```
// example 11 : RayleighBenard-stat.edp   [slide page 56]
// Navier-Stokes flow around a cylinder
// P2/P1 element with Characteristic Galerkin
// initial data for Newton iteration needs to be prepared in "rb.data"
// by Rayleigh-Benard-stat.edp
// for tutorial by Japan SIAM, Tokyo, 11-12 Feb.2016, Atsushi Suzuki

int n1 = 80;
int n2 = 20;
real Pr = 0.71;
real Ra = 1500.0;
int timestepmax = 600;

mesh Th=square(n1,n2,[x*4.0,y]);

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

Wh [u1,u2,p,th], [v1, v2, q, psi];
Vh up1, up2, thp;
Qh pp, ss, rr;
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 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;     // penalization parameter to avoid pressure ambiguity

varf vDRB ([u1,u2,p,th],[v1,v2,q,psi]) =
    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
              + ((Ugrad(u1,u2,up1,up2)'*[v1,v2] -           // trick  by Temam
                  Ugrad(u1,u2,v1,v2)'*[up1,up2]
              +   Ugrad(up1,up2,u1,u2)'*[v1,v2] -
                  Ugrad(up1,up2,v1,v2)'*[u1,u2]) / (2.0 *  Pr))
             )
  - int2d(Th)( Ra * th * v2 )
  + int2d(Th)( dx(th) * dx(psi) + dy(th) * dy(psi) )
  + int2d(Th)( (ugrad(up1,up2,th)*psi -
                ugrad(up1,up2,psi)*th) / 2.0 )
  + int2d(Th)( (ugrad(u1,u2,thp)*psi -
                ugrad(u1,u2,psi)*thp) / 2.0 )
  + on(1,3,u2=0)
  + on(2,4,u1=0)
  + on(1,th=0)
  + on(3,th=0);

// [up1, up2, pp] are obtained from the previous step
varf vRB ([u1,u2,p,th],[v1,v2,q,psi]) =
    int2d(Th)( 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 * Pr) )
  - int2d(Th)( Ra * thp * v2 )
  + int2d(Th)( dx(thp) * dx(psi) + dy(thp) * dy(psi) )
  + int2d(Th)( (ugrad(up1,up2,thp)*psi -
                ugrad(up1,up2,psi) * thp) / 2.0 )
  + on(1,3,u2=0)
  + on(2,4,u1=0)
  + on(1,th=0)  // force homogeneous Dirichlet B.C. to be compatible with
  + on(3,th=0); // Jacobian matrix of Newton iteration

problem streamlines(ss,rr,solver=UMFPACK) =
      int2d(Th)( dx(ss)*dx(rr) + dy(ss)*dy(rr))
```

```
   +  int2d(Th)( rr*(dy(up1)-dx(up2)))
   +  on(1,2,3,4,ss=0.0);

plot(Th,wait=1);
// read stationary data computed by RayleighBenrad.edp
{
  ifstream file("rb.data", binary);
  for (int i = 0; i < up1[].n; i++) {
    file >> up1[](i);
  }
  for (int i = 0; i < up2[].n; i++) {
    file >> up2[](i);
  }
  for (int i = 0; i < pp[].n; i++) {
    file >> pp[](i);
  }
  for (int i = 0; i < thp[].n; i++) {
    file >> thp[](i);
  }
}

[u1, u2, p, th] = [up1, up2, pp, thp];
plot(th,cmm="temperature is read from the file",value=true, wait=1);
cout << "start newton" << endl;
for (int n = 0; n < 20; n++) {
  up1 = u1;
  up2 = u2;
  pp = p;
  thp = th;
  plot(thp,cmm="newton n="+n,wait=0,value=true);
  real[int] b = vRB(0, Wh);
  cout << "||b||_l2 " << b.l2 << endl;
  matrix A = vDRB(Wh, Wh, solver=UMFPACK);
  real[int] w = A^-1 * b;
  u1[] -= w;
  cout << "iter = "<< n << "  " << w.l2 << endl;
  if (w.l2 < 1.e-6) break;
}
plot(thp,value=true);
up1 = u1;
up2 = u2;
streamlines;
plot(ss,nbiso=30);
```