```
// example 15 : NS-cylinder-Uzawa.edp   [slide page 65]
// Navier-Stokes flow around a cylinder
// P2/P1 element with Characteristic Galerkin
// generalized Stokes problem is solved by Uzawa method
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki

int n1 = 30;
int n2 = 60;
real nu = 1.0/400.0;
real dt = 0.05;
real alpha = 1.0/dt;
int timestepmax = 400;

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);
fespace Xh(Th,P2);
fespace Vh(Th,[P2,P2]);
fespace Qh(Th,P1);


Vh [u1,u2], [v1,v2], [up1,up2];
Qh p, q, pp;
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)) //

Qh psi,phi;
func inflow = 1.0 - y*y;
func stinflow=y-y*y*y/3.0;

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,psi=(-2.0/3.0))
   +  on(4,psi=stinflow)
   +  on(3,psi=(2.0/3.0))
   +  on(5,psi=0.0);

streamlines;
plot(psi,wait=1);

problem Stokes([u1,u2,p],[v1,v2,q],solver=UMFPACK) =
  int2d(Th)(
         + 2.0*nu * (d11(u1)*d11(v1)+2.0*d12(u1,u2)*d12(v1,v2)+d22(u2)*d22(v2))
         - p * div(v1, v2) - q * div(u1, u2)) //
+ on(1,3,u2=0)
+ on(4,u1=inflow,u2=0)
+ on(5,u1=0,u2=0);

varf stiff([u1, u2], [v1, v2]) =
  int2d(Th)( alpha * (u1*v1 + u2*v2)
         + 2.0*nu*(d11(u1)*d11(v1)+2.0*d12(u1,u2)*d12(v1,v2)+d22(u2)*d22(v2)))
         + on(1,3,u2=0)
         + on(4,u1=inflow,u2=0)
         + on(5,u1=0,u2=0);

varf external([u1, u2], [v1, v2]) =
  int2d(Th)( alpha * (convect([up1,up2],-dt,up1)*v1
                     +convect([up1,up2],-dt,up2)*v2) );

// matrix for divergence operator
varf bbb([u1,u2], [q]) = int2d(Th)(- q * dx(u1) - q * dy(u2));

// matrix for preconditioner
```

```
varf lapp(p,q)=int2d(Th)(dx(p)*dx(q)+dy(p)*dy(q))
               + on(2,p=0);
varf massp(p, q)= int2d(Th)(p * q);

matrix AA = stiff(Vh, Vh, solver=UMFPACK);
matrix Lt = lapp(Qh, Qh, solver=UMFPACK);
matrix Mp = massp(Qh, Qh, solver=UMFPACK);
matrix BB = bbb(Vh,Qh);
real[int] bbc = stiff(0, Vh);          // to treat inhomogenous Dirichlet data

func real[int] UzawaStokes(real[int] &ppp)
{
  real[int] vv = BB'*ppp;
  real[int] uu= AA^-1 * vv;
  ppp = BB * uu;
  return ppp;
}

func real[int] PreconMass(real[int] &ppp)
{
  real[int] ppp1 = Lt^-1 * ppp;
  real[int] ppp2 = Mp^-1 * ppp;
  ppp = alpha * ppp1 + nu * ppp2;
  return ppp;
}

plot(Th,wait=1);

Stokes; // initial condition is given by Stokes flow

for (int i = 0; i < timestepmax; i++) {
   [up1, up2] = [u1, u2];
   pp = p;
//   NS;
   real[int] ff0 = external(0, Vh); // saving for after CG.
   real[int] ff1 = ff0;
             // preparation of RHS of CG method
   {
     ff1 += bbc;
     real[int] uu = AA^-1 * ff1;
     q[] = BB * uu;
   }
   LinearCG(UzawaStokes, p[], q[], precon=PreconMass, nbiter=300, eps=-1.0e-14,
             verbosity=100);
   {
     real[int] uu = BB'*p[];
     ff0 -= uu;
     ff0 += bbc;
     u1[] = AA^-1 * ff0;
   }

   streamlines;
   plot(psi, nbiso=30,wait=0);
   if (i % 20 == 0) {
     plot([up1,up2],pp,wait=0,value=true,coef=0.1);
   }
}
```