内容に関するご質問は
ida@cc.u-tokyo.ac.jp
まで、お願いします。

第62回　お試しアカウント付き
並列プログラミング講習会
「ライブラリ利用：科学技術計算の効率化入門」

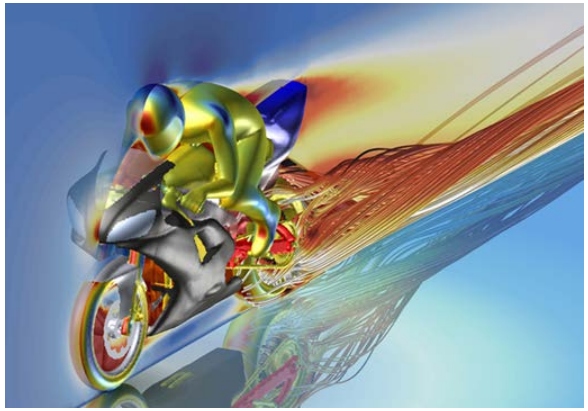
階層型行列法とℋACApKライブラリ

東京大学情報基盤センター　特任准教授　伊田 明弘

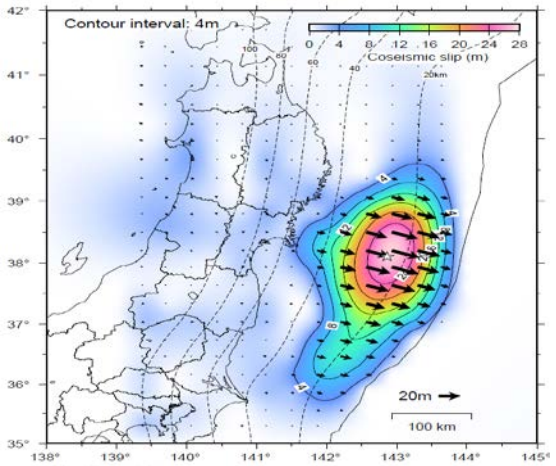
チュートリアルの流れ

1. ppOpen-APPL/BEM
2. 境界要素法とBEM-BBフレームワーク
3. プログラム実習 Ⅰ
   - BEM-BB
4. 階層型行列法と$\mathcal{H}$ACApKライブラリ
5. プログラム実習 Ⅱ
   - BEM-BB+ $\mathcal{H}$ACApK

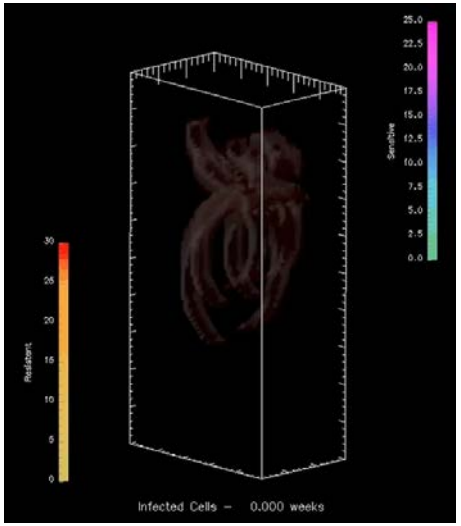
多次元空間でシミューレートする



・Transcribed from **VINAS co.LTD**



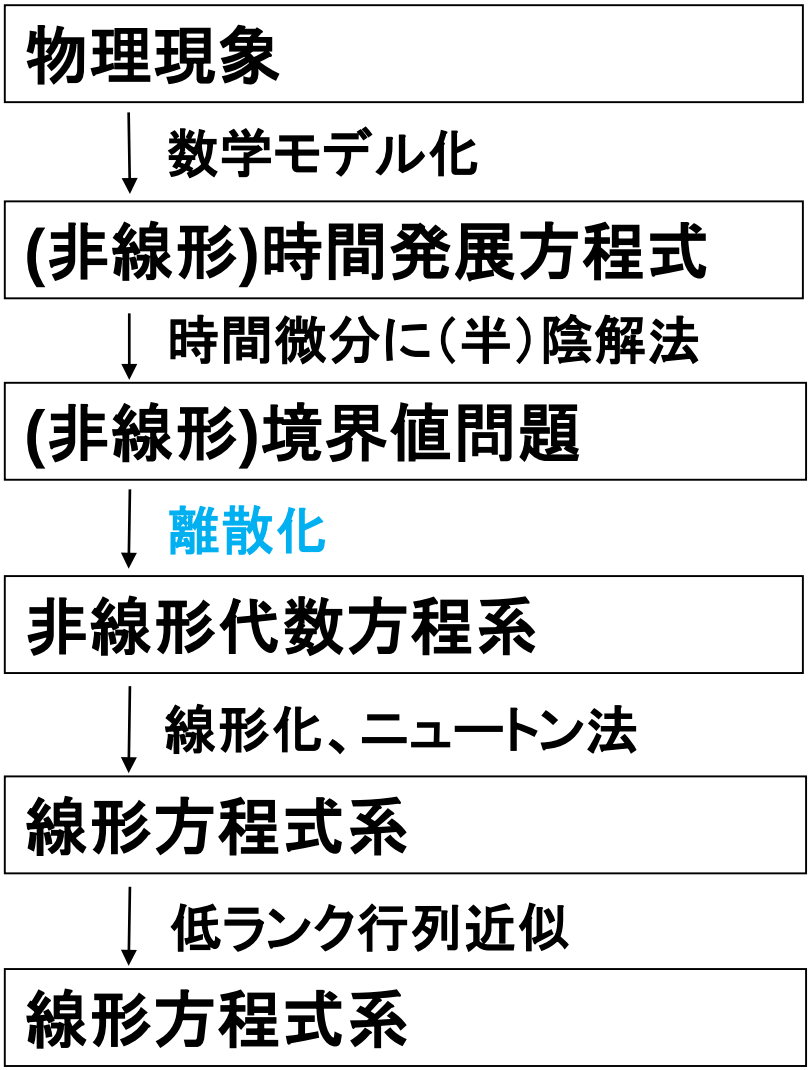
・Transcribed from **Ohtani at al (2011)**



どうやって？

科学技術計算の定式化例

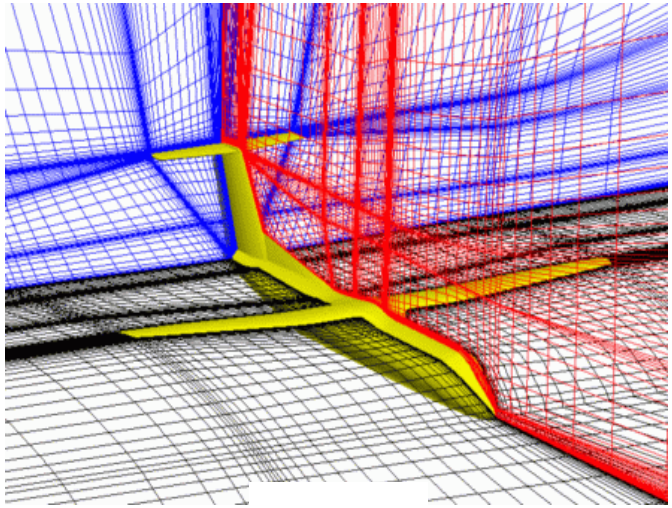
■どんどん近似していき，どこかで観念して計算する



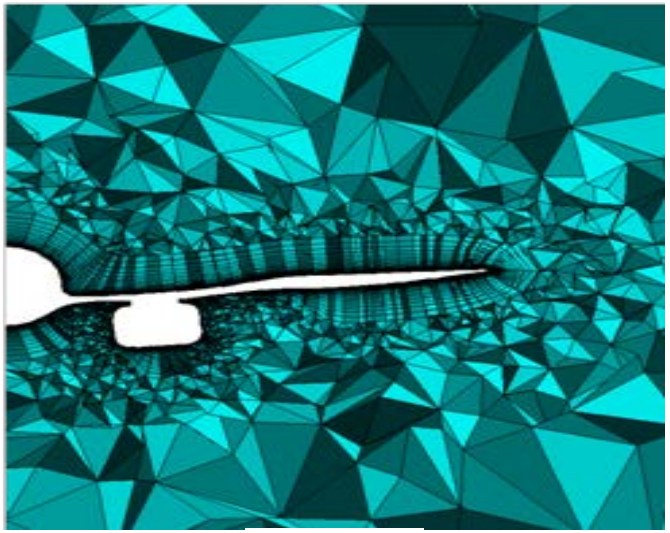
離散化手法の選択が重要な分かれ目

この辺りがスパコンの出番

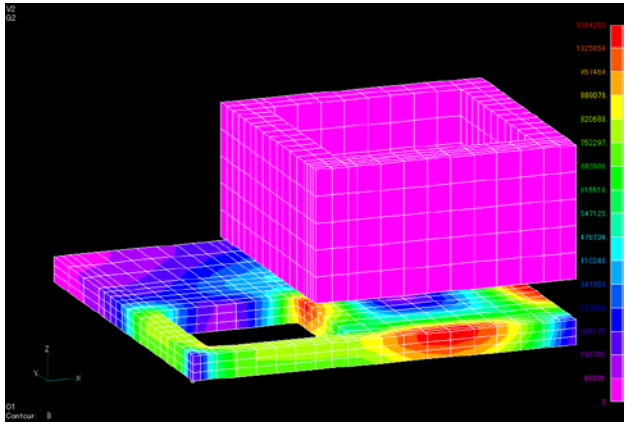
Mesh used in FDM, FEM, BEM

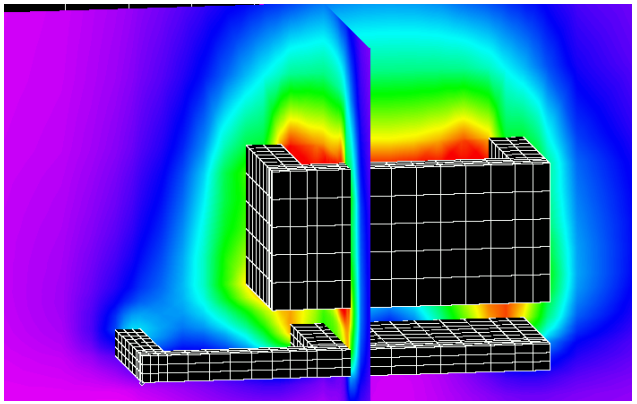


FDM



FEM





BEM

Pictures are transcribed from

FDM: de Havilland Aircraft Company, VINAS Co. Ltd
FEM: Prof. Nakazima (Tokyo Univ.)
BEM: μ-TEC Co. Ltd

FDM, FEM, BEM for Elliptic PDE

continuous | discrete

For $x \in \Omega \subset \mathbb{R}^3$, find $\phi(x)$ s.t.

$$\Delta\phi(x) = f(x)$$

$$\mathrm{Az} = b$$

- Bi-linear form

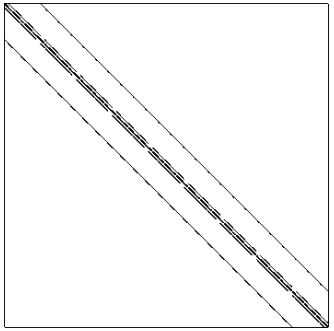
$$\int_\Omega \psi(x)\Delta\phi(x)\mathrm{d}x = \int_\Omega \psi(x)\rho\,\mathrm{d}x$$

- Green's identity

$$\int_\Omega \psi(x)\Delta\phi(x)\mathrm{d}x = \int_{\partial\Omega} \psi(x)\phi_n(x)\,\mathrm{d}x - \int_\Omega \nabla\psi(x)\cdot\nabla\phi(x)\mathrm{d}x$$

FDM

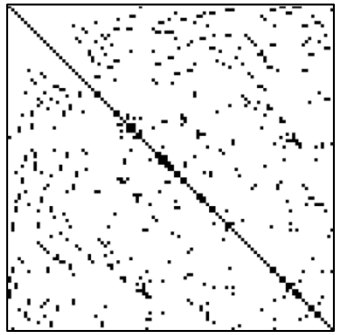
- Power series



$$\int_\Omega \nabla\psi(x)\cdot\nabla\phi(x)\mathrm{d}x = \langle\psi, \bar{f}\rangle$$

FEM

- Base function



- Fredholm integral equation

$$\int_{\partial\Omega} \bar{\psi}(x,y)\phi_n(x)\,\mathrm{d}x = \bar{\bar{f}}$$

$$\int_{\partial\Omega} \varphi(y)\int_{\partial\Omega} \bar{\psi}(x,y)\phi_n(x)\,\mathrm{d}x\mathrm{d}y = \langle\varphi, \tilde{f}\rangle$$

BEM

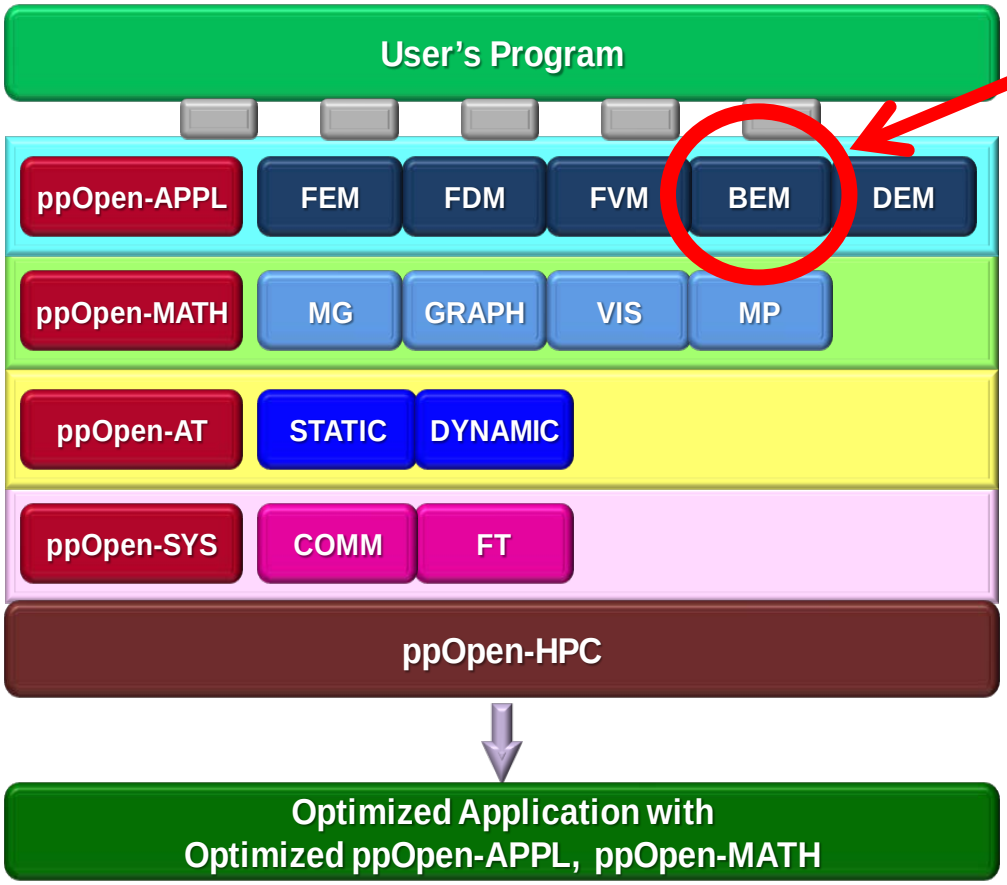
- Base function

■

概要：ppOpen-HPC

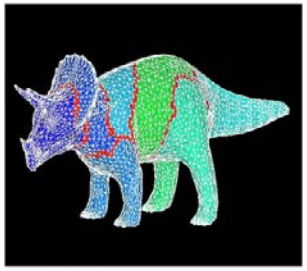
■JST-CREST
- “ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出”領域
- “自動チューニング機構を有するアプリケーション開発・実行環境**ppOpen-HPC**”
  研究代表者: 中島 研吾(東京大学情報基盤センター)

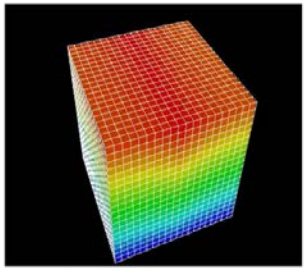
■Download site: http://ppopenhpc.cc.u-tokyo.ac.jp/



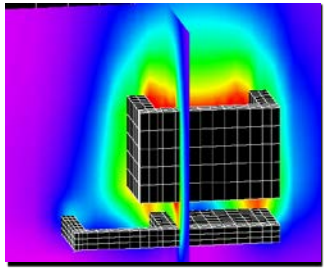
このチュートリアルで使用

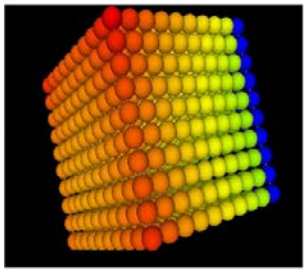
ppOpen-HPC covers …


FEM
Finite Element Method


FDM
Finite Difference Method


FVM
Finite Volume Method


BEM
Boundary Element Method

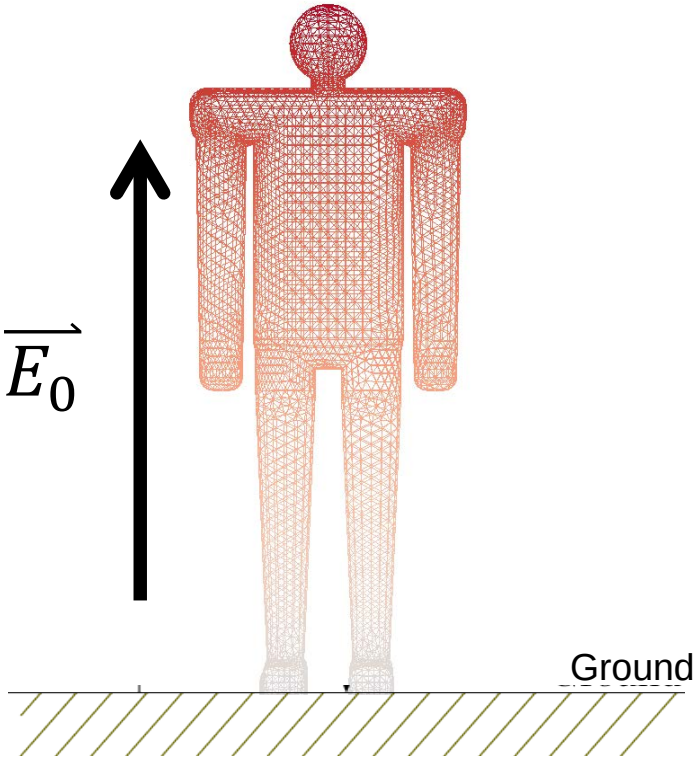

DEM
Discrete Element Method

ppOpen-APPL/BEMを用いた解析例

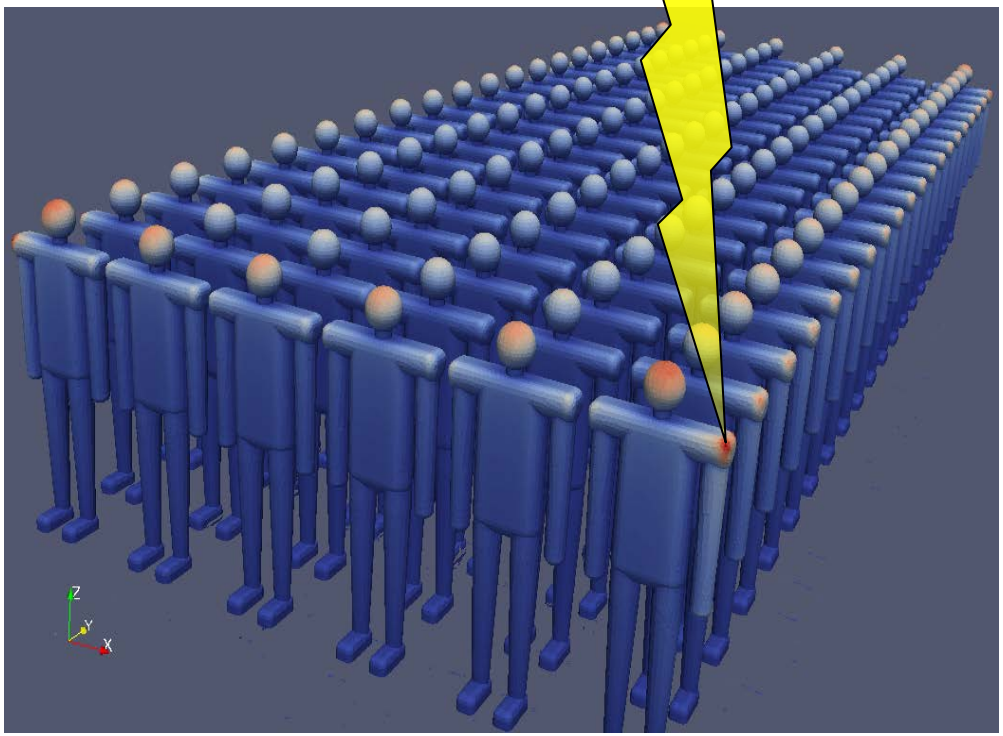
■静電場解析

・Potential operator: $\mathcal{V}[u](x) := \int_\Gamma \frac{u(y)}{4\pi\|x-y\|}$

・半無限解析領域において物体の表面電化を計算する



解析条件

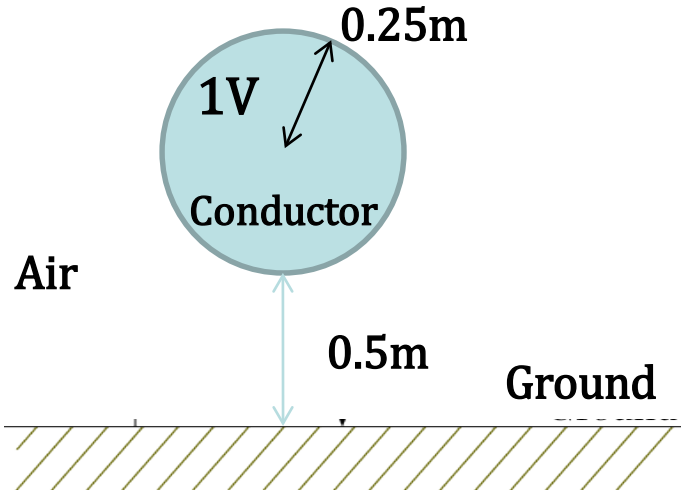


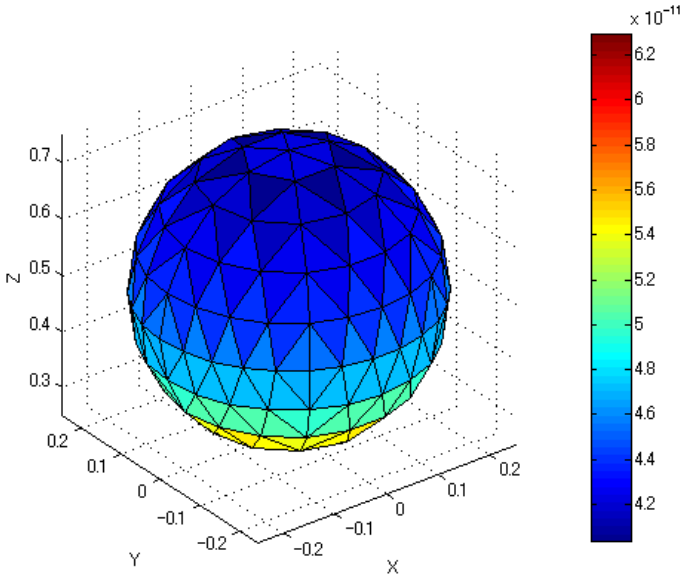
解析結果

ppOpen-APPL/BEMを用いた解析例

■静電場解析

・Potential operator : $\mathcal{V}[u](x) := \int_\Gamma \frac{1}{4\pi\|x-y\|} u(y)dy, \quad x \in \Gamma$





解析条件

・半径0.25mの導体球に1Vを与える
・表面に現れる電荷を求める
（半無限領域内の電位が導出される）

解析結果

導体球の下半分に大きな電荷密度が生じる

ppOpen-APPL/BEMの構成

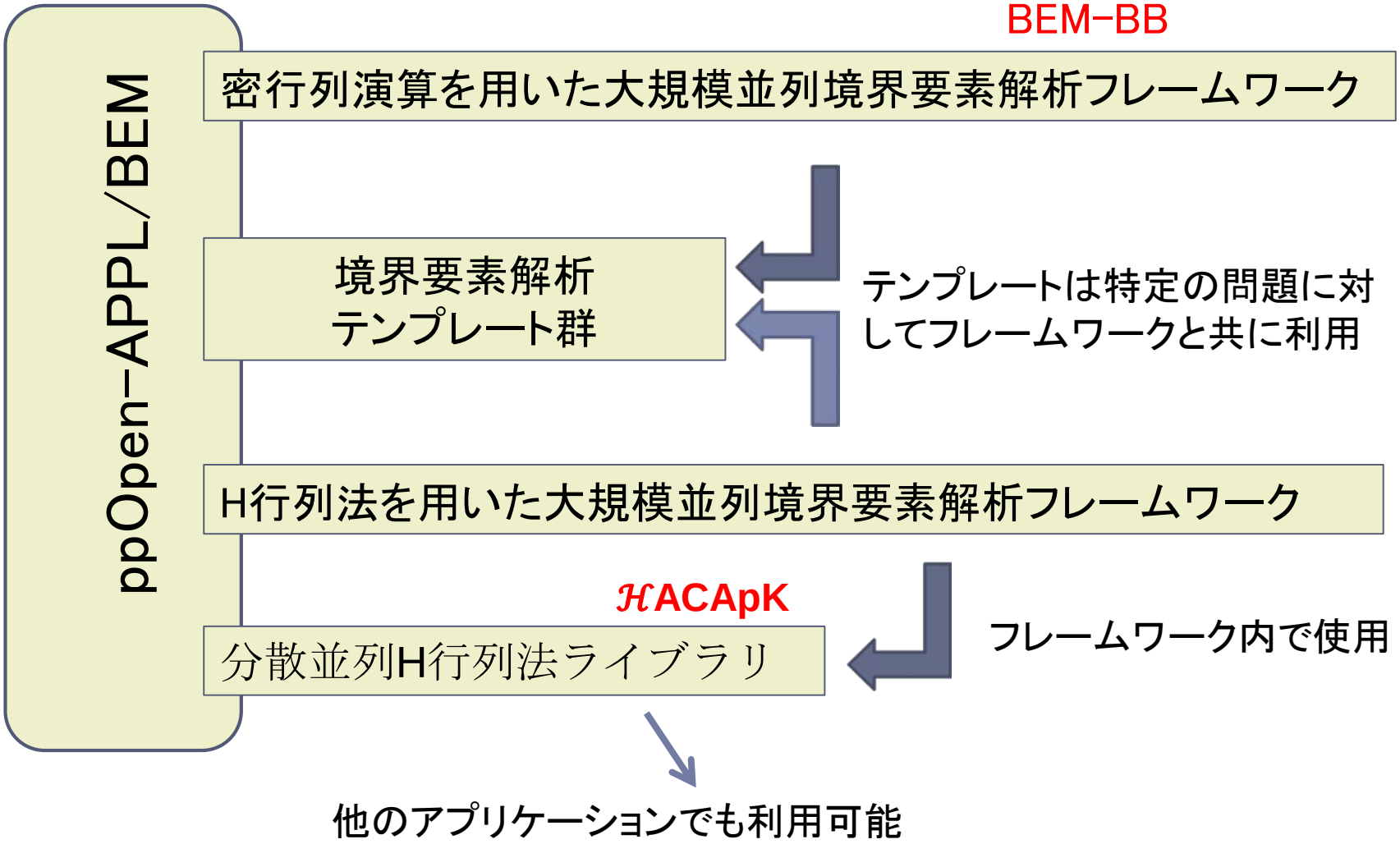
■BEM－BB

・境界要素法・積分方程式法用ソフトウェアフレームワーク
（ライブラリ、API集ではない）
・使用法：境界要素積分関数をユーザが用意しBEM-BBに埋め込む
（ユーザプログラムから必要な関数を呼ぶのではない）
・密行列演算利用版、階層型行列（H-matrix）利用版
・Fortran90, MPI+OpenMP, BLAS

■HACApK

・階層型行列法ライブラリ
・使用法：ユーザプログラムから必要な関数を呼ぶ
HACApKから呼ばれる境界要素積分関数を用意しておく
・Fortran90, MPI+OpenMP

ppOpen-APPL/BEMの構成



チュートリアルの流れ

1. ppOpen-APPL/BEM
2. 境界要素法とBEM-BBフレームワーク
3. プログラム実習 Ⅰ
   - BEM-BB
4. 階層型行列法と$\mathcal{H}$ACApKライブラリ
5. プログラム実習 Ⅱ
   - BEM-BB+ $\mathcal{H}$ACApK

境界要素法(積分方程式法)

- **偏微分方程式の境界値問題**に対する代表的な**離散化解法**の一つ
- モデルの境界面での離散化のみで解析を実行
  - 少ない未知変数で大規模なモデルを扱うことが可能
  - 開領域問題，形状最適化，移動物体問題に有効
- 一般的には**密行列計算**が必要
  - 実応用解析では，FMMやH行列法等の高速行列演算技術の利用が不可欠
- ターゲットアプリケーション
  - 電磁場解析
  - 地震サイクルシミュレーション

境界要素積分方程式の離散化

■密行列を係数行列に持つ線形方程式系が得られる.

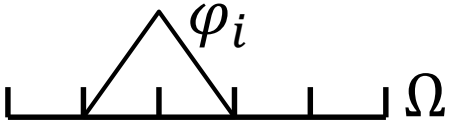
$$\mathcal{g}[u] = f, \text{ where } \mathcal{g}[u](x) := \int_\Omega \kappa(x,y)\, u(y)\mathrm{d}y$$

singular kernel: $\kappa(x,y) \in \mathrm{span}(\{|x-y|^{-p}, p>0\})$

e.g. $\kappa(x,y) \propto |x-y|^{-1}$

Discretization by W.R.M., e.g. Ritz-Galerkin

$$u \cong u^h = \sum_{i \in \mathcal{I}} \phi_i \varphi_i$$



$$A\phi = b,$$

$$A_{ij} = \int_\Omega \varphi_i(x) \int_\Omega g(x,y)\, \varphi_j(y)\mathrm{d}y\mathrm{d}x, \quad b_i = \int_\Omega f(x)\, \varphi_i(x)\mathrm{d}x$$

A：密行列

境界要素解析の手順

1. モデルデータの入力
2. 係数行列の作成
   - 要素間（数値／解析）積分の実行
3. 右辺ベクトルの作成
   - 境界条件の適用
4. 連立一次方程式の求解
   - 密な係数行列を使用
5. 解析結果の出力

BEM-BB
フレームワーク

BEM-BBの利用目的

- **想定利用ユーザ**
  - 主：境界要素法を使用し，大規模並列計算環境で効率的に動作するシミュレーションプログラムを開発する研究者・技術者
  - 副：特定のアプリケーション領域の問題を境界要素法により解く研究者・技術者
    - 商用のアプリケーションソフトウェアと同様にモデルに関する情報を入力するだけで必要な解を得たい。
    - PC等の逐次計算環境や小規模なクラスタ、マルチコアプロセッサでも動作することが望ましい
      - 大規模計算環境へのシームレスな移行

BEM-BBの設計思想

- 計算科学者が自らのアイデアを反映できる余地が残されていなければならない
  - サポートの範囲を限定する
    - 要素間積分には様々な方法があり，新しい提案も今後あり得る.
  - 並列処理はやはり面倒
    - 必要なプロセス間通信は提供されるソフトウェアが正しく，ブラックボックス的にやってくれればうれしい

BEM-BBの機能とユーザの準備

■Framework for parallel BEM analyses

＜Frame work＞

Model data input

Making matrix

・Call user function

Liner solver

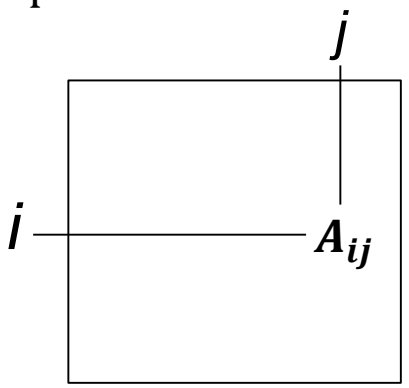
on distributed memory
parallel computer systems

＜user＞

Input file
・mesh, physical parameters and so on

Boundary element integral
・Function to calculate entry A_{ij}

e.g. : real*8 function entry_ij(i, j, param)
integer :: i, j
type(phys) :: param



OpenMP/MPI ハイブリッド

並列処理の活用

BEM-BBの機能とユーザの準備

■Framework for parallel BEM analyses

＜Frame work＞

Model data input

Making matrix

・Call user function

Liner solver

on distributed memory
parallel computer systems

OpenMP/MPI ハイブリッド

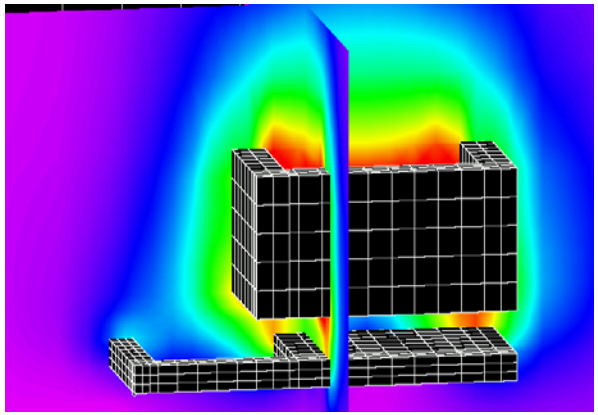
並列処理の活用

＜user＞

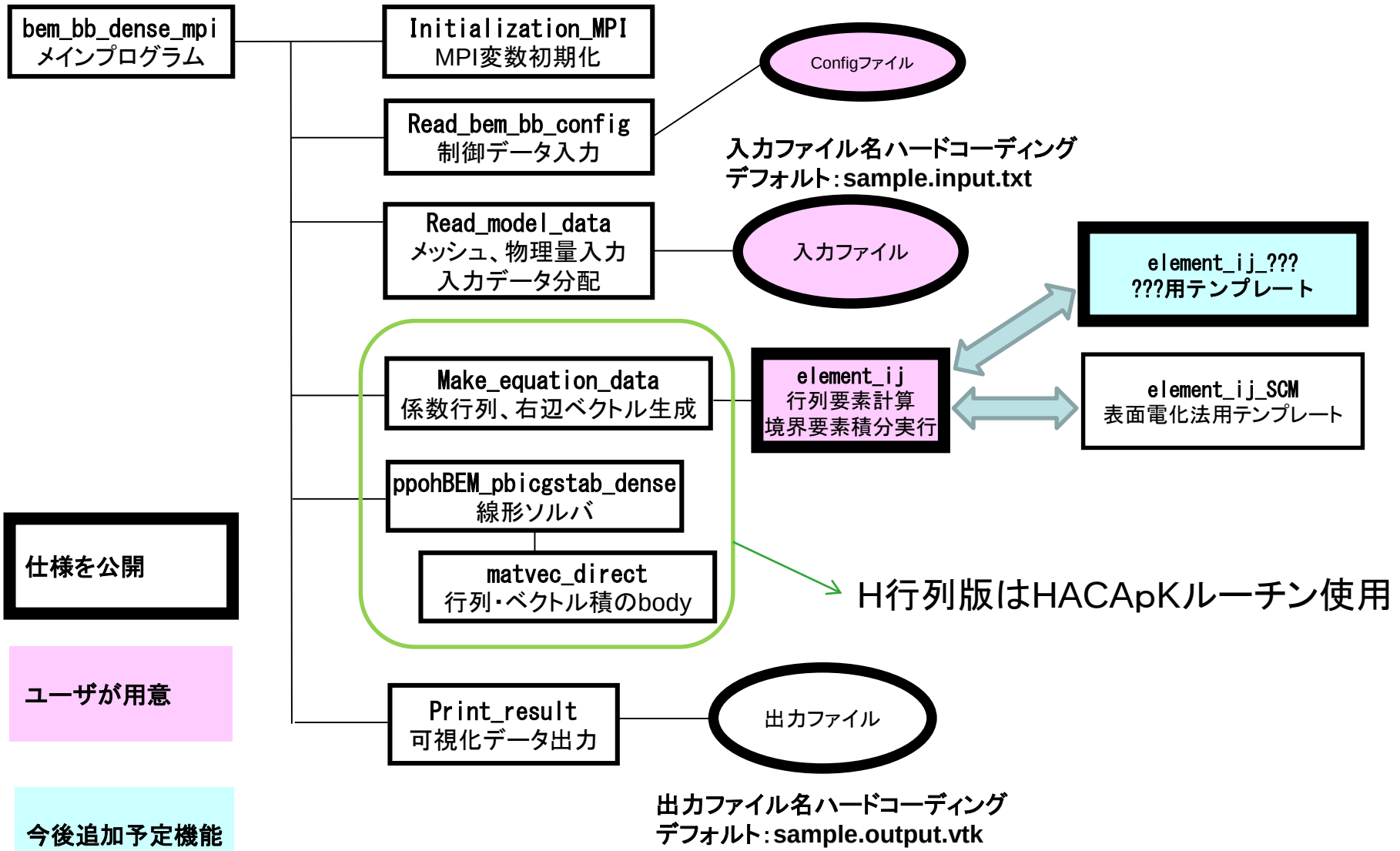
Input file
・mesh, physical parameters and so on

Templates
・specific application domains

■Static electric filed



BEM-BB構成


bem_bb_dense_mpi
メインプログラム
Initialization_MPI
MPI変数初期化
Configファイル
Read_bem_bb_config
制御データ入力
入力ファイル名ハードコーディング
デフォルト:sample.input.txt
Read_model_data
メッシュ、物理量入力
入力データ分配
入力ファイル
element_ij_???
???用テンプレート
Make_equation_data
係数行列、右辺ベクトル生成
element_ij
行列要素計算
境界要素積分実行
element_ij_SCM
表面電化法用テンプレート
ppohBEM_pbicgstab_dense
線形ソルバ
matvec_direct
行列・ベクトル積のbody
H行列版はHACApKルーチン使用
仕様を公開
ユーザが用意
今後追加予定機能
Print_result
可視化データ出力
出力ファイル
出力ファイル名ハードコーディング
デフォルト:sample.output.vtk

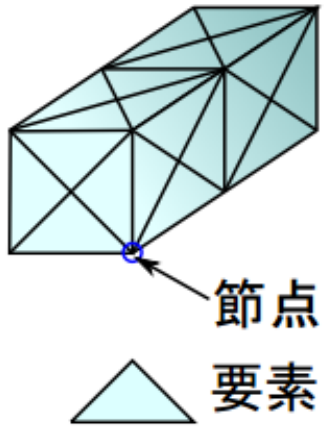
BEM-BBの入力データ形式

■3次元空間上の多角形メッシュで表現される面を想定

■入力データファイル（拡張子 .pbf）フォーマット

(i) Number of nodes (1integer*4 number): nond
(ii) Coordinates of the nodes in three dimensions (3 real*8 numbers for each node)
(ii) Number of faces (1integer*4 number): nofc
(iv) Number of nodes on each face (1 integer*4 number): nond_on_fc
(v) Number of integer parameters set on each face (1 integer*4 number): nint_para_fc
(vi) Number of real parameters set on each face (1 integer*4 number): ndble_para_fc
(vii) Node number constructing for each face (nond_on_fc x nofc integer*4 numbers)
(viii) Integer parameters set on faces (nond_on_fc x nint_para_fc integer*4 numbers)
(ix) Real parameters set on faces (nond_on_fc x ndble_para_fc real*8 numbers)

```
4
0.000000000000e+00 0.000000000000e+00 0.000000000000e+00
0.000000000000e+00 1.000000000000e+00 0.000000000000e+00
1.000000000000e+00 1.000000000000e+00 0.000000000000e+00
2.000000000000e+00 0.000000000000e+00 2.000000000000e+01
2
3
0
1
1 2 3
2 4 3
0.000000000000e+00
1.000000000000e+00
```



Configファイル

■入力データに含まれていない制御パラメータを指定

(i) number_element_dof (integer*4)
(ii) linear_solver (character*16)
(iii) tor (real*8)
(iv) max_steps (integer*4)

```
1
BICGSTAB
1d-8
5000
```

付属configファイル

ユーザ定義関数 element_ij

■2つの要素番号と各要素上の情報を受け取り、行列要素値を返す

```
real(8) function matrix_element_ij(i, j, nond, nofc, nond_on_fc, np,int_para_fc,
                nint_para_fc, dble_para_fc, ndble_para_fc, face2node)

type :: coordinate
   real(8) :: x ,y ,z
end type coordinate

integer,intent(in) :: i,j,nond,nofc,nond_on_fc,nint_para_fc,ndble_para_fc

type(coordinate), intent(in) :: np(*) ! 節点座標

integer,intent(in) :: face2node(nond_on_fc,*), ! 要素を構成する節点
                      int_para_fc(nint_para_fc,*) ! 要素上にintで与えられている情報

real(8),intent(in) :: dble_para_fc(dble_para_fc,*) !要素上にreal8で与えられている情報
```

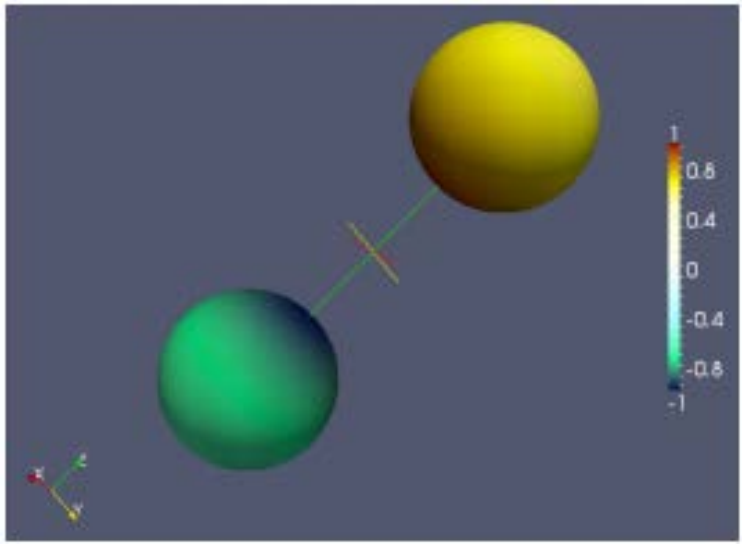
BEM-BBによる解析結果の可視化

■オープンソース可視化ソフトウェアParaViewに対応

・VTK形式ファイルを出力

```
# vtk DataFile Version 2.0   ] (1)
Title(max:256 characters)    ] (2)
ASCII or BINARY              ] (3)
DATASET type
...                          ] (4)
...
POINT_DATA n
...
...                          ] (5)
CELL_DATA n
...
...
```

図 C.1: VTK ファイル形式



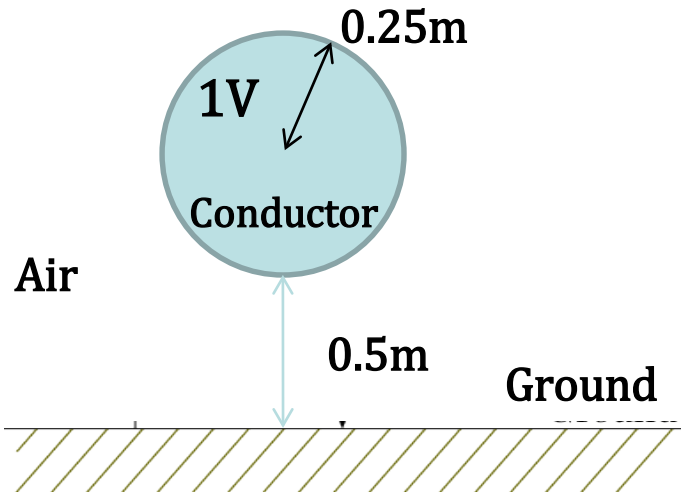
チュートリアルの流れ

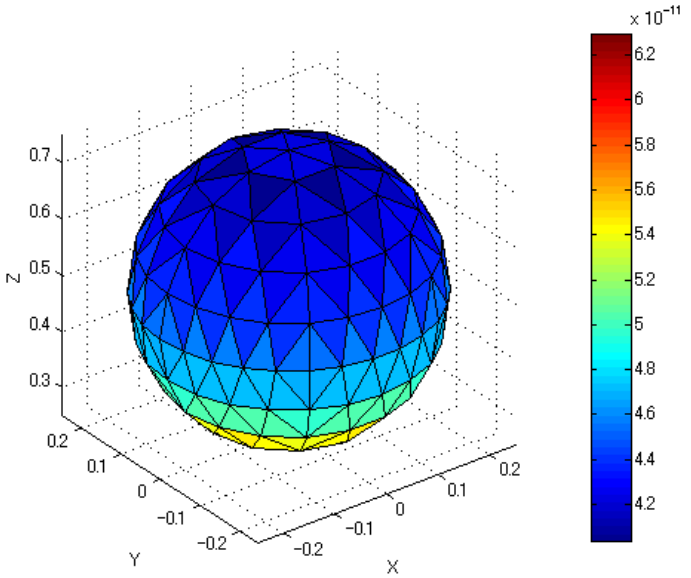
1. ppOpen-APPL/BEM
2. 境界要素法とBEM-BBフレームワーク
3. プログラム実習 Ⅰ
   - BEM-BB
4. 階層型行列法と$\mathcal{H}$ACApKライブラリ
5. プログラム実習 Ⅱ
   - BEM-BB+ $\mathcal{H}$ACApK

サンプルプログラムで計算する問題

■静電場解析

・Potential operator : $\mathcal{V}[u](x) := \int_\Gamma \frac{1}{4\pi\|x-y\|} u(y)dy, \quad x \in \Gamma$





解析条件

・半径0.25mの導体球に1Vを与える
・表面に現れる電荷を求める
（半無限領域内の電位が導出される）

解析結果

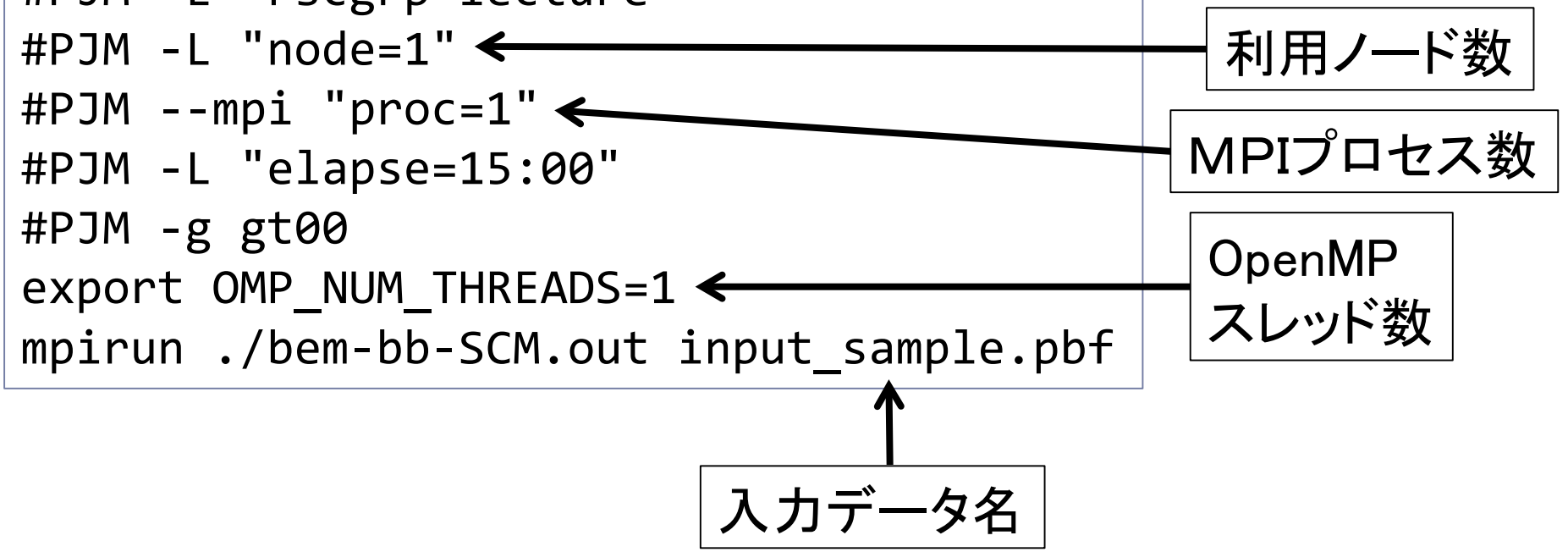
導体球の下半分に大きな電荷密度が生じる

実習を行うあたっての注意点(BEM-BB)

- ファイル名
  ppohBEM_0.4.1.tar.gz (Oakleaf-fxのみ)
- ジョブスクリプトファイルbem-bb.bash中のキュー名を
  lecture から tutorial に変更してから
  pjsubしてください。
  - lecture : 実習時間外のキュー(同時実行数1)
  - tutorial : 実習時間内のキュー(同時実行数4+)

BEM-BB用ジョブスクリプト(Oakleaf-fx)

```
#!/bin/bash
#PJM -L "rscgrp=lecture"
#PJM -L "node=1"
#PJM --mpi "proc=1"
#PJM -L "elapse=15:00"
#PJM -g gt00
export OMP_NUM_THREADS=1
mpirun ./bem-bb-SCM.out input_sample.pbf
```



ノード数、MPIプロセス数、OpenMPスレッド数の設定は
Oakleaf-fxは1ノードあたり16コアであることに注意して行う

BEM-BBの実行（Oafleaf-fx）

- Oakleaf-fxにログインする
- 以下のコマンドを実行する

```
$ cp /home/z30107/ppohBEM_0.4.1.tar.gz ./
$ tar xvzf ppohBEM_0.4.1.tar.gz
$ cd ppohBEM_0.4.1
$ cd bem-bb-framework_dense
$ cd src
$ cd framework_with_templates
$ make
$ pjsub bem-bb.bash
```

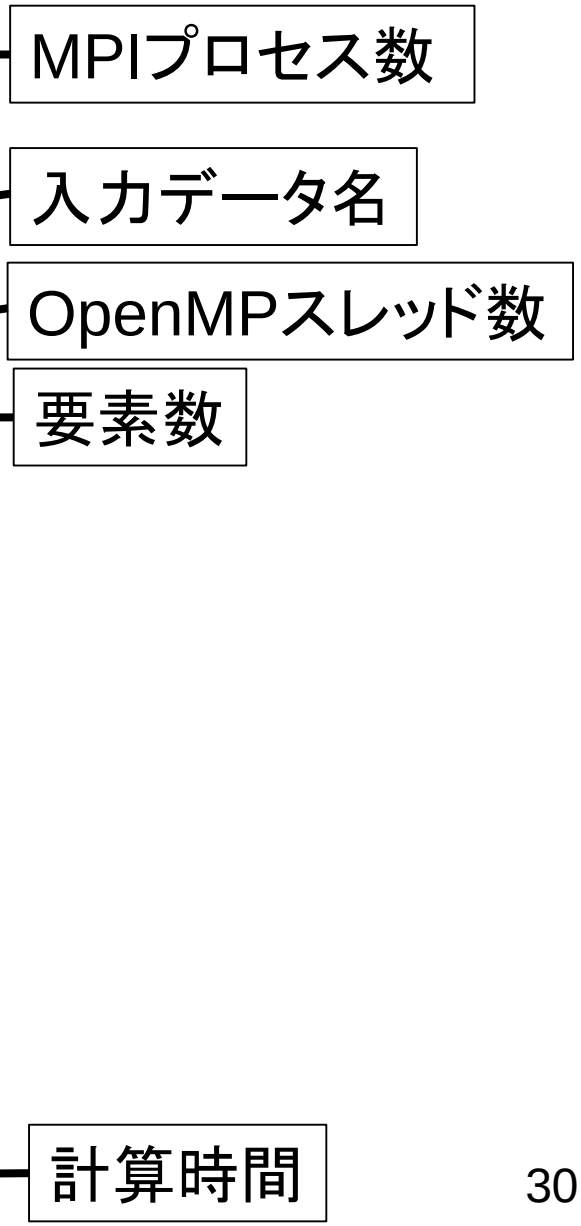
- 実行が終了したら、以下を実行する

```
$ cat bem-bb-fx.bash.oXXXXXX (XXXXXXは数値)
```

BEM-BBの実行（Oafleaf-fx）

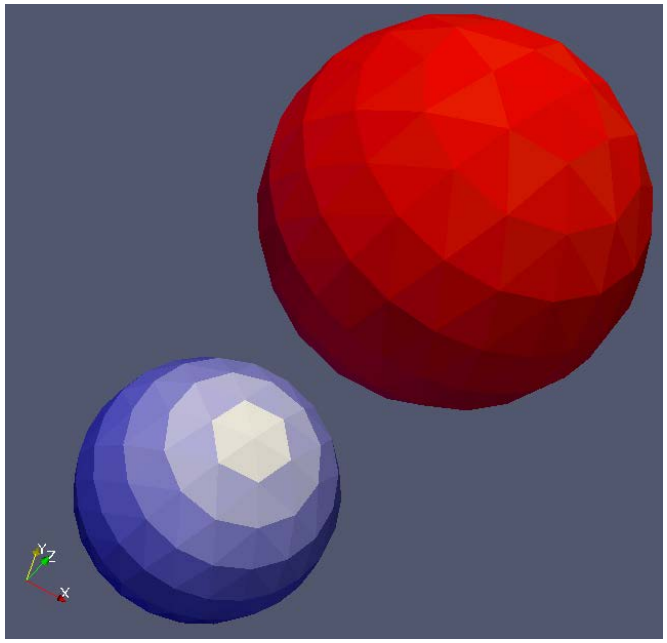
■以下のような結果が見えれば成功

```
 Number of processes  1
 Number of unknowns set on each face element =  1
 Selected linear solver = BICGSTAB
 Convergence criterion =  1.000000000000000E-08
 Upper limit of iteration counts =  5000
 Load data file :input_sample.pbf

 Number of total threads, Thread number  1 0
 ext_ndim =  600 , ndim =  600
Original relative residual norm =  1.0000000E+00
Step    1 relative residual =  5.3560399E-03

                （中略）

Step   11 relative residual =  1.3362906E-08
Step   12 relative residual =  2.5294315E-09

Relative residual norm =   2.5294314E-09

 before user_func time =  1.17245559E-02  [sec]
       user_func time =  0.704002202  [sec]

        bicgstab time =  8.27010907E-03  [sec]
  bicgstab time/steps =  6.89175737E-04  [sec/times]

_____________all time =  0.723996818  [sec]
```



演習課題1(BEM-BB)

1. 解析結果可視化用ファイル「 sample.output.vtk 」
   を自分のPCにダウンロードし、
   ParaViewを用いて下図のような可視化を行ってください。



演習課題2(BEM-BB)

2-1 サンプル以外のデータを用いた計算

演習1で使用したデータはinput_sample.pbf(要素数：600）でした。

Input_10ts.pbf(要素数：約1000)および

Input_216h.pbf(要素数：約2,1000)のデータを用いた計算を行い、計算時間を測定してください。

・ヒント：ジョブスクリプトに書かれているデータ名を変更する

・ヒント：データ216hの計算時間は約15分かかります。ジョブが流れたことを確認したら次の実習に進んでください

2-2 計算オーダーの計測

演習2-1で測定した計測時間を使って、

密行列を用いた境界要素解析の計算時間が、

要素数をNとして、O(N^2)であることを確かめて下さい。

演習課題3、4(BEM-BB)

3-1 コードの並列実行
　いろいろなOpenMPスレッド数やMPIプロセス数で
　計算を実行し、計算時間を計測してください。

3-2 台数効果による性能向上の評価
　逐次実行に比べ、どれだけ高速化されたか、
　性能評価をしてください。

4　計算環境の移動
　ppohBEM_0.4.1.tar.gzをReedBushへコピーし、
　BEM-BBを動作させてください。
- ヒント：Makeファイルの先頭数行で計算環境を設定しています
- ヒント：ジョブスクリプトは他の演習で使ったものを参考に自作

チュートリアルの流れ

1. ppOpen-APPL/BEM
2. 境界要素法とBEM-BBフレームワーク
3. プログラム実習 Ⅰ
   - BEM-BB
4. 階層型行列法と$\mathcal{H}$ACApKライブラリ
5. プログラム実習 Ⅱ
   - BEM-BB+ $\mathcal{H}$ACApK

階層型行列法ライブラリ $\mathcal{H}$ACApK

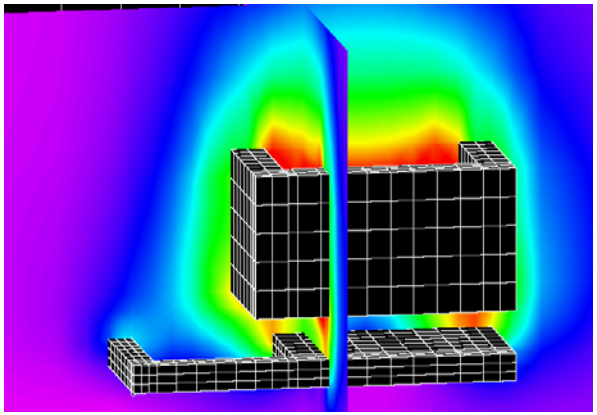
■積分方程式を用いたシミュレーションのための高速計算ライブラリ

► 行列近似手法
階層型行列法 ：$\boldsymbol{O(N^2) \Rightarrow O(N\log N)}$
(Hierarchical matrices, $\mathcal{H}$-matrices)

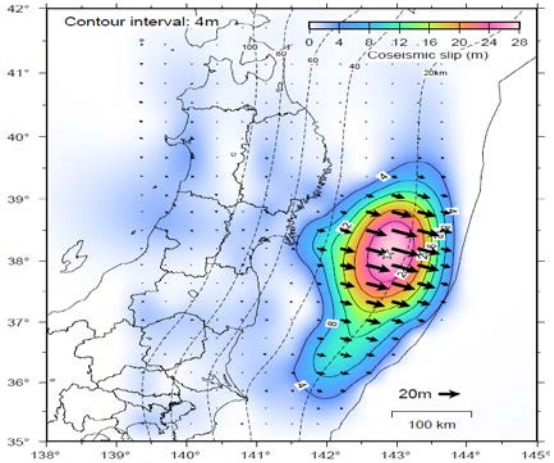
► 並列計算
MPI+OpenMPハイブリッド・プログラミングモデルを採用

■Target applications

・Electromagnetic fields

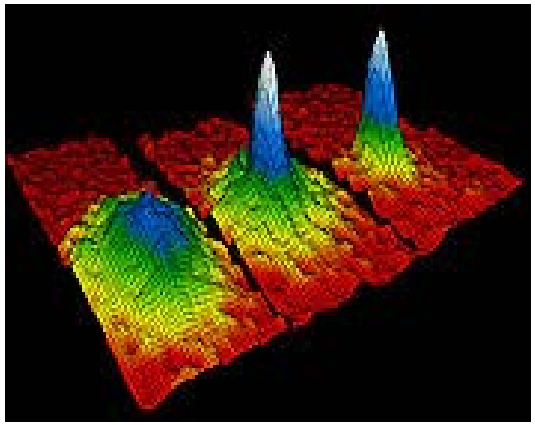


・Earthquake cycle



・Transcribed from **Ohtani at al (2011)**

・Quantum mechanics



・Transcribed from **NIST Image Gallery**

階層型行列法

■積分方程式法に現れる密行列に対する近似手法の一種

適用対象例

$$g[u](x) = \int_{\Omega} g(x,y)\,u(y)\mathrm{d}y$$

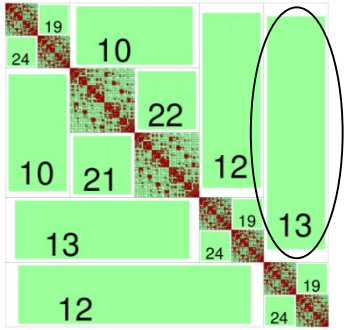
singular kernel: $g(x,y) \in \mathrm{span}(\{|x-y|^{-p}, p>0\})$

離散化

階層型行列法

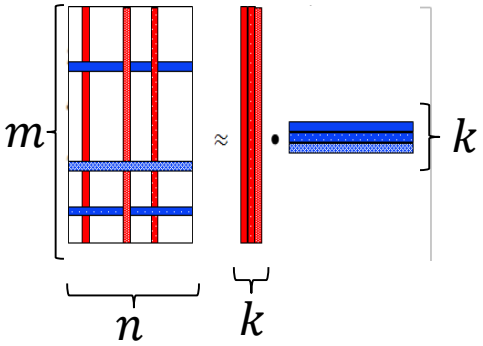
Full rank 密行列

Permutation
Partition



Full-Rank
Low-Rank

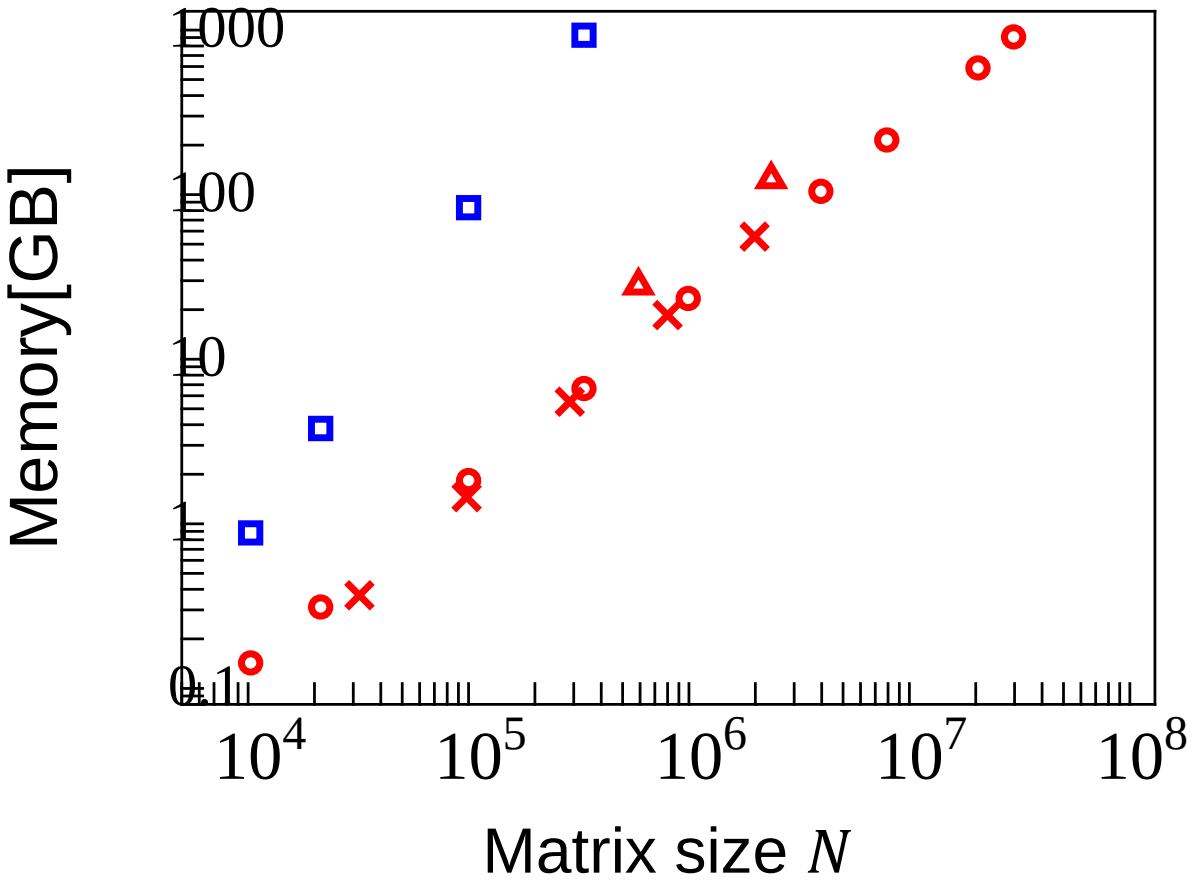
LRA



$O(\boldsymbol{mn}) \Rightarrow O(\boldsymbol{k(m+n)})$

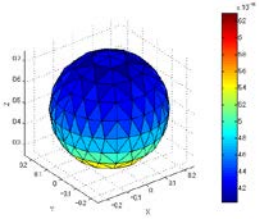
・行列要素数
・行列ベクトル積演算数

メモリ使用量（密行列、階層型行列）

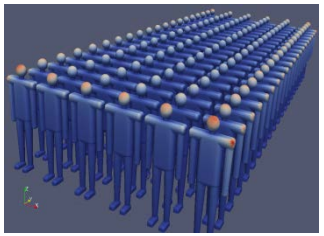


Memory usage (Log-Log scale)

- Dense matrices
- $\mathcal{H}$ACApK (Static electric field)



- $\mathcal{H}$ACApK (Static electric field)



- $\mathcal{H}$ACApK (Earthquake cycle)

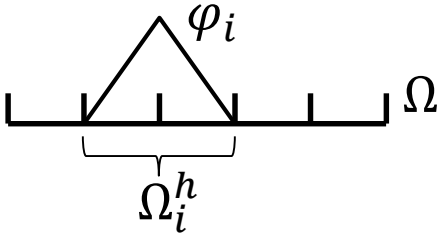
階層型行列法の近似原理

■ツリー法，FMM，パネルクラスタリング法などと近似原理は同じ．

線形積分作用素：$g[u](x) = \int_\Omega g(x,y)\,u(y)\mathrm{d}y$,　例：$g(x,y) = |x-y|^{-1}$

Ritz-Galerkin法 ⇩

$$u \cong u^h = \sum_{i\in\mathcal{I}} \phi_i \varphi_i$$



$$g[u](x) \cong A\phi, \quad A \in \mathbb{R}^{\mathcal{I}\times\mathcal{I}}, \phi \in \mathbb{R}^{\mathcal{I}}$$

遠く離れた2点x, yに対しては，

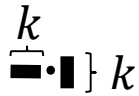
$$g(x,y) \cong \sum_{\nu=1}^{k} g_1^\nu(x) g_2^\nu(y)$$

変数分離した関数積の級数で近似可能

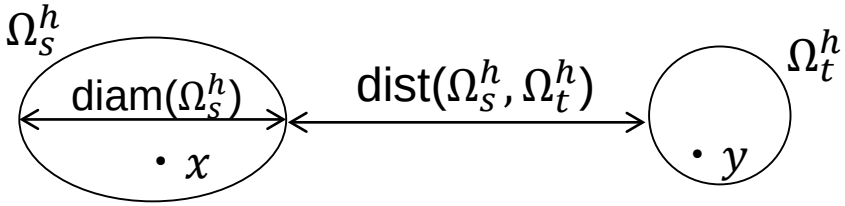
$$A_{ij} = \int_\Omega \varphi_i(x) \int_\Omega g(x,y)\,\varphi_j(y)\mathrm{d}y\mathrm{d}x$$

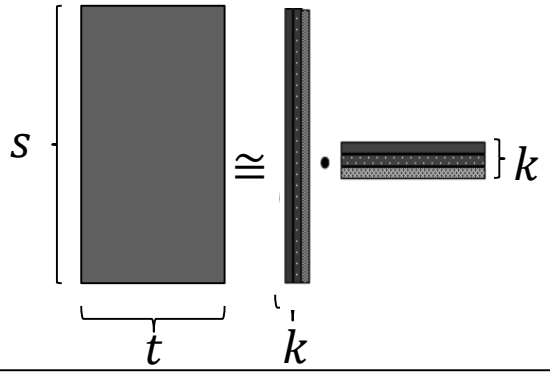
$$\cong \sum_{\nu=1}^{k} \left(\int_\Omega g_1^\nu(x)\varphi_i(x)\,\mathrm{d}x \right) \left(\int_\Omega g_2^\nu(y)\,\varphi_j(y)\mathrm{d}y \right)$$

積分実行



遠く離れた2領域 $\Omega_s^h \ni x$, $\Omega_t^h \ni y$ で考えると，低ランク行列近似 $A|_{s\times t} \cong VW^T$ が可能





近似許容：min{diam(Ω_s^h), diam(Ω_t^h)}≤ ηdist(Ω_s^h, Ω_t^h)

ベクトルの直積と低ランク行列近似

■ベクトルの直積は「ランク1行列」になる

$$a \otimes b = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} \otimes \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix} = \begin{pmatrix} a_1b_1 & a_1b_2 & a_1b_3 \\ a_2b_1 & a_2b_2 & a_2b_3 \\ a_3b_1 & a_3b_2 & a_3b_3 \\ a_4b_1 & a_4b_2 & a_4b_3 \\ a_5b_1 & a_5b_2 & a_5b_3 \\ a_6b_1 & a_6b_2 & a_6b_3 \end{pmatrix}$$

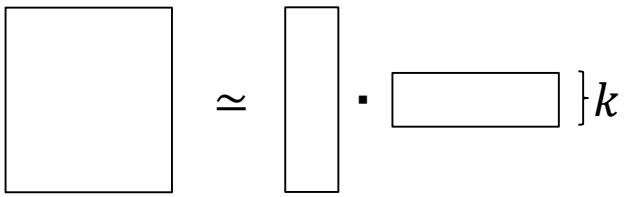
■任意の行列はベクトルの直積（ランク1行列）で近似できる（精度は度外視）

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ c_{41} & c_{42} & c_{43} \\ c_{51} & c_{53} & c_{53} \\ c_{61} & c_{62} & c_{63} \end{pmatrix} \simeq \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} \otimes \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}$$

■基底の異なるランク1行列をk個足すと「ランクk行列」になる

■任意の行列はランクk行列で近似できる

・kが十分小さければ低ランク近似



行列の特異値分解とσ_k-近似

行列 $A \in \mathbb{C}^{m\times n}, m \geq n$ に対して、

■ SVD: $A = U\Sigma V$　　$U \in \mathbb{C}^{m\times n}, \Sigma \in \mathbb{R}^{n\times n}, V \in \mathbb{C}^{n\times n}$

対角行列 Σ の成分 $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$

■ Aのσ_l-近似: $A_l := U\Sigma_l V$,　$\Sigma_l := (\sigma_1, \cdots, \sigma_l, 0, \cdots, 0)$

・ σ_l-近似の誤差: $\|A - A_l\|_F = \sqrt{\sum_{i=l+1}^{n} {\sigma_i}^2}$

・ 許容誤差εに対して $\|A - A_l\|_F < \varepsilon\|A\|_F$ を満たす$l(\varepsilon)$は、

$$l(\varepsilon) := \min\{l \in \mathbb{N}: \sqrt{\sum_{i=l+1}^{n} {\sigma_i}^2} < \varepsilon \sqrt{\sum_{i=1}^{n} {\sigma_i}^2}\}$$

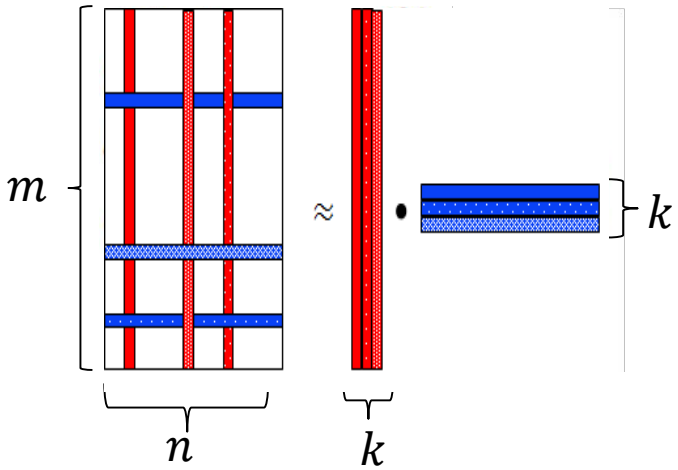
■行列の特異値分解(SVD)は計算コストが高いので、HACApKでは近似手法を使う

ACAを用いた低ランク行列近似

Adaptive Cross Approximation

ACA

$$a \approx \tilde{a}_k = \sum_{\nu=1}^{k} v^{\nu}(w^{\nu})^T$$



- ピボット列とピボット行を交互に選択

  v^1: 任意の列ベクトル (e.g. 1列目)

  w^1: j^1行ベクトル, $j^1 := \mathrm{argmax}_j |v^1{}_j|$

- 推定される近似誤差:

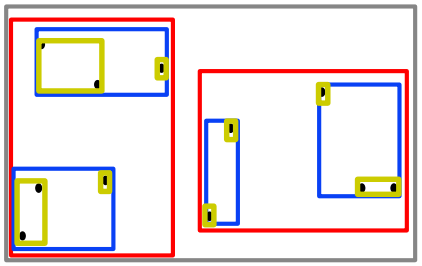
$$\epsilon_k := \frac{||v^k|| \cdot ||w^k||}{\sqrt{\sum_{\nu=1}^{k} ||v^{\nu}|| \cdot ||w^{\nu}||}}$$

- 経験則: $\epsilon_k < \epsilon_{k-1}$

■選択するベクトルの本数により使用メモリ量と近似精度を制御可能.

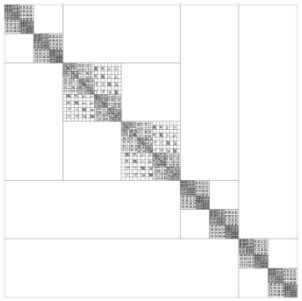
階層型行列の生成プロセス

■$\mathcal{H}$ACApKでは青字の方法を採用している



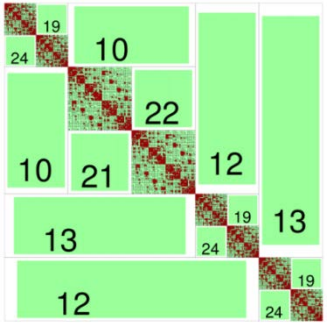
Step 1 Clustering & Construction of a cluster tree

- Method based on geometric information
- Method based on nested dissection
- AMG-like method



Step 2 Construction of a partition

■ There're only blocks(frames of sub-matrices).



Step 3 Fill in sub-matrices

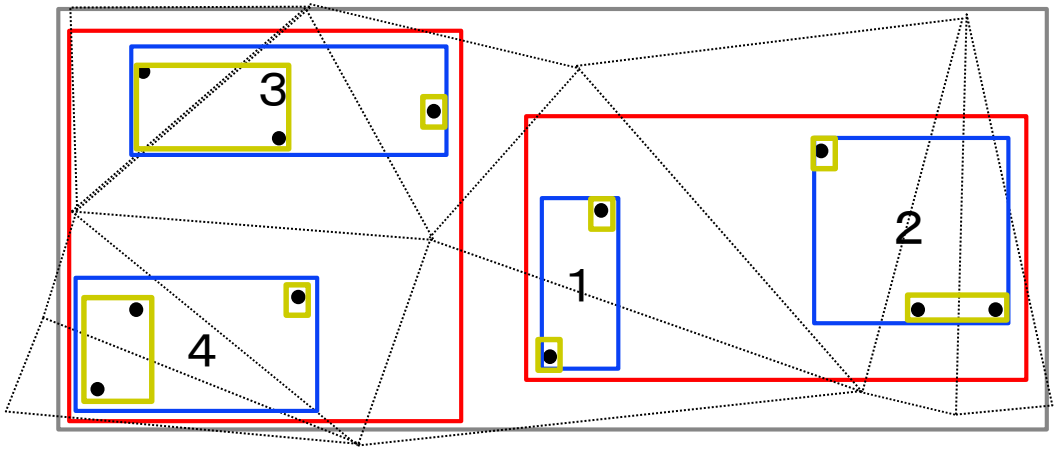
■Low-rank approximation

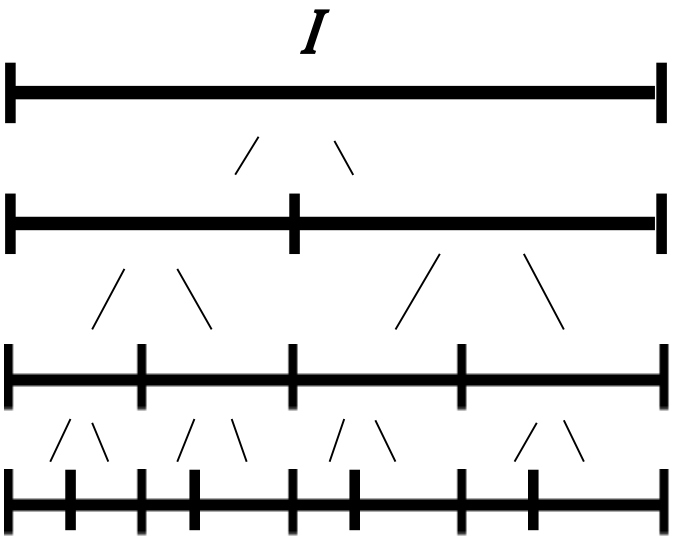
- Cross approximation i) ACA ii) ACA+
- Interpolation
- Hybrid method of interpolation and ACA

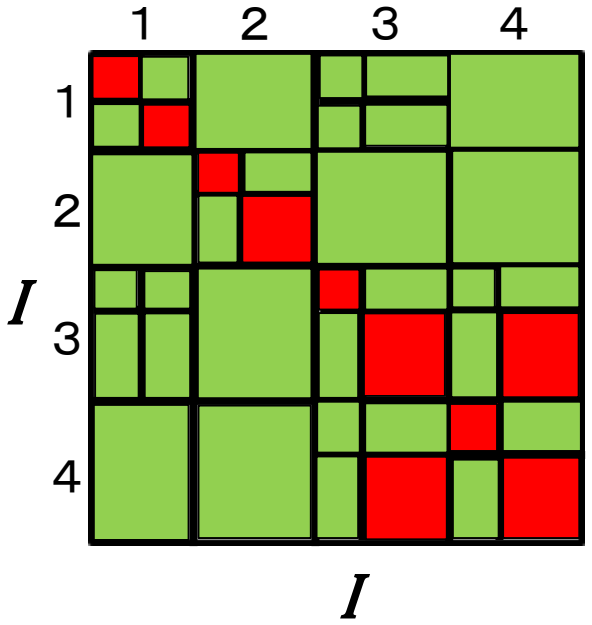
HACApKにおける行列分割の方法

■幾何情報に基づくクラスタリングを利用

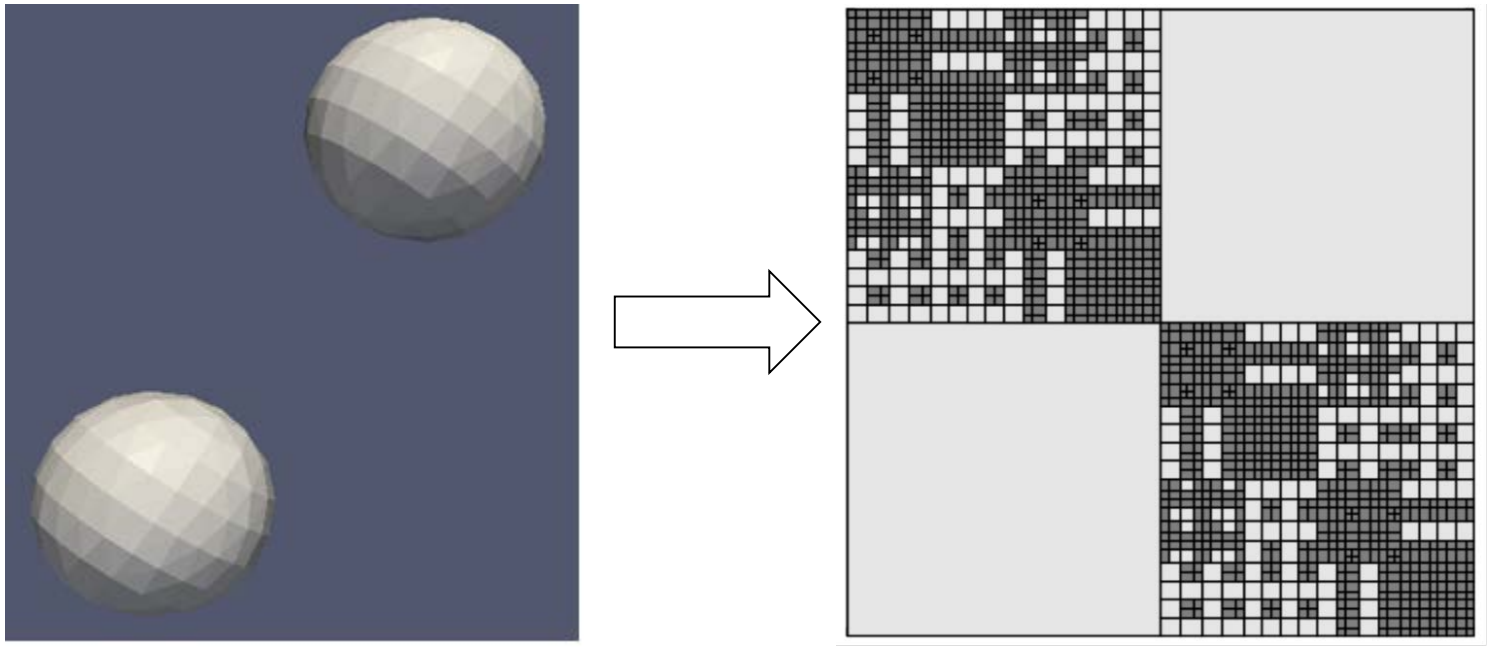
Admissible condition：$\min\{\mathrm{diam}(\Omega_s^h), \mathrm{diam}(\Omega_t^h)\} \leq \eta \mathrm{dist}(\Omega_s^h, \Omega_t^h)$



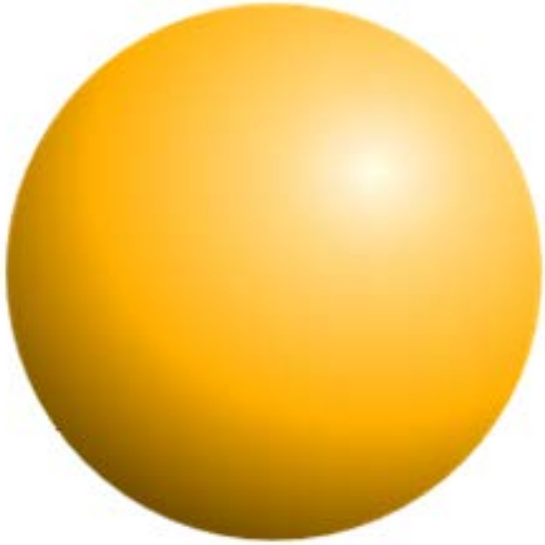


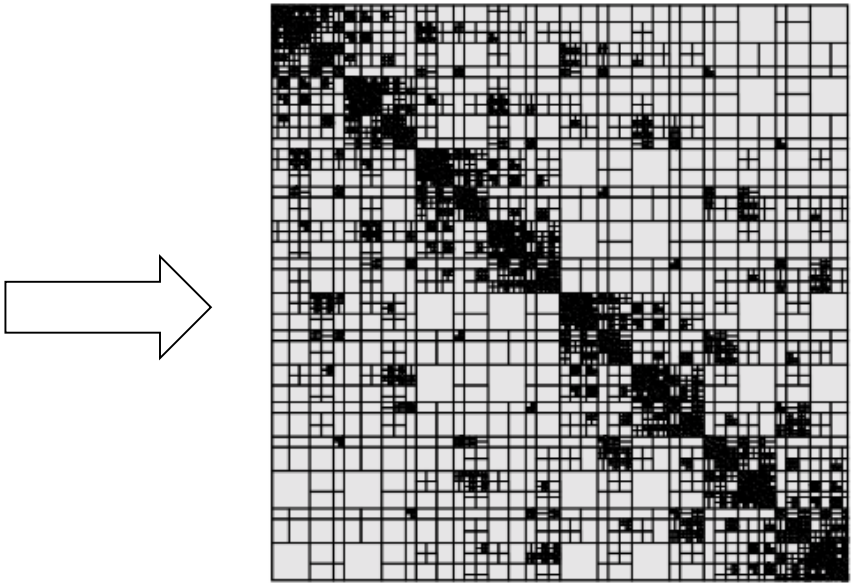


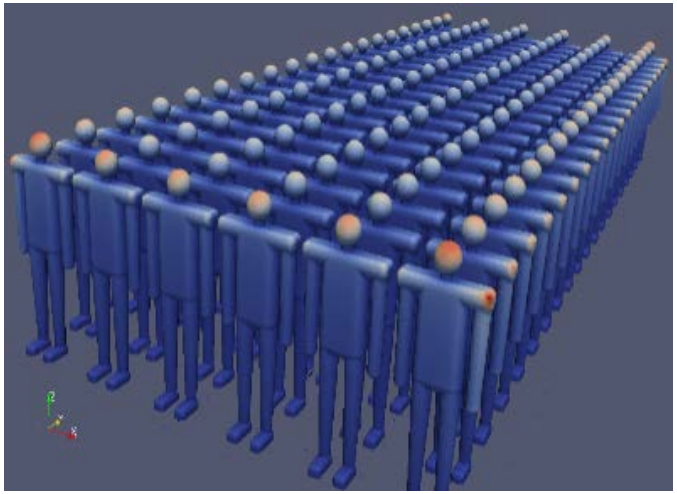
HACApKによる行列分割の例

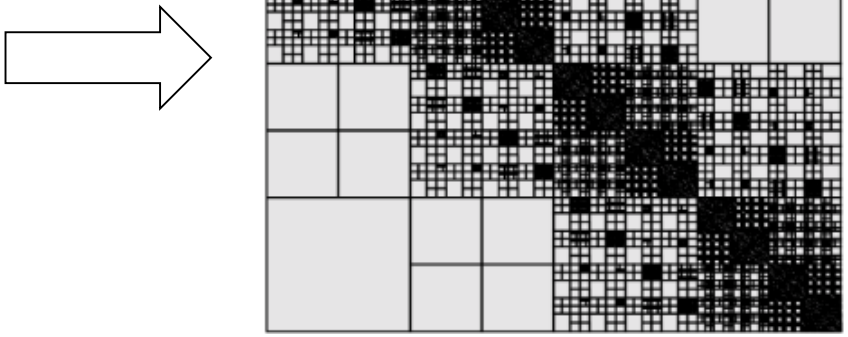


HACApKによる行列分割の例









$\mathcal{H}$ACApKにおける階層型行列法の並列化手法

■階層型行列の生成において

- step 3（最も時間がかかる部分）のみ並列化.
- MPI の通信は不要.

step1 Cluster tree 作成 step2 H-matrix構造 作成	全MPIプロセスで冗長計算
step3 部分行列の計算(ACA)	並列計算

■階層型行列-ベクトル積計算において

- 全MPIプロセスがベクトル全体を持つ.
- MPIプロセス間の通信が必要.

■上記の両方の並列計算において

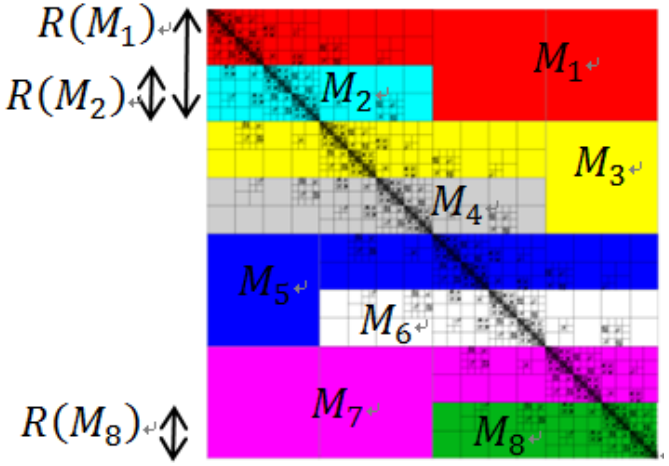
- 計算コアへのデータ割り当て方式は共通にする.
- 演算は部分行列を単位として行う.
- 計算コアへ割り当てられたデータは部分行列の集合となる.

データ割り当て戦略と意図

■MPIプロセスについて

① 可能な限り$R(M_k)$の最大値を最少にする
⇒ 転送データサイズを少なくする

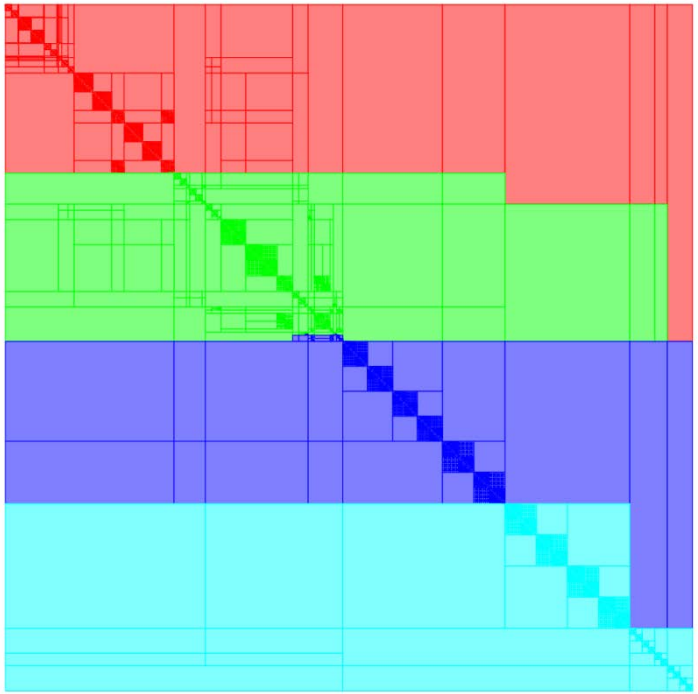
② MPIプロセス間の計算負荷不均衡を最小化する



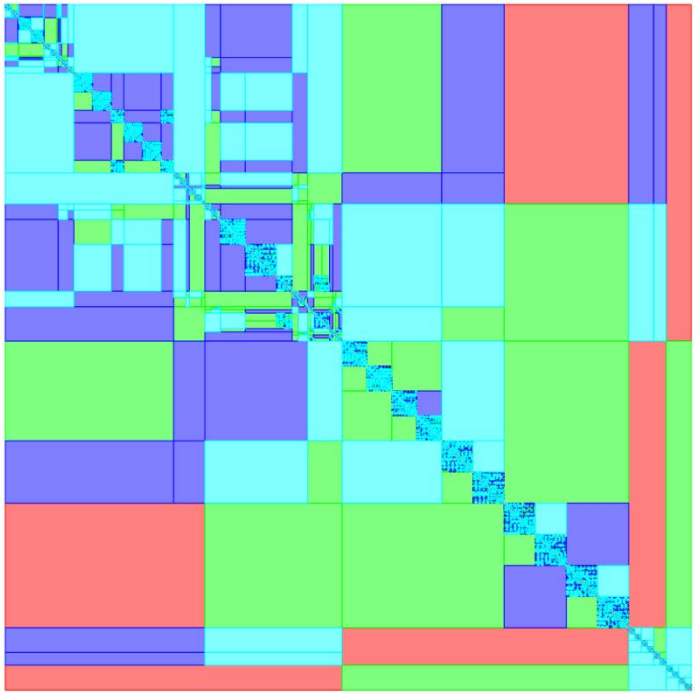
■OpenMPスレッドについて

・OpenMPスレッド間の計算負荷不均衡を最小化する

戦略の違いによる割り当てデータ領域の違い



HACApKで各MPIプロセスへ割り当てられた部分行列



計算負荷が最適になるような部分行列の割り当て

$\mathcal{H}$ACApKの並列計算性能テスト

次の2つの計算の並列計算性能を調べた

- 階層型行列の生成
- HMVM（階層型行列-ベクトル積）

■Computer: 東京大学 Oakleafー fx

Processor : SPARC64™ Ixfx (16cores/node)

Memory : 32GB

Network : 5 GB/s, Tofu.

■要素数: N

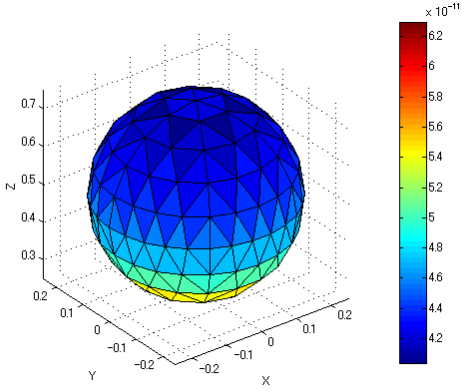
case 1 : $N=$ 1,000

case 2 : $N=$ 10,000

case 3 : $N=$ 100,000

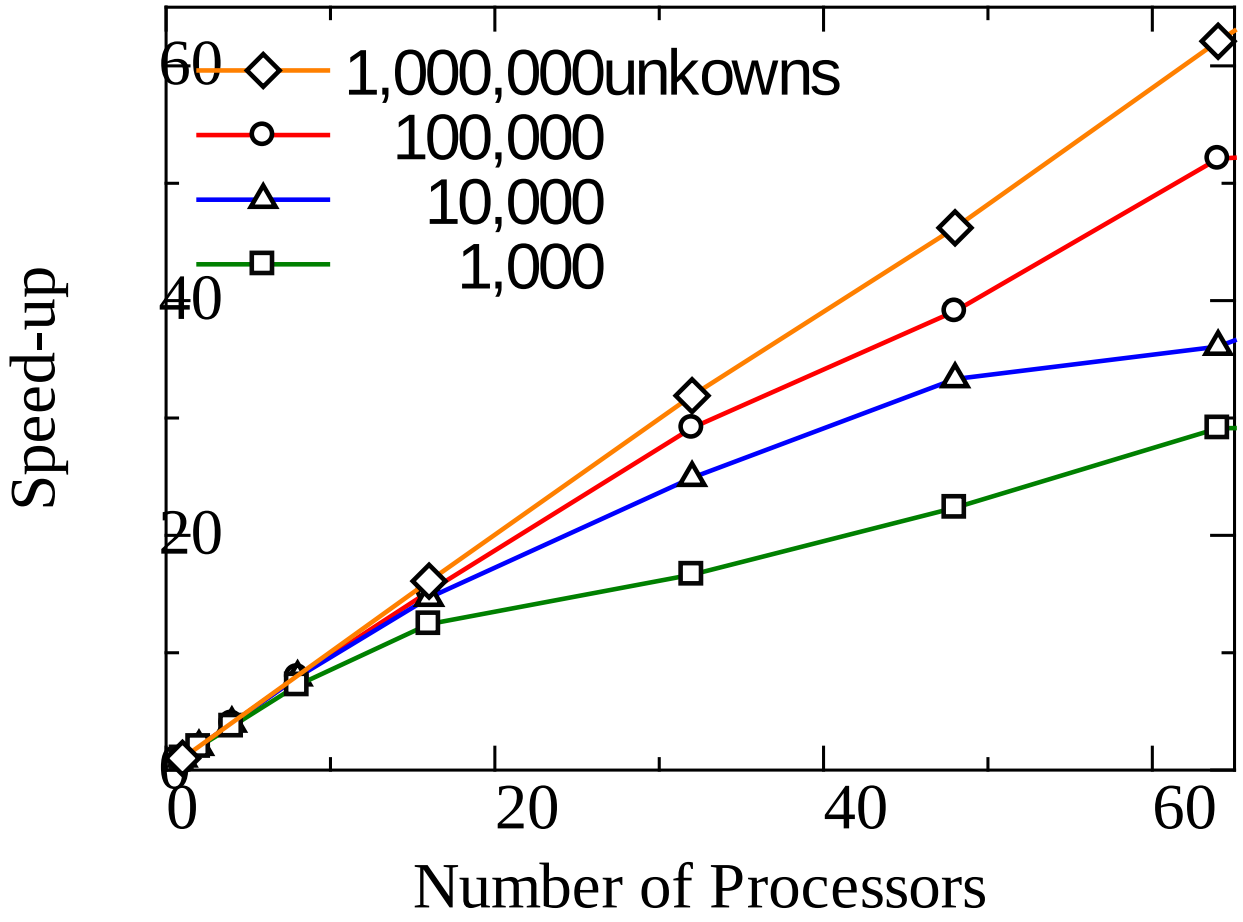
case 4 : $N=$1,000,000

■Test model



階層型行列を生成する計算におけるスケーラビリティ

■データサイズが大きくなるにつれて,
よりよいスケーラビリティが$\mathcal{H}$**ACApK**では得られた.

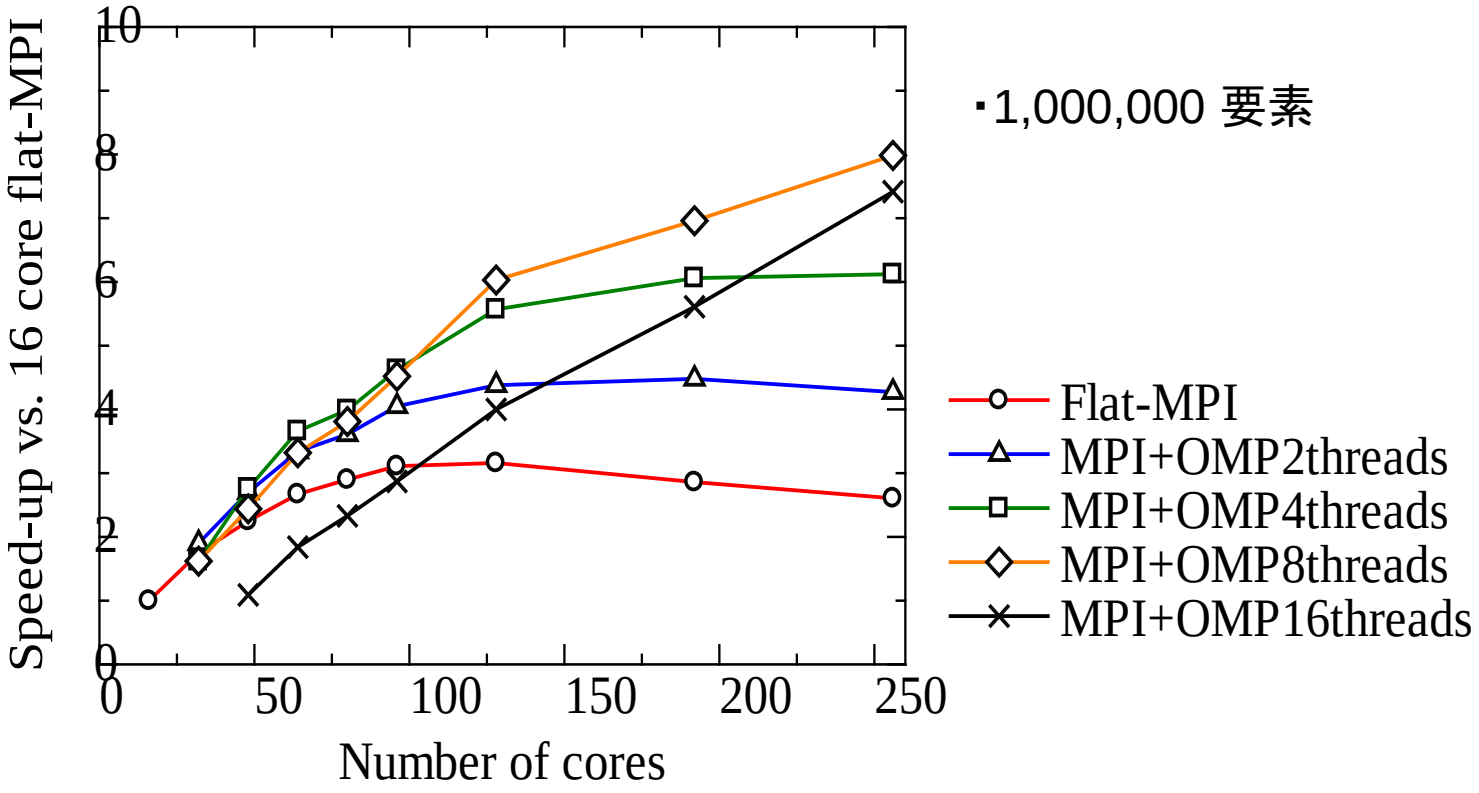


階層型行列ベクトル積計算におけるスケーラビリティ

Flat-MPI版HACApKの1ノードによる計算時間を基準とした.

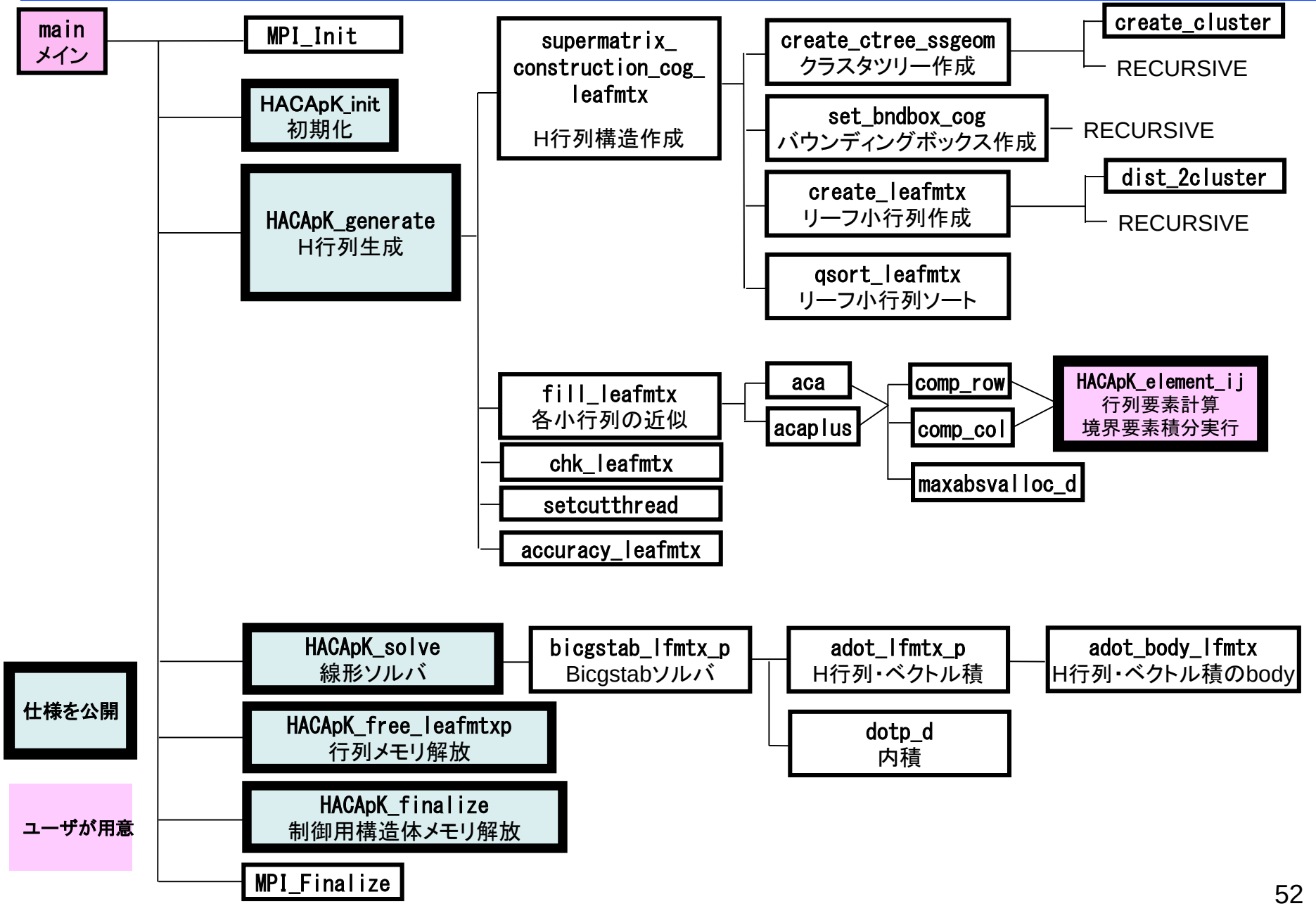
■ MPI+OpenMPハイブリッド版はFlat-MPI版より高い並列計算性能が得られた。MPIの通信の削減効果によるものと考えられる

■Flat-MPIでは、計算速度向上が96コアまでしか得られなかった.



Parallel scalability when performing an H-matrix vector multiplication

HACApK利用コードの構成例



HACApKの使用方法

・決められた順番に関数を呼んでいくだけで使用可能

```
program Example_Using_HACApK
use m_HACApK_solve
use m_HACApK_base
implicit real*8(a-h,o-z)
type(st_HACApK_lcontrol) :: st_ctl
type(st_HACApK_leafmtxp) :: st_leafmtxp
type(st_HACApK_calc_entry) :: st_bemv
real*8,dimension(:,:),allocatable :: coord
real*8,dimension(:),allocatable :: rhs,sol
   :
<<< User code part 1 >>>
   :
lrtrn=HACApK_init(nd,st_ctl,st_bemv)
allocate(coord(nd,3),rhs(nd),sol(nd))
   :
<<< User code part 2 >>>
   :
ztol=1.0e-5
lrtrn=HACApK_generate(st_leafmtxp,st_bemv,st_ctl,coord,ztol)
lrtrn=HACApK_solve(st_leafmtxp,st_bemv,st_ctl,rhs,sol,ztol)
lrtrn=HACApK_free_leafmtxp(st_leafmtxp)
lrtrn=HACApK_finalize(st_ctl)
   :
<<< User code part 3 >>>
   :
end program
```

HACApKイニシャライズ

H行列生成

H行列を係数に持つ線形方程式系を解く

H行列メモリ解放

HACApKファイナライズ

Pseudo code 2.4.1: example of the basic usage of the $\mathcal{H}$ACApK.

ユーザ定義関数 HACApK_element_ij

■2つの要素番号と各要素上の情報を受け取り、行列要素値を返す

```
real*8 function HACApK_entry_ij(i,j,st_bemv)
integer :: i,j
type(st_HACApK_calc_entry) :: st_bemv
    ⋮
 return HACApK_entry_ij
end function
```

■境界要素積分に必要な情報を格納する構造体を宣言しておく

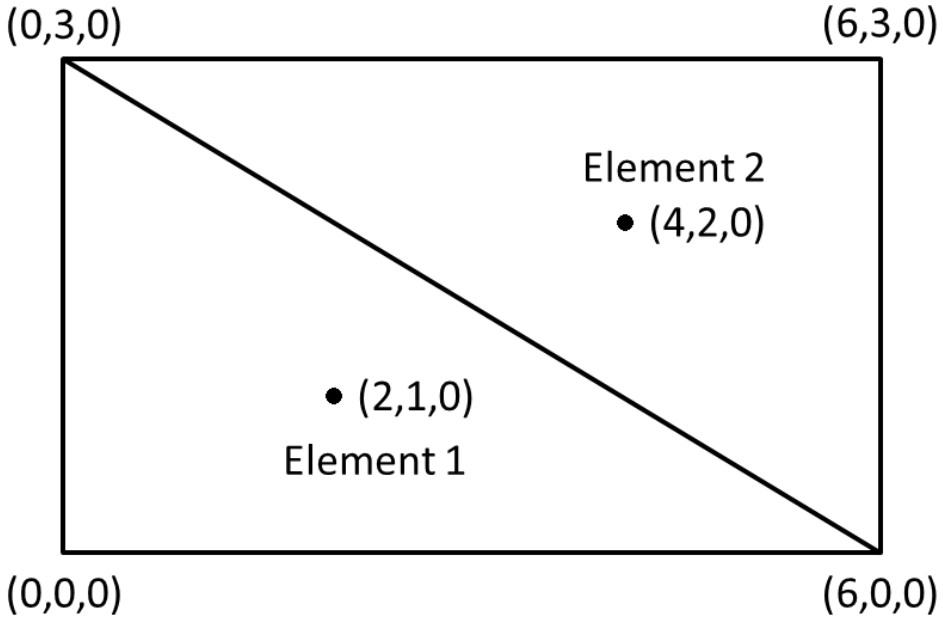
```
type(st_HACApK_calc_entry)
 integer :: nd,lp61
  real*8,pointer :: ao(:)

 integer ::  int_ex
 real*8  ::  dbl_ex
 integer,pointer :: int_ex1(:),int_ex2(:,:)
 real*8,pointer  :: dbl_ex1(:),dbl_ex2(:,:)
end function
```

座標情報coord

■クラスタリングに必要な座標情報

・行列インデックスと関連付けられている三次元デカルト座標を配列として与える

・行列インデックスと関連付けられているのが要素の場合には、重心の座標を与える




```
nd=2
coord(1,1)=2.0; coord(1,2)=1.0; coord(1,3)=0.0
coord(2,1)=4.0; coord(2,2)=2.0; coord(2,3)=0.0
```

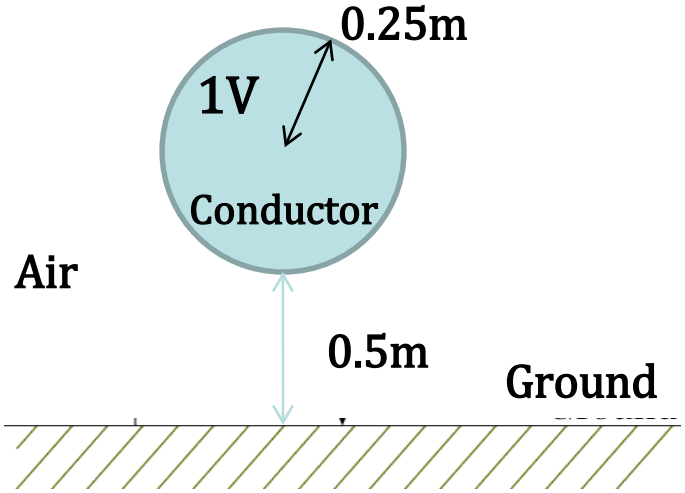
チュートリアルの流れ

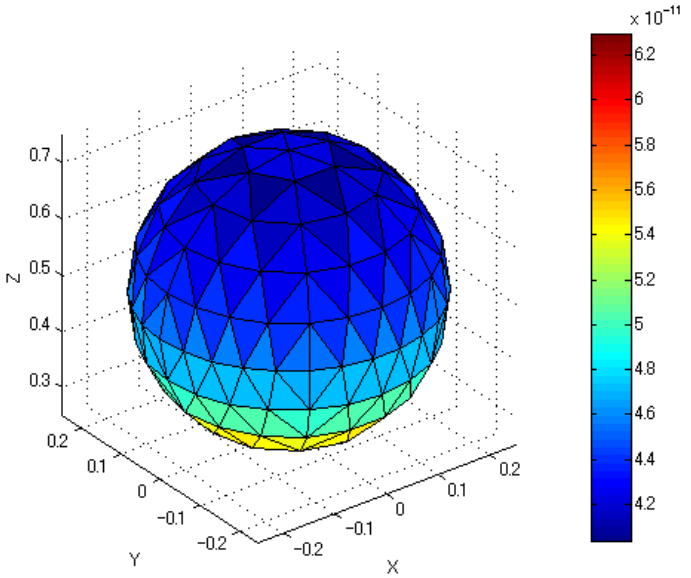
1. ppOpen-APPL/BEM
2. 境界要素法とBEM-BBフレームワーク
3. プログラム実習 Ⅰ
   - BEM-BB
4. 階層型行列法と$\mathcal{H}$ACApKライブラリ
5. プログラム実習 Ⅱ
   - BEM-BB+ $\mathcal{H}$ACApK

サンプルプログラムで計算する問題

■静電場解析

・Potential operator : $\mathcal{V}[u](x) := \int_\Gamma \frac{1}{4\pi\|x-y\|} u(y)dy, \quad x \in \Gamma$





解析条件

・半径0.25mの導体球に1Vを与える
・表面に現れる電荷を求める
（半無限領域内の電位が導出される）

解析結果

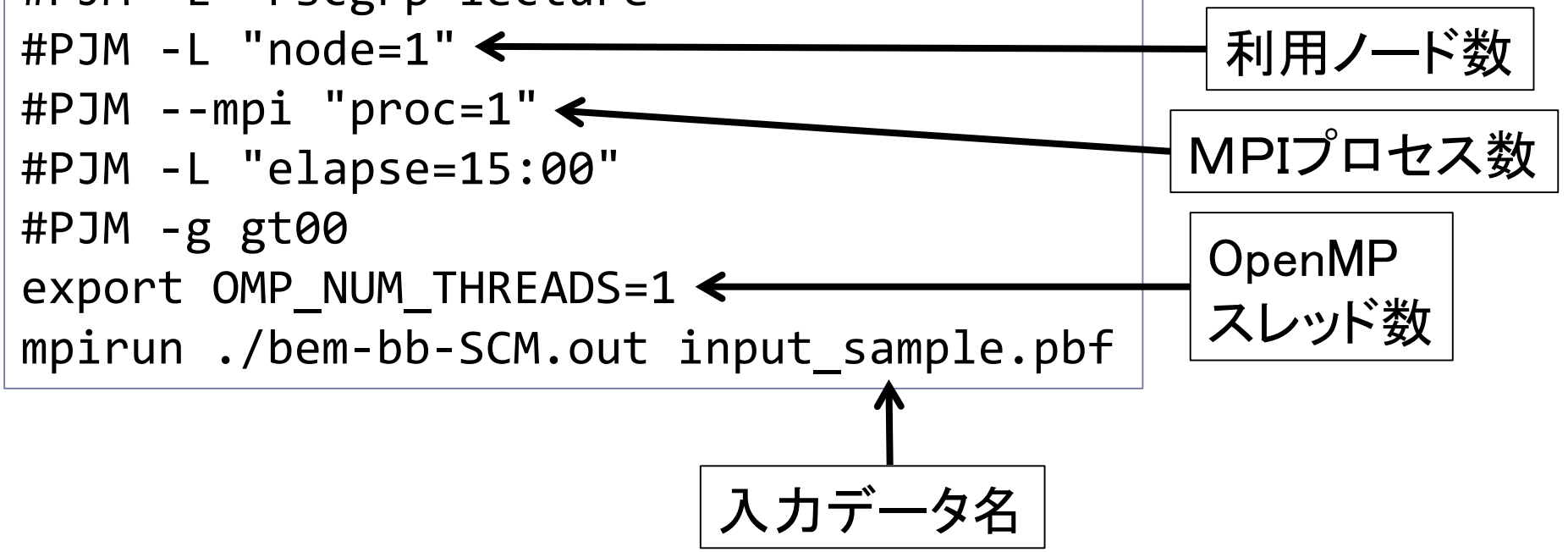
導体球の下半分に大きな電荷密度が生じる

実習を行うあたっての注意点(HACApK)

- ファイル名
  BEM-BBと同時にインストール済みです
- ジョブスクリプトファイルbem-bb.bash中のキュー名を
  lecture から tutorial に変更してから
  pjsubしてください。
  - lecture：実習時間外のキュー（同時実行数1）
  - tutorial：実習時間内のキュー（同時実行数4+）

HACApK用ジョブスクリプト(Oakleaf-fx)(BEM-BB用と同一)

```
#!/bin/bash
#PJM -L "rscgrp=lecture"
#PJM -L "node=1"
#PJM --mpi "proc=1"
#PJM -L "elapse=15:00"
#PJM -g gt00
export OMP_NUM_THREADS=1
mpirun ./bem-bb-SCM.out input_sample.pbf
```



ノード数、MPIプロセス数、OpenMPスレッド数の設定は
Oakleaf-fxは1ノードあたり16コアであることに注意して行う

BEM-BB＋HACApKの実行（Oafleaf-fx）

- Oakleaf-fxにログインする
- ppOpenBEM_***がインストールされていることを確認
- 以下のコマンドを実行する

```
$ cd ppohBEM_0.4.1
$ cd HACApK_1.0.0
$ cd src
$ cd HACApK_with_BEM-BB-framework_1.0.0
$ make
$ pjsub bem-bb.bash
```

- 実行が終了したら、以下を実行する

```
$ cat bem-bb.bash.oXXXXXX （XXXXXXは数値）
```

BEM-BB＋HACApKの実行（Oafleaf-fx）

■以下のような結果が見えれば成功

```
 Number of processes  1
 Number of unknowns set on each face element =  1
 Selected linear solver = BICGSTAB
 Convergence criterion =  1.000000000000000E-06
 Upper limit of iteration counts =  5000
 Number of total threads, Thread number  1 0
 ext_ndim =  21600 , ndim =  21600
 ****************** HACApK start **********************
nd=     21600 nofc=     21600 nffc=         1
nrank=         1 nth=         1
No. of cluster=        3903
 No. of nsmtx 17002 33096 16699
    1:Rk-matrix 2: dense-mat 3:H-matrix
ncpc=    24306128  ncpc/nrank=    24306128
 time_supermatrix            = 0.1259064860641956
 time_fill_hmtx              = 87.13198765483685
 time_construction_Hmatrix    = 87.25789824104868
 Memory of the H-matrix= 327.1665954589844 (Mbyte)
 Memory compression v.s. dense matrix=
9.191182270233195 (%)

                （中略）

before HACApK time =  0.266355455  [sec]
       HACApK time =  98.4935837  [sec]
 _____________all time =  98.7606354  [sec]
```

MPIプロセス数

OpenMPスレッド数

要素数

H行列の使用メモリ

密行列に対する
H行列の圧縮率

計算時間

演習課題1(HACApK)

1-1 サンプル以外のデータを用いた計算

演習1で使用したデータはinput_sample.pbf(要素数：600）でした。

Input_10ts.pbf(要素数：約1000)および
Input_216h.pbf(要素数：約2,1000)のデータを用いた計算を行い、計算時間を測定してください。

・ヒント：ジョブスクリプトに書かれているデータ名を変更する

密行列を用いた場合の計算時間と比較してください。

1-2 計算オーダーの計測

演習1-1で測定した計測時間を使って、
階層型行列を用いた境界要素解析の計算時間が、
要素数をNとしてO($N\log N$)であることを確かめて下さい。

演習課題2、3(HACApK)

3-1 コードの並列実行
いろいろなOpenMPスレッド数やMPIプロセス数で
計算を実行し、計算時間を計測してください。

3-2 台数効果による性能向上の評価
逐次実行に比べ、どれだけ高速化されたか、
性能評価をしてください。

3　計算環境の移動
ppohBEM_0.4.1.tar.gzをReedBushへコピーし、
BEM-BB＋HACApKを動作させてください。

- ヒント：Makeファイルの先頭数行で計算環境を設定しています
- ヒント：ジョブスクリプトは他の演習で使ったものを参考に自作

お疲れさまでした