Introduction to finite element computation by FreeFem++ – towards numerical simulation of fluid flow problems

Atsushi Suzuki[1]

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

Workshop on Mathematical Analysis
for Nonlinear Phenomena, 15-16, October 2015

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

Basics of FEM by examples from the Poisson equation

- Poisson equation with inhomogeneous Dirichlet conditions
- error estimate by theory and FreeFem++ implementation
- matrix formulation and linear solver

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
- matrix formulation and linear solver

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
- characterstic Galerkin method

Basics of FEM by examples from the Poisson equation

- Poisson equation with inhomogeneous Dirichlet conditions
- error estimate by theory and FreeFem++ implementation
- matrix formulation and linear solver

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
- matrix formulation and linear solver

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
- characterstic Galerkin method

Poisson equation and a weak formulation

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

$$-\triangle u = f \text{ in } \Omega$$
$$u = g \text{ on } \partial\Omega$$

- function space $H^1(\Omega)$
- subset $V(g) = \{v \in H^1(\Omega)\,;\, v = g \text{ on } \partial\Omega\}$
- subspace $V = V(0) = H_0^1(\Omega)$

bilinear form and weak formulation :

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

Find $u \in V(g)$ s.t. $a(u,v) = (f,v)\ \forall v \in V$.

finite element space : $S_h \subset H^1(\Omega)$ by triangulation, $P1, P2$, etc.

- affine space $V_h(g) = \{v_h = w_h + \tilde{g}_h\,;\, w_h \in V_h\}$
- subspace $V_h = \{v_h\,;\, v_h(P) = 0\ P \in \partial\Omega\} \subset S_h$

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

$u_h(P) = g(P)\ P \in \partial\Omega$: inhomogeneous Dirichlet data

FreeFem++ script to solve Poisson equation by P2 element

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

▸ matrix poisson

```
mesh Th=square(20,20);
fespace Vh(Th,P2);
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);

solve poisson(uh,vh)=
int2d(Th)( dx(uh)*dx(vh)+dy(uh)*dy(vh) )
-  int2d(Th)( f*vh )
+  on(1,2,3,4,uh=g);
plot(uh);
```

homogeneous data to inhomogeneous one on the boundary

inhomogeneous Dirichlet problem:
Find $u \in V(g)$ s.t. $a(u,v) = (f,v)\ \forall v \in V$.

homogeneous Dirichlet problem:
Find $u \in V$ s.t. $a(u,v) = (f,v) - a(\tilde{g},v)\ \forall v \in V$.

$\tilde{g} \in H^1(\Omega),\ \tilde{g} = g$ on $\partial\Omega$.

error estimate : theory 1 /2

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

Lemma (Galerkin orthogonality)

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

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

Lemma (Céa)

$$||u - u_h|| \leq (1 + \frac{\gamma}{\alpha}) \inf_{v_h \in V_h} ||u - v_h||.$$

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

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

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

error estimate : theory 2 /2

$\Pi_h : C(\bar{\Omega}) \to V_h, \quad \Pi_h u = \sum_{1\leq i\leq N} u(P_i)\phi_i,$

$\{\phi_i\}_{1\leq i\leq N}$: P_k finite element basis, span$[\{\phi_i\}] = S_h$.

Theorem (interpolation error by polynomial)

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

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

$0 \leq s \leq \min\{k+1, l\}$.

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:

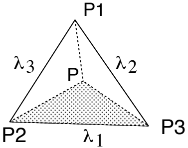
$\{P_i\}_{i\leq i\leq m}$: integration points in K, $\{\omega_i\}_{i\leq i\leq m}$: weights

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

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

```
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,P2);
  ...
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;
```

error estimate on unstructured mesh

unstructured mesh is generated by Delaunay triangulation

```
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

It seems to be better to look $\min_K h_K$, $\sum_K h_K/\#\mathcal{T}_h$, $\max_K h_K$, corresponding to mesh refinement.

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

matrix formulation of discretized form

Find $u_h \in V_h(g)$ s.t. $a(u_h, v_h) = (f, v_h)\ \forall v_h \in V_h$.
finite element basis: $\{\phi_i\}_{1\leq i\leq N}$, span$[\{\phi_i\}] = S_h \supset V_h$
$\phi_i(P_j) = \delta_{i\,j}$, P_j : finite element node.
$u_h \in S_h$, $u_h = \sum_{1\leq j\leq N} u_j\phi_j$.
Dirichlet data :
$u_h(P_k) = g(P_k) \ \Leftrightarrow\ u_k = g(P_k)$, $k \in \Lambda_D \subset \{1, \dots, N\}$.
matrix : $A \in \mathbb{R}^{N\times N}$, $[A]_{i\,j} = a(\phi_j, \phi_i)$,
symmetric positive semi-definite.

Find $\{u_j\}\,;\, u_k = g(P_k), k \in \Lambda_D$ s.t.
$a(\sum_{1\leq j\leq N} u_j\phi_j, v_h) = (f, v_h)\ \forall v_h \in V_h$.

Find $\{u_j\}\,;\, u_k = g(P_k), k \in \Lambda_D$ s.t.
$\sum_{1\leq j\leq N} u_j a(\phi_j, \phi_i) = (f, \phi_i)\ \forall i \in \{1, \dots, N\} \setminus \Lambda_D$.

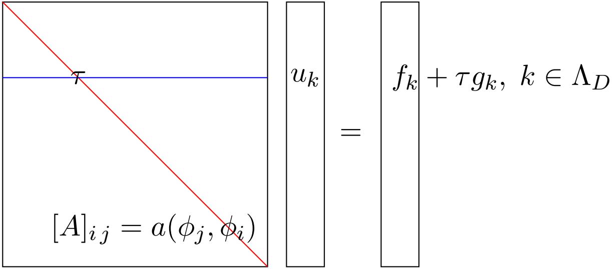
Find $\{u_j\} \in \mathbb{R}^N$ s.t.

$$\sum_{1\leq j\leq N} [A]_{i\,j} u_j = f_i, \qquad \forall i \in \{1, \dots, N\} \setminus \Lambda_D,$$
$$u_k = g(P_k), \qquad k \in \Lambda_D.$$

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`



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

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

keeping symmetry of the matrix without changing index numbering.

FreeFem++ script to solve Poisson using matrix

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

▸ solve poisson

```
Vh u,v;
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+30;
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; // ff = bc ? bc : ff;
u[] = A^-1 * ff;
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

Basics of FEM by examples from the Poisson equation

- Poisson equation with inhomogeneous Dirichlet conditions
- error estimate by theory and FreeFem++ implementation
- matrix formulation and linear solver

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
- matrix formulation and linear solver

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
- characterstic Galerkin method

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.

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

Stokes equations and a weak formulation : 2/3

Lemma (Gauss-Green's formula)

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

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

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

$$\begin{aligned} -2 \int_\Omega \textstyle\sum_i \sum_j \partial_j \frac{1}{2}(\partial_i u_j + \partial_j u_i) v_i \, dx &= \int_\Omega \textstyle\sum_{i,j} (\partial_i u_j + \partial_j u_i) \partial_j v_i \, dx \\ &\quad - \int_{\partial\Omega} \textstyle\sum_{i,j} (\partial_i u_j + \partial_j u_i) n_j v_i \, ds \\ &= \int_\Omega 2D(u) : D(v) \, dx - \int_{\partial\Omega} 2D(u) \, n \cdot v \, ds \end{aligned}$$

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 :
  $$\exists \beta_0 > 0 \quad \sup_{v \in V, v \neq 0} \frac{b(v,q)}{||v||_1} \geq \beta_0 ||q||_0 \ \ \forall\, q \in Q.$$

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

Lemma

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

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

Lemma

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

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

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

stabilized finite element method (penalty type)

$V_h \subset V$: 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 :

  $\exists \beta_0, \, \beta_1 > 0 \; \forall h > 0 \; \sup_{v_h \in V_h} \dfrac{b(v_h, q_h)}{||v_h||_1} \geq \beta_0 ||q_h||_0 - \beta_1 |q_h|_h \; \forall \, q_h \in Q_0$.

Lemma

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

$$\forall \, (v_h, q)_h \in V_h \times Q_h.$$

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

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

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

```
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,v1)+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) / int2d(Th)(1.0);
p = p - meanp;
plot([u1,u2],p,wait=1,value=true,coef=0.1);
```

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

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

```
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,v1)+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 : homonegenous 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 : $\mathbb{R}^{N_S} \to$ span$[\{\vec{1}\}]^\perp$,
  $P\,\vec{q} = \vec{q} - (\vec{q},\vec{1})/(\vec{1},\vec{1})\vec{1}$.
  preconditioner $[M]_{\mu\,\nu} = \int_\Omega \psi_\nu \psi_\mu dx$.

conjugate gradient with Uzawa method : inhomogeneous Dirichlet

$$\begin{bmatrix} A_\tau & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \vec{u} \\ \vec{p} \end{bmatrix} = \begin{bmatrix} \vec{f}_\tau \\ \vec{0} \end{bmatrix}$$

$[A_\tau]_{k\,k} = \tau$, $[f_\tau]_k = \tau\, g_k$ for $k \in \Lambda_D$.

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

Preconditioned CG method with projection `LinearCG()`

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

$\vec{g}^{\,0} = B A_\tau^{-1} \vec{f}_\tau$, $\quad \vec{g}^{\,0\prime} = P M^{-1} \vec{g}^{\,0}$, $\quad \vec{w}^{\,0} = \vec{g}^{\,0\prime}$.

loop $n = 0, 1, \ldots$

$$\alpha_n = (\vec{g}^{\,n\prime}, \vec{g}^{\,n}) / (B A_\tau^{-1} B^T \vec{w}^{\,n}, \vec{w}^{\,n}),$$
$$\vec{p}^{\,n+1} = \vec{p}^{\,n} + \alpha_n \vec{w}^{\,n},$$
$$\vec{g}^{\,n+1} = \vec{g}^{\,n} - \alpha_n (B A_\tau^{-1} B^T) \vec{w}^{\,n},$$
$$\vec{g}^{\,n+1\prime} = P M^{-1} \vec{g}^{\,n+1},$$
$$\beta_n = (\vec{g}^{\,n+1\prime}, \vec{g}^{\,n+1}) / (\vec{g}^{\,n\prime}, \vec{g}^{\,n}),$$
$$\vec{w}^{\,n+1} = \vec{g}^{\,n+1\prime} + \beta_n \vec{w}^{\,n}.$$

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

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

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

FreeFem++ script to generate Stokes matrix

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

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

```
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=g1,u2=g2);
varf external([u1,u2,p],[v1,v2,q])=
    int2d(Th)(f1 * v1 + f2 *v2);
matrix A = stokes(VQh,VQh,solver=UMFPACK);
real[int] bc = stokes(0, VQh);
real[int] ff = external(0, VQh);
ff += bc;
u1[] = A^-1 * ff;
```

FreeFem++ script for CG with Uzawa 1/2

```
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=1,u2=1);
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 .* bcsol1[];  // [bscol1 bscol2] keeps B.C.
real[int] uu = A^-1 * ff;
q[] = B * u;
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 .* bcsol1[];
u1[] = A^-1 * ff;       // to access [u1, u2]
```

- Basics of FEM by examples from the Poisson equation
  - Poisson equation with inhomogeneous Dirichlet conditions
  - error estimate by theory and FreeFem++ implementation
  - matrix formulation and linear solver
- 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
  - matrix formulation and linear solver
- 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
  - characterstic Galerkin method

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

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

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

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

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.

initial guess is taken from the stationary state of lower Reynolds number

stream line for visualization of 2D flow

stream function $\varphi : \Omega \to \mathbb{R}$

$$
\begin{aligned}
-\nabla^2\varphi =& \nabla \times u = \partial_1 u_2 - \partial_2 u_1 && \text{in } \Omega \\
\varphi =& 0 && \text{on } \Omega
\end{aligned}
$$

$u = \begin{bmatrix} \partial_2\varphi \\ -\partial_1\varphi \end{bmatrix} \;\Leftrightarrow\; u \perp \nabla\varphi$.

```
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);
```

FreeFem++ script for stationary cavity driven flow : 1/2

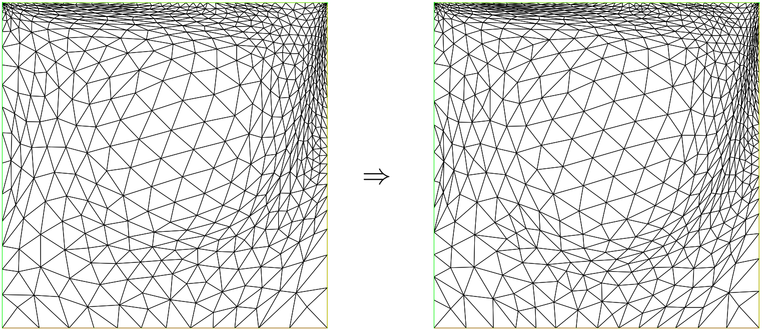
```
fespace XXMh(Th,[P2,P2,P1]);
XXMh [u1,u2,p], [v1,v2,q];
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //
macro div(u1,u2) (dx(u1)+dy(u2))//
macro grad(u1,u2) [dx(u1),dy(u2)]//
macro ugrad(u1,u2,v) (u1*dx(v)+u2*dy(v)) //
macro Ugrad(u1,u2,v1,v2) [ugrad(u1,u2,v1),ugrad(u1,u2,v2)]//

real epsln = 1.0e-6;
solve Stokes ([u1,u2,p],[v1,v2,q],solver=UMFPACK) =
  int2d(Th)(2.0*(dx(u1)*dx(v1)+2.0*d12(u1,u2)*d12(v1,v2) +dy(u2)*dy(v2))
            - p * div(v1,v2) - q * div(u1,u2)
            - p * q * epsln)
  + on(3,u1=4*x*(1-x),u2=0)  // boundary condition for the top flow
  + on(1,2,4,u1=0,u2=0);
real nu=1.0;                 // being updated during incremental loop
XXMh [up1,up2,pp];
varf vDNS ([u1,u2,p],[v1,v2,q]) =
  int2d(Th)(nu * 2.0*(d11(u1)*d11(v1)+2.0*d12(u1,u2)*d12(v1,v2)+d22(u2)*d22(v2))
            - p * div(v1, v2) - q * div(u1, u2)
            - p * q * epsln
            // Temam's trick
            + (Ugrad(u1,u2,up1,up2)'*[v1,v2] - Ugrad(u1,u2,v1,v2)'*[up1,up2]) / 2.0
            + (Ugrad(up1,up2,u1,u2)'*[v1,v2] - Ugrad(up1,up2,v1,v2)'*[u1,u2]) / 2.0 )
  + on(1,2,3,4,u1=0,u2=0);   // homogeneous Dirichlet b.c.

varf vNS ([u1,u2,p],[v1,v2,q]) =
  int2d(Th)(nu * 2.0*(d11(up1)*d11(v1)+2.0*d12(up1,up2)*d12(v1,v2)+d22(up2)*d22(v2))
            - pp * div(v1, v2) - q * div(up1, up2)
            - pp * q * epsln
            + (Ugrad(up1,up2,up1,up2)'*[v1,v2] - Ugrad(up1,up2,v1,v2)'*[up1,up2]) / 2.0 )
  + on(1,2,3,4,u1=0,u2=0);   // homogeneous Dirichlet b.c.
```

FreeFem++ script for stationary cavity driven flow : 2/2

```
Xh uu1=u1, uu2=u2; // initial condition is given by Stokes eqs. : Re=0.
up1[] = 0.0;                            // initializing of [up1, up2, pp]
real reyini = 100.0;
real reymax = 12800.0;
real re = reyini;
int kreymax = log(reymax / reyini)/log(2.0) * 2.0;
for(int k = 0; k < kreymax; k++) {
  re *= sqrt(2.0);
  real lerr=0.02; // parameter to controle mesh refinmente
  if(re>8000) lerr=0.01;
  if(re>10000) lerr=0.005;
  for(int step= 0 ;step < 2; step++) {
    Th=adaptmesh(Th, [u1,u2], p, err=lerr, nbvx=100000);
    [u1, u2, p]=[u1, u2, p];            // update of velocity/preesue on new mesh
    [up1, up2, pp]=[up1, up2, pp];
    plot(Th, wait=1);
    for (i = 0 ; i < 20; i++) {
      nu = 1.0 / re;
      up1[] = u1[];                     // access to [up1,up2,pp]
      real[int] b = vNS(0, XXMh);
      matrix Ans = vDNS(XXMh, XXMh, solver=UMFPACK);
      real[int] w = Ans^-1*b;
      u1[] -= w;                        // access to [u1,u2,p]
      cout << "iter = "<< i << "  " << w.l2 << " Reynolds number = " << re
           << endl;
      if(w.l2 < 1.0e-6) break;
    } // loop : i
  }   // loop : step
  streamlines;
  plot(psi,wait=1,nbiso=30);
  uu1 = u1;                             // extract velocity component from [u1, u2, p]
  uu2 = u2;
  plot(coef=0.2,cmm="rey="+re+" [u1,u2] and p  ",psi,[uu1,uu2],wait=1,nbiso=20);
}  // loop : re
```

mesh adaptation

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



`err` : P_1 interpolation error level

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

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++;
}
```

array, finite element space, 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
```

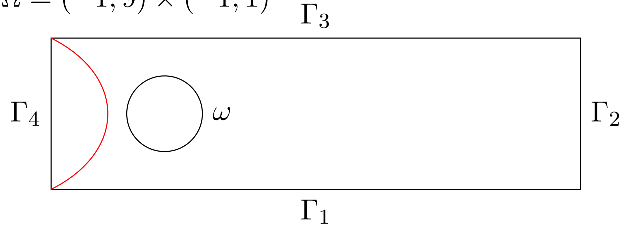
procedure (function)

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

- Basics of FEM by examples from the Poisson equation
  - Poisson equation with inhomogeneous Dirichlet conditions
  - error estimate by theory and FreeFem++ implementation
  - matrix formulation and linear solver
- 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
  - matrix formulation and linear solver
- 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
  - characterstic Galerkin method

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:

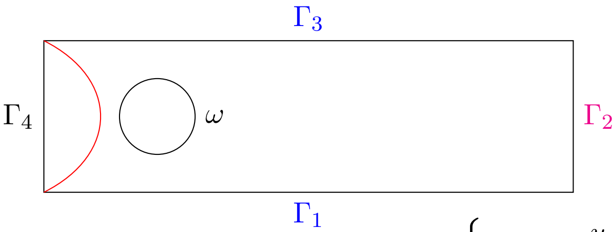
Poiseuille flow on Γ_4 : $u = (1 - y^2, 0)$.

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



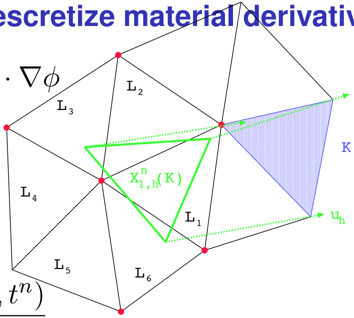
slip boundary condition on $\Gamma_1 \cup \Gamma_3$: $\left\{ \begin{array}{c} u \cdot n = 0 \\ (2\nu\, D(u)n - n\,p) \cdot t = 0 \end{array} \right.$

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

non-slip

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

characteristic Galerkin method to descretize material derivative



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

using characteristic line :

$$\frac{dX}{dt}(t) = u(X(t), t)$$

$$\frac{D\phi}{Dt} = \frac{d}{dt}\phi(X(t), t).$$

$$\frac{D\phi}{Dt} \sim \frac{\phi(X(t^{n+1}), t^{n+1}) - \phi(X(t^n), t^n)}{\Delta t}$$

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

approximation by Euler method, $X^n(x) = x - u(x, t^n)\Delta t$.

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

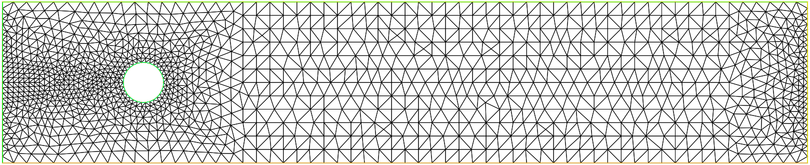
FreeFem++ provides `convect` to compute $(u^n \circ X^n, \cdot)$.

```
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);
```



References

- F. Hecht, FreeFem++ manual, 3rd ed, 2015.
- D. Braess, Finite elements – Theory, fast solvers and application in solid mechanics, 3rd ed. Cambridge Univ. Press, 2007.
- H. Elman, D. Silvester, A. Wathen, Finite elements and fast iterative solvers – with applications in incompressible fluid dynamics, 2nd ed. Oxford Univ. Press, 2014.
- M. Tabata, Numerical solution of partial differential equations II (in Japanese), Iwanami Shoten, 1994.
- A.H. Stroud, Approximate calculation of multiple integrals, Prentice-Hall, 1971.
- L. P. Franca, R. Stenberg, Error analysis of some Galerkin least squares methods for the elasticity equations, SINUM, Vol.28 1680-1697, 1991.