```
// Calcul de l'?ecoulement autour d'un obstacle non profile? plac?e dans un canal
// la largeur de l'obstacle est fix?e ?a 1 ainsi que la vitesse moyenne
// on peut faire varier le rapport de blocage de l'e?coulement
// reprise du calcul à partir d'un calcul précédent
//
load "iovtk"
load "UMFPACK64";
bool restartok=false;
string yes;
// lecture du maillage et de la solution précédente
string meshfile,solfile;

cout << "entrer le nom du fichier de maillage : "; cin >> meshfile;
cout << "entrer le nom du fichier de solution : "; cin >> solfile;
mesh th=readmesh(meshfile);
plot(cmm="maillage",th,wait=1);
real[int] bb(4);
boundingbox(th, bb);
cout << "limites du domaine de calcul:" << endl;
cout << "xmin = " << bb[0]
         << ", xmax = " << bb[1]
         << ", ymin = " << bb[2]
         << ", ymax = " << bb[3] << endl;
real bloc;
bloc=bb[3]-bb[2];
real lcanal ;
lcanal=bb[1]-bb[0];

// espaces d'elements finis
fespace Xh(th,P2); // velocity space
fespace Mh(th,P1); // pressure space
Xh u1,u2;
Mh p;
// lecture des champs de pression et de vitesse calcules precedemment
{
ifstream pfile(solfile);
pfile >> p[] >> u1[] >> u2[];
}
plot(cmm="champ de pression",p,nbiso=50,fill=true,wait=1);
plot(cmm="champ de vitesse",[u1,u2],wait=1);

real tinit;
int ninit ;
cout << "entrer le temps (adimensionnel) initial  : "; cin >> tinit;

// centre de l'obstacle
real xob,yob;
xob=2.*bloc; yob=0.5*bloc;

Xh v1,v2;
Xh vor, sigma11, sigma22, sigma12 ;
Mh q;
```

```
// de?finition de la condition aux limites en entr?e du canal
// profil de vitesse parabolique
// u1(y=0) = 0 ; u1(y=bloc) = 0 : vitesse moyenne <u1> = 1
func uin=6*y*(bloc-y)/(bloc*bloc) ;
//func uin=1 ;


// definition du calcul de la fonction de courant
Xh psi,phi;
problem streamlines(psi,phi) =
int2d(th)( dx(psi)*dx(phi) + dy(psi)*dy(phi))
+ int2d(th)( -phi*(dy(u1)-dx(u2)))
+ on(1,psi=0);



// nu est l'inverse du nb de Reynolds
// attention ?a la d?efinition de Reynolds.
real reynolds ;
cout << " Entrer le nombre de Reynolds :"; cin >> reynolds;
real nu=1./reynolds;

// d?efinition du pas de temps
real dt=0.1;

cout << "Pas de temps : " << dt << "\n";

// numéro d'itération initiale
ninit=floor(tinit/dt);
//
// raffinement automatique du maillage ou pas ?
bool refinemesh=false;
cout << " Raffinement automatique du maillage pendant le calcul ? (o,n)"; cin >> yes;
if (yes == "o") refinemesh=true;
//
real alpha=1/dt;
// d?efinition du proble?me de Navier-Stokes
// la condition aux limites sur la face d'entr?ee dans le canal (d) prend en compte la perturbati
// d?ependant du temps (=dt*i)
Xh up1,up2;
int i;
problem NS (u1,u2,p,v1,v2,q,solver=UMFPACK,init=i) =
int2d(th)(
alpha*( u1*v1 + u2*v2)
+ nu * ( dx(u1)*dx(v1) + dy(u1)*dy(v1)
+ dx(u2)*dx(v2) + dy(u2)*dy(v2) )
- p*q*(0.000001)
- p*dx(v1) - p*dy(v2)
- dx(u1)*q - dy(u2)*q
)
+ int2d(th) ( -alpha*
convect([up1,up2],-dt,up1)*v1 -alpha*convect([up1,up2],-dt,up2)*v2 )
+ on(4,u1=uin,u2=0)
+ on(2,u2=0)
```

```
+ on(1,3,5,u1=0,u2=0);
//
// d?efinition des valeurs pour la visualisation de la vorticite? de -vmax ? vmax
real[int] vorval(51);
real vmax=10;
for (i=0;i<51;i++){
vorval[i]=vmax*(-1.+0.04*i);
}
// definition des couleurs pour l'affichage de la vorticite
// les couleurs sont definies par leur coordonnees hsv (hue, saturation, value ; teinte, saturati
real[int] colors(153) ;
for (i=0;i<26;i++){
colors[3*i]=0.008*i;
//colors[3*i+1]=1-0.04*i;
colors[3*i+1]=1;
colors[3*i+2]=1;
}
for (i=26;i<51;i++){
colors[3*i]=0.5+0.008*(i-26);
//colors[3*i+1]=0.04*(i-25);
colors[3*i+1]=1;
colors[3*i+2]=1;
}


// fichier de sortie pour le'?volution temporelle de la force sur l'obstacle
string fvstfile="f_vs_t_b"+bloc+"_re"+reynolds+"_ti"+tinit+ ".txt" ;
ofstream fvst(fvstfile,append);
// fichiers de sortie des donnees au format vtk pour affichage avec Paraview
string omegatkfile,vtkfile,ptkfile,psitkfile;
// nombre d'ite?rations en temps
int nbiter ;
cout << " Entrer le nombre d iterations :"; cin >> nbiter;

// de?finition des param?tres pour les sorties graphiques
bool calcf =true ;
real fx,fy ;
bool sortiep=true, sortiev=true, zoom=true, sortier=true;
real llx,lly,urx,ury;
llx=15. ;
lly=0. ;
urx=60. ;
ury=bloc ;

// d?efinition des vecteurs pour l'affichage force et vitesse en fonction du temps
real[int] temps(nbiter),fox(nbiter),foy(nbiter) ;
for (i=0;i<nbiter;i++){
temps[i]=tinit+dt*i;
fox[i]=0 ;
foy[i]=0 ;
}
// nouveau numero d'iteration
int in;
// off we go, iteration en temps
```

```
for (i=0;i<nbiter;i++){
in=i+ninit;
up1=u1;
up2=u2;
NS;
// calcul des contraintes et de la force sur l'obstacle
if (calcf){
sigma11=2*nu*dx(u1)-p;
sigma22=2*nu*dy(u2)-p ;
sigma12=nu*(dy(u1)+dx(u2));
fx=-int1d(th,5) (sigma11*N.x+sigma12*N.y);
fy=-int1d(th,5) (sigma12*N.x+sigma22*N.y);
fox[i]=fx;
foy[i]=fy;
fvst << in << "," << fx << "," << fy << "\n";
}
plot(cmm="Forces sur l'obstacle  iteration no "+in+" temps "+dt*in,[temps,fox],[temps,foy]);
if ( !(i % 10)) {
vor = dy(u1)-dx(u2);
if (sortiep) {
plot(cmm="iteration no "+in+" temps "+dt*in,p,nbiso=50,fill=1,bb=[[llx,lly],[urx,ury]]);
ptkfile="p_b"+bloc+"_re"+reynolds+"_t"+in+".vtk";
savevtk(ptkfile,th,p,dataname="pression");
}
if (sortiev) {
// calcul de la fonction de courant de l'ecoulement
streamlines ;
plot (cmm="iteration no "+in+" temps "+dt*in,psi,nbiso=50,bb=[[llx,lly],[urx,ury]]);
vtkfile="vit_b"+bloc+"_re"+reynolds+"_t"+in+".vtk";
savevtk(vtkfile,th,[u1,u2],dataname="vitesse");
}
if (sortier) {
plot(cmm="iteration no "+in+" temps "+dt*in,vor,viso=vorval,hsv=colors,fill=1,bb=[[llx,lly],[urx,
omegatkfile="vor_b"+bloc+"_re"+reynolds+"_t"+in+".vtk";
savevtk(omegatkfile,th,vor,dataname="vorticite");
}
}
if (refinemesh) {
// adaptation du maillage et interpolation des fonctions sur le nouveau maillage
th=adaptmesh(th,u1,u2,hmin=0.01);
u1=u1;
u2=u2;
p=p;
}
}
// sauvegarde du maillage final et des champs de vitesse et de pression
meshfile="mesh_b"+bloc+"_re"+reynolds+"_t"+in+".msh";
solfile="pu_b"+bloc+"_re"+reynolds+"_t"+in+".sol";
savemesh(th,meshfile);
{
ofstream solf(solfile);
solf << p[] << u1[] << u2[] << endl ;
}
```