fill-in
  - もとの行列と同じ非ゼロパターン（fill-in無し）を持っているのがILU（0），IC（0）

ILU(0), IC(0)

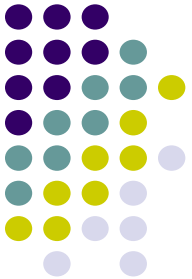
- 最もよく使用されている前処理（疎行列用）
  - Incomplete LU Factorization
  - Incomplete Cholesky Factorization（対称行列）

ILU(0) : keep non-zero pattern of the original coefficient matrix

```
do i= 2, n
  do k= 1,  i-1
    if ((i,k)∈ NonZero(A)) then
      a_ik := a_ik/a_kk
    endif
    do j= k+1, n
      if ((i,j)∈ NonZero(A)) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

- 不完全な直接法
  - もとの行列が疎でも，逆行列は疎とは限らない。
  - fill-in
  - もとの行列と同じ非ゼロパターン（fill-in無し）を持っているのがILU(0)，IC(0)

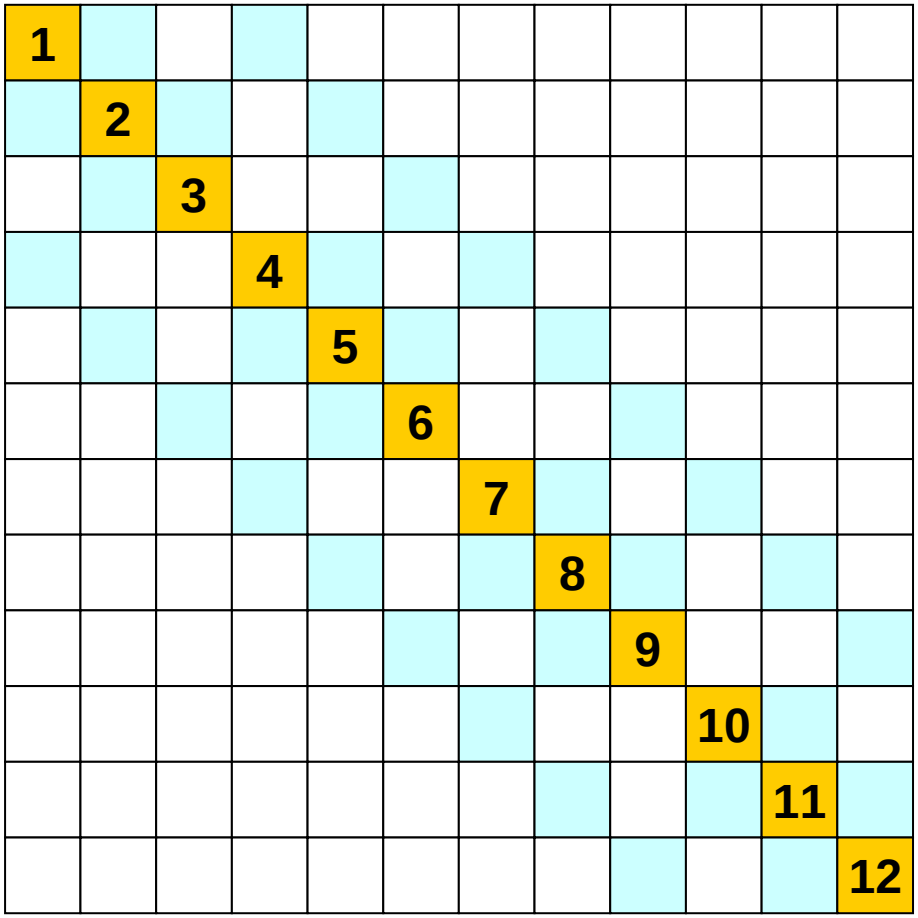
LU分解法：完全LU分解法

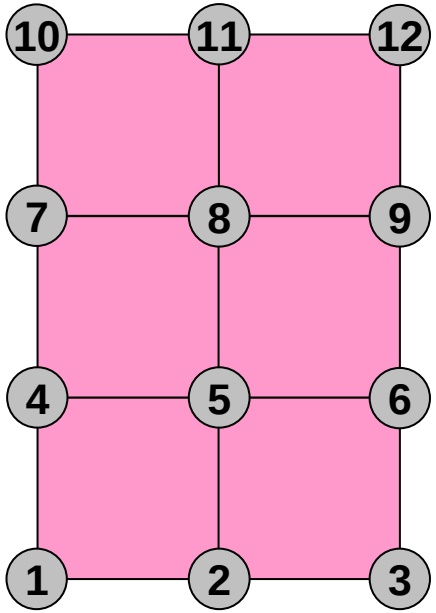


- 直接法の一種
  - 逆行列を直接求める手法
  - 「逆行列」に相当するものを保存しておけるので，右辺が変わったときに計算時間を節約できる
  - 逆行列を求める際にFill-in（もとの行列では0であったところに値が入る）が生じる
- 不完全LU分解法
  - Fill-inの発生を制限して，前処理に使う手法
    - 不完全な逆行列，少し弱い直接法

実例：差分法による熱伝導等

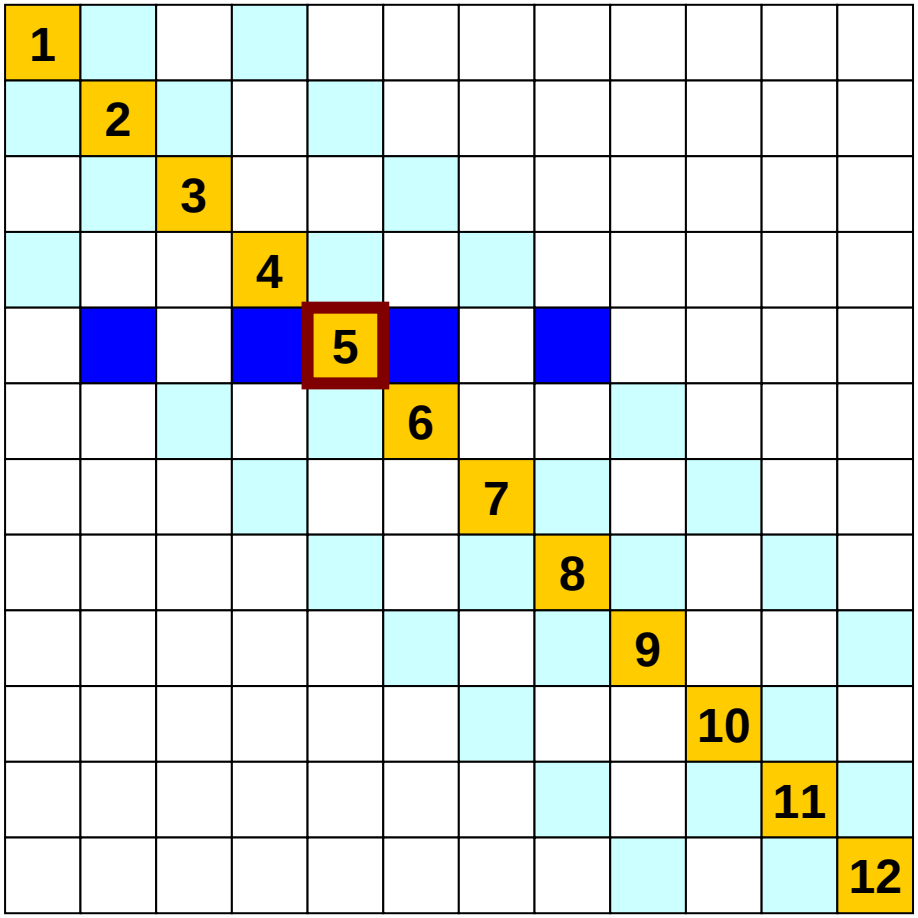
5点差分

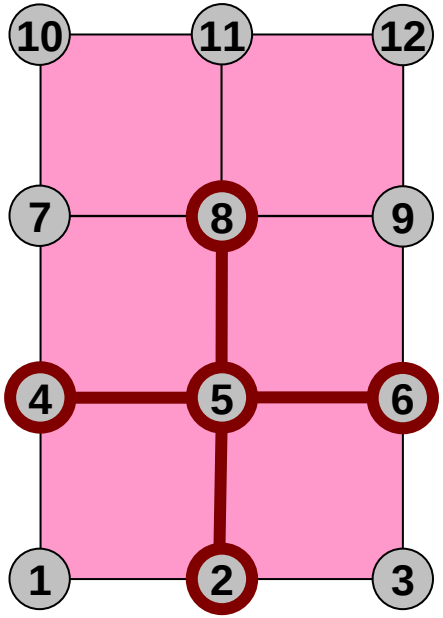




実例：差分法による熱伝導等

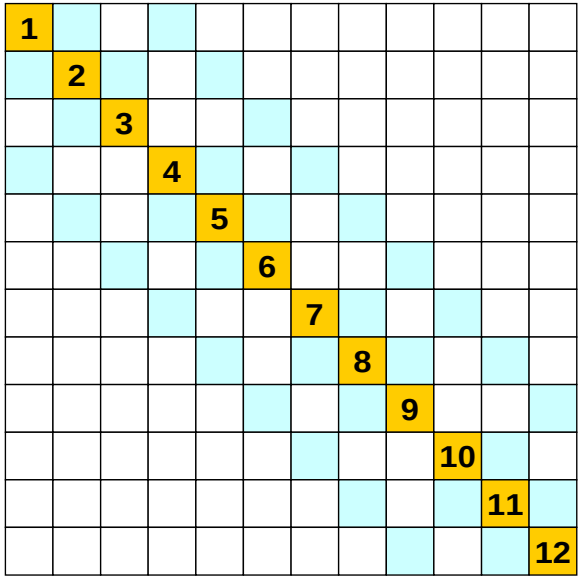
5点差分

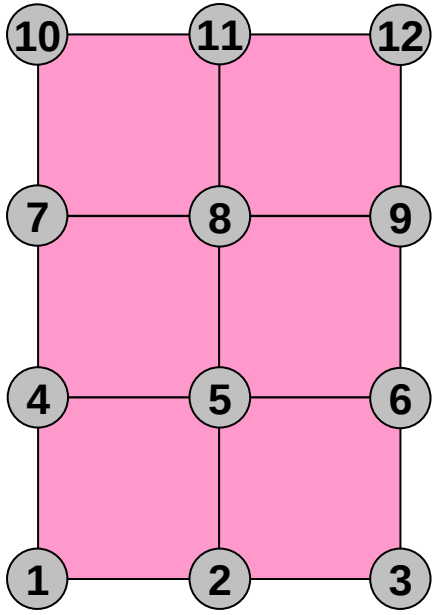




実例：係数マトリクス

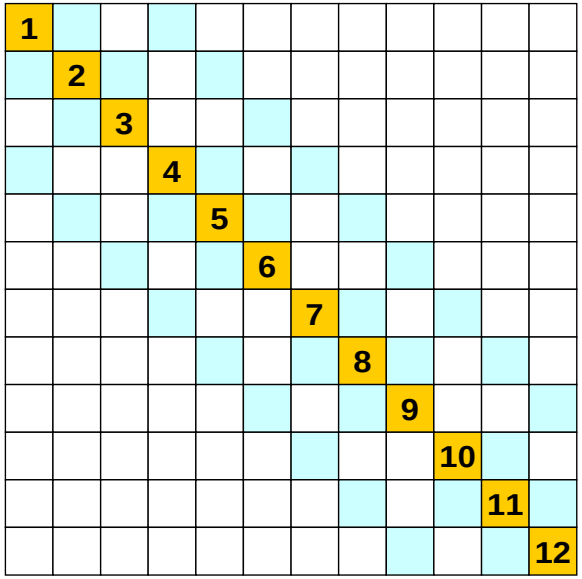
$$\begin{pmatrix}
4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00
\end{pmatrix}
\begin{pmatrix} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix}
X
=
\begin{pmatrix}
10.0 \\ 17.0 \\ 20.0 \\ 29.0 \\ 40.0 \\ 41.0 \\ 50.0 \\ 64.0 \\ 62.0 \\ 58.0 \\ 74.0 \\ 68.0
\end{pmatrix}$$

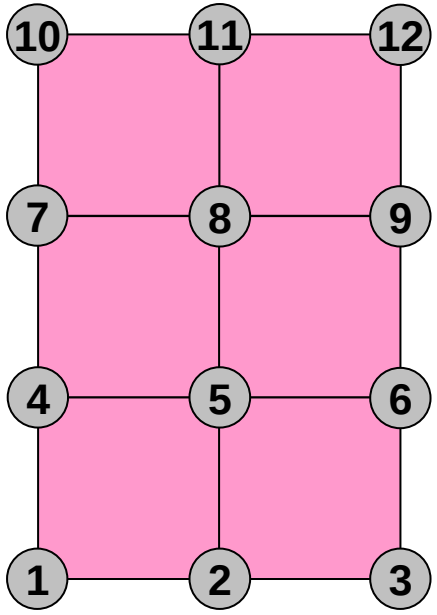




実例:解

$$\begin{pmatrix}
4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00
\end{pmatrix}
\begin{pmatrix}
1.00 \\ 2.00 \\ 3.00 \\ 4.00 \\ 5.00 \\ 6.00 \\ 7.00 \\ 8.00 \\ 9.00 \\ 10.00 \\ 11.00 \\ 12.00
\end{pmatrix}
=
\begin{pmatrix}
10.0 \\ 17.0 \\ 20.0 \\ 29.0 \\ 40.0 \\ 41.0 \\ 50.0 \\ 64.0 \\ 62.0 \\ 58.0 \\ 74.0 \\ 68.0
\end{pmatrix}$$





ファイルコピー

```
>$ cd <$07S>

FORTRAN
>$ cp /home/nakajima/class/2007summer/F/s3-f.tar .
>$ tar xvf s3-f.tar

C
>$ cp /home/nakajima/class/2007summer/C/s3-c.tar .
>$ tar xvf s3-c.tar
```

直下に「/S3」というディレクトリができている。
<$07S>/S3を<$S3>と呼ぶ。

(完全)LU分解，不完全LU分解

```
>$ cd <$07S>/S3
```

`lu1.f`	**完全**LU**分解によるプログラム**（FORTRAN）
`lu2.f`	**不**完全LU分解（fill-in無し）によるプログラム（FORTRAN）
`lu1`	`lu1.f` をコンパイル，リンクしたもの
`lu2`	`lu2.f` **をコンパイル，リンクしたもの**

C**ユーザーも**FORTRAN**でお願いします**

完全LU分解したマトリクス

./lu1 とタイプ

もとのマトリクス

$$\begin{pmatrix}
4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 0.00 & 0.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 4.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 1.00 & 4.00
\end{pmatrix}$$

LU分解したマトリクス（fill-inが生じている。もともと0だった成分が非ゼロになっている）

$$\begin{pmatrix}
4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & 3.75 & 1.00 & -0.25 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.27 & 3.73 & 0.07 & -0.27 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & -0.07 & 0.02 & 3.73 & 1.07 & -0.02 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.27 & -0.07 & 0.29 & 3.41 & 1.08 & -0.29 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.27 & 0.00 & 0.32 & 3.39 & 0.10 & -0.32 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.27 & -0.08 & 0.03 & 3.71 & 1.09 & -0.03 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.29 & -0.09 & 0.30 & 3.35 & 1.10 & -0.30 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & -0.01 & 0.33 & 3.34 & 0.10 & -0.33 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.27 & -0.09 & 0.03 & 3.70 & 1.10 & -0.03 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.30 & -0.10 & 0.30 & 3.34 & 1.11 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.30 & -0.01 & 0.33 & 3.33
\end{pmatrix}$$

不完全LU分解したマトリクス（fill-in無し）

./lu2 とタイプ

不完全LU分解した
マトリクス（fill-in無し）

4.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.25	3.75	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.27	3.73	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
0.25	0.00	0.00	3.75	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
0.00	0.27	0.00	0.27	3.47	1.00	0.00	1.00	0.00	0.00	0.00	0.00
0.00	0.00	0.27	0.00	0.29	3.44	0.00	0.00	1.00	0.00	0.00	0.00
0.00	0.00	0.00	0.27	0.00	0.00	3.73	1.00	0.00	1.00	0.00	0.00
0.00	0.00	0.00	0.00	0.29	0.00	0.27	3.44	1.00	0.00	1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.29	0.00	0.29	3.42	0.00	0.00	1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.27	0.00	0.00	3.73	1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.29	0.00	0.27	3.44	1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.29	0.00	0.29	3.42

完全LU分解した
マトリクス
（fill-inが生じている。もともと0だった成分が非ゼロになっている）

4.00	1.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.25	3.75	1.00	−0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.27	3.73	0.07	−0.27	1.00	0.00	0.00	0.00	0.00	0.00	0.00
0.25	−0.07	0.02	3.73	1.07	−0.02	1.00	0.00	0.00	0.00	0.00	0.00
0.00	0.27	−0.07	0.29	3.41	1.08	−0.29	1.00	0.00	0.00	0.00	0.00
0.00	0.00	0.27	0.00	0.32	3.39	0.10	−0.32	1.00	0.00	0.00	0.00
0.00	0.00	0.00	0.27	−0.08	0.03	3.71	1.09	−0.03	1.00	0.00	0.00
0.00	0.00	0.00	0.00	0.29	−0.09	0.30	3.35	1.10	−0.30	1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.29	−0.01	0.33	3.34	0.10	−0.33	1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.27	−0.09	0.03	3.70	1.10	−0.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	−0.10	0.30	3.34	1.11
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	−0.01	0.33	3.33

解の比較：ちょっと違う

不完全LU分解

$$\begin{pmatrix}
4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & 3.75 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.27 & 3.73 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & 0.00 & 0.00 & 3.75 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.27 & 0.00 & 0.27 & 3.47 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.27 & 0.00 & 0.29 & 3.44 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.27 & 0.00 & 0.00 & 3.73 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.00 & 0.27 & 3.44 & 1.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.00 & 0.29 & 3.42 & 0.00 & 0.00 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.27 & 0.00 & 0.00 & 3.73 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.00 & 0.27 & 3.44 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & 0.00 & 0.29 & 3.42
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1.0 \\ 1.9 \\ 2.8 \\ 4.0 \\ 4.6 \\ 5.8 \\ 7.0 \\ 7.3 \\ 8.4 \\ 9.6 \\ 10.6 \\ 12.2
\end{pmatrix}$$

完全LU分解

$$\begin{pmatrix}
4.00 & 1.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & 3.75 & 1.00 & -0.25 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.27 & 3.73 & 0.07 & -0.27 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.25 & -0.07 & 0.02 & 3.73 & 1.07 & -0.02 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.27 & -0.07 & 0.29 & 3.41 & 1.08 & -0.29 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.27 & 0.00 & 0.32 & 3.39 & 0.10 & -0.32 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.27 & -0.08 & 0.03 & 3.71 & 1.09 & -0.03 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.29 & -0.09 & 0.30 & 3.35 & 1.10 & -0.30 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.29 & -0.01 & 0.33 & 3.34 & 0.10 & -0.33 & 1.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.27 & -0.09 & 0.03 & 3.70 & 1.10 & -0.03 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.30 & -0.10 & 0.30 & 3.34 & 1.11 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.30 & -0.01 & 0.33 & 3.33
\end{pmatrix}
\rightarrow
\begin{pmatrix}
1.00 \\ 2.00 \\ 3.00 \\ 4.00 \\ 5.00 \\ 6.00 \\ 7.00 \\ 8.00 \\ 9.00 \\ 10.00 \\ 11.00 \\ 12.00
\end{pmatrix}$$

ILU(0), IC(0) 前処理

- Fill-inを全く考慮しない「不完全な」分解
  - 記憶容量，計算量削減
- これを解くと「不完全な」解が得られるが，本来の解とそれほどずれているわけではない
  - 問題に依存する

大規模線形ソルバーの動向

- 反復法がより広く使用されるようになりつつある
  - 100PEを超えるような大規模並列システムでは直接法は並列性能が出ない：逆にそれより小さければ直接法でもOKということになる。
  - 密行列も反復法で解くような試みがなされている。
- 密行列を使わないで済ませられるようなアルゴリズムの開発
  - 高速多重極展開（Fast Multipole）
  - 遠方からの効果をクラスタリング，あるいは無視
  - 密行列
    - メモリースケーラブルではない
- 前処理付き反復法（preconditioned iterative solvers）
  - 安定した前処理の必要性
  - 安定した前処理は概して「並列化」が困難
    - 前回のGauss-Seidelの「並列化」と同じ

前処理手法の分類:Trade-Off

Point Jacobi

Diagonal Blocking

ILU(0)

ILU(1)

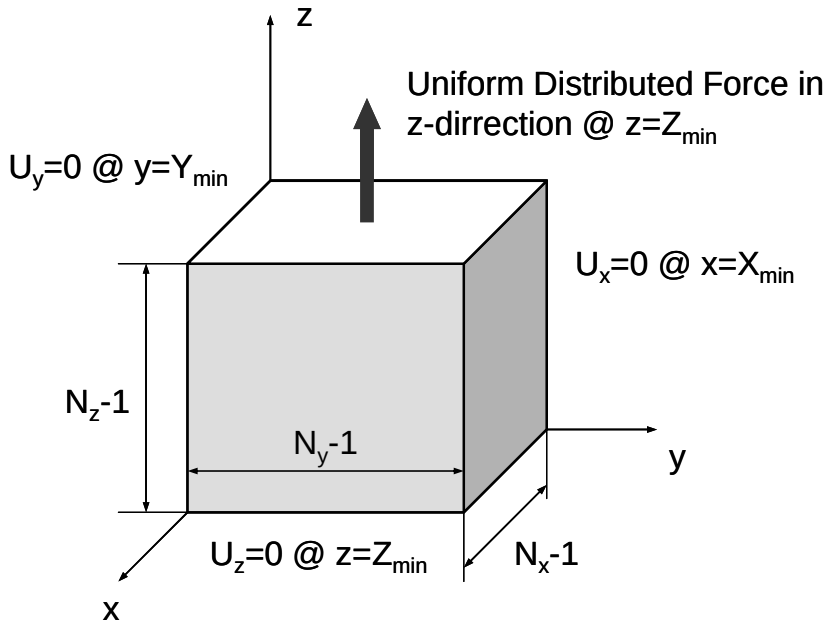
ILU(2)

Gaussian Elimination

- Simple
- Easy to be Parallelized
- Cheap

- Complicated
- Global Dependency
- Expensive

三次元弾性解析問題の例

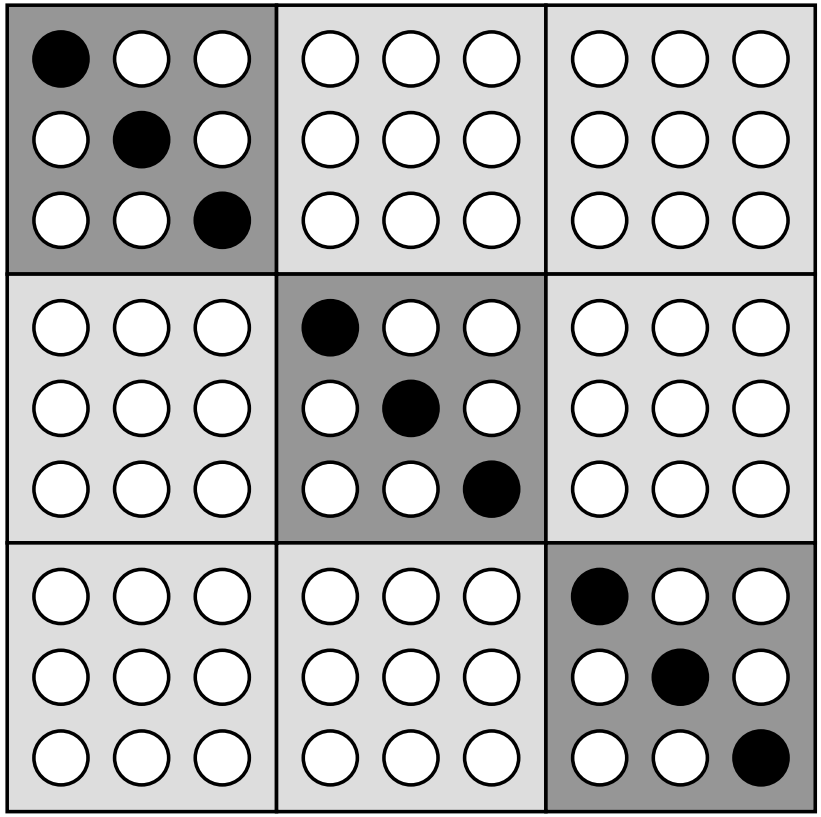


3×32×32×32= 98,304 DOF
一様物性，境界条件
条件は良い問題
1 PE

- 計算結果（$\varepsilon=10^{-8}$，cenju）
  - ILU（0）: 82回，12.22秒
  - Block Scaling: 279回，11.59秒
  - Point Jacobi（対角スケーリング） 283回，11.35秒
  - 前処理無し 298回，11.65秒

三次元弾性解析

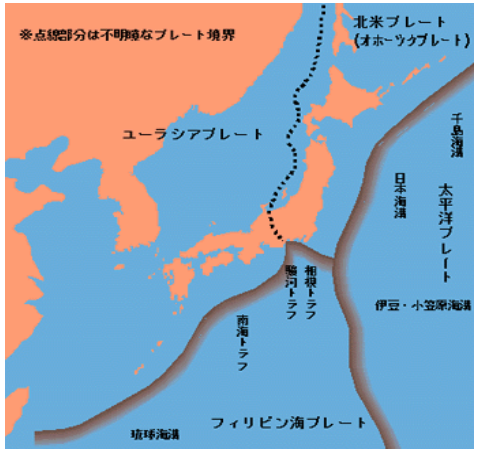
3自由度/節点をブロックとして扱う

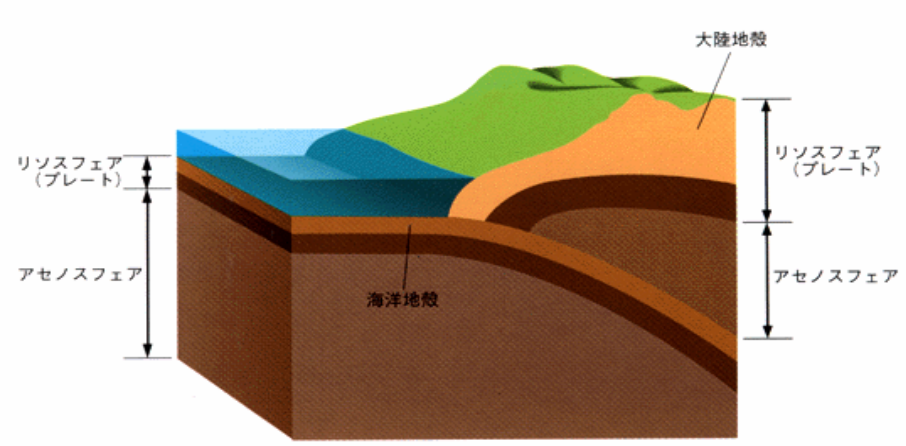


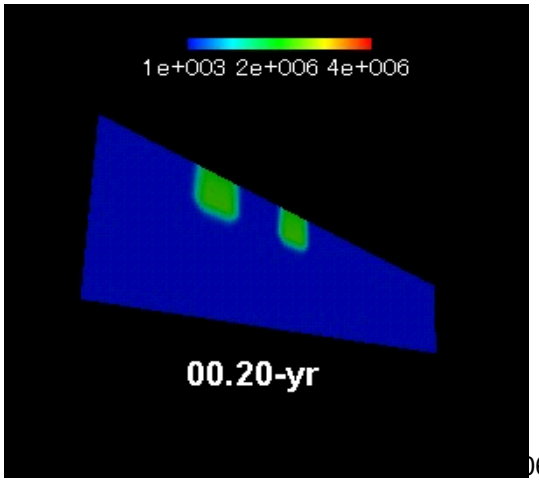
- 線形ソルバーの概要
  - 直接法
  - 並列法
  - 共役勾配法（Conjugate Gradient）
  - 前処理
- 接触問題の例（前処理）
  - Selective Blocking Preconditioning
- 課題S3

接触問題における前処理手法

- 地震発生サイクルシミュレーションにおける接触問題
  - 有限要素法
  - プレート境界における準静的応力蓄積過程
  - 非線形接触問題をNewton-Raphson法によって解く
  - ALM法（Augmented Lagrangean,拡大ラグランジェ法）による拘束条件：ペナルティ数







仮想仕事の原理と接触付帯条件

Iizuka,M.

内力項　　外力項　　体積力項

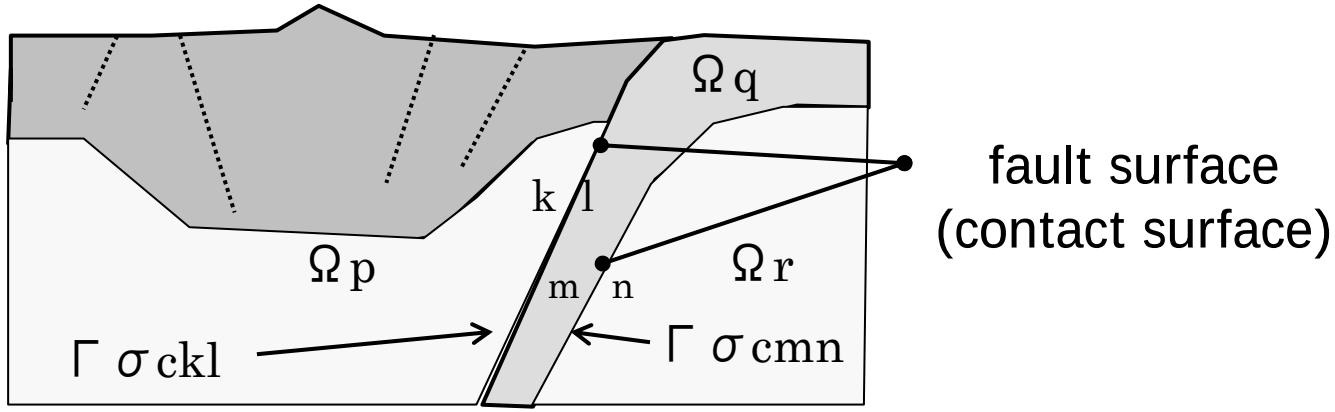
$$\sum_p \left[\int_{\Omega_p} \lfloor \delta\varepsilon \rfloor \{\sigma\} dV - \int_{\Gamma_{\sigma p}} \lfloor \delta u \rfloor \{f_o\} dS - \int_{\Omega_p} \lfloor \delta u \rfloor \{r_o\} dV \right]$$

$$= -\sum_{kl} \left[\int_{\Gamma_{\sigma ckl}} \lfloor \delta \left(\bar{\Delta} u \right) \rfloor \{f_{oc}\} dS \right]$$

接触力項

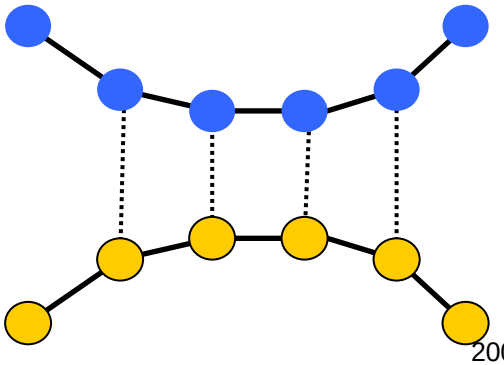
$$\int_{\Gamma_{\sigma ckl}} \lfloor \delta \left(\bar{f}_c^n \right) \rfloor \{g\} dS = 0$$

接触面での物体の重なりは無い



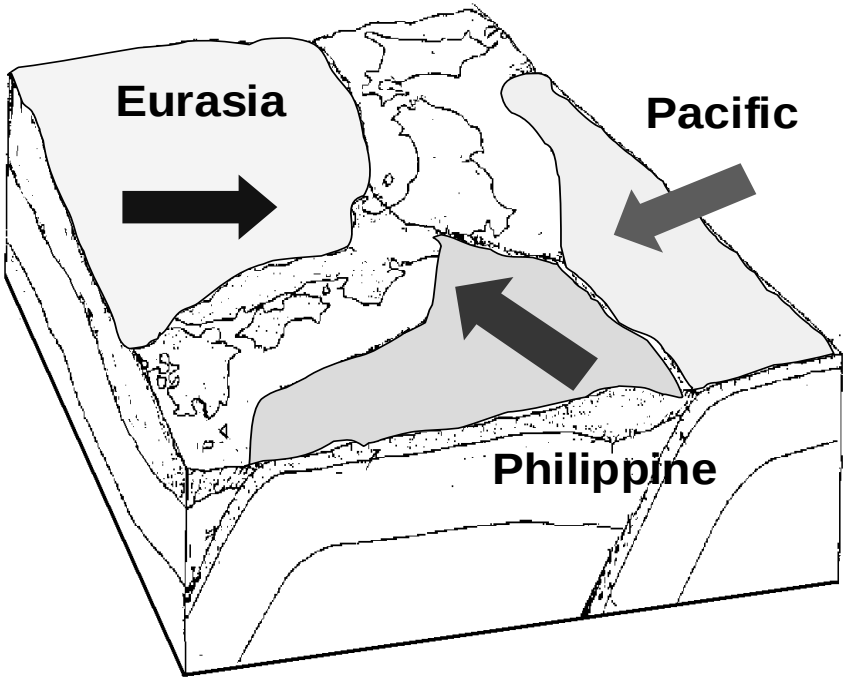
接触問題における前処理手法(続き)

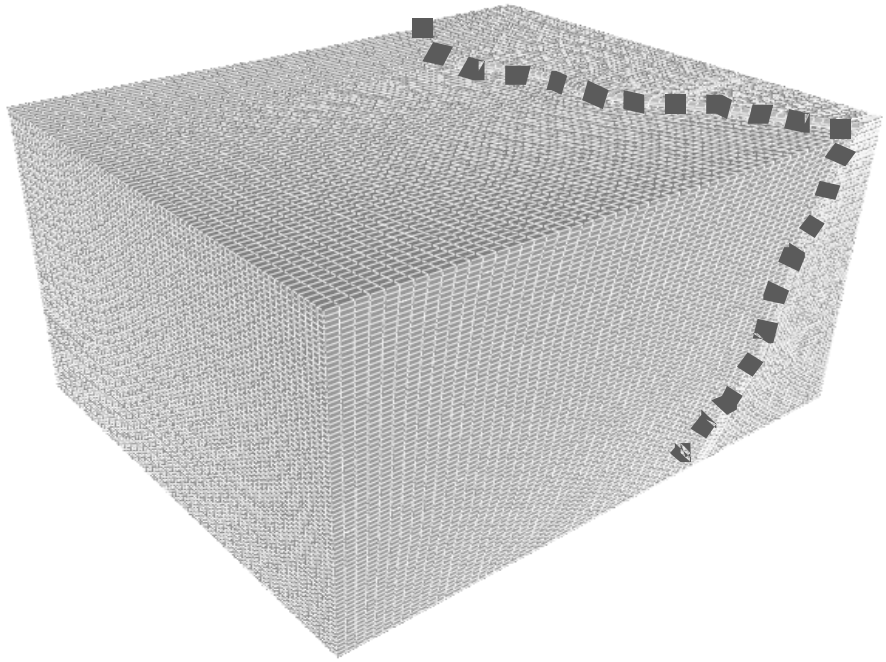
- 仮定
  - 微小変形理論に基づく，静的接触(接触グループに属する節点の座標は同一)
  - 摩擦なし:対称行列(最終的には摩擦ありの場合も計算)
- 特殊な前処理手法を開発: ***Selective Blocking.***
  - 三次元接触問題において，効率的に解を得ることのできる，効率的な前処理手法である
- 計算
  - 日立SR2201(東大):2001～2002
  - 地球シミュレータ:2002～



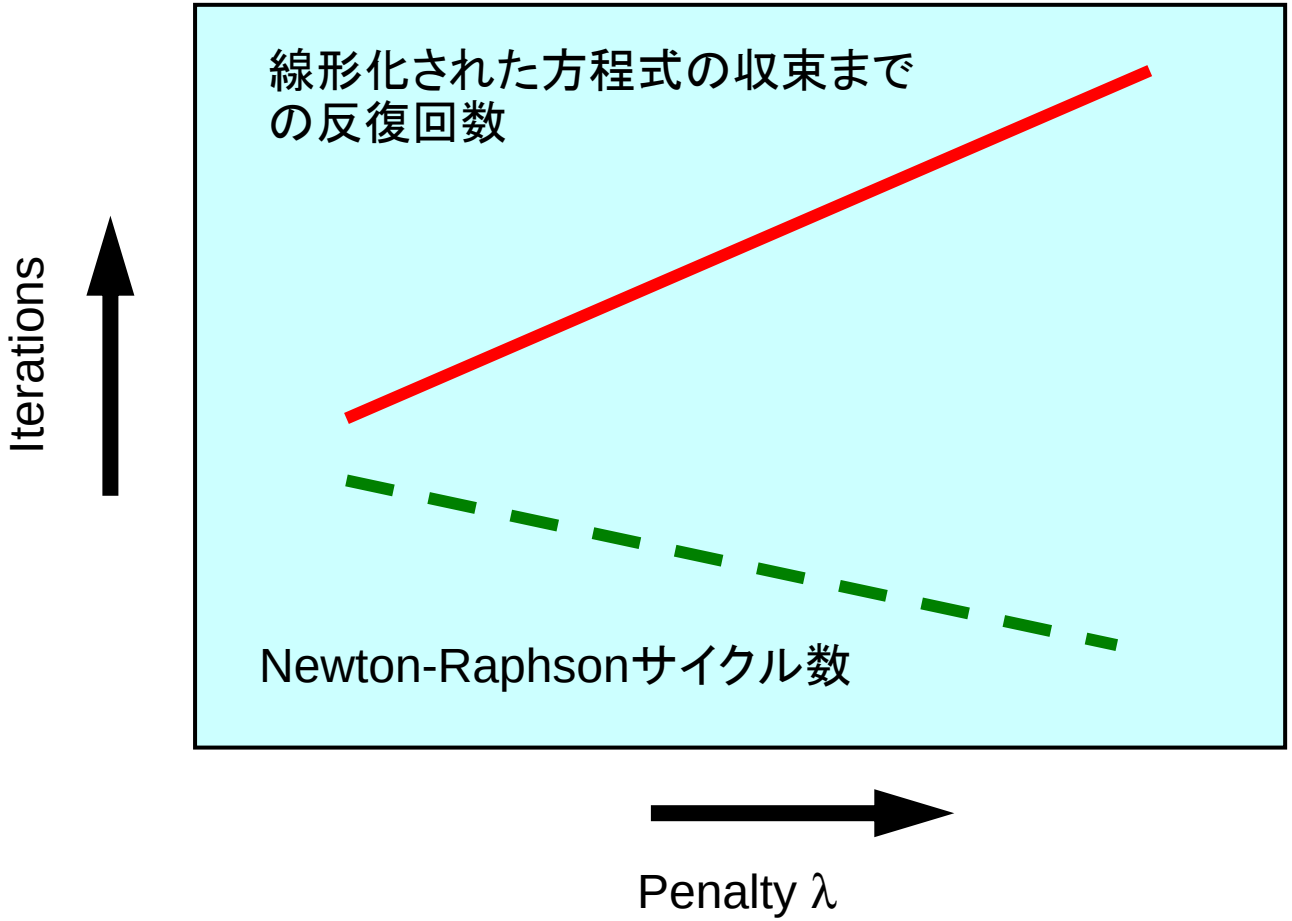
Geophysics Application w/Contact

Augmented Lagrangean Method with Penalty Constraint Condition for Contact





拡大ラグランジェ法
接触問題におけるペナルティ～反復回数の関係
Newton-Raphson / Iterative Solver



ペナルティ数が大きいと，接触条件の精度は高くなり，Newton-Raphsonサイクルも少ない反復回数で収束する。

しかし，線形化された方程式は悪条件となる。

予備的計算結果

ペナルティ拘束条件を含む弾性解析

27,888 nodes, 83,664 DOFs, $\varepsilon=10^{-8}$

Single PE case (Xeon 2.8MHz)

GeoFEM's Original Solvers (Scalar Version)

Preconditioning	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up+Solve (sec.)	Single Iteration (sec.)	Memory Size (MB)
Diagonal Scaling	10^2	1531	<0.01	75.1	75.1	0.049	119
	10^6	No Conv.	-	-	-	-	
IC(0) (Scalar Type)	10^2	401	0.02	39.2	39.2	0.098	119
	10^6	No Conv.	-	-	-	-	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

悪条件問題（Ill-Conditioned Problems）

- 基本的に直接法を使うべき問題。
- しかし，直接法では並列計算において限界がある。
- 安定した前処理手法が必要

- 対策
  - 直接法にできるだけ近い前処理手法
    - 深いFill-in：より多くのFill-inを考慮するということ
      - より正確な逆行列
  - ブロッキングとオーダリング

Deep Fill-in : LU and ILU(0)/IC(0)

Gaussian Elimination

```
do i= 2, n
  do k= 1,  i-1
    a_ik := a_ik/a_kk
    do j= k+1, n
      a_ij := a_ij - a_ik*a_kj
    enddo
  enddo
enddo
```

ILU(0) : keep non-zero pattern of the original coefficient matrix

```
do i= 2, n
  do k= 1,  i-1
    if ((i,k)∈ NonZero(A)) then
      a_ik := a_ik/a_kk
    endif
    do j= k+1, n
      if ((i,j)∈ NonZero(A)) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

DEEP Fill-in

Deep Fill-in : ILU(p)/IC(p)

LEV_{ij}=0 if ((i,j)∈ NonZero(A)) otherwise LEV_{ij}= p+1

```
do i= 2, n
  do k= 1,  i-1
    if (LEV_ik≦p) then
      a_ik := a_ik/a_kk
    endif
    do j= k+1, n
      if (LEV_ij = min(LEV_ij,1+LEV_ik+ LEV_kj)≦p) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

DEEP Fill-in

深いFill-inの効用

- 本ケースでは，有限要素法のため，もともとの係数行列がかなり「疎＝0が多い」
- Fill-inを深くとる，すなわち多くのFill-inを考慮することによって
  - 正確な逆行列に近づく
  - 必要メモリ量，計算量が増加する。ILU(0)からILU(1)で2倍。

DEEP Fill-in

Blocking：ILU/ICの前進交代代入

$M = (L+D)D^{-1}(D+U)$

前進代入：Forward Substitution

$(L+D)p = q$: $p = D^{-1}(q-Lp)$

後退代入：Backward Substitution

$(I+D^{-1}U)p_{new} = p_{old}$: $p = p - D^{-1}Up$

- D^{-1}を乗ずるところで，対角成分で単に割るのではなく，3×3ブロックにLU分解（ガウスの消去法）を施す。
  - 三次元固体力学の場合
  - 1節点に強くカップルした変位3成分がある
  - 間接参照が減り，計算効率も上がる

BLOCKING

Results in the Benchmark

27,888 nodes, 83,664 DOFs, $\varepsilon=10^{-8}$

Single PE case (Xeon 2.8MHz)

Effect of Blocking/Fill-in

Preconditioning	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up+Solve (sec.)	Single Iteration (sec.)	Memory Size (MB)
Diagonal Scaling	10^2	1531	<0.01	75.1	75.1	0.049	119
	10^6	No Conv.	-	-	-	-	
IC(0) (Scalar Type)	10^2	401	0.02	39.2	39.2	0.098	119
	10^6	No Conv.	-	-	-	-	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

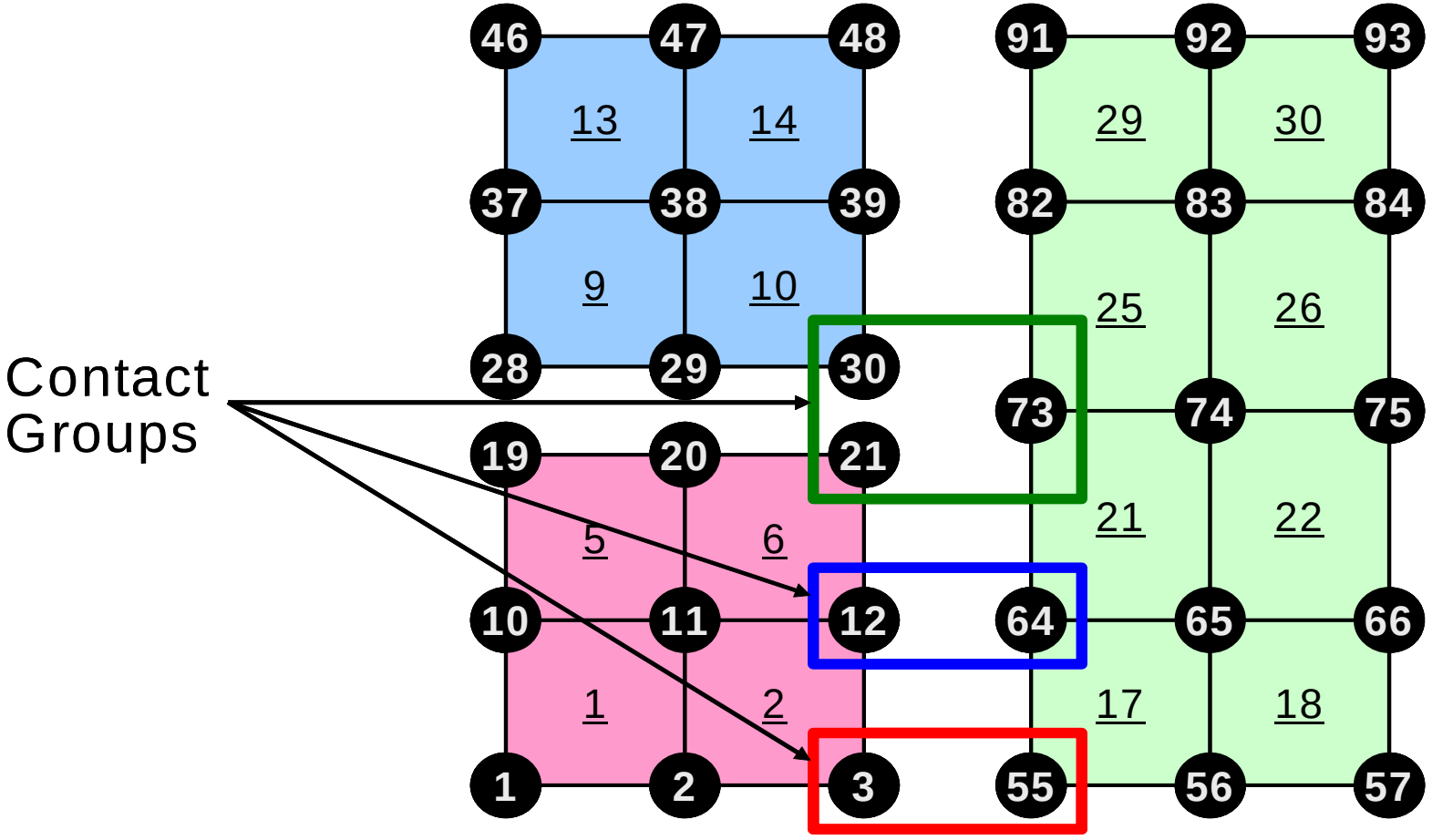
DEEP Fill-in　BLOCKING

ブロッキングとFill-inレベルの増加により，困難な問題が解けるようになった。

Selective Blocking

接触問題向けの特別な前処理手法

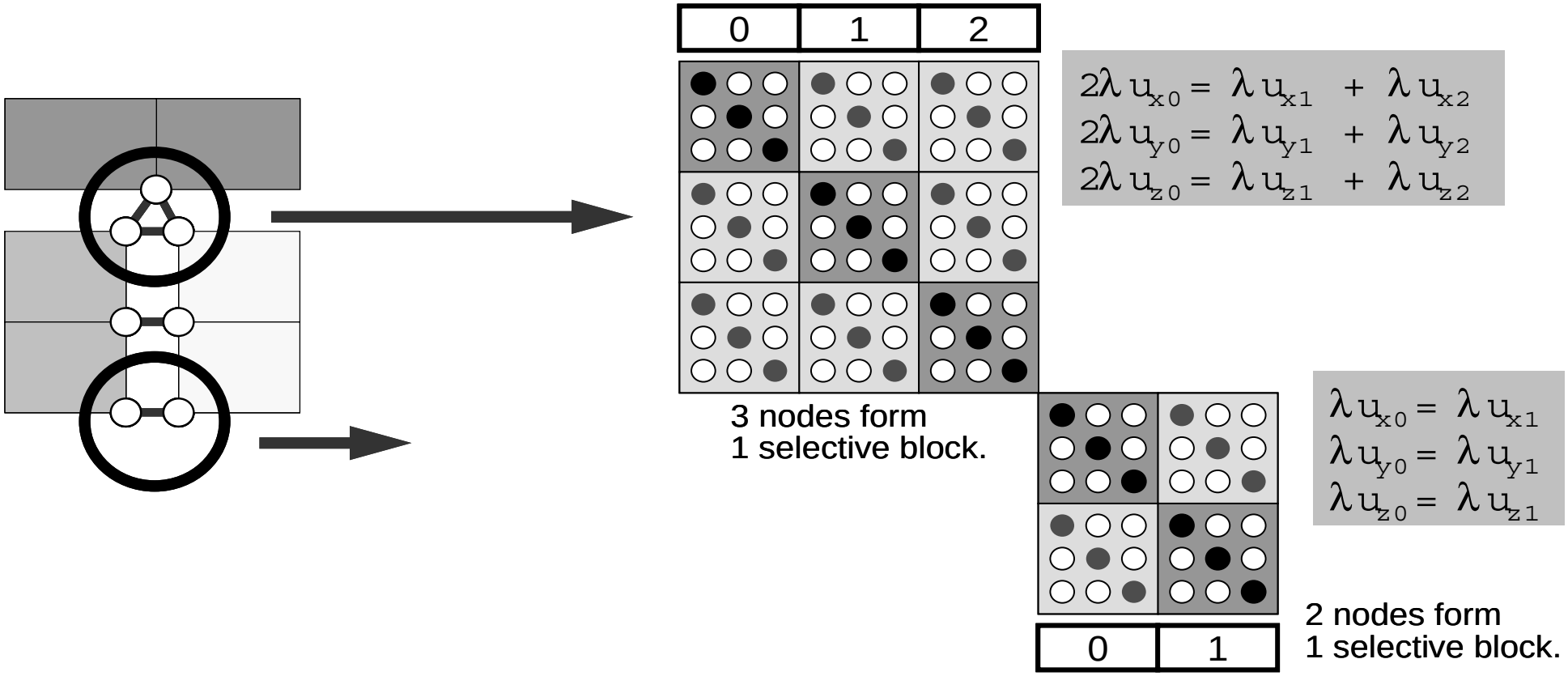
接触条件により物理的に強くカップルした節点群をブロック化



Selective Blocking

接触問題向けの特別な前処理手法

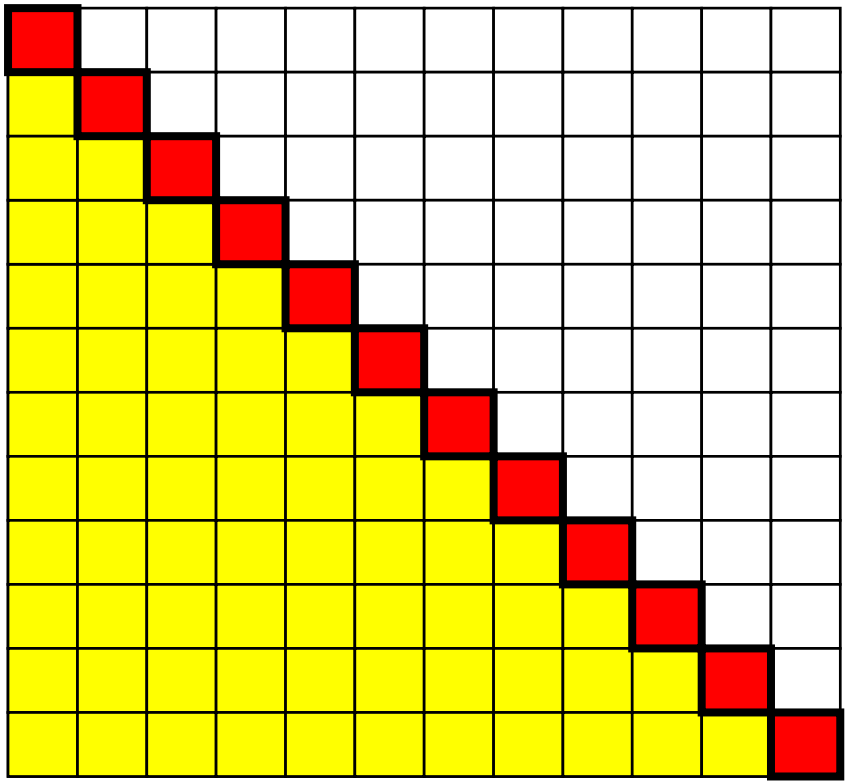
接触条件により物理的に強くカップルした節点群をブロック化



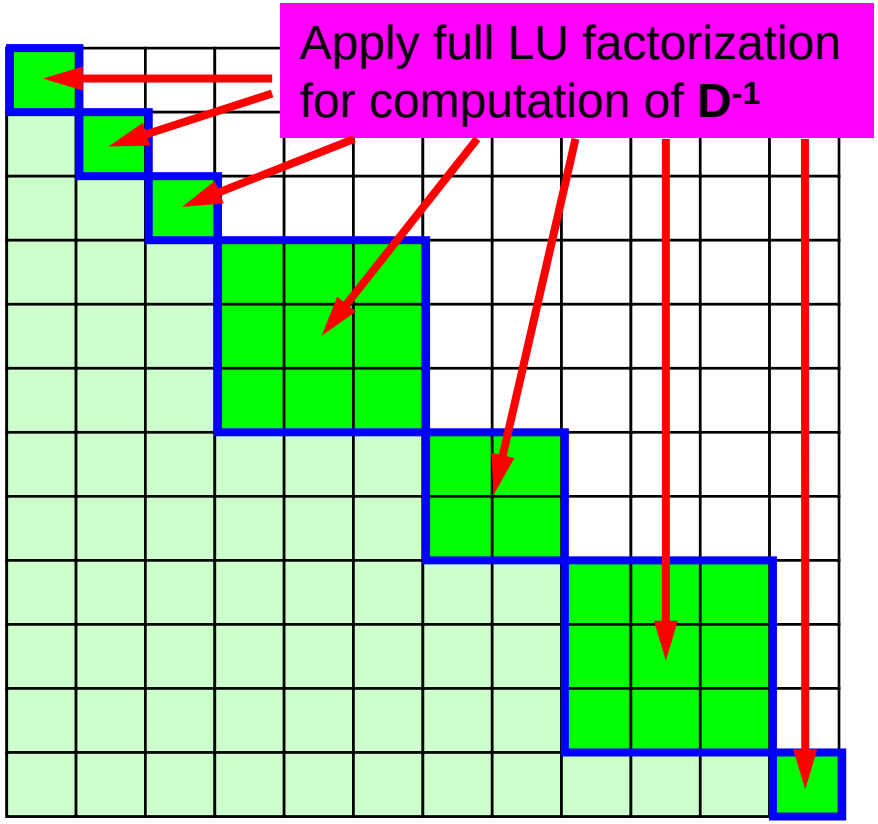
Selective Blocking

接触問題向けの特別な前処理手法

接触条件により物理的に強くカップルした節点群をブロック化



Block ILU/IC



Selective Blocking/ Supernode

size of each diagonal block depends on contact group size

Results in the Benchmark

27,888 nodes, 83,664 DOFs, $\varepsilon=10^{-8}$

Single PE case (Xeon 2.8MHz)

Selective Blocking：高速，省メモリ

Preconditioning	λ	Iterations	Set-up (sec.)	Solve (sec.)	Set-up+Solve (sec.)	Single Iteration (sec.)	Memory Size (MB)
Diagonal Scaling	10^2	1531	<0.01	75.1	75.1	0.049	119
	10^6	No Conv.	-	-	-	-	
IC(0) (Scalar Type)	10^2	401	0.02	39.2	39.2	0.098	119
	10^6	No Conv.	-	-	-	-	
BIC(0)	10^2	388	0.02	37.4	37.4	0.097	59
	10^6	2590	0.01	252.3	252.3	0.097	
BIC(1)	10^2	77	8.5	11.7	20.2	0.152	176
	10^6	78	8.5	11.8	20.3	0.152	
BIC(2)	10^2	59	16.9	13.9	30.8	0.236	319
	10^6	59	16.9	13.9	30.8	0.236	
SB-BIC(0)	10^0	114	0.10	12.9	13.0	0.113	67
	10^6	114	0.10	12.9	13.0	0.113	

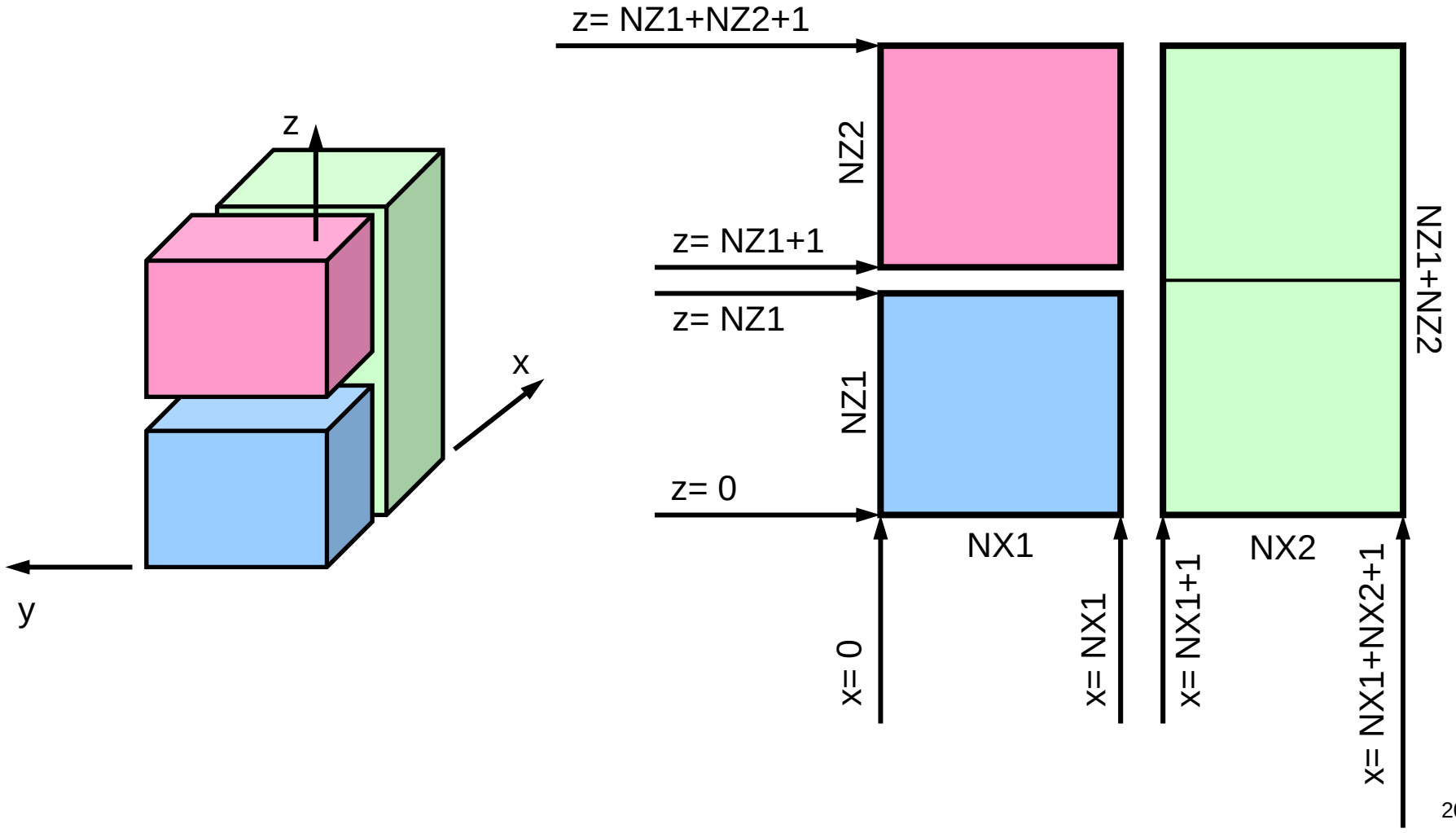
Selective Blockingの特徴

BILU(1)/BILU(2)との比較

- $[M]^{-1}[A]$ **の固有値から計算される条件数（**condition nunmber**，最大最小固有値の比）は**BILU(1)**より大きいが，反復あたりの計算量は少ない。**
- 1PE**を使用したベンチマーク問題における固有値解析**
  - Simple Block
  - SWJ（Southwest Japan）

Simple Block Model

Description



Simple Block Model

Preconditioning	λ	Iter #	sec.
BIC(0)	10^2	388	202.
	10^4	No Conv.	N/A
BIC(1)	10^2	77	89.
	10^6	77	89.
	10^{10}	78	90.
BIC(2)	10^2	59	135.
	10^6	59	135.
	10^{10}	60	137.
SB-BIC(0)	10^2	114	61.
	10^6	114	61.
	10^{10}	114	61.

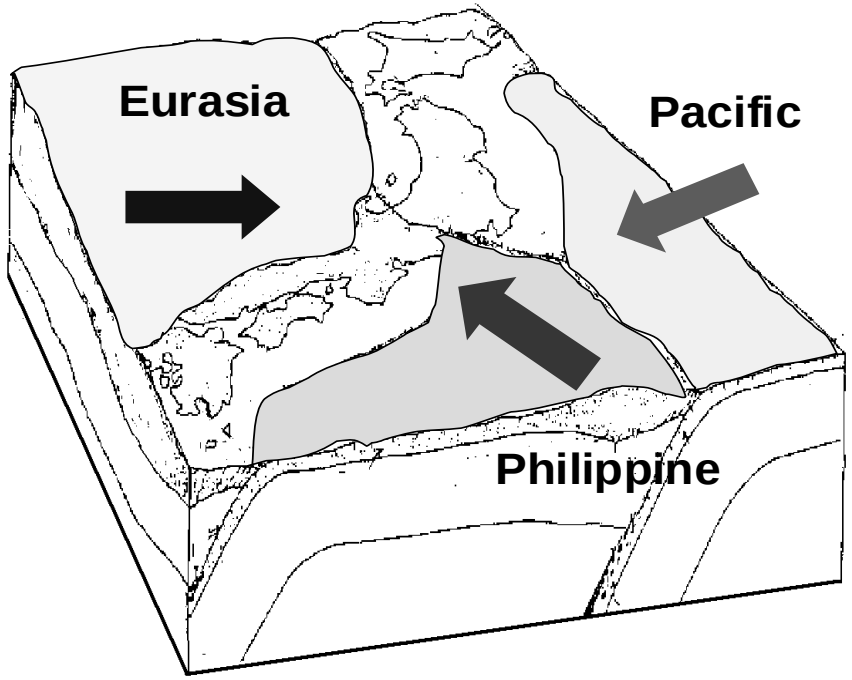
Simple Block Model ($\kappa = E_{max}/E_{min}$)：条件数

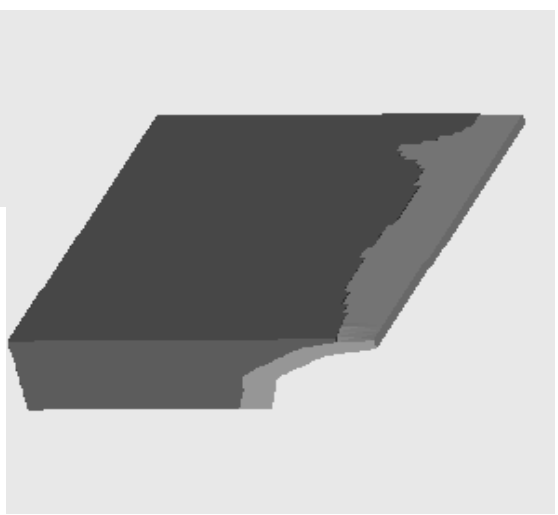
κ 条件数
E_{max} 最大固有値
E_{min} 最小固有値

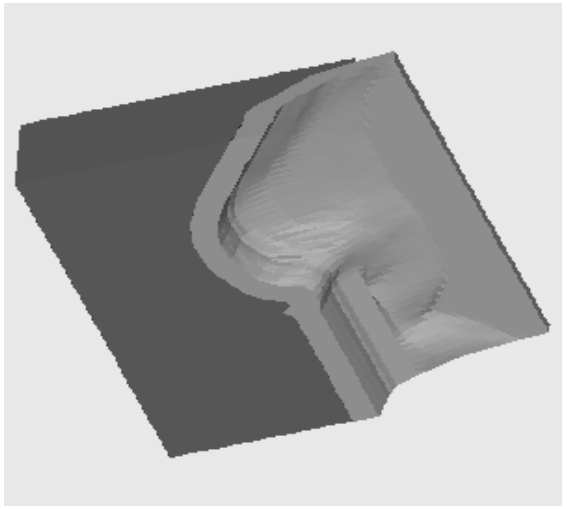
Preconditioning		$\lambda=10^2$	$\lambda=10^6$	$\lambda=10^{10}$
BIC(0)	E_{min}	4.845568E-03	4.865363E-07	4.865374E-11
	E_{max}	1.975620E+00	1.999998E+00	2.000000E+00
	κ	4.077170E+02	4.110686E+06	4.110681E+10
BIC(1)	E_{min}	8.901426E-01	8.890643E-01	8.890641E-01
	E_{max}	1.013930E+00	1.013863E+00	1.013863E+00
	κ	1.139065E+00	1.140371E+00	1.140371E+00
BIC(2)	E_{min}	9.003662E-01	8.992896E-01	8.992895E-01
	E_{max}	1.020256E+00	1.020144E+00	3.020144E+00
	κ	1.133157E+00	1.134388E+00	1.134389E+00
SB-BIC(0)	E_{min}	6.814392E-01	6.816873E-01	6.816873E-01
	E_{max}	1.005071E+00	1.005071E+00	1.005071E+00
	κ	1.474924E+00	1.474387E+00	1.474387E+00

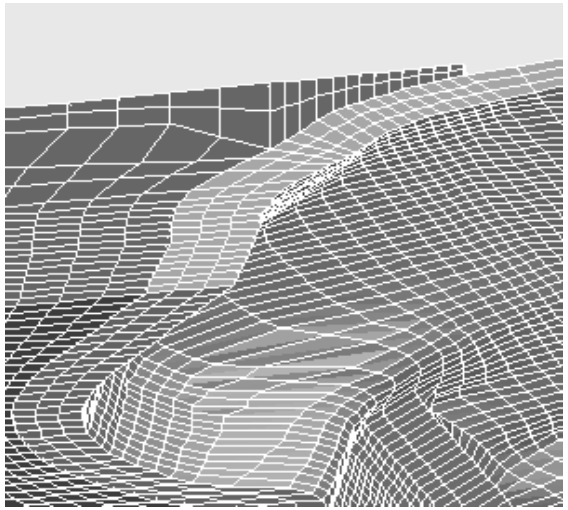
South-West Japan (SWJ)

fixed at $z=z_{min}$ + body force









SWJ

Preconditioning	λ	Iter #	sec.
BIC(0)	10^2	344	172.
	10^4	> 1000	N/A
BIC(1)	10^2	201	192.
	10^4	256	237.
	10^6	256	237.
BIC(2)	10^2	176	288.
	10^4	229	360.
	10^6	230	361.
SB-BIC(0)	10^2	297	149.
	10^4	295	148.
	10^6	295	148.

SWJ ($\kappa = E_{max}/E_{min}$) ：条件数

κ 条件数
E_{max} 最大固有値
E_{min} 最小固有値

Preconditioning		$\lambda=10^2$	$\lambda=10^4$	$\lambda=10^6$	$\lambda=10^{10}$
BIC(0)	E_{min}	1.970395E-02	1.999700E-04	1.999997E-06	2.000000E-10
	E_{max}	1.005194E+00	1.005194E+00	1.005194E+00	1.005194E+00
	κ	5.101486E+01	5.026725E+03	5.025979E+05	5.025971E+09
BIC(1)	E_{min}	3.351178E-01	2.294832E-01	2.286390E-01	2.286306E-01
	E_{max}	1.142246E+00	1.142041E+00	1.142039E+00	1.142039E+00
	κ	3.408491E+00	4.976580E+00	4.994944E+00	4.995128E+00
BIC(2)	E_{min}	3.558432E-01	2.364909E-01	2.346180E-01	2.345990E-01
	E_{max}	1.058883E+00	1.088397E+00	1.089189E+00	1.089196E+00
	κ	2.975702E+00	4.602277E+00	4.642391E+00	4.642800E+00
SB-BIC(0)	E_{min}	2.380572E-01	2.506369E-01	2.507947E-01	2.507963E-01
	E_{max}	1.005194E+00	1.005455E+00	1.005465E+00	1.005466E+00
	κ	4.222491E+00	4.011600E+00	4.009117E+00	4.009092E+00

並列計算をやってみると・・・

27,888 nodes, 83,664 DOFs, $\varepsilon=10^{-8}$

Single/4PE PE case (Xeon 2.8MHz), $\lambda=10^{6}$

Single PE

Block IC(0)	:	2,590 iters,	252.3 sec.
Block IC(1)	:	78 iters,	20.3 sec.
Block IC(2)	:	59 iters,	30.8 sec.
SB-BIC(0)	:	114 iters,	13.0 sec.

4 PEs

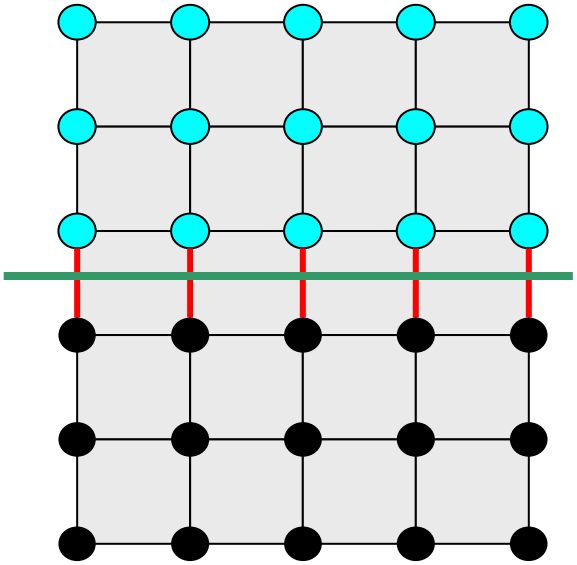
Block IC(0)	:	4,825 iters,	50.6 sec.
Block IC(1)	:	2,701 iters,	47.7 sec.
Block IC(2)	:	2,448 iters,	73.9 sec.
SB-BIC(0)	:	3,498 iters,	58.2 sec.

- **反復回数が増加して計算時間がかかってしまう・・・**
  - Selective Block**内の節点が違う領域にばらばらに分割された場合**

領域分割法の工夫

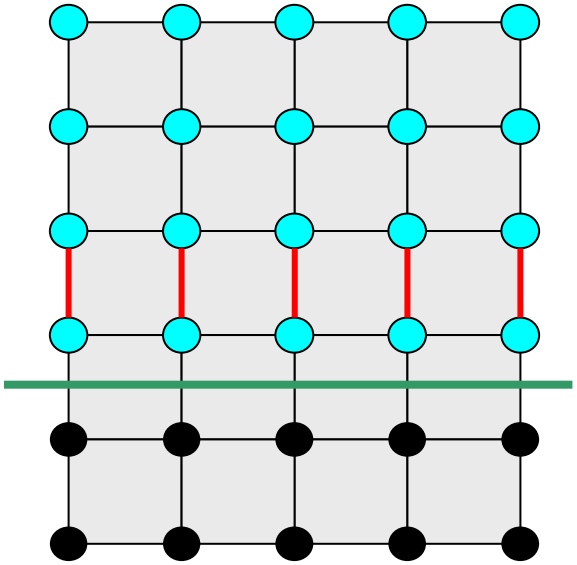
Selectiveブロック内の節点が違う領域に分割されると収束が遅くなる。

Selectiveブロック内の節点が同じ領域の「内点」となるように再領域分割を実施する。＋負荷分散



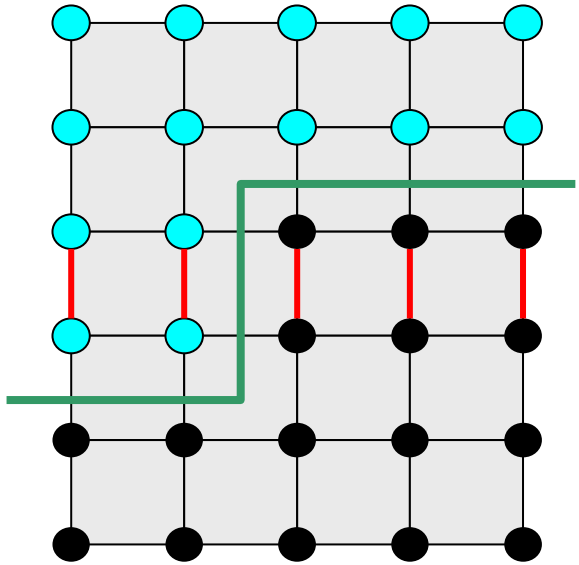
BEFORE repartitioning

Nodes in selective blocks are on separated partition.



AFTER repartitioning

Nodes in selective blocks are on same partition, but no load-balancing.



AFTER load-balancing

Nodes in selective blocks are on same partition, and load-balanced.

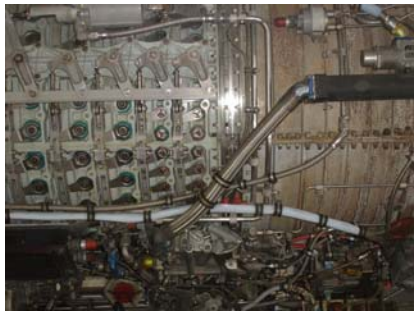
再領域分割の効果

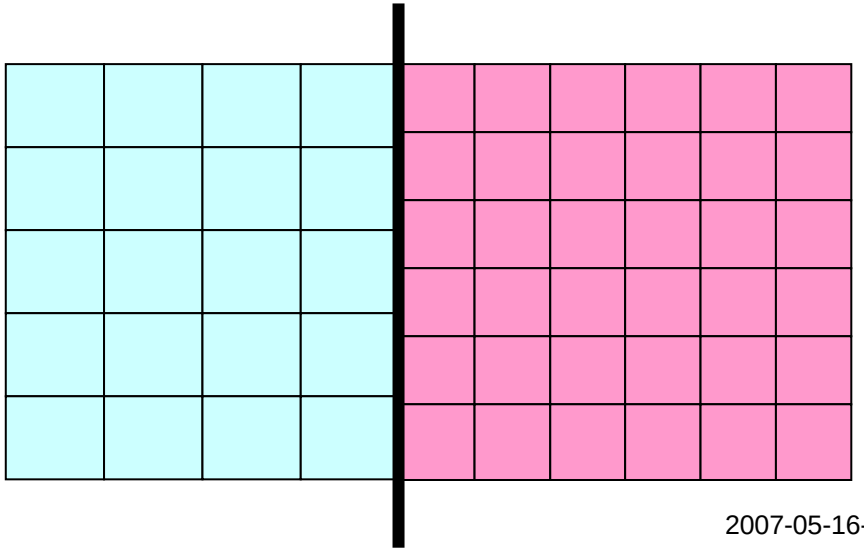
Benchmark: 4 PE cases

		ORIGINAL Partitioning		**IMPROVED** Partitioning	
Preconditioning	λ	Iterations	Set-up+Solve (sec.)	Iterations	Set-up+Solve (sec.)
BIC(0)	10^2	703	7.5	489	5.3
	10^6	4825	50.6	3477	37.5
BIC(1)	10^2	613	11.3	123	2.7
	10^6	2701	47.7	123	2.7
BIC(2)	10^2	610	19.5	112	4.7
	10^6	2448	73.9	112	4.7
SB-BIC(0)	10^0	655	10.9	165	2.9
	10^6	3498	58.2	166	2.9

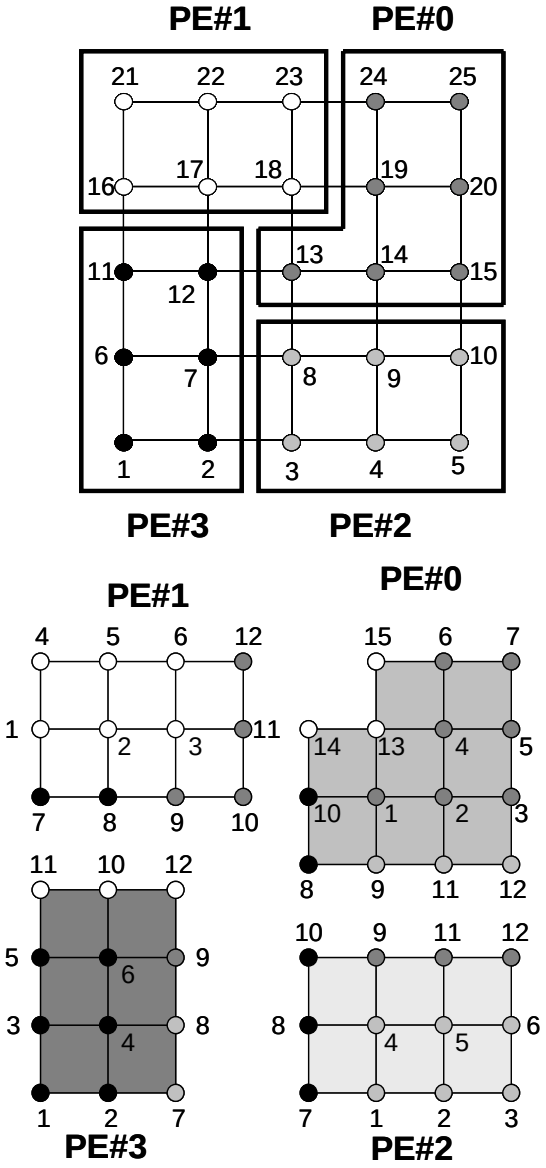
より一般的な問題

- 大すべりで，接触面が移動する場合。
- 元々，接触面の節点位置がずれている場合
  - 機械部品（はめ込み，ねじ止め）の解析で多用される。各部分を別々にメッシュ生成するのでこのようなケースは多い。
- 前項で述べたような特殊な領域分割は適用が困難な場合もある。

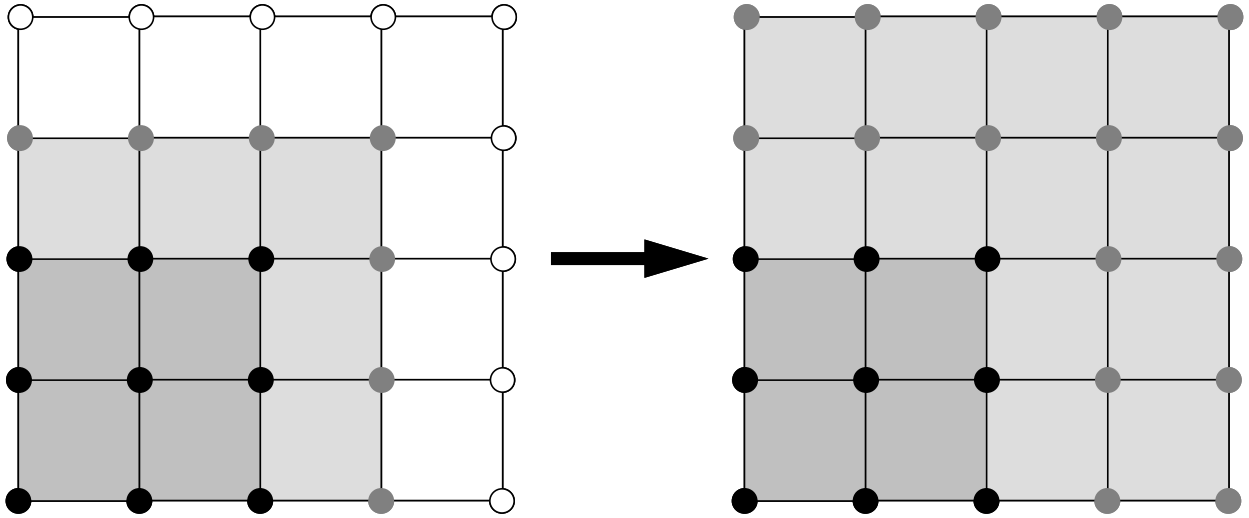




対策：領域間オーバーラップの拡張



計算量，通信量は増加



●：Internal Nodes, ●：External Nodes
■：Overlapped Elements

ここまでのまとめ

- 線形ソルバー
  - 密行列，疎行列
  - 直接法，反復法
- 前処理付き反復法の例：接触問題
  - 問題の特性（物理的，数学的）に基づいた前処理手法
    - SMASH
  - 適切な前処理を施すことによって，安定な解を得られる
  - 並列化
    - 一筋縄では行かない，概して難しい問題ほど並列にはしにくい
- 実問題への適用
  - 既存手法，公開ライブラリ ⇒ 個別の対応が必要な場合あり
- ILU/IC前処理，マルチグリッドなどの前処理手法については後期授業でとりあげる予定

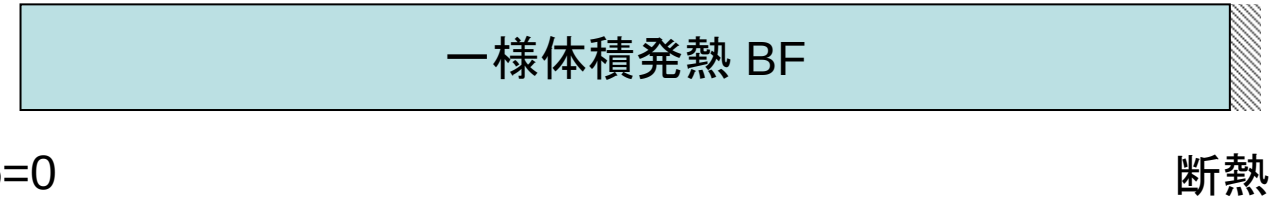
- 線形ソルバーの概要
  - 直接法
  - 反復法
  - 共役勾配法（Conjugate Gradient）
  - 前処理
- 接触問題の例（前処理）
  - Selective Blocking Preconditioning
- 課題S3

一次元熱伝導方程式(1/4)

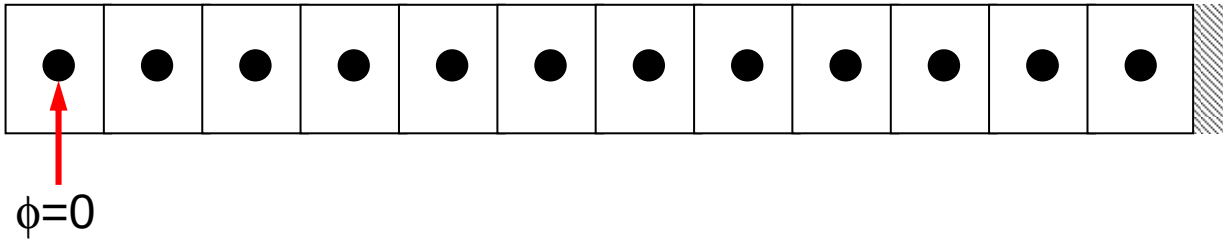
支配方程式:熱伝導率＝1(一様)

$$\frac{d^2\phi}{dx^2} + BF = 0, \quad \phi = 0 @ x = 0, \frac{d\phi}{dx} = 0 @ x = x_{\max}$$

$$\phi = -\frac{1}{2} BF\, x^2 + BF\, x_{\max}\, x$$



以下のような離散化(要素中心で従属変数を定義)をしているので注意が必要

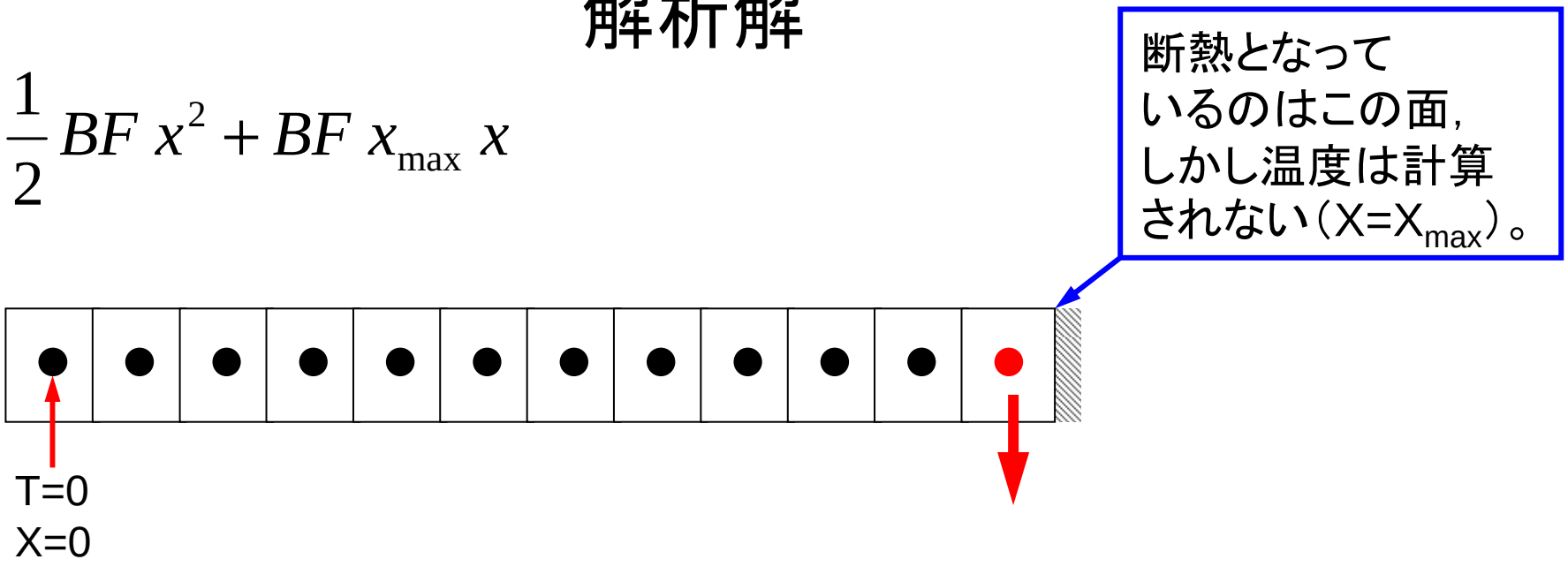


断熱となっているのはこの面,しかし温度は計算されない

一次元熱伝導方程式(2/4)

解析解

$$\phi = -\frac{1}{2} BF\, x^2 + BF\, x_{\max}\, x$$



Δx=1.d0, メッシュ数=50, とすると, X_{max}=49.5,
●の点のX座標は49.0となる。BF=1.0d0とすると●での温度は:

$$\phi = -\frac{1}{2} 49^2 + 49.5 \times 49 = -1200.5 + 9850.5 = 1225$$

一次元熱伝導方程式(3/4)

連立一次方程式

- 差分法による離散化

$$\left(\frac{d^2\phi}{dx^2}\right)_i \approx \frac{\left(\frac{d\phi}{dx}\right)_{i+1/2} - \left(\frac{d\phi}{dx}\right)_{i-1/2}}{\Delta x} = \frac{\frac{\phi_{i+1}-\phi_i}{\Delta x} - \frac{\phi_i-\phi_{i-1}}{\Delta x}}{\Delta x} = \frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}$$

- 各要素における線形方程式は以下のような形になる
  - これを共役勾配法(Conjugate Gradient法)で解く

$$\frac{d^2\phi}{dx^2} + BF = 0 \quad \rightarrow$$

$$\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2} + BF(i) = 0 \quad (1 \le i \le N)$$

$$\frac{1}{\Delta x^2}\phi_{i+1} - \frac{2}{\Delta x^2}\phi_i + \frac{1}{\Delta x^2}\phi_{i-1} + BF(i) = 0 \quad (1 \le i \le N)$$

$$A_L(i)\times\phi_{i-1} + A_D(i)\times\phi_i + A_R(i)\times\phi_{i+1} = BF(i) \quad (1 \le i \le N)$$

$$A_L(i) = \frac{1}{\Delta x^2},\ A_D(i) = -\frac{2}{\Delta x^2},\ A_R(i) = \frac{1}{\Delta x^2}$$

一次元熱伝導方程式(4/4)

係数行列の格納形式

各要素における線形方程式

$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

より一般的な形で格納すると・・・

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad(i=1,\ldots,N)$$

対角成分　　非対角成分

N=8の場合：連立一次方程式

自分とその周囲のみに非ゼロ成分：疎行列

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

$\times$

$\phi(1)$	BF(1)
$\phi(2)$	BF(2)
$\phi(3)$	BF(3)
$\phi(4)$	BF(4)
$\phi(5)$	BF(5)
$\phi(6)$	BF(6)
$\phi(7)$	BF(7)
$\phi(8)$	BF(8)

$=$

$$A_D(1)\times\phi(1)+A_R(1)\times\phi(2)=BF(1)$$
$$A_L(2)\times\phi(1)+A_D(2)\times\phi(2)+A_R(2)\times\phi(3)=BF(2)$$
$$A_L(3)\times\phi(2)+A_D(3)\times\phi(3)+A_R(3)\times\phi(4)=BF(3)$$
$$A_L(4)\times\phi(3)+A_D(4)\times\phi(4)+A_R(4)\times\phi(5)=BF(4)$$
$$A_L(5)\times\phi(4)+A_D(5)\times\phi(5)+A_R(5)\times\phi(6)=BF(5)$$
$$A_L(6)\times\phi(5)+A_D(6)\times\phi(6)+A_R(6)\times\phi(7)=BF(6)$$
$$A_L(7)\times\phi(6)+A_D(7)\times\phi(7)+A_R(7)\times\phi(8)=BF(7)$$
$$A_L(8)\times\phi(7)+A_D(8)\times\phi(8)=BF(8)$$

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

対角成分　　非対角成分

解いている方程式自体は課題S2と同じ ！

係数行列の格納形式(1/8)

非ゼロ成分のみを格納，疎行列向け方法

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

DIAG(i)	対角成分(実数，i=1,N)
INDEX(i)	非対角成分数に関する一次元配列 (整数，i=0,N)
ITEM(k)	非対角成分の要素番号 (整数，k=1， INDEX(N))
AMAT(k)	非対角成分 (実数，k=1， INDEX(N))

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

行列ベクトル積への適用

非ゼロ成分のみを格納，疎行列向け方法

DIAG(i)	対角成分(実数，i=1,N)
INDEX(i)	非対角成分数に関する一次元配列 (整数，i=0,N)
ITEM(k)	非対角成分の要素番号 (整数，k=1， INDEX(N))
AMAT(k)	非対角成分 (実数，k=1， INDEX(N))

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

```
{Y}= [A]{X}

do i= 1, N
  Y(i)= D(i)*X(i)
  do k= INDEX(i-1)+1, INDEX(i)
    Y(i)= Y(i) + AMAT(k)*X(ITEM(k))
  enddo
enddo
```

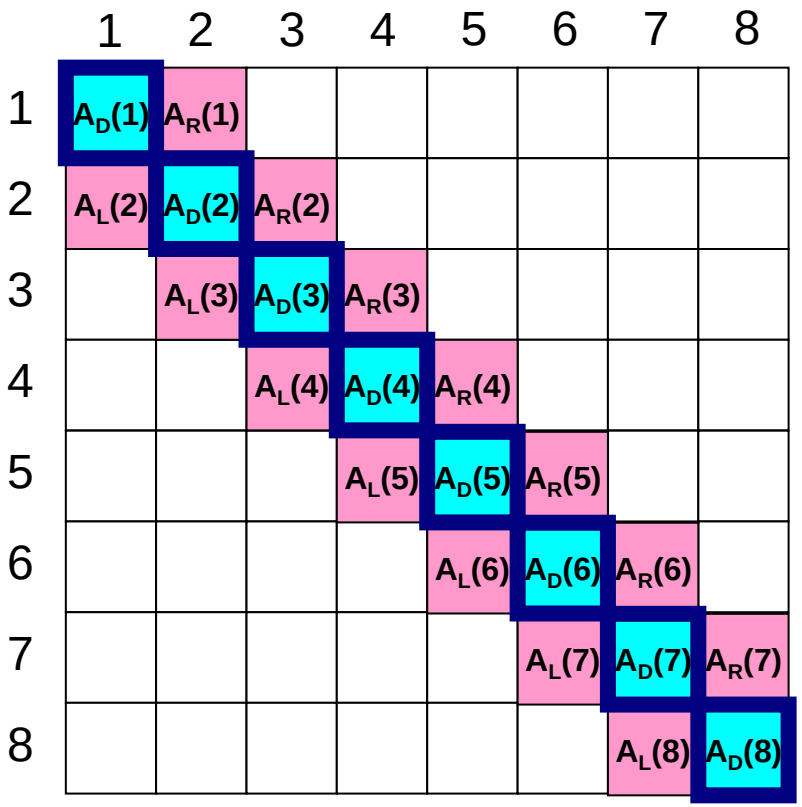
	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

係数行列の格納形式(2/8)
対角成分:DIAG

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

DIAG(i)　対角成分(実数, i=1,N)

DIAG(1)= $A_D(1)$
DIAG(2)= $A_D(2)$
DIAG(3)= $A_D(3)$
…
DIAG(8)= $A_D(8)$



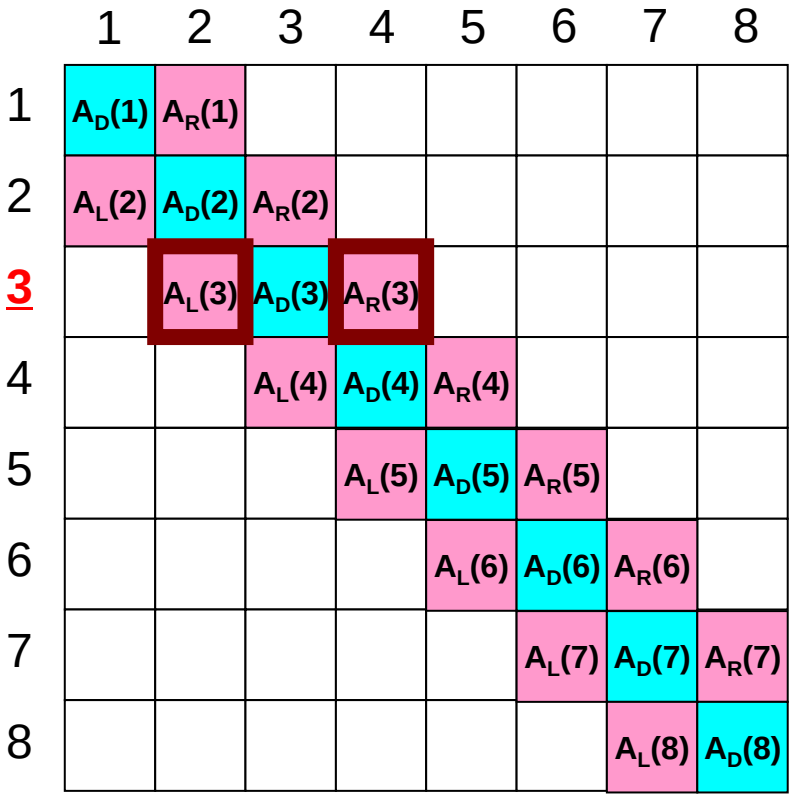
係数行列の格納形式(3/8)
INDEX, AMATの関係

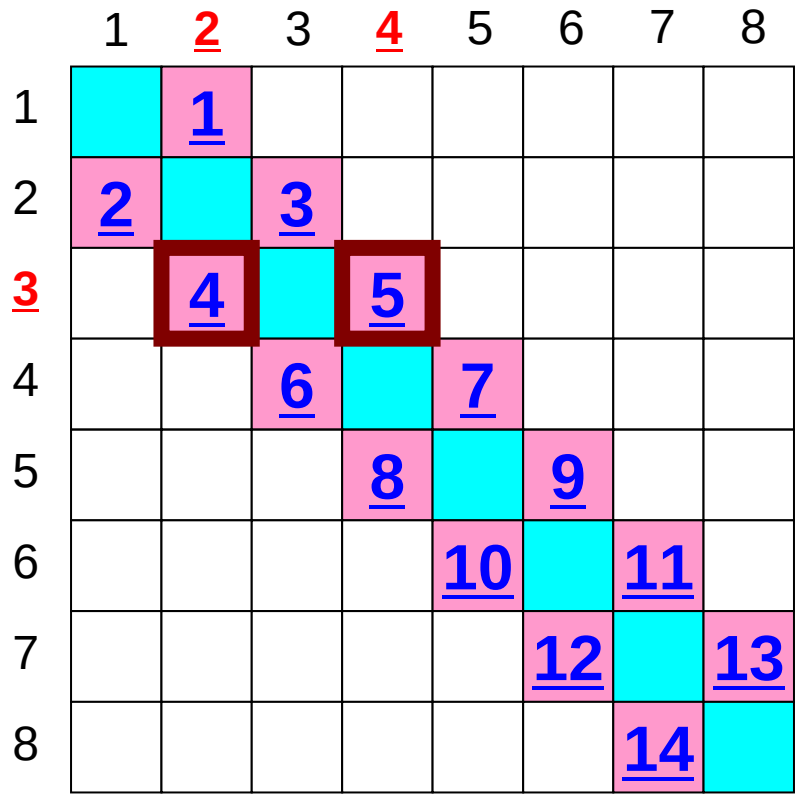
	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

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

```
INDEX(2)=3, INDEX(3)=5
ITEM(4)=2, AMAT(4)= A_L(3)
ITEM(5)=4, AMAT(5)= A_R(3)
```

係数行列の格納形式(3/8)
INDEX, AMATの関係





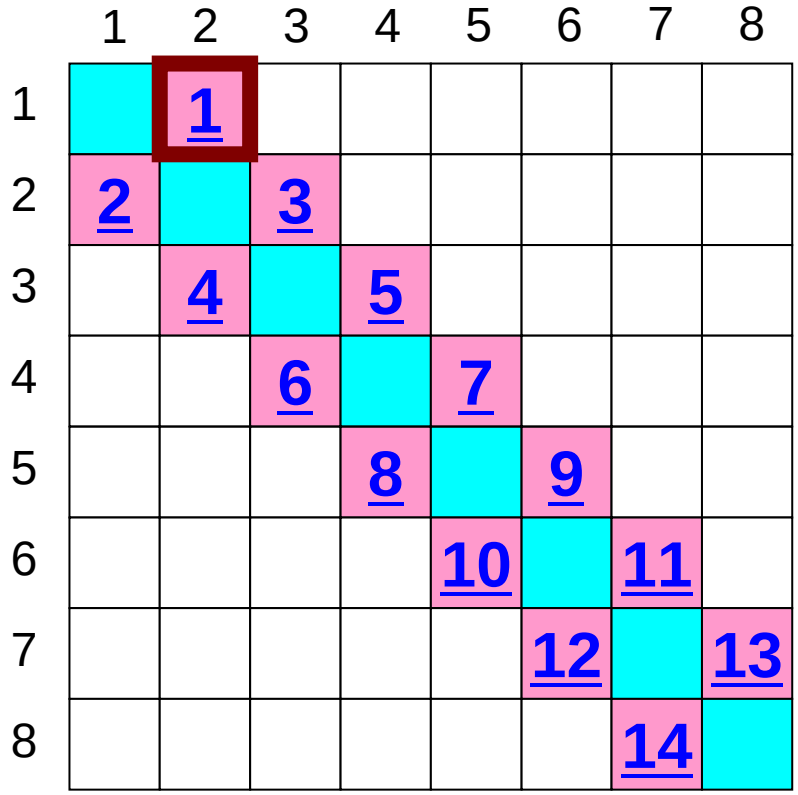

```
INDEX(2)=3, INDEX(3)=5
ITEM(4)=2, AMAT(4)= A_L(3)
ITEM(5)=4, AMAT(5)= A_R(3)
```

係数行列の格納形式(4/8)
非対角成分: i=1

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

INDEX(i) 非対角成分数に関する一次元配列
(整数, i=0,N)
ITEM(k) 非対角成分の要素番号
(整数, k=1, INDEX(N))
AMAT(k) 非対角成分
(実数, k=1, INDEX(N))

INDEX(0)=0
INDEX(1)=1
ITEM(1)=2, AMAT(1)= $A_R(1)$

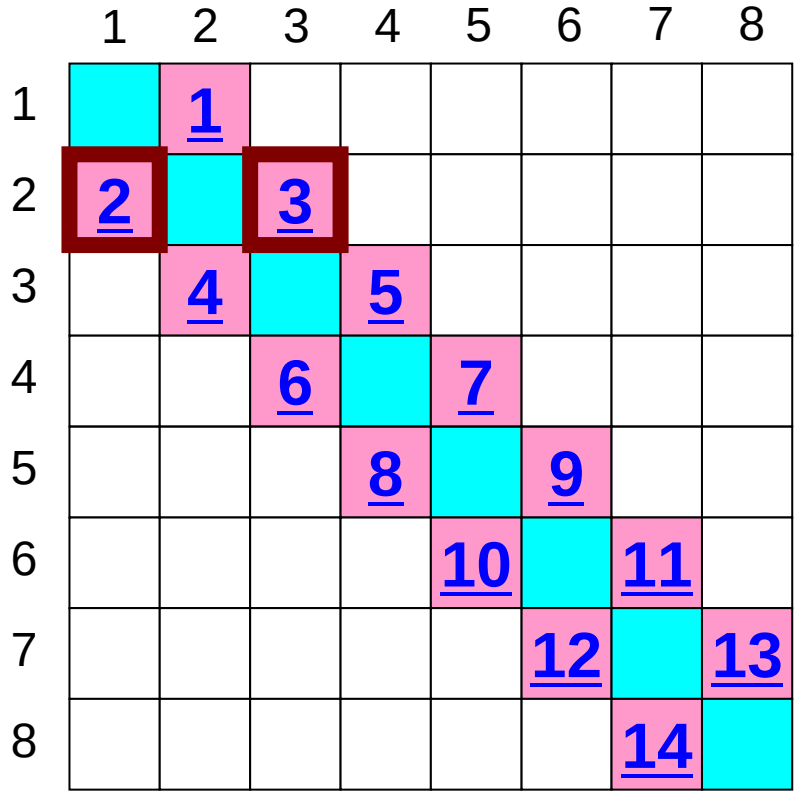


係数行列の格納形式(5/8)
非対角成分:i=2

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

INDEX(i) 非対角成分数に関する一次元配列
(整数, i=0,N)
ITEM(k) 非対角成分の要素番号
(整数, k=1, INDEX(N))
AMAT(k) 非対角成分
(実数, k=1, INDEX(N))

INDEX(1)=1
INDEX(2)=3
ITEM(2)=1, AMAT(2)= $A_L(2)$
ITEM(3)=3, AMAT(3)= $A_R(2)$

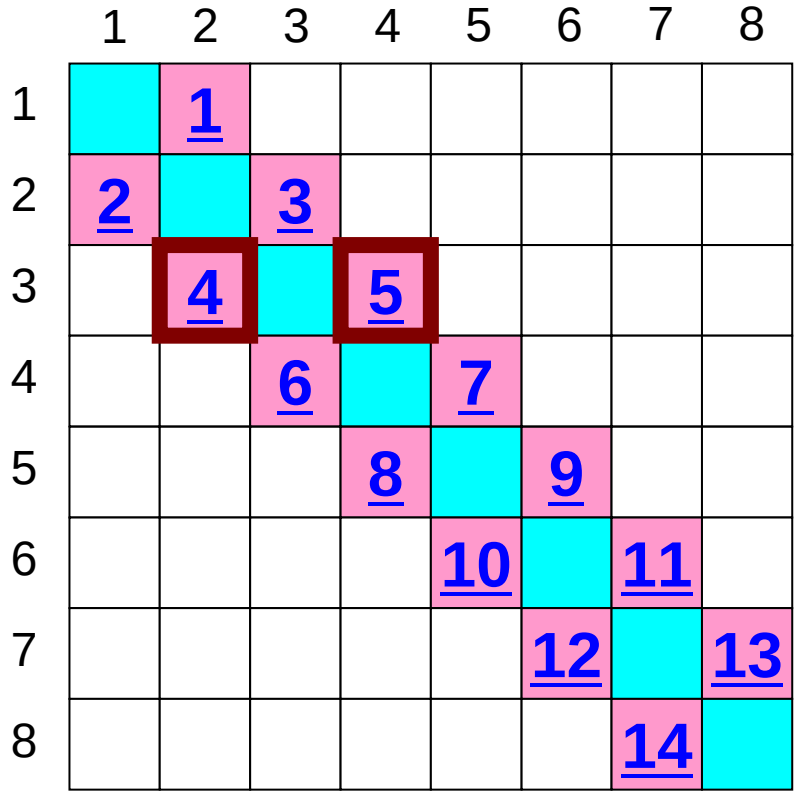


係数行列の格納形式(6/8)
非対角成分:i=3

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

INDEX(i) 非対角成分数に関する一次元配列
(整数, i=0,N)
ITEM(k) 非対角成分の要素番号
(整数, k=1, INDEX(N))
AMAT(k) 非対角成分
(実数, k=1, INDEX(N))

INDEX(2)=3
INDEX(3)=5
ITEM(4)=2, AMAT(4)= $A_L(3)$
ITEM(5)=4, AMAT(5)= $A_R(3)$

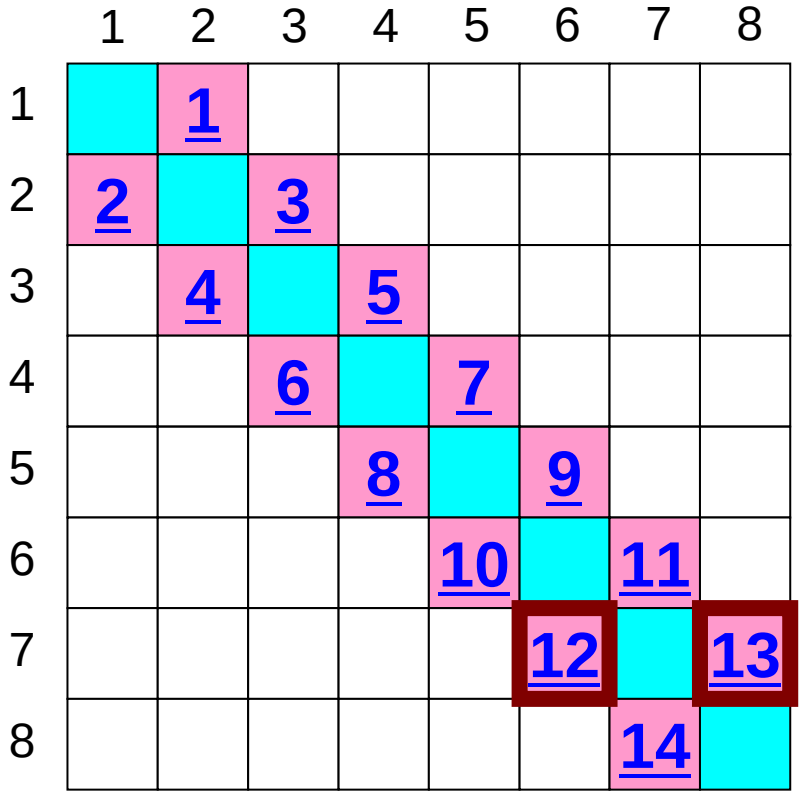


係数行列の格納形式(7/8)
非対角成分:i=7

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

INDEX(i) 非対角成分数に関する一次元配列
(整数, i=0,N)
ITEM(k) 非対角成分の要素番号
(整数, k=1, INDEX(N))
AMAT(k) 非対角成分
(実数, k=1, INDEX(N))

INDEX(6)=11
INDEX(7)=13
ITEM(12)=6, AMAT(12)= $A_L(7)$
ITEM(13)=8, AMAT(13)= $A_R(7)$

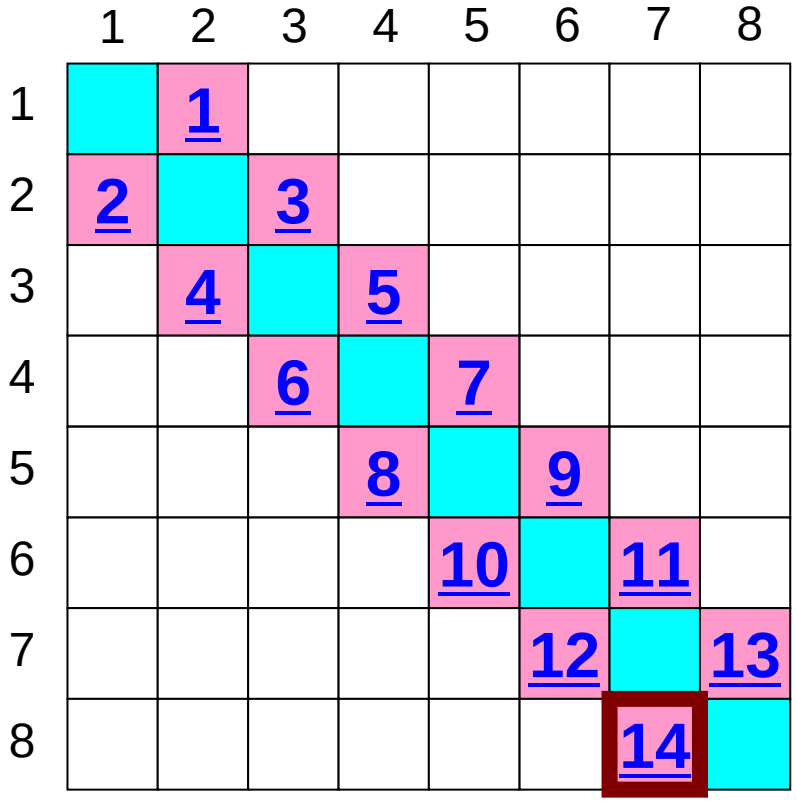


係数行列の格納形式(8/8)
非対角成分:i=8

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad (i=1,\ldots,N)$$

INDEX(i) 非対角成分数に関する一次元配列
(整数, i=0,N)
ITEM(k) 非対角成分の要素番号
(整数, k=1, INDEX(N))
AMAT(k) 非対角成分
(実数, k=1, INDEX(N))

INDEX(7)=13
INDEX(8)=14
ITEM(14)=7, AMAT(14)= $A_L(8)$



課題S3：プログラム実行法

- 実行法

```
$ cd <$S3>
$ pgf90 -O3 heat_cg.f
$ ./a.out
```

```
$ cd <$S3>
$ pgcc -O3 heat_cg.c
$ ./a.out
```

input.dat

```
100              N              メッシュ数
1.d0 1.d0        dx, BF         メッシュ幅，体積発熱量
5000             ITERmax        最大反復回数
1.d-7  1.95d0    EPS, OMEGA     打切誤差，　SORのω
```

CG法による一次元熱伝導方程式計算プログラム

- 前処理付き共役勾配法
  - Preconditioned Conjugate Gradient Method
  - 対称正定行列用
- 対角スケーリング（Diagonal Scaling）
  - [A]の対角成分のみからなる行列を前処理行列[M]とする。
  - 簡単（な問題にしか適用できない）。
  - 点ヤコビ（Point Jacobi）前処理とも呼ばれる。

```
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)
    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
```

対角スケーリング，点ヤコビ前処理

- 前処理行列として，もとの行列の対角成分のみを取り出した行列を前処理行列 [M] とする。
  - 対角スケーリング，点ヤコビ（point-Jacobi）前処理

$$[M]=\begin{bmatrix} D_1 & 0 & \dots & 0 & 0 \\ 0 & D_2 & & 0 & 0 \\ \dots & & \dots & & \dots \\ 0 & 0 & & D_{N-1} & 0 \\ 0 & 0 & \dots & 0 & D_N \end{bmatrix}$$

- **solve [M]$z^{(i-1)}$= $r^{(i-1)}$**という場合に逆行列を簡単に求めることができる。

CG法による一次元熱伝導方程式 プログラム概要(1/7)

```
!C
!C   1D Poisson Equation Solver by
!C   CG (Conjugate Gradient) Method
!C
!C   d/dx(dPHI/dx) + BF = 0
!C   PHI=0@x=0
!C
      program CG_poi
      implicit REAL*8 (A-H,O-Z)

      integer :: N, ITERmax
      integer :: R, Z, P, Q, DD

      real(kind=8) :: dx, OMEGA, RESID, dPHI, dPHImax, BF, EPS
      real(kind=8), dimension(:), allocatable :: PHI, RHS
      real(kind=8), dimension(:  ), allocatable :: DIAG, AMAT
      real(kind=8), dimension(:,:), allocatable :: W

      integer, dimension(:  ), allocatable :: INDEX, ITEM

!C
!C-- INIT.
      open  (11, file='input.dat', status='unknown')
       read (11,*) N
       read (11,*) dX, BF
       read (11,*) ITERmax
       read (11,*) EPS
      close (11)
```

CG法による一次元熱伝導方程式 プログラム概要(2/7)

```
allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))
 PHI= 0.d0
AMAT=  1.d0/dX
DIAG= -2.d0/dX
 RHS=  -BF * dX
```

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

$$\frac{d^2\phi}{dx^2} + BF = 0, \quad \phi = 0 @ x = 0, \frac{d\phi}{dx} = 0 @ x = x_{\max}$$

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times V + BF \times V = 0$$

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

したがって，非対角成分の総数は「2*N-2」

CG法による一次元熱伝導方程式 プログラム概要(2/7)

```
      allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
      allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))
       PHI= 0.d0
      AMAT=  1.d0/dX
      DIAG= -2.d0/dX
       RHS=  -BF * dX
```

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times V + BF \times V = 0$$

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times \Delta x + BF \times \Delta x = 0$$

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

したがって，非対角成分の総数は「2*N-2」

CG法による一次元熱伝導方程式 プログラム概要(2/7)

```
allocate (PHI(N), DIAG(N), AMAT(2*N-2), RHS(N))
allocate (INDEX(0:N), ITEM(2*N-2), W(N,4))
 PHI= 0.d0
AMAT=  1.d0/dX
DIAG= -2.d0/dX
 RHS=  -BF * dX
```

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

$$\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x^2} \right) \times \Delta x + BF \times \Delta x = 0$$

$$\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x} = \left(\frac{1}{\Delta x} \right) \phi_{i-1} - \left(\frac{2}{\Delta x} \right) \phi_i + \left(\frac{1}{\Delta x} \right) \phi_{i+1} = -BF \times \Delta x$$

$A_L(i)$ $A_D(i)$ $A_R(i)$ RHS(i)

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

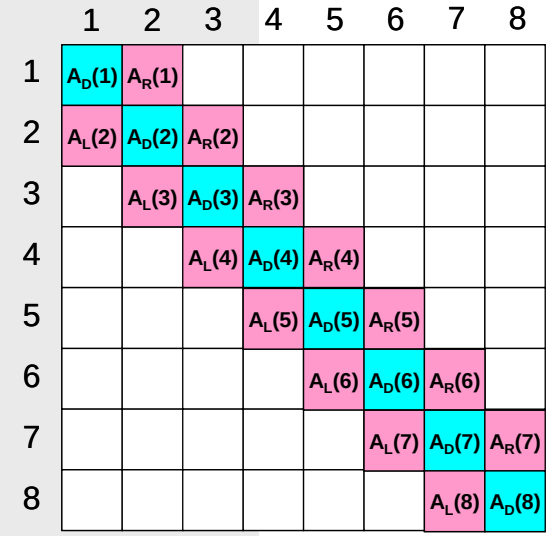
したがって，非対角成分の総数は「2*N-2」

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
      INDEX(0)= 0
      INDEX(1)= 1
      INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo

      do i= 1, N
        jS= INDEX(i-1)
        if (i.eq.1) then
          ITEM(jS+1)= i+1
          AMAT(jS+1)= 0.d0
          DIAG(i   )= 1.d0
           RHS(i   )= 0.d0
         else if
     &     (i.eq.N) then
          ITEM(jS+1)= i-1
          DIAG(i   )= -1.d0/dX
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
          if (i-1.eq.1) then
            AMAT(jS+1)= 0.d0
          endif
        endif
      enddo
```

非対角成分の数は「2」
ただし，i=1，i=Nのときは 「1」

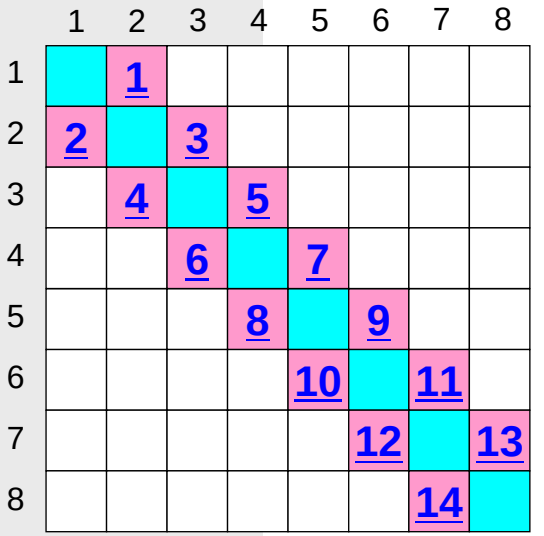
このルールに従って「INDEX」生成



&

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
      INDEX(0)= 0
      INDEX(1)= 1
      INDEX(N)= 1

      do i= 1, N
        INDEX(i)= INDEX(i) + INDEX(i-1)
      enddo

      do i= 1, N
        jS= INDEX(i-1)
        if (i.eq.1) then
          ITEM(jS+1)= i+1
          AMAT(jS+1)= 0.d0
          DIAG(i   )= 1.d0
           RHS(i   )= 0.d0
         else if
     &     (i.eq.N) then
          ITEM(jS+1)= i-1
          DIAG(i   )= -1.d0/dX
         else
          ITEM(jS+1)= i-1
          ITEM(jS+2)= i+1
          if (i-1.eq.1) then
            AMAT(jS+1)= 0.d0
          endif
        endif
      enddo
```




```
INDEX(0)=0
INDEX(1)=1
                ITEM(1)=2, AMAT(1)= AR1
INDEX(2)=3
                ITEM(2)=1, AMAT(2)= AL2
                ITEM(3)=3, AMAT(3)= AR3
INDEX(3)=5
                ITEM(4)=2, AMAT(4)= AL3
                ITEM(5)=4, AMAT(5)= AR3
…
INDEX(N-1)= 2*N-3
                ITEM(2*N-4)= N-2, AMAT(2*N-4)= AL(N-1)
                ITEM(2*N-3)= N  , AMAT(2*N-3)= AR(N-1)
INDEX(N)  = 2*N-2
                ITEM(2*N-2)= N-1, AMAT(N*N-2)= AL(N)
```

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
       INDEX(0)= 0
       INDEX(1)= 1
       INDEX(N)= 1

       do i= 1, N
         INDEX(i)= INDEX(i) + INDEX(i-1)
       enddo

       do i= 1, N
         jS= INDEX(i-1)
         if (i.eq.1) then
           ITEM(jS+1)= i+1
           AMAT(jS+1)= 0.d0
           DIAG(i   )= 1.d0
            RHS(i   )= 0.d0
          else if
     &      (i.eq.N) then
           ITEM(jS+1)= i-1
           DIAG(i   )= -1.d0/dX
          else
           ITEM(jS+1)= i-1
           ITEM(jS+2)= i+1
           if (i-1.eq.1) then
             AMAT(jS+1)= 0.d0
           endif
         endif
       enddo
```

境界条件(i=1):後述

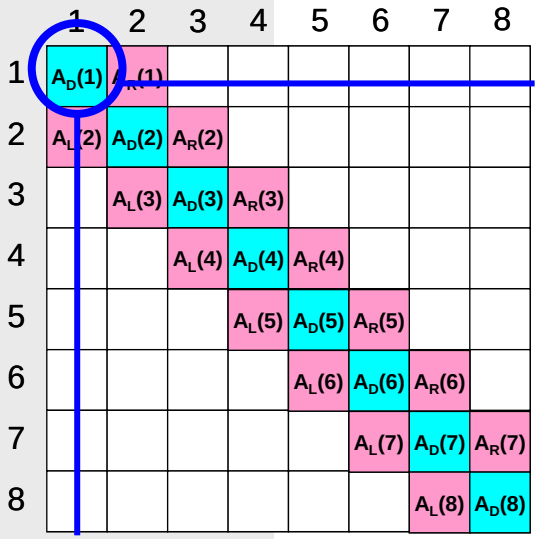
境界条件(i=N):S2のときと同じ

それ以外

	1	2	3	4	5	6	7	8
1	$A_D(1)$	$A_R(1)$						
2	$A_L(2)$	$A_D(2)$	$A_R(2)$					
3		$A_L(3)$	$A_D(3)$	$A_R(3)$				
4			$A_L(4)$	$A_D(4)$	$A_R(4)$			
5				$A_L(5)$	$A_D(5)$	$A_R(5)$		
6					$A_L(6)$	$A_D(6)$	$A_R(6)$	
7						$A_L(7)$	$A_D(7)$	$A_R(7)$
8							$A_L(8)$	$A_D(8)$

CG法による一次元熱伝導方程式 プログラム概要(3/7)

```
      INDEX= 2
!C
!C-- CONNECTIVITY
       INDEX(0)= 0
       INDEX(1)= 1
       INDEX(N)= 1

       do i= 1, N
         INDEX(i)= INDEX(i) + INDEX(i-1)
       enddo

       do i= 1, N
         jS= INDEX(i-1)
         if (i.eq.1) then
           ITEM(jS+1)= i+1
           AMAT(jS+1)= 0.d0
           DIAG(i   )= 1.d0
            RHS(i   )= 0.d0
          else if
     &      (i.eq.N) then
           ITEM(jS+1)= i-1
           DIAG(i   )= -1.d0/dX
          else
           ITEM(jS+1)= i-1
           ITEM(jS+2)= i+1
           if (i-1.eq.1) then
             AMAT(jS+1)= 0.d0
           endif
         endif
       enddo
```



境界条件(i=1):後述

境界条件(i=N):SORのときと同じ

それ以外

固定境界条件が指定されている点を
非対角成分として持っている場合
非対角成分をゼロクリアする。

境界条件の処理：i=N

```
i = N
jS= INDEX(i-1)

ITEM(jS+1)= i-1
AMAT(jS+1)= +1.d0/dx  デフォルト値
DIAG(i   )= -1.d0/dX
```

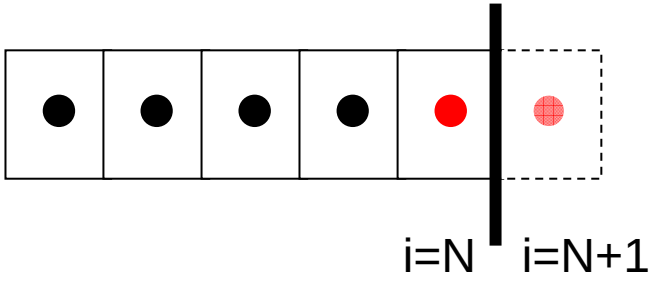
$$\frac{d\phi}{dx} = 0 @ x = x_{\max} \Rightarrow \frac{\phi_{N+1} - \phi_N}{\Delta x} = 0$$

$$\left(\frac{\phi_{N+1} - 2\phi_N + \phi_{N-1}}{\Delta x^2} \right) \times \Delta x + BF \times \Delta x = 0$$

$$\Rightarrow \left(\frac{-\phi_N + \phi_{N-1}}{\Delta x^2} \right) \times \Delta x + BF \times \Delta x = 0$$

$$\Rightarrow (0)\phi_{N+1} + \left(\frac{-1}{\Delta x} \right) \phi_N + \left(\frac{1}{\Delta x} \right) \phi_{N-1} = -BF \times \Delta x$$

DIAG(i)　AMAT(jS+1)　RHS



境界面で断熱条件が成立するためには，$\phi_{N+1}=\phi_N$を満たすような仮想的な要素があると都合が良い

境界条件の処理：i=1（1/4）

```
ITEM(jS+1)= i+1
AMAT(jS+1)= 0.d0
DIAG(i   )= 1.d0
RHS (i   )= 0.d0
```

$$\phi = 0 @ x = 0 \Rightarrow \phi_1 = 0$$

$$\Rightarrow (0)\phi_2 + (1)\phi_1 + (0)\phi_0 = 0$$

DIAG **RHS**

	1	2	3	4	5	6	7	8
1	1	0						
2	a	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

$$a = \frac{1}{\Delta x}$$

- 課題S2ではこのようにしていた
- これでは係数行列が対称とならないため共役勾配法が使えない。

境界条件の処理：i=1（2/4）

```
ITEM(jS+1)= i+1
AMAT(jS+1)= 0.d0
DIAG(i   )= 1.d0
RHS (i   )= 0.d0
```

$$\phi = 0 @ x = 0 \Rightarrow \phi_1 = 0$$

$$\Rightarrow (0)\phi_2 + (1)\phi_1 + (0)\phi_0 = 0$$

DIAG **RHS**

	1	2	3	4	5	6	7	8
1	1	0						
2	a	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

$$a = \frac{1}{\Delta x}$$

- 課題S2ではこのようにしていた
- これでは係数行列が対称とならないため共役勾配法が使えない。

境界条件の処理：i=1（3/4）

- i=2における方程式

$$(a)\phi_3 + (-2a)\phi_2 + (a)\phi_1 = RHS_2$$

- ここでϕ_1は既知の値 $\bar{\phi}_1$

	1	2	3	4	5	6	7	8
1	1	0						
2	a	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

$$a = \frac{1}{\Delta x}$$

境界条件の処理：i=1（4/4）

- i=2における方程式

$$(a)\phi_3 + (-2a)\phi_2 + (a)\phi_1 = RHS_2$$

- ここでϕ_1は既知の値 $\bar{\phi}_1$

$$(a)\phi_3 + (-2a)\phi_2 + (0)\phi_1 = RHS_2 - (a)\bar{\phi}_1$$

- このケースの場合は，$\bar{\phi}_1 = 0$であるため，右辺（RHS）の修正は不要である。

	1	2	3	4	5	6	7	8
1	1	0						
2	0	-2a	a					
3		a	-2a	a				
4			a	-2a	a			
5				a	-2a	a		
6					a	-2a	a	
7						a	-2a	a
8							a	-a

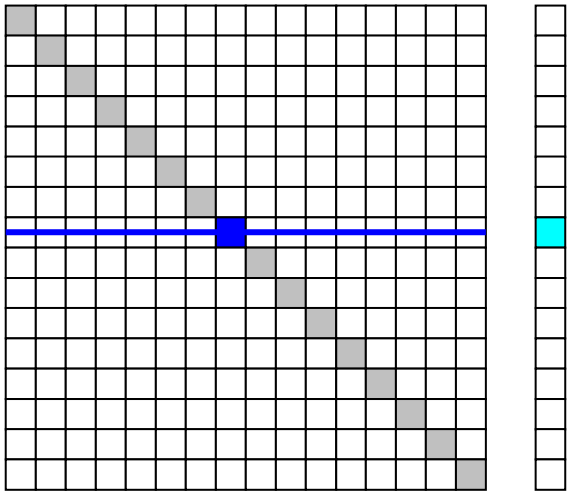
$$a = \frac{1}{\Delta x}$$

固定境界条件の処理：消去

i=kkの点でPHIxの値に固定されている場合

```
do i= 1, N
  if (i.eq.kk) then
    DIAG(i)= 1.d0
    RHS (i)= PHIx
    do j= INDEX(i-1)+1, INDEX(i)
      AMAT(j)= 0.d0
    enddo
  endif
endif

do i= 1, N
  if (i.ne.kk) then
    do j= INDEX(i-1)+1, INDEX(i)
      jj= ITEM(j)
      if (jj.eq.kk) then
        RHS (i)= RHS(i) - AMAT(j)*PHIx
        AMAT(j)= 0.d0
      endif
    enddo
  enddo
enddo
```

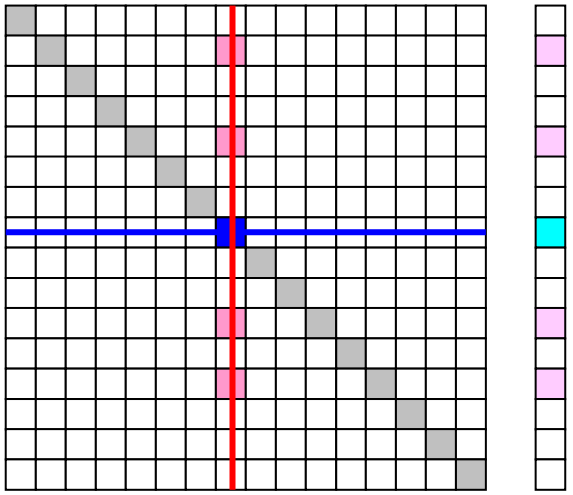
ゼロクリア



固定境界条件の処理：消去
i=kkの点でPHIxの値に固定されている場合

```
      do i= 1, N
        if (i.eq.kk) then
          DIAG(i)= 1.d0
          RHS (i)= PHIx
          do j= INDEX(i-1)+1, INDEX(i)
            AMAT(j)= 0.d0
          enddo
        endif
      endif

      do i= 1, N
        if (i.ne.kk) then
          do j= INDEX(i-1)+1, INDEX(i)
            jj= ITEM(j)
            if (jj.eq.kk) then
              RHS (i)= RHS(i) - AMAT(j)*PHIx
              AMAT(j)= 0.d0
            endif
          enddo
        enddo
      enddo
```

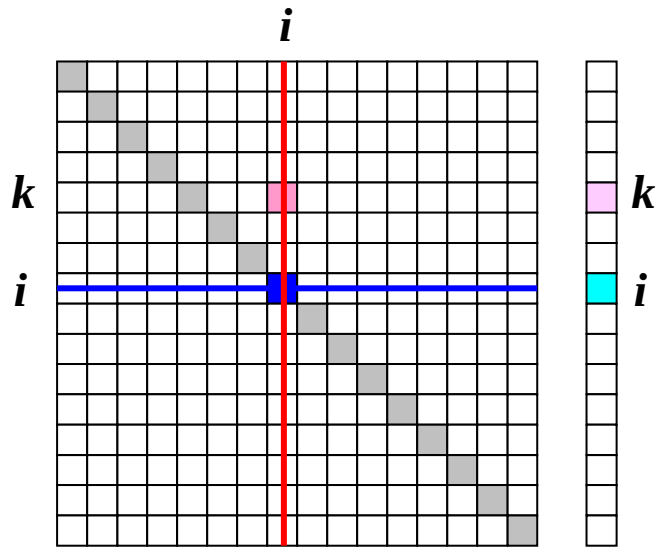
ゼロクリア

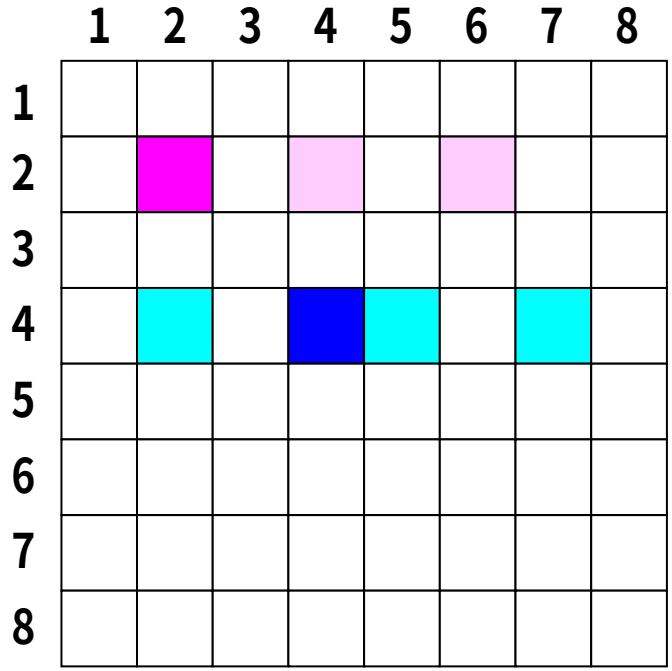


今回の問題ではPHIX(jj)に相当するものが0だったのでこの右辺の処理は不要であった。

固定境界条件の処理:具体例

点「i」で固定境界条件を適用しているものとする $\phi = \tilde{\phi}_i$





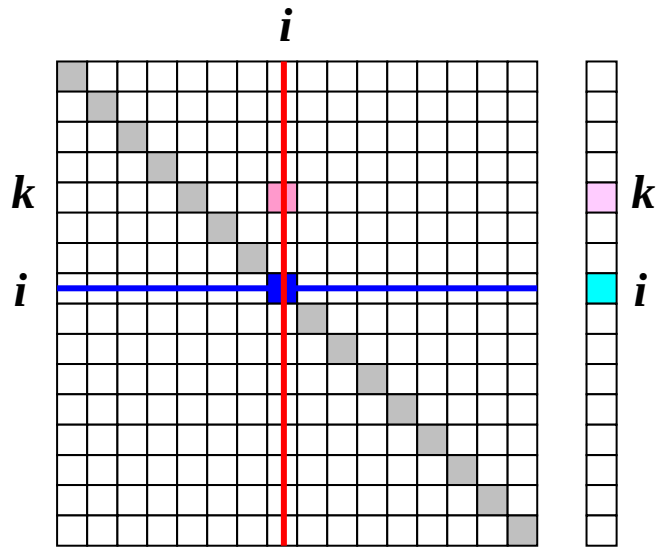
$i=4, k=2$ とすると

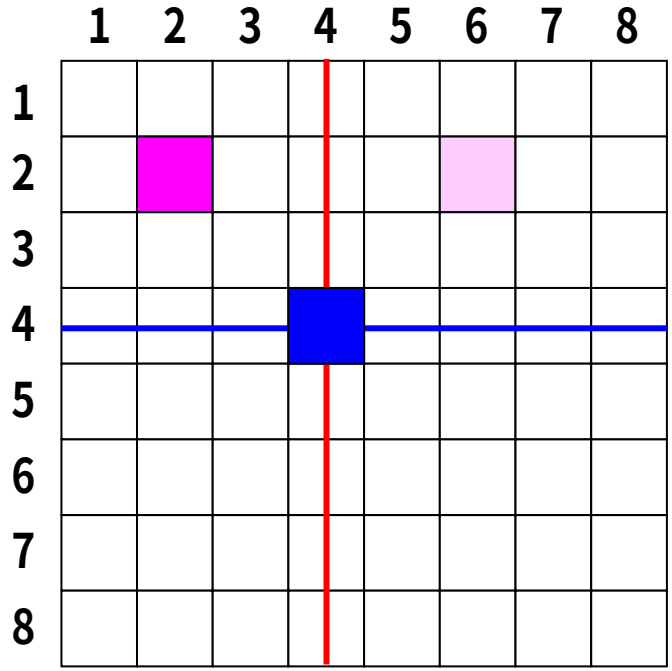
$$DIAG_2\phi_2 + AMAT_{24}\phi_4 + AMAT_{26}\phi_6 = RHS_2$$

$$DIAG_4\phi_4 + AMAT_{42}\phi_2 + AMAT_{45}\phi_5 + AMAT_{47}\phi_7 = RHS_4$$

固定境界条件の処理:具体例

点「i」で固定境界条件を適用しているものとする $\phi = \tilde{\phi}_i$





$i=4, k=2$ とすると

$$DIAG_2\phi_2 + AMAT_{24}\phi_4 + AMAT_{26}\phi_6 = RHS_2$$

$$DIAG_4\phi_4 + AMAT_{42}\phi_2 + AMAT_{45}\phi_5 + AMAT_{47}\phi_7 = RHS_4$$

$$DIAG_2\phi_2 + AMA_{26}\phi_6 = RHS_2 - AMAT_{24}\tilde{\phi}_4$$

$$\phi_4 = \tilde{\phi}_4$$

CG法による一次元熱伝導方程式 プログラム概要(4/7)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
!C
!C-- {r0}= {b} - [A]{xini} |

      do i= 1, N
        W(i,R) = DIAG(i)*PHI(i)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,R) = W(i,R) + AMAT(j)*PHI(ITEM(j))
        enddo
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2  + RHS(i)  **2
        W(i,R)= RHS(i) - W(i,R)
      enddo
!C*****************************************************************
```

対角成分の逆数(前処理用)
その都度，除算をすると効率が悪いため，予め配列に格納

CG法による一次元熱伝導方程式 プログラム概要(4/7)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
```

W(i,1)= W(i,R) ⇒ {r}
W(i,2)= W(i,Z) ⇒ {z}
W(i,2)= W(i,Q) ⇒ {q}
W(i,3)= W(i,P) ⇒ {p}

W(i,4)= W(i,DD) ⇒ 1/DIAG

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

CG法による一次元熱伝導方程式 プログラム概要(4/7)

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4

      do i= 1, N
        W(i,DD)= 1.0D0 / DIAG(i)
      enddo
!C
!C-- {r0}= {b} - [A]{xini} |

      do i= 1, N
        W(i,R) = DIAG(i)*PHI(i)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,R) = W(i,R) + AMAT(j)*PHI(ITEM(j))
        enddo
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2  + RHS(i)  **2
        W(i,R)= RHS(i) - W(i,R)
      enddo
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

BNRM2： 収束判定用

CG法による一次元熱伝導方程式 プログラム概要(5/7)

```
      do iter= 1, ITERmax
!C
!C-- {z}= [Minv]{r}

      do i= 1, N
        W(i,Z)= W(i,DD) * W(i,R)
      enddo

!C
!C-- RHO= {r}{z}

      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo

!C
!C-- {p} = {z} if       ITER=1
!C   BETA= RHO / RHO1  otherwise

      if ( iter.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
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

CG法による一次元熱伝導方程式 プログラム概要(5/7)

```
      do iter= 1, ITERmax
!C
!C-- {z}= [Minv]{r}

      do i= 1, N
        W(i,Z)= W(i,DD) * W(i,R)
      enddo

!C
!C-- RHO= {r}{z}

      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo

!C
!C-- {p} = {z} if       ITER=1
!C   BETA= RHO / RHO1  otherwise

      if ( iter.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
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

CG法による一次元熱伝導方程式 プログラム概要(5/7)

```
      do iter= 1, ITERmax
!C
!C-- {z}= [Minv]{r}

      do i= 1, N
        W(i,Z)= W(i,DD) * W(i,R)
      enddo

!C
!C-- RHO= {r}{z}

      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo

!C
!C-- {p} = {z} if       ITER=1
!C   BETA= RHO / RHO1  otherwise

      if ( iter.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
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

CG法による一次元熱伝導方程式 プログラム概要(6/7)

```
!C
!C-- {q}= [A]{p}

      do i= 1, N
        W(i,Q) = DIAG(i)*W(i,P)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
        enddo
      enddo

!C
!C-- ALPHA= RHO / {p}{q}

      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1

!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}

      do i= 1, N
        PHI(i)   = PHI(i) + ALPHA * W(i,P)
        W  (i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

CG法による一次元熱伝導方程式 プログラム概要(6/7)

```
!C
!C-- {q}= [A]{p}

      do i= 1, N
        W(i,Q) = DIAG(i)*W(i,P)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
        enddo
      enddo

!C
!C-- ALPHA= RHO / {p}{q}

      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1

!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}

      do i= 1, N
        PHI(i)   = PHI(i) + ALPHA * W(i,P)
        W  (i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

CG法による一次元熱伝導方程式 プログラム概要(6/7)

```
!C
!C-- {q}= [A]{p}

      do i= 1, N
        W(i,Q) = DIAG(i)*W(i,P)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
        enddo
      enddo

!C
!C-- ALPHA= RHO / {p}{q}

      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1

!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}

      do i= 1, N
        PHI(i)   = PHI(i) + ALPHA * W(i,P)
        W  (i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $\mathbf{x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}}$
  $\mathbf{r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}}$
  check convergence $|r|$
end

CG法による一次元熱伝導方程式 プログラム概要(7/7)

BNRM2： 無次元化に使用

```
      DNRM2 = 0.0
      do i= 1, N
        DNRM2= DNRM2 + W(i,R)**2
      enddo

        RESID= dsqrt(DNRM2/BNRM2)

        write (*, 1000) iter, RESID
 1000   format (i5, 1pe16.6)

        if ( RESID.le.EPS) goto 900
        RHO1 = RHO

      enddo
!C***********************************************************

      IER = 1

  900 continue
!C===

!C
!C-- OUTPUT
      do i= 1, N
        write (*,'(i8, 1pe16.6)') i, PHI(i)
      enddo

      end program CG_poi
```

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
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  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

課題S3：実施内容

- 「<$S3>/heat_cg.f」，「<$S3>/heat_cg.c」を参考にして，一次元熱伝導方程式をCG法により解くプログラムを並列化せよ。
  - 「一般化された通信テーブル（CS05, 1d-srb1/b2）」を使用せよ
- 実施課題
  - N=100およびN=103の場合について，1，2，4，8CPUを使用して計算してみよ。
    - 実は，N=100程度では並列化の効果は余り・・・全く得られない
  - プロセッサ数を変えた場合，反復回数はどうなるか？
    - その理由についても検討せよ
  - Nを大きくして（例えば10^4，10^5），反復回数を100程度に固定して，プロセッサ数を変更した場合について計算を実施してみよ。
- 締め切りは9月19日（水）17：00

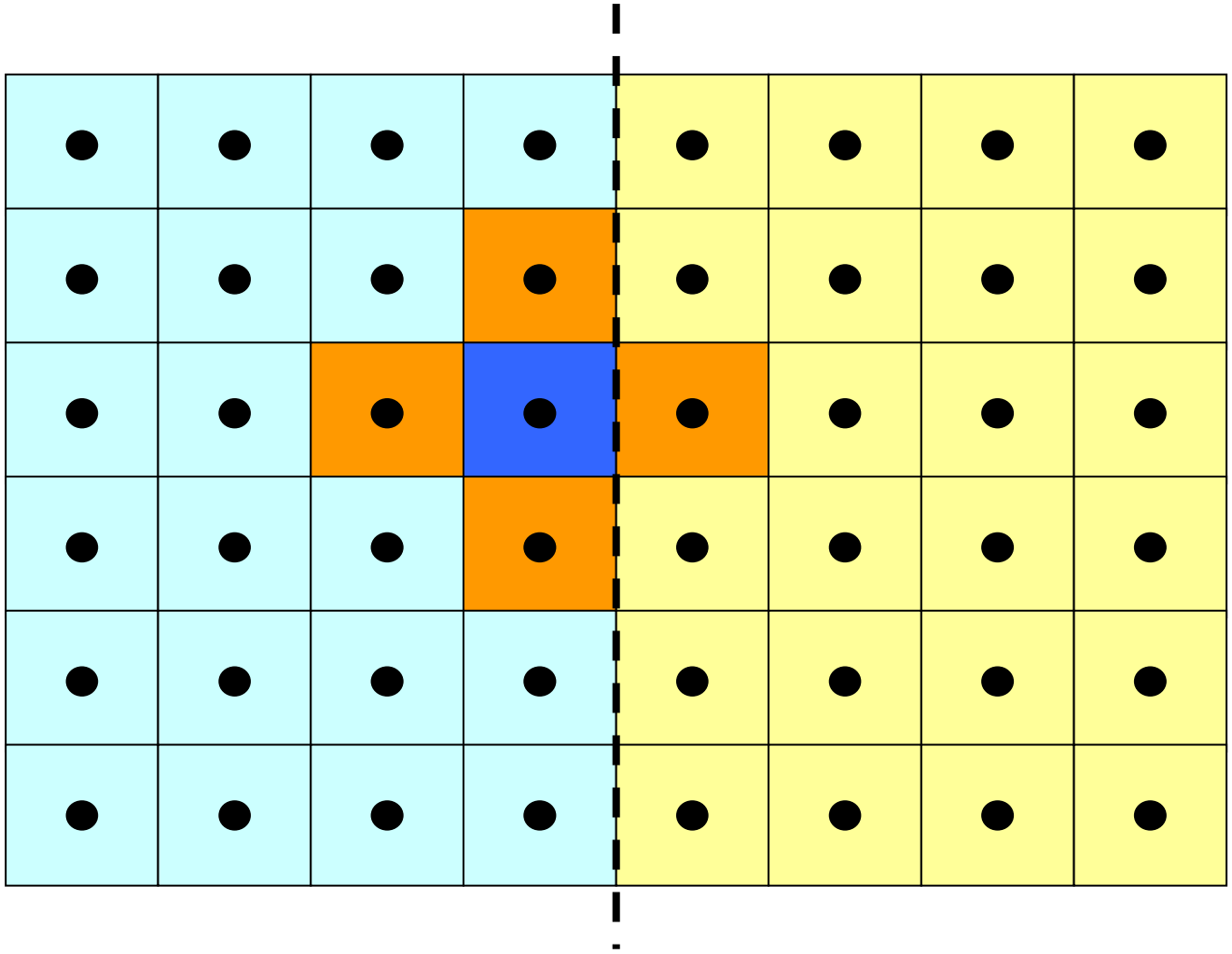
前処理付き共役勾配法
Preconditioned Conjugate Gradient Method（CG）

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

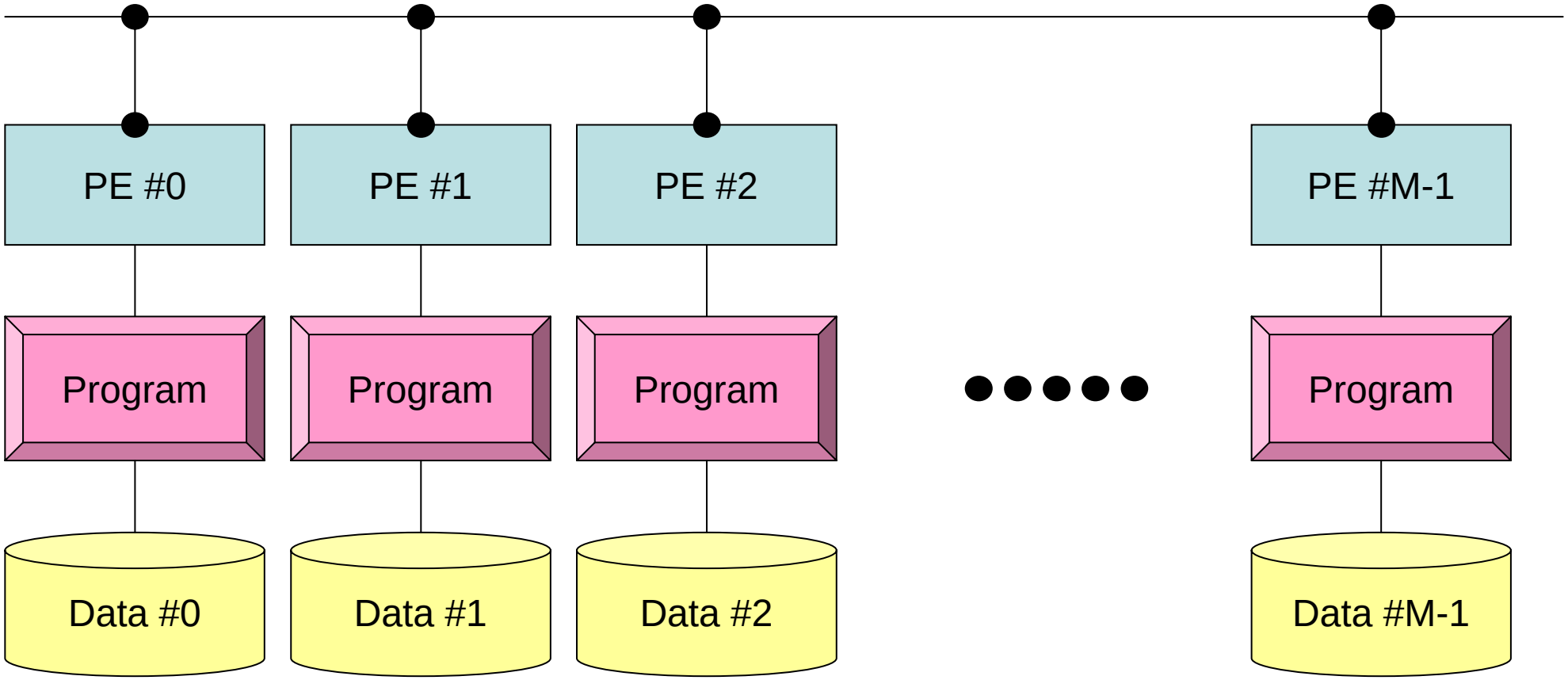
並列計算，領域間通信が必要な部分

- 行列ベクトル積
- 内積

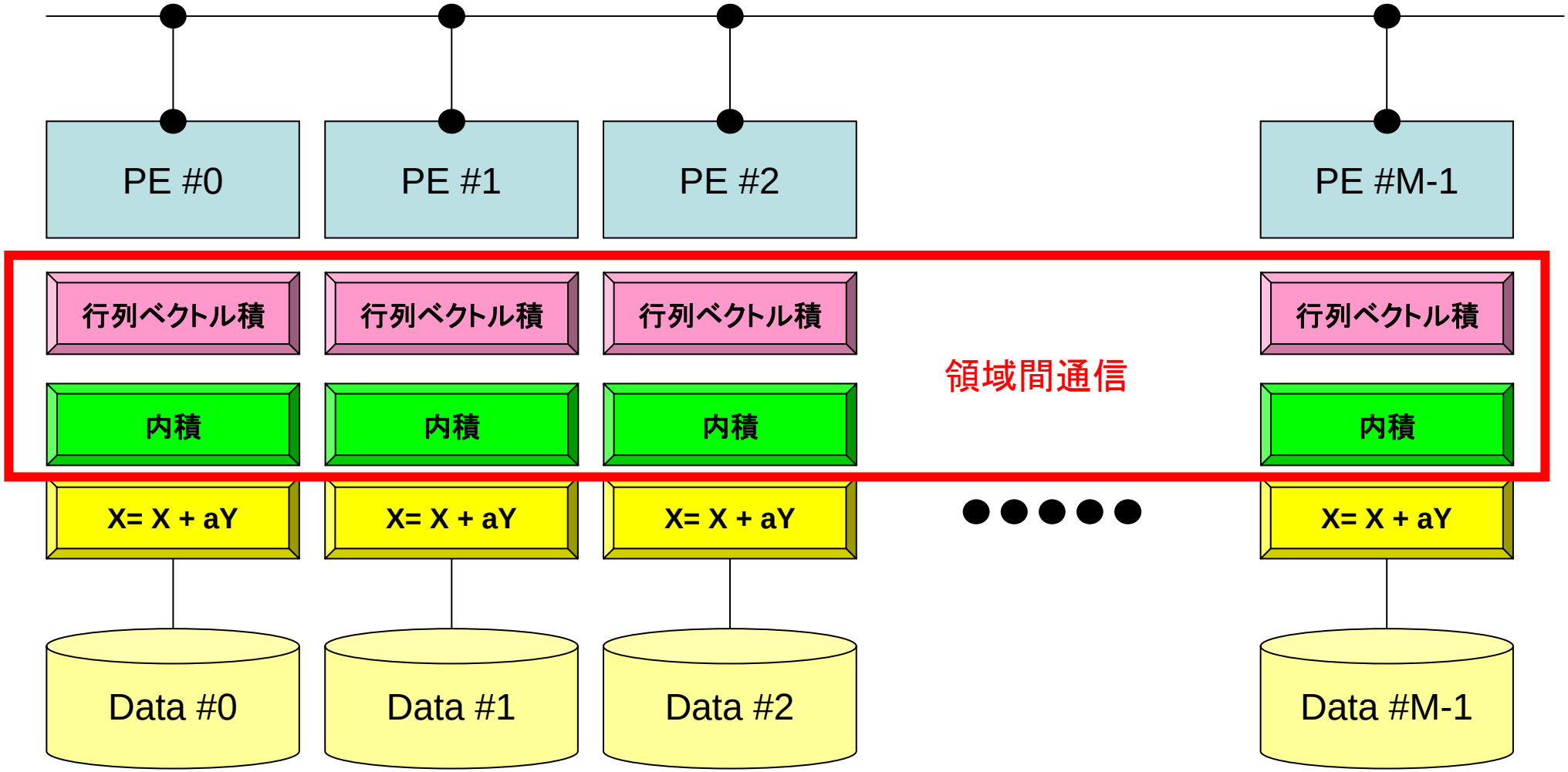
行列ベクトル積：他領域の値が必要
オーバーラップ＋通信テーブル



再びSPMD


PE #0
PE #1
PE #2
PE #M-1
Program
Program
Program
Program
Data #0
Data #1
Data #2
Data #M-1

CG法でははこうなる


PE #0
PE #1
PE #2
PE #M-1
行列ベクトル積
行列ベクトル積
行列ベクトル積
行列ベクトル積
領域間通信
内積
内積
内積
内積
X= X + aY
X= X + aY
X= X + aY
X= X + aY
Data #0
Data #1
Data #2
Data #M-1

共役勾配法の「並列化」(1/2)

- 行列ベクトル積
  - 課題**S2**で作成したような，領域境界データの交換を実施して，外点のベクトル値の最新値を得ておく。

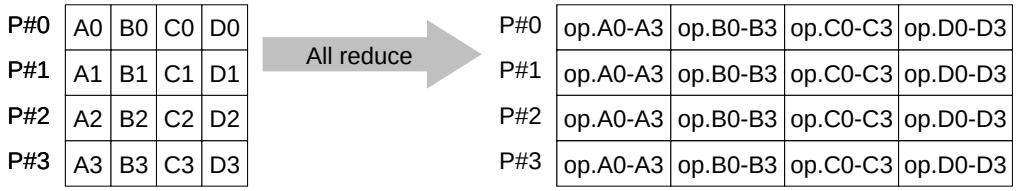
```
!C
!C-- {q}= [A]{p}

      exchange W(i,P)

      do i= 1, N
        W(i,Q) = DIAG(i)*W(i,P)
        do j= INDEX(i-1)+1, INDEX(i)
          W(i,Q) = W(i,Q) + AMAT(j)*W(ITEM(j),P)
        enddo
      enddo
```

共役勾配法の「並列化」(2/2)

- 内積
  - MPI_ALLREDUCE

```
!C
!C +--------------+
!C | RHO= {r}{z} |
!C +--------------+
!C===
      RHO= 0.d0

      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo

      allreduce RHO

!C===
```

MPI_ALLREDUCE



- MPI_REDUCE + MPI_BCAST
- 総和，最大値を計算したら，各プロセスで利用したい場合が多い

- **call MPI_ALLREDUCE**
  (sendbuf,recvbuf,count,datatype,op, comm,ierr)
  - **sendbuf** 任意 I 送信バッファの先頭アドレス，
  - **recvbuf** 任意 O 受信バッファの先頭アドレス，
    タイプは「datatype」により決定
  - **count** 整数 I メッセージのサイズ
  - **datatype** 整数 I メッセージのデータタイプ
  - **op** 整数 I 計算の種類
  - **comm** 整数 I コミュニケータを指定する
  - **ierr** 整数 O 完了コード

MPI_Reduce/Allreduceの “op”

```
call MPI_REDUCE
(sendbuf,recvbuf,count,datatype,op,root,comm,ierr)
```

- **MPI_MAX, MPI_MIN** 最大値，最小値
- **MPI_SUM, MPI_PROD** 総和，積
- **MPI_LAND** 論理AND

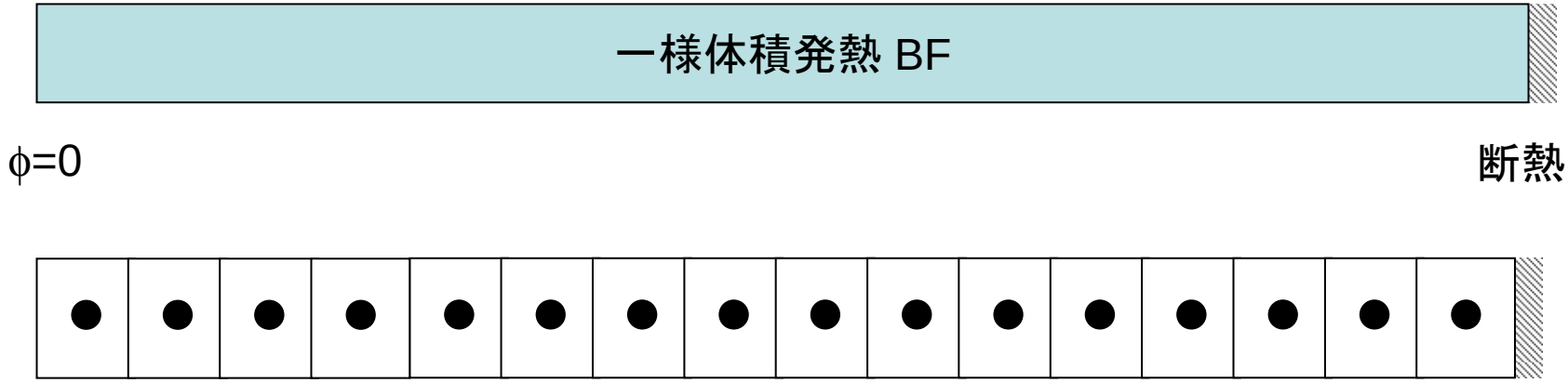
```
real(kind=8):: X0, X1

call MPI_REDUCE
(X0, X1, 1, MPI_DOUBLE_PRECISION, MPI_MAX, 0, <comm>, ierr)
```

```
real(kind=8):: X0(4), XMAX(4)

call MPI_REDUCE
(X0, XMAX, 4, MPI_DOUBLE_PRECISION, MPI_MAX, 0, <comm>, ierr)
```

一次元熱伝導方程式ソルバーの並列化
基本的に課題S2と全く同じアプローチ

$$\frac{d^2\phi}{dx^2} + BF = 0, \quad \phi = 0 @ x = 0, \frac{d\phi}{dx} = 0 @ x = x_{\max}$$

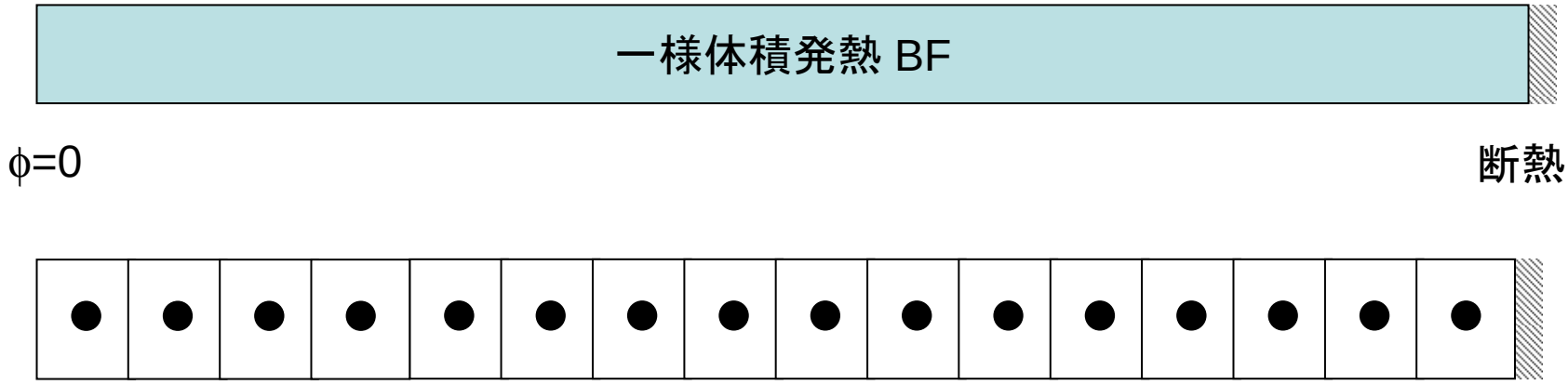
$$\phi = -\frac{1}{2} BF\, x^2 + BF\, x_{\max}\, x$$



復習

一次元熱伝導方程式ソルバーの並列化

- 全体データと局所データ
- 例えば，NG=16の問題を4PEで実行する場合，各PEにおいてはNL=16/4=4となる



復習

全体データと局所データ
NG=16, PE#=4

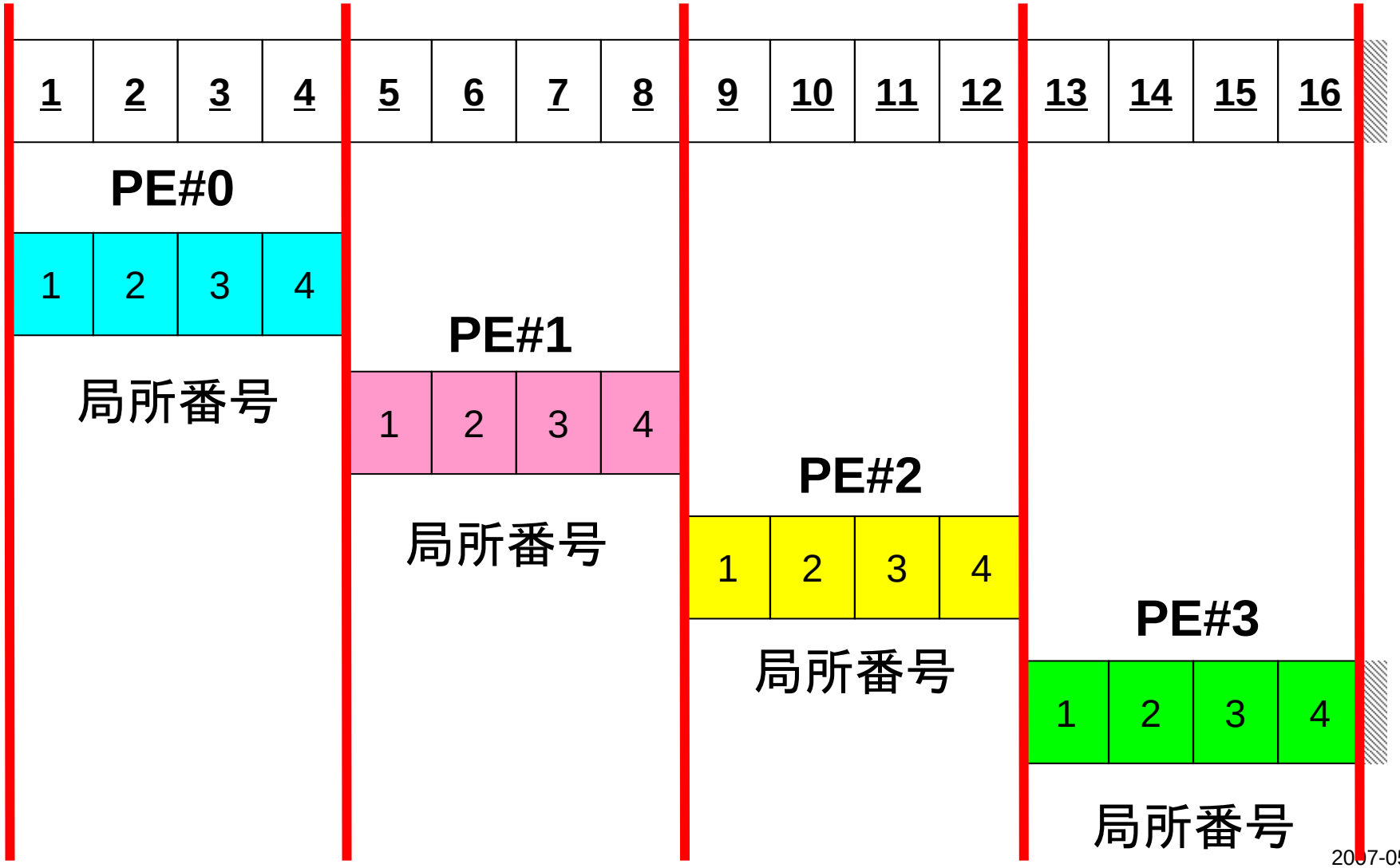
全体番号

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

復習

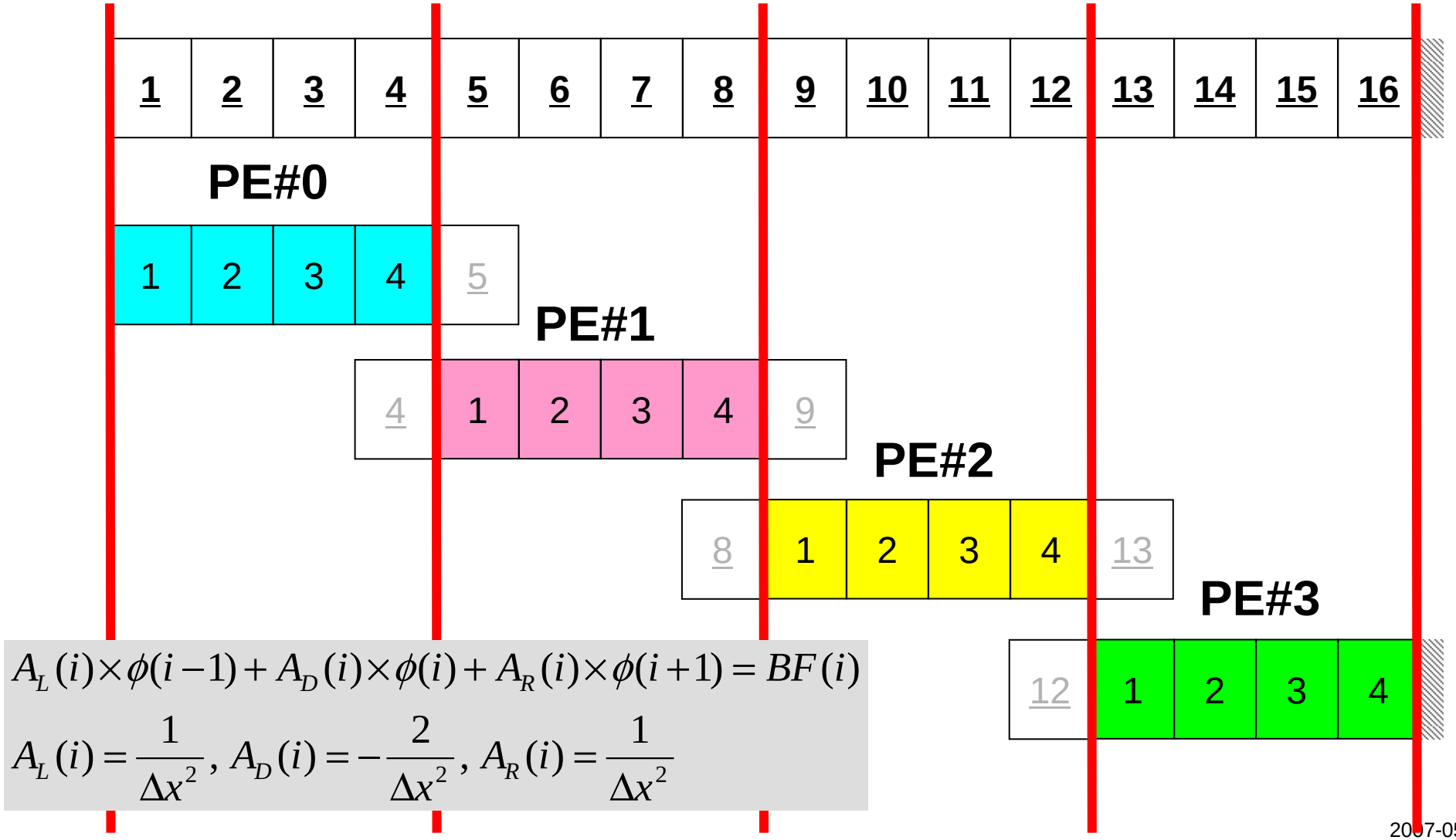
全体データと局所データ
NG=16, PE#=4

全体番号



差分法のオペレーションを実施するためには隣接要素の情報が必要

復習



$$A_L(i)\times\phi(i-1)+A_D(i)\times\phi(i)+A_R(i)\times\phi(i+1)=BF(i)$$

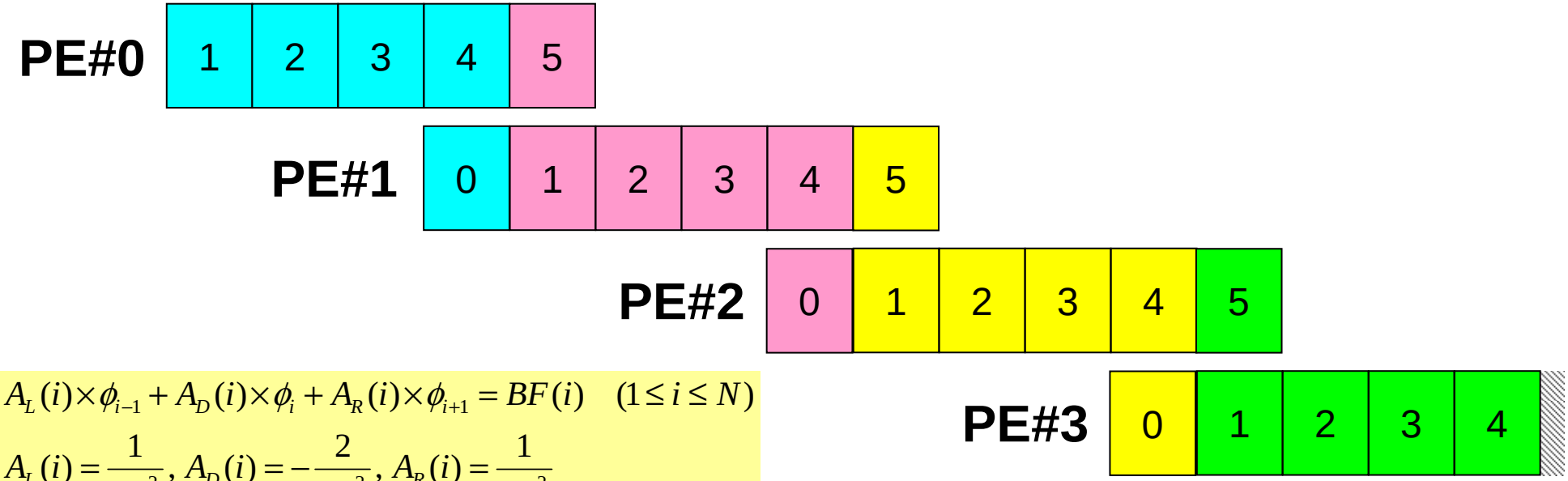
$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

一次元差分法モデル局所データ構造(3/5)

領域外の要素

復習

- 差分法(二次精度中央差分)の計算を並列に実施するためには，本来領域外の要素の影響を考慮する必要がある。
  - 領域間オーバーラップ
- 要素番号が0，N+1(=5)の要素を加えた，局所データ構造



$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

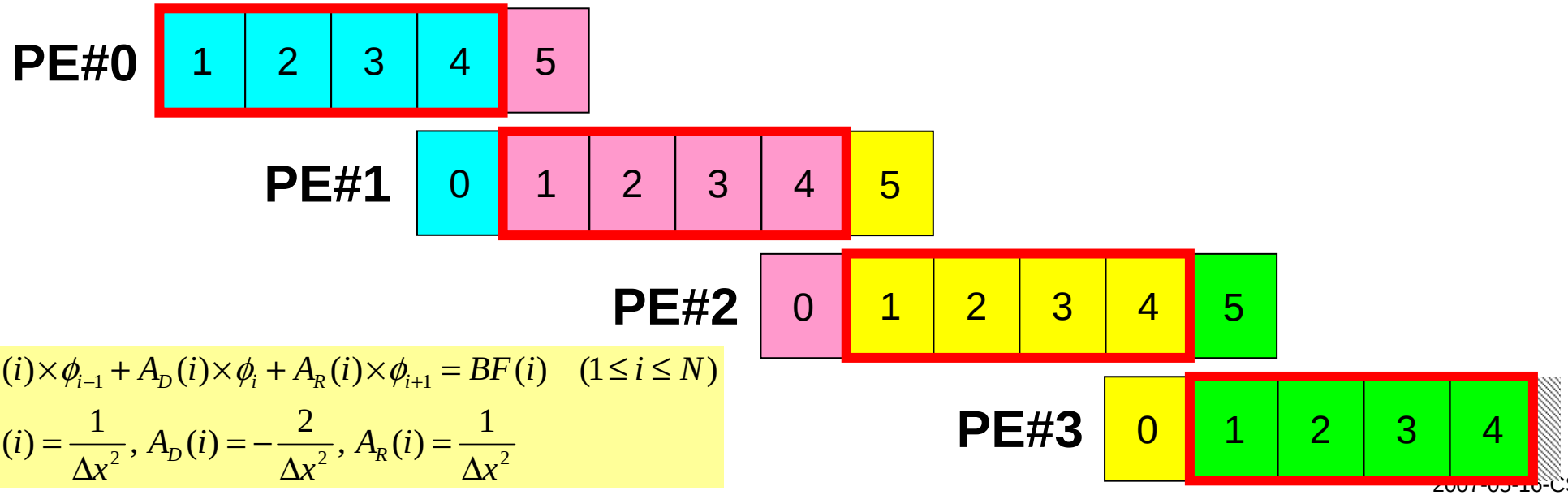
$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

一次元差分法モデル局所データ構造(4/5)

必要な情報

復習

- 領域内の要素，オーバーラップ(本来領域外)要素の区別
  - 領域内要素(i=1~N)　　:内点(Interior/Internal Points)



$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

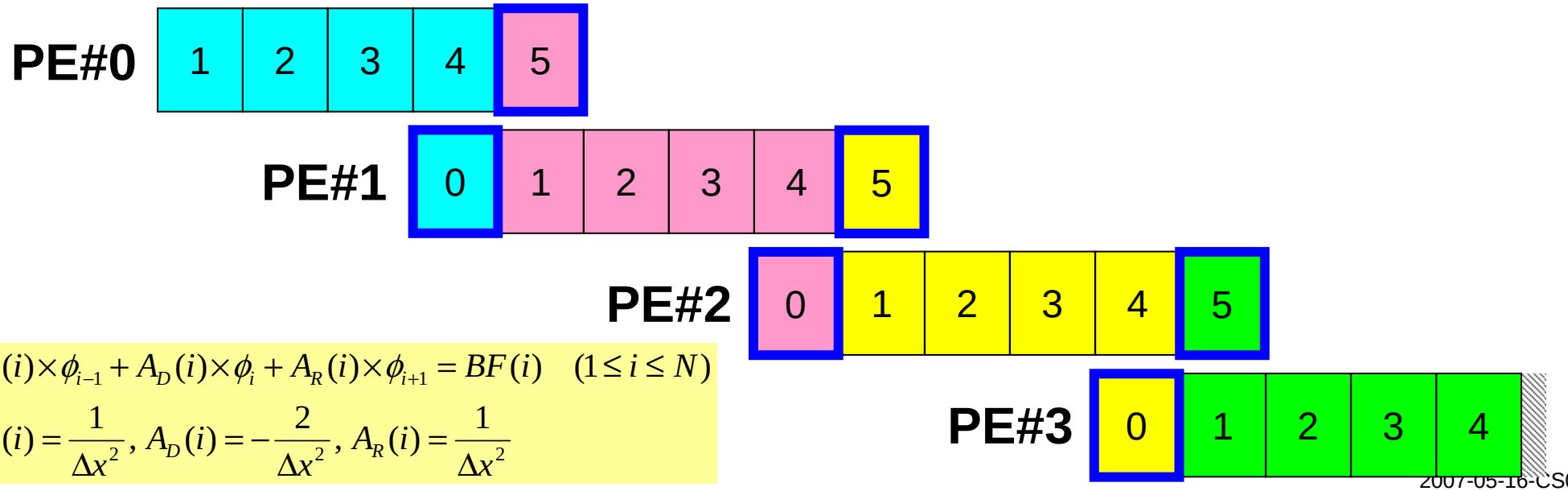
$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

一次元差分法モデル局所データ構造(4/5)

必要な情報

復習

- 領域内要素，オーバーラップ(本来領域外)要素の区別
  - 領域内要素(i=1~N)　　:内点(Interior/Internal Points)
  - オーバーラップ要素(i=0, N+1):外点(Exterior/External Points)
    - i=0，i=N+1の要素が存在しない場合もある



$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

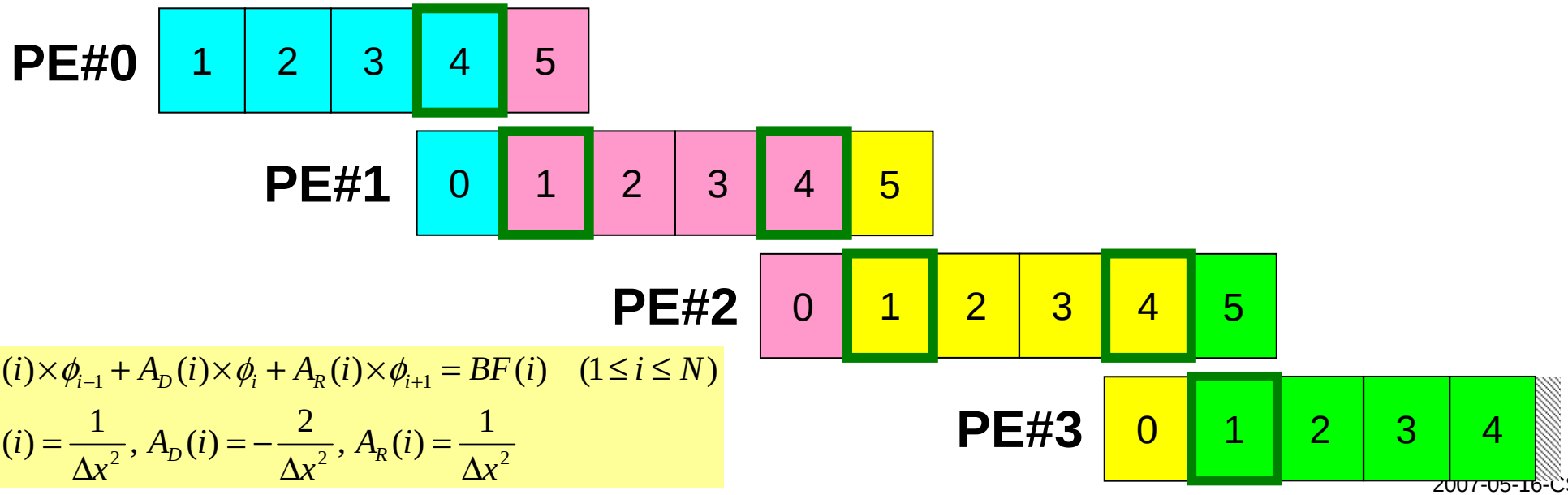
$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

一次元差分法モデル局所データ構造(4/5)

必要な情報

復習

- 領域内要素，オーバーラップ(本来領域外)要素の区別
  - 領域内要素(i=1~N)　　:内点(Interior/Internal Points)
  - オーバーラップ要素(i=0, N+1):外点(Exterior/External Points)
    - i=0，i=N+1の要素が存在しない場合もある
  - 「内点」のうち他領域の「外点」となっている要素:境界点(Boundary Points)



$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

$$A_L(i)=\frac{1}{\Delta x^2},\ A_D(i)=-\frac{2}{\Delta x^2},\ A_R(i)=\frac{1}{\Delta x^2}$$

一次元差分法モデル局所データ構造(5/5)

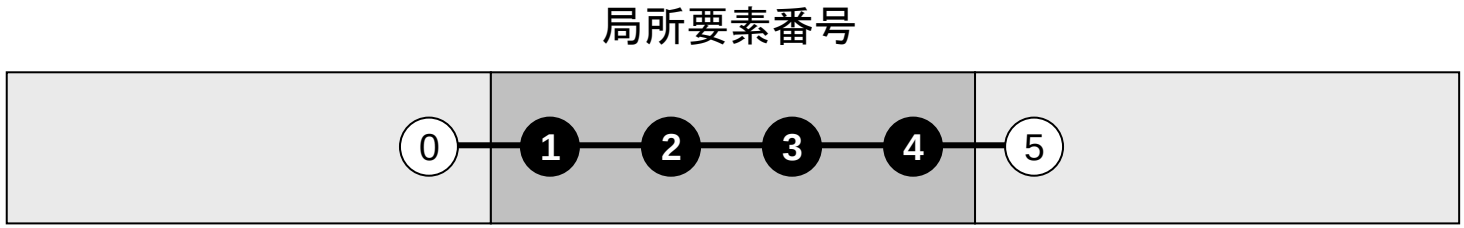
必要な情報(続き)

復習

- 「領域間オーバーラップ」を考慮した並列計算を実現するためには？
  - 各領域(プロセッサ)における「境界点」の情報を，各隣接領域に「外点」の情報として配信する必要がある。
  - そのための局所データ構造が必要
- 隣接している領域数，隣接している領域番号
  - 1対1通信の場合重要
  - オーバーラップ要素を共有している領域(プロセッサ)
- 隣接領域との「通信テーブル」
  - 境界点:隣接領域への「送信」(send)
  - 外点　:隣接領域からの「受信」(recv)

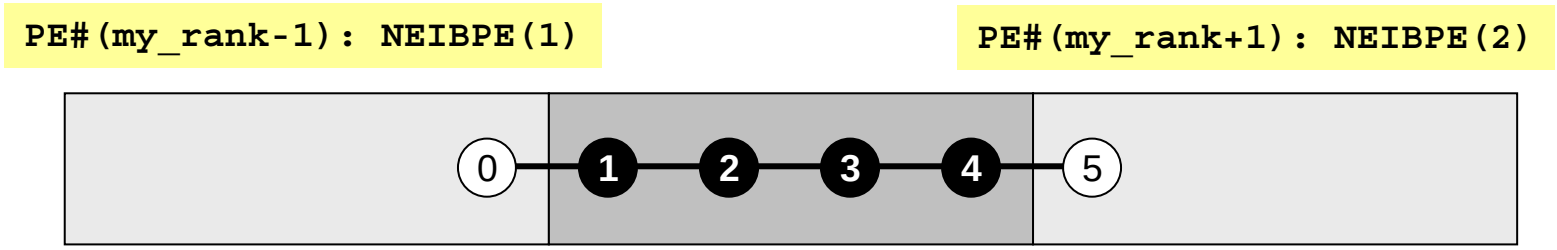
1D差分法の並列計算に必要な情報(1/3)
内点・外点・(境界点)

復習



1D差分法の並列計算に必要な情報(2/3)
隣接プロセッサ情報

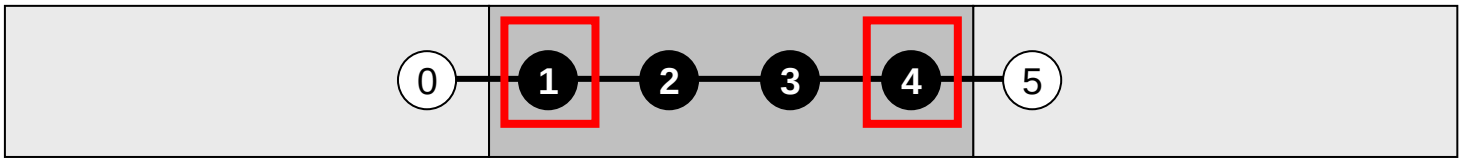
復習



1D差分法の並列計算に必要な情報(3/3)
通信テーブル

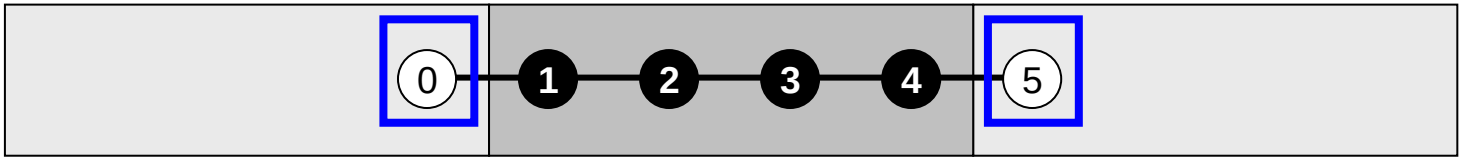
復習

隣接PEへの
送信(境界点)

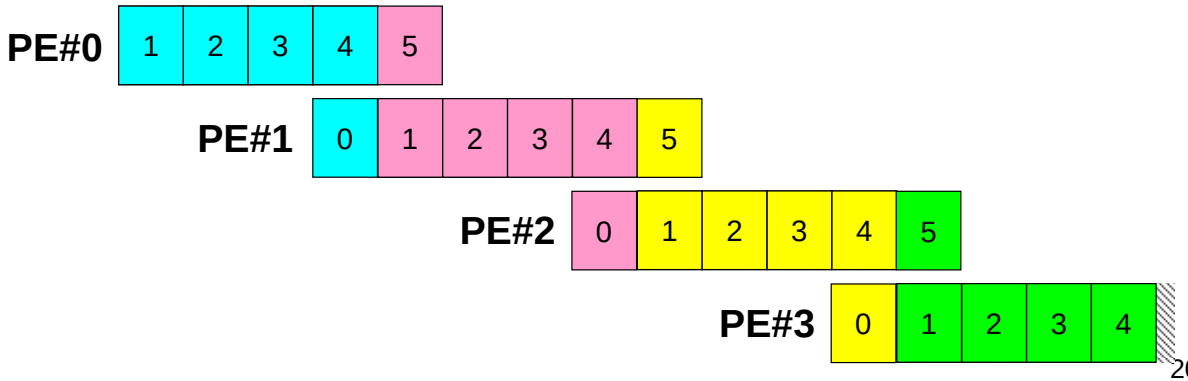



```
SENDbuf(1)=BUF(1)
SENDbuf(2)=BUF(4)
```

隣接PEからの
受信(外点)




```
BUF(0)=RECVbuf(1)
BUF(5)=RECVbuf(2)
```



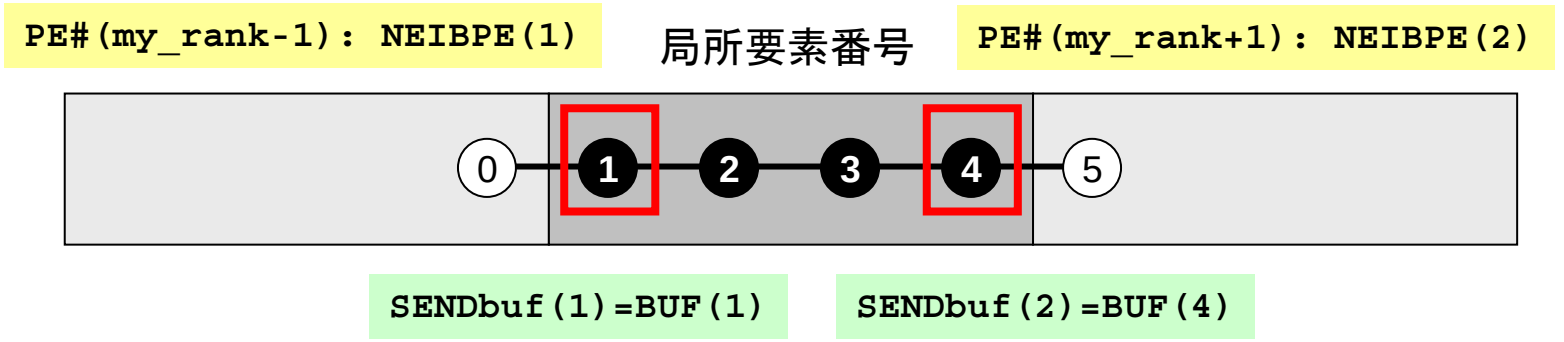
復習

一次元モデル例題: 1d-sr1.f（7/10）

境界点情報送信準備

```
!C
!C +-----------------------+
!C | PREPARE send  buffer |
!C +-----------------------+
!C===
      SENDbuf(1)= BUF(1)
      SENDbuf(2)= BUF(N)
      if (my_rank.eq.0       ) SENDbuf(1)= BUF(N)
      if (my_rank.eq.PETOT-1) SENDbuf(1)= BUF(1)
!C===
```

隣接プロセッサに BUF(1), BUF(N) を送る準備をする
境界点情報を SENDbufに代入



復習 一次元モデル例題: 1d-sr1.f（8/10）

境界点情報の送信

```
!C
!C +------+
!C | SEND |
!C +------+
!C===
      do neib= 1, NEIBPETOT
        call MPI_ISEND (SENDbuf(neib), BUFlength, MPI_INTEGER,          &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                &
     &                  request_send(neib), ierr)
      enddo
!C===


!C
!C +------+
!C | RECV |
!C +------+
!C===
      do neib= 1, NEIBPETOT
        call MPI_IRECV (RECVbuf(neib), BUFlength, MPI_INTEGER,          &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                &
     &                  request_recv(neib), ierr)
      enddo
!C===

!C
!C +------------------+
!C | WAITall for RECV |
!C +------------------+
!C===
      call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)
!C===
```

SENDbuf(neib)から始まる，長さBUFlength(=1)のメッセージを隣接PE(NEIBPE(neib))に送信する。

復習 一次元モデル例題: 1d-sr1.f（8/10）

外点情報の受信

```
!C
!C +------+
!C | SEND |
!C +------+
!C===
      do neib= 1, NEIBPETOT
        call MPI_ISEND (SENDbuf(neib), BUFlength, MPI_INTEGER,           &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                 &
     &                  request_send(neib), ierr)
      enddo
!C===


!C
!C +------+
!C | RECV |
!C +------+
!C===
      do neib= 1, NEIBPETOT
        call MPI_IRECV (RECVbuf(neib), BUFlength, MPI_INTEGER,           &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                 &
     &                  request_recv(neib), ierr)
      enddo
!C===

!C
!C +------------------+
!C | WAITall for RECV |
!C +------------------+
!C===
      call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)
!C===
```

RECVbuf(neib)から始まる，長さBUFlength(=1)のメッセージを隣接PE(NEIBPE(neib))から受信する。

なぜ SENDbuf，RECVbuf を用意するか?

- BUFを直接使っても良いのであるが・・・

```
!C
!C +------+
!C | SEND |
!C +------+
!C===
      do neib= 1, NEIBPETOT
        call MPI_ISEND (SENDbuf(neib), BUFlength, MPI_INTEGER,          &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                &
     &                  request_send(neib), ierr)
      enddo
!C===

!C
!C +------+
!C | RECV |
!C +------+
!C===
      do neib= 1, NEIBPETOT
        call MPI_IRECV (RECVbuf(neib), BUFlength, MPI_INTEGER,          &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                &
     &                  request_recv(neib), ierr)
      enddo
!C===
```

なぜ SENDbuf，RECVbuf を用意するか?

- BUFを直接使っても良いのであるが・・・
  - こんな感じになって使いにくい

```
!C
!C +------+
!C | SEND |
!C +------+
!C===
      do neib= 1, NEIBPETOT

        if (neib.eq.1) then
          ikk= 1
         else
          ikk= N
        endif

        if (my_rank.eq.0 .and. neib.eq.1) then
          ikk= N
        endif

        call MPI_ISEND (BUF(ikk), BUFlength, MPI_INTEGER,                &
     &                  NEIBPE(neib), 0, MPI_COMM_WORLD,                 &
     &                  request_send(neib), ierr)
      enddo
!C===
```

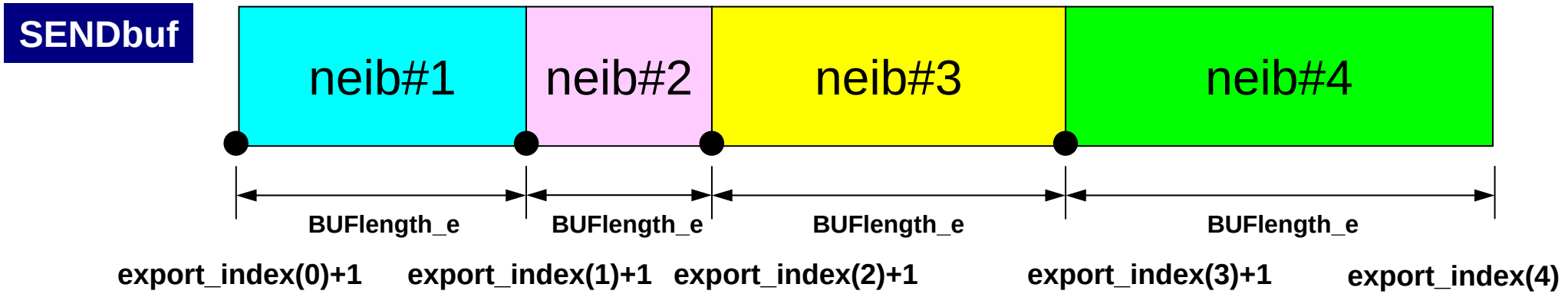
復習

一般化された通信テーブル：送信

- 送信相手
  - NEIBPETOT，NEIB(neib)
- それぞれの送信相手に送るメッセージサイズ
  - export_index(neib), neib= 1, NEIBPETOT
- 「境界点」番号
  - export_item(k), k= 1, export_index(NEIBPETOT)
- それぞれの送信相手に送るメッセージ
  - SENDbuf(k), k= 1, export_index(NEIBPETOT)

復習

送信（MPI_Isend/Irecv/Waitall）

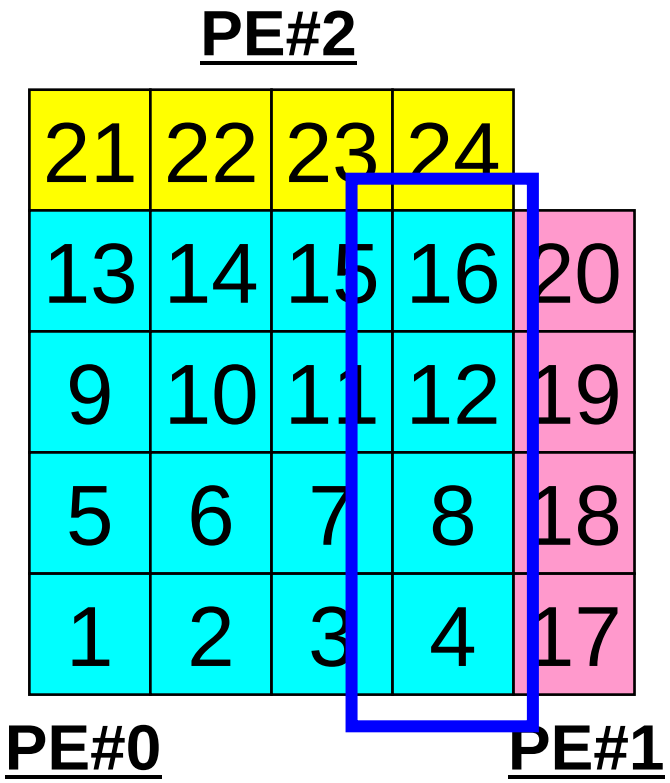



```
 do neib= 1, NEIBPETOT
   do k= export_index(neib-1)+1, export_index(neib)
     kk= export_item(k)
     SENDbuf(k)= VAL(kk)
   enddo
 enddo

 do neib= 1, NEIBPETOT
   iS_e= export_index(neib-1) + 1
   iE_e= export_index(neib  )
   BUFlength_e= iE_e + 1 - iS_e

   call MPI_ISEND                                                    &
&          (SENDbuf(iS_e), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&           MPI_COMM_WORLD, request_send(neib), ierr)
 enddo

 call MPI_WAITALL (NEIBPETOT, request_send, stat_recv, ierr)
```

送信バッファへの代入
温度などの変数を直接送信，受信に使うのではなく，このようなバッファへ一回代入して計算することを勧める。

送信バッファの効能

```
 do neib= 1, NEIBPETOT
   iS_e= export_index(neib-1) + 1
   iE_e= export_index(neib  )
   BUFlength_e= iE_e + 1 - iS_e

   call MPI_ISEND                                                  &
&          (VAL(...), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&           MPI_COMM_WORLD, request_send(neib), ierr)
 enddo
```



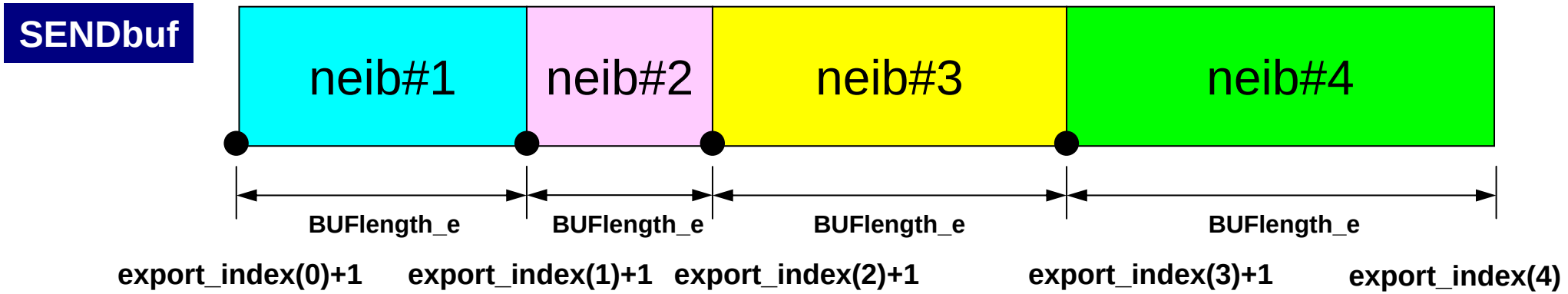
たとえば，この境界点は連続していないので，

- 送信バッファの先頭アドレス
- そこから数えて●●のサイズのメッセージ

というような方法が困難

復習

送信（MPI_Sendrecv）



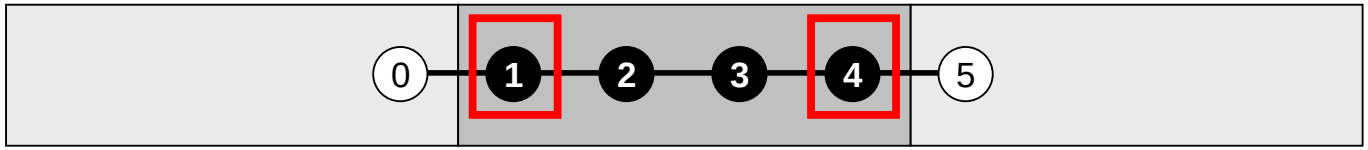

```
do neib= 1, NEIBPETOT
  do k= export_index(neib-1)+1, export_index(neib)
    kk= export_item(k)
    SENDbuf(k)= VAL(kk)
  enddo
enddo
```

送信バッファへの代入

```
do neib= 1, NEIBPETOT
  iS_e= export_index(neib-1) + 1
  iE_e= export_index(neib  )
  BUFlength_e= iE_e + 1 - iS_e

  call MPI_SENDRECV                                                  &
&         (SENDbuf(iS_e), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&          RECVbuf(iS_i), BUFlength_i, MPI_INTEGER, NEIBPE(neib), 0,&
&          MPI_COMM_WORLD, stat_sr, ierr)
  enddo
```

復習

送信:一次元問題



- 送信相手
  - NEIBPETOT, NEIB(neib)
    - NEIBPETOT=2, NEIB(1)= my_rank-1, NEIB(2)= my_rank+1
- それぞれの送信相手に送るメッセージサイズ
  - export_index(neib), neib= 1, NEIBPETOT
    - export_index(0)=0, export_index(1)= 1, export_index(2)= 2
- 「境界点」番号
  - export_item(k), k= 1, export_index(NEIBPETOT)
    - export_item(1)= 1, export_item(2)= N
- それぞれの送信相手に送るメッセージ
  - SENDbuf(k), k= 1, export_index(NEIBPETOT)
    - SENDbuf(1)= BUF(1), SENDbuf(2)= BUF(N)

```
SENDbuf(1)=BUF(1)
```

```
SENDbuf(2)=BUF(4)
```

復習

一般化された通信テーブル：受信

- 受信相手
  - NEIBPETOT，NEIB(neib)
- それぞれの受信相手から受け取るメッセージサイズ
  - import_index(neib)
- 「外点」番号
  - import_item(k), k= 1, import_index(NEIBPETOT)
- それぞれの受信相手から受け取るメッセージ
  - RECVbuf(k), k= 1, import_index(NEIBPETOT)

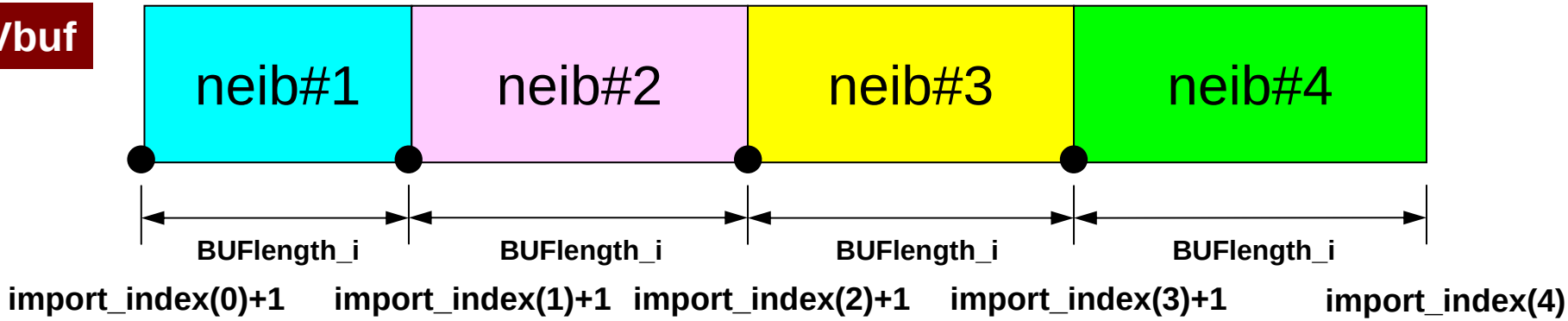
復習

受信（MPI_Isend/Irecv/Waitall）

```
 do neib= 1, NEIBPETOT
   iS_i= import_index(neib-1) + 1
   iE_i= import_index(neib  )
   BUFlength_i= iE_i + 1 - iS_i

   call MPI_IRECV                                                  &
&          (RECVbuf(iS_i), BUFlength_i, MPI_INTEGER, NEIBPE(neib), 0,&
&           MPI_COMM_WORLD, request_recv(neib), ierr)
  enddo

  call MPI_WAITALL (NEIBPETOT, request_recv, stat_recv, ierr)

  do neib= 1, NEIBPETOT
     do k= import_index(neib-1)+1, import_index(neib)
       kk= import_item(k)
       VAL(kk)= RECVbuf(k)
     enddo
  enddo
```

受信バッファから代入

RECVbuf



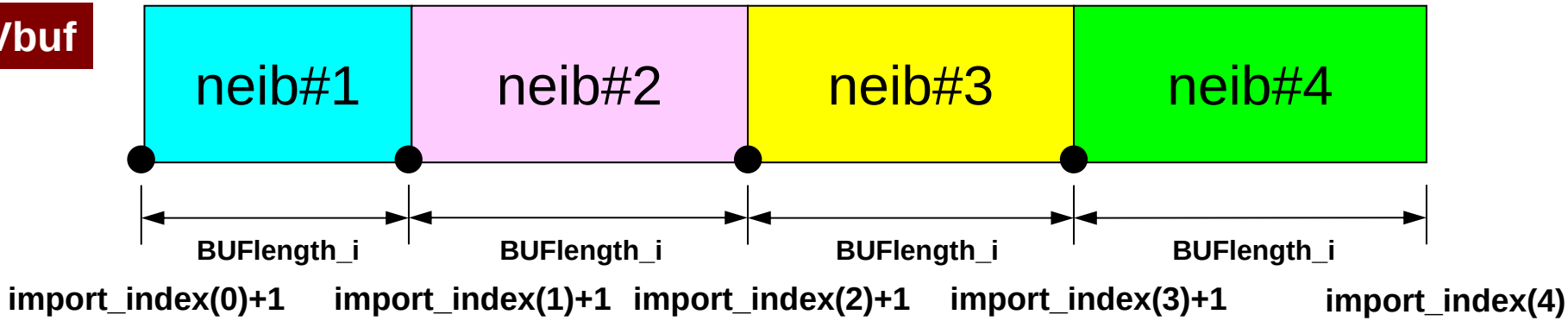
復習

受信（MPI_Sendrecv）

```
 do neib= 1, NEIBPETOT
   iS_i= import_index(neib-1) + 1
   iE_i= import_index(neib  )
   BUFlength_i= iE_i + 1 - iS_i

   call MPI_SENDRECV                                                  &
&         (SENDbuf(iS_e), BUFlength_e, MPI_INTEGER, NEIBPE(neib), 0,&
&          RECVbuf(iS_i), BUFlength_i, MPI_INTEGER, NEIBPE(neib), 0,&
&          MPI_COMM_WORLD, stat_sr, ierr)
  enddo

  do neib= 1, NEIBPETOT
    do k= import_index(neib-1)+1, import_index(neib)
      kk= import_item(k)
      VAL(kk)= RECVbuf(k)
    enddo
  enddo
```

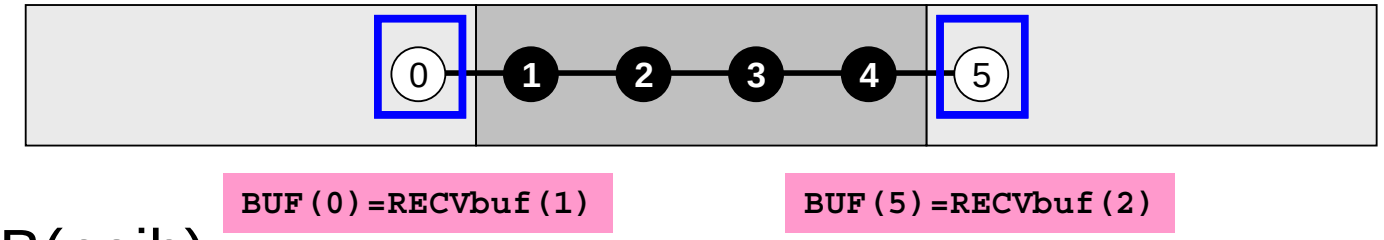
受信バッファからの代入

RECVbuf



復習

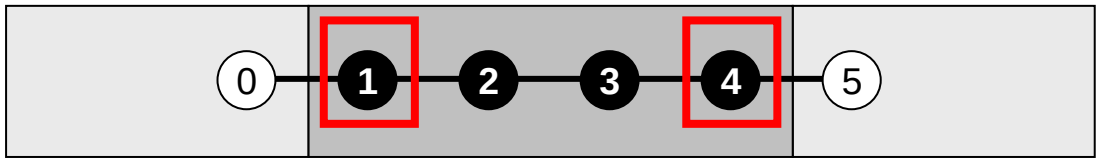
受信:一次元問題



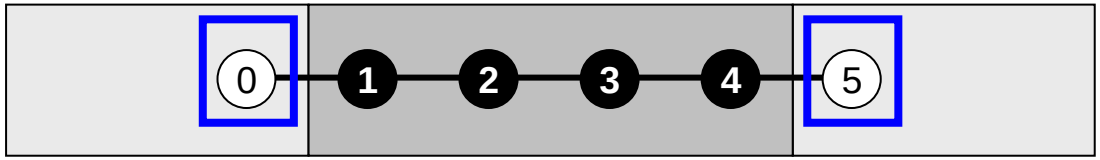
- 受信相手
  - NEIBPETOT, NEIB(neib)
    - NEIBPETOT=2, NEIB(1)= my_rank-1, NEIB(2)= my_rank+1
- それぞれの受信相手から受け取るメッセージサイズ
  - import_index(neib), neib= 1, NEIBPETOT
    - import_index(0)=0, import_index(1)= 1, import_index(2)= 2
- 「外点」番号
  - import_item(k), k= 1, import_index(NEIBPETOT)
    - import_item(1)= 0, import_item(2)= N+1
- それぞれの受信相手から受け取るメッセージ
  - RECVbuf(k), k= 1, import_index(NEIBPETOT)
    - BUF(0)=RECVbuf(1), BUF(N+1)=RECVbuf(2)

復習

一般化された通信テーブル



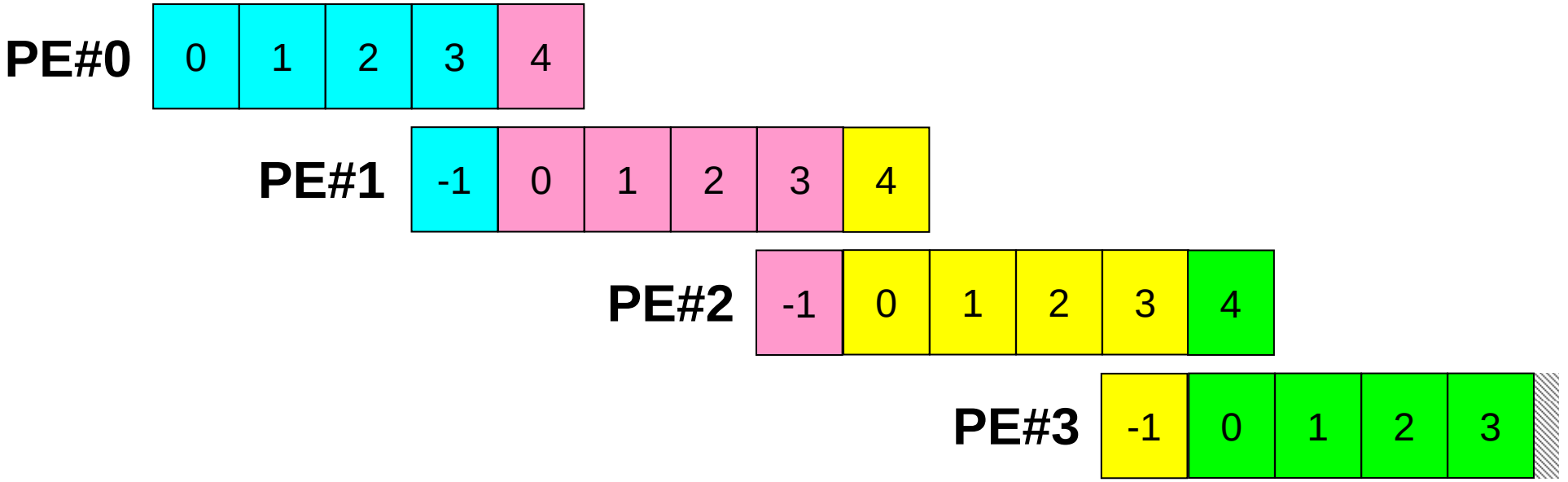
SENDbuf(1)=BUF(1)　　SENDbuf(2)=BUF(4)

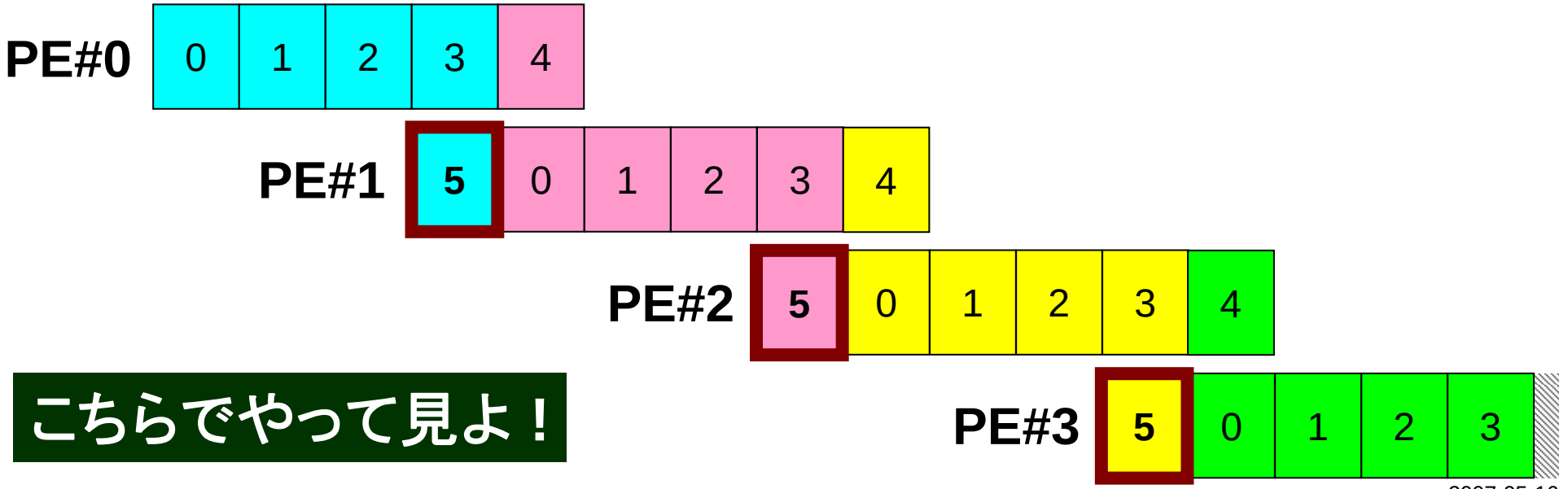


BUF(0)=RECVbuf(1)　　BUF(5)=RECVbuf(2)

```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1

import_index(1)= 1
import_index(2)= 2
import_item  (1)= 0
import_item  (2)= N+1

export_index(1)= 1
export_index(2)= 2
export_item  (1)= 1
export_item  (2)= N

if (my_rank.eq.0) then
  import_item  (1)= N+1
  export_item  (1)= N
  NEIBPE(1)= my_rank+1
endif
```

課題S3：Cユーザー

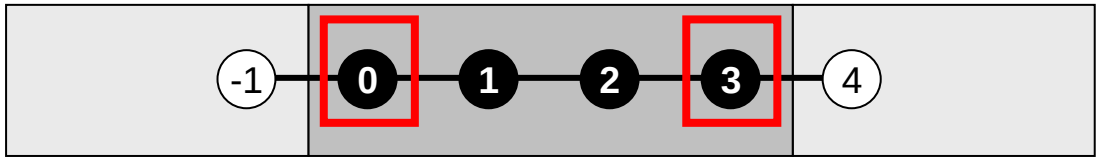




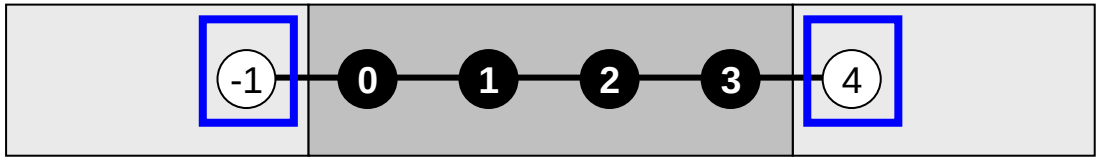
こちらでやって見よ！

復習

一般化された通信テーブル：C言語（1/2）



SENDbuf(1)=BUF(1)　　SENDbuf(2)=BUF(4)

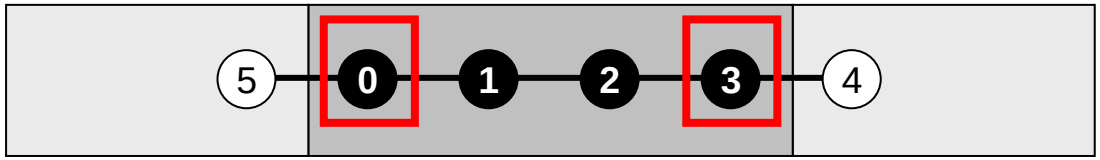


BUF(0)=RECVbuf(1)　　BUF(5)=RECVbuf(2)

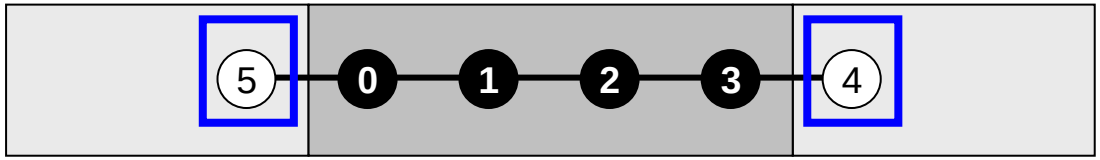
C言語の場合BUF[-1]という配列は本来存在しないが，
コンパイラによっては通ってしまう場合がある。

```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1

import_index(1)= 1
import_index(2)= 2
import_item (1)= -1
import_item (2)= N

export_index(1)= 1
export_index(2)= 2
export_item (1)= 0
export_item (2)= N-1

if (my_rank.eq.0) then
  import_item (1)= N
  export_item (1)= N-1
  NEIBPE(1)= my_rank+1
endif
```

一般化された通信テーブル：C言語（2/2）



SENDbuf(1)=BUF(1)　　SENDbuf(2)=BUF(4)



BUF(0)=RECVbuf(1)　　BUF(5)=RECVbuf(2)

[-1]の代わりに[N+1]を使用すれば問題は無くなる
（JacobiやSORではこれがやりにくい）。
並列化時に元のプログラムの大幅な書き直しが必要。

```
NEIBPETOT= 2
NEIBPE(1)= my_rank - 1
NEIBPE(2)= my_rank + 1

import_index(1)= 1
import_index(2)= 2
import_item (1)= N+1
import_item (2)= N

export_index(1)= 1
export_index(2)= 2
export_item (1)= 0
export_item (2)= N-1

if (my_rank.eq.0) then
  import_item (1)= N
  export_item (1)= N-1
  NEIBPE(1)= my_rank+1
endif
```

補足：課題S2のアプローチ

- 今回紹介した，一般的な形でやっても良い

$$A_L(i)\times\phi_{i-1}+A_D(i)\times\phi_i+A_R(i)\times\phi_{i+1}=BF(i)\quad(1\le i\le N)$$

$$DIAG(i)\times PHI(i)+\sum_{k=INDEX(i-1)+1}^{INDEX(i)}\left[AMAT(k)\times PHI(ITEM(k))\right]=RHS(i),\quad(1\le i\le N)$$

今後の予定

5月23日	休講	• 日本地球惑星科学連合大会 –「情報地球惑星科学」もお忘れなく:5月20日（日）午前 • 中島は日本計算工学会 –大規模シミュレーションと並列前処理手法
5月30日	休講	• ICCS 2007，北京 – International Conference on Computational Science – International Workshop on Advances in Computational Geomechanics and Geophysics (IACGG2007) • Grid World 2007 –5月30日・31日，東京国際フォーラム –詳細は講義HP
6月　6日		課題S1，S2解説
6月13日		可視化，チューニング
6月20日～		並列アプリケーション