```
// Calcul de l'ecoulement autour d'un obstacle non profile place dans un canal
// la largeur de l'obstacle est fixe a 1 ainsi que la vitesse moyenne
// on peut faire varier le rapport de blocage de l'ecoulement
// on peut etudier la reponse de l'ecoulement a diverses perturbations de la vitesse moyenne en e
// cette procedure est similaire a l'etude experimentale de Provansal et al. (J. Fluid Mech. 1987
// on peut soit imposer une impulsion de forme gaussienne (au temps tp, de largeur dtp et d'ampli
// soit une fluctuation sinusoi•dale (amplitude amp2, frequence adimensionne 2*Pi*omega)
// soit une variation du debit moyen (variation du debit correspondant a une variation du reynold
// valeur finale ref, au temps tpm, sur un intervalle de temps dtpm
//
// on peut chosir d'adapter automatiquement le maillage ? l'ecoulement autour du calcul (plus pre
// le maillage et la solution sont sauvegardes ? la fin du calcul
//
load "iovtk" ;
load "UMFPACK64";
real bloc, lcanal , lcrel;
string yes;
bloc=10;
cout << "Rapport de blocage (largeur du canal/largeur de l'obstacle ) :" << bloc << "\n";
lcanal=100;
// la longueur du canal est au moins 10 fois la largeur de l'obstacle
cout << " Longueur du canal  :" << lcanal << "\n" ;

// definition du nombre minimal de mailles en fonction de lcanal et bloc
int nbloc,ncanal ;
nbloc=floor(bloc);
ncanal=floor(lcanal);
// limites du canal
// fronti?eres decrites dans le sens trigo direct
border a(t=0,lcanal) {x=t;y=0;label=1;}; //bas
border b(t=0,bloc) {x=lcanal;y=t;label=2;}; // sortie
border c(t=0,lcanal) {x=lcanal-t;y=bloc;label=3;}; // haut
border d(t=0,bloc) {x=0;y=bloc-t;label=4;}; // entree
// obstacle
// frontiere decrite dans le sens trigo inverse
// le centre de l'obstacle est place a 2 largeurs en aval de l'entree du canal
int C1=99,top=97,bottom=96 ;
real xob,yob;
xob=2.*bloc;
yob=0.5*bloc;
// cylindre
//border e(t=2*pi,0) {x=2+0.5*cos(t);y=0.5+0.5*sin(t);};
// section trapeze
border e(t=-0.5, 0.5) {x=xob;y=yob+t;label=5;};
border f(t=0, 1) {x=xob+t;y=yob+0.5-0.1*t;label=5;};
border g(t=0.4, -0.4) {x=xob+1;y=yob+t;label=5;};
border h(t=0, 1) {x=xob+1-t;y=yob-0.4-0.1*t;label=5;};
int i=0;
int n=1;
int nr=1;
// 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;
else {
n=2;
nr=3;
}
mesh th = buildmesh(a(ncanal*n)+b(nbloc*n)+c(ncanal*n)+d(nbloc*n)+e(nr*n)+f(nr*n)+g(nr*n)+h(nr*n)
//plot (th,wait=1,ps="maillage.ps") ;
plot (cmm="maillage initial",th,wait=1) ;



// definition des espaces d'elements finis
// P2 elements quadratiques continus par morceaux

fespace Xh(th,P2); // velocity space
fespace Mh(th,P1); // pressure space
// u1,u2,p sont les deux composantes de vitesse et la pression calculees
Xh u2,v2;
Xh u1,v1;
Xh vor, sigma11, sigma22, sigma12 ;
Mh p,q;

// definition de la condition aux limites en entre 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 ;



// calcul de la solution de Stokes pour initialiser le champ de vitesse

solve Stokes ([u1,u2,p],[v1,v2,q],solver=UMFPACK) =
    int2d(th)( ( 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
           )
+ on(2,4,u1=uin,u2=0)
+ on(1,3,5,u1=0,u2=0);

// calcul et affichage 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);


streamlines ;
// affichage de la solution de Stokes
plot(cmm="solution a Re = 0",psi,nbiso=50,wait=1);
```

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


// definition du pas de temps
real dt=0.1;
cout << " Pas de temps adimensionnel : (<1) " << dt << "\n";
//
real alpha=1/dt;
// definition du probleme de Navier-Stokes
// u parabolique en entree (frontiere n°4)
Xh up1,up2;
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);
//
// definition des valeurs pour la visualisation de la vorticite de -vmax a 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 levolution temporelle de la force sur l'obstacle
string fvstfile="f_vs_t_b"+bloc+"_re"+reynolds+".txt" ;
ofstream fvst(fvstfile,append);
// fichiers de sortie des donnees au format vtk pour affichage avec Paraview
string omegatkfile,vtkfile,ptkfile,psitkfile;
// nombre d'iterations en temps
int nbiter ;
cout << " Entrer le nombre d iterations :"; cin >> nbiter;

// definition des parametres pour les sorties graphiques
//string store_dir,base_dir ;
bool calcf =true ;
real fx,fy ;
bool sortiep=true, sortiev=true,zoom, sortier=true;
zoom = true;
real llx,lly,urx,ury;
llx=15. ;
lly=0. ;
urx=60 ;
ury=10 ;

// sauvegarde des parametres de calcul dans un fichier
string paramfile="parametres_b"+bloc+"_re"+reynolds+".txt" ;
ofstream param(paramfile,append);
param << "blocage " << "," << bloc << "\n";
param << "longueur totale " << "," << lcanal << "\n";
param << "reynolds" << "," << reynolds << "\n";
param << "pas de temps " << "," << dt << ", nb iterations" << "," << nbiter <<"\n";
// definition 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]=dt*i;
fox[i]=0 ;
foy[i]=0 ;
}
// off we go, iteration en temps
for (i=0;i<nbiter;i++){
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 << i << "\t" << fx << "\t" << fy << "\n";
}
plot(cmm="Forces sur l'obstacle  iteration no "+i+" temps "+dt*i,[temps,fox],[temps,foy]);
if ( !(i % 10)) {
vor = dy(u1)-dx(u2);
```

```
if (sortiep) {
plot(cmm="iteration no "+i+" temps "+dt*i,p,nbiso=50,fill=1,bb=[[llx,lly],[urx,ury]]);
ptkfile="p_b"+bloc+"_re"+reynolds+"_t"+i+".vtk";
savevtk(ptkfile,th,p,dataname="pression");
}
if (sortiev) {
// calcul de la fonction de courant de l'ecoulement
streamlines ;
plot (cmm="iteration no "+i+" temps "+dt*i,psi,nbiso=50,bb=[[llx,lly],[urx,ury]]);
vtkfile="vit_b"+bloc+"_re"+reynolds+"_t"+i+".vtk";
savevtk(vtkfile,th,[u1,u2],dataname="vitesse");
}
if (sortier) {
plot(cmm="iteration no "+i+" temps "+dt*i,vor,viso=vorval,hsv=colors,fill=1,bb=[[llx,lly],[urx,ur
omegatkfile="vor_b"+bloc+"_re"+reynolds+"_t"+i+".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
string meshfile="mesh_b"+bloc+"_re"+reynolds+"_t"+i+".msh";
string solfile="pu_b"+bloc+"_re"+reynolds+"_t"+i+".sol";
savemesh(th,meshfile);
{
ofstream solf(solfile);
solf << p[] << u1[] << u2[] << endl ;
}
```