```
// example 8 : cavityNewton.edp   [slide page 40]
// stational Navier-Stokes equations in a cavity by Newton iteration
// P2/P1 element
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki
// based on examples++-tutorial/cavityNewtow.edp

mesh Th=square(40,40);
fespace Xh(Th,P2);
fespace Mh(Th,P1);
fespace XXMh(Th,[P2,P2,P1]);
XXMh [u1,u2,p], [v1,v2,q];

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))//
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*(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)
  + on(3,u1=4*x*(1-x),u2=0)
  + on(1,2,4,u1=0,u2=0);

plot(coef=0.2,cmm=" [u1,u2] and p  ",p,[u1,u2],wait=1);

Mh psi,phi;

problem streamlines(psi,phi,solver=UMFPACK) =
      int2d(Th)( dx(psi)*dx(phi) + dy(psi)*dy(phi))
   +  int2d(Th)( -phi*(dy(u1)-dx(u2)))
   +  on(1,2,3,4,psi=0);

streamlines;
plot(psi,wait=1);

real nu=1.0;
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);

// [up1, up2, pp] are obtained from the previous step
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);

Xh uu1=u1, uu2=u2; // initial condition is given by Stokes eqs : Re=0.
up1[] = 0.0; // initialize for [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 control 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];
    [up1, up2, pp]=[up1, up2, pp];
    plot(Th, wait=1);
    for (int i = 0 ; i < 20; i++) {
      nu = 1.0 / re;
      up1[] = u1[];
      real[int] b = vNS(0, XXMh);
      matrix Ans = vDNS(XXMh, XXMh, solver=UMFPACK);
      real[int] w = Ans^-1*b;
      u1[] -= w;
      cout << "iter = "<< i << "  " << w.l2 <<  " Reynolds number = " << re
           << endl;
      if(w.l2 < 1.0e-6) break;
    } // loop : i
  }   // loop : step
  streamlines;
  plot(psi,nbiso=30,wait=1);
  // extract velocity component from [u1, u2, p]
  uu1 = u1;
  uu2 = u2;
  plot(coef=0.2,cmm="rey="+re+" [u1,u2] and p  ",psi,[uu1,uu2],wait=1,nbiso=20);
}  // loop : re
```