線形ソルバー

2011年夏季集中講義

中島研吾

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

TOC

- 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

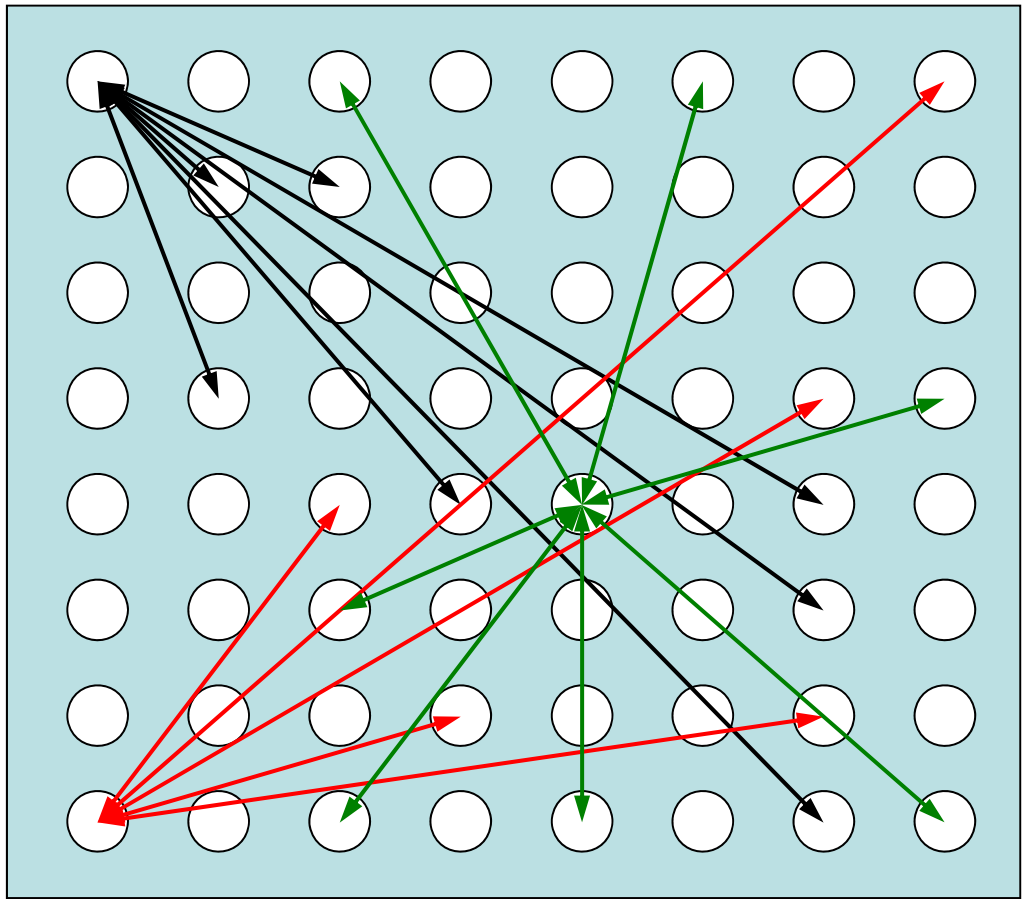
科学技術計算における大規模線形方程式の解法

- 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

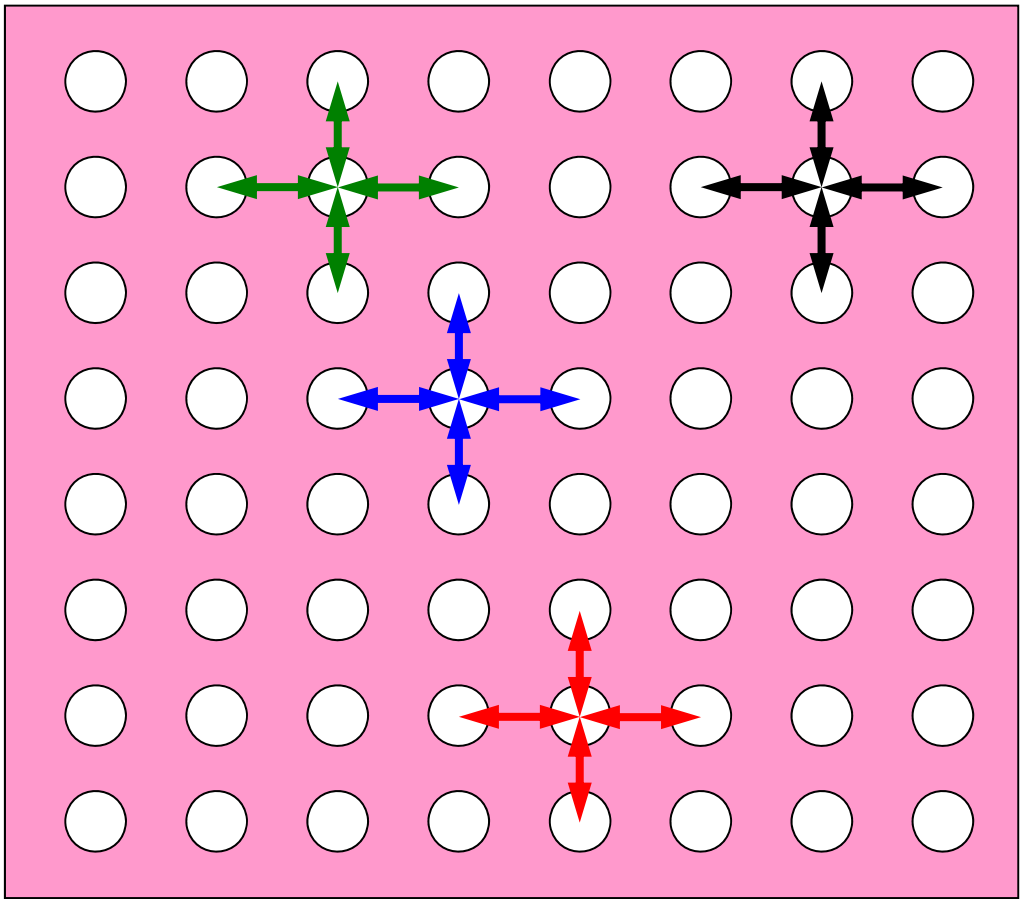
境界要素法，スペクトル法，MD法



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

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
  - 概して遅い

- 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）
  - 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
  - （ガラーキン法）熱伝導，弾性，ねじり：本コードの場合もSPD
- アルゴリズム
  - 最急降下法（Steepest Descent Method）の変種
  - $\boldsymbol{x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}}$
    - $x^{(i)}$：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 $\boldsymbol{\{x-y\}^T[A]\{x-y\}}$を最小とするような$\{x\}$を求める。
  - 詳細は参考文献参照
    - 例えば：森正武「数値解析（第2版）」（共立出版）

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。

前処理付共役勾配法

Preconditioned Conjugate Gradient Method（PCG）

```
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 z(i-1)
    endif
    q(i)= [A]p(i)
    αi  = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

実際にやるべき計算は：

$$\{z\} = [M]^{-1}\{r\}$$

「近似逆行列」の計算が必要：

$$[M]^{-1} \approx [A]^{-1}, \quad [M] \approx [A]$$

究極の前処理：本当の逆行列

$$[M]^{-1} = [A]^{-1}, \quad [M] = [A]$$

対角スケーリング：簡単＝弱い

$$[M]^{-1} = [D]^{-1}, \quad [M] = [D]$$

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)

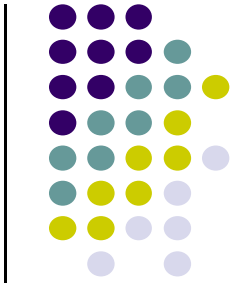
- 最もよく使用されている前処理（疎行列用）
  - Incomplete LU Factorization
  - Incomplete Cholesky Factorization（対称行列）

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)

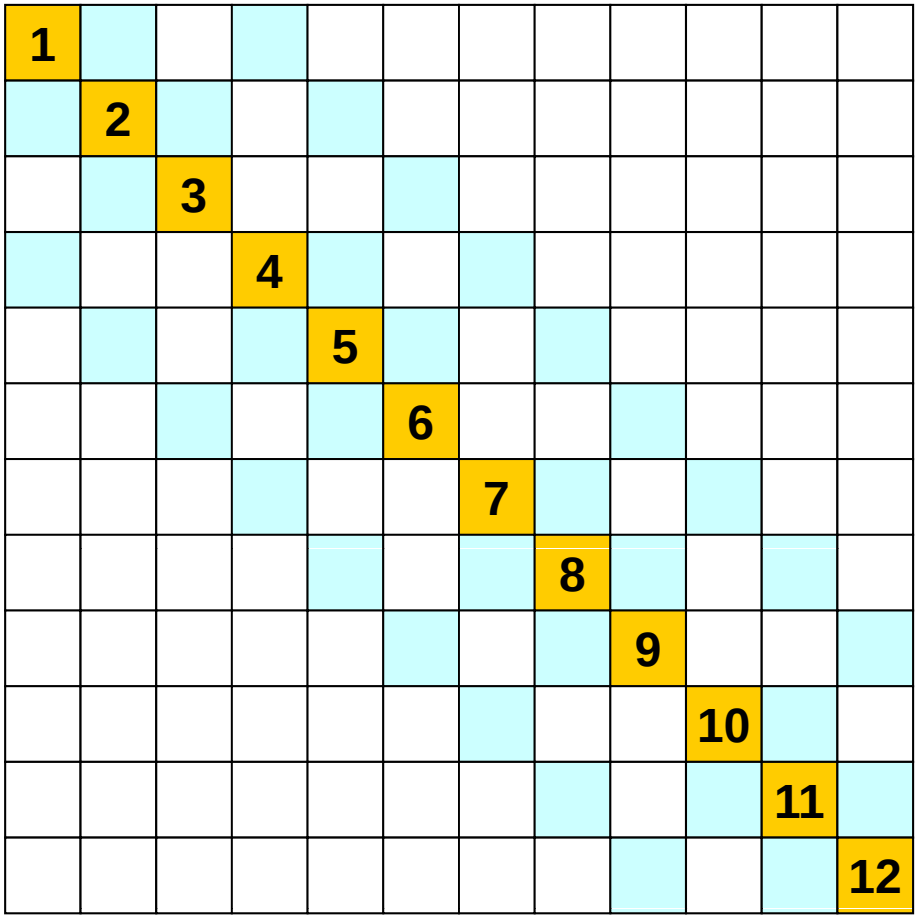
LU分解法：完全LU分解法

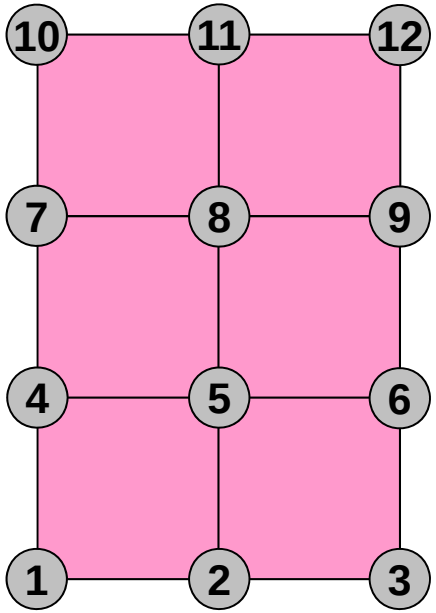


- 直接法の一種
  - 逆行列を直接求める手法
  - 「逆行列」に相当するものを保存しておけるので，右辺が変わったときに計算時間を節約できる
  - 逆行列を求める際にFill-in（もとの行列では0であったところに値が入る）が生じる
- 不完全LU分解法
  - Fill-inの発生を制限して，前処理に使う手法
    - 不完全な逆行列，少し弱い直接法

実例：差分法による熱伝導等

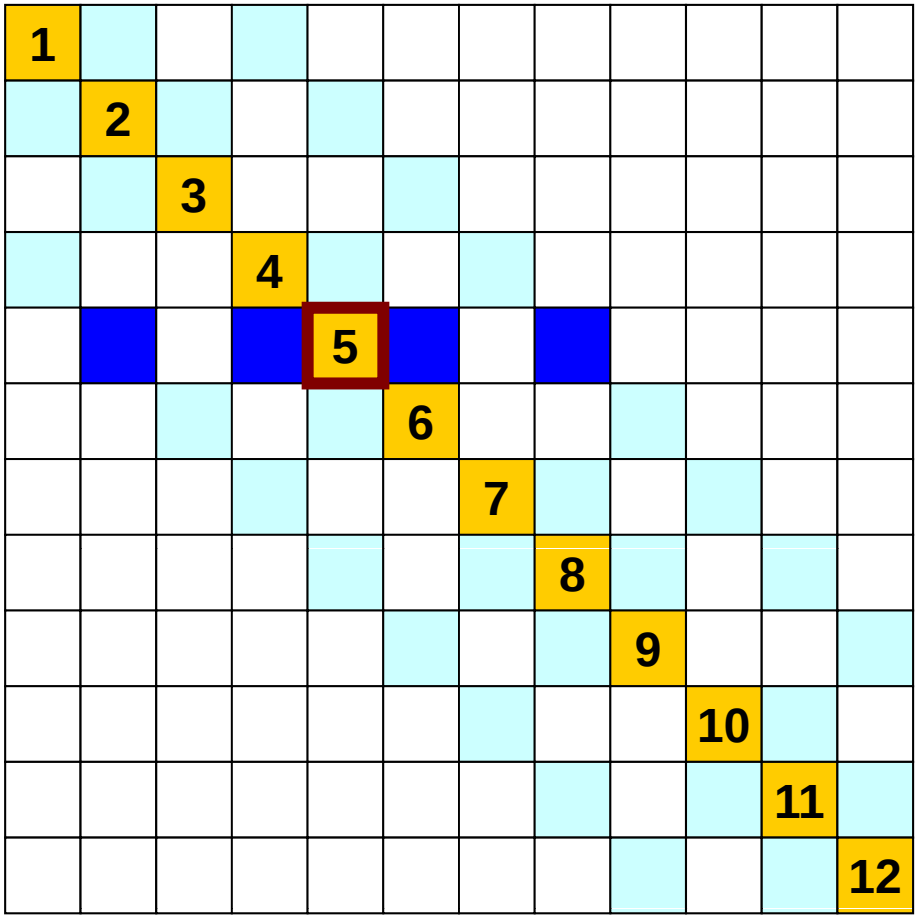
5点差分

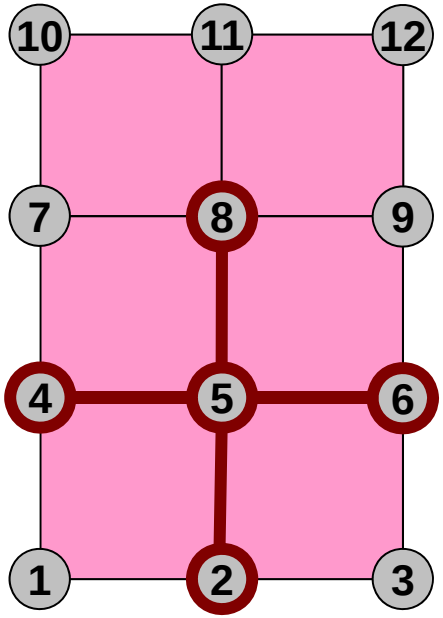




実例：差分法による熱伝導等

5点差分



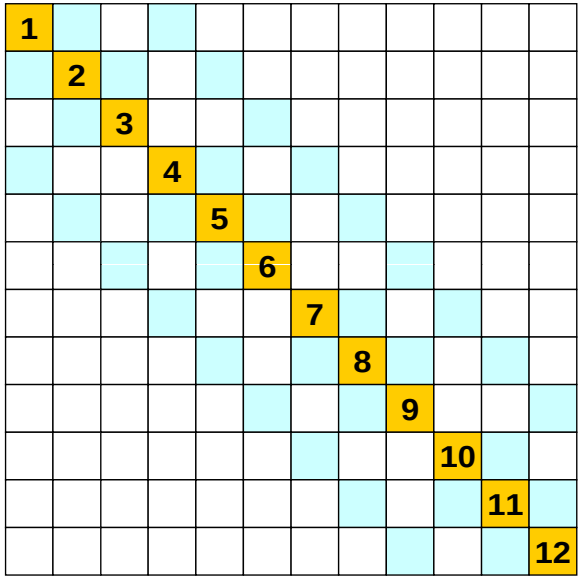


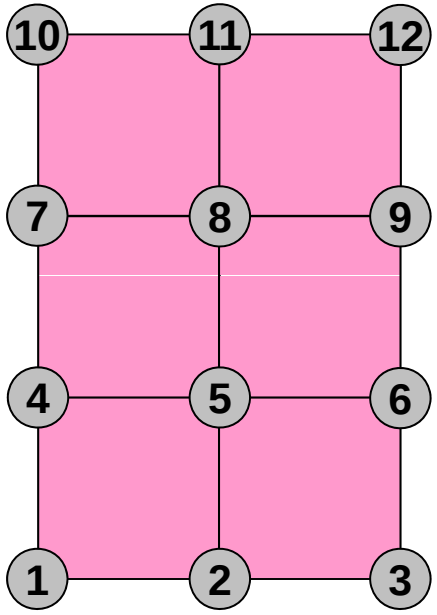
実例：係数マトリクス

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00

X =

0.00
3.00
10.00
11.00
10.00
19.00
20.00
16.00
28.00
42.00
36.00
52.00



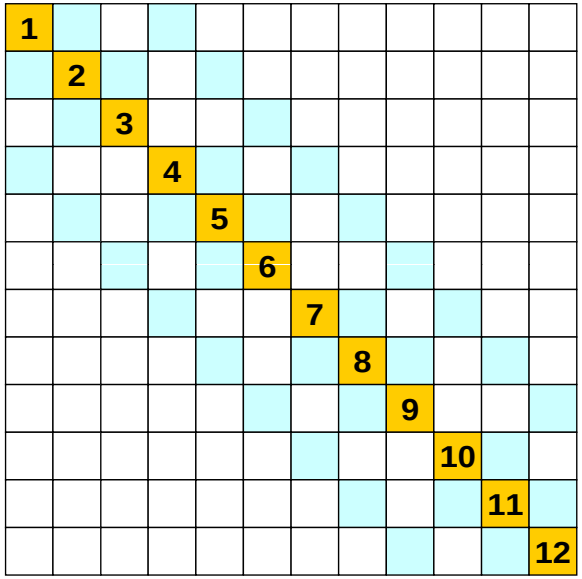


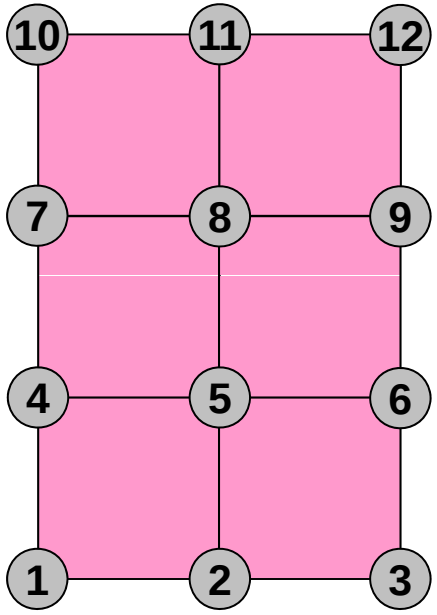
実例：解

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00

1.00	0.00
2.00	3.00
3.00	10.00
4.00	11.00
5.00	10.00
6.00	19.00
7.00	20.00
8.00	16.00
9.00	28.00
10.00	42.00
11.00	36.00
12.00	52.00

=





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

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	0.00	6.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−1.00	0.00	−1.00	6.00

LU分解したマトリクス

[L][U]同時に表示

[L]対角成分(=1)省略

(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)

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	−0.03	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	−0.03	0.00	5.83	−1.03	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	−0.03	−0.18	5.64	−1.03	−0.18	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.64	−0.03	−0.18	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03	−0.18	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63	−0.03	−0.18	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63

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）
[L][U]同時に表示
[L]対角成分（=1）省略

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	0.00	−0.17	5.66	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.65	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65

完全LU分解した
マトリクス
[L][U]同時に表示
[L]対角成分（=1）省略
（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）

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	−0.03	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	−0.03	0.00	5.83	−1.03	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	−0.03	−0.18	5.64	−1.03	−0.18	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.64	−0.03	−0.18	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03	−0.18	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63	−0.03	−0.18	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63

解の比較：ちょっと違う

不完全LU分解

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	0.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	0.00	−0.17	5.66	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.65	0.00	0.00	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00	0.00	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65	0.00	0.00	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	0.00	0.00	5.83	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.17	5.65	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.65

→

0.92
1.75
2.76
3.79
4.46
5.57
6.66
7.25
8.46
9.66
10.54
11.83

完全LU分解

6.00	−1.00	0.00	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	5.83	−1.00	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	5.83	−0.03	−0.17	−1.00	0.00	0.00	0.00	0.00	0.00	0.00
−0.17	−0.03	0.00	5.83	−1.03	0.00	−1.00	0.00	0.00	0.00	0.00	0.00
0.00	−0.17	−0.03	−0.18	5.64	−1.03	−0.18	−1.00	0.00	0.00	0.00	0.00
0.00	0.00	−0.17	0.00	−0.18	5.64	−0.03	−0.18	−1.00	0.00	0.00	0.00
0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01	−1.00	0.00	0.00
0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03	−0.18	−1.00	0.00
0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63	−0.03	−0.18	−1.00
0.00	0.00	0.00	0.00	0.00	0.00	−0.17	−0.03	−0.01	5.82	−1.03	−0.01
0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	−0.03	−0.18	5.63	−1.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	−0.18	0.00	−0.18	5.63

→

1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
11.00
12.00

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
  - 100コアを超えるような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（の場合もある）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)

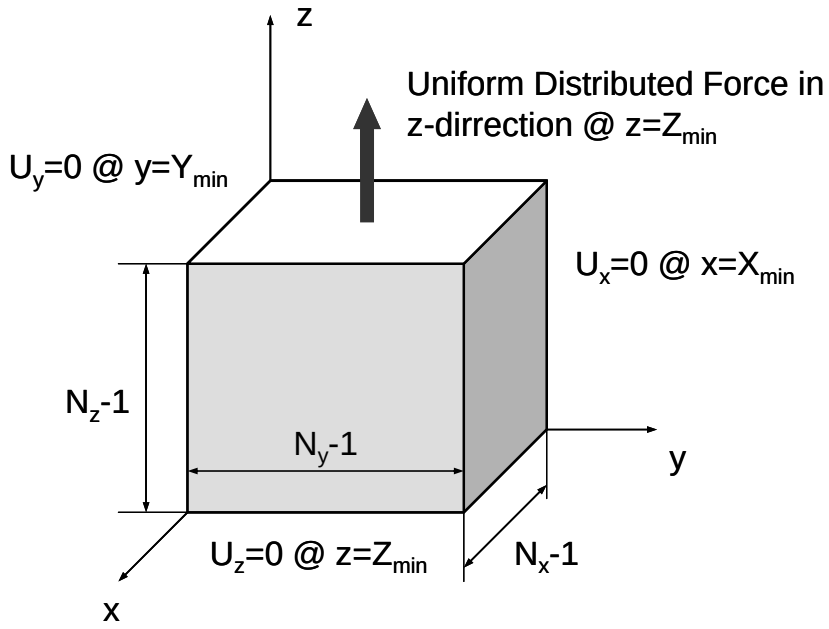
ILU(2)

Gaussian Elimination

- Simple
- Easy to be Parallelized
- Cheap

- Complicated
- Global Dependency
- Expensive

三次元弾性解析問題の例

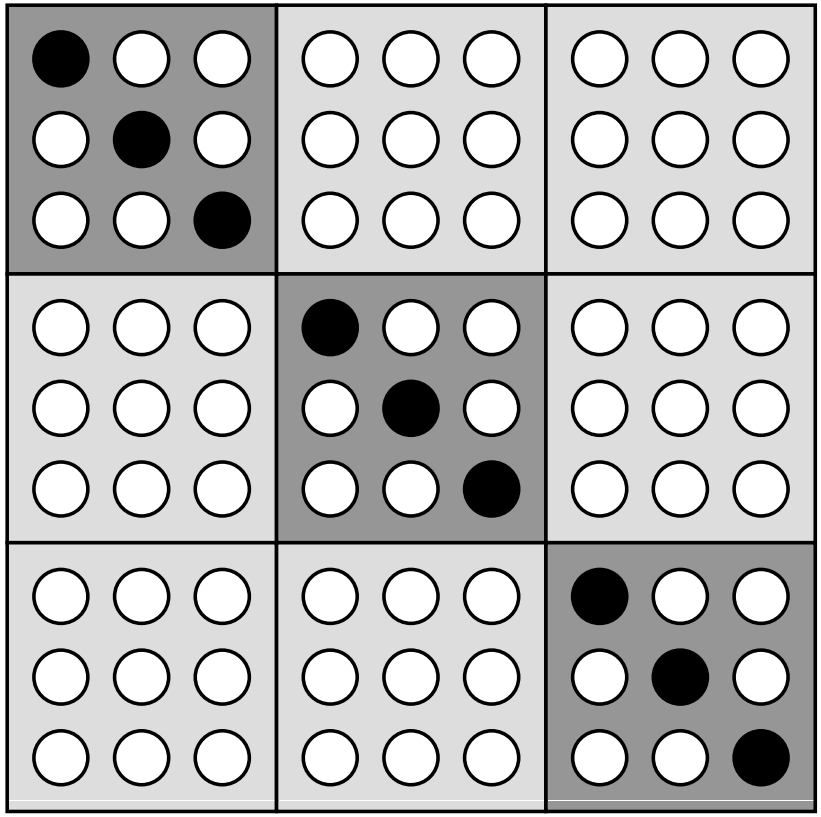


3×32×32×32= 98,304 DOF
一様物性，境界条件
条件は良い問題
1 PE

- 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

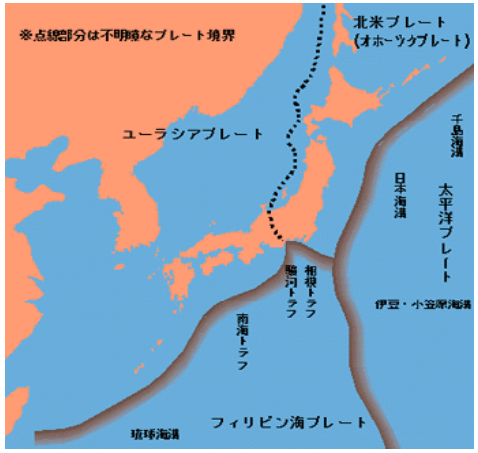
3自由度/節点をブロックとして扱う

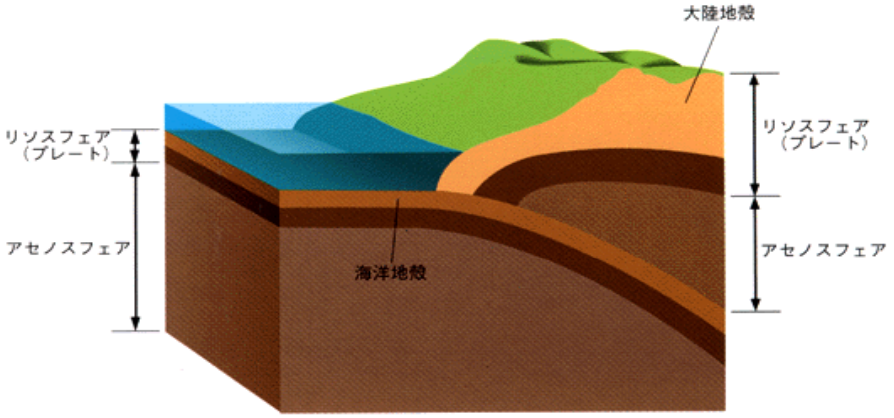


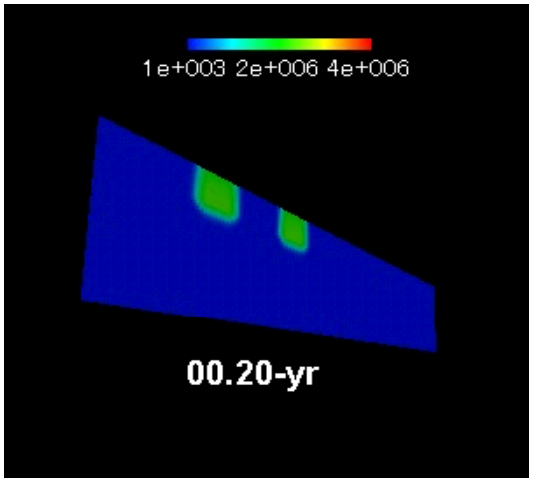
- 線形ソルバーの概要
  - 直接法
  - 並列法
  - 共役勾配法（Conjugate Gradient）
  - 前処理
- 接触問題の例（前処理）
  - Selective Blocking Preconditioning

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

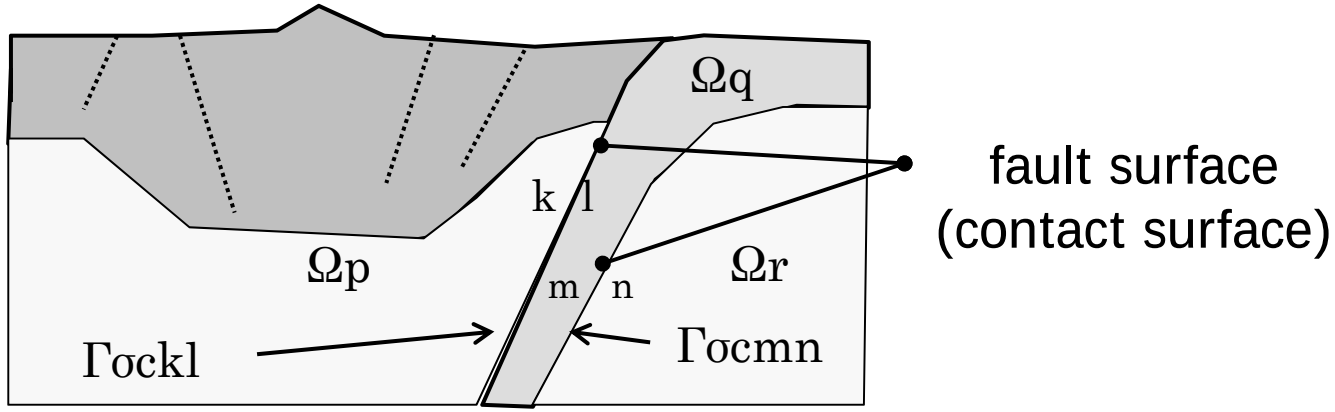
$$\sum_p \left[\int_{\Omega_p} \lfloor \delta\varepsilon \rfloor \{\sigma\} dV - \int_{\Gamma_{\sigma p}} \lfloor \delta u \rfloor \{f_o\} dS - \int_{\Omega_p} \lfloor \delta u \rfloor \{r_o\} dV \right]$$

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

接触力項

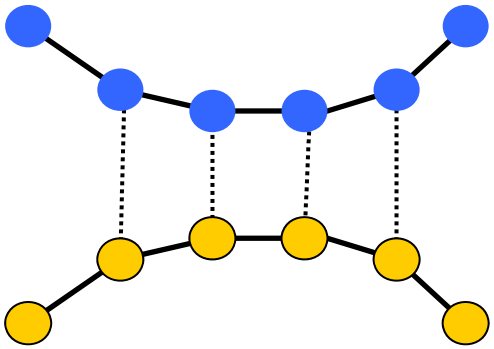
$$\int_{\Gamma_{\sigma ckl}} \lfloor \delta \left(\bar{f}_c^n \right) \rfloor \{g\} dS = 0$$

接触面での物体の重なりは無い



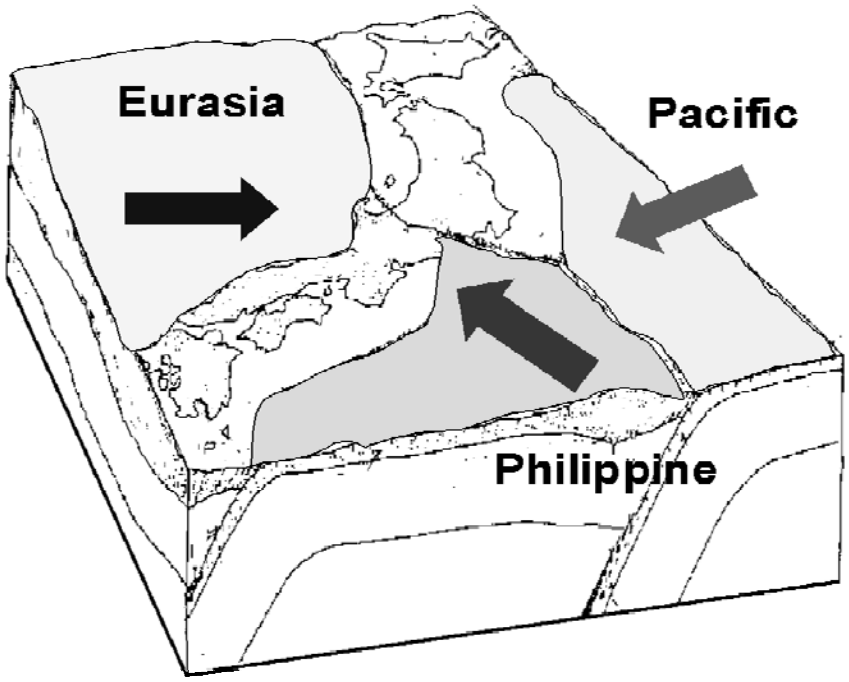
接触問題における前処理手法(続き)

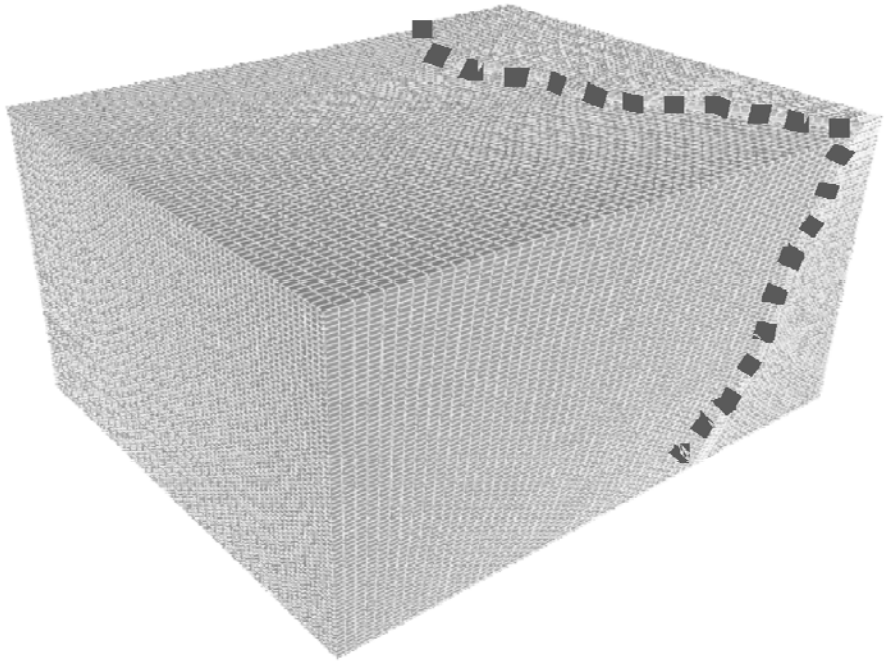
- 仮定
  - 微小変形理論に基づく，静的接触(接触グループに属する節点の座標は同一)
  - 摩擦なし:対称行列(最終的には摩擦ありの場合も計算)
- 特殊な前処理手法を開発: ***Selective Blocking.***
  - 三次元接触問題において，効率的に解を得ることのできる，効率的な前処理手法である
- 計算
  - 日立SR2201(東大):2001～2002
  - 地球シミュレータ:2002～



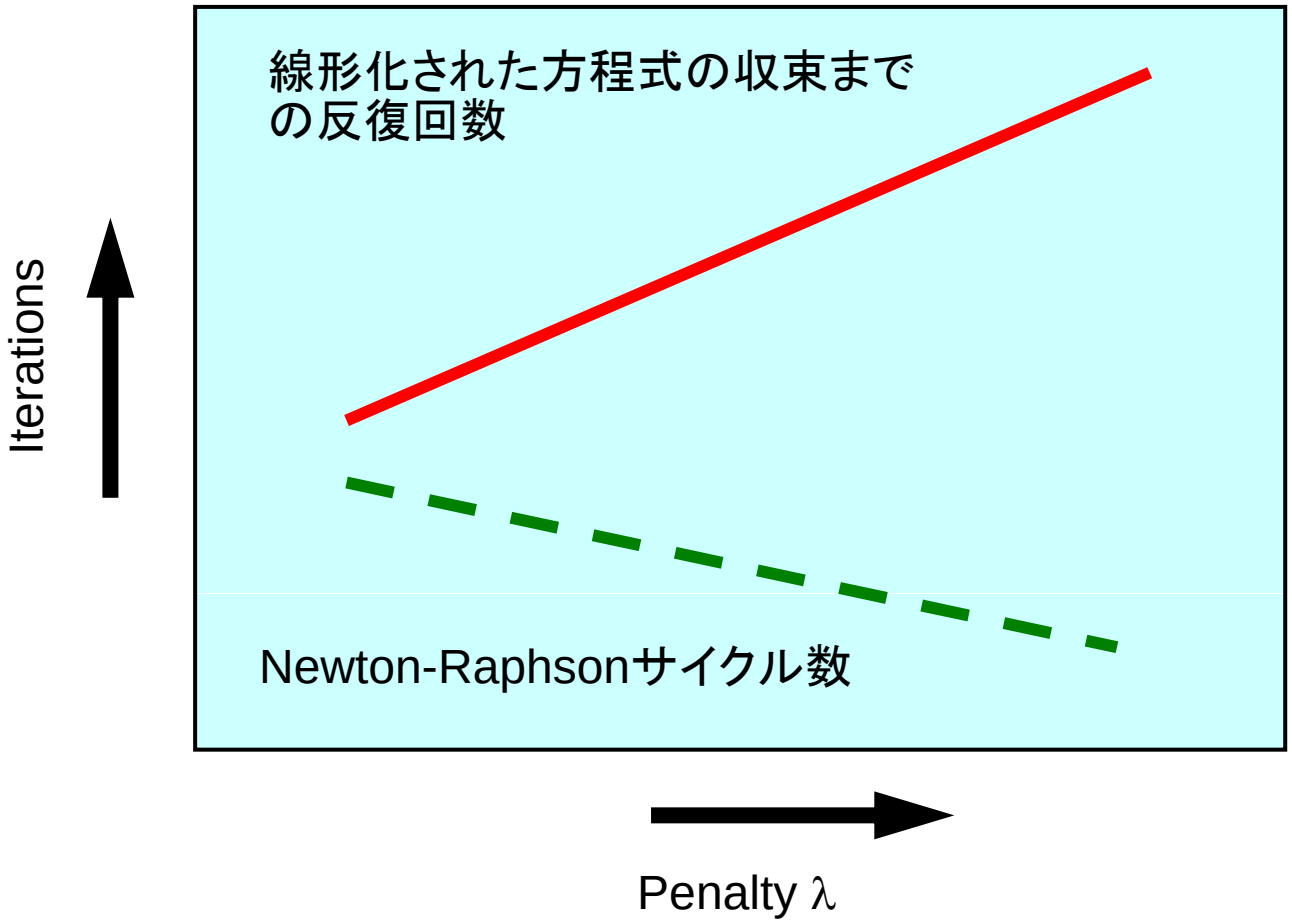
Geophysics Application w/Contact

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	

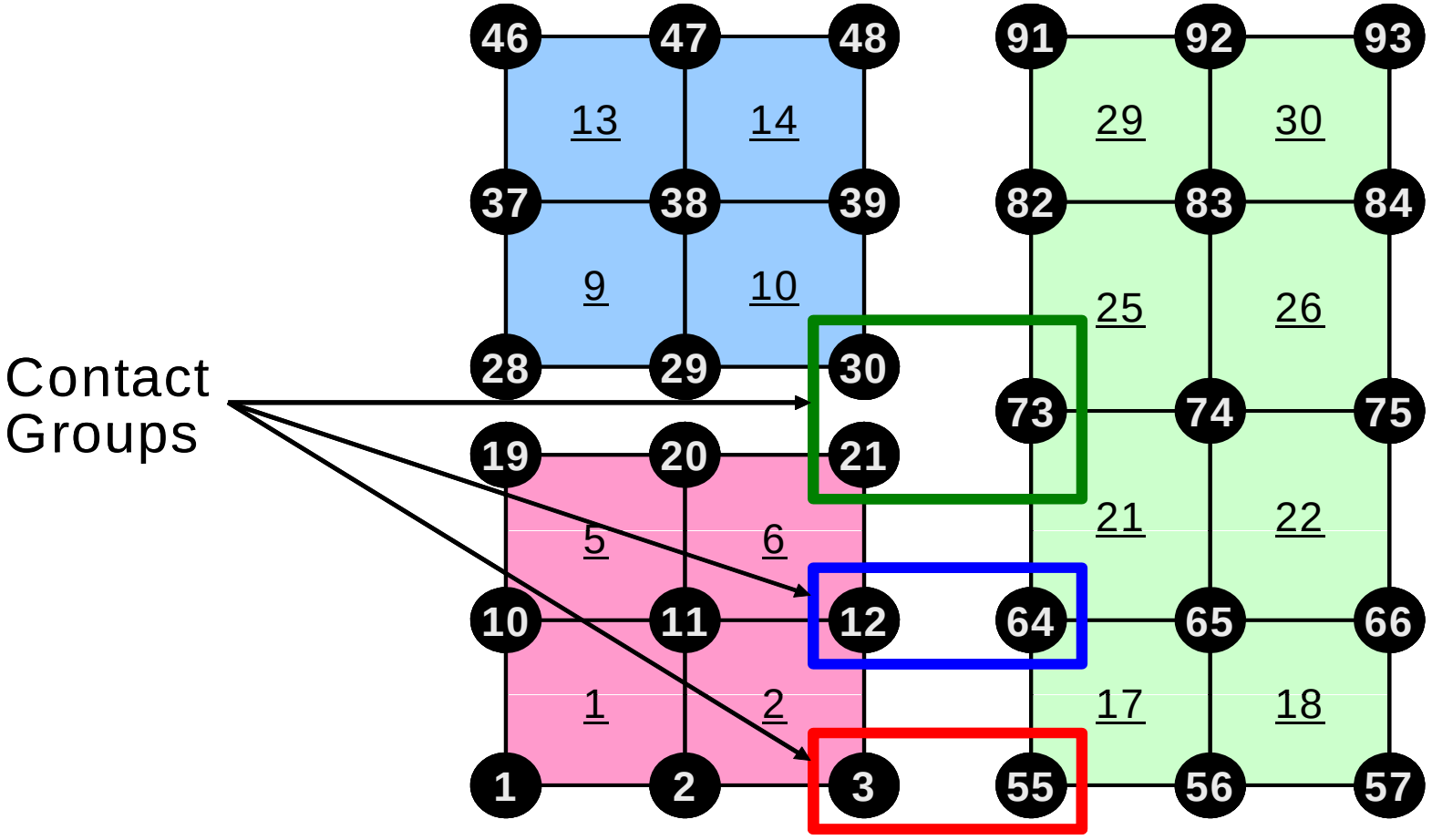
DEEP Fill-in　BLOCKING

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

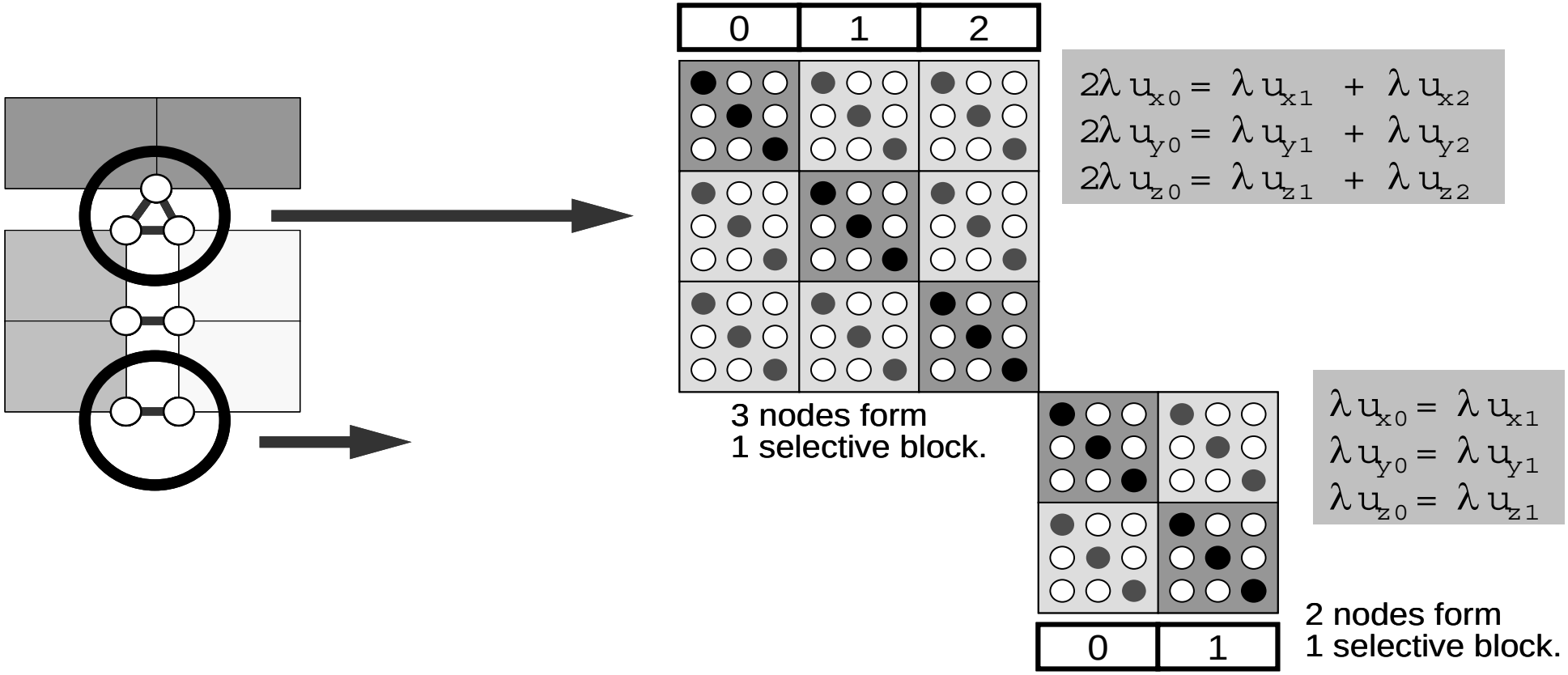
接触条件により物理的に強くカップルした節点群をブロック化



Selective Blocking

接触問題向けの特別な前処理手法

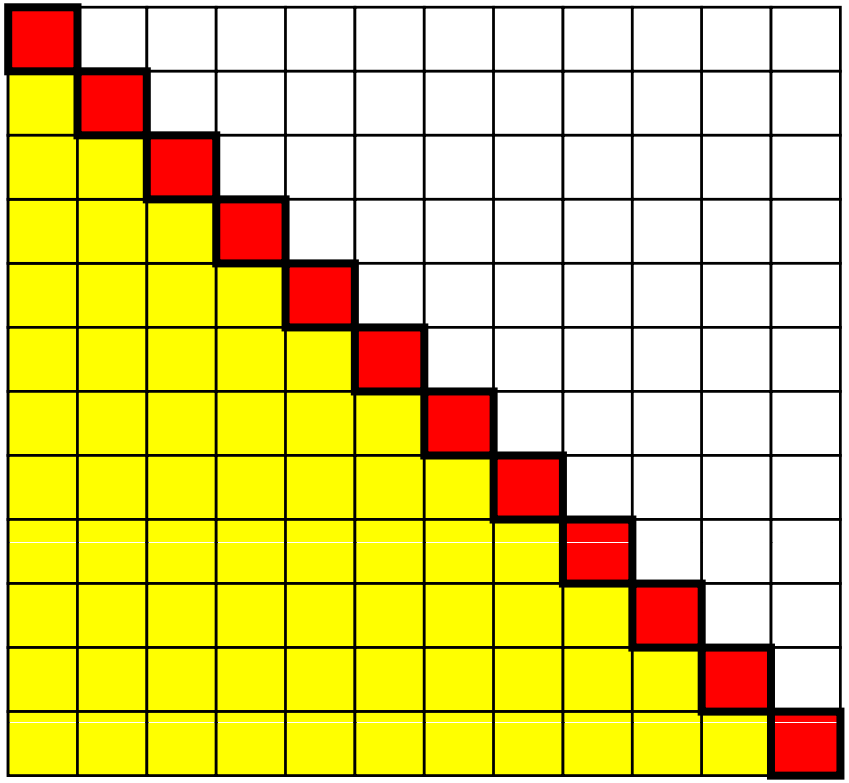
接触条件により物理的に強くカップルした節点群をブロック化



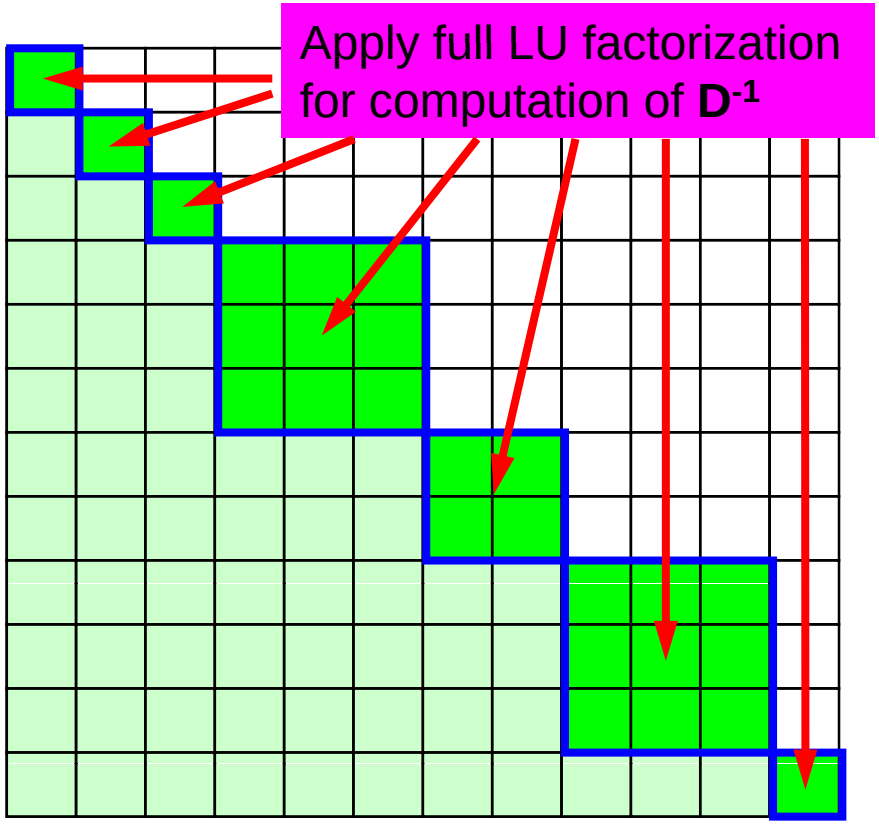
Selective Blocking

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

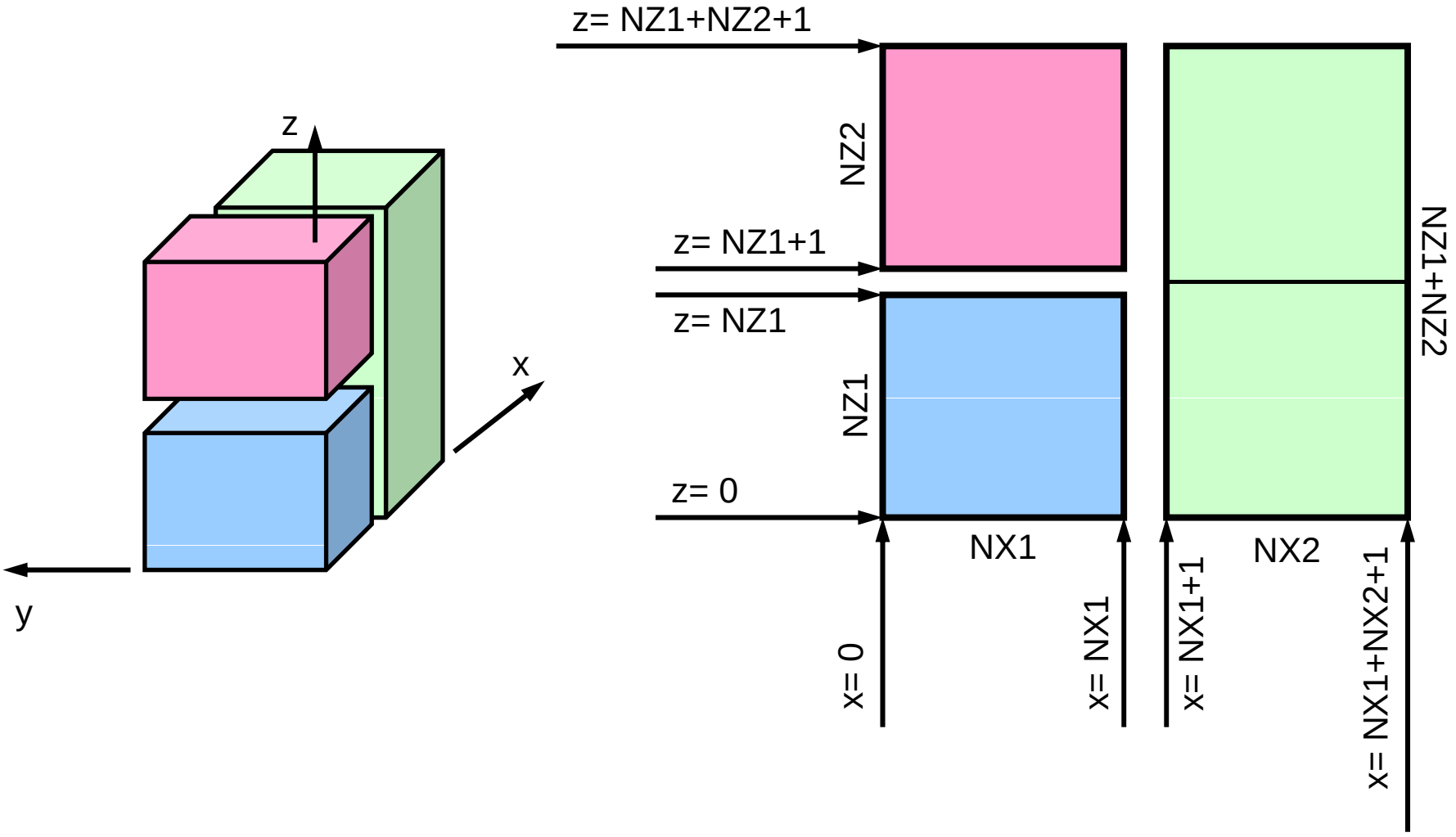
BILU(1)/BILU(2)との比較

- $[M]^{-1}[A]$ 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.

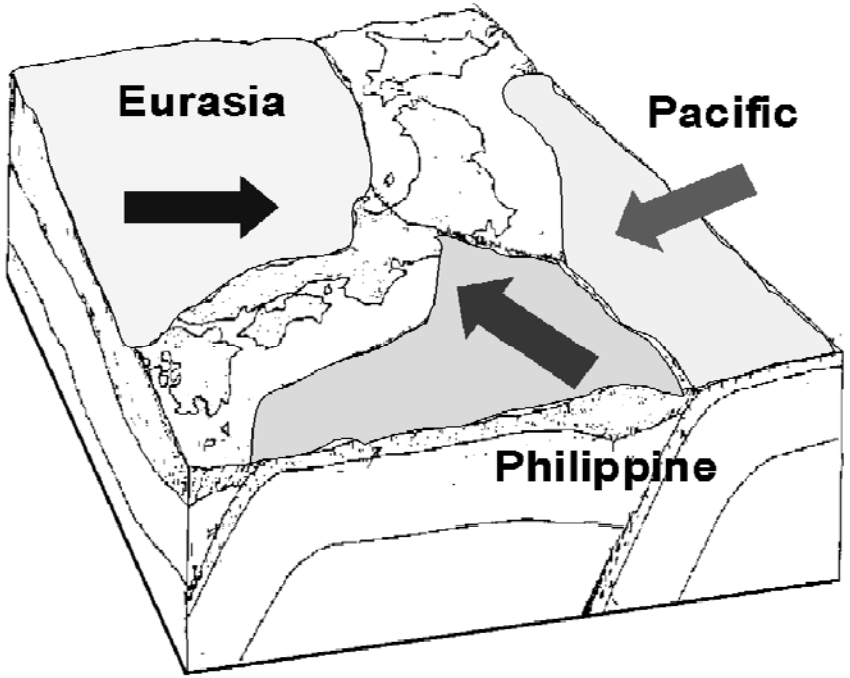
Simple Block Model ($\kappa = E_{max}/E_{min}$)：条件数

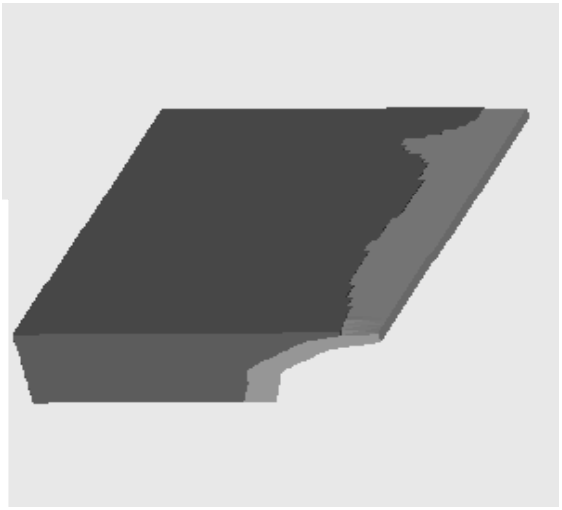
κ 条件数
E_{max} 最大固有値
E_{min} 最小固有値

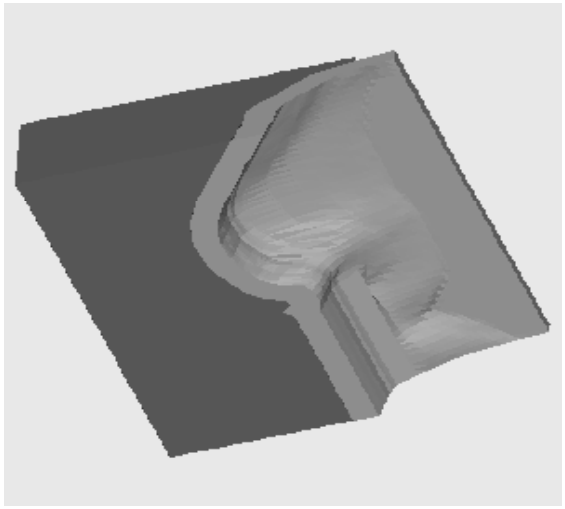
Preconditioning		$\lambda=10^2$	$\lambda=10^6$	$\lambda=10^{10}$
BIC(0)	E_{min}	4.845568E-03	4.865363E-07	4.865374E-11
	E_{max}	1.975620E+00	1.999998E+00	2.000000E+00
	κ	4.077170E+02	4.110686E+06	4.110681E+10
BIC(1)	E_{min}	8.901426E-01	8.890643E-01	8.890641E-01
	E_{max}	1.013930E+00	1.013863E+00	1.013863E+00
	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

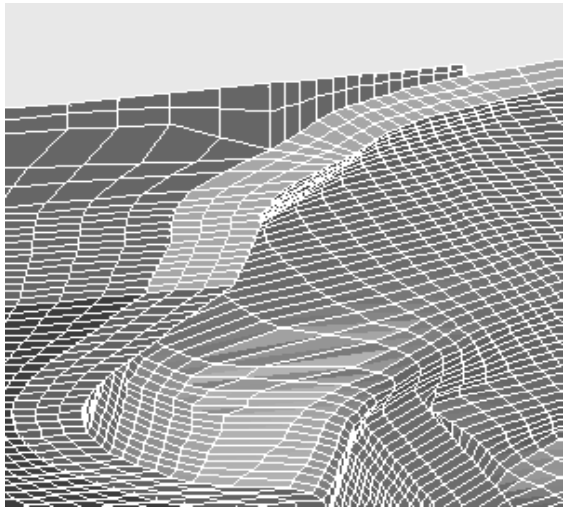
South-West Japan (SWJ)

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

並列ILU,並列IC

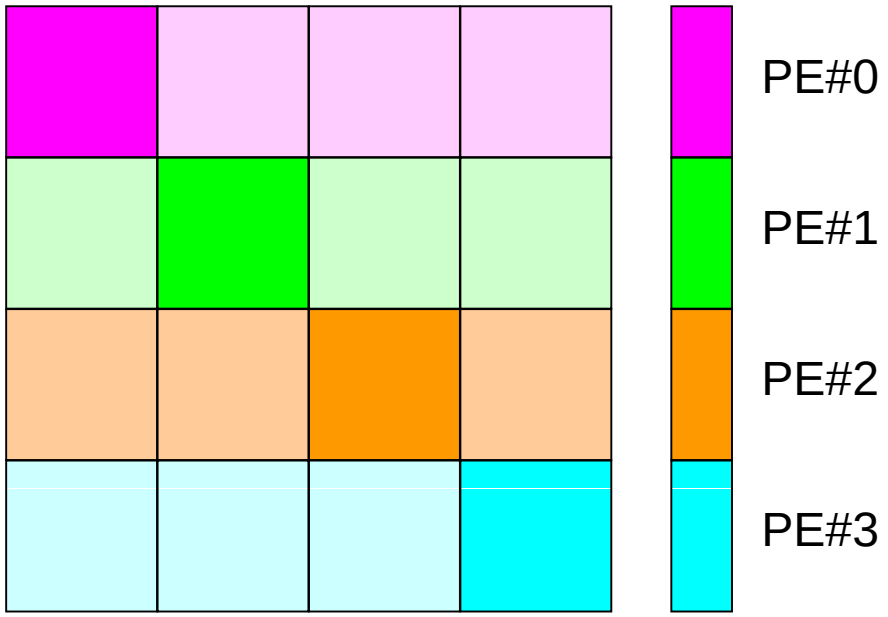
IC分解，ILU分解は本来並列化がしにくい処理である

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
```

並列ILU,並列IC

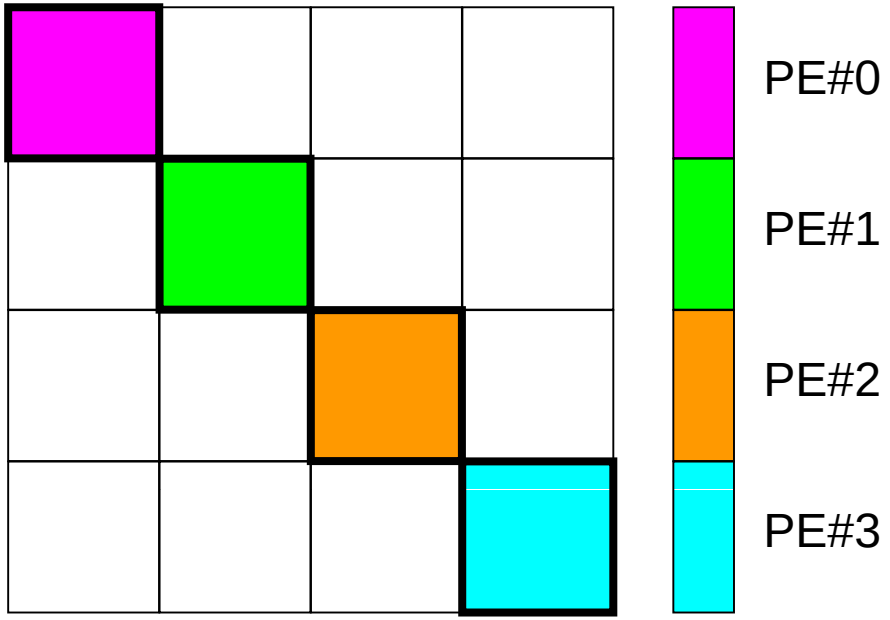
IC分解，ILU分解は本来並列化がしにくい処理である




```
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
```

- グローバルな計算が必要
- まともに計算しようとすると，通信が非常に多いプログラムになってしまう

局所化処理：ブロックJacobi



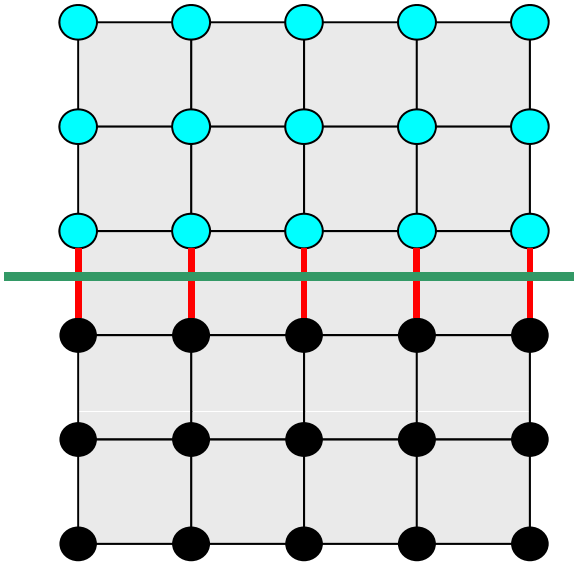

```
LEV_ij=0 if ((i,j)∈ NonZero(A)) otherwise LEV_ij= p+1
do i= 2, n
  do k= 1,  i-1
    if (LEV_ik≦p) then
      a_ik := a_ik/a_kk
    endif
    do j= k+1, n
      if (LEV_ij = min(LEV_ij, 1+LEV_ik+ LEV_kj)≦p) then
        a_ij := a_ij - a_ik*a_kj
      endif
    enddo
  enddo
enddo
```

- 前処理時には領域外からの影響を考慮しない
  - 並列性は高まるが，前処理としては「弱く」なる
  - 反復回数が増加する可能性あり

領域分割法の工夫

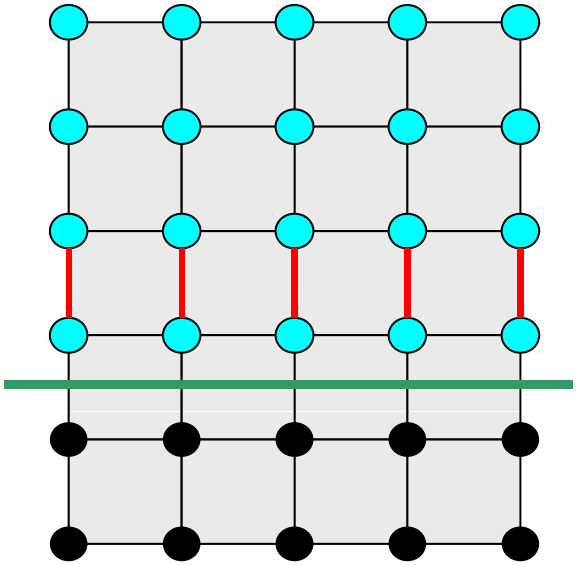
Selectiveブロック内の節点が違う領域に分割されると収束が遅くなる。

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



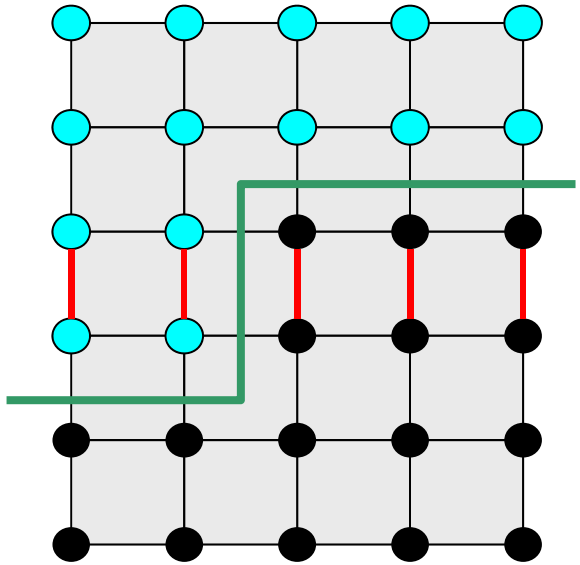
BEFORE repartitioning

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

Results on Hitachi SR2201 (U.Tokyo)

Parallel Performance of SB-BIC(0)-CG

NX1=NX2=70, NY=40, NZ1=NZ2=70, **Repartitioned.**

2,471,439 DOF, 784,000 Elements, $\lambda/E=10^6$

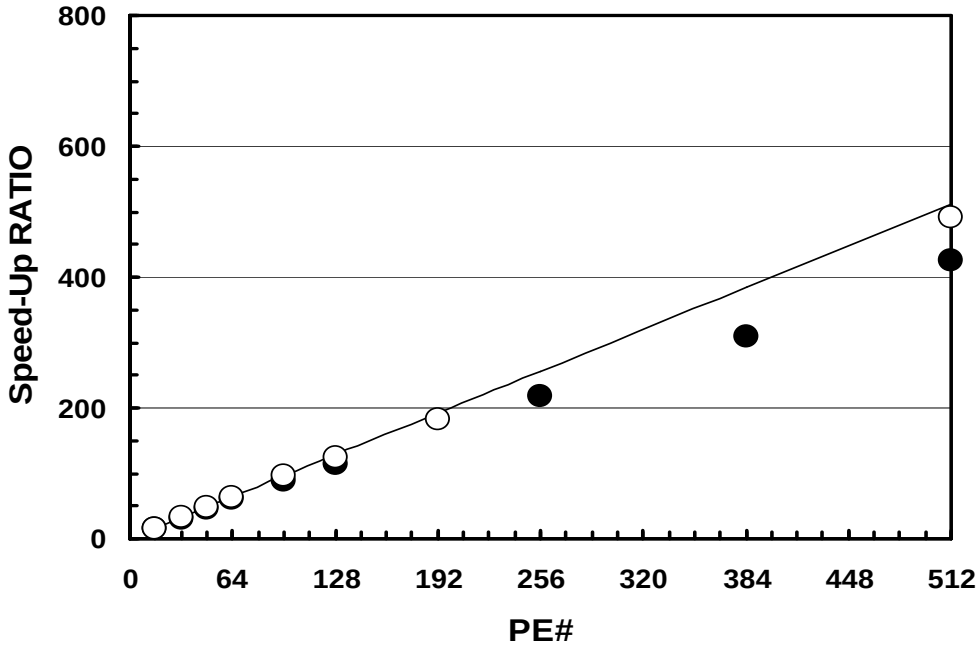
Iterations/CPU time until convergence ($\varepsilon=10^{-8}$)

Precon-ditioning		16 PEs	32 PEs	48 PEs	64 PEs	96 PEs	144 PEs	192 PEs	256 PEs	Memory Size (GB)
BIC(0)	Iterations	14459	14583	15018	15321	15523	15820	16084	16267	
	sec.	13500	7170	4810	3630	2410	1630	1270	1230	3.10
	Speed-up	16	30	45	60	90	133	170	211	
BIC(1)	Iterations			379	390	402	424	428	452	
	sec.	N/A	N/A	236	175	119	81	62	48	8.39
	Speed-up			48	65	95	140	183	236	
BIC(2)	Iterations					364	387	398	419	
	sec.	N/A	N/A	N/A	N/A	212	140	112	86	14.4
	Speed-up					96	145	182	217	
SB-BIC(0)	Iterations	511	524	527	538	543	567	569	584	
	sec.	555	295	193	144	96	64	48	38	3.52
	Speed-up	16	30	46	62	92	139	185	235	

Parallel Performance of SB-BIC-CG

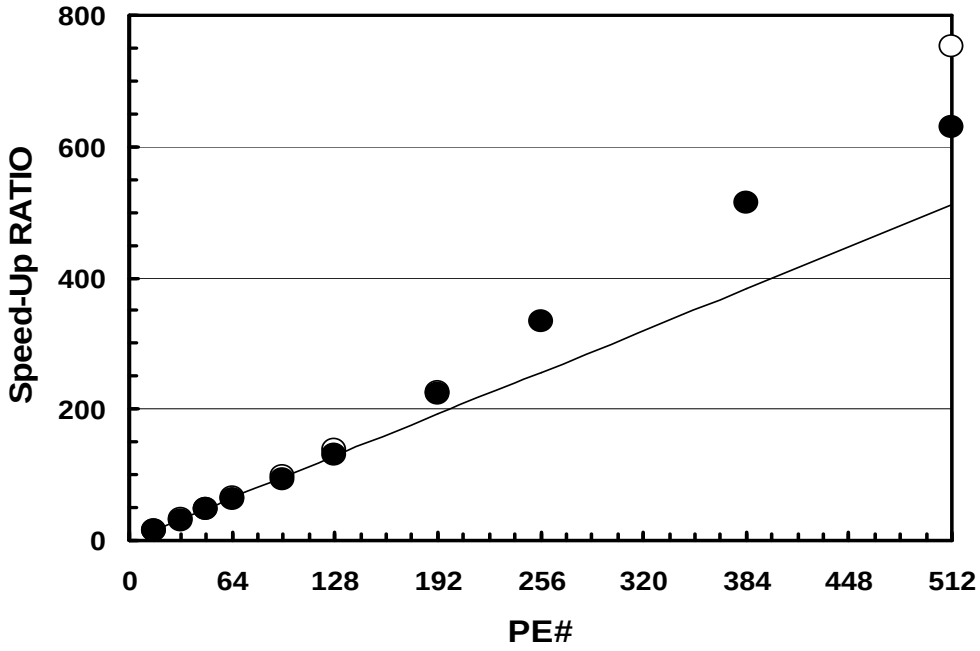
$\lambda/E=10^6$, 16-512 PEs, Entire Prob. Size Fixed.

IBM BG/L Prototype

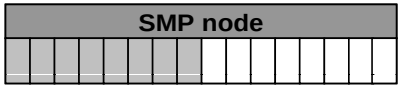


● Simple Block(2,471,439 DOF)
○ SWJ(2,992,266 DOF)

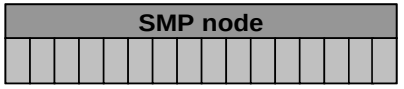
IBM SP-3/Seaborg: type-2A



Type-2A
8 of 16 PE's are used

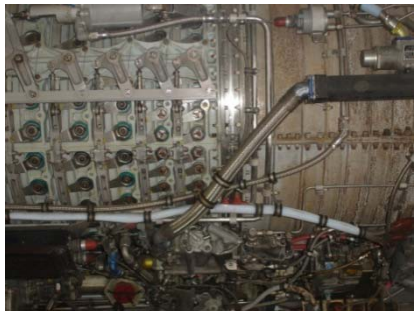


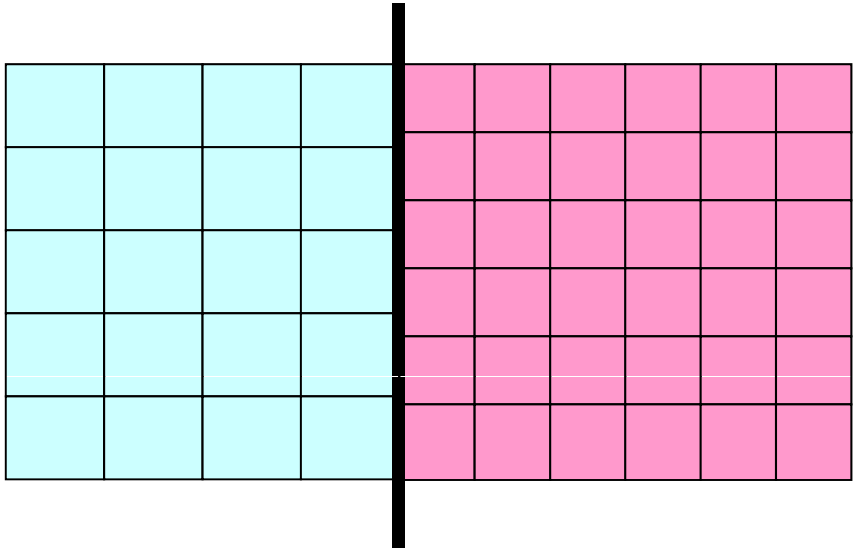
Type-2C
16 of 16 PE's are used
Problem size/PE is
as same as that of Type-2A.



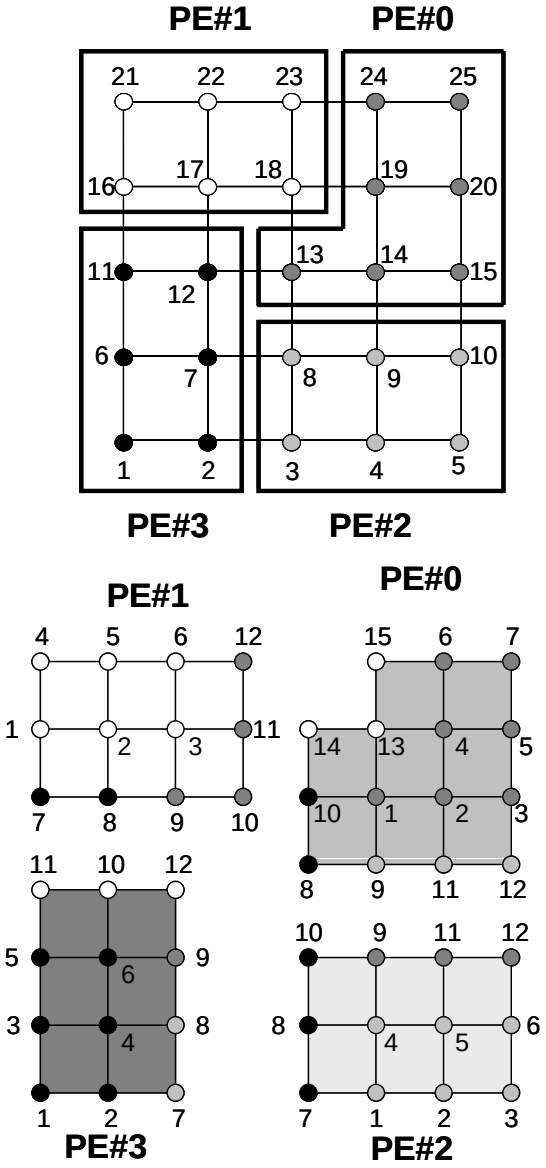
より一般的な問題

- 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。

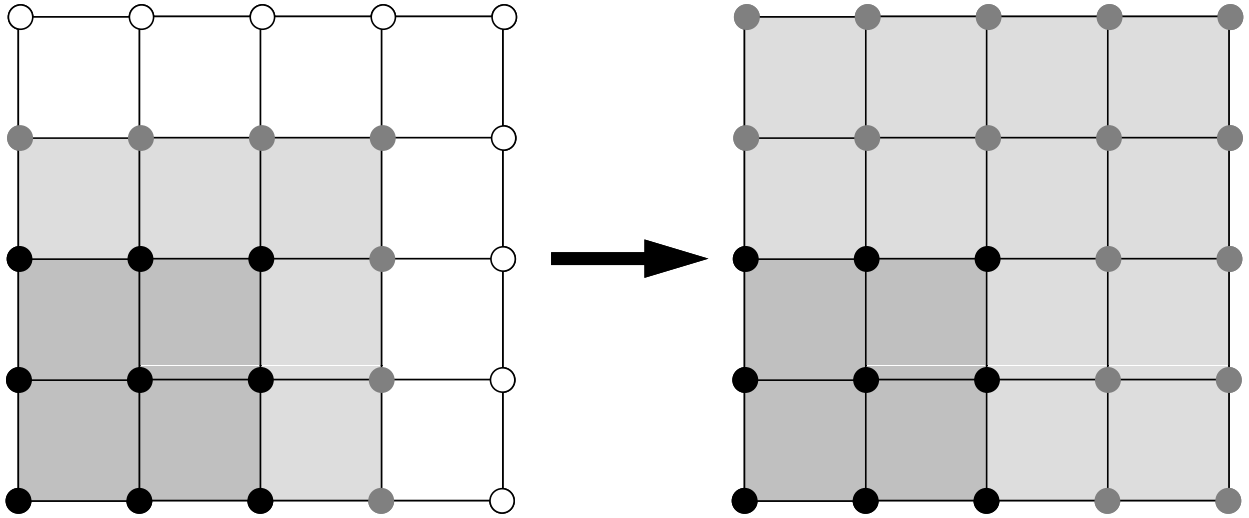




対策：領域間オーバーラップの拡張



計算量，通信量は増加



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

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