```
// example 6 : stokes-stabilized-error.edp   [slide page 32]
// error estimation for finite element solution of Stokes equations
// with stabilization of penalty type for P1/P1 element, parameter delta = 0.01
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki

int n1 = 40;
int n2 = n1 * 2;
mesh Th1=square(n1,n1);
mesh Th2=square(n2,n2);

fespace Vh1(Th1,P1),Qh1(Th1,P1);
fespace Vh2(Th2,P1),Qh2(Th2,P1);

// external force
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 sol1 = sin(pi * x) * sin(pi * y / 2.0) / 2.0;
func sol2 = cos(pi * x) * cos(pi * y / 2.0);
func solp = x * x + y * y - 2.0/3.0;

func sol1x = pi * cos(pi * x) * sin(pi * y / 2.0) / 2.0;
func sol1y = pi / 2.0 * sin(pi * x) * cos(pi * y / 2.0) / 2.0;

func sol2x = (-pi) * sin(pi * x) * cos(pi * y / 2.0);
func sol2y = (-pi / 2.0) * cos(pi * x) * sin(pi * y / 2.0);

// finite element solutions and test functions
Vh1 u11,u12,v11,v12;
Qh1 p1,q1;
Vh2 u21,u22,v21,v22;
Qh2 p2,q2;

real delta = 0.01;
real epsln = 1.0e-6;
// definitions of macros for strain rate tensor
macro d11(u1)     dx(u1) //
macro d22(u2)     dy(u2) //
macro d12(u1,u2)  (dy(u1) + dx(u2))/2.0 //

// stokes problem
solve stokes1(u11,u12,p1, v11,v12,q1) =
int2d(Th1)(
        2.0*(d11(u11)*d11(v11)+2.0*d12(u11,u12)*d12(v11,v12)+d22(u12)*d22(v12))
        - p1*dx(v11) - p1*dy(v12)
        - dx(u11)*q1 - dy(u12)*q1
        - delta * hTriangle * hTriangle * (dx(p1)*dx(q1) + dy(p1)*dy(q1))
        - p1*q1*epsln )
-  int2d(Th1)( f1 * v11 + f2 * v12 ) //
+  on(1,2,3,4,u11=sol1,u12=sol2);

real meanp,err1,err2,hh1,hh2;

meanp = int2d(Th1)(p1) / Th1.area;
p1 = p1 - meanp;

plot([u11,u12],p1,wait=1,value=true,coef=0.1);

solve stokes2(u21,u22,p2, v21,v22,q2) =
int2d(Th2)(
        2.0*(d11(u21)*d11(v21)+2.0*d12(u21,u22)*d12(v21,v22)+d22(u22)*d22(v22))
        - p2*dx(v21) - p2*dy(v22)
        - dx(u21)*q2 - dy(u22)*q2
        - delta * hTriangle * hTriangle * (dx(p2)*dx(q2) + dy(p2)*dy(q2))
        - p2*q2*epsln
)
-  int2d(Th2)( f1 * v21 + f2 * v22) //
+  on(1,2,3,4,u21=sol1,u22=sol2);

meanp = int2d(Th2)(p2) / Th2.area;
p2 = p2 - meanp;
```

```
plot([u21,u22],p2,wait=1,value=true,coef=0.1);

hh1 = 1.0 / n1 * sqrt(2.0);
hh2 = 1.0 / n2 * sqrt(2.0);

err1 = int2d(Th1)( (dx(u11) - sol1x) * (dx(u11) - sol1x)
                   + (dy(u11) - sol1y) * (dy(u11) - sol1y)
                   + (u11 - sol1) * (u11 - sol1) +
                   (dx(u12) - sol2x) * (dx(u12) - sol2x)
                   + (dy(u12) - sol2y) * (dy(u12) - sol2y)
                   + (u12 - sol2) * (u12 - sol2) +
                   (p1 - solp) * (p1 - solp));
err1 = sqrt(err1);

err2 = int2d(Th2)( (dx(u21) - sol1x) * (dx(u21) - sol1x)
                   + (dy(u21) - sol1y) * (dy(u21) - sol1y)
                   + (u21 - sol1) * (u21 - sol1) +
                   (dx(u22) - sol2x) * (dx(u22) - sol2x)
                   + (dy(u22) - sol2y) * (dy(u22) - sol2y)
                   + (u22 - sol2) * (u22 - sol2) +
                   (p2 - solp) * (p2 - solp));

err2 = sqrt(err2);

cout << "coarse mesh: h=" << hh1 << " err-H1/L2=" << err1 << endl;
cout << "fine mesh:   h=" << hh2 << " err-H1/L2=" << err2 << endl;
cout << "O(h)=" << log(err1/err2)/log(hh1/hh2) << endl;
```