```
// Calcul de l'ecoulement autour d'un obstacle profile place dans une cuve circulaire
// la largeur de l'obstacle est fixee e 1 ainsi que la vitesse moyenne
// on peut faire varier le rapport de cuveage de l'ecoulement
// on etudie la reponse de l'ecoulement e diverses
// perturbations de la vitesse moyenne en entree
// cette procedure est similaire e l'etude experimentale de
// Provansal et al. (J. Fluid Mech. 1987)
//
//definition des parametres geometriques
real cuveD, cuve, DistD, Dist, mobileD, mobile;
cout << " Entrer le rayon de la cuve (en cm) :"; cin >> cuveD;
assert (cuveD>1);
// la distance du mobile e l'axe est dans la cuve
cout << " Entrer la distance du mobile a l'axe  (en cm):"; cin >> DistD;
assert (DistD>0);
assert (DistD<cuveD);
// la taille du mobile
cout << " Entrer le rayon du mobile (en cm) :"; cin >> mobileD;
assert (mobileD>0);
assert (mobileD<(cuveD-DistD)/2);
//adimensionnement par la taille du mobile
mobile = 1;
Dist = DistD/mobileD;
cuve = cuveD/mobileD;
// definition du nombre minimal de mailles en fonction de lcanal et cuve
int ncuve,nmobile ;
//ncuve=floor(cuve);
//nmobile=floor(mobile*pi*2*10);
ncuve=30;
nmobile=50;
// limites du canal
// frontieres decrites dans le sens trigo direct

border a(t=0,2*pi){ x=cuve*cos(t); y=cuve*sin(t);}
border b(t=0,2*pi){ x=Dist+mobile*cos(t); y=mobile*sin(t);}

// obstacle
// frontiere decrite dans le sens trigo inverse
int C1=99,top=97,bottom=96 ;
int i=0;
int n=2;
int j=0;
int k=0;
cout << " Entrer la resolution du maillage (>1) :"; cin >> n;
mesh th = buildmesh(a(ncuve*n)+b(-nmobile*n));
plot (th,wait=1,ps="maillage.ps") ;


// 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=cuve) = 0 : vitesse moyenne <u1> = 1



// calcul de la solution de Stokes pour initialiser le champ de vitesse
func g = y/Dist; func f = -x/Dist;
solve Stokes ([u1,u2,p],[v1,v2,q],solver=Crout) =
    int2d(th)( ( dx(u1)*dx(v1) + dy(u1)*dy(v1)
             +  dx(u2)*dx(v2) + dy(u2)*dy(v2) )
             + p*q*(0.0000001) // terme de stabilisation
             - p*dx(v1)- p*dy(v2)
             + dx(u1)*q+ dy(u2)*q
           )
+ on(a,u1=g,u2=f)
+ on(b,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(a,psi=0);

streamlines ;
// affichage de la solution de Stokes
plot(psi,nbiso=50,wait=1);
plot(p,fill=1,wait=1);
// nu est l'inverse du nb de Reynolds
// attention e la definition de Reynolds.
real reynolds=10.;
cout << "Entrer le Reynolds:"; cin >> reynolds;

real nu=1./reynolds;

string yes ;

// definition du pas de temps
real dt=0.01;
cout << " Entrer le pas de temps (<0.1) :"; cin >> dt;
cout << "Un tour par " + int(3.14/dt) +" iterations" + "\n";
real alpha=1/dt;
// definition du probleme de Navier-Stokes
Xh up1,up2;
problem NS (u1,u2,p,v1,v2,q,solver=Crout,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*(0.00000001)
- 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(b,u1=0,u2=0)
+ on(a,u1=g,u2=f);
//
```

```
// definition des valeurs pour la visualisation de la vorticite de -vmax e vmax
real[int] vorval(51);
real vmax=50;
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 force sur l'obstacle
string fvstfile="f_vs_t_b"+Dist+"_re"+int(reynolds)+".txt" ;
ofstream fvst(fvstfile,append);
// nombre d'iterations en temps
int nbiter, nuv ;
cout << "␣Entrer␣le␣nombre␣d␣iterations␣:"; cin >> nbiter;
cout << "␣Entrer␣le␣nombre␣d'iteration␣entre␣chaque␣fichier␣txt␣:"; cin >>nuv;
// definition de la position du point de mesure de la vitesse

 real xmes,ymes ;
xmes=8;
ymes=-4;


// 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 << "affichage␣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,zone,zoneD;
llx=-cuve ;
lly=-cuve ;
urx=cuve ;
ury=cuve ;
```

```
cout << "␣taille␣de␣la␣zone␣ou␣recuperer␣les␣donnees␣(en␣cm␣depuis␣le␣centre␣de␣la␣cuve)";
 cin >> zoneD;
zone = zoneD*1.0/mobileD;
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"+Dist+"_re"+int(reynolds)+".txt" ;
ofstream param(paramfile,append);
param << "diametre␣cuve␣" << "," << cuve << "\n";
param << "distance␣au␣centre␣" << "," << Dist << "\n";
param << "taille␣objet␣" << "," << mobile << "\n";
param << "reynolds␣" << "," << int(reynolds) << "\n";
param << "pas␣de␣temps␣" << "," << dt << ",␣nb␣iterations" << "," << nbiter <<"\n";
param << "bounding␣box␣" << "," << llx << "," << lly << "," << urx << "," << ury <<"\n";

// sauvegarde des coordonnees des points de mesures
real Npoints=1.;
cout << "nombre␣de␣points␣de␣mesure␣:"; cin >> Npoints;
//string coordfile="coordonnees_b"+Dist+"_re"+int(reynolds)+"Npoints"+Npoints+".txt" ;
//ofstream coord(coordfile,append);
//coord << x << "," << y << "\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) ;
for (i=0;i<nbiter;i++){
        temps[i]=dt*i;
        vitx[i]=0 ;
        vity[i]=0 ;
        fox[i]=0 ;
        foy[i]=0 ;
}
// off we go
real up12, up22,circ,Rj,pas,pas2 ;
int millier, cent, dec, unit;
for (i=0;i<nbiter;i++){
// numerotation du fichier
        millier = int(i/1000);
        cent = int((i-millier*1000)/100);
        dec = int((i-millier*1000-cent*100)/10);
        unit = int((i-millier*1000-cent*100-dec*10));
        up1=u1;
        up2=u2;
        NS; // resolution du probleme de NS avec les conditions initiales precedentes
        if ( !(i % nuv)) {
        string uvstxfile="ux-Re"+int(reynolds)+"dt"+1/dt+"-"+"rB"+int(cuve)+"_"
        +millier+cent+dec+unit+".txt" ;
        ofstream uvstx(uvstxfile,append);
        uvstx << "victor␣est␣" + "\n";
        uvstx << "un␣caca" + "\n";
        uvstx << "x␣" << "y␣" << "u␣" << "v" << "\n";
                for (k=0; k<Npoints; k++){ // exportation des donnees du champs de vitesse
                        for (j=0; j<Npoints; j++){
                                x=2*k*zone*1.0/Npoints-zone;
                                y=2*j*zone*1.0/Npoints-zone;
                                uvstx << x << "␣" << y << "␣";
```

```
                                uvstx << u1 << "␣" << u2 << "\n";
                        }
                }
        }
        x = xmes;
        y = ymes;
        vitx[i]=u1;
        vity[i]=u2;
        if (calcf){ // exportation de la force resultante sur l'obstacle
                sigma11=2*nu*dx(u1)-p;
                sigma22=2*nu*dy(u2)-p ;
                sigma12=nu*(dy(u1)+dx(u2));
                fx=-int1d(th,b) (sigma11*N.x+sigma12*N.y);
                fy=-int1d(th,b) (sigma12*N.x+sigma22*N.y);
                fvst << i << "␣" << fx << "␣" << fy << "\n";
        }
        // affichage des composantes x et y de la vitesse au point de mesure
        plot(cmm="iteration␣no␣"+i,[temps,vitx],[temps,vity]);
        if ( !(i % nuv)) {
                vor = dy(u1)-dx(u2);
                if (sortiep) {
                        plot(cmm="iteration␣no␣"+i,p,fill=1,bb=[[llx,lly],[urx,ury]],
                        ps="p_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+".eps"
                }
                if (sortiev) {
                        // calcul de la fonction de courant
                        streamlines ;
                        plot (cmm="iteration␣no␣"+i,psi,nbiso=50,bb=[[llx,lly],[urx,ury]],
                        ps="stream_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+"
                }
                if (sortier) {
//                      plot(cmm="iteration no "+i,vor,viso=vorval,hsv=colors,fill=1,bb=[[l
ps="vor_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+".eps");
                        plot(cmm="iteration␣no␣"+i,vor,fill=1,bb=[[llx,lly],[urx,ury]]
                        //,ps="vor_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+"
                        );
                }
                if (plotv) {
                        // affichage du champ de vitesse
                        plot (cmm="iteration␣no␣"+i,[u1,u2],bb=[[llx,lly],[urx,ury]],
                        ps="vit_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+".ep
                }
        }
        if (!(i % (20*nuv))) {
                vor = dy(u1)-dx(u2);
//                      plot(cmm="iteration no "+i,vor,viso=vorval,hsv=colors,fill=1,
//                      bb=[[llx,lly],[urx,ury]],
//                      ps="vor_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+".ep
                        plot(cmm="iteration␣no␣"+i,vor,fill=1,bb=[[llx,lly],[urx,ury]],
                        ps="vor_b"+Dist+"_re"+int(reynolds)+"_t"+millier+cent+dec+unit+".ep
        }
}
```