RIKEN
Center for
Computational Science

2024 年 6 月 27 日

有限要素解析のための並列ソルバーに関するスクール 2024

Krylov 部分空間法のための 加法的 Schwarz 前処理

鈴木 厚 [1]

[1] 理研計算科学研究センター 大規模並列数値計算技術研究チーム
atsushi.suzuki.aj@a.riken.jp

Krylov 部分空間法のための前処理

前処理 $Q \simeq A^{-1}$, 残差ベクトル $\vec{r}$ に作用させる：
経済的 (軽い) あるいは 演算集約的 (重い)
右前処理の方程式

$$AQ(Q^{-1}\vec{x}) = \vec{b} \Leftrightarrow A\vec{x} = \vec{b} \text{ 初期残差 } \vec{r}_0 = \vec{b} - A\vec{x}_0$$

前処理付き GMRES

次を満す$\vec{x}'_m \in Q^{-1}x_0 + K_m(AQ, \vec{r}_0)$ を見付けよ

$$||\vec{b} - AQ\vec{x}'_m|| \leq ||\vec{b} - AQ\vec{y}|| \quad \forall \vec{y} \in Q^{-1}x_0 + K_m(AQ, \vec{r}_0)$$

- 対角前処理 $[Q]_{ii} = |[A]_{ii}|^{-1}$
- SOR 前処理 $A = L + D + U$ パラメータ $0 < \omega < 2$,
  $Q = (\frac{D}{\omega} + U)^{-1} \frac{2-\omega}{\omega} D (\frac{D}{\omega} + L)^{-1}$
- 不完全分解, ILU 前処理
- multigrid 前処理 / 代数的 multigrid : $\Rightarrow$ HYPRE, gamg in PETSc
- 重なりのある領域分割を用いた部分問題での直接法による加法的 Schwarz 前処理GenEO + 2 段階法 $\Rightarrow$ HPDDM

不完全 LU 分解による前処理

非零パターン $NZ(A) := \{(i,j)\,;\, a_{ij} \neq 0\}$

アルゴリズム (ILU(0))

do i=2,$\cdots$, N
  do k=1$\cdots$, $i-1$; $(i,k) \in NZ(A)$
    $a_{ik} = a_{ik}/a_{kk}$
    do j=$k+1, \cdots, N$; $(i,j) \in NZ(A)$
      $a_{ij} = a_{ij} - a_{ik}/a_{kj}$

intel Math Kernel Library (MKL) のサブルーチン

```
int nrow, ierr;
double *coefs, *ilu;   // non-zero values
int *ia, *ja;          // CSR non-zero indexes
int ipar[128];         // ipar[30]=1 to conitnue for 0 diagonal
double dpar[128];      // dpar[30]=1.0e-16, dpar[31]=1.0e-10

dcsrilu0(&nrow, coefs, ia, ja, ilu, ipar, dpar, &ierr);
```

Schwarz による前処理

添字による行列の重なりのある分解 $\Lambda = \bigcup_{p=1}^{P} \Lambda_p,\ \Lambda_p \cap \Lambda_q \neq \emptyset$

- R_p: 全自由度から部分行列への制限：$\Lambda \to \Lambda_p$
- D_p：離散的な単位の分解

$$\sum_{p=1}^{P} R_p^T D_p R_p = I_N,$$

$$[D_p]_{kk} = \begin{cases} 1 & k \in \Lambda_p,\ k \notin \Lambda_q, \forall q \neq p, \\ 1/\#\{p; k \in \Lambda_p\} & \text{それ以外} \end{cases}$$

ASM 前処理

$$M_{\mathrm{ASM}}^{-1} = \sum_{p=1}^{P} R_p^T (R_p A R_p^T)^{-1} R_p$$

ASM は不動点反復法としては収束しない, しかしながら, M_{ASM}^{-1} は対称で CG 法の前処理として機能する.

RAS preconditioner

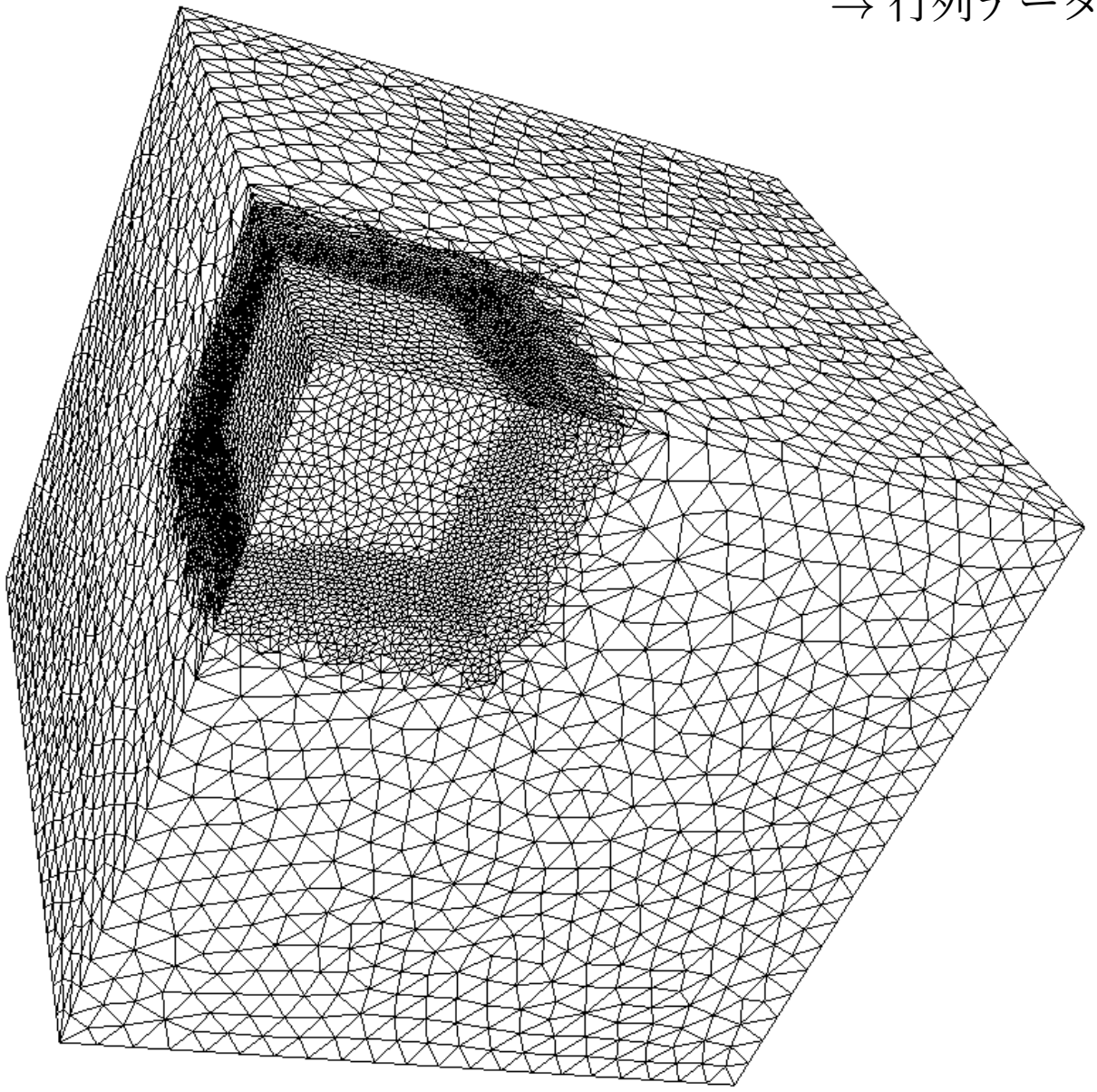
$$M_{\mathrm{RAS}}^{-1} = \sum_{p=1}^{P} R_p^T D_p (R_p A R_p^T)^{-1} R_p$$

RAS は収束するが M_{RAS}^{-1} は非対称のため GMRES 法の前処理として機能する.

数値例：1/3

`tetgen` と `FreeFEM` によって生成された非構造メッシュP2 有限要素,
$N = 677,163$, $nnz = 28,853,844$

→ 行列データ `Navier3DmeshP2`



数値例：2/3

停止条件：$\leq 10^{-12}$ を相対誤差
Navier 方程式：P2 有限要素により離散化された弾性体問題

$N = 750,141, nnz = 31,214,610$：立方体の構造メッシュ ($32^3$ 節点)

`Navier3D.32.P2`

		経過時間 (秒)			
重なり幅	反復回数	全体	LU-分解	反復計算	誤差
1	59	81.397	44.749	36.647	4.41361e-12
2	41	87.871	55.339	32.532	1.70397e-12
3	33	106.13	72.975	33.158	1.45462e-12
ILU(0)	186	58.347	16.146	42.201	4.66513e-11

$N = 677,163, nnz = 28,853,844$：要素細分による非構造メッシュ

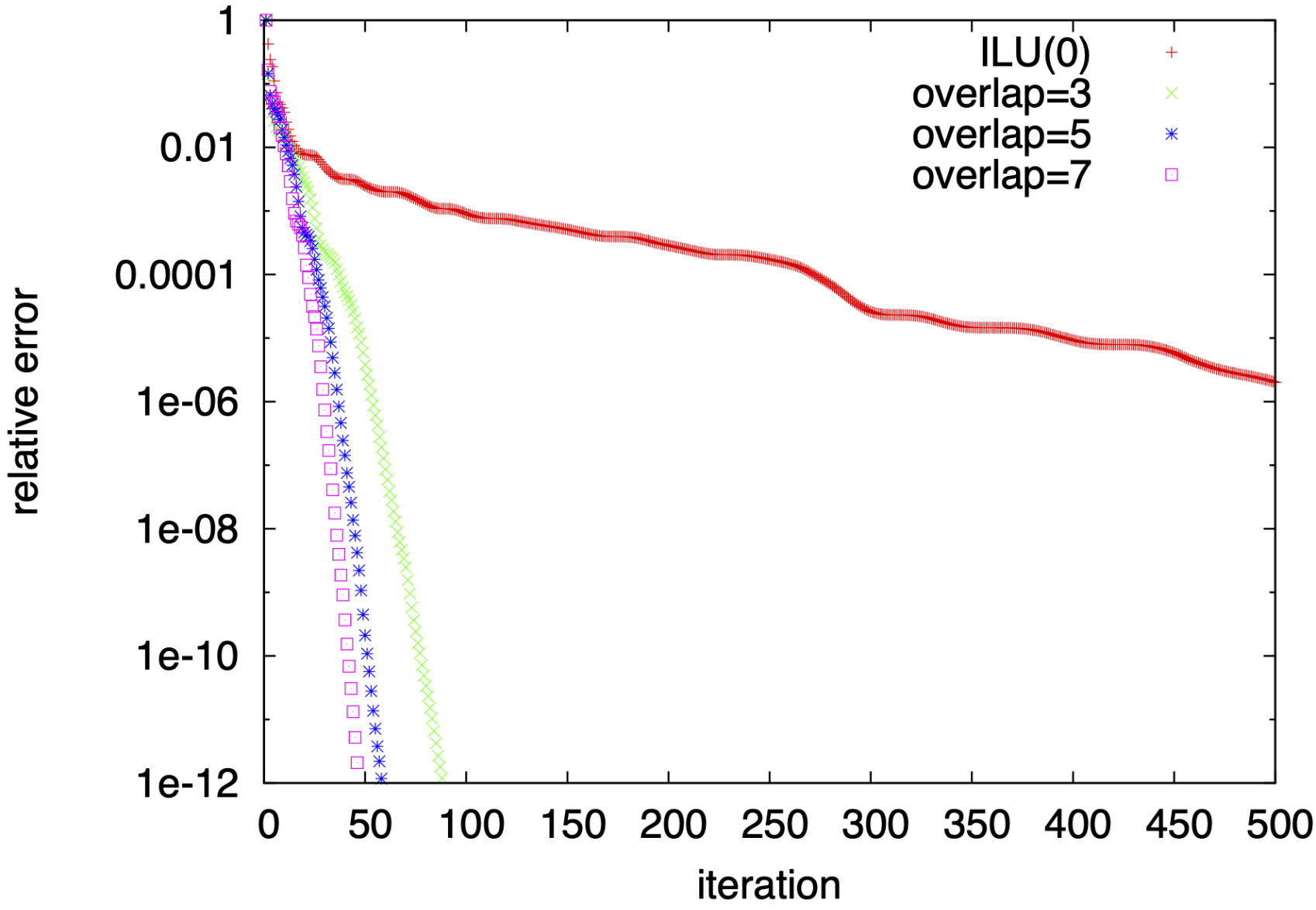
`Navier3DmeshP2`

		経過時間 (秒)			
重なり幅	反復回数	全体	LU-分解	反復計算	誤差
1	88	77.978	29.413	48.565	4.23959e-12
2	58	87.975	45.055	42.920	4.62264e-12
3	46	111.68	69.063	47.112	8.12039e-12
ILU(0)	500	209.91	30.791	179.12	1.12751e-2

ILU(0) 前処理は非構造メッシュの問題に対しては十分ではない

数値例：3/3

収束履歴


1
0.01
0.0001
1e-06
1e-08
1e-10
1e-12
relative error
0
50
100
150
200
250
300
350
400
450
500
iteration
ILU(0)
overlap=3
overlap=5
overlap=7

直接法の概要

係数行列 A は $\Pi_L^T L\, D\, U \Pi_R$ と分解される

- Π_L, Π_R：全軸選択を実現する左/右置換行列
- 対称軸選択 $\Pi_L = \Pi_R = \Pi$ に ε-擾乱を追加するか 2×2 軸選択を更に加えるか, あるいはそれのみ追加するか

置換行列 Π は次の二つから成る

- シンボリック分解によりフィルインを最小化する添字並べ替え
- 分解途中に対角成分の絶対値最大を見付けて動的に行と列を並べ替え

直接法の 3 つのフェーズ

- シンボリック分解
  フィルインを最小化し並列化を実現するためのマルチフロンタル法, 置換行列は Π_S
- 数値分解
  置換行列を Π_N とする動的な軸選択
- 前進消去/後退代入

$$\Pi^T L\, D\, U \Pi \vec{x} = \vec{b}$$

$$\vec{z} = L^{-1} \Pi \vec{b} \qquad \text{前進消去 } L\vec{z} = \Pi\vec{b}$$

$$\vec{y} = D^{-1} \vec{z}$$

$$\vec{x} = \Pi^T U^{-1} \vec{y} \qquad \text{後退代入 } U\Pi\vec{x} = \vec{y}$$

疎行列直接法ソフトウェアパッケージ

ソフトウェア	並列環境	消去方法	データ管理	軸選択	核判定
UMFPACK	—-	multi-frontal	static	yes	no
SuperLU_MT	shared	super-nodal	dynamic	yes	no
Pardiso	shared/dist.	super-nodal	dynamic	yes + $\sqrt{\varepsilon}$-p.	no
SuperLU_DIST	distributed	super-nodal	static	no, $\sqrt{\varepsilon}$-p.	no
MUMPS	dist./shared	multi-frontal	dynamic	yes	yes
Dissection	shared	multi-frontal	static	yes	yes

T. A. Davis, I. S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices,
ACM Trans. Math. Software, 25 (1999), 1–20.

J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, J. W. H. Liu.
A supernodal approach to sparse partial pivoting,
SIAM J. Matrix Anal. Appl., 20 (1999), 720–755.

O. Schenk, K. Gärtner. Solving unsymmetric sparse systems of liner equations with PARDISO,
Future Generation of Computer Systems, 20 (2004), 475–487.

X. S. Li, J. W. Demmel. SuperLU_DIST : A scalable distributed-memory sparse direct solver for unsymmetric linear systems,
ACM Trans. Math. Software, 29 (2003), 110–140.

P. R. Amestoy, I. S. Duff, J.-Y. L'Execellent. Mutlifrontal parallel distributed symmetric and unsymmetric solvers,
Comput. Methods Appl. Mech. and Engrg, 184 (2000) 501–520.

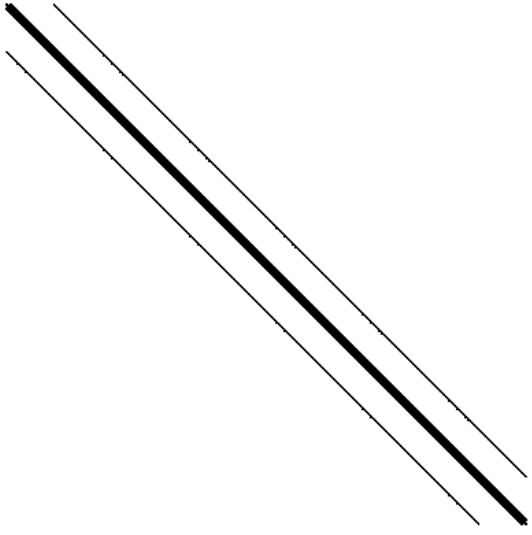
A. Suzuki, F.-X. Roux, A dissection solver with kernel detection for symmetric finite element matrices on shared memory computers,
Int. J. Numer. Meth. in Engng, 100 (2014) 136–164.

疎行列の変数添字の並べ替え

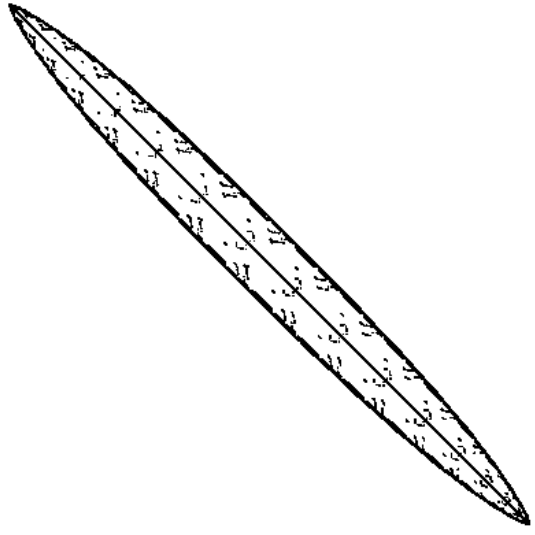
疎行列の変数添字は並べ替える必要がある

- フィルインを最小化するため
- 分解の並列度を高めるため → マルチフロント
- ブロック構造のサイズの粒度を大きくするため → スーパーノード

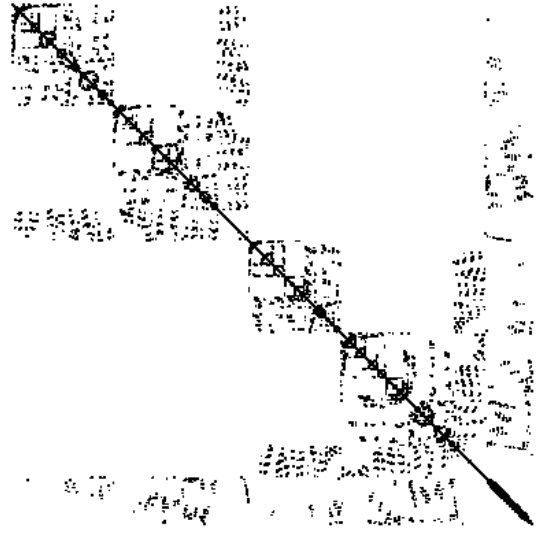
例:
3 次元 Poisson 方程式の 7 重対角行列, 合計 11^3 節点



もとの行列
帯行列

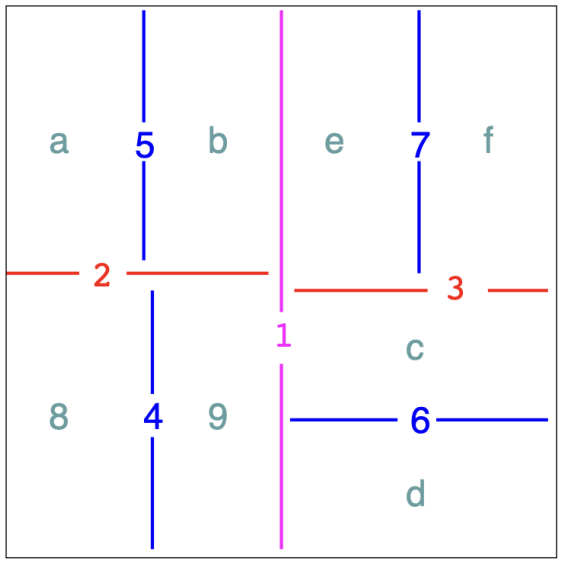


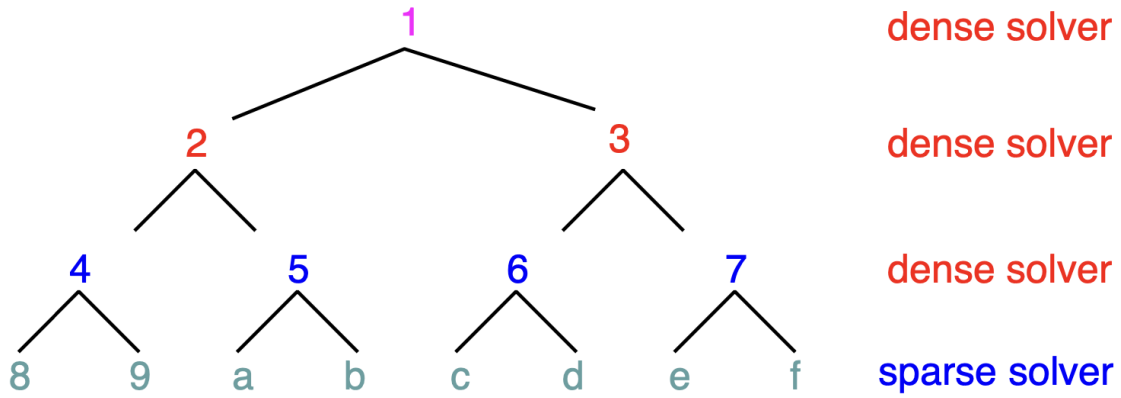
reverse Cuthill-McKee
逐次計算



nested-dissection
並列計算

グラフ分割による nested dissection 法





A. George. Numerical experiments using dissection methods to solve n by n grid problems. SIAM J. Num. Anal. 14 (1977),161–179.

software package:

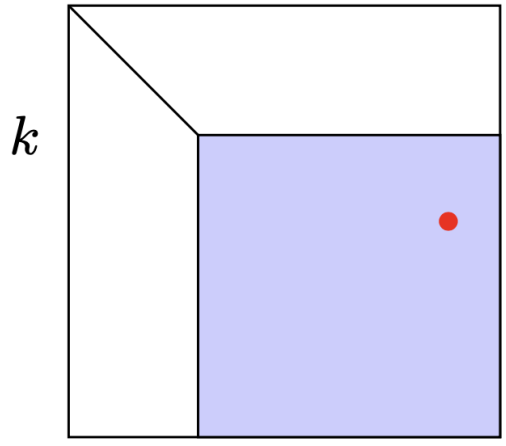
METIS : V. Kumar, G. Karypis, A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. 20 (1998) 359–392.

SCOTCH : F. Pellegrini J. Roman J, P. Amestoy, Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. Concurrency: Pract. Exper. 12 (2000) 69–84.

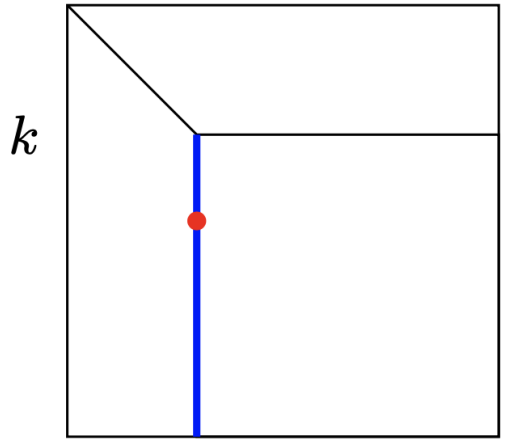
- 二分木の各々のノードは並列に計算できる ⇐ マルチフロント

軸選択

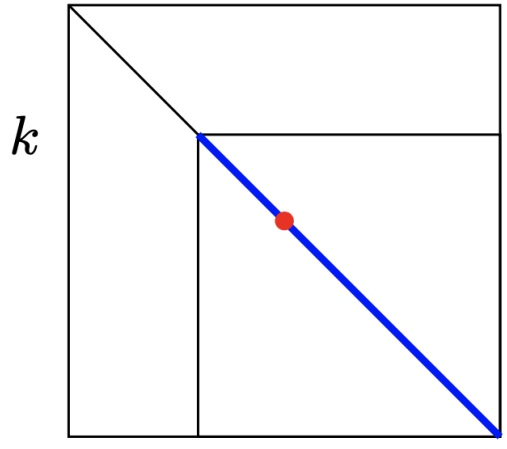
full pivoting : $A = \Pi_L^T LU \Pi_R$
find max$_{k<i,j\le n}|A(i,j)|$



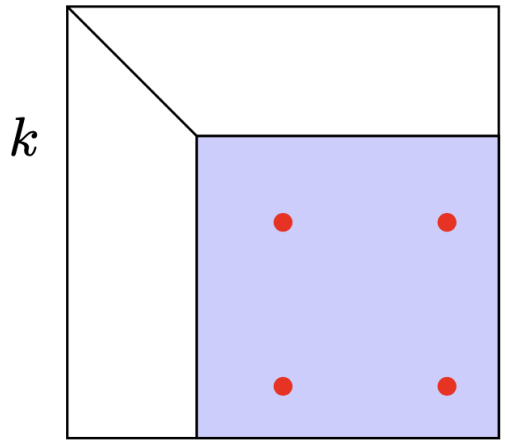
partial pivoting : $A = \Pi LU$
find max$_{k<i\le n}|A(i,k)|$



symmetric pivoting : $A = \Pi^T LDU\Pi$
find max$_{k<i\le n}|A(k,k)|$



2×2 pivoting : $A = \Pi^T L \tilde{D} U \Pi$
find max$_{k<i,j\le n}$det$\begin{vmatrix} A(i,i) & A(i,j) \\ A(j,i) & A(j,j) \end{vmatrix}$



sym. pivoting is mathematically not always possible

rank-1 更新による LDU 分解

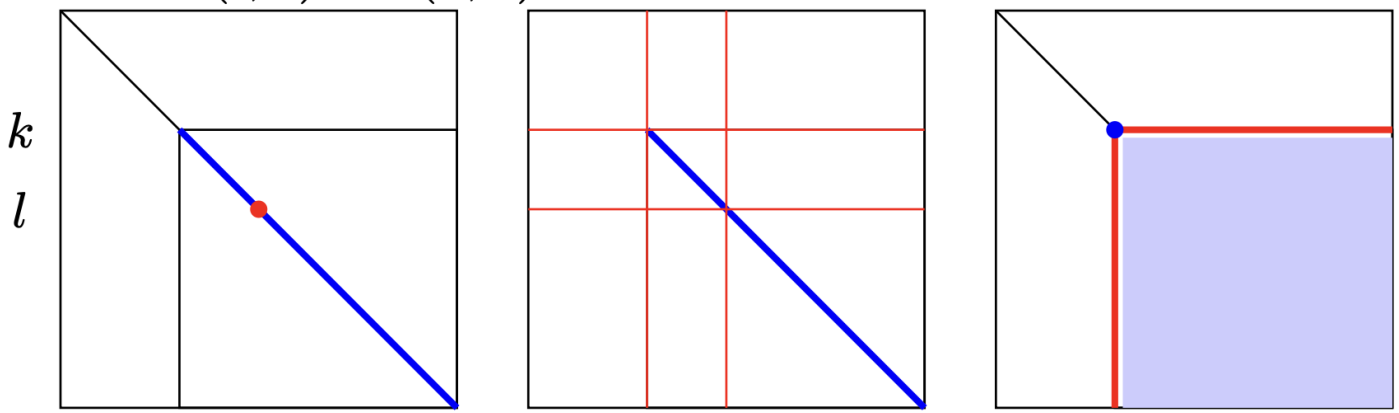
$$\begin{bmatrix} a_{11} & \beta_1^T \\ \alpha_1 & A_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \alpha_1 a_{11}^{-1} & I_2 \end{bmatrix} \begin{bmatrix} a_{11} & \beta_1^T \\ 0 & S_{22} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ \alpha_1 a_{11}^{-1} & I_2 \end{bmatrix} \begin{bmatrix} a_{11} & 0 \\ 0 & S_{22} \end{bmatrix} \begin{bmatrix} 1 & a_{11}^{-1}\beta_1^T \\ 0 & I_2 \end{bmatrix}$$

Schur 補行列 $S_{22} = A_{22} - \alpha_1 a_{11}^{-1}\beta_1^T$ は rank-1 更新 によって計算でき, BLAS ライブラリーの `dger` を用いる

対称軸選択による LDU-分解アルゴリズム

do $k = 1, \cdots, N$
  find max$|A(l,l)|$ with $k < l \leq n$,
  exchange rows and columns : $A(k,*) \leftrightarrow A(l,*)$, $A(*,k) \leftrightarrow A(*,l)$.
  `dscal` $A(k,j)/{=}A(k,k) \quad k < j \leq N$,
  `dger` $A(i,j){-}{=}A(i,k)A(k,j) \quad k < i,j \leq N$,
  `dscal` $A(i,k)/{=}A(k,k) \quad k < i \leq N$.



不定値行列向けの 2×2 軸選択

対称行列

$$\begin{bmatrix} \frac{1}{4} & \frac{5}{4} & \frac{1}{2} \\ \frac{5}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 5 & 1 & \\ 2 & \frac{1}{3} & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{4} & & \\ & -6 & \\ & & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 1 & 5 & 2 \\ & 1 & \frac{1}{3} \\ & & 1 \end{bmatrix}$$

対称軸選択を作用すると

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{5}{4} \\ \frac{1}{2} & \frac{5}{4} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ \frac{1}{2} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 0 & 1 \\ & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \\ & & 1 \end{bmatrix}$$

対角の絶対値最大の要素を見付ける軸選択は不定値行列では失敗することがある.
解決方法として 2×2 軸選択を用いる.
対称な不定値行列に対して 1×1 と 2×2 軸選択を組み合せる手法が提案されている.

J. R. Bunch, L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems, *Math. Comput*, 31 (1977) 163–179.

$\sqrt{\varepsilon}$-擾乱

正則化手法の一つ

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & \begin{matrix} 0 & \alpha \\ \beta & 0 \end{matrix} \end{bmatrix} \rightarrow \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & \begin{matrix} \sqrt{\varepsilon} & \alpha \\ \beta & 0 \end{matrix} \end{bmatrix}$$

- 方程式を変形してしまうので異なる問題を解いていることになるため反復改良手法により解の精度を向上させる
- ユーザーは非対称行列に対しては適切なパラメータの大きさを選択できる/しなければならない `Pardiso` のデフォルト値は 10^{-13}

C/C++ からの Paridso の利用

```
MKL_INT *ptrow = new MKL_INT[n + 1]; // CSR data
MKL_INT *indcol = new MKL_INT[nnz];
double *coef = new double[nnz];
double *x = new double[n]; // solution
double *y = new double[n]; // RHS
void *pt[64];          // to keep internal pointers
for (int i = 0; i < 64; i++)
   pt[i] = (void *)0; // zero clear
MKL_INT *iparm = new MKL_INT[64]; // parameters!
MKL_INT mtype = 11;  // structurally symmetric
MKL_INT nrhs = 1;
MKL_INT phase;
MKL_INT maxfct = 1, mnum = 1, msglvl = 1, error;
MKL_INT idum;          // dummy pointer instaed of user
                       // providing permutation
phase = 11;  // symbolic factorization
pardiso(pt, &maxfct, &mnum, &mtype, &phase, &n,
        (void *)coef, ptrow, indcol, &idum, &nrhs,
        iparm, &msglvl, (void *)y, (void *)x,
        &error);
phase = 22;  // numeric factorization
phase = 33;  // Fw/Bw substitution
phase = -1;  // free working data
```

C/C++ からの MUMPS の利用

```
MUMPS_INT *irow = new MUMPS_INT[nnz];
MUMPS_INT *jcol  = new MUMPS_INT[nnz];
double *coef = new double[nnz];
double *x = new double[n]; // solution&RHS
DMUMPS_STRUC_C id;
id.job = (-1); // job init
id.par = 1;
id.sym = isSym ? 2 : 0;
id.comm_fortran = USE_COMM_WORLD; // dummy MPI communicator
dmumps_c(&id);
id.job = 1; // symbolic facotrization
id.n = n; id.nz = nnz; id.irn = irow; id.jcn = jcol;
// id.icntl[]  set parameters
dmumps_c(&id);
id.job = 2; // numeric factorization
id.a = coef;
dmumps_c(&id);
id.job = 3;  // Fw/Bw substitution
id.nrhs = 1;
id.rhs = x;
dmumps_c(&id);
id.job = -2;  // free working data
```