```
// example 7 : stokes-matrix.edp   [slide page 35]
// Stokes equations with P2/P1 element
// to create indefinite stiffness matrix
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki

int n = 20;
mesh Th=square(n,n);
fespace Vh(Th, P2);
fespace Qh(Th, P1);
fespace VQh(Th, [P2, P2, 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 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);

VQh [u1,u2,p], [v1,v2,q]; // finite element solutions and test functions

// 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 //
macro div(u1, u2) (dx(u1) + dy(u2)) //

real epsln = 1.0e-6;
varf stokes([u1, u2, p], [v1, v2, q])=
  int2d(Th)(2.0 * (d11(u1) * d11(v1)
                   +2.0 * d12(u1,u2) * d12(v1,v2) +
                   d22(u2) * d22(v2))
            +q * div(u1, u2)
            -p * div(v1, v2)
            + p * q * epsln)
  + on(1,2,3,4,u1=1.0,u2=1.0); // enough to say inhomogeneous Dirichlet

varf external([u1,u2,p],[v1,v2,q])=
  int2d(Th)(f1 * v1 + f2 *v2)
  + on(1,2,3,4,u1 = sol1,u2 = sol2); // Dirichlet data are given

matrix A = stokes(VQh,VQh,solver=UMFPACK);
real[int] ff = external(0, VQh);

u1[] = A^-1 * ff;

// extract each compoment of velocity / pressure
Qh pp;
Vh uu1, uu2;
uu1 = u1; uu2 = u2; pp = p;
//

real meanp;
meanp = int2d(Th)(pp) / Th.area;
pp = pp - meanp;
plot(pp,wait=1,value=true,coef=0.1);
plot([uu1,uu2],wait=1,value=true,coef=0.1);
real err;
err = int2d(Th)( (dx(uu1) - sol1x) * (dx(uu1) - sol1x)
                 + (dy(uu1) - sol1y) * (dy(uu1) - sol1y)
                 + (uu1 - sol1) * (uu1 - sol1) +
                 (dx(uu2) - sol2x) * (dx(uu2) - sol2x)
                 + (dy(uu2) - sol2y) * (dy(uu2) - sol2y)
                 + (uu1 - sol2) * (uu2 - sol2)
                 + (pp - solp) * (pp - solp));
err = sqrt(err);
cout << "error-H1 velocity -L2 pressure " << err<< endl;
```