```
// 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
entree
// 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'amplitude amp1)
// soit une fluctuation sinusoidale (amplitude amp2, frequence adimensionne 2*Pi*omega)
// soit une variation du debit moyen (variation du debit correspondant a une variation du
reynolds de sa valeur initiale vers une
// valeur finale ref, au temps tpm, sur un intervalle de temps dtpm
//
// on peut chosir d'adapter automatiquement le maillage à l'écoulement autour du calcul (plus
précis, mais plus couteux en temps de calcul)
// le maillage et la solution sont sauvegardés à la fin du calcul
//
load "iovtk" ;
load "UMFPACK64";
real bloc, lcanal , lcrel;
cout << " Entrer le rapport de blocage (largeur du canal/largeur de l'obstacle >1) :"; cin >> bloc;
assert (bloc>1);
// la longueur du canal est au moins 10 fois la largeur de l'obstacle
cout << " Entrer la longueur du canal (>10 et < 150) :"; cin >> lcanal;
assert (lcanal>10);
assert (lcanal<150);
// definition du nombre minimal de mailles en fonction de lcanal et bloc
int nbloc,ncanal ;
nbloc=floor(bloc);
ncanal=floor(lcanal);
// limites du canal
// frontieres 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  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 trapèze
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;
cout << " Entrer la resolution du maillage (>1) :"; cin >> n;
cout << " Entrer le raffinement sur l'obstacle (>1) :"; cin >> nr;
```

```
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 (th,wait=1) ;
// determination de la surface min,max et moyenne des mailles
int nbtriangles=th.nt;
real minarea=100000., maxarea=0., avarea=0.;
real minlength=100000., maxlength=0., avlength=0.;
for (i=0;i<nbtriangles;i++){
        avarea=avarea+th[i].area;
        minarea=min(minarea,th[i].area);
        maxarea=max(maxarea,th[i].area);
}
avarea=avarea/nbtriangles;
cout << "surface min : " << minarea << " surface max : " << maxarea << " surface moyenne : "
<< avarea ;
minlength=sqrt(minarea);
maxlength=sqrt(maxarea);
avlength=sqrt(avarea);


// definition des espaces d'elments 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(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 des perturbations de vitesse
real amp1=0.,amp2=0.,ref=reynolds,ampm,tp=0.,dtp =1.,tpm=0.,dtpm=1.,omega=1.;
bool pert=false, pulse=false, sinus = false , marche =false ;
string yes ;
cout << "Perturbation de l'ecoulement ? (o, n)"; cin >> yes;
pert = (yes == "o");
if (pert) {
        cout << " Definition des parametres des perturbations de l'ecoulement";
        cout << " Impulsion ? (o, n) :"; cin >> yes;
        pulse = (yes == "o");
        if (pulse) {
                cout << " Entrer l'amplitude de l'impulsion de vitesse (<1) :"; cin >> amp1;
                cout << " Entrer la position en temps de l'impulsion  :"; cin >> tp;
                cout << " Entrer la largeur en temps de l'impulsion :"; cin >> dtp;
        }
        dtp=1/(dtp*dtp) ;
        cout << " Variation de vitesse moyenne ? (o, n) :"; cin >> yes;
        marche = (yes == "o");
        if (marche){
                cout << " Entrer le nb de Reynolds final :"; cin >> ref;
                cout << " Entrer la position en temps de la variation  :"; cin >> tpm;
                cout << " Entrer la largeur en temps de la variation  :"; cin >> dtpm;
        }
        ampm=ref/reynolds-1 ;
        cout << " Perturbation sinusoidale ? (o, n) :"; cin >> yes;
        sinus = (yes == "o");
        if (sinus){
                cout << " Entrer l'amplitude de la perturbation (<1) :"; cin >> amp2;
                cout << " Entrer la frequence de la perturbation  :"; cin >> omega;
        }
        omega=omega*2*pi ;
}
// definition du pas de temps
real dt=0.1;
bool courantnotok=true;
while (courantnotok){
cout << " Entrer le pas de temps : (<1) "; cin >> dt;
cout << "nombres de Courant sur long. min : " << dt/minlength << " sur long. max : " << dt/
maxlength <<  " sur long. moyenne : " << dt/avlength;
```

```
cout << " pas de temps de correct ? (o,n) "; cin >> yes;
if (yes == "o") {
        courantnotok=false;
        }
}
// 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;
// definition du probleme de Navier-Stokes
// la condition aux limites sur la face d'entree dans le canal (d) prend en compte la perturbation
de vitesse
// dependant du temps (=dt*i)
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*(1+ampm*0.5*(1+tanh((dt*i-tpm)/dtpm)))+amp1*exp(-(dt*i-tp)*(dt*i-tp)*dtp)
+amp2*sin(omega*dt*i),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  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, saturation,
luminosite) qui varient entre 0 et 1
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 l'evolution temporelle de la vitesse derriere l'obstacle
string uvstfile="u_vs_t_b"+bloc+"_re"+reynolds+".txt" ;
ofstream uvst(uvstfile,append);
// fichier de sortie pour le'volution 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 de la position du point de mesure de la vitesse
 real xmes,ymes,xrel,yrel ;
cout << " Position x du point de mesure de vitesse par rapport a l obstacle :"; cin >> xrel;
xmes=xob+xrel;
cout << " Position y du point de mesure de vitesse par rapport a l obstacle :"; cin >> yrel;
ymes=yob+yrel;
// definition des parametres pour les sorties graphiques
//string store_dir,base_dir ;
bool calcf ;
real fx,fy ;
cout << "calcul des forces sur l'obstacle ? (o, n)"; cin >> yes;
calcf = (yes == "o");
bool sortiep, sortiev, plotv, zoom, sortier;
cout << "enregistrement des champs de pression ? (o, n)"; cin >> yes;
sortiep = (yes == "o");
cout << "enregistrement des lignes de courant ? (o, n)"; cin >> yes;
sortiev = (yes == "o");
cout << "enregistrement du champ de vorticite ? (o, n)"; cin >> yes;
sortier = (yes == "o");
//cout << "affichage du champ de vitesse ? (o, n)"; cin >> yes;
//plotv = (yes == "o");
cout << " sorties graphiques sur une partie seulement du domaine de calcul ? (o, n)"; cin >>
yes;
zoom = (yes == "o");
real llx,lly,urx,ury;
llx=0. ;
lly=0. ;
urx=lcanal ;
ury=bloc ;
if (zoom) {
        cout << "coordonnee x du point inferieur gauche :"; cin >> llx;
        cout << "coordonnee y du point inferieur gauche :"; cin >> lly;
        cout << "coordonnee x du point superieur droit :"; cin >> urx;
        cout << "coordonnee y du point superieur droit :"; cin >> ury;
}
// sauvegarde des parametres de calcul dans un fichier
string paramfile="parametres_b"+bloc+"_rei"+reynolds+"_ref"+ref+".txt" ;
ofstream param(paramfile,append);
param << "blocage " << "," << bloc << "\n";
param << "longueur totale " << "," << lcanal << "\n";
param << "reynolds initial" << "," << reynolds << "\n";
param << "reynolds final" << "," << ref << "\n";
param << "temps de chgt de re" << "," << tp << "," << "largeur de marche" << "," << dtp <<
"\n" ;
```

```
param << "pas de temps " << "," << dt << ", nb iterations" << "," << nbiter <<"\n";
param << "bounding box " << "," << llx << "," << lly << "," << urx << "," << ury <<"\n";
param << "point de mesure ," << xmes << "," << ymes <<"\n" ;
// definition des vecteurs pour l'affichage force et vitesse en fonction du temps
real[int] temps(nbiter),vitx(nbiter),vity(nbiter),fox(nbiter),foy(nbiter),pp(nbiter) ;
for (i=0;i<nbiter;i++){
	temps[i]=dt*i;
	vitx[i]=0 ;
	vity[i]=0 ;
	fox[i]=0 ;
	foy[i]=0 ;
	pp[i]=0;
}
// off we go, iteration en temps
for (i=0;i<nbiter;i++){
	up1=u1;
	up2=u2;
	NS;
	x=xmes ; y=ymes ;
	uvst << i << "," << u1 << "," << u2 << "\n";
	vitx[i]=u1;
	vity[i]=u2;
	pp[i]=p;
	// 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 << "," << fx << "," << fy << "\n";
	}
	// affichage des composantes x et y de la vitesse au point de mesure
	//plot(cmm="Composantes de vitesse au point "+xmes+","+ymes+"  iteration no "+i+"
temps "+dt*i,[temps,vitx],[temps,vity],[temps,pp]);
	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]],ps="p_b"+bloc+"_re"+reynolds+"_t"+i+".eps");
			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]],ps="stream_b"+bloc+"_re"+reynolds+"_t"+i+".eps");
			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,ury]],ps="vor_b"+bloc+"_re"+reynolds+"_t"+i+".eps");
                plot(cmm="iteration no "+i+" temps
"+dt*i,vor,viso=vorval,hsv=colors,fill=1,bb=[[llx,lly],[urx,ury]]);
                omegatkfile="vor_b"+bloc+"_re"+reynolds+"_t"+i+".vtk";
                savevtk(omegatkfile,th,vor,dataname="vorticite");
            }
//          if (plotv) {
                // affichage du champ de vitesse
//              plot (cmm="iteration no "+i,[u1,u2],bb=[[llx,lly],[urx,ury]]);
//          }
        }
        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 ;
}


// dans le cas d'une perturbation sinusoidale calcul de la valeur rms de la fluctuation transverse
de vitesse
// on ne prend pas en compte le regime transitoire au debut du calcul
if (sinus) {
        int nstart ;
        nstart=floor(lcanal/dt) ;
        real uyrms=0.,uyav=0. ;
        for (i=nstart;i<nbiter;i++){
            uyav=uyav+vity[i] ;
        }
        uyav=uyav/(nbiter-nstart) ;
        for (i=nstart;i<nbiter;i++){
            uyrms=uyrms+(vity[i]-uyav)*(vity[i]-uyav) ;
        }
        uyrms=sqrt(uyrms/(nbiter-nstart));
        cout << "valeur quadratique moyenne de uy :" << uyrms ;
}
```