Post Peta CRESTで開発した $\mathcal{H}$行列ライブラリ$\mathcal{H}$**ACApK**（ライブラリと使用法について）

伊田　明弘（東京大学）

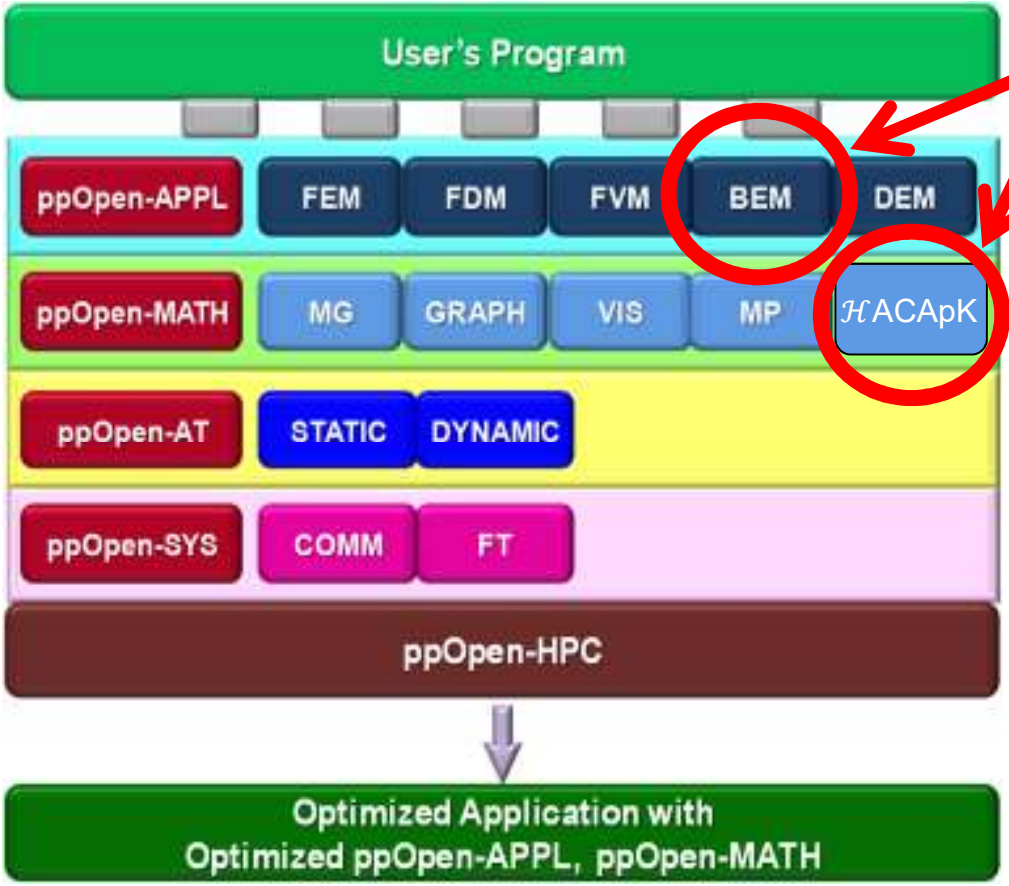
PCCCワークショップin大阪2020
2020年 2月 21日
富士通株式会社　関西システムラボラトリ（大阪）

目次

概要：ppOpen-HPC

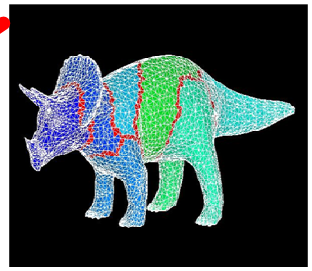
■JST-CREST
- “ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出”領域
- “自動チューニング機構を有するアプリケーション開発・実行環境**ppOpen-HPC**”
  研究代表者: 中島 研吾(東京大学・情報基盤センター)

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

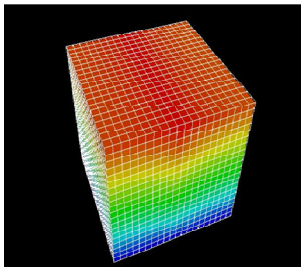


この話の主題

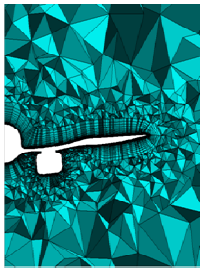
ppOpen-HPC covers …



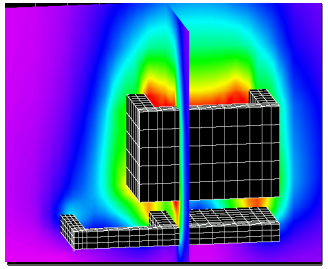
FEM
Finite Element Method



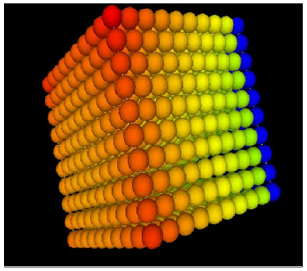
FDM
Finite Difference Method



FVM
Finite Volume Method

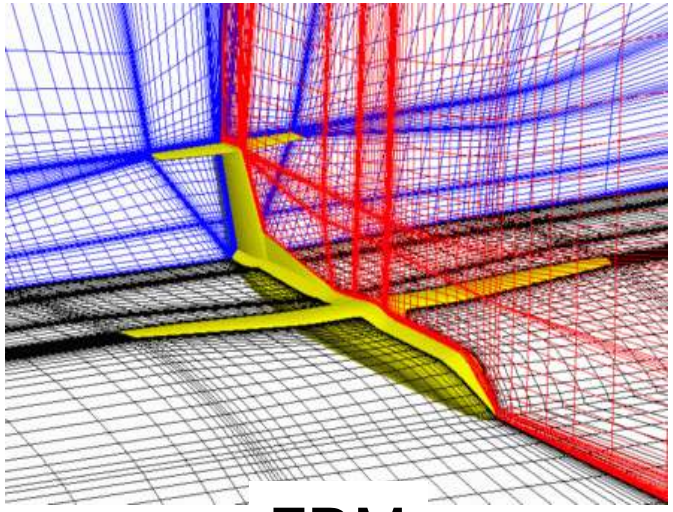


BEM
Boundary Element Method

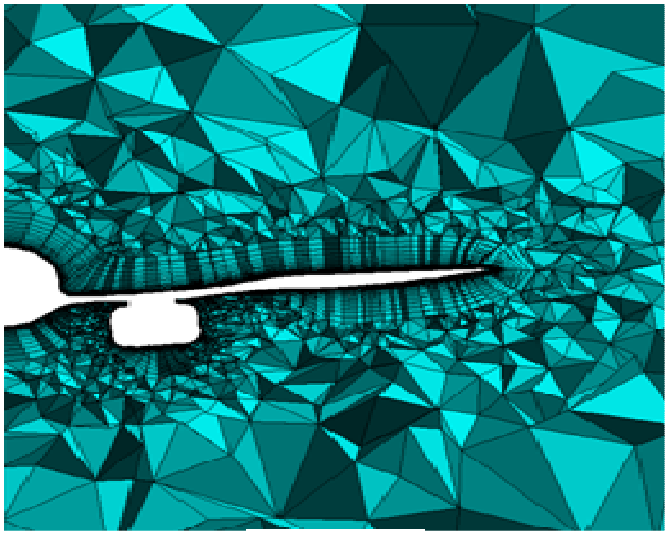


DEM
Discrete Element Method

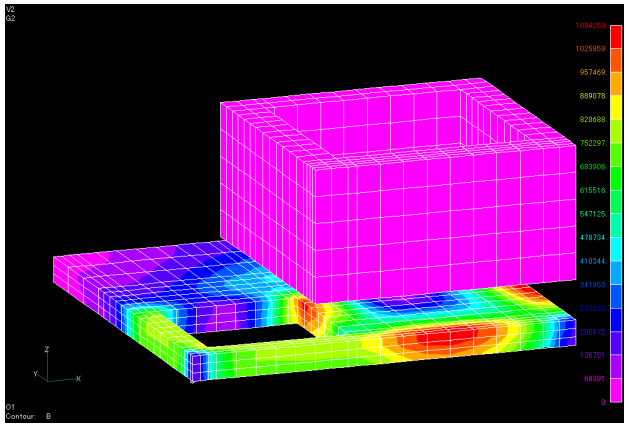
FDM, FEM, BEMで用いられる計算格子

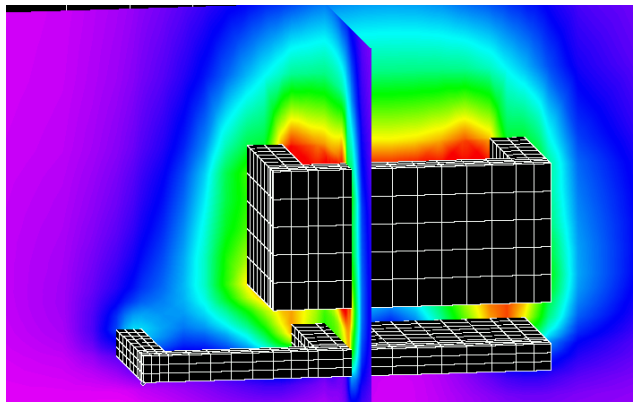


FDM



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 による離散化

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

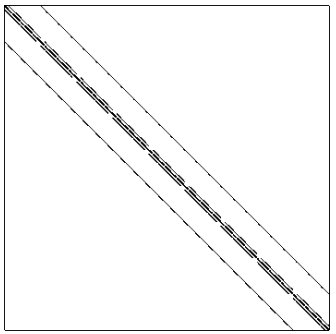
$$\int_\Omega \psi(x)\Delta\phi(x)\mathrm{d}x = \int_\Omega \psi(x)\rho\,\mathrm{d}x$$

・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

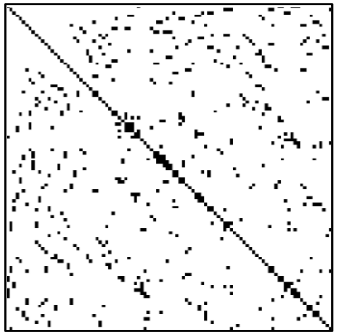
・Power series



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

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

■

$\mathcal{H}$ACApK ライブラリ

■境界要素法（BEM）シミュレーション向け高速計算ライブラリ

■境界要素法のボトルネック

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

・Degenerate kernel : $\kappa(x,y) \cong \sum_{\nu=1}^{r} \kappa_1^\nu(x)\kappa_2^\nu(y)$,for far remote $\boldsymbol{x}, \boldsymbol{y}$

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

・旧来手法では密行列を扱いが必要 ⇨ 計算量$O(\boldsymbol{N^2})$

■大規模計算に向けて

► 密行列に対し近似手法の導入

・低ランク構造行列法：$\boldsymbol{O(N^2) \Rightarrow O(N\log N)}$

$\mathcal{H}$行列，格子$\mathcal{H}$行列, BLR行列

► 分散メモリ並列計算

・Hybrid MPI+X programming model

$\mathcal{H}$ACApK ライブラリ

■対応プラットフォーム
・CPU clusters　・Multi- GPUs（整備中）

■ プログラミング言語
・Fortran90+OpenMP
・CUDA, CUDA Fortran, OpenACC (GPU対応)

■必要な外部ライブラリ
・MPI

■ 公開形態
・Open source　・MIT license

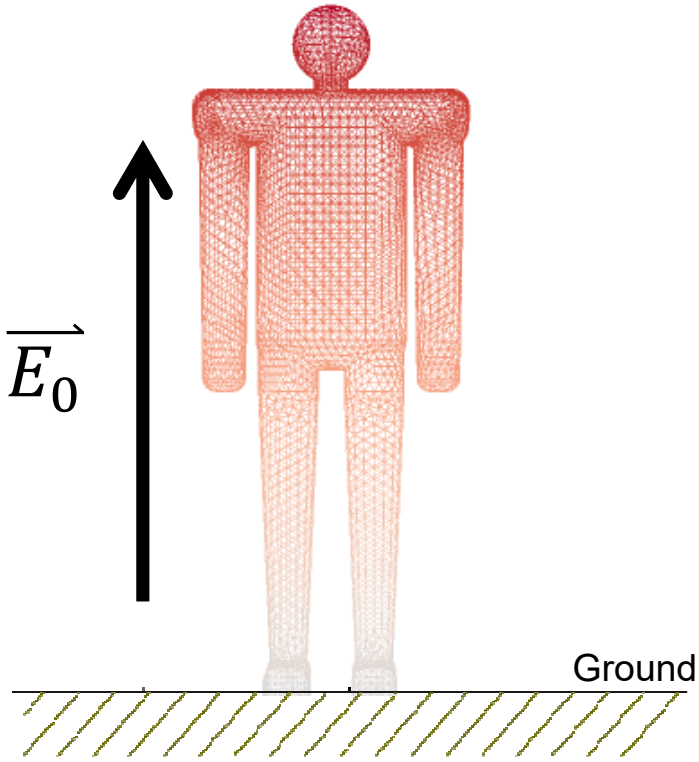
■ ダウンロード サイト
・https://github.com/Post-Peta-Crest/ppOpenHPC/tree/MATH/HACApK
・http://ppopenhpc.cc.u-tokyo.ac.jp（旧Version）

$\mathcal{H}$ACApKの適用例

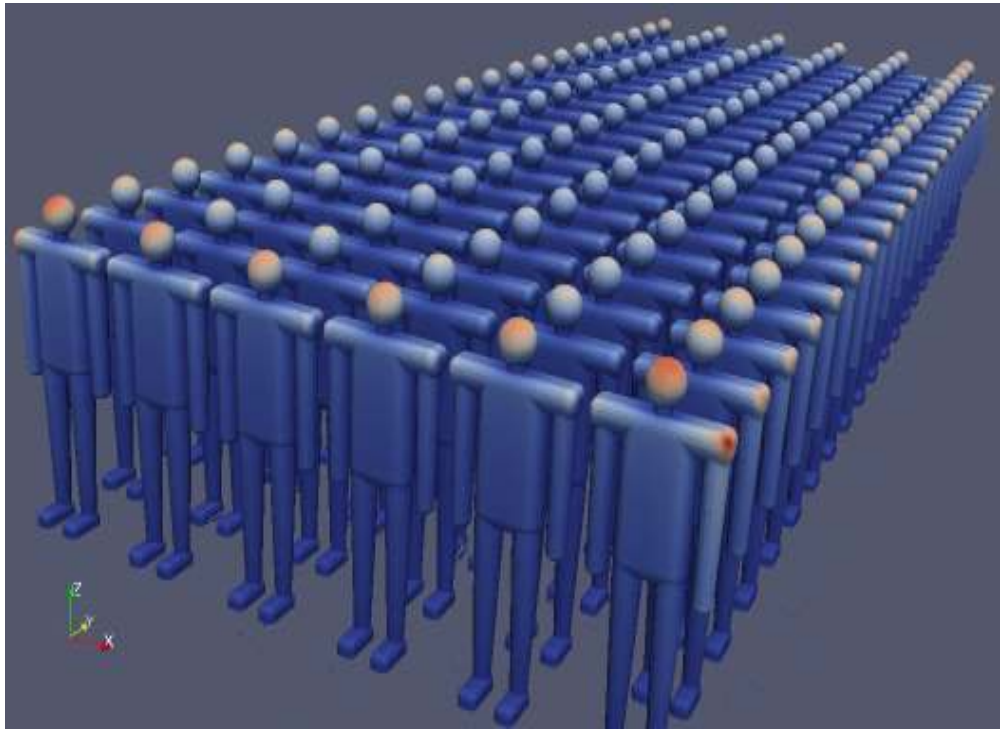
■Static electric field analysis

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

・Induced surface charge is calculated in half-infinite domain.



Analysis condition



Analysis result

$\mathcal{H}$ACApKの適用例（CDWM）

■Calculate LLG equation with terms of spin torque effect

$$\frac{\partial \boldsymbol{M}}{\partial t} = -\gamma \boldsymbol{M} \times \boldsymbol{H} - \frac{\alpha}{M_S} \boldsymbol{M} \times \frac{\partial \boldsymbol{M}}{\partial t} \underbrace{- (\mathbf{u} \cdot \boldsymbol{\nabla}) \boldsymbol{M} - \frac{\beta}{M_S} \boldsymbol{M} \times [(\mathbf{u} \cdot \boldsymbol{\nabla}) \boldsymbol{M}]}_{\text{Spin torque term}}$$

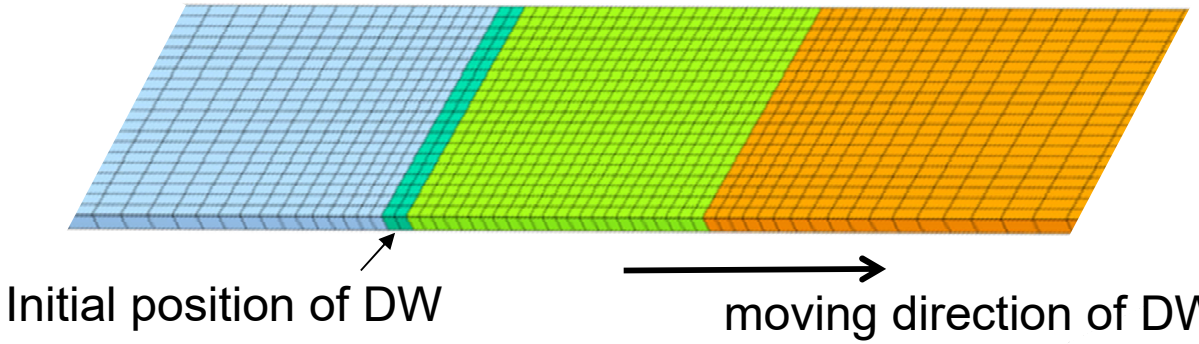
・**Effective field**

$$\boldsymbol{H} = \underbrace{\frac{1}{4\pi\mu_0} \int_V \boldsymbol{\nabla} \cdot \boldsymbol{M} \frac{\boldsymbol{r}}{r^3} \, dV}_{\text{Demagnetizing field } \boldsymbol{H}^D} + \underbrace{\frac{2K_u}{M_s^2} (\boldsymbol{M} \cdot \boldsymbol{k}) \boldsymbol{k} + \frac{2A}{M_s^2} \Delta \boldsymbol{M} + \boldsymbol{H}_{app}}_{\text{Other field components } \boldsymbol{H}^O}$$

■Discretization method

・**Time** : Explicit Runge–Kutta $\quad \boldsymbol{M}|_t, \boldsymbol{H}|_t \Longrightarrow \boldsymbol{M}|_{t+\Delta t}$

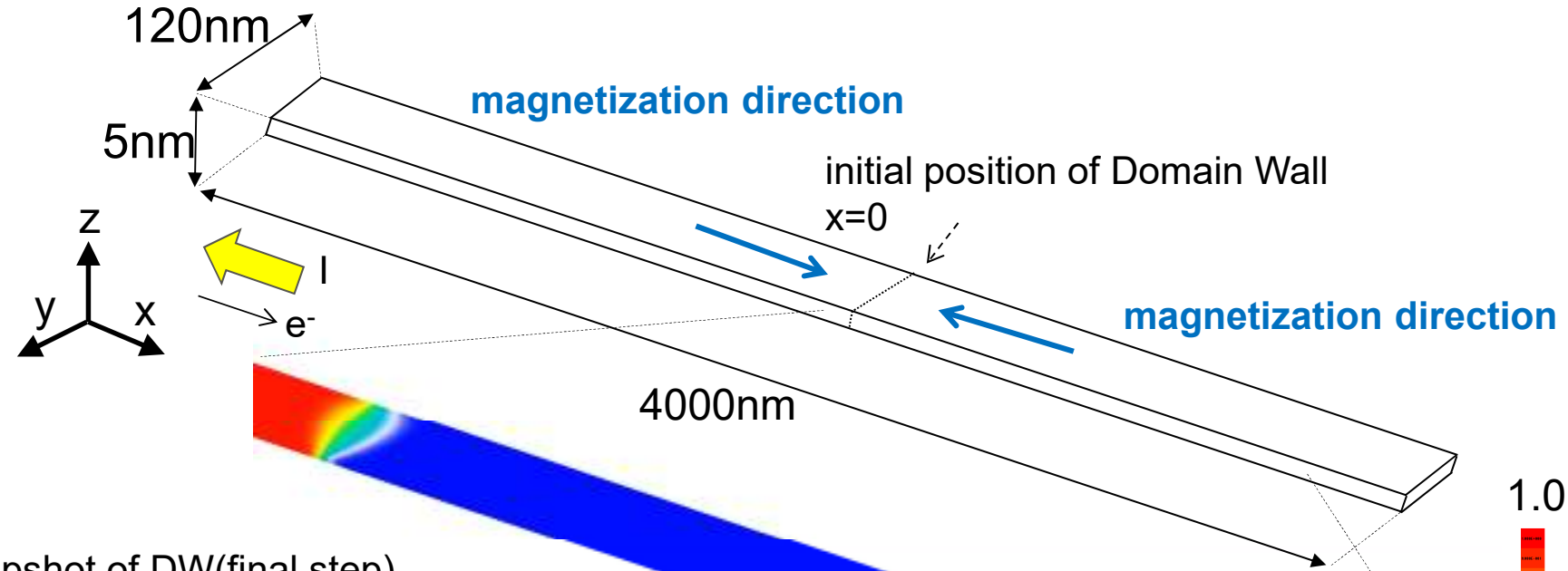
・**Space** : Magnetic material is divided into N cubic elements



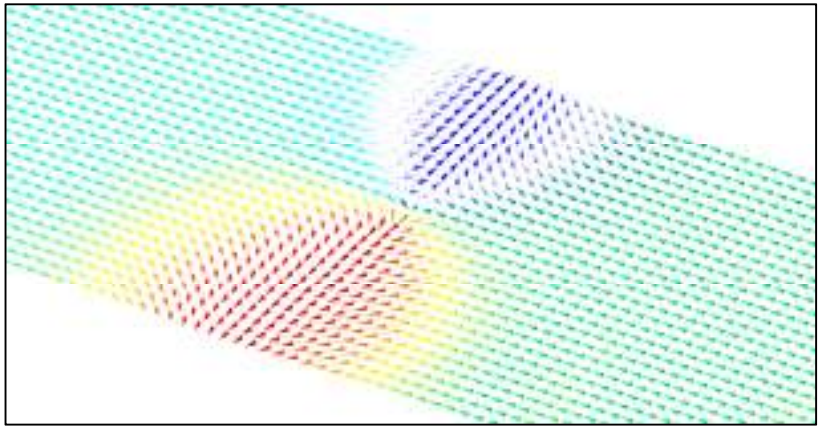
$\mathcal{H}$ACApKの適用例（CDWM）

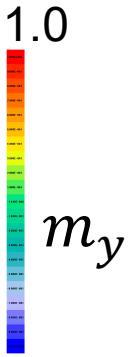
■Current-induced Domain Wall Motion (CDWM)

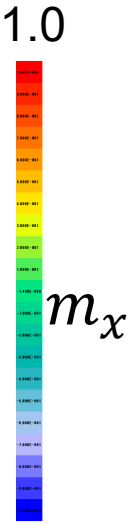
■CDWM is induced by a spin-polarized current in magnetic nanowires.



Snapshot of DW(final step)







[Ida : *IEEE Trans. on Mag. 2020*]

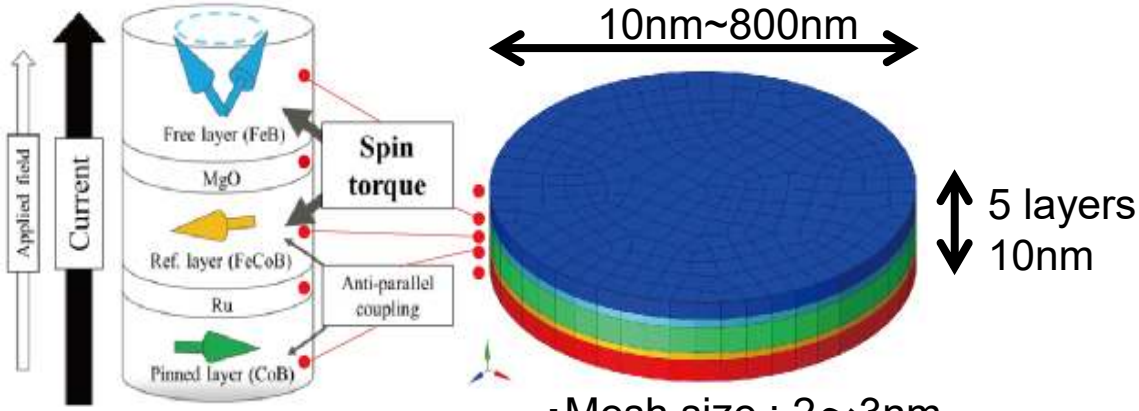
$\mathcal{H}$ACApKの適用例

[Ida : *IEEE Trans. on Mag. 2017*]

■Micromagnetic simulation of spin torque oscillator

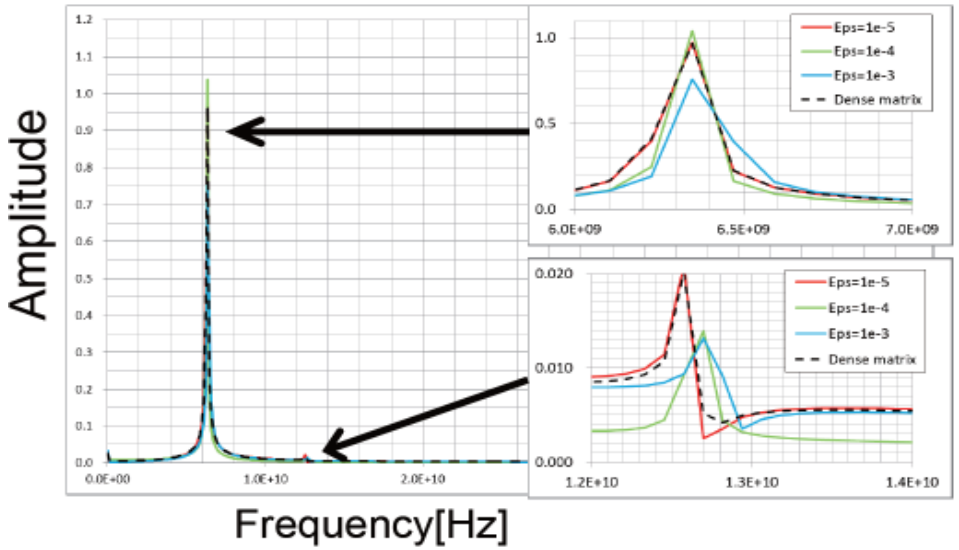
・Potential operator of demagnetic field : $\frac{1}{4\pi}\int_{\partial V}\varphi_1(\boldsymbol{y})\nabla|\boldsymbol{x}-\boldsymbol{y}|^{-1}\cdot \mathrm{d}\boldsymbol{S}$

・2-nd order singularities
・Vector quantities



・Mesh size : 2～3nm

・Oscillating frequency spectrum is calculated in magnetic materials.



・H-matrices accuracy of $\varepsilon = 10^{-5}$ is required to reproduce oscillation processes.

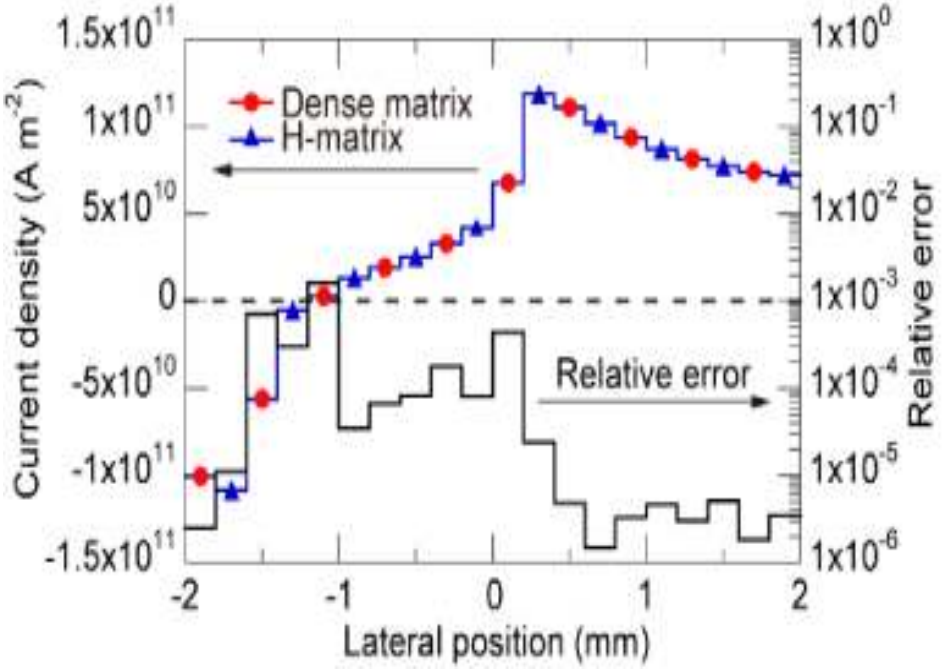
$\mathcal{H}$ACApKの適用例

[Tominaga : *IEEE Trans. on Applied Superconductivity 2018*]

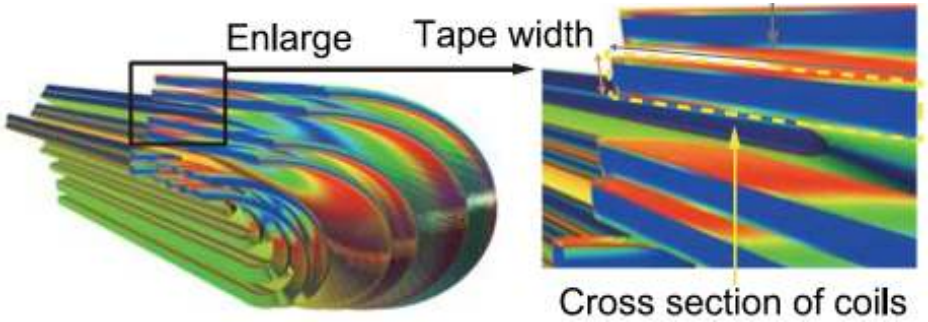
■Superconductor Simulation

・Potential operator : $\dfrac{\mu_0 t_s}{4\pi}\int_{S'}\dfrac{(\nabla\times \boldsymbol{n}'T')\times \boldsymbol{r}\cdot\boldsymbol{n}}{r^3}\mathrm{d}S'$

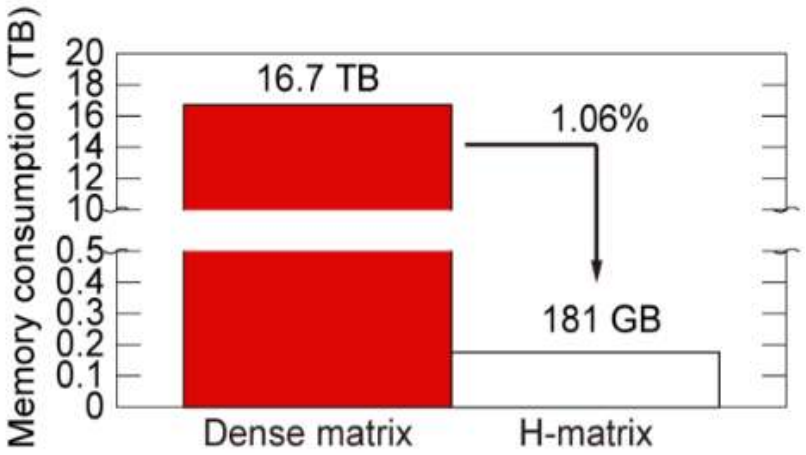
・Current density is calculated in coils wound with coated conductors.



Current distribution



Model (N=1,516,060)



Memory usage

$\mathcal{H}$ACApKの適用例

[Mifune : *Superconductor Science and Technology 2019*]

■Superconductor Simulation accelerated by AMG

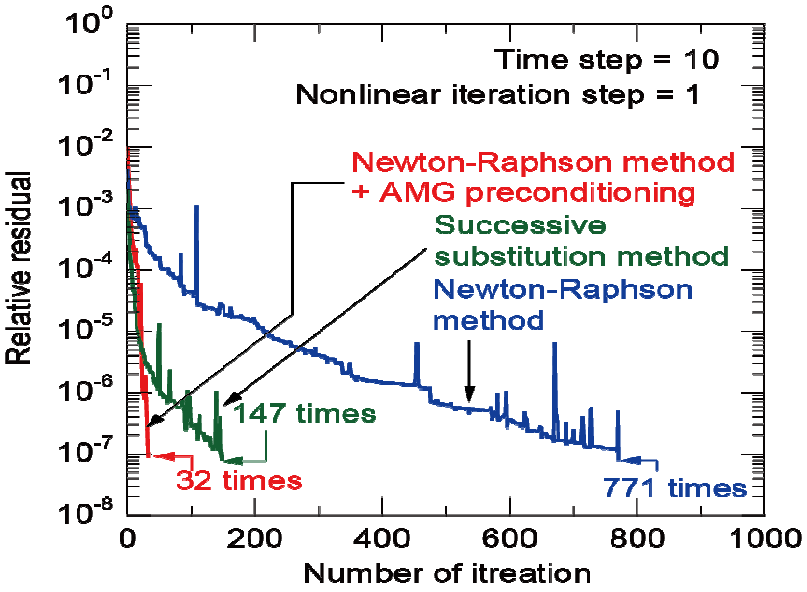
・Governing equation： Faraday's law and Biot–Savart's

$$\nabla \times \left(\frac{1}{\sigma_s[T]} \nabla \times \boldsymbol{n}T \right) \cdot \boldsymbol{n} + \frac{\partial}{\partial t} \left(\frac{\mu_0 t_s}{4\pi} \int_{S'} \frac{(\nabla \times \boldsymbol{n}'T') \times \boldsymbol{r} \cdot \boldsymbol{n}}{r^3} \mathrm{d}S' \right) = 0$$

⇩ Galerkin and backward finite difference methods

$$A[\boldsymbol{T}_{[m-1]}] \cdot \boldsymbol{T}_{[m]} + B \cdot (\boldsymbol{T}_{[m]} - \boldsymbol{T}_{[m-1]}) = 0.$$

Sparse　　Dense



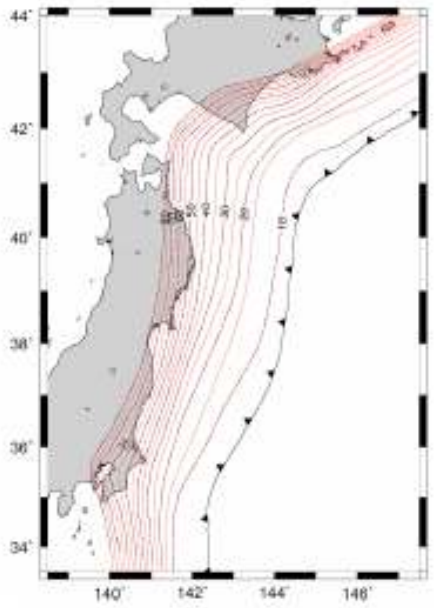
・**Previous study**
- Apply $\mathcal{H}$ACApK to integral
- Solve by Newton-Raphson using BiCG-STAB

・**New approach**
- Set A as preconditioning matrix in the BiCG-STAB
- Apply AMG to the preconditioner

$\mathcal{H}$ACApKの適用例

■Earthquake Cycle Simulation

[Ohtani : Proc. Comp. Sci. *2014*]



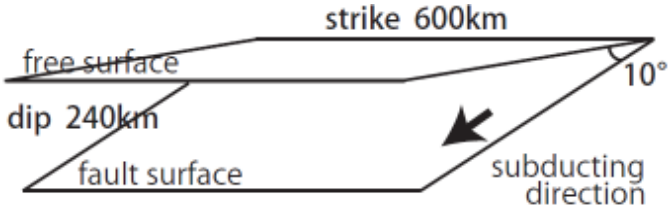
Integral operator with $g(x,y) \propto |x-y|^{-3}$

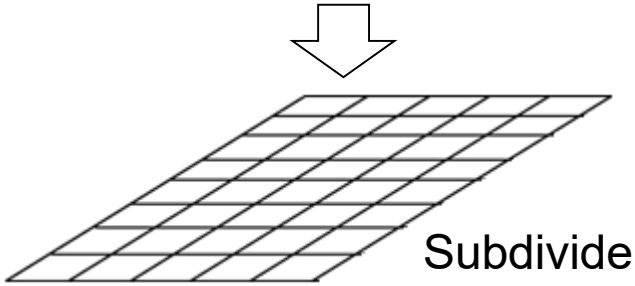
➢eq. motion $\tau_i = \sum_{i=1}^{N} K_{ij}(u_j - V_{pl}t) - \frac{\xi G}{2V_s} V_i$

➢friction law

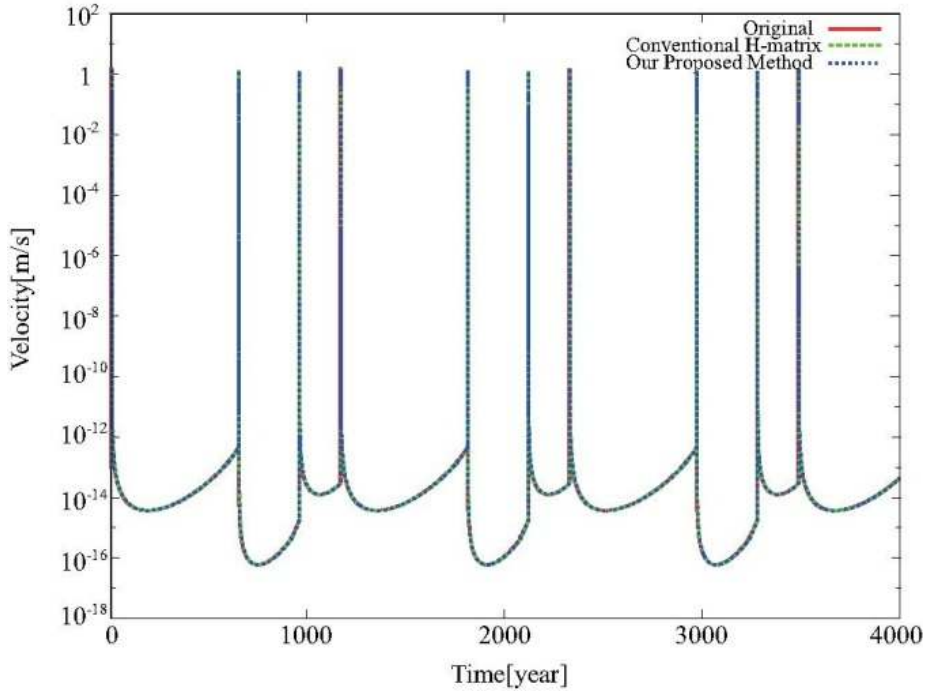
$$\tau_i = \mu\sigma_n^{eff} = \tau_* + A_i \ln(V_i / V_*) + B_i \ln(V_*\theta_i / L_i)$$

$$\frac{d\theta_i}{dt} = \exp(-V_i / V_c) - V_i\theta_i / L_i \ln(V_i\theta_i / L_i)$$



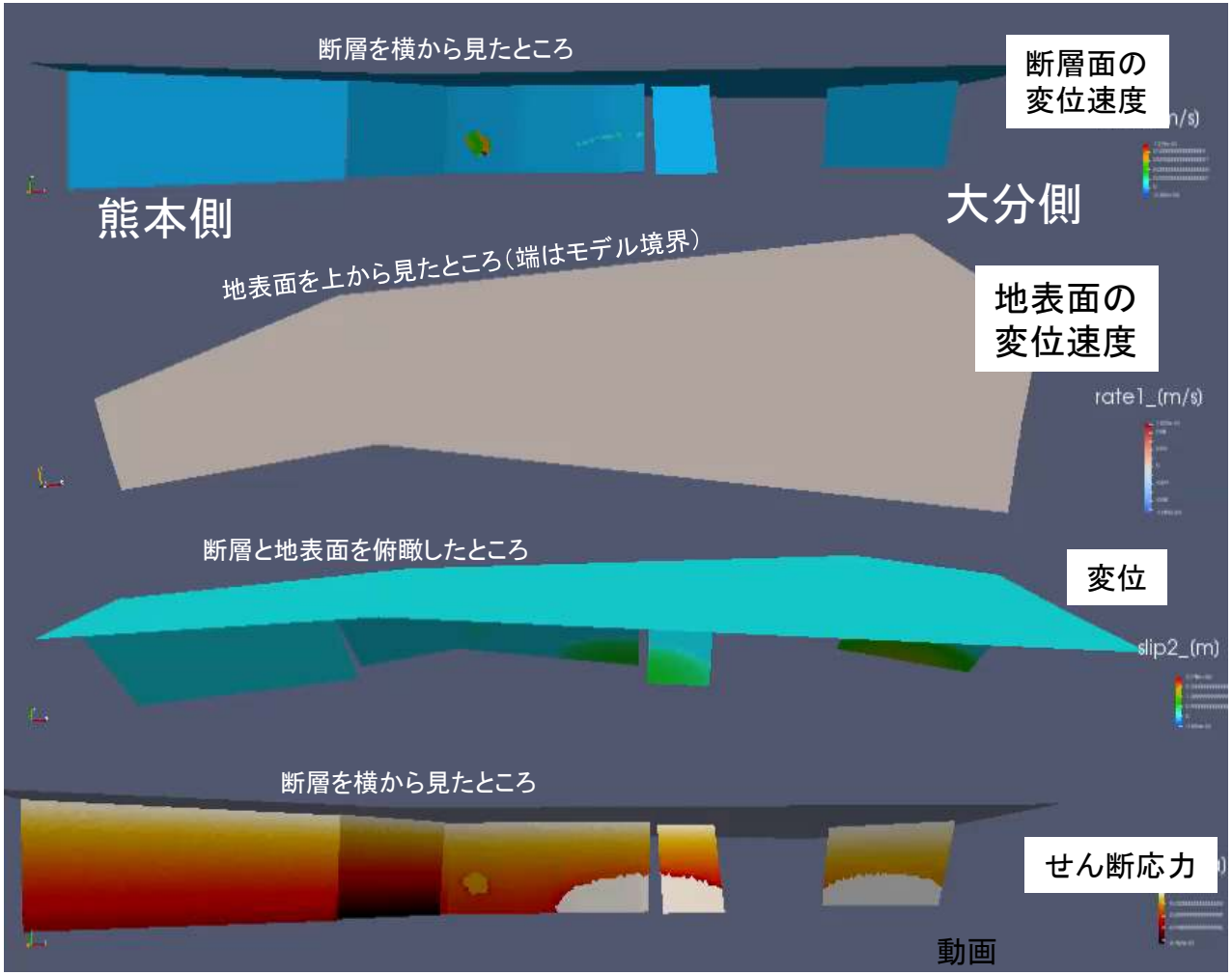


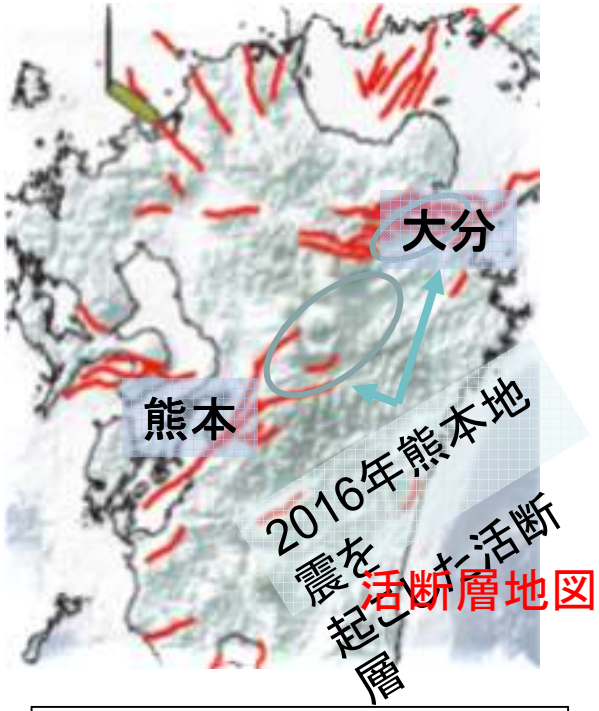
Subdivide fault surface



$\mathcal{H}$ACApKの適用例(地震発生物理学:動的破壊シミュレーション)

2016年熊本地震の例:
- 熊本側の本震と，大分側の動的誘発地震の発生を動力学的に再現
- 現実的な断層形状を考慮
- 動的誘発地震のメカニズムを解明:　本震が放射した強い応力波が50km以上を伝播





空間要素数:18,000
時間要素数:　1,200
計算機:Xeonクラスタ
並列数:24コア×2ノード
計算時間:約10時間

$\mathcal{H}$ACApKの適用例(地震発生物理学:動的破壊シミュレーション)

- オリジナルBEM

  積分核: K_{ijm}

- 高速領域分割法(FDPM): Ando (2016, GJI)

  積分核の領域ごとの時間依存性と積分領域形状の違いを利用

  - 積分核: $\begin{cases} K_{i\,j\,m} & in\ \text{Domain F} \\ K^{IA}_{i\,j}\ G_m + K^{IB}_{i\,j} & in\ \text{Domain I} \\ K^{S}_{i\,j} & in\ \text{Domain S} \end{cases}$

    →時空間依存だが高々面積分

    →時間項を分離した体積積分

    →時間項のない面積分

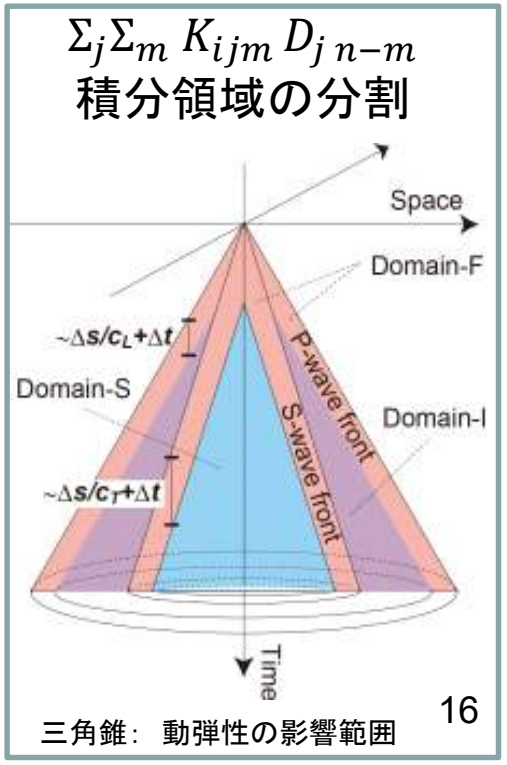
$$K^{IA}_{i\,j}\ G_m = \begin{bmatrix} k_1 \\ \vdots \\ k_j \end{bmatrix}_i [g_1 \quad \cdots \quad g_m]$$

効率化:密行列→ベクトルの積

- FDPMによる計算コストの削減
  - $O(M^2N) \rightarrow O(M^2)$ メモリ
  - $O(MN^3) \rightarrow O(MN^2) \sim O(M^2)$ 演算数

    (M: 要素数. N :時間ステップ数)

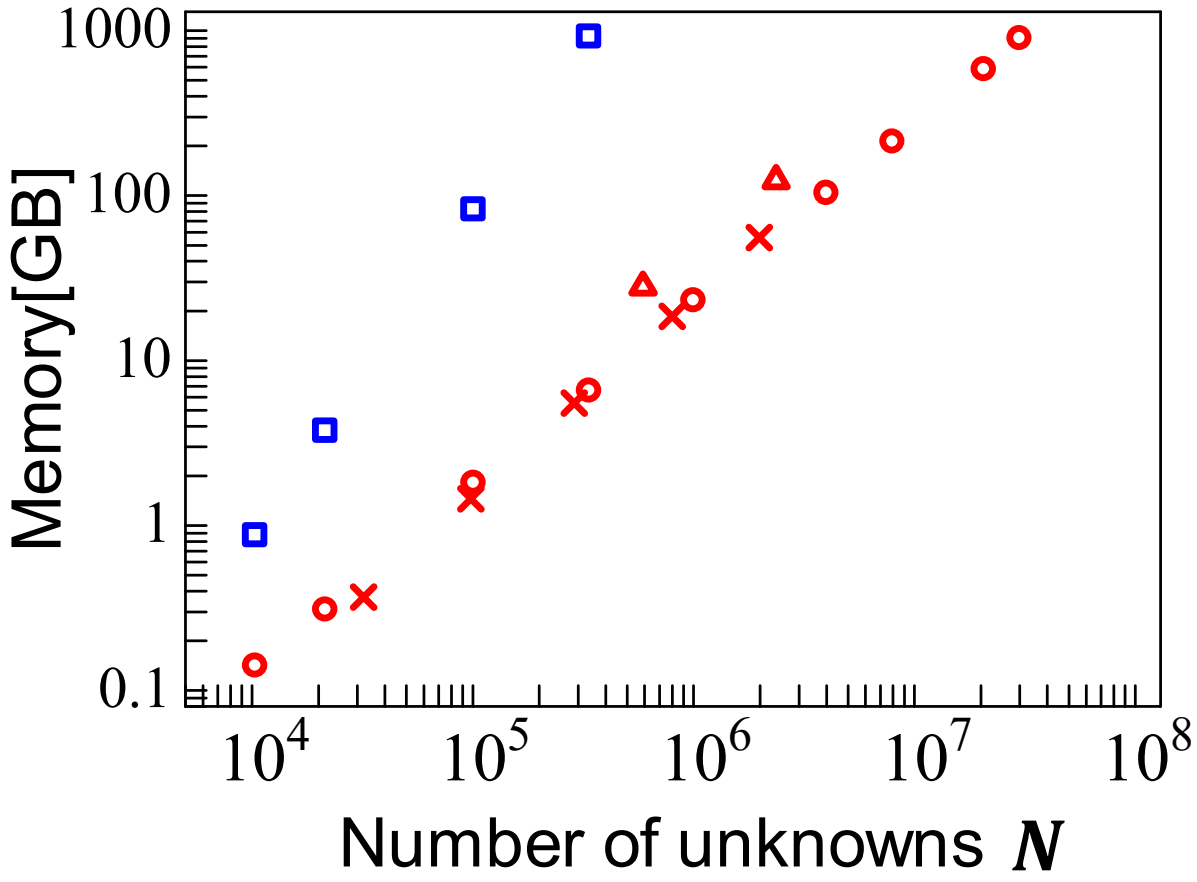
まだ減らしたい→H行列法適用



三角錐: 動弾性の影響範囲

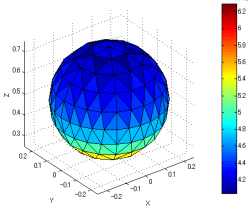
$\mathcal{H}$ACApKと密行列のメモリ使用量

■ $\mathcal{H}$-matrices with ACA reduce memory usage.

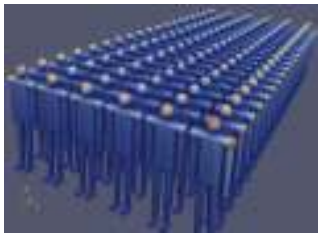


Memory usage (Log-Log scale)

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



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



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

1　$\mathcal{H}$**ACApK**ライブラリの概要

2　低ランク構造行列法

3　(格子)$\mathcal{H}$行列法の並列化

4　$\mathcal{H}$**ACApK**ライブラリの使用方法

5　今後の取り組み

低ランク行列近似(LRA)とベクトルの直積

■ベクトルの直積は「ランク1の行列」

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

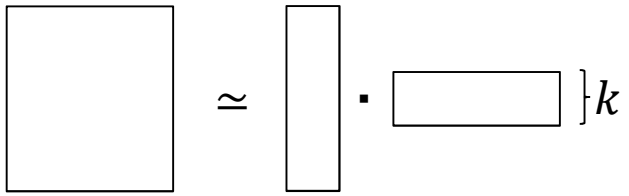
■任意の行列はベクトルの直積（ランク1行列）で近似できる（精度は度外視）

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

■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



低ランク構造行列法

■積分方程式法に現れる密行列に対する近似手法

適用対象例

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

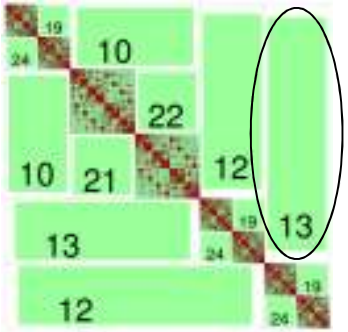
Degenerate kernel: $g(x,y) \cong \sum_{r=1}^{k} g_1^r(x) g_2^r(y)$ for far remote $\boldsymbol{x}, \boldsymbol{y}$

離散化

低ランク構造行列法

Full rank 密行列

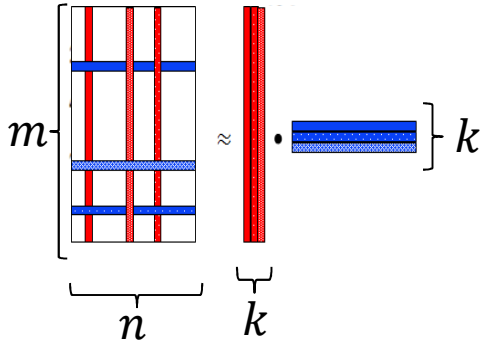
Permutation
Partition



Full-Rank
Low-Rank

・行列全体ではfull rank

LRA



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

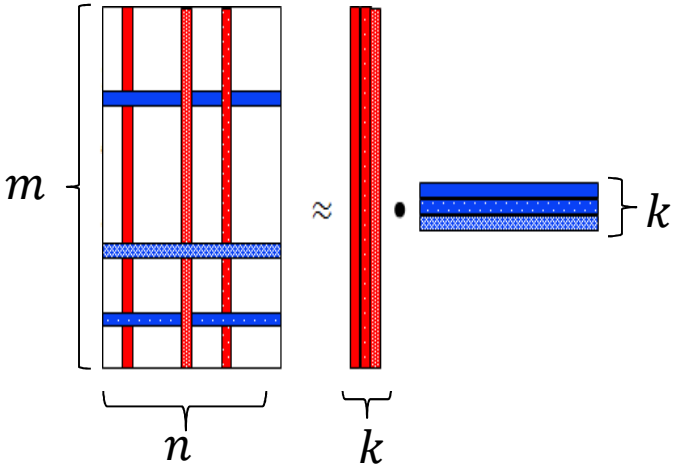
・行列要素数
・行列ベクトル積演算数

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: $\epsilon_k < \epsilon_{k-1}$

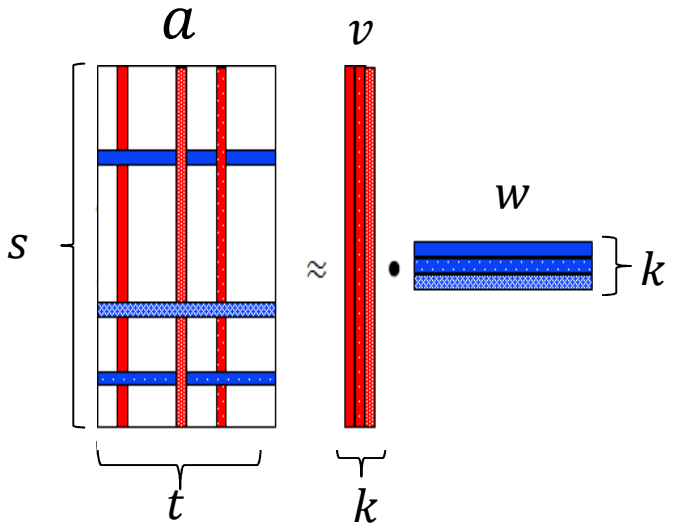
■選択するベクトルの本数により使用メモリ量と近似精度を制御可能.

Procedure of ACA with full pivoting (ACA_full)

■Pivoting and updating matrix need computational cost $\boldsymbol{O(N^2)}$.

Input: $a \in \mathbb{R}^{s \times t}$.
Output: $v^k \in \mathbb{R}^{|s|}, w^k \in \mathbb{R}^{|t|}$
s. t. $\tilde{a} = \sum_{k=1}^{r} v^k (w^k)^T$.

$r = 0, \epsilon_r = 1$
For a given error tolerance $\varepsilon < 1$,
Do while $\boldsymbol{\epsilon_r} > \varepsilon$
$r = r + 1$
$(i^*, j^*) := \mathrm{argmax}_{(i,j)} |a_{i,j}|,\ \delta := a_{i^*,j^*}$ ← Pivoting : $\boldsymbol{O(N^2)}$ searching over whole matrix
$(v^r)_i := a_{i,j^*},\ i \in s,$
$(w^r)_j := a_{i^*,j}/\delta,\ j \in t,$
$\epsilon_r := ||v^r||_2 ||w^r||_2 / \sqrt{\sum_{k=1}^{r} ||v^k||_2^2 ||w^k||_2^2}$ ← Estimation of approximation error
$a_{i,j} := a_{i,j} - (v^r)_i (w^r)_j, i \in s, j \in t$ ← Updating matrix : $\boldsymbol{O(N^2)}$ Subtracting rank 1 matrix
End do



Procedure of ACA with partial pivoting (ACA_part)

Input: $a \in \mathbb{R}^{s \times t}$.

Output: Sequences of vectors $v^k \in \mathbb{R}^{|s|}, w^k \in \mathbb{R}^{|t|}$ such that $\tilde{a} = \sum_{k=1}^{r} v^k (w^k)^T$.

Set $R^1 = a$, $j^* := \forall j \in t$, $i^* := \operatorname{argmax}_{i \in s,} |R^1_{i,j^*}|$,

$r = 1, \epsilon_r = 1$ and $\wp := \{\}$.

For a given error tolerance $\varepsilon < 1$,

Do while $\epsilon_r > \varepsilon$

$j^* := \operatorname{argmax}_{j \in t} |R^r{}_{i^*,j}|$

$\delta := R^r{}_{i^*,j^*} - \sum_{k=1}^{r-1} (v^k)_{i^*} \cdot (w^k)_{j^*}$

$(v^r)_i := R^r{}_{i,j^*} - \sum_{k=1}^{r-1} (v^k)_i \cdot (w^k)_{j^*}, \quad i \in s,$

$(w^r)_j := \left(R^r{}_{i^*,j} - \sum_{k=1}^{r-1} (v^k)_{i^*} \cdot (w^k)_j\right)/\delta, \quad j \in t,$

$r = r + 1$

$\epsilon_r := \|v^r\|_2 \|w^r\|_2 / \sqrt{\sum_{k=1}^{r} \|v^k\|_2^2 \|w^k\|_2^2}$

$\wp := \wp \cup \{i^*\}$

$i^* := \operatorname{argmax}_{i \in s \setminus \wp} |(v^r)_i|$

End do

■Computational cost : $\boldsymbol{O(N)}$

Partial pivoting : $\boldsymbol{O(N)}$
searching around a vector

Vector calculations : $\boldsymbol{O(N)}$
instead of updating matrix

Estimation of approximation error

Omitting the selected row from searching space

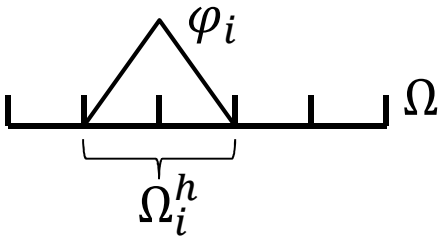
低ランク構造行列法の近似原理

■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 ⇩

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



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

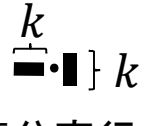
遠く離れた2点x, yに対しては，

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

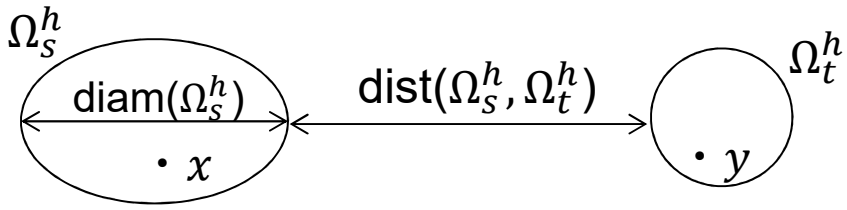
変数分離した関数積の級数で近似可能

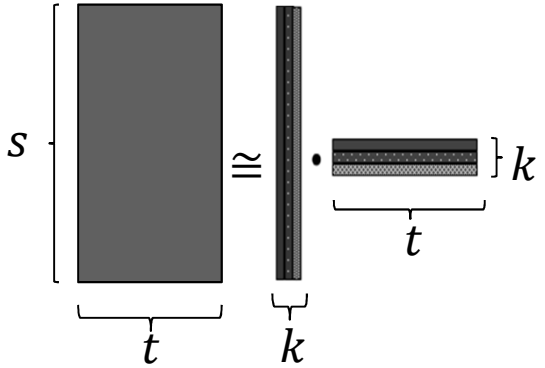
$$A_{ij} = \int_\Omega \varphi_i(x) \int_\Omega g(x,y)\,\varphi_j(y)\mathrm{d}y\mathrm{d}x$$

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



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





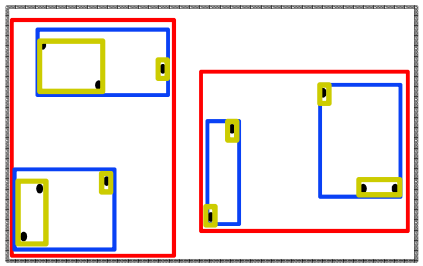
近似許容：min{diam(Ω_s^h), diam(Ω_t^h)}≤ ηdist(Ω_s^h, Ω_t^h)

低ランク構造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

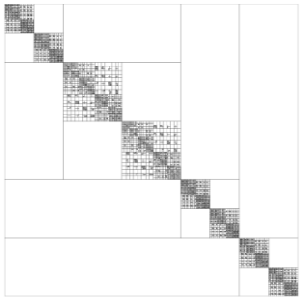
Step 1 Clustering & Construction of a cluster tree

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



Step 2 Construction of a partition

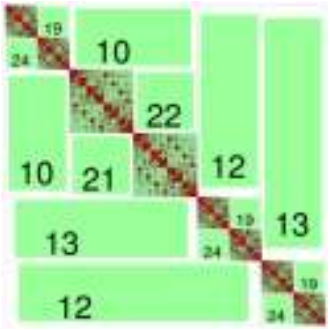
■ There're only blocks(frames of sub-matrices).



Step 3 Fill in sub-matrices

■Low-rank approximation

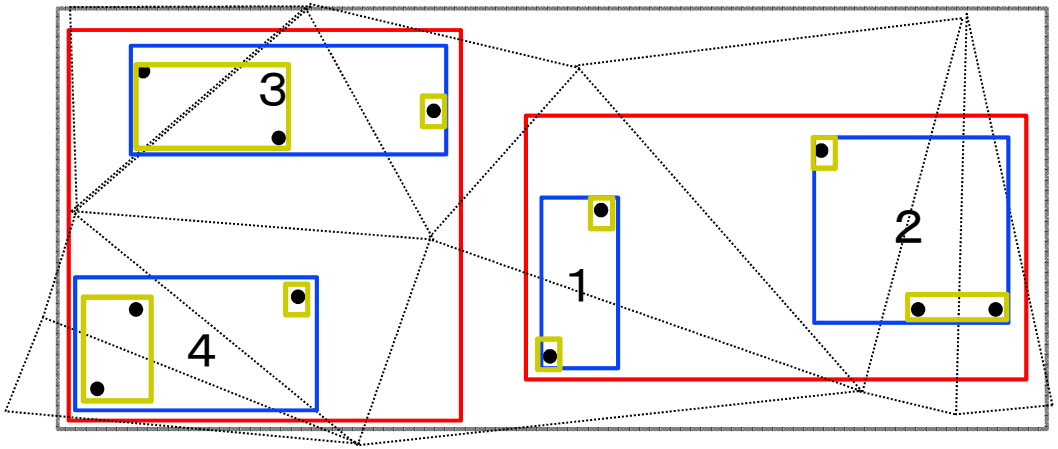
- Cross approximation i) ACA ii) ACA+
- Interpolation
- Hybrid method of interpolation and ACA

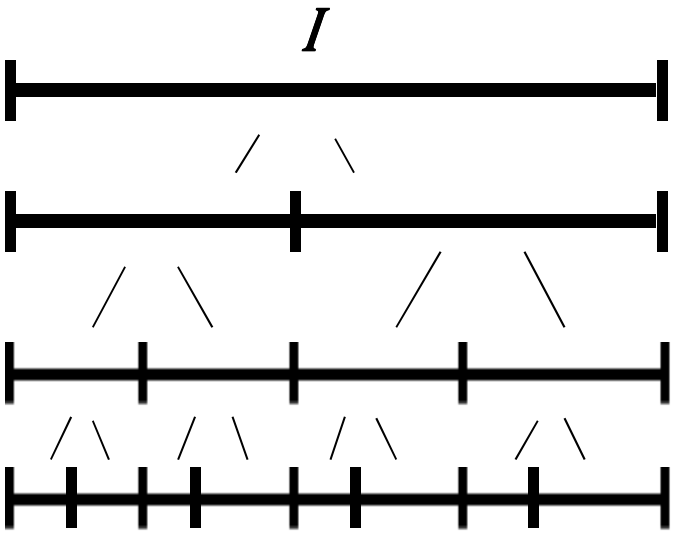


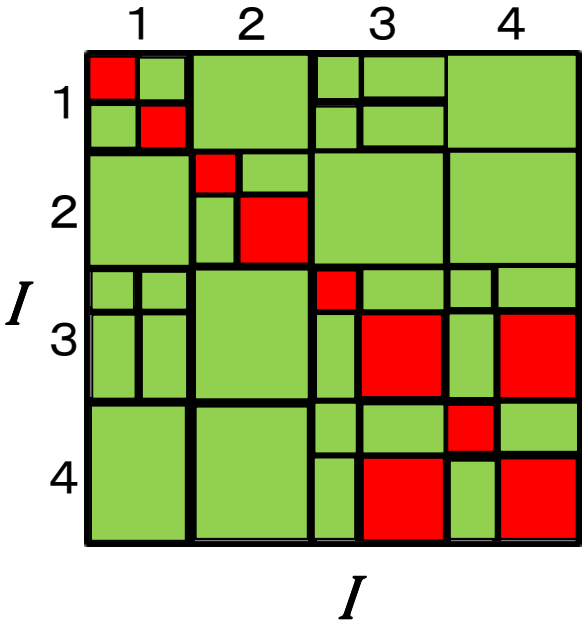
$\mathcal{H}$ACApKにおける行列分割の方法（従来$\mathcal{H}$行列の場合）

■幾何情報に基づくクラスタリングを利用

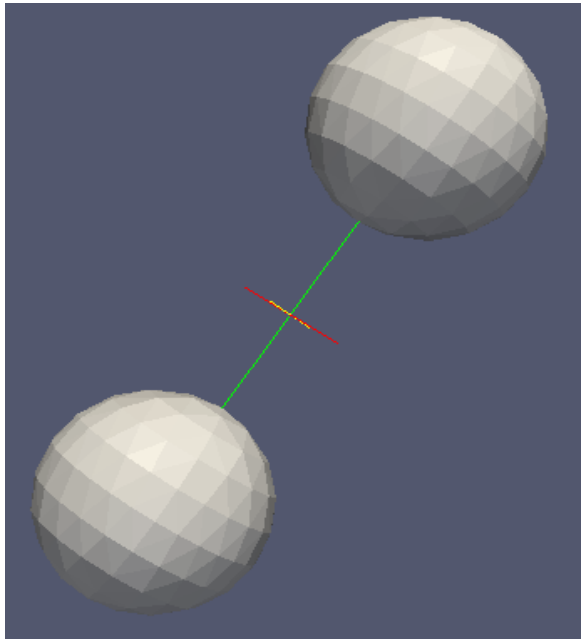
Admissible condition：$\min\{\mathrm{diam}(\Omega_s^h), \mathrm{diam}(\Omega_t^h)\} \leq \eta \mathrm{dist}(\Omega_s^h, \Omega_t^h)$

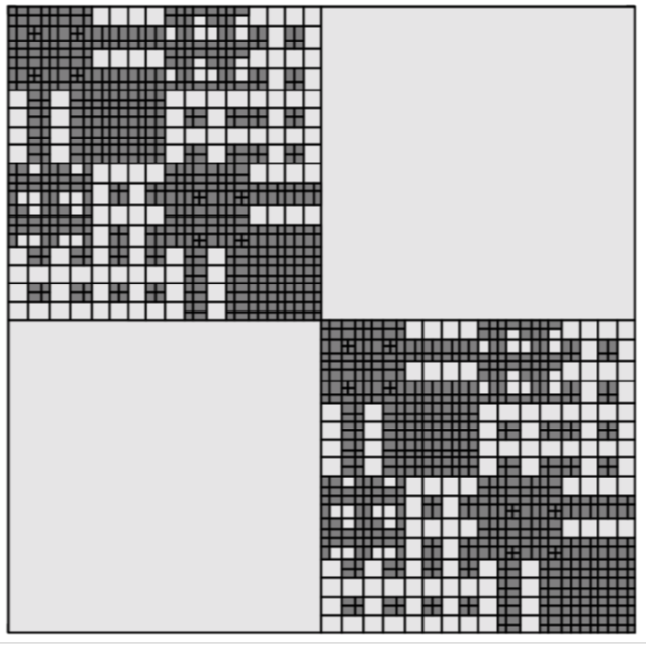




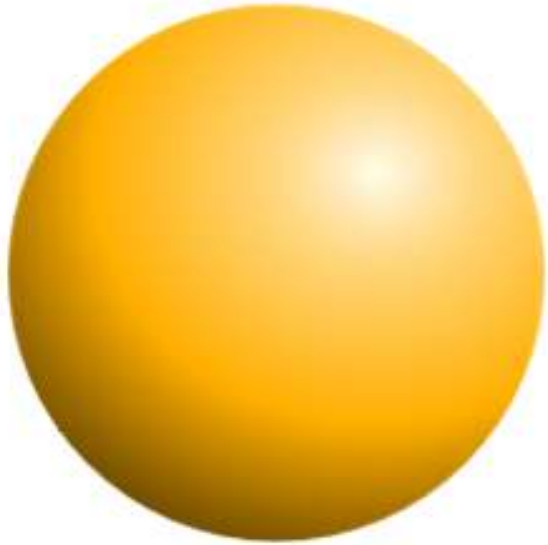


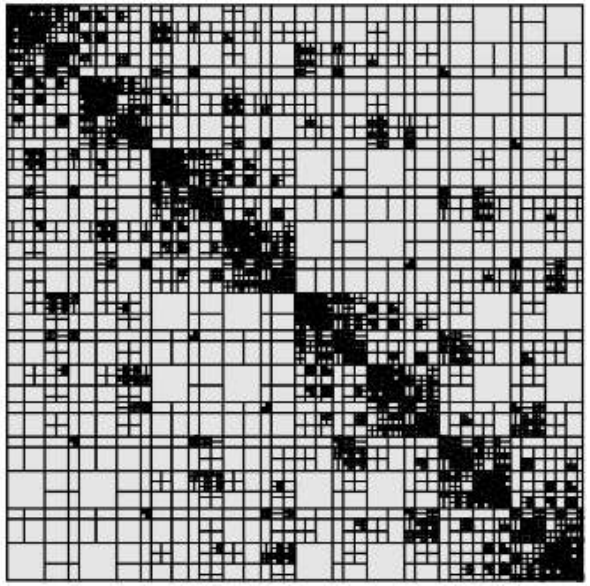
従来$\mathcal{H}$行列法を用いた行列分割の例

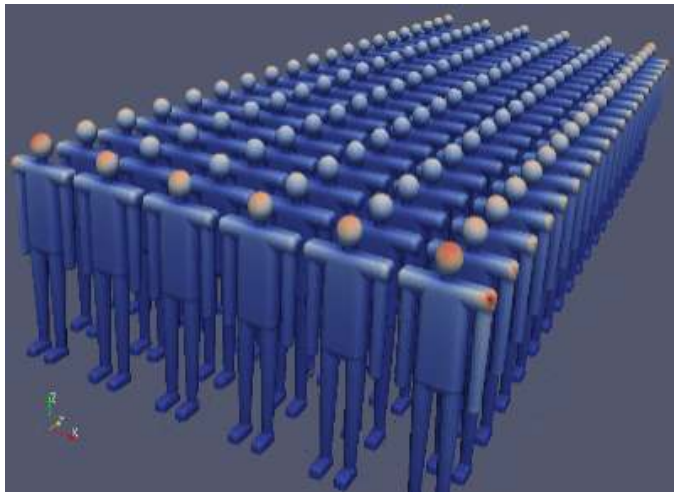


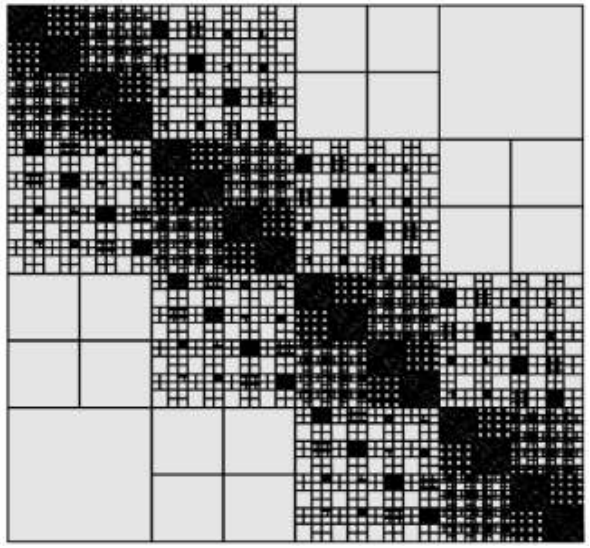


従来$\mathcal{H}$行列法を用いた行列分割の例



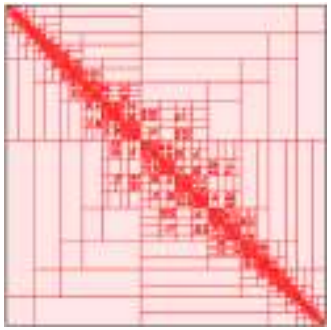
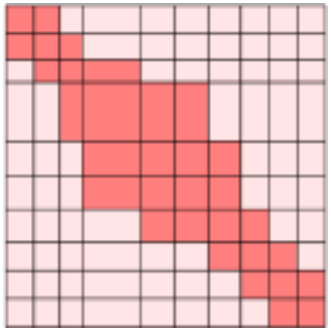
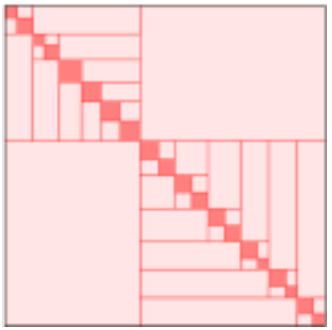






低ランク構造行列法と近縁手法の種類

■行列分割構造と要素保存法

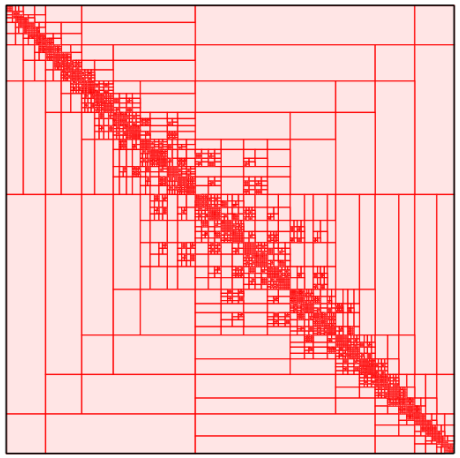
行列要素保存法 ＼ 分割構造	Strong admissibility 	Block 	Weak admissibility 
数値を直接保存	従来$\mathcal{H}$-行列	BLR-行列	HODLR
	格子$\mathcal{H}$-行列		
基底ベクトルと係数	$\mathcal{H}^2$-行列		HSS
基底関数と係数	FMM		Tree

・青字：$\mathcal{H}$ACApKで取り扱える手法

既存の低ランク構造行列法の比較

■分割構造の違いにより、一長一短

・従来$\mathcal{H}$-行列



・Complicated structure

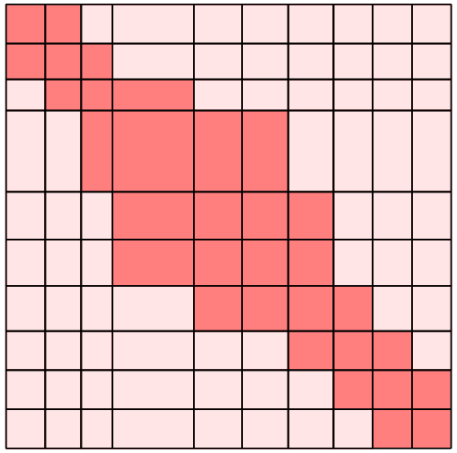
効率的な演算・通信パターンの構築が困難

・**High compressibility**

Memory : $\boldsymbol{O(N\log N)}$

・Strong admissible cond.

・**BLR, TLR**



・**Simple structure**

like block-divided matrix

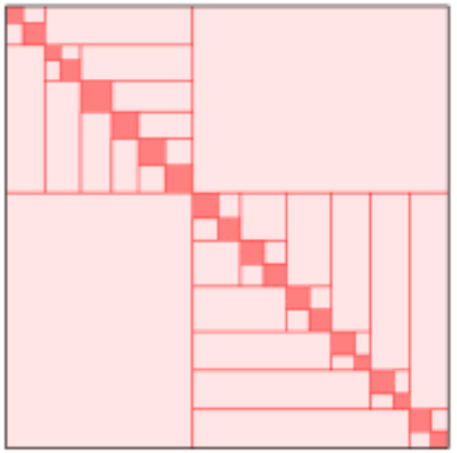
密行列用アルゴリズムを流用可能

・Low compressibility

Memory : $\boldsymbol{O(N^{1.5})}$

・Strong admissible cond.

・HSS, HODLR



・Relatively simple structure

・Very high compressibility for low dimensional problem

・Worse asymptotic complexity for high dimensional problem

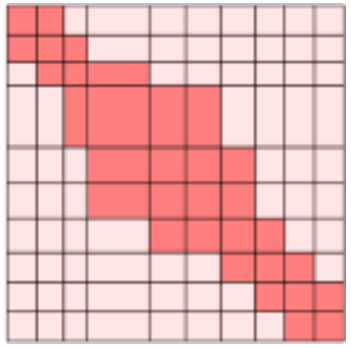
・Weak admissible cond.
(All off-diagonal : Low-rank)

格子$\mathcal{H}$行列法 [A.Ida : IPDPS'18]

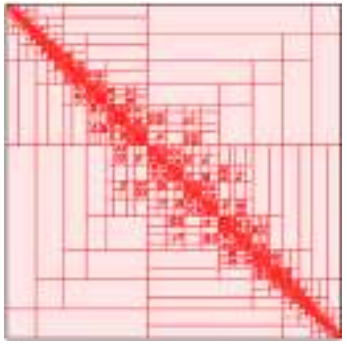
■低ランク構造行列法の新手法

- Hybrid of $\mathcal{H}$-matrices and BLR-matrices
- Advantages of lattice $\mathcal{H}$-matrices
  - High compressibility of $\mathcal{H}$-matrices
  - Convenience of matrix arithmetic with BLR-matrices

・BLR-matrix



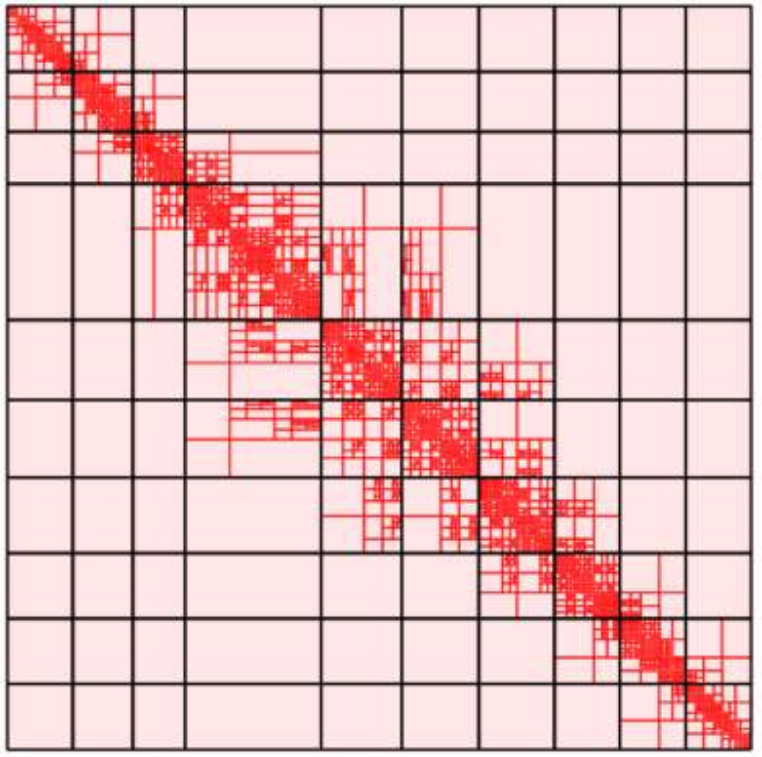
・$\mathcal{H}$-matrix



Lattice structure

$\mathcal{H}$-submatrices
in blocks on lattice

・Lattice $\mathcal{H}$-matrix



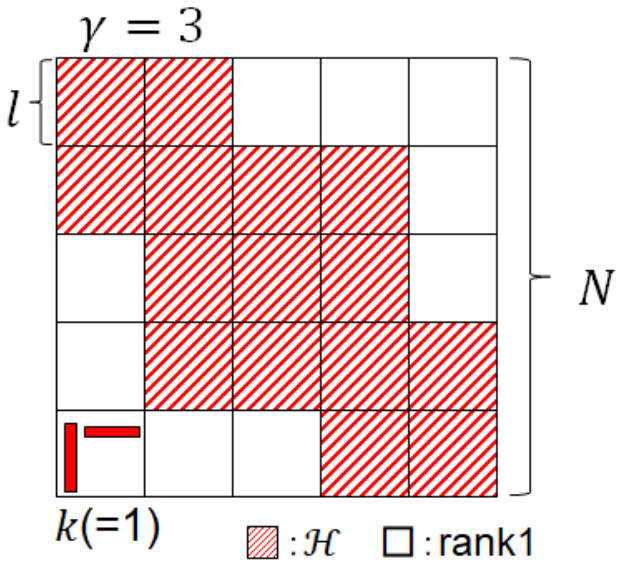
格子$\mathcal{H}$行列のメモリ使用量

■格子サイズの決め方に依存

・l：格子サイズ

・$l = N$ ⇨ 従来$\mathcal{H}$行列

・$l = 1$ ⇨ 密行列



・$l = \alpha N$ for a constant α $(\frac{1}{N} \leq \alpha \leq 1)$: 格子数固定

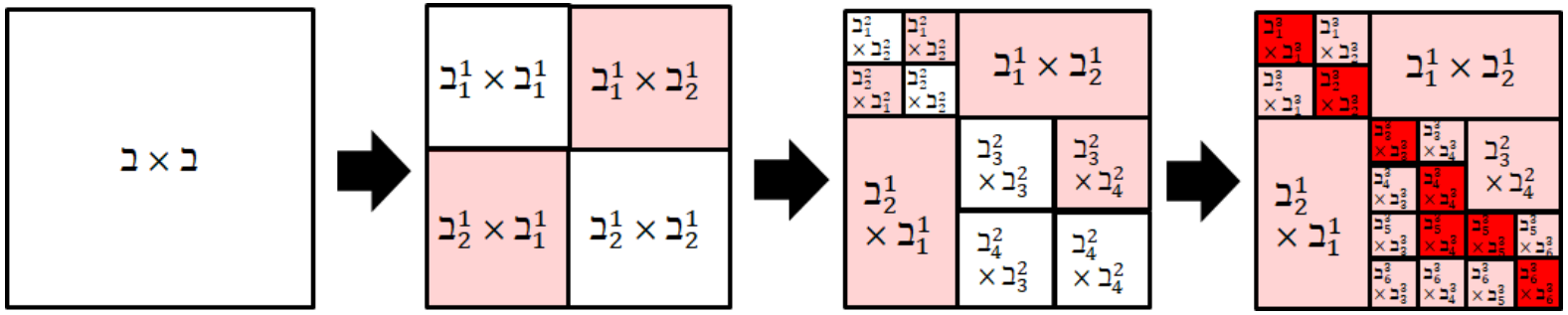
⇨ 要素数 : $f(N) = 2\gamma \boldsymbol{N \log N} + N(2\gamma \log \alpha + 2/\alpha - \gamma)$

・格子サイズ l はMPIプロセス数に応じて決められるべき

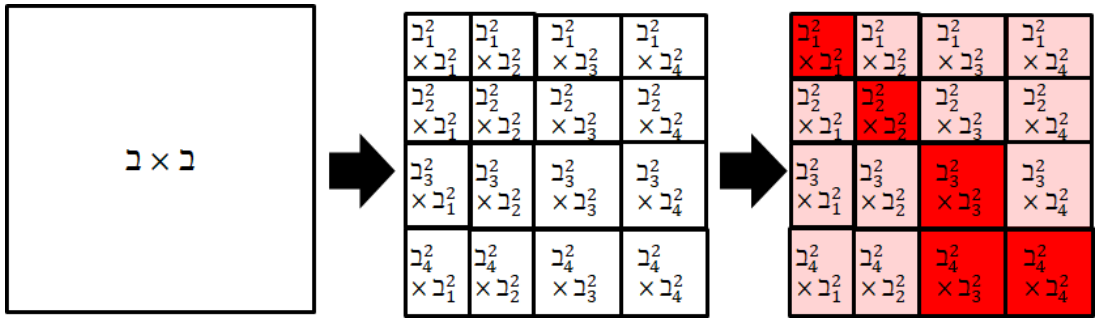
⇨ 利用可能なMPIプロセス数は行列サイズNに応じて制限

低ランク構造行列の行列分割方法

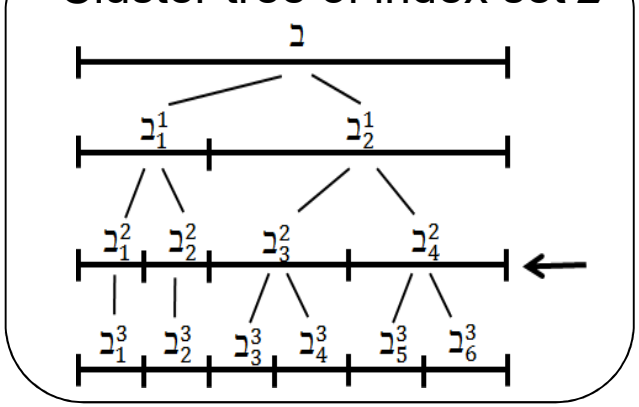
- 既存$\mathcal{H}$行列



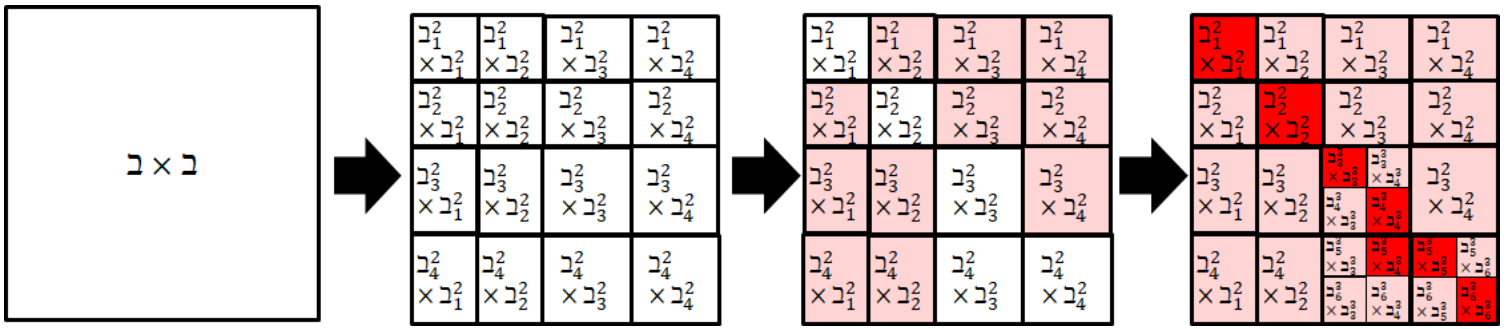
- BLR行列



- Cluster tree of index set I



- 格子 $\mathcal{H}$行列



1　$\mathcal{H}$**ACApK**ライブラリの概要

2　低ランク構造行列法

3　（格子）$\mathcal{H}$行列法の並列化

4　$\mathcal{H}$**ACApK**ライブラリの使用方法

5　今後の取り組み

境界要素法解析で必要なℋACApKの2機能

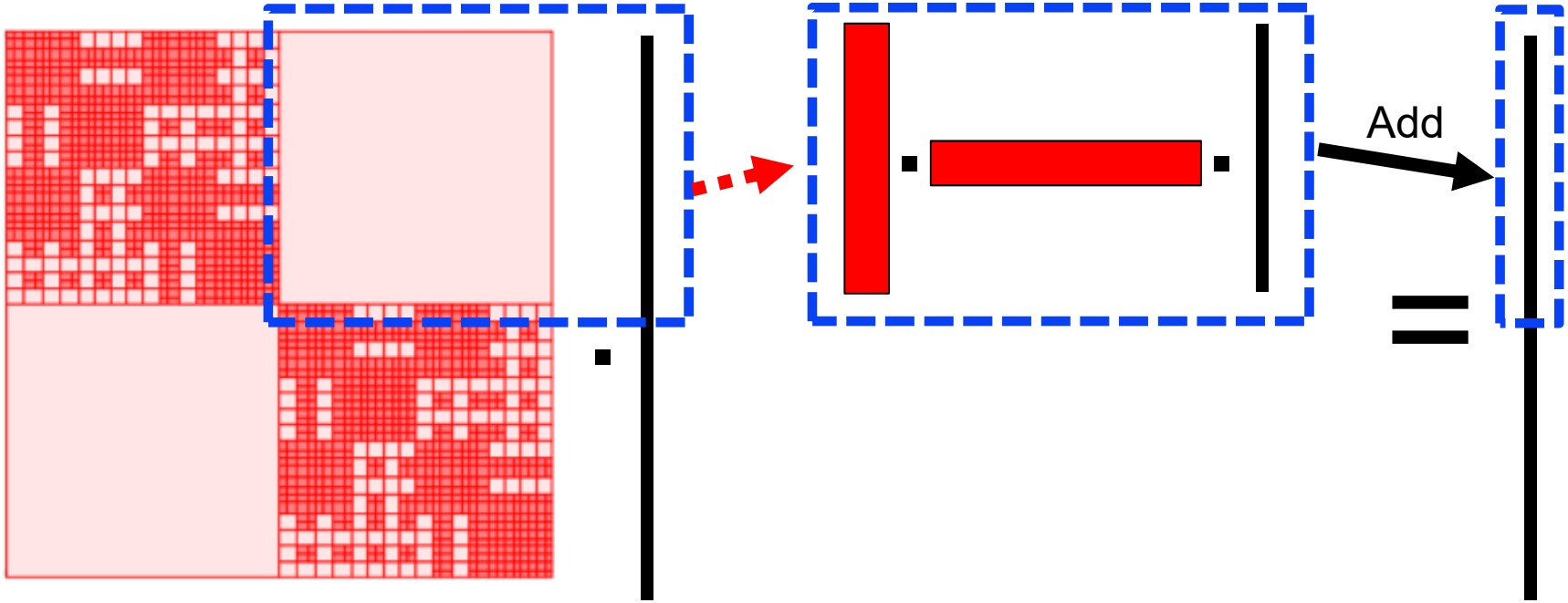
演算は部分行列ごとに実施

■ 低ランク構造行列生成

・低ランク近似(LRA)は各ブロックで個別独立に実施

■ 低ランク構造行列–ベクトル積

・部分行列-ベクトル積の集まり



$\mathcal{H}$ACApKにおける低ランク構造行列法の並列化

■低ランク構造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.
  （step1,2も並列化した版は公開準備中）
- MPI の通信は不要.

step1 クラスタ ツリー 作成 step2 $\mathcal{H}$行列構造 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

■低ランク構造行列-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（ベクトル分散版は公開準備中）
- 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.

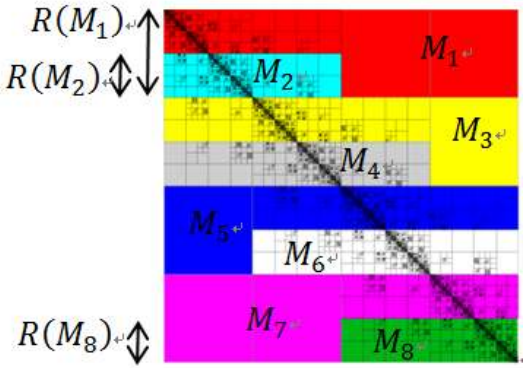
$\mathcal{H}$ACApKにおける従来$\mathcal{H}$行列の並列計算手法

基本的に$\mathcal{H}$行列を行方向にスライスするが、部分行列は切断しない

■ MPIプロセス割当て戦略

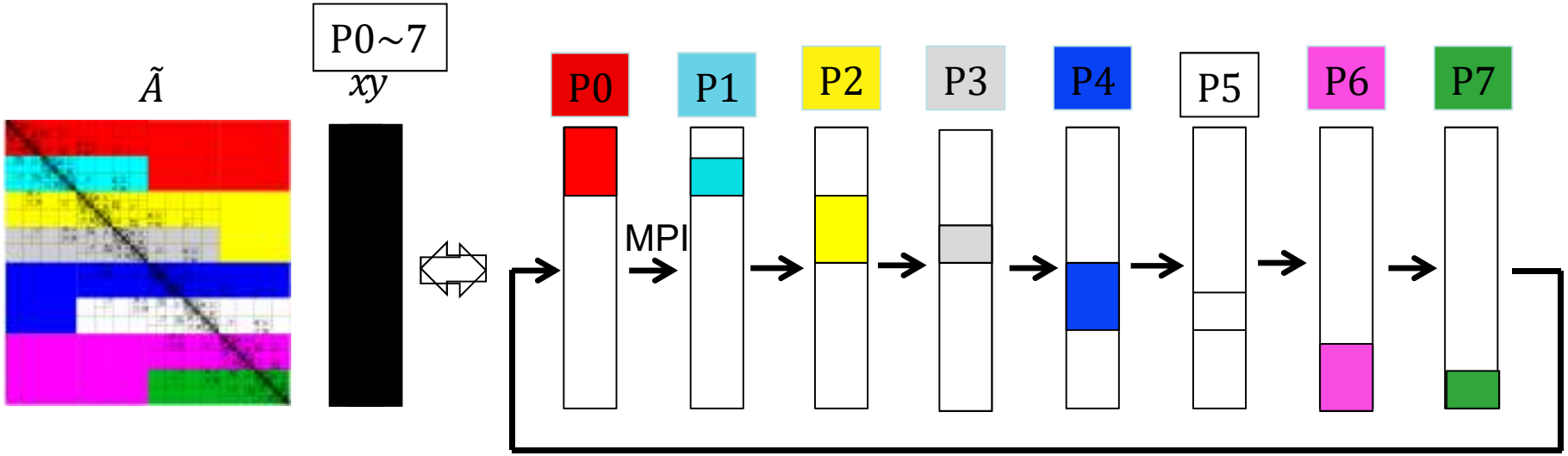
① 各ブロックの高さ$R(M_k)$を可能な限り最小化
⇒ 通信データサイズの最小化

② MPIプロセス間の計算負荷不均衡を最小化



■ 従来$\mathcal{H}$行列-ベクトル積 $y := \tilde{A}x$

・各MPIプロセスで部分行列-ベクトル積を行った後に隣接間通信



格子$\mathcal{H}$行列に対するMPIプロセス割当て戦略

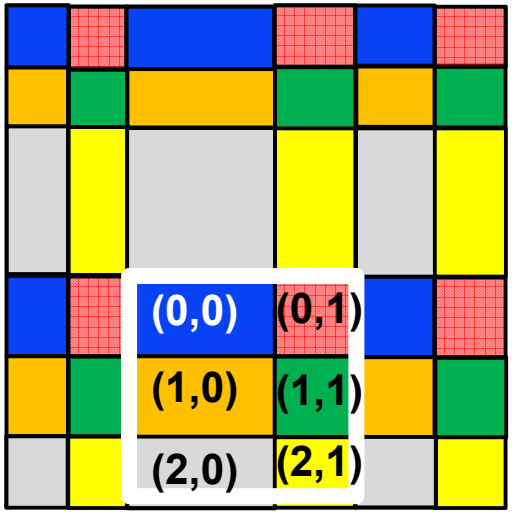
■「プロセスグリッド」に基づく2次元サイクリック割当て

・各MPIプロセスへの割当て：同色で塗られた格子

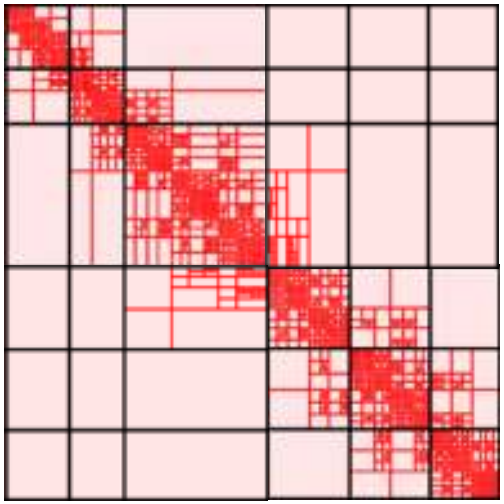
-複雑な構造は逐次orスレッド並列では許容範囲

-密行列向けの既存アルゴリズムを流用可能

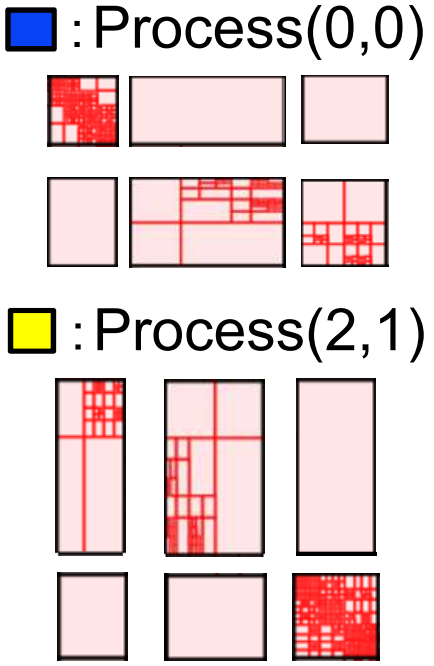
■ 例: 3×2 プロセスグリッド (6 MPI-processes)



Assignment global view



Lattice $\mathcal{H}$-matrix



Local view

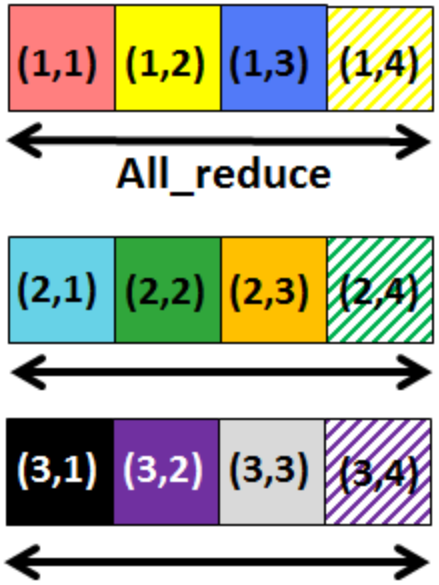
格子$\mathcal{H}$行列-ベクトル積, $y := \widetilde{A}x$

アルゴリズム 1 : “LatticeHMVM_fullVector”

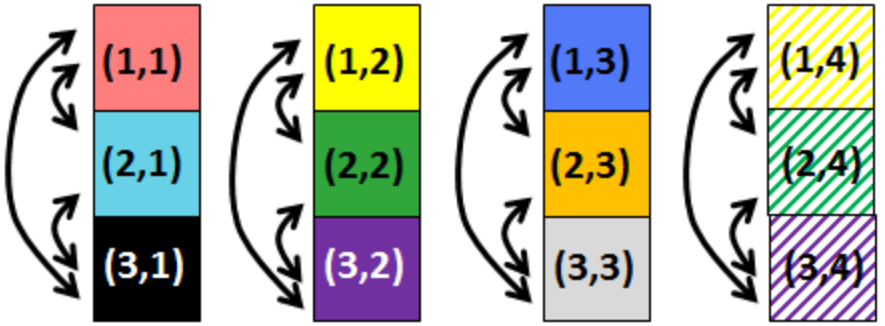
■各MPIプロセスで部分行列-ベクトル積を実行し部分ベクトルを得る

■全プロセスで全ベクトル y を得るために,

① Obtain a part of the result vector
by MPI_All_reduce
in each row in process grid.



② Exchange parts of vector
by MPI_iSEND and iRECV
in each column.



格子$\mathcal{H}$行列-ベクトル積, $y := \tilde{A}x$（ベクトル分散版）

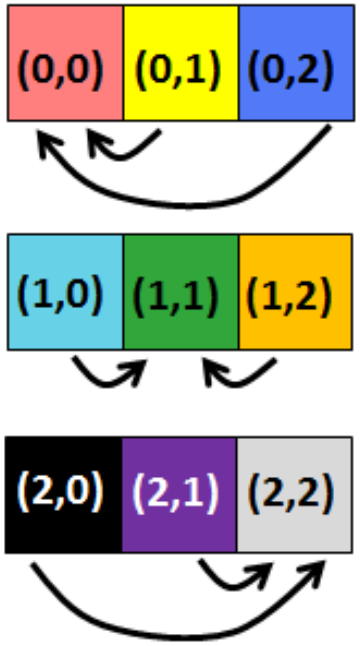
アルゴリズム 2 : “LatticeHMVM _partVector”

■正方形プロセスグリッド限定: $N_{pr} = N_{pl} = \sqrt{N_p}$

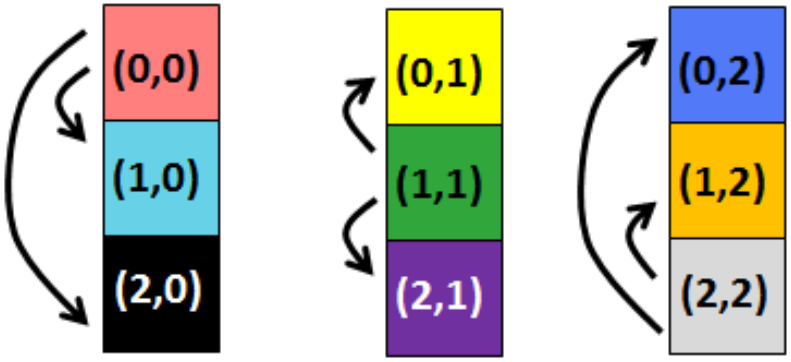
・各MPIプロセスが全ベクトルを持つ必要がない

■次の $\tilde{A}x$を計算するため必要な部分ベクトルのみを通信

① MPI_reduce in each row to diagonal processes



② MPI_Broadcast in each column from diagonal processes to other processes



数値実験：Current induced Domain Wall Motion(CDWM)

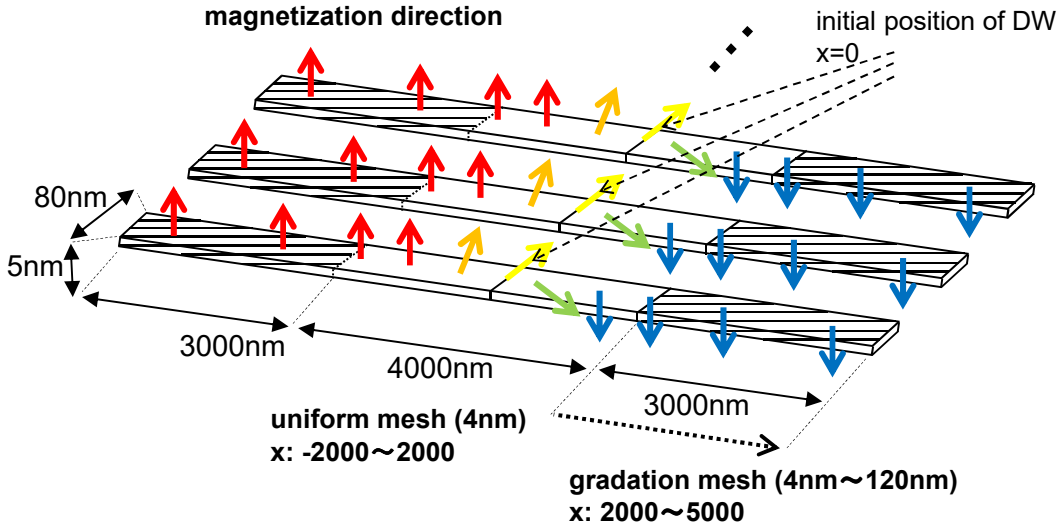
■ (格子) $\mathcal{H}$-行列の近似精度

・Tolerance ε =1.0e-4 for $||A - \tilde{A}||_F \leq \varepsilon$.

A : Dense matrix,
$\tilde{A}$: $\mathcal{H}$-matrix

■計算メッシュ

・ Thin films of magnetic material



・Single film model
N =20,910
N =31,860
N =116,100

・Double film model
N =232,200

・Triple film model
N =348,300

■計算機: Fujitsu PRIMERGY CX2550 (北海道大学)

・Processor : Intel Xeon™ Gold 6148 (40cores/node)

・Memory : 384GB/node, ・Network : 12.5 GB/s, Fat Tree.

格子$\mathcal{H}$-行列の計算オーダー

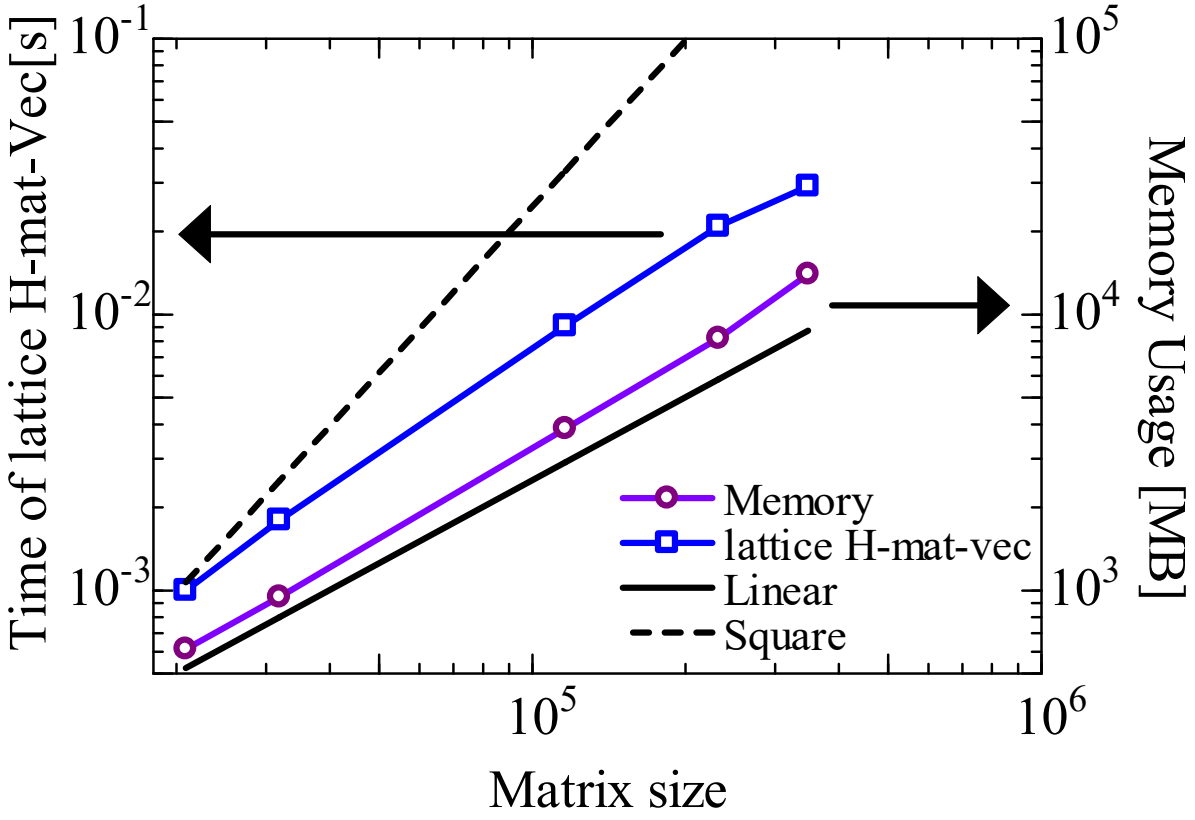
■ 格子サイズ l の決め方で $O(N \log N) \sim O(N^2)$

・$l = N$ ⇨ 従来 $\mathcal{H}$-行列　　・$l = 1$ ⇨ 密行列

■ MPI数で格子サイズを決定 ($l = N/\sqrt{9N_p}$)： $\boldsymbol{O(N \log N)}$

・Memory usage of Lattice $\mathcal{H}$-matrices

・Calculation time of Lattice $\mathcal{H}$MVM when using 400 MPI processes



MPI プロセス間の計算負荷均衡

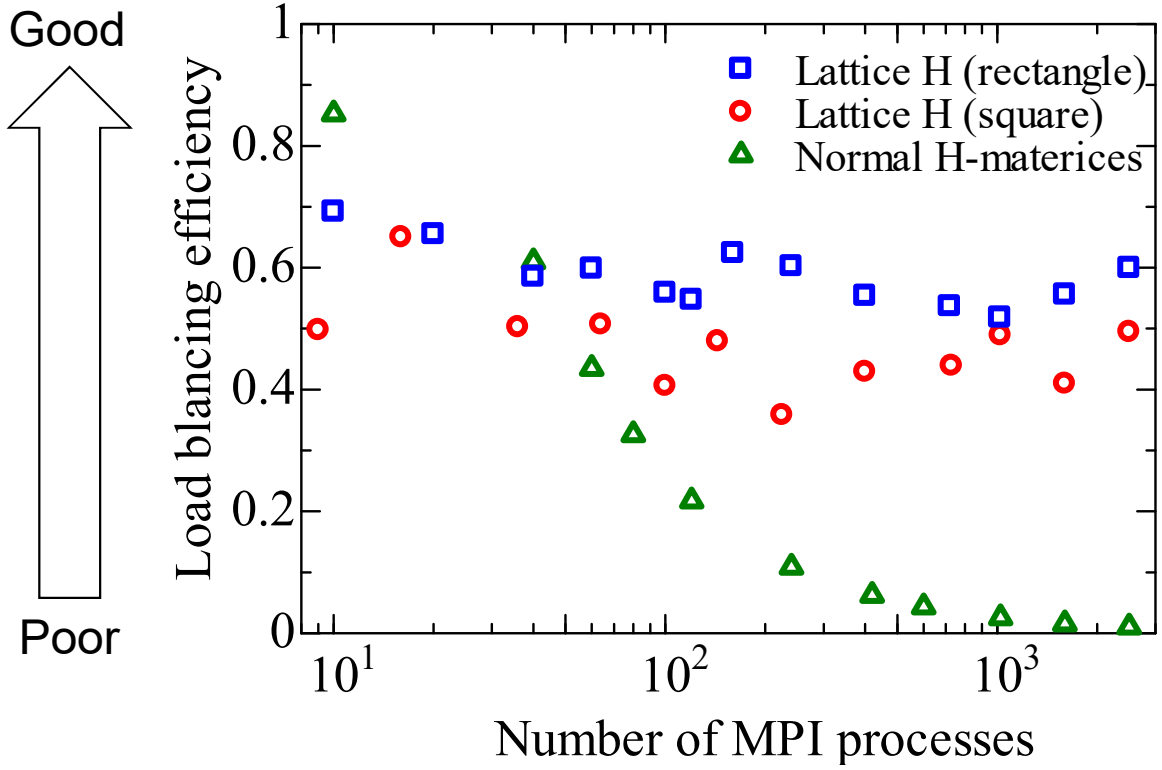
■計算負荷均衡効率

$$E := \frac{\mu}{\mu_{\mathrm{pmax}} N_p}$$

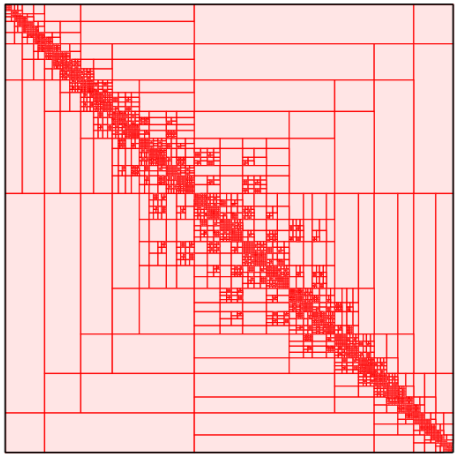
μ：Memory usage (entire matrix)

μ_{pmax}：The largest memory (an MPI process)

- 従来$\mathcal{H}$行列：MPIプロセス数の増加により効率が急速に低下
- 格子$\mathcal{H}$行列：効率低下の度合いが穏やか

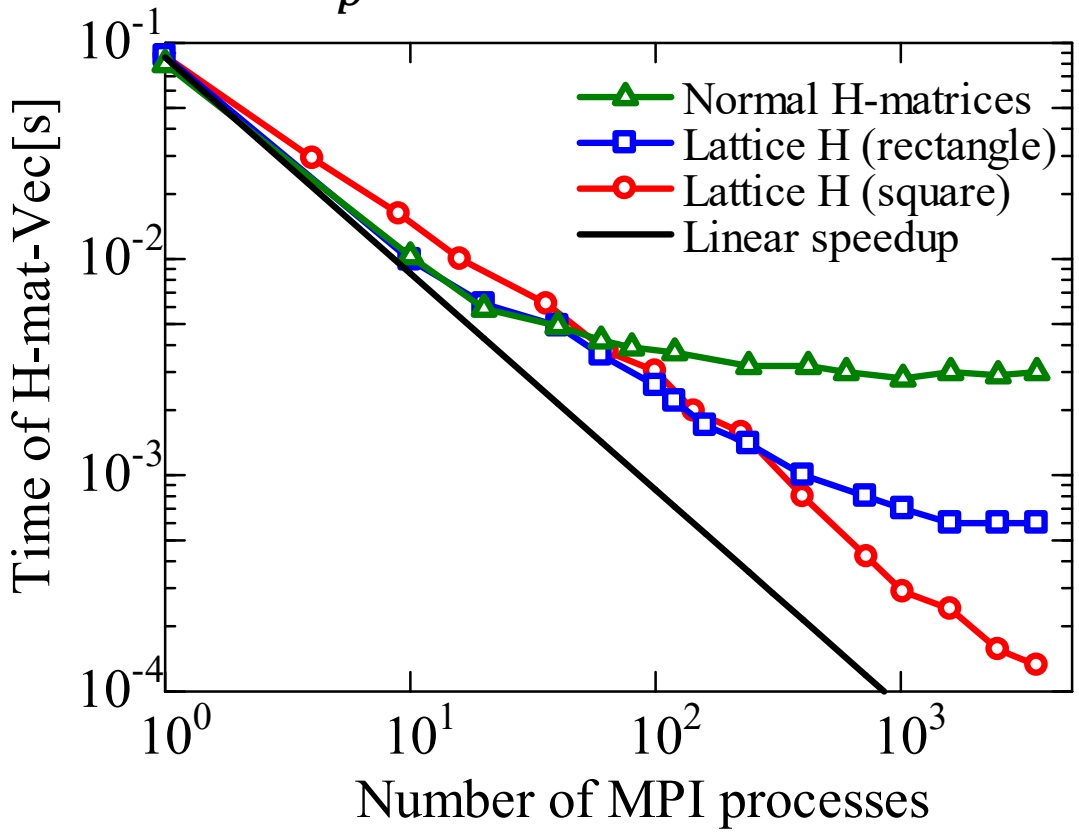


- ” Single film model”
  N =20,910

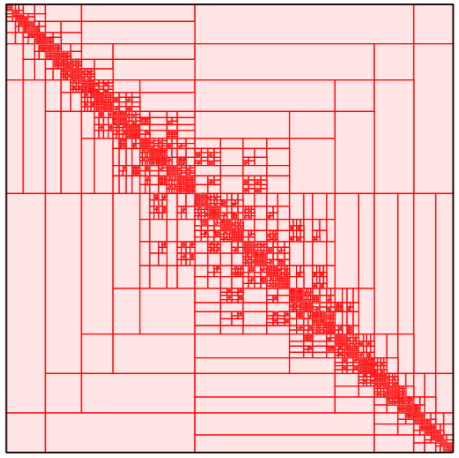


(格子) $\mathcal{H}$行列-ベクトル積の並列計算性能

- 以下のMPIプロセス数まで計算速度向上を観測
  - 従来$\mathcal{H}$行列：約100 MPI プロセスまで
  - 格子$\mathcal{H}$行列（全ベクトル保持）： 約1,600 MPIプロセスまで
  - 格子$\mathcal{H}$行列（ベクトル分散）：少なくとも 3,600 MPIプロセスまで
- MPIプロセス数N_p> 100では格子$\mathcal{H}$行列が従来$\mathcal{H}$行列より高速
- MPIプロセス数N_p> 600では格子$\mathcal{H}$行列（ベクトル分散）が最速



”Single film model”
N =20,910



1　$\mathcal{H}$ACApKライブラリの概要

2　低ランク構造行列法

3　(格子)$\mathcal{H}$行列法の並列化

4　$\mathcal{H}$ACApKライブラリの使用方法

5　今後の取り組み

$\mathcal{H}$ACApKで仕様を公開している関数

■関数一覧と機能

赤字：必ず使用される関数

関数名	機能、コメント
HACApK_init	環境設定；最初に呼ばれる必要あり
HACApK_generate	低ランク構造行列の生成
HACApK_solve	線形方程式の求解
HACApK_adot_pmt_lfmtx_p	行列・ベクトル積計算
HACApK_free_leafmtxp	低ランク構造行列のメモリ開放
HACApK_finalize	後始末

・境界要素法シミュレーションに最低限必要な機能を供給

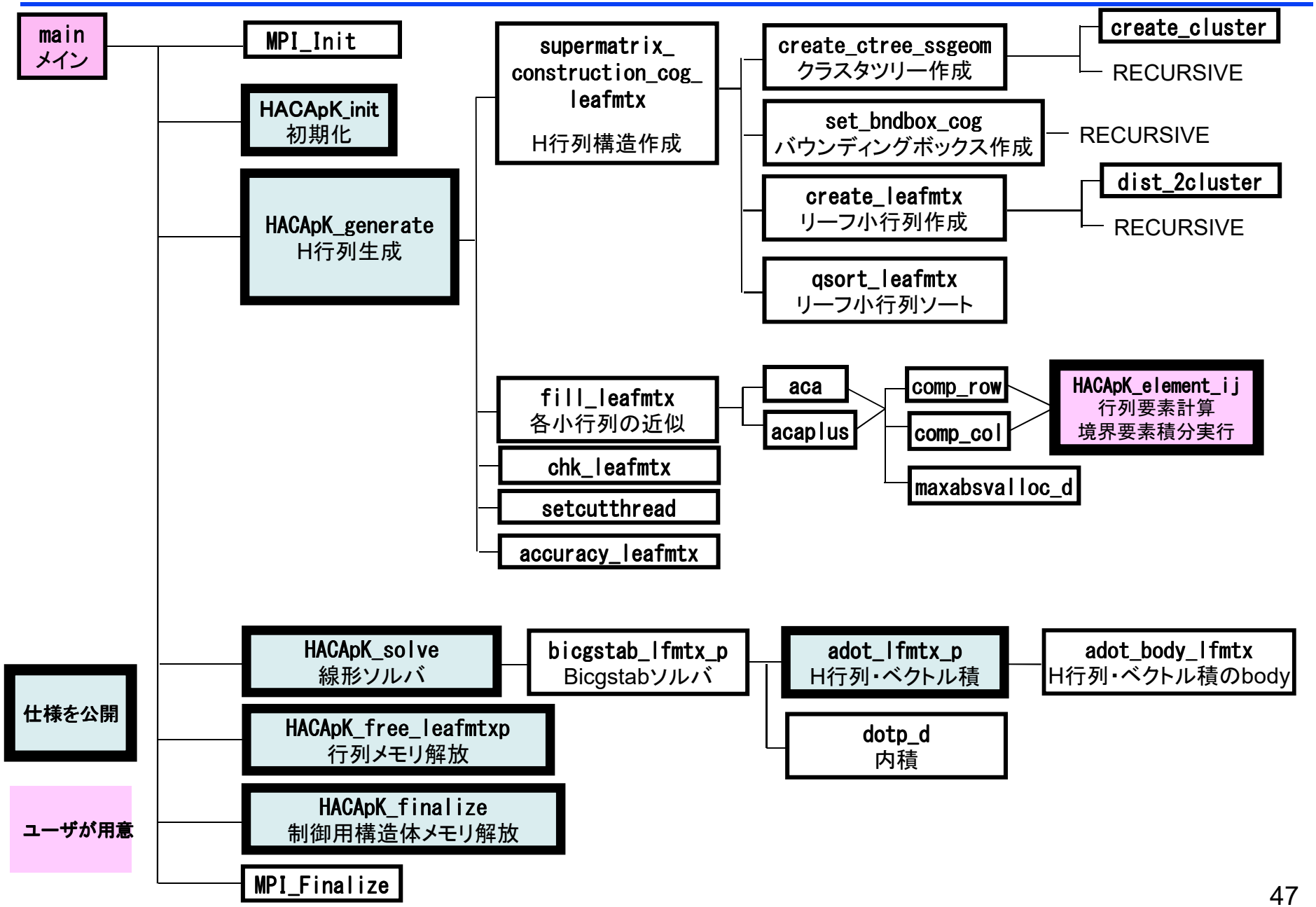
■静的な問題を解く場合に使用される関数：HACApK_solve

例：静電場解析

■動的な問題を解く場合に使用される関数：HACApK_adot_pmt_lfmtx_p

例：マイクロマグネティクス計算、超電導解析、地震サイクル解析、動的破壊解析

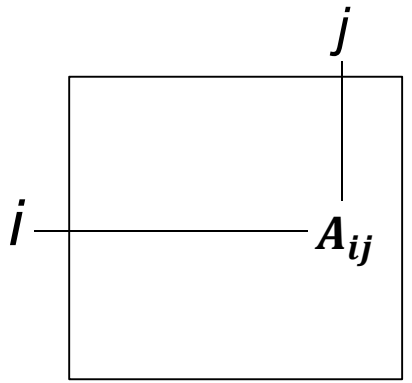
$\mathcal{H}$ACApK利用コードの構成例



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
```

ℋACApKの使用方法

・モジュールと構造体を宣言し、
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_use

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モジュール呼出

ℋACApK使用構造体宣言

ℋACApKイニシャライズ

配列のメモリ確保

ℋ行列の要求精度設定

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 ℋACApK.

$\mathcal{H}$ACApKへの入力情報

■行列サイズ：nd

```
lrtrn=HACApK_init(nd,st_ctl,st_bemv)
allocate(coord(nd,3),rhs(nd),sol(nd))
```

■$\mathcal{H}$行列への要求精度：ztol

```
lrtrn=HACApK_generate(st_leafmtxp,st_bemv,st_ctl,coord,ztol)
lrtrn=HACApK_solve(st_leafmtxp,st_bemv,st_ctl,rhs,sol,ztol)
```

■線形方程式の右辺ベクトル：rhs

```
lrtrn=HACApK_solve(st_leafmtxp,st_bemv,st_ctl,rhs,sol,ztol)
```

■行列ベクトル積の元ベクトル：x

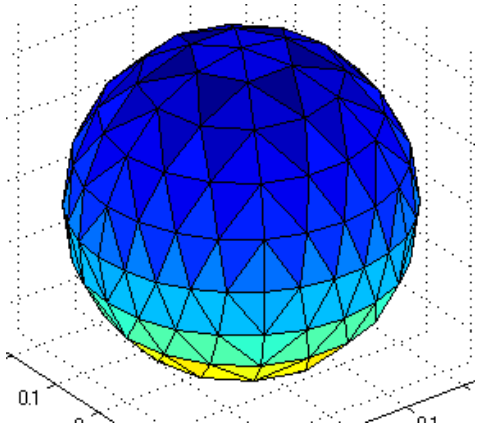
```
lrtrn=HACAkK_adot_pmt_lfmtx_p(st_leafmtxp,st_bemv,st_ctl,ax,x)
```

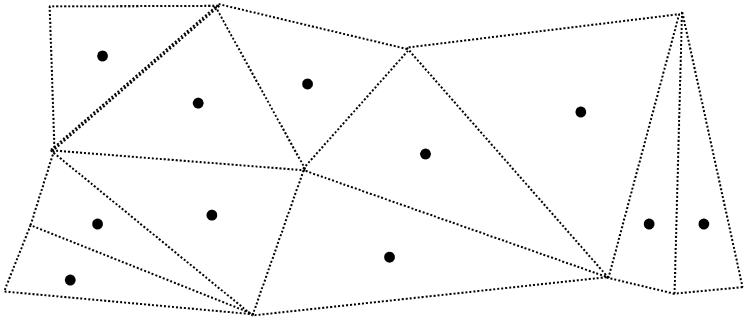
ℋACApKへの入力情報：座標情報coord(:,:)

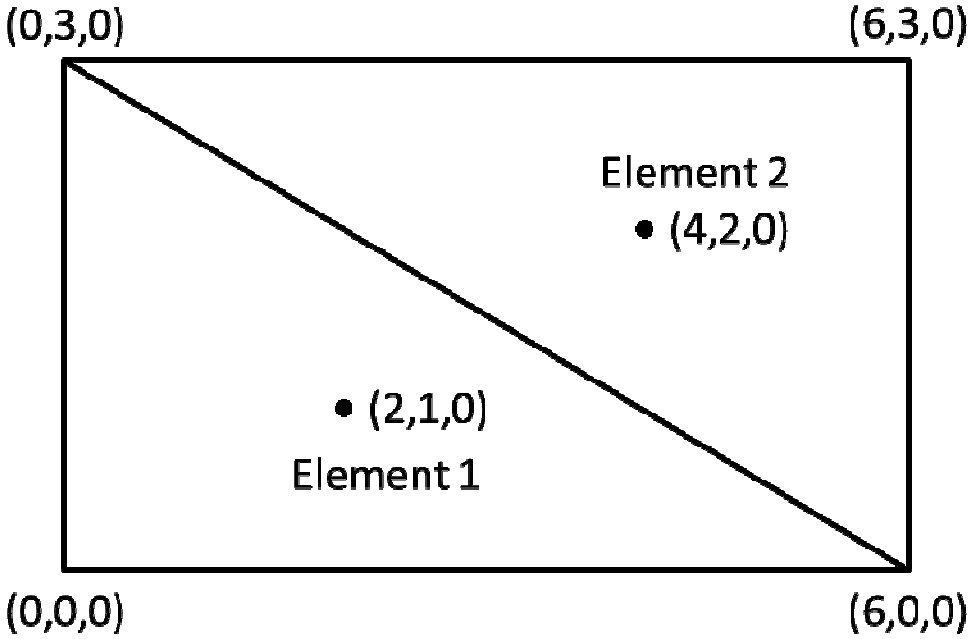
■クラスタリングに必要な座標情報

・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

・要素が関連付けられている場合には、要素重心の座標



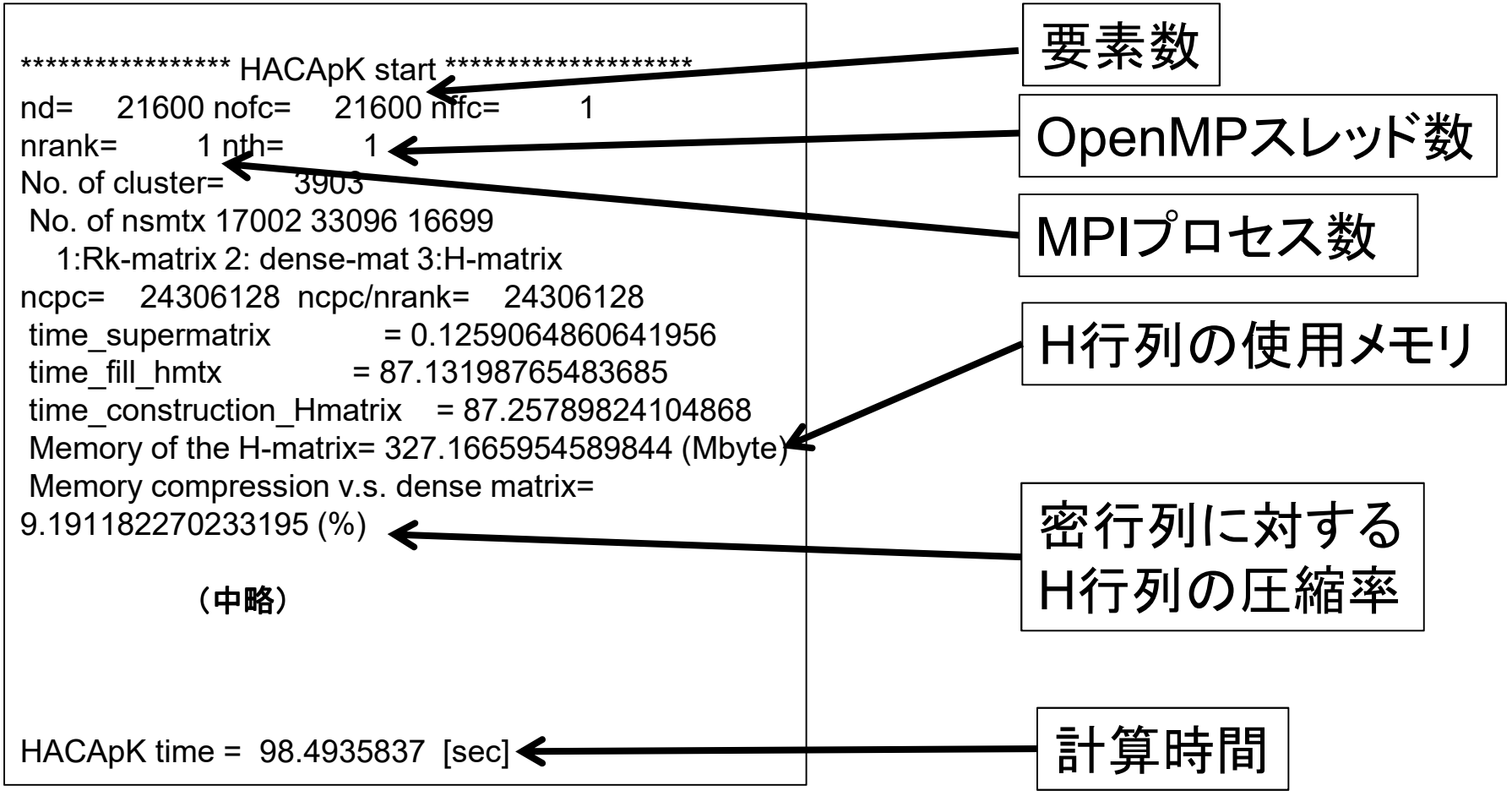





```
nd=2
coord(1,1)=2.0; coord(1,2)=1.0; coord(1,3)=0.0
coord(2,1)=4.0; coord(2,2)=2.0; coord(2,3)=0.0
```

HACApKのログ

■ 制御パラメタst_ctl%param(1)の値が1以上で出力



$\mathcal{H}$ACApKの一歩進んだ使い方

■制御パラメタ：st_ctl%param(1:100)

番号	初期値	説明
1	1	ログ出力（0:エラーのみ，1:標準出力，2:デバグ用出力）
10	1	低ランク構造行列の種類（1:通常$\mathcal{H}$行列，10:格子$\mathcal{H}$行列，20:BLR行列）
11	0	近似行列の精度チェック（0：チェックしない，　1：チェックする）
21	15	最小クラスタサイズ
51	2.0	クラスタ間の距離判定パラメータ
61	1	部分行列での低ランク近似手法選択（1:ACA法，2:ACA+法）
63	1000	部分行列の最大ランク

■制御パラメタの設定方法

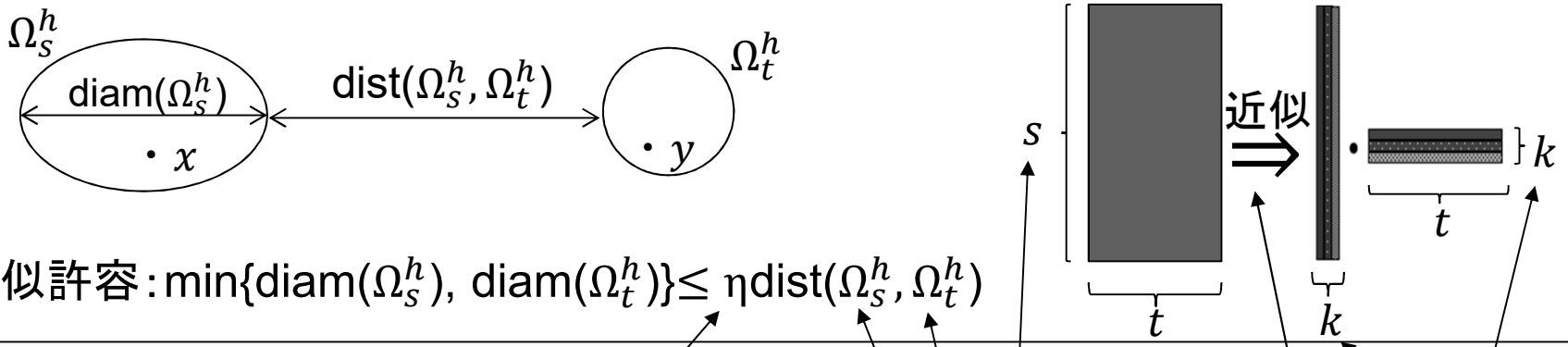
・初期化関数HACApK_initと近似行列生成関数HACApK_generateの間で設定

```
lrtrn=HACApK_init(nd,st_ctl,st_bemv)
allocate(coord(nd,3),rhs(nd),sol(nd))
ztol=1.0e-5; st_ctl%param(10)=10; st_ctl%param(11)=1;
lrtrn=HACApK_generate(st_leafmtxp,st_bemv,st_ctl,coord,ztol)
```

制御パラメタ：st_ctl%paramの解説

■低ランク構造行列法の近似原理

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)

クラスタ間の
距離判定
param(51)

最小クラスタサイズ
param(21)

手法選択
param(61)

最大値設定
param(62)

1　$\mathcal{H}$**ACApK**ライブラリの概要

2　低ランク構造行列法

3　(格子)$\mathcal{H}$行列法の並列化

4　$\mathcal{H}$**ACApK**ライブラリの使用方法

5　今後の取り組み

低ランク構造行列分解への取り組み

■取り組み中の行列分解

►LU分解　►QR分解

■想定する利点

►密行列計算より軽い

- メモリ使用量：$\boldsymbol{O(NlogN)}$ $< O(N^2)$
- 演算量：$\boldsymbol{O(\mathrm{N}\log^2 N)}$ $\ll O(N^3)$

►疎行列の不完全分解より高精度

►ランク数の調整により精度が制御可能？

■想定する使い道

►低精度：クリロフ部分空間法の前処理、固有値の近似値計算

►高精度：密行列による完全分解の代替

格子$\mathcal{H}$行列に基づく数値線形代数の構築

■データ科学計算および科学技術計算の大規模化・高速化

►$O(N^2)$以上の密行列演算を$O(N\log^p N)$以下に低減

►密行列線形代数ライブラリ（ BLAS, LAPACK, ScaLAPACK）の置き換え

■格子$\mathcal{H}$行列計算アルゴリズム群の開発:

►行列演算種類の拡充

・開発済み：格子H行列生成，行列-ベクトル積，LU 分解

・開発中：QR 分解(BLRでは開発済)

・開発予定：行列-行列積，固有値計算，特異値計算

►格子$\mathcal{H}$行列法の深化

・複素数化および長方行列への対応（現状は実数正方行列のみ対応可能）

・テンソルへの拡張

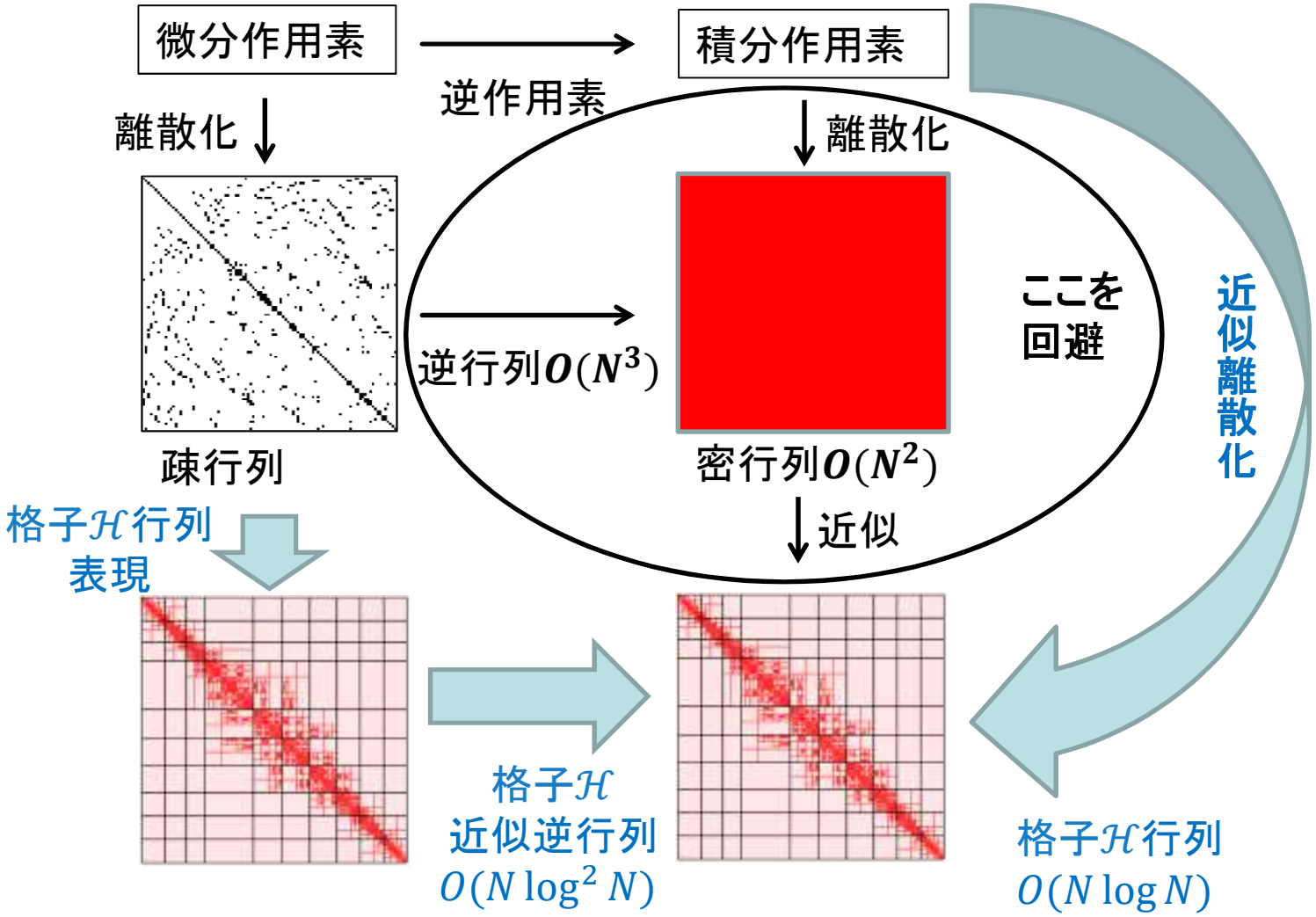
►データ科学分野の問題への適用

・最尤推測法に用いられる各種確率分布に基づく共分散行列への適用

・3次元ユークリッド空間と高次元データ空間の違いを克服

・ベイズ推論および深層学習へと適用範囲拡大

科学技術計算での密行列演算回避



階層的低ランク行列近似の適用範囲

① 線形積分作用素から得られる密行列の効率的な近似行列表現を得る
② 上記①で得られた階層型行列に対し，
行列計算（加法，乗法，LU分解，QR分解など）を行う
③ 線形微分作用素から得られる疎行列に対し，近似逆行列を計算する
④ 線形微分作用素から得られる疎行列に対し，近似LU分解を計算する

③の直感的理解(Bebendorf（2008）より） ・微分の逆作用素は積分のはず

$$-\Delta u = f \text{ in } \Omega := (a,b), u(a) = u(b) = 0$$

⇩ 2次精度中心差分で離散化

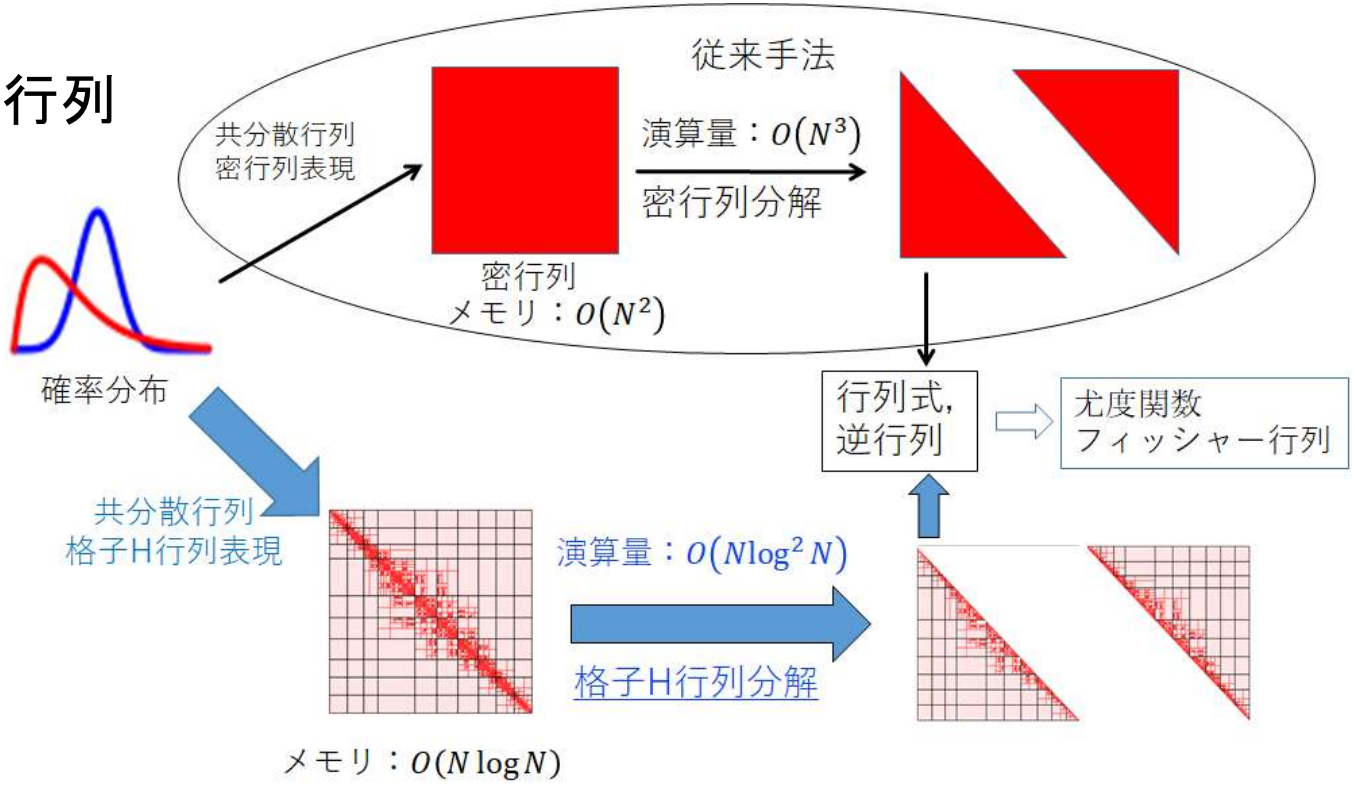
$$A = \begin{bmatrix} 2 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & -1 & 2 & -1 & & & \\ & & & -1 & 2 & -1 & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -1 & 2 & -1 \\ & & & & & & -1 & 2 \end{bmatrix}$$

線形積分作用素の離散化行列に類似 ⇒階層型行列

$$A^{-1} = \frac{1}{9}\begin{bmatrix} 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 7 & 14 & 12 & 10 & 8 & 6 & 4 & 2 \\ 6 & 12 & 18 & 15 & 12 & 9 & 6 & 3 \\ 5 & 10 & 15 & 20 & 16 & 12 & 8 & 4 \\ 4 & 8 & 12 & 16 & 20 & 15 & 10 & 5 \\ 3 & 6 & 9 & 12 & 15 & 18 & 12 & 6 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

データ科学計算への格子$\mathcal{H}$行列適用

■尤度関数，
フィッシャー情報行列



■Dimensionality Reduction

►主成分分析のホットスポット：共分散行列の固有値・固有ベクトル計算

・密行列　　：メモリ$\boldsymbol{O(N^2)}$，演算量$\boldsymbol{O(N^3)}$

・格子$\mathcal{H}$行列：メモリ$\boldsymbol{O(N\log N)}$，演算量$\boldsymbol{O(N\log^2 N)}$

■ディープニューラルネットワークのハイパーパラメータ予測

►ヘッセ行列とヤコビ行列の関係式から最適な学習率やバッチサイズを見積り

・McCandlishらの研究では対角近似 ⇒ 格子$\mathcal{H}$行列適用により予測精度向上

格子$\mathcal{H}$行列の最新アーキテクチャへの高性能実装法

■ハードウェアの性能を引き出せるアルゴリズム開発

■最新計算機アーキテクチャ向け格子H行列計算コード開発:

►次世代CPU 向けコード開発
- ・スパコン「京」の次世代機「富岳」で採用されるCPU「A64FX」
- ・メニーコア型CPU, ベクトル命令セット採用, 小容量高速メモリ搭載

►GPU向けコード開発
- ・マルチコアCPU 向けが既開発の演算（格子H 行列生成, 行列-ベクトル積, LU 分解）
- ・テンソルコアの効率的な利用法

►FPGA を用いた格子$\mathcal{H}$行列分割生成および行列分解
- ・格子$\mathcal{H}$ 行列分割構造の作成ルーチン（条件分岐が多く、CPU・GPU に不向き）
- ・格子$\mathcal{H}$行列分解（深い演算パイプラインと細粒度での並列処理が可能なFPGA向き）

►混合精度演算による計算の効率化
- ・部分行列のランク数に応じて数値の保存精度を決定
- ・誤差が蓄積する縮約演算の量に応じて演算精度を決定

►動的手法を用いた高速化（ヘテロジニアスなシステムへの対応）
- ・タスク並列言語(Cilk Plus, Tascell)を用いた動的負荷分散
- ・ランタイムライブラリ（StarPU, PaRSECなど）の活用

Acceleration of $\mathcal{H}$-mat. generation using KNL

[Hoshino : CLUSTER *2018*]

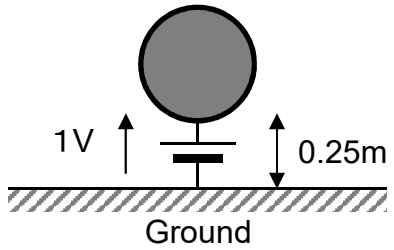
■ Test model of electrostatic field analysis

►Perfect conducting sphere $P[u](x) := \int_\Omega \frac{1}{4\pi\|x-y\|} u(y)\mathrm{d}y, x \in \Omega$

►Dielectric sphere $D[u](x) := \int_\Omega \frac{\langle x-y, n(y)\rangle}{4\pi\|x-y\|^3} u(y)\mathrm{d}y, x \in \Omega$

- including branch divergence

User-defined functions depend on these integral equations



■ Evaluation Environments

►BDW : Intel Xeon E5-2695 v4, 18 core

►KNL : Intel Xeon Phi 7250, 68 core

►Compiler : Intel compiler 18.0.1

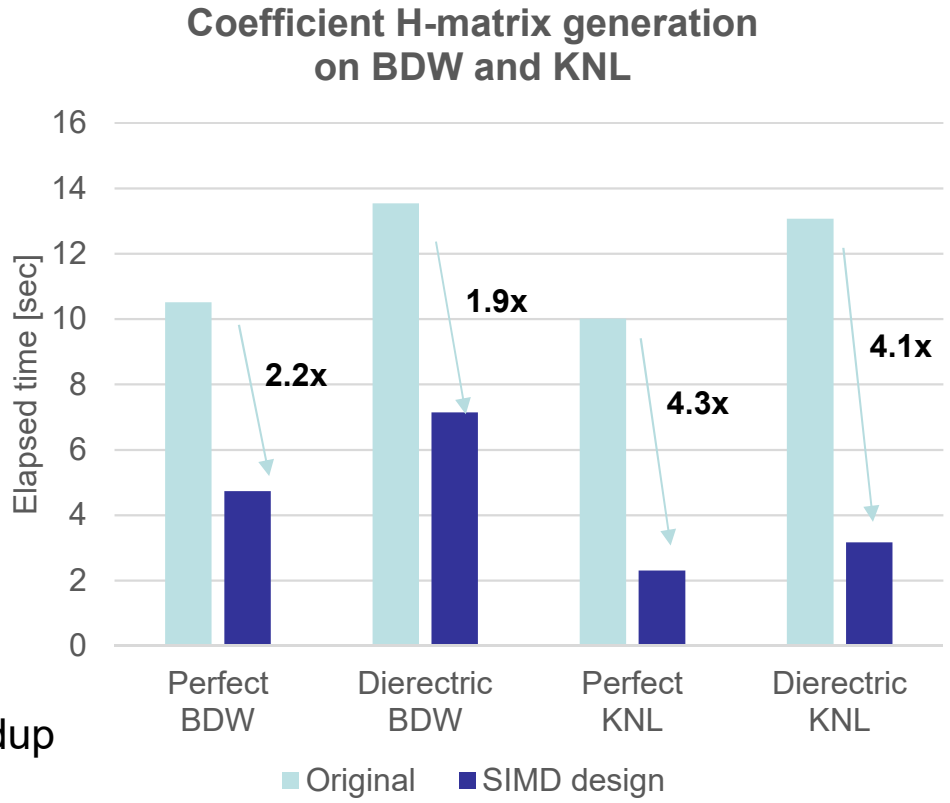
- -qopenmp -O3 -ipo -align array64byte -xAVX2 (BDW) –xMIC-AVX512 (KNL)

■ Performance comparison

►BDW : approximately **2x** speedup

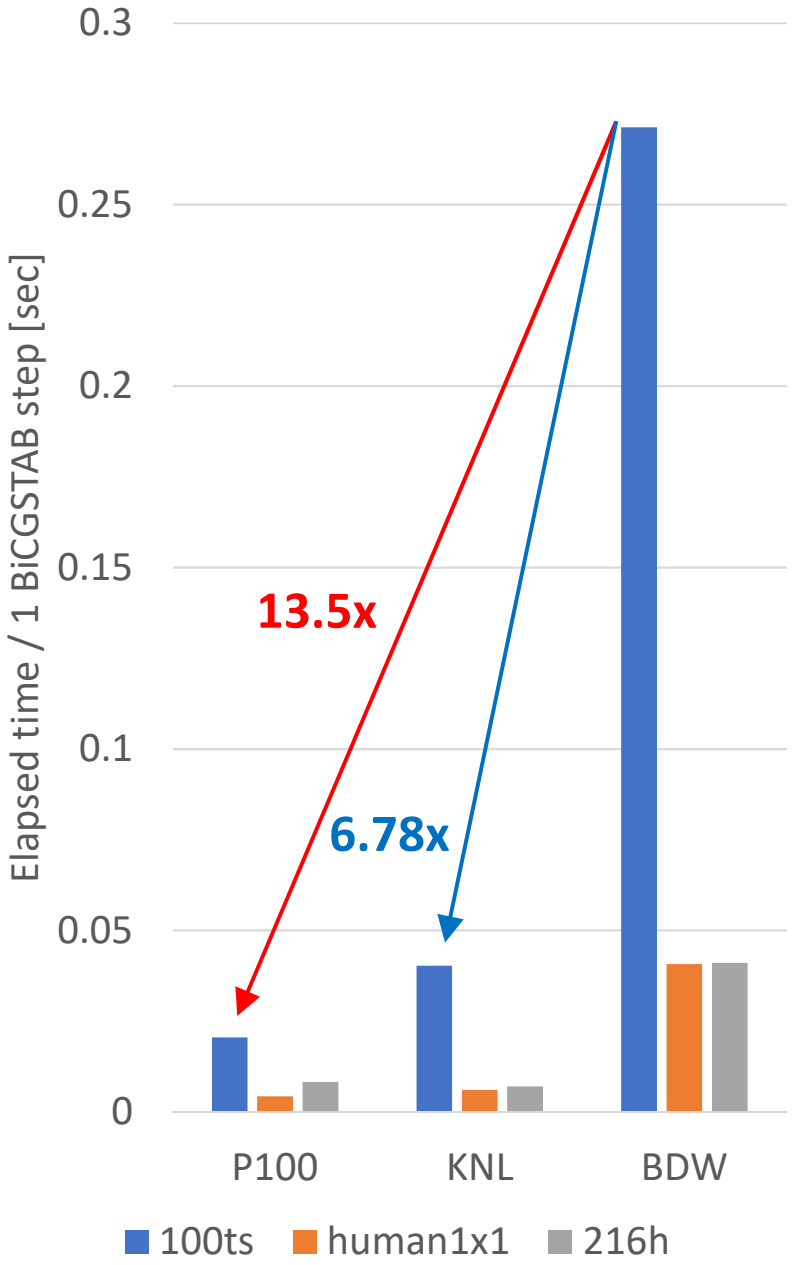
►KNL : over **4x** speedup

- In the case of dense matrix generation, new design achieved at most **6.6x** speedup



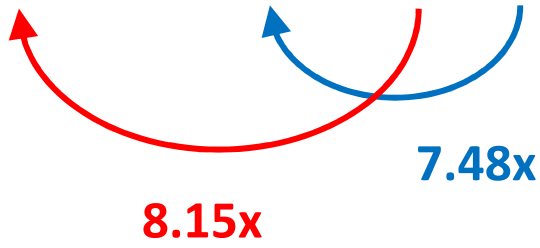
Acceleration of $\mathcal{H}$-mat.-vec. using GPU, KNL

[Hoshino : ICCS *2018*]



■Memory bandwidth bounded

略　称	P100	KNL	BDW
動作周波数（GHz）	1.480	1.40	2.10
コア数（最大有効スレッド数）	3,584	68 (272)	18 (18)
理論演算性能（GFLOPS）	5,304	3,046.4	604.8
主記憶容量（GB）	16	16	128
メモリバンド幅（GB/sec., Stream Triad）	**534**	**490**	**65.5**



HACApK関連の査読付き論文リスト

1. Rise Ooi, Takeshi Fukaya, Takeshi Iwashita, Akihiro Ida and Rio Yokota, “Accelerating Multithreaded Linear Solver with Mixed Precision Hierarchical Matrix Computation and Data Structure”, International Conference on High Performance Computing in Asia-Pacific Redion (HPC Asia 2020), pp.92-101.
2. Yu Pei, George Bosilca, Ichitaro Yamazaki, Akihiro Ida and Jack Dongarra, “Evaluation of Programming Models to Address Load Imbalance on Distributed Multi-Core CPUs: A Case Study with Block Low-Rank Factorization”, In: 2019 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM). IEEE, 2019. (In press)
3. Akihiro Ida, Tadashi Ataka and Atsushi Furuya, “Lattice H-matrices for Massively Parallel Micromagnetic Simulations of Current-induced Domain Wall Motion”, IEEE Transactions on Magnetics, (In press)
4. Zhengyang Bai, Tasuku Hiraishi, Hiroshi Nakashima, Akihiro Ida and Masahiro Yasugi. “Parallelization of Matrix Partitioning in Construction of Hierarchical Matrices using Task Parallel Languages” , Journal of Information Processing, Vol. 27 (2019) pp. 840-851.
5. Akihiro Ida, Hiroshi Nakashima, Tasuku Hiraishi, Ichitaro Yamazaki, Rio Yokota and Takeshi Iwashita, “QR Factorization of Block Low-Rank Matrices with Weak Admissibility Condition”, Journal of Information Processing, Vol. 27 (2019) pp. 831-839.
6. Satoshi Ohshima, Ichitaro Yamazaki, Akihiro Ida and Rio Yokota, “Optimization of Numerous Small Dense-Matrix–Vector Multiplications in H-matrix Arithmetic on GPU”, In: Auto-Tuning for Multicore and GPU (ATMG) In conjunction with the IEEE MCSoC-19, Singapore (Oct 1-4, 2019)
7. Ichitaro Yamazaki, Akihiro Ida, Rio Yokota and Jack Dongarra, “Distributed Memory Lattice H-matrix Factorization”, The International Journal of High Performance Computing Applications (IHPCA) (2019) 1094342019861139.
8. Takeshi Mifune, Naoki Tominaga, Yusuke Sogabe, Yudai Mizobata, Masahiro Yasunaga, Akihiro Ida, Takeshi Iwashita and Naoyuki Amemiya, “Large-scale electromagnetic field analyses of coils wound with coated conductors using a current-vector-potential formula-tion with a thin-strip approximation”, Superconductor Science and Technology(SUST) .
9. Tetsuya Hoshino, Akihiro Ida, Toshihiro Hanawa and Kengo Nakajima,“Load-balancing-aware Parallel Algorithms of H-matrices with Adaptive Cross Approximation for GPUs”, In Proceedings of IEEE International Conference on Cluster Computing (CLUSTER 2018)

HACApK関連の査読付き論文リスト

10. Tetsuya Hoshino, Akihiro Ida, Toshihiro Hanawa and Kengo Nakajima,“Design of Paral-lel BEM Analyses Framework for SIMD Processors”, In: Shi Y. et al. (eds) Computational Science ? ICCS 2018. ICCS 2018. Lecture Notes in Computer Science, vol 10860. Spring-er, Cham
11. Akihiro Ida, “Lattice H-Matrices on Distributed-Memory Systems”, 32nd IEEE Interna-tional Parallel & Distributed Processing Symposium (IPDPS 2018), Vancouver, Canada (May 21-25, 2018), pp.389-898.
12. Ichitaro Yamazaki, Ahmad Abdelfattah, Akihiro Ida, Satoshi Ohshima, Stanimire Tomov, Rio Yokota, and Jack Dongarra “Analyzing Performance of BiCGStab with Hierarchical Matrix on GPU clusters”, 32nd IEEE International Parallel & Distributed Processing Sym-posium (IPDPS 2018), Vancouver, Canada (May 21-25, 2018) ,pp930-939.
13. Naoki Tominaga, Takeshi Mifune, Akihiro Ida, Yusuke Sogabe, Takeshi Iwashita, Naoyuki Amemiya, “Application of hierarchical matrices to large-scale electromagnetic field analyses of coils wound with coated conductors”, IEEE Transactions on Applied Su-perconductivity, Volume 28, Issue 3 (April 2018), pp.1-5.
14. Satoshi Ohshima, Ichitaro Yamazaki, Akihiro Ida and Rio Yokota, “Optimization of Hierarchical matrix computation on GPU”, In: Asian Conference on Supercomputing Frontiers. Springer, Cham, 2018. p. 274-292.
15. Akihiro Ida, Hiroshi Nakashima and Masatoshi Kawai, “Parallel Hierarchical Matrices with Block Low-rank Representation on Distributed Memory Computer Systems”, International Conference on High Performance Computing in Asia-Pacific Redion (HPC Asia 2018), Tokyo, Japan (Jan. 28-31, 2018), pp.232-240.
16. Akihiro Ida, Tadashi Ataka, Takeshi Mifune, Yasuhito Takahashi, Takeshi Iwashita and Tadashi Furuya, “Application of Improved H-matrices in Micromagnetic Simulations of Spin Torque Oscillator”, IEEE Transactions on Magnetics, Vol. 54, Issue 3
17. Takeshi Iwashita, Akihiro Ida, Takeshi Mifune and Yasuhito Takahashi, "Software Framework for Parallel BEM Analyses with H-matrices Using MPI and OpenMP", Procedia Computer Science 108C (2017) pp.2200–2209.
18. Akihiro Ida, Takeshi Iwashita, Takeshi Mifune and Yasuhito Takahashi, “Variable Preconditioning of Krylov Subspace Methods for Hierarchical Matrices with Adaptive Cross Approximation,” IEEE Transactions on Magnetics, Vol. 52, Issue 3

HACApK関連の査読付き論文リスト

19. Takeshi Iwashita, Naokazu Takemura, Akihiro Ida and Hiroshi Nakashima, “A new fill-in strategy for IC factorization preconditioning considering SIMD instructions”, The 13-th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA-15), Helsinki Finland (2015)
20. Akihiro Ida, Takeshi Iwashita, Makiko Ohtani and Kazuro Hirahara, “Improvement of Hierarchical Matrices with Adaptive Cross Approximation for Large-scale Simulation”, Journal of Information Processing, Vol. 23, No. 3 (2015) pp. 366-372.
21. Akihiro Ida, Takeshi Iwashita, Takeshi Mifune and Yasuhito Takahashi, “Parallel Hierarchical Matrices with Adaptive Cross Approximation on Symmetric Multiprocessing Clusters”, Journal of Information Processing, Vol. 22, No. 4 (2014) pp. 642-650.