Finite element programming by FreeFem++ – intermediate course

Atsushi Suzuki[1]

[1]Cybermedia Center, Osaka University
atsushi.suzuki@cas.cmc.osaka-u.ac.jp

Japan SIAM tutorial
11-12, February 2016

Numerical simulation with finite element method

- mathematical modeling
- discretization of time for evolution problem
- discretization scheme for the space
  - mesh generation / adaptive mesh refinement
  - stiffness matrix from finite elements and variational formulation
  - linear solver $\Leftarrow$ `CG`, `GMRES`, direct solver: `UMFPACK`, `MUMPS`

  FreeFem++ provides vast amounts of tools
- nonlinear solver
- optimization solver

parallel computation is another topic.
distributed parallelization by MPI needs to be described by FreeFem++ script.

Outline I

Basics of FEM by examples from the Poisson equation

- Poisson equation with mixed boundary conditions
- error estimate by theory and FreeFem++ implementation

Mixed formulation for the Stokes equations

- Stokes equations with inhomogenous Dirichlet conditions
- mixed formulation and inf-sup conditions
- finite element pair satisfying inf-sup conditions

Nonlinear finite element problem by Newton method

- stationary Navier-Stokes equations
- differential calculus of nonlinear operator and Newton iteration

Time-dependent Navier-Stokes equations around a cylinder

- boundary conditions of incompressible flow around a cylinder
- characteristic Galerkin method
- stream line for visualization

Outline II

- thermal convection problem by Rayleigh-Bénard eqs.
  - governing equations by Boussinesq approximation
  - time-dependent solution by characteristic Galerkin method
  - stationary solution by Newton method
- Conjugate Gradient solver in FreeFem++
  - basic CG method with preconditioning
  - CG method with orthogonal projection onto the image
  - CG method in Uzawa method
- Syntax and Data types of FreeFem++
  - syntax for loop and procedure
  - Data types
- Compilation from the source on Unix system
  - configure script with option and BLAS library
- References

Poisson equation with mixed B.C. and a weak formulation: 1/2

$\Omega \subset \mathbb{R}^2$, $\partial\Omega = \Gamma_D \cup \Gamma_N$

$$-\triangle u = f \text{ in } \Omega,$$
$$u = g \text{ on } \Gamma_D,$$
$$\frac{\partial u}{\partial n} = h \text{ on } \Gamma_N.$$

weak formulation

V : function space, $V(g) = \{u \in V\,;\, u = g \text{ on } \Gamma_D\}$.

$V = C^1(\Omega) \cap C^0(\bar{\Omega})$?

Find $u \in V(g)$ s.t.

$$\int_\Omega -\triangle u\, v dx = \int_\Omega f\, v dx \quad \forall v \in V(0)$$

Lemma (Gauss-Green's formula)

$u, v \in V$, $n = \begin{bmatrix} n_1 \\ n_2 \end{bmatrix}$: outer normal to $\partial\Omega$

$$\int_\Omega (\partial_i u) v\, dx = -\int_\Omega u \partial_i v\, dx + \int_{\partial\Omega} u\, n_i v\, ds\,.$$

Poisson equation with mixed B.C. and a weak formulation: 2/2

$$\int_\Omega (-\partial_1^2 - \partial_2^2) u\, v\, dx = \int_\Omega (\partial_1 u \partial_1 v + \partial_2 u \partial_2 v)\, dx - \int_{\partial\Omega} (\partial_1 u\, n_1 + \partial_2 u\, n_2) v\, ds$$

$$= \int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\Gamma_D \cup \Gamma_N} \nabla u \cdot n\, v\, ds$$

$$v = 0 \text{ on } \Gamma_D \Rightarrow \quad = \int_\Omega \nabla u \cdot \nabla v\, dx - \int_{\Gamma_N} h v\, ds$$

Find $u \in V(g)$ s.t.

$$\int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega f\, v dx + \int_{\Gamma_N} h\, v ds \quad \forall v \in V(0)$$

- $a(\cdot,\cdot) : V \times V \to \mathbb{R}$: bilinear form
- $F(\cdot) : V \to \mathbb{R}$: functional

Find $u \in V(g)$ s.t.

$$a(u, v) = F(v) \quad \forall v \in V(0)$$

FreeFem++ script to solve Poisson equation

finite element basis, span$[\varphi_1, \ldots, \varphi_N] = V_h \subset V$
$u_h \in V_h \;\Rightarrow\; u_h = \sum_{1\le i\le N} u_i \varphi_i$
Dirichlet data : $u(P_j) = g(P_j) \quad P_j \in \Gamma_D$
Find $u_h \in V_h(g)$ s.t.

$$\int_\Omega \nabla u_h \cdot \nabla v_h dx = \int_\Omega f\, v_h dx + \int_{\Gamma_N} h\, v_h ds \quad \forall v_h \in V_h(0).$$

▸ example1.edp ▸▸ varf+matrix

```
mesh Th=square(20,20);
fespace Vh(Th,P1);
Vh uh,vh;
func f = 5.0/4.0*pi*pi*sin(pi*x)*sin(pi*y/2.0);
func g = sin(pi*x)*sin(pi*y/2.0);
func h = (-pi)/2.0 * sin(pi * x);
solve poisson(uh,vh)=
   int2d(Th)( dx(uh)*dx(vh)+dy(uh)*dy(vh) )
 - int2d(Th)( f*vh ) - int1d(Th,1)( h *vh )
 + on(2,3,4,uh=g);          // boudary 1 : (x,0)
plot(uh);
```

discretization and matrix formulation : 1/2

finite element basis, span$[\varphi_1, \dots, \varphi_N] = V_h \subset V$
$u_h \in V_h \;\Rightarrow\; u_h = \sum_{1\leq i\leq N} u_i\varphi_i$
finite element nodes $\{P_j\}_{j=1}^N$, $\varphi_i(P_j) = \delta_{i\,j}$ Lagrange element
$\Lambda_D \subset \Lambda = \{1, \dots, N\}$: index of node on the Dirichlet boundary

$$V_h(g) = \{u_h \in V_h \,;\, u_h = \sum u_i\varphi_i, u_k = g_k \; (k \in \Lambda_D)\}$$

Find $u_h \in V_h(g)$ s.t.

$$a(u_h, v_h) = F(v_h) \quad \forall v_h \in V_h(0).$$

Find $\{u_j\}, u_k = g_k(k \in \Lambda_D)$ s.t.

$$a(\sum_j u_j\varphi_j, \sum_i v_i\varphi_i) = F(\sum_i v_i\varphi_i) \; \forall\{v_i\}, v_k = 0(k \in \Lambda_D)$$

Find $\{u_j\}_{j\in\Lambda}$s.t.

$$\sum_j a(\varphi_j, \varphi_i)u_j = F(\varphi_i) \qquad \forall i \in \Lambda \setminus \Lambda_D$$

$$u_k = g_k \qquad \forall k \in \Lambda_D$$

discretization and matrix formulation : 2/2

Find $\{u_j\}_{j\in\Lambda\setminus\Lambda_D}$ s.t.

$$\sum_{j\in\Lambda\setminus\Lambda_D} a(\varphi_j,\varphi_i)u_j = F(\varphi_i) - \sum_{k\in\Lambda_D} a(\varphi_k,\varphi_i)g_k \quad \forall i\in\Lambda\setminus\Lambda_D$$

$A = \{a(\varphi_j,\varphi_i)\}_{i,j\in\Lambda\setminus\Lambda_D}$: symmetric.
$A\in\mathbb{R}^{n\times n}$, $f\in\mathbb{R}^n$, $n = \#(\Lambda\setminus\Lambda_D)$

Lemma

A *: (symmetric) positive definite* $\Leftrightarrow (Au,u) > 0\ \forall u\neq 0$
$\Rightarrow\ Au = f$ *has a unique solution.*

A : bijective

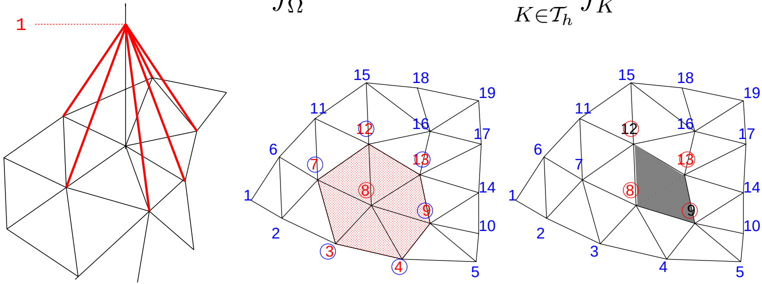
- injective: $Au = 0$, $0 = (Au,u) > 0 \ \Rightarrow u = 0$.
- surjective:
  $\mathbb{R}^n = \mathrm{Im}A\oplus(\mathrm{Im}A)^\perp$, $u\in(\mathrm{Im}A)^\perp \Rightarrow (Av,u) = 0\ \forall v\in\mathbb{R}^n$
  by putting $v = u$, $0 = (Au,u) \ \Rightarrow u = 0$
  $(\mathrm{Im}A)^\perp = \{0\} \Rightarrow \mathrm{Im}A = \mathbb{R}^n$.

A : S.P.D. $\Rightarrow$ solution by LDL^T-factorization, CG method

P1 finite element and sparse matrix

$\mathcal{T}_h$: triangulation of a domain Ω, triangular element $K \in \mathcal{T}_h$
piecewise linear element : $\varphi_i|_K(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_2$
$\varphi_i|_K(P_j) = \delta_{i\,j}$

$$[A]_{i\,j} = a(\varphi_j, \varphi_i) = \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i \, dx = \sum_{K \in \mathcal{T}_h} \int_K \nabla\varphi_j \cdot \nabla\varphi_i \, dx.$$



A : sparse matrix, CRS (Compressed Row Storage) format to store

FreeFem++ script to solve Poisson eq. using matrix

Find $u_h \in V_h(g)$ s.t. $a(u_h, v_h) = F(v_h)\ \forall v_h \in V_h(0)$.

▸ example2.edp ▸▸ solve

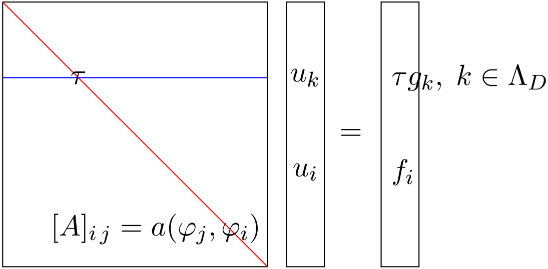
```
Vh u,v;
varf aa(u,v)=int2d(Th)( dx(u)*dx(v)+dy(u)*dy(v) )
            +on(2,3,4,u=g);
varf external(u,v)=int2d(Th)(f*v)+int1d(Th,1)(h*v)
            +on(2,3,4,u=g);
real tgv=1.0e+30;
matrix A = aa(Vh,Vh,tgv=tgv,solver=CG);
real[int] ff = external(0,Vh,tgv=tgv);
u[] = A^-1 * ff;     // u : fem unknown, u[] : vector
plot(u);
```

useful liner solver; `solver=`

`CG`	iterative solver for SPD matrix
`GMRES`	iterative solver for nonsingular matrix
`UMFPACK`	direct solver for nonsingular matrix
`sparsesolver`	other solvers called by dynamic link

penalty method to solve inhomogeneous Dirichlet problem

modification of diagonal entries of A where index $k \in \Lambda_D$
penalization parameter $\tau = 1/\varepsilon$; tgv



$$\tau u_k + \sum_{j \neq k} a_{k\,j} u_j = \tau g_k \;\Leftrightarrow\; u_k - g_k = \varepsilon(-\sum_{j \neq k} a_{k\,j} u_j),$$

$$\sum_j a_{i\,j} u_j = f_i \qquad \forall i \in \{1, \ldots, N\} \setminus \Lambda_D.$$

keeping symmetry of the matrix without changing index numbering.

abstract framework

V: Hilbert space with inner product $(\cdot,\cdot)$ and norm $||\cdot||$.
bilinear form $a(\cdot,\cdot) : V \times V \to \mathbb{R}$

- coercive : $\exists \alpha > 0 \quad a(u,u) \geq \alpha ||u||^2 \; \forall u \in V$.
- continuous : $\exists \gamma > 0 \quad |a(u,v)| \leq \gamma ||u|| \, ||v|| \; \forall u, v \in V$.

functional $F(\cdot) : V \to \mathbb{R}$.
find $u \in V$ s.t. $a(u,v) = F(v) \quad \forall v \in V$
has a unique solution : Lax-Milgram's theorem

inf-sup conditions + continuity of $a(\cdot,\cdot)$

- $\exists \alpha_1 > 0 \qquad \sup_{v \in V, v \neq 0} \dfrac{a(u,v)}{||v||} \geq \alpha_1 ||u|| \; \forall u \in V$.
- $\exists \alpha_2 > 0 \qquad \sup_{u \in V, u \neq 0} \dfrac{a(u,v)}{||u||} \geq \alpha_2 ||v|| \; \forall v \in V$.

find $u \in V$ s.t. $a(u,v) = F(v) \quad \forall v \in V$ has a unique solution.

error estimate : theory 1 /2

V : Hilbert space, $V_h \subset V$: finite element space.

- $u \in V$, $a(u, v) = F(v) \;\; \forall v \in V$.
- $u_h \in V_h$, $a(u_h, v_h) = F(v_h) \;\; \forall v_h \in V_h \subset V$.

$a(u, v_h) = F(v_h) \;\; \forall v_h \in V_h \subset V$.

Lemma (Galerkin orthogonality)

$$a(u - u_h, v_h) = 0 \;\; \forall v_h \in V_h.$$

assuming coercivity and continuity of $a(\cdot, \cdot)$.

Lemma (Céa)

$$||u - u_h|| \leq \frac{\gamma}{\alpha} \inf_{v_h \in V_h} ||u - v_h||.$$

proof: $||u - u_h|| \leq ||u - v_h|| + ||v_h - u_h||$

$$\begin{aligned}
\alpha ||u_h - v_h||^2 &\leq a(u_h - v_h, u_h - v_h) \\
&= a(u_h, u_h - v_h) - a(v_h, u_h - v_h) \\
&= a(u, u_h - v_h) - a(v_h, u_h - v_h) \\
&= a(u - v_h, u_h - v_h) \leq \gamma ||u - v_h|| ||u_h - v_h||.
\end{aligned}$$

Sobolev space : 1/2

P1 element element space does not belong to $C^1(\Omega)$.
$V = H^1(\Omega)$, $(u, v) = \int_\Omega u\, v + \nabla u \cdot \nabla v$, $||u||_1^2 = (u, u) < +\infty$.
$||u||_0^2 = \int_\Omega u\, u, \quad |u|_1^2 = \int_\Omega \nabla u \cdot \nabla u$.
$H_0^1 = \{u \in H^1(\Omega)\,;\, u = 0 \text{ on } \partial\Omega\}$.

Lemma (Poincaré's inequality)

$\exists C(\Omega)\ u \in H_0^1 \ \Rightarrow\ ||u||_0 \leq C(\Omega)|u|_1$.

proof:

$\Omega \subset B = (0, s) \times (0, s)$. $\quad v \in C_0^\infty(\Omega)$, $\tilde{v}(x) = 0\ (x \in B \setminus \bar{\Omega})$.

$$v(x_1, x_2) = v(0, x_2) + \int_0^{x_1} \partial_1 v(t, x_2) dt$$

$$|v(x_1, x_2)|^2 \leq \int_0^{x_1} 1^2 dt \int_0^{x_1} |\partial_1 v(t, x_2)|^2 dt \leq s \int_0^s |\partial_1 v(t, x_2)|^2 dt$$

$$\int_0^s |v(x_1, x_2)|^2 dx_1 \leq s^2 \int_0^s |\partial_1 v(x)|^2 dx_1$$

$$\int_\Omega |v|^2 = \int_B |v|^2 dx_1 dx_2 \leq s^2 \int_B |\partial_1 u|^2 dx_1 dx_2 = s^2 \int_\Omega |\partial_1 u|^2.$$

Sobolev space : 2/2

$a(u,v) = \int_\Omega \nabla u \cdot \nabla v, \quad u, v \in H^1(\Omega)$.

- $a(\cdot,\cdot)$ is coercive on $H_0^1(\Omega)$.
- $a(\cdot,\cdot)$ is coercive on $H^1(\Omega)/\mathbb{R}$.

full-Neumann boundary problem

$$-\triangle u = f \text{ in } \Omega,$$
$$\partial_n u = h \text{ on } \partial\Omega.$$

- $(F, v) = F(v) = \int_\Omega f\, v + \int_{\partial\Omega} h\, v$
- compatibility condition : $(F, 1) = \int_\Omega f + \int_{\partial\Omega} h = 0$

(N) Find $u \in H^1(\Omega)$ s.t.

$$a(u,v) = F(v) \quad \forall v \in H^1(\Omega)$$

u: solution of (N) $\Rightarrow$ $u + 1$: solution of (N)
$[A]_{i\,j} = a(\varphi_j, \varphi_i)$. A : singular , Ker$A = \vec{1}$.

(N) has a unique solution in $H^1(\Omega)/\mathbb{R} \simeq \{u \in H^1(\Omega)\,;\, \int_\Omega u = 0\}$.

error estimate : theory 2 /2

$\Pi_h : C(\bar{\Omega}) \to V_h, \quad \Pi_h u = \sum_i u(P_i)\varphi_i,$
$\{\varphi_i\}$: P_k finite element basis, span$[\{\varphi_i\}] = V_h$.

Theorem (interpolation error by polynomial)

$K \in \mathcal{T}_h$, $P_k(K) \subset H^l(K)$, $v \in H^{k+1}(\Omega)$

$$\Rightarrow \quad \exists c > 0 \quad |v - \Pi_h v|_{s,K} \leq C\, h_K^{k+1-s} |v|_{k+1,K}\,, \quad 0 \leq s \leq \min\{k+1, l\}.$$

Theorem (finite element error)

$u \in H^{k+1}$, u_h *: finite element solution by* P_k *element.*

$$\Rightarrow \quad \exists c > 0 \quad ||u - u_h||_{1,\Omega} \leq C\, h^k |u|_{k+1,\Omega}$$

proof:

$$\begin{aligned} ||u - u_h||_{1,\Omega} &\leq C \inf_{v_h \in V_h} ||u - v_h||_{1,\Omega} \\ &\leq C||u - \Pi_h u||_{1,\Omega} \\ &\leq C\textstyle\sum_{K \in \mathcal{T}_h} (h_K^k + h_K^{(k+1)})|u|_{k+1,K} \\ &\leq C h^k |u|_{k+1,\Omega} \end{aligned}$$

$\mathcal{T}_h$: finite element mesh, $h_K =$ diam(K), $h = \max_{K \in \mathcal{T}_h} h_K$.

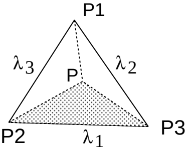
numerical integration

Numerical quadrature:
$\{P_i\}_{i\leq i\leq m}$: integration points in K, $\{\omega_i\}_{i\leq i\leq m}$: weights

$$|u-u_h|^2_{0,\Omega}=\sum_{K\in\mathcal{T}_h}\int_K|u-u_h|^2dx\sim\sum_{K\in\mathcal{T}_h}\sum_{i=1}^{m}|(u-u_h)(P_i)|^2\omega_i$$

formula : degree 5, 7 points, `qf5pT`,
P.C. Hammer, O.J. Marlowe, A.H. Stroud [1956]

area coordinates $\{\lambda_i\}_{i=1}^3$	weight	
$(\frac{1}{3},\frac{1}{3},\frac{1}{3})$	$\frac{9}{40}\lvert K\rvert$	$\times 1$
$(\frac{6-\sqrt{15}}{21},\frac{6-\sqrt{15}}{21},\frac{9+2\sqrt{15}}{21})$	$\frac{155-\sqrt{15}}{1200}\lvert K\rvert$	$\times 3$
$(\frac{6+\sqrt{15}}{21},\frac{6+\sqrt{15}}{21},\frac{9-2\sqrt{15}}{21})$	$\frac{155+\sqrt{15}}{1200}\lvert K\rvert$	$\times 3$



Remark
it is not good idea to use interpolation of continuous function to finite element space, for verification of convergence order.
$|\Pi_h u-u_h|_{1,\Omega}$ may be smaller (in extreme cases, super convergence)

numerical convergence order

for observing convergence order
$u \in H^2(\Omega)$: manufactured solution
$u_h \in V_h(g)$: finite element solution by P_k element.

$$||u - u_h||_{1,\Omega} = c\, h^k,$$
$$\frac{||u - u_{h_1}||_{1,\Omega}}{||u - u_{h_2}||_{1,\Omega}} = \frac{ch_1^k}{ch_2^k} = (\frac{h_1}{h_2})^k$$

numerical convergence order:
$\kappa = \log(\frac{||u - u_{h_1}||_{1,\Omega}}{||u - u_{h_2}||_{1,\Omega}}) / \log(\frac{h_1}{h_2})$.

FreeFem++ script for error estimation

example3.edp

```
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);
mesh Th2=square(n2,n2);
fespace Vh1(Th1,P1);
  ...
solve poisson1(u1,v1) = ...
  ...
err1 = int2d(Th1)((dx(u1)-solx)*(dx(u1)-solx) +
                  (dy(u1)-soly)*(dy(u1)-soly) +
                  (u1-sol)*(u1-sol));
err1 = sqrt(err1);
...
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;
```

unstructured mesh is generated by Delaunay triangulation

▸ example4.edp

```
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;
hh1 = h1[].max;
...
```

Remark

$\min_K h_K$, $\sum_K h_K/\#\mathcal{T}_h$, $\max_K h_K$, corresponding to mesh refinement are observed by following:

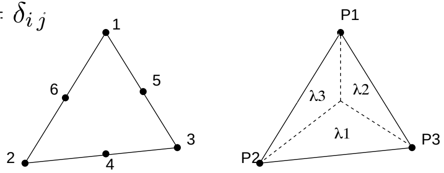
```
h1[].min;  hh1 = h1[].sum / h1[].n;  h1[].max;
```

P2 finite element

$\mathcal{T}_h$: triangulation of a domain Ω, triangular element $K \in \mathcal{T}_h$
piecewise quadratic element : 6 DOF on element K.

$$\varphi_i|_K(x_1, x_2) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_1 x_2 + a_5 x_2^2$$

$$\varphi_i|_K(P_j) = \delta_{i\,j}$$



by using area coordinates $\{\lambda_1, \lambda_2, \lambda_3\},\ \lambda_1 + \lambda_2 + \lambda_3 = 1$.

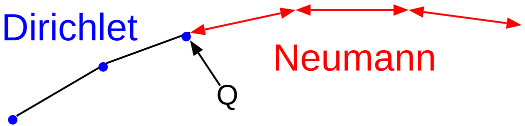
$$\begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \varphi_4 \\ \varphi_5 \\ \varphi_6 \end{pmatrix} = \begin{pmatrix} 1 & & & & -1 & -1 \\ & 1 & & -1 & & -1 \\ & & 1 & -1 & -1 & \\ & & & 4 & & \\ & & & & 4 & \\ & & & & & 4 \end{pmatrix} \begin{pmatrix} \lambda_1^2 \\ \lambda_2^2 \\ \lambda_3^2 \\ \lambda_2\lambda_3 \\ \lambda_3\lambda_1 \\ \lambda_1\lambda_2 \end{pmatrix} = \begin{pmatrix} \lambda_1(2\lambda_1 - 1) \\ \lambda_2(2\lambda_2 - 1) \\ \lambda_3(2\lambda_3 - 1) \\ 4\lambda_2\lambda_3 \\ 4\lambda_3\lambda_1 \\ 4\lambda_1\lambda_2 \end{pmatrix}$$

```
fespace Vh(Th,P2);
```

treatment of Neumann data around mixed boundary

Neumann data is evaluated by line integral with FEM basis φ_i.

$$\int_{\Gamma_N} h\,\varphi_i\,ds$$



For given discrete Neumann data, h is interpolated in FEM space, $h = \sum_j h_j \varphi_j|_{\Gamma_N}$,

$$\sum_j h_j \int_{\Gamma_N} \varphi_j \varphi_i\,ds.$$

On the node $Q \in \bar{\Gamma}_D \cap \bar{\Gamma}_N$, both Dirichlet and Neumann are necessary.

advantages of finite element formulation

- weak formulation is obtained by integration by part with clear description on the boudnary
- Dirichlet boundary condition is embedded in a functional space, called as essential boundary condition
- Neumann boundary condition is treated with surface/line integral by Gauss-Green's formula, called as natural boundary condition
- solvability of linear system is inherited from solvability of continuous weak formulation
- error of finite element solution is evaluated by approximation property of finite element space

better to learn for efficient computation

- treatment of Dirichlet boudary conditions in FreeFem++ with explicit usage of matrix and linear solver

Stokes equations and a weak formulation : 1/3

$\Omega = (0,1) \times (0,1)$

$$\begin{aligned} -2\nabla \cdot D(u) + \nabla p &= f \text{ in } \Omega \\ \nabla \cdot u &= 0 \text{ in } \Omega \\ u &= g \text{ on } \partial\Omega \end{aligned}$$

strain rate tensor : $[D(u)]_{i\,j} = \frac{1}{2}(\partial_i u_j + \partial_j u_i)$.

- $V(g) = \{v \in H^1(\Omega)^2\,;\, v = g \text{ on } \partial\Omega\},\ V = V(0)$
- $Q = L^2_0(\Omega) = \{p \in L^2(\Omega)\,;\, \int_\Omega p\,dx = 0\}$

bilinear form and weak formulation :

$$a(u,v) = \int_\Omega 2D(u) : D(v)\,dx \quad u,v \in H^1(\Omega)^2$$

$$b(v,p) = -\int_\Omega \nabla \cdot v\,p\,dx \quad v \in H^1(\Omega)^2,\ p \in L^2(\Omega)$$

Find $(u,p) \in V(g) \times Q$ s.t.

$$\begin{aligned} a(u,v) + b(v,p) &= (f,v) \quad \forall v \in V, \\ b(u,q) &= 0 \quad \forall q \in Q. \end{aligned}$$

Lemma (Gauss-Green's formula)

$u, v \in H^1(\Omega)$, n *: outer normal to* $\partial\Omega$

$$\int_\Omega (\partial_i u) v \, dx = -\int_\Omega u \partial_i v \, dx + \int_{\partial\Omega} u \, n_i v \, ds \, .$$

$$-2 \int_\Omega (\nabla \cdot D(u)) \cdot v \, dx =$$

$$-2 \int_\Omega \sum_i \sum_j \partial_j \tfrac{1}{2}(\partial_i u_j + \partial_j u_i) v_i \, dx = \int_\Omega \sum_{i,j} (\partial_i u_j + \partial_j u_i) \partial_j v_i \, dx$$

$$- \int_{\partial\Omega} \sum_{i,j} (\partial_i u_j + \partial_j u_i) n_j v_i \, ds$$

$$= \int_\Omega 2D(u) : D(v) \, dx - \int_{\partial\Omega} 2D(u) \, n \cdot v \, ds$$

from the symmetry of $D(u)$
$\sum_{i,j} (\partial_i u_j + \partial_j u_i) \partial_j v_i = \sum_{i,j} (\partial_i u_j + \partial_j u_i)(\partial_j v_i + \partial_i v_j)/2 = 2D(u):D(v)$.

Stokes equations and a weak formulation : 3/3

$$\int_\Omega \sum_i (\partial_i p)\, v_i\, dx = -\int_\Omega \sum_i p \partial_i v_i\, dx + \int_{\partial\Omega} \sum_i p\, n_i v_i$$
$$= -\int_\Omega p \nabla \cdot v + \int_{\partial\Omega} p\, n \cdot v$$

On the boundary $\partial\Omega$,

$$\int_{\partial\Omega} (2D(u)n - n\, p) \cdot v\, ds = 0 \quad v \in V \Rightarrow v = 0 \text{ on } \partial\Omega.$$

Remark

compatibility condition on Dirichlet data :

$$0 = \int_\Omega \nabla \cdot u = -\int_\Omega u \cdot \nabla 1 + \int_{\partial\Omega} u \cdot n\, ds = \int_{\partial\Omega} g \cdot n\, ds.$$

Remark

$-2[\nabla \cdot D(u)]_i = -\sum_j \partial_j (\partial_i u_j + \partial_j u_i) = -\sum_j \partial_j^2 u_i = -[\triangle u]_i.$

existence of a solution of the Stokes equations

Find $(u,p) \in V(g) \times Q$ s.t.

$$a(u,v) + b(v,p) = (f,v) \quad \forall v \in V,$$
$$b(u,q) = 0 \quad \forall q \in Q.$$

- coercivity : $\exists \alpha_0 > 0 \quad a(u,u) \geq \alpha_0 ||u||_1^2 \;\; \forall u \in V$.
- inf-sup condition :
  $\exists \beta_0 > 0 \quad \sup_{v \in V, v \neq 0} \dfrac{b(v,q)}{||v||_1} \geq \beta_0 ||q||_0 \;\; \forall\, q \in Q$.

bilinear form : $A(u,p\,;\,v,q) = a(u,v) + b(v,p) + b(u,q)$

Lemma

$\exists \alpha > 0 \;\; \sup_{(u,p) \in V \times Q} \dfrac{A(u,p\,;\,v,q)}{||(u,p)||_{V \times Q}} \geq \alpha ||(v,q)||_{V \times Q} \;\; \forall\, (v,q) \in V \times Q.$

Here, $||(u,p)||_{V \times Q}^2 = ||u||_1^2 + ||p||_0^2$.

Find $(u,p) \in V(g) \times Q$ s.t.

$$A(u,p\,;\,v,q) = (f,v) \quad \forall (v,q) \in V \times Q.$$

mixed finite element method

$V_h \subset V$: P2 finite element
$Q_h \subset Q$: P1 finite element + $\int_\Omega p_h \, dx = 0$.

- coercivity : $\exists \alpha_0 > 0 \quad a(u_h, u_h) \geq \alpha_0 ||u_h||_1^2 \ \ \forall u_h \in V_h$.
- uniform inf-sup condition :

$$\exists \beta_0 > 0 \ \forall h > 0 \qquad \sup_{v_h \in V_h, v_h \neq 0} \frac{b(v_h, q_h)}{||v_h||_1} \geq \beta_0 ||q_h||_0 \ \ \forall q_h \in Q_0.$$

Lemma

$$\exists \alpha > 0 \quad \sup_{(u_h, p_h) \in V_h \times Q_h} \frac{A(u_h, p_h \,;\, v_h, q_h)}{||(u_h, p_h)||_{V \times Q}} \geq \alpha ||(v_h, q_h)||_{V \times Q} \qquad \forall \, (v_h, q)_h \in V_h \times Q_h.$$

Find $(u_h, p_h) \in V_h(g) \times Q_h$ s.t.

$$A(u_h, p_h \,;\, v_h, q_h) = (f, v_h) \quad \forall (v_h, q_h) \in V_h \times Q_h.$$

Lemma

$$||u - u_h||_1 + ||p - p_h||_0 \leq C(\inf_{v_h \in V} ||u - v_h||_1 + \inf_{q_h \in Q} ||p - q_h||_0)$$

FreeFem++ script to solve Stokes equations by P2/P1

Find $(u,p) \in V_h(g) \times Q_h$ s.t.

$a(u,v) + b(v,p) + b(u,q) - \epsilon \int_\Omega p\,q = (f,v) \quad \forall (v,q) \in V_h \times Q_h$.

example5.edp

```
fespace Vh(Th,P2),Qh(Th,P1);
func f1=5.0/8.0*pi*pi*sin(pi*x)*sin(pi*y/2.0)+2.0*x;
func f2=5.0/4.0*pi*pi*cos(pi*x)*cos(pi*y/2.0)+2.0*y;
func g1=sin(pi*x)*sin(pi*y/2.0)/2.0;
func g2=cos(pi*x)*cos(pi*y/2.0);
Vh u1,u2,v1,v2; Qh p,q;
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //
real epsln=1.0e-6;
solve stokes(u1,u2,p1, v1,v2,q1) =
int2d(Th)( 2.0*(dx(u1)*dx(v1)
   +2.0*d12(u1,u2)*d12(v1,v2)+dy(u2)*dy(v2))
   -p*dx(v1)-p*dy(v2)-dx(u1)*q-dy(u2)*q
   -p*q*epsln )  // penalization
-  int2d(Th)( f1 * v1 + f2 * v1 )
+  on(1,2,3,4,u1=g1,u2=g2);
real meanp=int2d(Th)(p)/Th.area; //area=int2d(Th)(1.0)
p = p - meanp;
plot([u1,u2],p,wait=1,value=true,coef=0.1);
```

stabilized (penalty type) finite element method

$V_h \subset V$: P1 finite element
$Q_h \subset Q$: P1 finite element + $\int_\Omega p_h \, dx = 0$.
Find $(u_h, p_h) \in V_h(g) \times Q_h$ s.t.

$$a(u_h, v_h) + b(v_h, p_h) = (f, v_h) \quad \forall v_h \in V_h,$$
$$b(u_h, q_h) - \delta d(p_h, q_h) = 0 \quad \forall q_h \in Q_h.$$

$\delta > 0$: stability parameter, $d(p_h, q_h) = \sum_{K \in \mathcal{T}} h_K^2 \int_K \nabla p_h \cdot \nabla q_h \, dx$.
$|p_h|_h^2 = d(p_h, p_h)$: mesh dependent norm on Q_h.

- uniform weak inf-sup condition : Franca-Stenberg [1991]
  $\exists \beta_0, \, \beta_1 > 0 \; \forall h > 0 \; \sup_{v_h \in V_h} \frac{b(v_h, q_h)}{||v_h||_1} \geq \beta_0 ||q_h||_0 - \beta_1 |q_h|_h \; \forall \, q_h \in Q_0$.

Lemma

$$\exists \alpha > 0 \quad \sup_{(u_h, p_h) \in V_h \times Q_h} \frac{A(u_h, p_h \,;\, v_h, q_h)}{||(u_h, p_h)||_{V \times Q}} \geq \alpha ||(v_h, q_h)||_{V \times Q} \quad \forall \, (v_h, q)_h \in V_h \times Q_h.$$

FreeFem++ script to solve Stokes eqs. by P1/P1 stabilized

Find $(u,p) \in V_h(g) \times Q_h$ s.t.
$a(u,v)+b(v,p)+b(u,q)-\delta d(p,q)-\epsilon \int_\Omega p\, q = (f,v)\ \forall (v,q) \in V_h \times Q_h$.

▸ example6.edp

```
fespace Vh(Th,P1),Qh(Th,P1);
....
Vh u1,u2,v1,v2;
Qh p,q;
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //
real delta=0.01;
real epsln=1.0e-6;
solve stokes(u1,u2,p1, v1,v2,q1) =
int2d(Th)( 2.0*(dx(u1)*dx(v1)
   +2.0*d12(u1,u2)*d12(v1,v2)+dy(u2)*dy(v2))
   -p*dx(v1)-p*dy(v2)-dx(u1)*q-dy(u2)*q
   -delta*hTriangle*hTriangle*  // stabilization
          (dx(p)*dx(q)+dy(p)*dy(q))
   -p*q*epsln )                 // penalization
-  int2d(Th)( f1 * v1 + f2 * v1 )
+  on(1,2,3,4,u1=g1,u2=g2);
...
```

matrix formulation of discretized form : homogeneous Dirichlet

Find $(u_h, p_h) \in V_h \times Q_h$ s.t.

$$a(u_h, v_h) + b(v_h, p_h) = (f, v_h) \quad \forall v_h \in V_h,$$
$$b(u_h, q_h) = 0 \quad \forall q_h \in Q_h.$$

finite element bases, span$[\{\phi_i\}] = V_h$, span$[\{\psi_\mu\}] = S_h$.

$$[A]_{i\,j} = a(\phi_j, \phi_i) \qquad K \begin{bmatrix} \vec{u} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} \vec{f} \\ \vec{0} \end{bmatrix}$$
$$[B]_{\mu\,j} = b(\phi_j, \psi_\mu)$$

$K \in \mathbb{R}^{(N_V+N_S)\times(N_V+N_S)}$: symmetric, indefinite, Ker$K = \begin{bmatrix} \vec{0} \\ \vec{1} \end{bmatrix}$.

$B \in \mathbb{R}^{N_X \times N_S}$: on the whole FE nodes of velocity/pressure

$$\begin{aligned}[B^T\vec{1}]_i &= \textstyle\sum_\mu b(\phi_i, \psi_\mu) = b(\phi_i, \sum_\mu \psi_\mu) \\ &= b(\phi_i, 1) = -\int_\Omega \nabla\cdot\phi_i\, 1 = \int_\Omega \phi_i \cdot \nabla 1 - \int_{\partial\Omega} \phi_i \cdot n\, ds \\ &= 0 \ \text{ for } i \in \{1, \dots, N_X\} \setminus \Lambda_D.\end{aligned}$$

$b(\cdot,\cdot)$ satisfies inf-sup condition on $V_h \times S_h \Leftrightarrow$ Ker$B^T = \{\vec{1}\}$.

how to solve linear system of indefinite matrix

$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}$: symmetric, indefinite, singular :

$\#\{\lambda > 0\} = N_V$, $\#\{\lambda = 0\} = 1$, $\#\{\lambda < 0\} = N_S - 1$.

- penalization + direct factorization : UMFPACK

  $\begin{bmatrix} A & B^T \\ B & -\epsilon M \end{bmatrix}$: symmetric, indefinite, nonsingular :

  $\#\{\lambda > 0\} = N_V$, $\#\{\lambda < 0\} = N_S$.

  $[M]_{\mu\,\nu} = \int_\Omega \psi_\nu \psi_\mu dx$, $\epsilon > 0$: penalization parameter.

- preconditioned CG method with orthogonal projection

  Schur complement on pressure (aka Uzawa method)

$$-BA^{-1}B^T\vec{p} = -BA^{-1}\vec{f}$$

  $BA^{-1}B^T$: sym. positive definite on $\{\vec{q} \in \mathbb{R}^{N_S}\,;\,(\vec{q}, \vec{1}) = 0\}$.

  orthogonal projection $P : \mathbb{R}^{N_S} \to \text{span}[\{\vec{1}\}]^\perp$,

  $P\,\vec{q} = \vec{q} - (\vec{q}, \vec{1})/(\vec{1}, \vec{1})\vec{1}$ $\qquad [\vec{q}]_i = [\vec{q}]_i - \sum_{1 \leq j \leq n} [\vec{q}]_j / n$.

  preconditioner $[M]_{\mu\,\nu} = \int_\Omega \psi_\nu \psi_\mu dx$.

Uzawa-CG

FreeFem++ script to generate Stokes matrix

Find $(u,p) \in V_h(g) \times Q_h$ s.t.

$a(u,v) + b(v,p) + b(u,q) - \epsilon \int_\Omega p\, q = (f,v) \quad \forall (v,q) \in V_h \times Q_h.$

▸ example7.edp

```
fespace VQh(Th,[P2,P2,P1]);
... // func f1,f2,g1,g2 etc
Vh u1,u2,v1,v2; Qh p,q;
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //
real epsln=1.0e-6;
varf stokes([u1,u2,p], [v1,v2,q]) =
   int2d(Th)( 2.0*(dx(u1)*dx(v1)
     +2.0*d12(u1,u2)*d12(v1,v1)+dy(u2)*dy(v2))
     -p*dx(v1)-p*dy(v2)-dx(u1)*q-dy(u2)*q
     -p*q*epsln )  // penalization
   + on(1,2,3,4,u1=1.0,u2=1.0);
varf external([u1,u2,p],[v1,v2,q])=
    int2d(Th)(f1 * v1 + f2 *v2)
   + on(1,2,3,4,u1=g1,u2=g2); // Dirichlet data here
matrix A = stokes(VQh,VQh,solver=UMFPACK);
real[int] bc = stokes(0, VQh);
real[int] ff = external(0, VQh);
u1[] = A^-1 * ff;
```

FreeFem++ script to solve Stokes matrix by `Dissection`

Find $(u,p) \in V_h(g) \times Q_h$ s.t.

$a(u,v) + b(v,p) + b(u,q) = (f,v) \quad \forall (v,q) \in V_h \times Q_h$.

```
load "Dissection";      // loading dynamic module
defaulttoDissection(); // sparsesolver=Dissection
fespace VQh(Th,[P2,P2,P1]);
Vh u1,u2,v1,v2; Qh p,q;
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //
real epsln=1.0e-6;
varf stokes([u1,u2,p], [v1,v2,q]) =
   int2d(Th)( 2.0*(dx(u1)*dx(v1)
     +2.0*d12(u1,u2)*d12(v1,v2)+dy(u2)*dy(v2))
     -p*dx(v1)-p*dy(v2)-dx(u1)*q-dy(u2)*q)
   + on(1,2,3,4,u1=1.0,u2=1.0); // no penalty term
varf external([u1,u2,p],[v1,v2,q])=p
    int2d(Th)(f1 * v1 + f2 *v2)
    + on(1,2,3,4,u1=g1,u2=g2);  // Dirichlet data here
matrix A=stokes(VQh,VQh,solver=sparsesolver,
        strategy=2,tolpivot=1.0e-2); //new parameters
real[int] bc = stokes(0, VQh);
real[int] ff = external(0, VQh);
u1[] = A^-1 * ff;
```

stationary Navier-Stokes equations and a weak formulation

$\Omega = (0,1)\times(0,1)$

$$\begin{aligned} -2\nu\nabla\cdot D(u) + u\cdot\nabla u + \nabla p &= f \text{ in } \Omega \\ \nabla\cdot u &= 0 \text{ in } \Omega \\ u &= g \text{ on } \partial\Omega \end{aligned}$$

- $V(g) = \{v \in H^1(\Omega)^2\,;\, v = g \text{ on } \partial\Omega\},\ \ V = V(0)$
- $Q = L^2_0(\Omega) = \{p \in L^2(\Omega)\,;\, \int_\Omega p\,dx = 0\}$

outflow

bi/tri-linear forms and weak formulation :

$$a(u,v) = \int_\Omega 2\nu D(u) : D(v)\,dx \quad u,v \in H^1(\Omega)^2$$

$$a_1(u,v,w) = \frac{1}{2}\left(\int_\Omega (u\cdot\nabla v)\cdot w - (u\cdot\nabla w)\cdot v\,dx\right) \ \ u,v,w \in H^1(\Omega)^2$$

$$b(v,p) = -\int_\Omega \nabla\cdot v\,p\,dx \quad v \in H^1(\Omega)^2,\ p \in L^2(\Omega)$$

Find $(u,p) \in V(g)\times Q$ s.t.

$$\begin{aligned} a(u,v) + a_1(u,u,v) + b(v,p) &= (f,v) \quad \forall v \in V, \\ b(u,q) &= 0 \quad \forall q \in Q. \end{aligned}$$

trilinear form for the nonlinear term (Temam's trick)

$\nabla \cdot u = 0$, $w \in H_0^1(\Omega)$ or $u \cdot n = 0$ on $\partial\Omega$ $\Rightarrow$

$$a_1(u, v, w) = \int_\Omega (u \cdot \nabla v) \cdot w \, dx = \frac{1}{2} \left(\int_\Omega (u \cdot \nabla v) \cdot w \, - \, (u \cdot \nabla w) \cdot v \, dx \right).$$

$$\begin{aligned}
\int_\Omega (u \cdot \nabla) v \cdot w \, dx &= \int_\Omega \textstyle\sum_i \sum_j u_j (\partial_j v_i) \, w_i \, dx \\
&= -\int_\Omega \textstyle\sum_{i,j} v_i \partial_j (u_j \, w_i) \, dx + \int_{\partial\Omega} \sum_{i,j} v_i n_j u_j \, w_i \, ds \\
&= -\int_\Omega \textstyle\sum_{i,j} v_i (\partial_j u_j) \, w_i \, dx - \int_\Omega \sum_{i,j} v_i u_j \partial_j \, w_i \, dx \\
&= -\int_\Omega \textstyle\sum_{i,j} u_j (\partial_j w_i) \, v_i \, dx \\
&= -\int_\Omega (u \cdot \nabla) w \cdot v \, dx \, .
\end{aligned}$$

$a_1(u, u, u) = 0 \Rightarrow$ corecivity : $a(u, u) + a_1(u, u, u) \geq \alpha ||u||^2$.

nonlinear system of the stationary solution

$$A(u,p\,;\,v,q) = a(u,v) + a_1(u,u,v) + b(v,p) + b(u,q)$$

nonlinear problem:

Find $(u,p)\in V(g)\times Q$ s.t. $A(u,p\,;\,v,q) = (f,v)\ \forall(v,q)\in V\times Q$.

$a_1(\cdot,\cdot,\cdot)$: trilinear form,

$$\begin{aligned} a_1(u+\delta u, u+\delta u, v) &= a_1(u, u+\delta u, v) + a_1(\delta u, u+\delta u, v) \\ &= a_1(u,u,v) + a_1(u,\delta u,v) + a_1(\delta u,u,v) + a_1(\delta u,\delta u,v) \end{aligned}$$

$$\begin{aligned} &A(u+\delta u, p+\delta p\,;\,v,q) - A(u,p\,;\,v,q) \\ &= a(u+\delta u,v) - a(u,v) \\ &\quad + b(v,p+\delta p) - b(v,p) + b(u+\delta u,q) - b(u,q) \\ &\quad + a_1(u+\delta u,u+\delta u,v) - a_1(u,u,v) \\ &= a(\delta u,v) + b(v,\delta p) + b(\delta u,q) + a_1(\delta u,u,v) + a_1(u,\delta u,v) + O(||\delta u||^2) \end{aligned}$$

Find $(\delta u,\delta p)\in V\times Q$ s.t.

$$\begin{aligned} a(\delta u,v) + b(v,\delta p) + b(\delta u,q) + a_1(\delta u,u,v) + a_1(u,\delta u,v) = \\ - A(u,p\,;\,v,q) \quad \forall(v,q)\in V\times Q \end{aligned}$$

Newton iteration

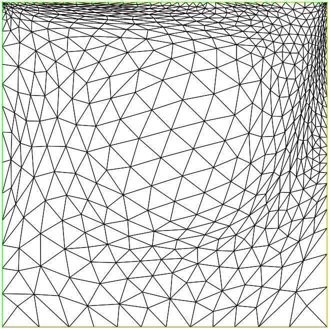
$(u_0, p_0) \in V(g) \times Q$
loop $n = 0, 1 \ldots$
Find $(\delta u, \delta p) \in V \times Q$ s.t.
$$a(\delta u, v) + b(v, \delta p) + b(\delta u, q) + a_1(\delta u, u_n, v) + a_1(u_n, \delta u, v) = A(u_n, p_n\,;\, v, q) \quad \forall (v, q) \in V \times Q$$
if $||(\delta u, \delta p)||_{V \times Q} \leq \varepsilon$ then break
$u_{n+1} = u_n - \delta u$,
$p_{n+1} = p_n - \delta p$.
loop end.

▸ example8.edp

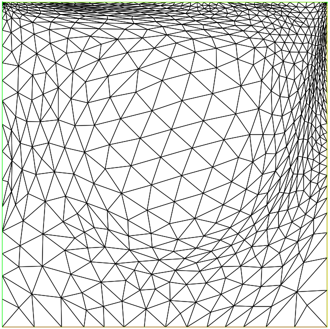
$(u^{(0)}, p^{(0)}) \in V(g) \times Q$: solution of the Stokes eqs., $\nu = 1$.
while ($\nu > \nu_{\text{min}}$)
Newton iteration $(u^{(k+1)}, p^{(k+1)}) \in V(g) \times Q$ from $(u^{(k)}, p^{(k)})$.
$\nu = \nu/2$, k++.
while end.
initial guess from the stationary state of lower Reynolds number

mesh adaptation

```
fespace XXMh(Th,[P2,P2,P1]);
XXMh [u1,u2,p];
real lerr=0.01;
Th=adaptmesh(Th, [u1,u2], p, err=lerr, nbvx=100000);
[u1,u2,p]=[u1,u2,p]; // interpolation on the new mesh
```



$\Rightarrow$



`err` : P_1 interpolation error level

`nbvx` : maximum number of vertexes to be generated.

stream line for visualization of 2D flow : 1/2

stream function $\psi : \Omega \to \mathbb{R}$, $u = \begin{bmatrix} \partial_2 \psi \\ -\partial_1 \psi \end{bmatrix} \quad \Leftrightarrow \quad u \perp \nabla \psi$.

$$-\nabla^2 \psi = \nabla \times u = \partial_1 u_2 - \partial_2 u_1 \quad \text{in } \Omega$$

boundary conditions for the stream line:

$$0 = \int_0^x u_2(t,0)dt = \int_0^x -\partial_1 \psi(t,0)dt = -\psi(x,0) + \psi(0,0)$$
$$\psi(x,0) = \psi(0,0).$$
$$0 = \int_0^y u_1(t,0)dt = \int_0^y \partial_2 \psi(0,t)dt = \psi(0,y) + \psi(0,0)$$
$$\psi(0,y) = \psi(0,0).$$
$$0 = \int_0^x u_2(t,1)dt = \int_0^x -\partial_1 \psi(t,1)dt = -\psi(x,1) + \psi(0,1)$$
$$\psi(x,1) = \psi(0,1) = \psi(0,0).$$

$\Rightarrow \qquad \psi = 0$ on $\partial\Omega$.

stream line for visualization of 2D flow : 2/2

$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in$ P2 finite element space $\quad \Rightarrow (\partial_1 u_2 - \partial_2 u_1) \in$ P1.

```
fespace Xh(Th,P2);
fespace Mh(Th,P1);
Xh u1, u2;       // computed from Navier-Stokes solver
Mh psi,phi;      // dy(u1),dx(u2) are polynomials of 1st
solve streamlines(psi,phi,solver=UMFPACK) =
      int2d(Th)( dx(psi)*dx(phi) + dy(psi)*dy(phi) )
   +  int2d(Th)( (dx(u2)-dy(u1))*phi )
   +  on(1,2,3,4,psi=0);
plot(psi,nbiso=30);
```

Application of finite element method to fluid problems

Time-dependent Navier-Stokes equations

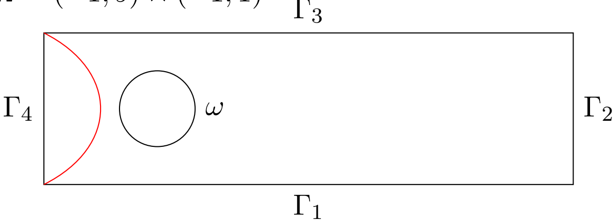
- material derivative is approximated by Characteristic Galerkin method
- functional space of pressure depends on boundary conditions of flow, e.g., inflow, non-slip, slip, and outflow.

Thermal convection problem by Rayleigh-Bénard equations

- time-dependent problems for fluid and temperature by convection are solved by Characteristic Galerkin method
- stationary solution is obtained by Newton iteration using an initial value obtained from time-evolutionary solution

incompressible flow around a cylinder : boundary conditions

$\Omega = (-1, 9) \times (-1, 1)$



$$\frac{\partial u}{\partial t} + u \cdot \nabla u - 2\nu \nabla \cdot D(u) + \nabla p = 0 \text{ in } \Omega$$

$$\nabla \cdot u = 0 \text{ in } \Omega$$

$$u = g \text{ on } \partial\Omega$$

boundary conditions:

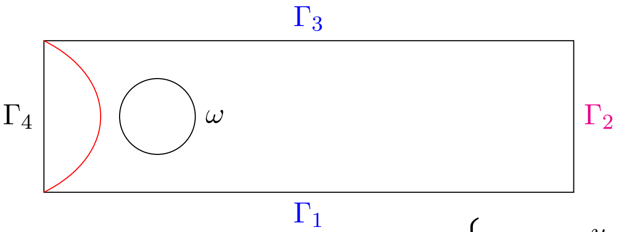
Poiseuille flow on Γ_4 : $u = (1 - y^2, 0)$.

slip boundary condition on $\Gamma_1 \cup \Gamma_3$: $\begin{cases} u \cdot n = 0 \\ (2\nu D(u)n - np) \cdot t = 0 \end{cases}$

no-slip boundary condition on ω : $u = 0$

outflow boundary condition on Γ_2 : $2\nu D(u)n - np = 0$

slip boundary conditions and function space



slip boundary condition on $\Gamma_1 \cup \Gamma_3$: $\begin{cases} u \cdot n = 0 \\ (2\nu\, D(u)n - n\, p) \cdot t = 0 \end{cases}$

- $V(g) = \{v \in H^1(\Omega)^2\,;\ v = g$ on $\Gamma_4 \cup \omega,\ v \cdot n = 0$ on $\Gamma_1 \cup \Gamma_3\}$,
- $Q = L^2(\Omega)$.

non-slip

$$\int_{\Gamma_1\cup\Gamma_3} (2\nu D(u)n - n\,p) \cdot v ds = \int_{\Gamma_1\cup\Gamma_3} (2\nu D(u)n - n\,p) \cdot (v_n n + v_t t) ds$$
$$= \int_{\Gamma_1\cup\Gamma_3} (2\nu D(u)n - n\,p) \cdot (v \cdot n) n\, ds$$
$$+ \int_{\Gamma_1\cup\Gamma_3} (2\nu D(u)n - n\,p) \cdot t v_t\, ds = 0$$

characteristic line and material derivative

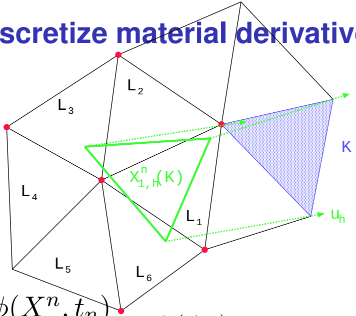
$u(x_1, x_2, t) : \Omega \times (0, T] \to \mathbb{R}^2$, given velocity field.
$\phi(x_1, x_2, t) : \Omega \times (0, T] \to \mathbb{R}$.
$X(t) : (0, T] \to \mathbb{R}^2$, characteristic curve :
$\frac{dX}{dt}(t) = u(X(t), t), X(0) = X_0$

$$
\begin{aligned}
\frac{d}{dt}\phi(X(t),t) &= \nabla\phi(X(t),t) \cdot \frac{d}{dt}X(t) + \frac{\partial}{\partial t}\phi(X(t),t) \\
&= \nabla\phi(X(t),t) \cdot u(X(t),t) + \frac{\partial}{\partial t}\phi(X(t),t)
\end{aligned}
$$

material derivative : $\frac{D\phi}{Dt} = \frac{\partial}{\partial t}\phi + u \cdot \nabla\phi$.

approximation by difference
$\frac{D\phi(X(t),t)}{Dt} \sim \frac{\phi(X(t),t) - \phi(X(t-\Delta t), t-\Delta t)}{\Delta t}$

characteristic Galerkin method to descretize material derivative



approximation by Euler method :

$t_n < t_{n+1}$, $t_{n+1} = \Delta t + t_n$.

$$X(t_{n+1}) = x$$

$$X(t_n) = X^n(x) + O(\Delta t^2)$$

$$X^n(x) = x - u(x, t_n)$$

$$\frac{D\phi(X(t_{n+1}), t_{n+1})}{Dt} = \frac{\phi(x, t_{n+1}) - \phi(X^n, t_n)}{\Delta t} + O(\Delta t)$$

$$\sim \frac{\phi^{n+1} - \phi^n \circ X^n}{\Delta t}.$$

u^n : obtained in the previous time step.

Find $(u^{n+1}, p^{n+1}) \in V(g) \times Q$ s.t.

$$\left(\frac{u^{n+1} - u^n \circ X^n}{\Delta t}, v\right) + a(u^{n+1}, v) + b(v, p^{n+1}) = 0 \quad \forall v \in V,$$

$$b(u^{n+1}, q) = 0 \quad \forall q \in Q.$$

FreeFem++ script using characteristic Galerkin method

FreeFem++ provides convect to compute $(u^n \circ X^n, \cdot)$.

▸ example9.edp

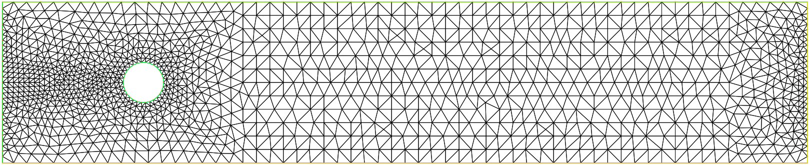
```
real nu=1.0/Re;
real alpha=1.0/dt;
int i;
problem NS([u1,u2,p],[v1,v2,q],solver=UMFPACK,init=i) =
  int2d(Th)(alpha*(u1*v1 + u2*v2)
       +2.0*nu*(dx(u1)*dx(v1)+2.0*d12(u1,u2)*d12(v1,v2)
               +dy(u2)*dy(v2))
       - p * div(v1, v2) - q * div(u1, u2))
- int2d(Th)(alpha*( convect([up1,up2],-dt,up1)*v1
                   +convect([up1,up2],-dt,up2)*v2) )
+ on(1,3,u2=0)+on(4,u1=1.0-y*y,u2=0)+on(5,u1=0,u2=0);

for (i = 0; i <= timestepmax; i++) {
   up1 = u1; up2 = u2; pp = p;
   NS;              // factorization is called when i=0
   plot([up1,up2],pp,wait=0,value=true,coef=0.1);
}
```

FreeFem++ script for mesh generation around a cylinder

Delaunay triangulation from nodes given on the boundary boundary segments are oriented and should be connected.

```
int n1 = 30;
int n2 = 60;
border ba(t=0,1.0){x=t*10.0-1.0;y=-1.0;label=1;};
border bb(t=0,1.0){x=9.0;y=2.0*t-1.0;label=2;};
border bc(t=0,1.0){x=9.0-10.0*t;y=1.0;label=3;};
border bd(t=0,1.0){x=-1.0;y=1.0-2.0*t;label=4;};
border cc(t=0,2*pi){x=cos(t)*0.25+0.75;
                    y=sin(t)*0.25;label=5;};
mesh Th=buildmesh(ba(n2)+bb(n1)+bc(n2)+bd(n1)+cc(-n1));
plot(Th);
```



stream line for visualization of flow around a cylinder : 1/2

stream function $\psi : \Omega \to \mathbb{R}$, $u = \begin{bmatrix} \partial_2 \psi \\ -\partial_1 \psi \end{bmatrix}$.

boudnary conditions for the stream line:

inlet: $y - \frac{y^3}{3} = \int_0^y u_1(x_1, t) dt = \int_0^y \partial_2 \psi(x_1, t) dt = \psi(x_1, y) - \psi(x_1, 0)$

$$\psi(x_1, y) = \psi(x_1, 0) + y - \frac{y^3}{3} = y - \frac{y^3}{3}.$$

slip: $0 = \int_{x_1}^x u_2(t, \pm 1) dt = \int_{x_1}^x -\partial_1 \psi(t, \pm 1) dt = \psi(x_1, \pm 1) - \psi(x, \pm 1)$

$$\psi(x, \pm 1) = \psi(x_1, \pm 1) = \psi(x_1, 0) \pm 2/3 = \pm 2/3.$$

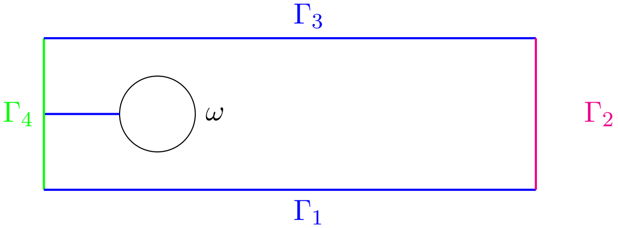
cylinder: $0 = \int_{-\pi}^{\theta} u \cdot n \, d\theta = \int_{-\pi}^{\theta} -\partial_1 \psi \, r \sin\theta + \partial_2 \psi \, r \cos\theta$

$$= \int_{-\pi}^{\theta} \frac{\partial}{\partial \theta} \psi(r, \theta) d\theta.$$

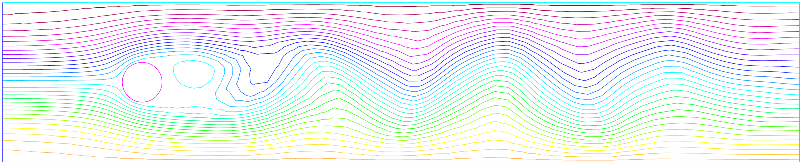
center from inlet: $u_2 = 0 \;\Rightarrow\;$ same as slip wall ,

$$\psi|_{\omega} = \psi(x_1, 0) = 0.$$

stream line for visualization of flow around a cylinder : 2/2



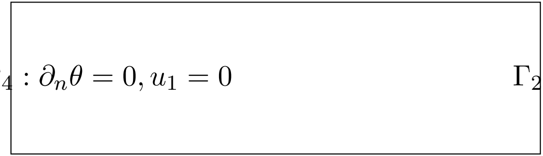
slip boundary condition on $\Gamma_1 \cup \Gamma_3$, outflow on Γ_2.



thermal convection in a box : 1/2

$$\Gamma_3 : \theta = \theta_0, u_2 = 0$$

$$\Gamma_4 : \partial_n \theta = 0, u_1 = 0 \qquad \Gamma_2 : \partial_n \theta = 0, u_1 = 0$$



$$\Gamma_1 : \theta = \theta_0 + \Delta\theta, u_2 = 0$$

Rayleigh-Bénard equations

$$\rho\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) - 2\nabla \cdot \mu_0 D(u) + \nabla p = -\rho\, g \vec{e}_2 \text{ in } \Omega\,,$$

$$\nabla \cdot u = 0 \text{ in } \Omega\,,$$

$$\frac{\partial \theta}{\partial t} + u \cdot \nabla \theta - \nabla \cdot (\kappa \theta) = 0 \text{ in } \Omega\,.$$

$\vec{e}_2$: unit normal of y-direction
d : height of the box, g : gravity acceleration,
κ : thermal diffusivity, μ_0 : viscosity

thermal convection in a box : 2/2

Boussinesq approximation : $\rho = \rho_0\{1 - \alpha(\theta - \theta_0)\}$, $\theta_0 = 0$.
ρ_0: representative density, α : thermal expansion coefficient.
non-dimensional Rayleigh-Bénard equations

$$\frac{1}{Pr}\left(\frac{\partial u}{\partial t} + u \cdot \nabla u\right) - 2\nabla \cdot D(u) + \nabla p = Ra\theta\vec{e}_2 \text{ in } \Omega\,,$$

$$\nabla \cdot u = 0 \text{ in } \Omega\,,$$

$$\frac{\partial \theta}{\partial t} + u \cdot \nabla\theta - \triangle\theta = 0 \text{ in } \Omega$$

$$u \cdot n = 0 \text{ on } \partial\Omega\,,$$

$$\theta = 1 \text{ on } \Gamma_1\,,$$

$$\theta = 0 \text{ on } \Gamma_3\,,$$

$$\partial_n\theta = 0 \text{ on } \Gamma_2 \cup \Gamma_4\,.$$

- $Pr = \dfrac{\mu_0}{\kappa\rho_0}$: Prandtl number,
- $Ra = \dfrac{\rho_0 g\alpha\Delta\theta d^3}{\kappa\mu_0}$: Rayleigh number.

a weak form to solve time-dependent Rayleigh-Bénard eqs.

- velocity : $V = \{v \in H^1(\Omega)^2\,;\, v \cdot n = 0$ on $\partial\Omega\}$,
- pressure : $Q = L^2_0(\Omega) = \{p \in L^2(\Omega)\,;\, \int_\Omega p\,dx = 0\}$,
- temperature : $\Psi_D = \{\theta \in H^1(\Omega)\,;\, \theta = 1$ on $\Gamma_1, \theta = 0$ on $\Gamma_0\}$.

bilinear forms:

$a_0(u, v) = \int_\Omega 2D(u) : D(v)$, $\quad b(v, p) = -\int_\Omega \nabla \cdot v\, p$,

$c_0(\theta, \psi) = \int_\Omega \nabla\theta \cdot \nabla\psi$.

using Characteristic Galerkin method: ▸ example10.edp

$(u^n, \theta^n) \in V \times \Psi_D$: from previous time step

Find $(u^{n+1}, p^{n+1}, \theta^{n+1}) \in V \times Q \times \Psi_D$ s.t.

$$\frac{1}{Pr}\left(\frac{u^{n+1} - u^n \circ X^n}{\Delta t}, v\right) + a_0(u^{n+1}, v) + b(v, p^{n+1}) = Ra(\theta^n \vec{e}_2, v) \quad \forall v \in V,$$

$$b(u^{n+1}, q) = 0 \quad \forall q \in Q,$$

$$\left(\frac{\theta^{n+1} - \theta^n \circ X^n}{\Delta t}, \psi\right) + c_0(\theta^{n+1}, \psi) = 0 \quad \forall \psi \in \Psi_0.$$

a weak form to solve stationary Rayleigh-Bénard eqs.

trilinear forms and bilinear form for the Navier-Stokes eqs.

- $a_1(u,v,w) = \frac{1}{2\,Pr}\left(\int_\Omega (u\cdot\nabla v)\cdot w - (u\cdot\nabla w)\cdot v\right)$
- $c_1(u,\theta,\psi) = \frac{1}{2}\left(\int_\Omega (u\cdot\nabla\theta)\cdot\psi - (u\cdot\nabla\psi)\cdot\theta\right)$
- $A(u,p\,;\,v,q) = a_0(u,v) + a_1(u,u,v) + b(v,p) + b(u,q)$

Newton iteration $(u_0,p_0,\theta_0)\in V\times Q\times\Psi_D$ ▸ example11.edp

loop $n=0,1\ldots$

Find $(\delta u,\delta p,\delta\theta)\in V\times Q\times\Psi_0$ s.t.

$$a_0(\delta u,v) + b(v,\delta p) + b(\delta u,q) + a_1(\delta u,u_n,v) + a_1(u_n,\delta u,v)$$
$$-Ra(\delta\theta\vec{e}_2,v) = A(u_n,p_n\,;\,v,q) - Ra(\theta_n\vec{e}_2,v) \quad \forall (v,q)\in V\times Q$$
$$c_0(\delta\theta,\psi) + c_1(u_n,\delta\theta,\psi) + c_1(\delta u,\theta_n,\psi) = c_0(\theta_n,\psi) + c_1(u_n,\theta_n,\psi)$$
$$\forall\psi\in\Psi_0$$

if $||(\delta u,\delta p,\delta\theta)||_{V\times Q\times\Psi}\leq\varepsilon$ then break

$u_{n+1}=u_n-\delta u,\quad p_{n+1}=p_n-\delta p,\quad \theta_{n+1}=\theta_n-\delta\theta$.

loop end.

initial data $\Leftarrow$ stationary solution by time-dependent problem.

Details on iterative linear solver

FreeFem++ provides iterative solvers for symmetric positive definite matrix and general unsymmetric invertible matrix.

- Conjugate Gradient method `LinearCG`
  `src/femlib/MatriceCreuse.hpp::ConjuguedGradient2()`
- Generalized Minimal Residual method `LinearGMRES`
  `src/femlib/gmres.hpp::GMRES()`

They are useful to get solution with less memory consumption.

- treatment of boundary condition is somewhat different from penalization technique for direct solver
- definition of SpMV (Sparse matrix vector multiplication) operator: $y = Ax$, and preconditioning operator by
  `func real[int] SpMV(real[int] &x);`

To use good preconditioner is very important for faster convergence.

- preconditioner for time-dependent generalized Stokes equations

conjugate gradient method

$$A_\tau \vec{u} = \vec{f}_\tau, \qquad [A_\tau]_{k\,k} = \tau,\ [f_\tau]_k = \tau\, g_k \text{ for } k \in \Lambda_D.$$

preconditioner $Q \sim A_\tau^{-1}$

Krylov subsp. :

$K_n(Q\vec{r}^{\,0}, QA_\tau) = \text{span}[Q\vec{r}^{\,0}, QA_\tau Q\vec{r}^{\,0}, \ldots, (QA_\tau)^n Q\vec{r}^{\,0}]$

Find $\vec{u}^{\,n} \in K_n(Q\vec{r}^{\,0}, QA_\tau) + \vec{u}^{\,0}$ s.t.

$$(A\vec{u}^{\,n} - \vec{f}_\tau, \vec{v}) = 0 \quad \vec{v} \in K_n(Q\vec{r}^{\,0}, QA_\tau).$$

Preconditioned CG method

$\vec{u}^{\,0}$: initial step for CG.

$\vec{r}^{\,0} = \vec{f}_\tau - A_\tau \vec{u}^{\,0}$

$\vec{p}^{\,0} = Q\vec{r}^{\,0}$.

loop $n = 0, 1, \ldots$

$\alpha_n = (Q\vec{r}^{\,n}, \vec{r}^{\,n})/(A_\tau \vec{p}^{\,n}, \vec{p}^{\,n})$,

$\vec{u}^{\,n+1} = \vec{u}^{\,n} + \alpha_n \vec{p}^{\,n}$,

$\vec{r}^{\,n+1} = \vec{r}^{\,n} - \alpha_n A_\tau \vec{p}^{\,n}$,

if $||\vec{r}^{\,n+1}|| < \epsilon$ exit loop.

$\beta_n = (Q\vec{r}^{\,n+1}, \vec{r}^{\,n+1})/(Q\vec{r}^{\,n}, \vec{r}^{\,n})$,

$\vec{p}^{\,n+1} = Q\vec{r}^{\,n+1} + \beta_n \vec{p}^{\,n}$.

```
LinearCG(opA,u,f,precon=opQ,nbiter=100,eps=1.0e-10)
```

FreeFem++ script for CG, diagonal preconditioner

▸ example12.edp

```
Vh u,v;
varf aa(u,v)=int2d(Th)( dx(u)*dx(v)+dy(u)*dy(v) )
            +on(2,3,4,u=1.0);
varf external(u,v)=int2d(Th)(f*v)+int1d(Th,1)(h*v)
real tgv=1.0e+30;   matrix A;
real[int] bc = aa(0,Vh,tgv=tgv);
func real[int] opA(real[int] &pp) {//SpMV operation with
  pp = bc ? 0.0 : pp;               //homogeneous data
  real[int] qq=A*pp;
  pp = bc ? 0.0 : qq; return pp;}//qq: locally allocated
func real[int] opQ(real[int] &pp) {
  for (int i = 0; i < pp.n; i++)
     pp(i)=pp(i)/A(i,i);
  pp = bc ? 0.0 : pp; reutrn pp;  }
A=aa(Vh,Vh,tgv=tgv,solver=sparsesolver);
real[int] ff = external(0,Vh);
u = bc ? v[] : 0.0;   // v : Dirichlet data without tgv
v[] = A * u[]; ff -= v[]; // Dirichlet data goes to RHS
ff = bc ? 0.0 : ff;       // CG works in zero-Dirichlet
LinearCG(opA,u[],ff,precon=opQ,nbiter=100,eps=1.0e-10);
```

conjugate gradient method on the image space

$\vec{f} \in \mathrm{Im}A$, find $u \in \mathrm{Im}A$ $A\vec{u} = \vec{f}$.
preconditioner $Q : \mathrm{Im}A \to \mathrm{Im}A$, $Q \sim A|_{\mathrm{Im}A}^{-1}$
orthogonal projection $P : \mathbb{R}^n \to \mathrm{Im}\ A$.

Preconditioned CG method

$\vec{u}^{\,0}$: initial step for CG.
$\vec{r}^{\,0} = P(\vec{f} - A\vec{u}^{\,0})$
$\vec{p}^{\,0} = Q\vec{r}^{\,0}$.
loop $n = 0, 1, \ldots$

$\alpha_n = (Q\vec{r}^{\,n}, \vec{r}^{\,n})/(A\vec{p}^{\,n}, \vec{p}^{\,n})$,
$\vec{u}^{\,n+1} = P(\vec{u}^{\,n} + \alpha_n \vec{p}^{\,n})$,
$\vec{r}^{\,n+1} = P(\vec{r}^{\,n} - \alpha_n A\vec{p}^{\,n})$,
if $||\vec{r}^{\,n+1}|| < \epsilon$ exit loop.
$\beta_n = (Q\vec{r}^{\,n+1}, \vec{r}^{\,n+1})/(Q\vec{r}^{\,n}, \vec{r}^{\,n})$,
$\vec{p}^{\,n+1} = P(Q\vec{r}^{\,n+1} + \beta_n \vec{p}^{\,n})$.

P is used to avoid numerical round-off error which perturbs vectors from the image space
`LinearCG` cannot handle this safe operation. PQ and PA only.

FreeFem++ script for full-Neumann problem

▸ example13.edp

```
Vh u,v;
varf aa(u,v)=int2d(Th)( dx(u)*dx(v)+dy(u)*dy(v) );
varf external(u,v)=int2d(Th)(f*v);
matrix A;
func real[int] opA(real[int] &pp) {
  real[in] qq=A*pp;
  pp = qq; pp -= pp.sum / pp.n; // projection
  return pp;
}
func real[int] opQ(real[int] &pp) {
  for (int i = 0; i < pp.n; i++)
     pp(i)=pp(i)/A(i,i);
  pp -= pp.sum / pp.n;              // projection
  reutrn pp;
}
A=aa(Vh,Vh,solver=CG); real[int] ff = external(0,Vh);
ff -= ff.sum / ff.n;                // projection
u[]=0.0;  // initial step for CG
LinearCG(opA,u[],ff,precon=opQ,nbiter=100,eps=1.0e-10);
```

conjugate gradient in Uzawa method for Stokes eqs.

▸▸ Stokes Solver

$$\begin{bmatrix} A_\tau & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} \vec{f}_\tau \\ \vec{0} \end{bmatrix} \quad [A_\tau]_{k\,k} = \tau,\ [\vec{f}_\tau]_k = \tau g_k \text{ for } k \in \Lambda_D.$$

orthogonal projection $P : \mathbb{R}^{N_S} \to \mathrm{span}[\{\vec{1}\}]^\perp$, preconditioner Q

$(BA^{-1}B^T)^{-1} \sim I_h^{-1} = Q$: inverse of mass matrix.

Preconditioned CG method with projection

$\vec{p}^{\,0} = \vec{0}$: initial step for CG.

$\vec{g}^{\,0} = PBA_\tau^{-1}\vec{f}_\tau$,

$\vec{w}^{\,0} = PQ\vec{g}^{\,0}$.

loop $n = 0, 1, \ldots$

$$\begin{aligned}
\alpha_n &= (PQ\vec{g}^{\,n}, \vec{g}^{\,n})/(P\,BA_\tau^{-1}B^T\vec{w}^{\,n}, \vec{w}^{\,n}),\\
\vec{p}^{\,n+1} &= \vec{p}^{\,n} + \alpha_n\vec{w}^{\,n},\\
\vec{g}^{\,n+1} &= \vec{g}^{\,n} - \alpha_n(BA_\tau^{-1}B^T)\vec{w}^{\,n},\\
\beta_n &= (PQ\vec{g}^{\,n+1}, \vec{g}^{\,n+1})/(\vec{g}^{\,n}, \vec{g}^{\,n}),\\
\vec{w}^{\,n+1} &= PQ\vec{g}^{\,n+1} + \beta_n\vec{w}^{\,n}.
\end{aligned}$$

$\vec{u}^{\,n+1} = A_\tau^{-1}(\vec{f}_\tau - B^T\vec{p}^{\,n+1})$.

$A_\tau^{-1}\vec{f}_\tau \Leftrightarrow A_\tau\vec{u} = \vec{f}_\tau$ with $u_k = g_k,\ \ k \in \Lambda_D$ ▸▸ penalty

$A_\tau^{-1}B^T\vec{w} \Leftrightarrow A_\tau\vec{u} = B^T\vec{w}$ with $u_k = 0,\ \ k \in \Lambda_D$

FreeFem++ script for CG with Uzawa 1/2

▸ example14.edp

```
fespace Vh(Th,[P2,P2]),Qh(Th,P1);
... // func f1,f2,g1,g2 etc
Vh [u1,u2], [v1,v2], [bcsol1, bcsol2];
Qh p,q;
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //

varf a([u1,u2], [v1,v2) =
   int2d(Th)( 2.0*(dx(u1)*dx(v1)
     +2.0*d12(u1,u2)*d12(v1,v1)+dy(u2)*dy(v2))
   + on(1,2,3,4,u1=g1,u2=g2);
varf b([u1,u2], [q])= int2d(Th)(- q*(dx(u1)+dy(u2)));
varf external([u1,u2],[v1,v2])=
    int2d(Th)(f1 * v1 + f2 *v2);
varf massp(p, q)= int2d(Th)(p * q);
matrix A = a(Vh,Vh,solver=UMFPACK,init=true);
matrix B = b(Vh,Qh);
matrix Mp = massp(Qh,Qh,solver=UMFPACK,init=true);
real[int] bc = a(0, Vh);
real[int] ff = external(0, Vh);
```

FreeFem++ script for CG with Uzawa 2/2

```
func real[int] UzawaStokes(real[int] &pp) {
  real[int] b = B'*pp;
  real[int] uu = A^-1 * b;
  pp = B * uu;  pp -= pp.sum / pp.n;
  return pp;
}
func real[int] PreconMass(real[int] &pp) {
  real[int] ppp = Mp^-1 * pp;
  pp = ppp; pp -= pp.sum / pp.n;
  return pp;
}
p = 0.0;
ff += bc;          //bc keeps Dirichlet data with tgv
real[int] uu = A^-1 * ff;
q[] = B * uu;
LinearCG(UzawaStokes, p[], q[], precon=PreconMass,
         nbiter=100,eps=1.0e-10,verbosity=100);
ff = external(0, Vh); real[int] b = B'*p[];
ff -= b; ff += bc; //bc keeps Dirichlet data with tgv
u1[] = A^-1 * ff;      // to access [u1, u2]
```

Uzawa method with CG for generalized Stokes eqs. ▸ example15.edp

descretized Navier-Stokes equations by characteristic Galerkin
Δt : time step, ν : Reynolds number
Find $(u^{n+1}, p^{n+1}) \in V(g) \times Q$ s.t.

$$\left(\frac{u^{n+1}}{\Delta t}, v\right) + a(\nu\,;\, u^{n+1}, v) + b(v, p^{n+1}) = -\left(\frac{u^n \circ X^n}{\Delta t}, v\right) \quad \forall v \in V,$$

$$b(u^{n+1}, q) = 0 \qquad \forall q \in Q.$$

- I_v, I_p : mass matrix for velocity, pressure
- A_p : sitffness matrix of Laplacian for pressure with B.C.

$$\begin{bmatrix} \frac{1}{\Delta t} I_v + \nu A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} \vec{f} \\ \vec{0} \end{bmatrix}$$

Preconditioner by Cahouet-Chabard [1988]

$$\left(B(\frac{1}{\Delta t} I_v + \nu A)^{-1} B^T\right)^{-1} \sim \frac{1}{\Delta t} A_p^{-1} + \nu I_p^{-1}$$

Uzawa with CG : to compute large problem with less memory

syntax of FreeFem++ script

loops

```
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++;
}
```

finite element space, variational form, and matrix

```
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
```

procedure (function)

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

fundametal data types

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

array

```
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
```

There are other operations such as $\ell^1, \ell^2, \ell^\infty$-norms, max, min.
cf. Finite element analysis by mathematical programming language FreeFem++, Ohtsuka-Takaishi [2014].

FEM data

```
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
```

dense and sparse matrices

```
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=sparsesovler); //sparse matrix
```

file I/O is same as C++, `ofstream/ifstream` ▸ example{10,11}.edp

Compilation with configure : 1/2

- download the latest source from `http://www.freefem.org/ff++/`
- run `configure` scritp.
  `% ./configure --enable-m64 CXXFLAGS=-std=c++11 --enable-download`
  this enables automatic downloading of all sources including MUMPS etc.
- run make.
  `% make`
- binaries will be created in `src/nw`

GNU bison and flex are necessary for FreeFem+++ language parser.
OpenGL compatible libraries are also necessary for `ffglut` graphics viewer.
Other options to minimize the capability,
`--disable-superlu --disable-scotch --without-mpi`

Compilation with configure : 2/2

Fortran is mandatory for MUMPS linear solver.
Without Fortran compiler, by adding
`--disable-fortran --disable-mumps`
It is necessary to remove `mumps-seq` from `LIST_SOFT` of `download/Makefile` and
to remove `ffnewuoa.$(DYLIB_SUFFIX)` from `LIST_COMPILE_PKG` of `example++-load/Makefile` when Fortran is disabled.
BLAS library is automatically detected by `configure` and information is written in
`examples++-load/WHERE_LIBRARY-config`.
`$(INTEL_MKL)` is described as appropriate directory:

```
lapack LD -L$(INTEL_MKL)/lib/intel64 -lmkl_rt \
  -lmkl_sequential -lmkl_core  -liomp5 -lpthread
lapack INCLUDE -I$(INTEL_MKL)/include
mkl LD -L$(INTEL_MKL)/lib/intel64 -lmkl_rt \
  -lmkl_intel_thread -lmkl_core -liomp5 -lpthread
mkl INCLUDE -I$(INTEL_MKL)/include
blas LD -L$(INTEL_MKL)/mkl/lib/intel64 \
  -lmkl_rt -lmkl_sequential -lmkl_core -liomp5 -lpthread
```

References : 1/2

FreeFem++:

- F. Hecht, FreeFem++ manual, 3rd ed., 2015.
- K. Ohtsuka, T. Takaishi, Finite element analysis by mathematical programming language FreeFem++ (in Japanese), Industrial and Applied Mathematics Series Vol.4, Kyoritsu, 2014.
- I. Danaila, F. Hecht, O. Pironneau, Simulation numérique en C++, Dunod, 2003.

Finite element theory:

- D. Braess, Finite elements – Theory, fast solvers and application in solid mechanics, 3rd ed., Cambridge Univ. Press, 2007.
- A. Ern, J.-L. Guermond, Theory and pracitce of finite elements, Springer Verlag, New-York, 2004.
- M. Tabata, Numerical solution of partial differential equations II (in Japanese), Iwanami Shoten, 1994.

References : 2/2

specialized toipcs:

- T. A. Davis, Direct Methods for Sparse Linear Systems, SIAM, Philadelphia, 2006.
- H. Elman, D. Silvester, A. Wathen, Finite elements and fast iterative solvers – with applications in incompressible fluid dynamics, 2nd ed., Oxford Univ. Press, 2014.
- Y. Saad, Iterative methods for sparse linear systems 2nd ed., SIAM, 2003.
- A.H. Stroud, Approximate calculation of multiple integrals, Prentice-Hall, 1971.
- J. Cahouet, J.-P. Chabard, Some fast 3D finite element solvers for the generalized Stokes problem, Int. J. Numer. Meth. Fluids, Vol. 8 869–895, 1988.
- L. P. Franca, R. Stenberg, Error analysis of some Galerkin least squares methods for the elasticity equations, SINUM, Vol.28 1680-1697, 1991.

Appendix: Lagrange multiplier approach for full-Nuemann problem

full-Neumann boundary problem

$$-\triangle u = f \text{ in } \Omega,$$

$$\partial_n u = h \text{ on } \partial\Omega.$$

- compatibility condition : $\int_\Omega f + \int_{\partial\Omega} h = 0$
- $[A]_{i\,j} = a(\varphi_j, \varphi_i)$. A : singular , Ker$A = \vec{1}$.
- $[\vec{b}]_i = F(\varphi_i)$: compatibility codition $\Leftrightarrow$ $\vec{b} \in$ ImA = (Ker$A$)$^\perp$.

solution in image of A : find $\vec{u} \in$ ImA $\quad A\vec{u} = \vec{b}$

Lagrange multiplier to deal with constraint $(\vec{x}, \vec{1}) = 0$.

$$\begin{bmatrix} A & \vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} \vec{b} \\ 0 \end{bmatrix}$$

$\vec{u} \in$ ImA and $\lambda = 0$.

▸ example13b.edp

```
matrix A = aa(Vh, Vh, solver=sparsesolver);
real[int] c(u[].n); c = 1.0; // kernel of A
matrix AA=[[A, c], [c', 0]]; // matrix with constraint
set(AA, solver=UMFPACK);
```