```
// example 14 : stokes-Uzawa.edp   [slide page 63]
// Stokes equations with P2/P1 element
// using block factorization aka Uzawa method with conjugate gradient
// for tutorial by Japan SIAM, Atsushi Suzuki, Feb.2016

int n = 40;
mesh Th=square(n,n);

fespace Xh(Th,P2);
fespace Vh(Th,[P2,P2]),Qh(Th,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
Vh [u1,u2], [v1,v2], [bcsol1, bcsol2];
Qh p,q;

// 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 //

varf a([u1,u2], [v1,v2])=
  int2d(Th)(2.0 * (d11(u1) * d11(v1)
                   +2.0 * d12(u1,u2) * d12(v1,v2) +
                   d22(u2) * d22(v2)))
  + on(1,2,3,4,u1=sol1,u2=sol2);

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);

real tgv=1.0e+30;
matrix A = a(Vh,Vh,solver=UMFPACK,tgv=tgv);
matrix B = b(Vh,Qh,tgv=tgv);
matrix Mp = massp(Qh,Qh,solver=UMFPACK,tgv=tgv);
real[int] bc = a(0, Vh);    // Dirichlet data with tgv : given by on (1,2,3,4, )
real[int] ff = external(0, Vh);

func real[int] UzawaStokes(real[int] &pp)
{
  real[int] b = B'*pp;
  real[int] uu =A^-1 * b;  // A^-1 = tgv^-1 on Dirichlet nodes
  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; // initial value of CG
```

```
// preparation of RHS of CG method
{
  ff += bc;                 // ff is modified to have Dirichelt data with tgv
  real[int] uu = A^-1 * ff; // uu has Dirichlet data
  q[] = B * uu;
}

LinearCG(UzawaStokes, p[], q[], precon=PreconMass, nbiter=100, eps=1.0e-10,
         verbosity=100);

// compute velocity with inhomogeenous Dirichlet data from pressure
{
  ff = external(0, Vh);
  real[int] b = B'*p[];
  ff -= b;
  ff += bc;                // ff is modified to have Dirichelt data with tgv
  u1[] = A^-1 * ff;        // u1 has Dirichlet data
}

real meanp = int2d(Th)(p) / Th.area;
p = p - meanp;

real err;
err = int2d(Th)( (dx(u1) - sol1x) * (dx(u1) - sol1x)
                 + (dy(u1) - sol1y) * (dy(u1) - sol1y)
                 + (u1 - sol1) * (u1 - sol1) +
                 (dx(u2) - sol2x) * (dx(u2) - sol2x)
                 + (dy(u2) - sol2y) * (dy(u2) - sol2y)
                 + (u1 - sol2) * (u2 - sol2)
                 + (p - solp) * (p - solp));
err = sqrt(err);
cout << "error-H1 velocity " << err<< endl;
```