RIKEN
Center for
Computational Science

2023 年 9 月 7 日

RIKEN HPC サマースクール 2023 - Society 5.0 に向けて -

FreeFEM 入門 - 非線形弾性体ソルバーを例に

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

疎行列を用いて Posisson 方程式を解く FreeFEM スクリプト

双一次形式 $a(u,v)=\int_\Omega \nabla u\cdot\nabla v\,dx$, L^2 と内積 $(f,v)=\int_\Omega fv$

次を満たす $u_h\in V_h(g)$ を見付けよ $a(u_h,v_h)=(f,v_h)\ \forall v_h\in V_h$.

```
mesh Th=square(20,20);  // (0,1)x(0,1) square domain
fespace Vh(Th,P1);
Vh u,v;
func f = cos(x)*sin(y);
func g = 1.0;
varf poisson(u,v)=int2d(Th)( dx(u)*dx(v)+dy(u)*dy(v) )
                    + on(1,2,3,4,u=g);
varf external(u,v)=int2d(Th)( f*v );
real tgv=1.0e+40; // penalty parameter > (machine eps)^-2
matrix A = poisson(Vh,Vh, tgv=tgv,solver=CG);
real[int] ff = external(0, Vh);
real[int] bc = poisson(0, Vh, tgv=tgv);
ff = bc ? bc : ff; // penalty term for inhomogenise data
u[] = A^-1 * ff;
plot(u);
```

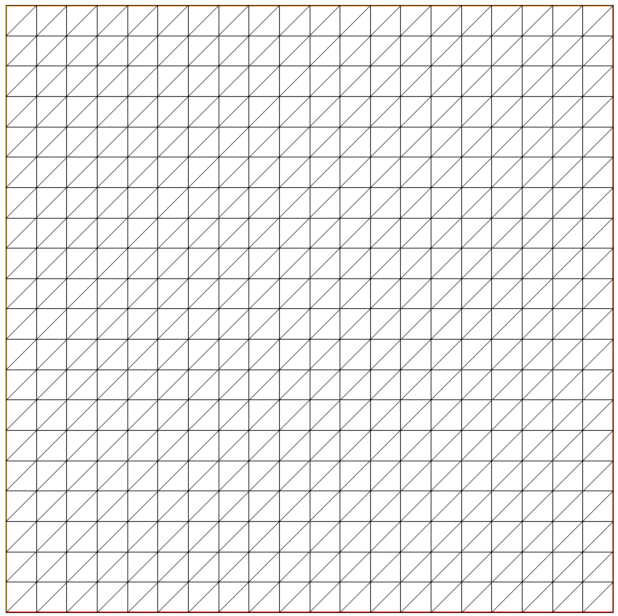
線形方程式に擾乱を加える

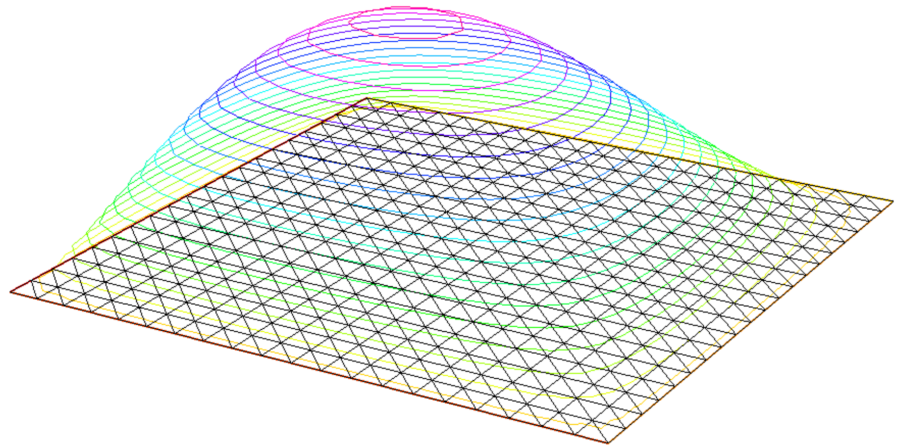
$$\tau u_k+\sum_{j\neq k}a_{kj}u_j=\tau g_k \quad \sum_j a_{ij}u_j=f_i \text{ for } i\neq k$$
$$\sum_j a_{ij}u_j=f_i \qquad \forall i\in\{1,\dots,N\}\setminus\Lambda_D.$$

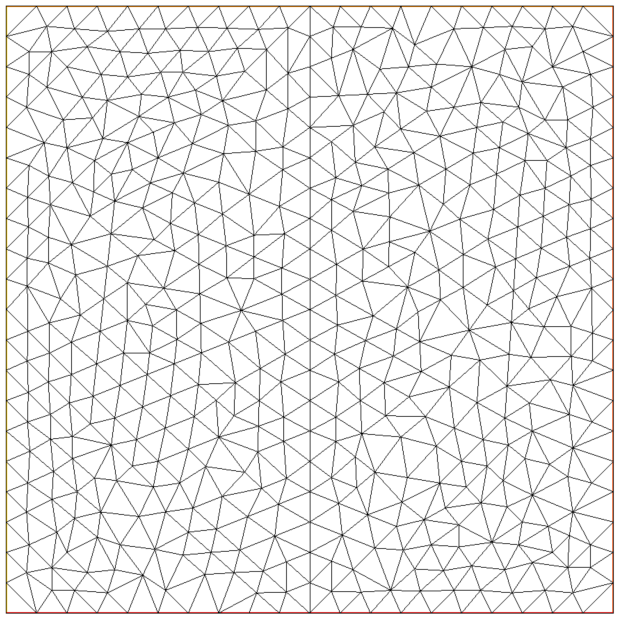
線形ソルバーの指定 `solver=`

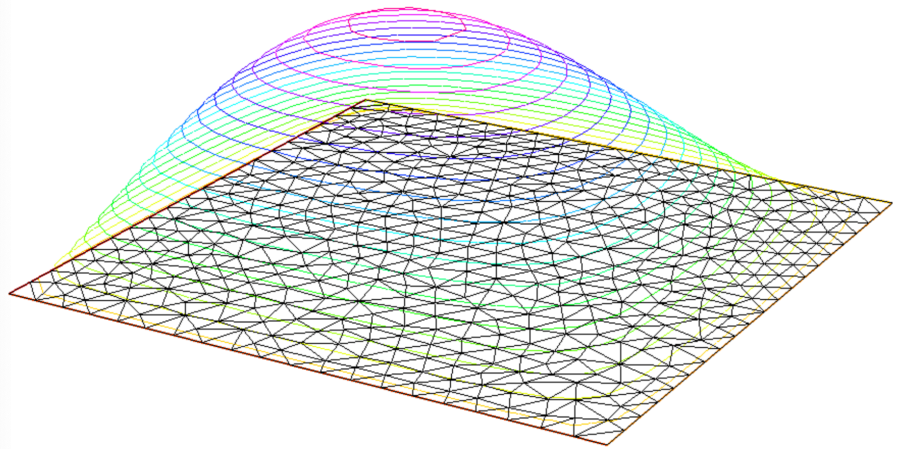
`CG` / `GMRES`	正定値対称行列 / 一般の行列
`sparsesolver`	Pardiso あるいは MUMPS を用いる

構造/非構造メッシュ









非構造メッシュを生成する FreeFEM スクリプト

```
int n1 = 20;
border bottom(t=0,1){x=t;y=0;   label=1;};
border right(t=0,1) {x=1;y=t;   label=2;};
border top(t=0,1)   {x=1-t;y=1; label=3;};
border left(t=0,1)  {x=0;y=1-t; label=4;};
mesh Th1=buildmesh(bottom(n1)+right(n1)+top(n1)
                   +left(n1));
...
fespace Vh10(Th1,P0);
Vh10 h1 = hTriangle;
real hmax = h1[].max;
...
```

メッシュ細分を用いる場合は分割の要素直径に何れを採用するかは議論が必要

$\min_K h_K$, $\sum_K h_K/\#\mathcal{T}_h$ と $\max_K h_K$ を計算しておくと良い

誤差評価を確認する FreeFEM スクリプト

```
int n1 = 20;
real hh1,hh2,err1,err2;
func sol = sin(pi*x)*sin(pi*y/2.0);
func solx = pi*cos(pi*x)*sin(pi*y/2.0);
func soly = (pi/2.0)*sin(pi*x)*cos(pi*y/2.0);
mesh Th1=square(n1,n1);
fespace Vh1(Th1,P2);
// calculation of a solution on Vh1 wth mesh Th1
matrix A = poisson(Vh1, Vh1, tgv=tgv,solver=CG);
real[int] ff = external(0, Vh1);
real[int] bc = poisson(0, Vh1, tgv=tgv);
ff = bc ? bc : ff;
u1[] = A^-1 * ff;
err1 = int2d(Th1)((dx(u1)-solx)*(dx(u1)-solx) +
                  (dy(u1)-soly)*(dy(u1)-soly) +
                  (u1-sol)*(u1-sol));
err1 = sqrt(err1);
// calculation of a solution on Vh2 with mesh Th2
hh1 = 1.0/n1*sqrt(2.0); hh2 = 1.0/n2*sqrt(2.0);
cout<<"O(h^2)="<<log(err1/err2)/log(hh1/hh2)<<endl;
```

誤差評価理論から有限要素解 $u_h \in S_h(Pk)$ の厳密解 $u \in H^2(\Omega)$ に対する誤差は要素サイズ h に関して次の評価がなりたつ

$$||u - u_h||_1 = c\,h^k, \quad \frac{||u - u_{h_1}||_1}{||u - u_{h_2}||_1} = \frac{c h_1^k}{c h_2^k} = \left(\frac{h_1}{h_2}\right)^k$$

FreeFEM の文法

繰り返し

```
for (int i=0; i<10; i++) {
  ...
  if (err < 1.0e-6) break;
}
//
int i = 0;
while (i < 10) {
  ...
  if (err < 1.0e-6) break;
  i++;
}
```

有限要素空間, 双一次形式と疎行列

```
fespace Xh(Th,P1)
Xh u,v;            // finite element data
varf a(u,v)=int2d(Th)( ... ) ;
matrix A = a(Xh,Xh,solver=UMFPACK);
real [int] v;  // array
v = A*u[];     // multiplication matrix to array
```

ユーザー定義関数

```
func real[int] ff(real[int] &pp) { // C++ reference
   ...
   return pp;                      // the same array
}
```

配列, ベクトル, FEM データ, 疎行列, ブロックデータ : 1/2

基本データ型

```
bool flag; // true or false
int i;
real w;
string st = "abc";
```

配列

```
real[int] v(10);  // real array whose size is 10
real[int] u;      // not yet allocated
u.resize(10);     // same as C++ STL vector
real[int] vv = v; // allocated as same size of v.n
a(2)=0.0  ;       // set value of 3rd index
a += b;           // a(i) = a(i) + b(i)
a = b .* c  ;      // a(i) = b(i) * c(i); element-wise
a = b < c ? b : c // a(i) = min(b(i), c(i)); C-syntax
a.sum;            // sum a(i);
a.n;              // size of array
```

ベクトルの $\ell^1, \ell^2, \ell^\infty$-ノルムを計算するための配列に対する操作が max, min に加えて利用できる.

配列, ベクトル, FEM データ, 疎行列, ブロックデータ : 2/2

有限要素オブジェクト

```
func fnc = sin(pi*x)*cos(pi*y); // function with x,y
mesh Th = ...;
fespace Vh(Th,P2);          // P2 space on mesh Th
Vh f;                       // FEM data on Th with P2
f[];                        // access data of FEM DOF
f = fnc;                    // interpolation onto FEM space
fespace Vh(Th,[P2,P2]); // 2 components P2 space
Vh [u1,u2];                 // u1[], u2[] is allocatd
u1[]=0.0;                   // access all data of [u1,u2];
real[int] uu([u1[].n+u2[].n);
u1[] = uu;                  // u1[], u2[] copied from uu
[u1[], u2[]] = uu;          // using correct block data
```

密行列と疎行列

```
real[int,int] B(10,10); // 2D array
varf aa(u,v)=int2d(Th)(u*v); // L2-inner prod. for mass
matrix A=aa(Vh,Vh,solver=sparsesolver); //sparse matrix
```

C++ と同様のファイル入出力,

```
Vh u;
{
  ofstream file("output.data");
  file << u; // with meta data
}  // scope of file object
```

```
Vh u;
{
  ifstream file("saved.data");
  for (int i=0; i<u[].n; i++) {
    file >> u[](i);// only number
  }
}  // scope of file object
```

2 次元での線形弾性体問題：1/2

$$\int_\Omega \Sigma(e(u)) : e(v) = \int_\Omega \lambda \mathrm{div}(u)\mathrm{div} v + 2\mu e(u) : e(v)$$

2×2 対称テンソルの積和は 3-次のベクトルと行列の積和で記述できることを利用する

$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{bmatrix}$.

$$\begin{aligned} \Sigma(A) : B &= (\lambda \mathrm{tr} A\, I + 2\mu A) : B \\ &= \begin{bmatrix} \lambda(a_{11}+a_{22}) + 2\mu a_{11} & 2\mu a_{12} \\ 2\mu a_{12} & \lambda(a_{11}+a_{22}) + 2\mu a_{22} \end{bmatrix} : \begin{bmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{bmatrix} \\ &= \begin{bmatrix} \lambda+2\mu & 0 & \lambda \\ 0 & \mu & 0 \\ \lambda & 0 & \lambda+2\mu \end{bmatrix} \begin{bmatrix} a_{11} \\ 2a_{12} \\ a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} \\ 2b_{12} \\ b_{22} \end{bmatrix} \end{aligned}$$

$$[e(u)] \leftrightarrow \left[\partial_1 u_1, (\partial_1 u_2 + \partial_2 u_1), \partial_2 u_2\right]$$

$$\Sigma(e(u)) : e(v) \leftrightarrow \begin{bmatrix} \lambda+2\mu & 0 & \lambda \\ 0 & \mu & 0 \\ \lambda & 0 & \lambda+2\mu \end{bmatrix} \begin{bmatrix} \partial_1 u_1 \\ \partial_1 u_2 + \partial_2 u_1 \\ \partial_2 u_2 \end{bmatrix} \cdot \begin{bmatrix} \partial_1 v_1 \\ \partial_1 v_2 + \partial_2 v_1 \\ \partial_2 v_2 \end{bmatrix}$$

Lamé 定数 λ と μ を含む 3×3 行列は対称である

2 次元での線形弾性体問題：2/2

```
load "PARDISO"
macro EL(u1, u2) [dx(u1), (dx(u2)+dy(u1)), dy(u2)] //
mesh Th = square(40, 10, [5*x, y]);
int[int] llabel = [1, 2, 2, 2, 3, 2, 4, 1];
Th = change(Th, label = llabel);
fespace Vh(Th, [P2, P2]);
real lambda = 1.0, mu = 1.0, gg = -0.5e-3;
func A = [ [lambda + 2.0 * mu, 0.0, lambda            ],
           [0.0,               mu , 0.0               ],
           [lambda,            0.0, lambda + 2.0 * mu] ];
Vh [u1, u2], [v1, v2];
varf lelasticity([u1, u2], [v1, v2])
= int2d(Th)(EL(u1, u2)' * A * EL(v1, v2)) //'
 + on (1, u1 = 0.0, u2 = 0.0);
varf rhs([u1, u2], [v1, v2])
= int2d(Th)(v2 * gg)  + on (1, u1 = 0.0, u2 = 0.0);
matrix AA = lelasticity(Vh, Vh);
set(AA, solver = "PARDISO", sym = 1);
real[int] bb = rhs(0, Vh);
real[int] xx = AA^-1 * bb;
u1[] = xx;
mesh Thm = movemesh(Th, [x + u1, y + u2]);   plot(Thm);
```

3 次元での線形弾性体問題：1/3

$$\int_\Omega \Sigma(e(u)) : e(v) = \int_\Omega \lambda \mathrm{div}(u)\mathrm{div}v + 2\mu e(u) : e(v)$$

3×3 対称テンソルの積和は 6-次のベクトルと行列の積和で記述できることを利用する

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix} \text{ and } B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{13} & b_{23} & b_{33} \end{bmatrix}.$$

$$\Sigma(A) : B = (\lambda \mathrm{tr} A\, I + 2\mu A) : B$$

$$= \begin{bmatrix} \lambda(\sum_k a_{kk}) + 2\mu a_{11} & 2\mu a_{12} & 2\mu a_{13} \\ 2\mu a_{12} & \lambda(\sum_k a_{kk}) + 2\mu a_{22} & 2\mu a_{23} \\ 2\mu a_{13} & 2\mu a_{23} & \lambda(\sum_k a_{kk}) + 2\mu a_{33} \end{bmatrix} : \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{12} & b_{22} & b_{23} \\ b_{31} & b_{23} & b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} \lambda+2\mu & & & \lambda & & \lambda \\ & \mu & & & & \\ & & \mu & & & \\ \lambda & & & \lambda+2\mu & & \lambda \\ & & & & \mu & \\ \lambda & & & \lambda & & \lambda+2\mu \end{bmatrix} \begin{bmatrix} a_{11} \\ 2a_{12} \\ 2a_{13} \\ a_{22} \\ 2a_{23} \\ a_{33} \end{bmatrix} \cdot \begin{bmatrix} b_{11} \\ 2b_{12} \\ 2b_{13} \\ b_{22} \\ 2b_{23} \\ b_{33} \end{bmatrix}$$

$$[e(u)] \leftrightarrow \left[\partial_1 u_1, (\partial_1 u_2 + \partial_2 u_1), (\partial_1 u_3 + \partial_3 u_1), \partial_2 u_2, (\partial_2 u_3 + \partial_3 u_2), \partial_3 u_3\right]$$

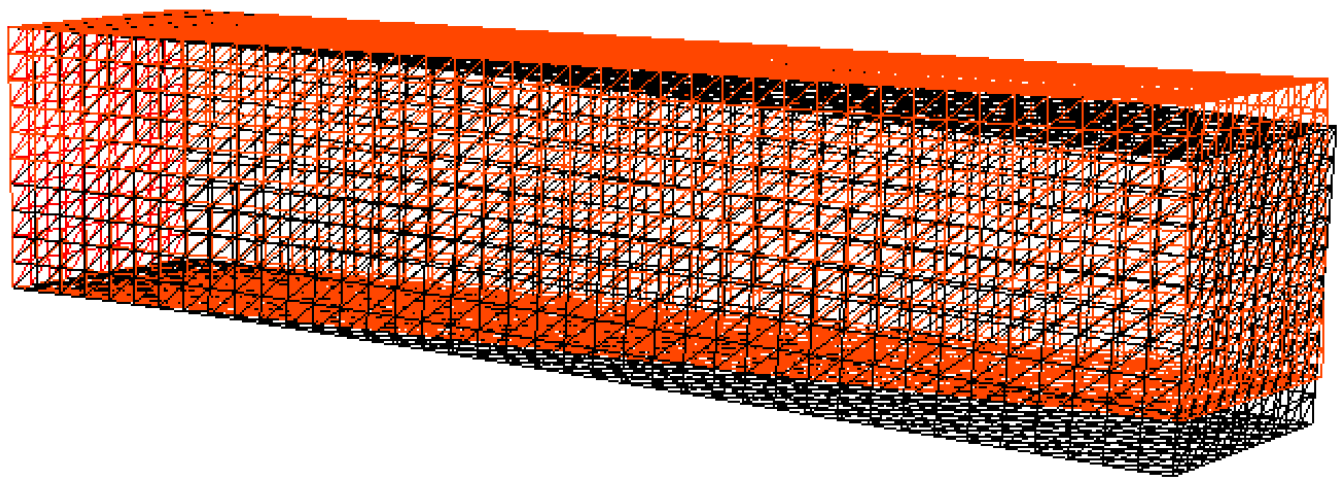
3 次元での線形弾性体問題：2/3

```
load "PARDISO"
macro EL(u1,u2,u3) [dx(u1),(dx(u2)+dy(u1)),(dx(u3)+dz(u1)),
                    dy(u2),(dy(u3)+dz(u2)),dz(u3)] //
include "cube.idp"
int[int]  Nxyz=[40,10,10];
real [int,int]  Bxyz=[[0.,5.],[0.,1.],[0.,1.]];
int [int,int]  Lxyz=[[1,2],[2,2],[2,2]];
mesh3 Th=Cube(Nxyz,Bxyz,Lxyz);
fespace Vh(Th, [P2, P2, P2]);
real lambda = 1.0, mu = 1.0, gg = -0.5e-3;
real a1 = lambda + 2.0 * mu;
real a2 = mu;
real a3 = lambda;
func A  = [ [a1,  0.0, 0.0, a3,  0.0, a3 ],
            [0.0, a2,  0.0, 0.0, 0.0, 0.0],
            [0.0, 0.0, a2,  0.0, 0.0, 0.0],
            [a3,  0.0, 0.0, a1,  0.0, a3 ],
            [0.0, 0.0, 0.0, 0.0, a2,  0.0],
            [a3,  0.0, 0.0, a3,  0.0, a1 ] ]; //

Vh [u1, u2, u3], [v1, v2, v3];
varf lelasticity([u1, u2, u3], [v1, v2, v3])
= int3d(Th,qfV=qfV1)(EL(u1, u2, u3)' * A * EL(v1, v2, v3)) //'
+ on (1, u1 = 0.0, u2 = 0.0, u3 = 0.0);

varf rhs([u1, u2, u3], [v1, v2, v3])
= int3d(Th,qfV=qfV1)(v3 * gg)
+ on(1, u1 = 0.0, u2 = 0.0, u3 = 0.0);
```

3 次元での線形弾性体問題：3/3

```
matrix AA = lelasticity(Vh, Vh);
set(AA, solver = "PARDISO", sym = 1);
real[int] bb = rhs(0, Vh);
real[int] xx = AA^-1 * bb;
u1[] = xx;

mesh3 Thm = movemesh3(Th,transfo=[x + u1, y + u2, z + u3]);
plot(Th, Thm);
```

3 次元の片持ち梁：左端を固定した場合の三次元の変位場



非線形弾性体問題：対称な Jacobian 行列

$$\int_\Omega \Sigma(dE(u^n)[w]) : dE(u^n)[v] + \Sigma(E(u^n)) : d^2E(u^n)[v,w]$$

```
macro ENL(u1, u2, u3) [
  (dx(u1)*dx(u1)+dx(u2)*dx(u2)+dx(u3)*dx(u3))*0.5,
  (dx(u1)*dy(u1)+dx(u2)*dy(u2)+dx(u3)*dy(u3)),
  (dx(u1)*dz(u1)+dx(u2)*dz(u2)+dx(u3)*dz(u3)),
  (dy(u1)*dy(u1)+dy(u2)*dy(u2)+dy(u3)*dy(u3))*0.5, // ...
macro dENL(u1, u2, u3, v1, v2, v3) [
  (dx(u1)*dx(v1)+dx(u2)*dx(v2)+dx(u3)*dx(v3)),
  (dx(u1)*dy(v1)+dx(v1)*dy(u1)+
   dx(u2)*dy(v2)+dx(v2)*dy(u2)+
   dx(u3)*dy(v3)+dx(v3)*dy(u3)),
  (dx(u1)*dz(v1)+dx(v1)*dz(u1)+
   dx(u2)*dz(v2)+dx(v2)*dz(u2)+
   dx(u3)*dz(v3)+dx(v3)*dz(u3)),
  (dy(u1)*dy(v1)+dy(u2)*dy(v2)+dy(u3)*dy(v3)), //...
macro E(u1, u2, u3) (EL(u1, u2, u3) + ENL(u1, u2, u3)) //
macro dE(u1, u2, u3, v1, v2, v3) (EL(v1, v2, v3)+
                                  dENL(u1, u2, u3, v1, v2, v3)) //
varf nelasticity([w1, w2, w3], [v1, v2, v3])
= int3d(Th,qfV=qfV1)(dE(u1, u2, u3, w1, w2, w3)' * A *
                     dE(u1, u2, u3, v1, v2, v3)) +
                     E(u1, u2, u3)' * A *
                     dENL(w1, w2, w3, v1, v2, v3)) + on( ... //
```

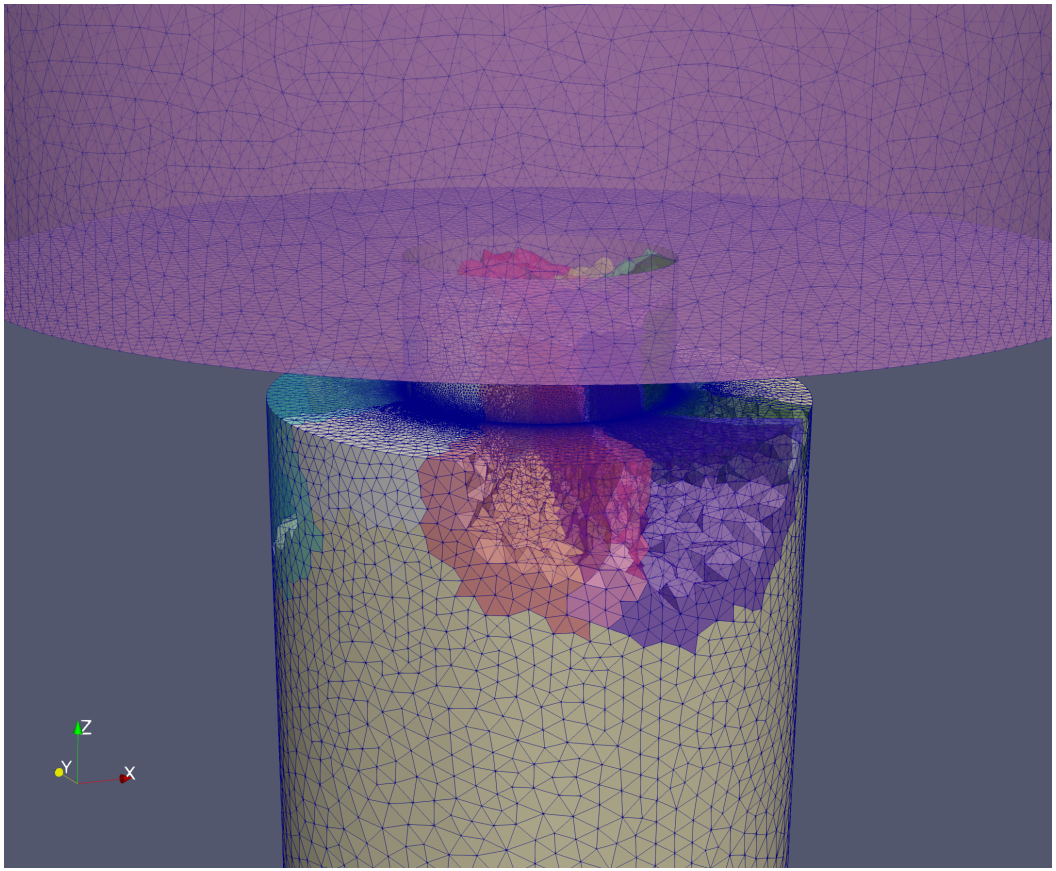
METIS による重なりのある領域分割と HPDDM による解法: 1/3

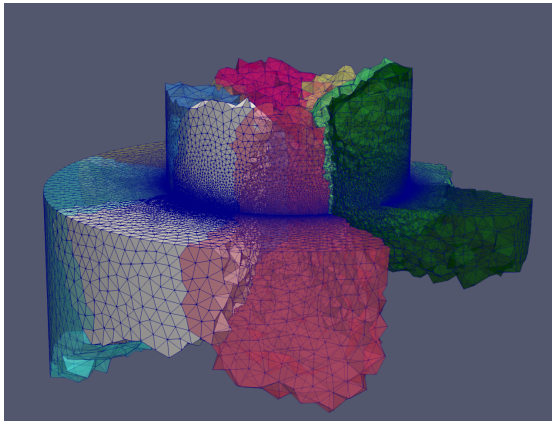
```
load "hpddm"
mpiComm comm(mpiCommWorld,0,0);
mesh3 ThGlobal, Th;
ThGlobal = gmshload3("cylinders.msh"); // read Gmsh mesh
func Pk = [P1, P1, P1];
fespace Vhg(ThGlobal, Pk);      // FE space on global domain
fespace Vh(Th, Pk);  // FE space on local domain
int[int][int] intersection;
real[int] D;
{
  Th = ThGlobal;
  build(Th, 1, intersection, D, Pk, comm, 3)
}
matrix Rgl = interpolate(Vh, Vhg);
matrix Dh = [D]; // diagonal matrix
```

- `mesh3 Th`：$\widetilde{\Omega_p}$：一層の重なりのある領域分割
- `matrix Rgl`：全体自由度から部分自由度への制限作用素 R_p
- `matrix D`：離散的な単位の分解の重み

METIS による重なりのある領域分割と HPDDM による解法: 2/3

12 部分領域への分割, ParaView による可視化





- macro `build()`：HPDDM の組み込みマクロで重なりのある領域分割を生成する
- `METIS` は双対要素分割による有限要素の領域を実現するが, 領域間の境界は滑らかでなはい
- 加法的 Schwarz 前処理は滑らかな境界を持つ領域分割でなくても堅牢な収束を実現する

METIS による重なりのある領域分割と HPDDM による解法: 3/3

```
load "hpddm"
mpiComm comm(mpiCommWorld,0,0);
mesh3 ThGlobal, Th;
fespace Vh(Th, Pk);  // FE space on local domain
// ...
totaldof = Vhg.ndof;
localdof = Vh.ndof;
real[int] xxg(totaldof), xdl(localdof), xxl(localdof);
matrix AA = lelasticity(Vh, Vh, tgv = -1, sym=1);
real[int] bb = rhs(0, Vh, tgv= -1);
schwarz dA(AA, intersection, D);
set(dA, sparams = "-hpddm_schwarz_method ras " +
  "-hpddm_variant right -hpddm_tol 1.e-8 " +
  "-hpddm_gmres_restart 150 -hpddm_max_it 150" +
  "-hpddm_verbosity 10 -hpddm_tol 1.0e-10")
xxl = AA^-1 * bb;     // solution is distributed
xdl = Dh * xxl;       // adjust with weights for overlapping
xxg = Rgl' * xdl;      //'  local -> global
mpiAllReduce(wg, u1[], comm, mpiSUM); //
```

- 剛性行列は双一次形式から部分領域の要素分割 `Th` と有限要素空間 `Vh` により並列に生成される
- 全体の解は MPI の総和演算 $\sum_p R_p^T D_p u_p$ により部分領域の解をまとめることで得られる

計算効率

2,076,809 自由度での線形/非線形弾性体問題の解

ソルバー	線形弾性体			メモリー
	行列生成	右辺生成	ソルバー	
`pardiso` symmetric	52.17	0.04	432.50	40.2GB
`hpddm` unsymmetric	20.30	0.01	352.69	5.1G×12
`hypre` unsymmetric	20.33	0.01	72.03	1.5G×12

ソルバー	非線形弾性体			メモリー
	行列生成	右辺生成	ソルバー	
`pardiso` symmetric	233.34	129.33	430.60	40.2GB
`hpddm` unsymmetric	42.44	22.15	366.02	5.1G×12
`hypre` unsymmetric	45.01	20.1	87.42	1.5G×12

- 非線形弾性体問題では剛性行列の生成にかかる時間は無視できない
- `hypre` multigrid 前処理は非線形性が強くなると収束しなくなる

FreeFEM とオープンソフトウェア

最新の FreeFem のバージョンは 4.13.
インストール方法は https://doc.freefem.org/introduction/installation.html

- `FreeFEM` : 高レベルの複合物理シミュレーションのための有限要素法ソフトウェア
  `https://freefem.org`
- `gmsh` : 3 次元のメッシュ生成ソフトウェア, 領域記述のためのスクリプト言語を用いるか, `STEP` ファイルを `OpenCASCADE` エンジンにより解釈
  `https://gmsh.info`
- `mmg3d` : 要素細分のための 3 次元のメッシュソフトウェア
  `https://www.mmgtools.org`
- `hpddm` : 加法的 Schwarz 前処理などの領域分割を用いるソルバー
  `https://github.com/hpddm/hpddm`
- `paraview` : 3 次元の可視化ソフトウェア
  `https://www.paraview.org`

外部ソフトウェアの FreeFEM へのインターフェースは
FreeFem-soucres/plugin/{seq,mpi}/
FreeFEM のスクリプトの例は FreeFem-soucres/examples/
FreeFEM の計算結果を ParaView により可視化するには次のスクリプトを用いる.

```
load "iovtk"
mesh3 Th = ... ;
fespace Vh(Th, P1);
Vh u;
...
int[int] vtkorder = [1];
savevtk("output.vtk", Th, u, order = vtkorder);
```