# EDP - Final Project

Réaliser par : Dinar Amine

Aout 8, 2016

# Table des matières

# I. Introduction

À la fin de la première année de Master Calcul Scientifique et Mathématiques de L'Information à l'Université de Strasbourg il y a un stage de 2 mois a effectué. Celui-ci permet de découvrir les opportunités possibles avec ce master, mais aussi d'appliquer ses connaissances acquises durant l'année.

J'ai eu la chance de réaliser mon stage au siens de l'université de Strasbourg, encadre par Mr Prud'homme Christophe.

# II. Le Laplacien

## 1. Présentation

L'opérateur laplacien, est l'opérateur différentiel défini par l'application de l'opérateur gradient suivie de l'application de l'opérateur divergence. Intuitivement, il combine et relie la description statique d'un champ (décrit par son gradient) aux effets dynamiques (la divergence) de ce champ dans l'espace et le temps.

On considère le problème suivant : soit Ω un cercle unité tel que $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_R$

$$-\Delta u = fu = g \; sur \; \Gamma_D$$

$$u = g \; sur \; \Gamma_D$$

$$\frac{\partial u}{\partial n} = m \; sur \; \Gamma_N$$

$$u + \frac{\partial u}{\partial n} = \text{l sur } \Gamma_R$$

## 2. Maillage

```
border D (t=0, pi /2) {x=cos ( t ) ; y=s i n ( t ) ; l a b e l =1;};
border N (t=pi /2 , pi ) {x=cos ( t ) ; y=s i n ( t ) ; l a b e l =2;};
border R (t=pi , 2_ pi ) {x=cos ( t ) ; y=s i n ( t ) ; l a b e l =3;};

mesh Th=bui ldmesh (D(2 5) + N( 25) + R( 50 ) ) ;
```

## 3. Formulation Variationnelle

On trouve la formulation variationnelle à l'aide de la formule de Green :

$$\int_\Omega \Delta u * u = \int_{\partial\Omega} \frac{\partial u}{\partial \gamma} * v - \int_\Omega \nabla u \nabla v$$

On s'intéresse au deuxième terme, regardons qu'est-ce que les conditions aux bords nous donnent :

$$\Gamma_D : \int_\Omega \nabla u \nabla v = \int_\Omega f v$$

$$\Gamma_N : \int_\Omega \nabla u \nabla v - \int_{\partial\Omega N} \mu v = \int_\Omega f v$$

$$\Gamma_R : \int_\Omega \nabla u \nabla v - \int_{\partial\Omega R} l v + \int_{\partial\Omega R} u v = \int_\Omega f v$$

Finalement, on obtient :

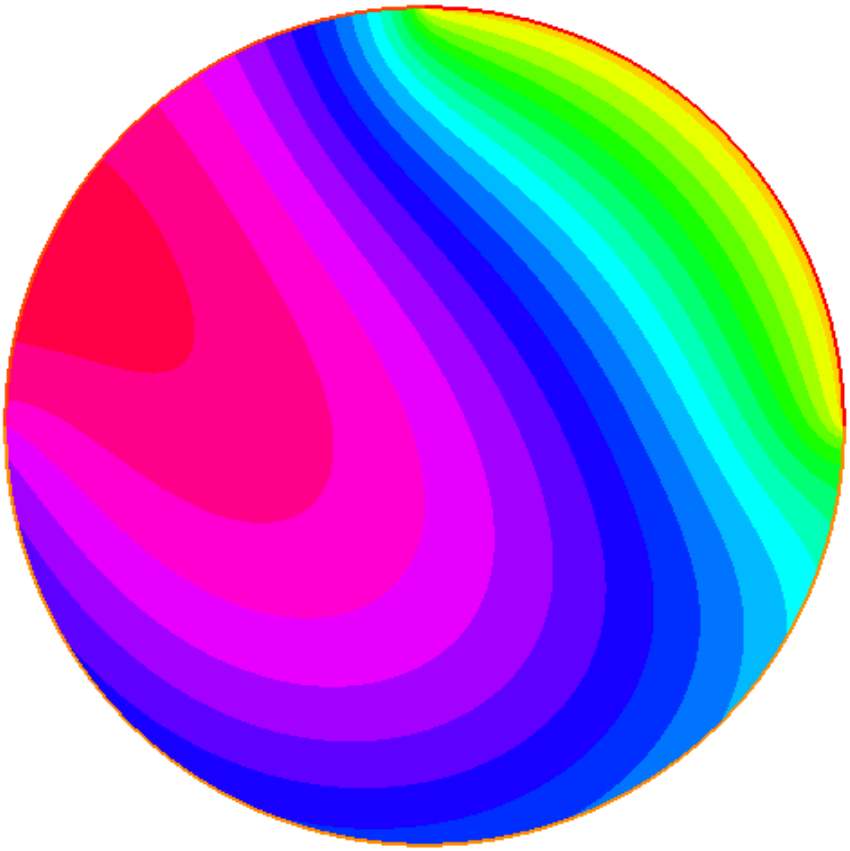
$$\int_{\Omega} \nabla u \nabla v \; - \; \int_{\partial \Omega N} \mu v - \int_{\partial \Omega R} lv + \int_{\partial \Omega R} uv = \; \int_{\Omega} fv$$

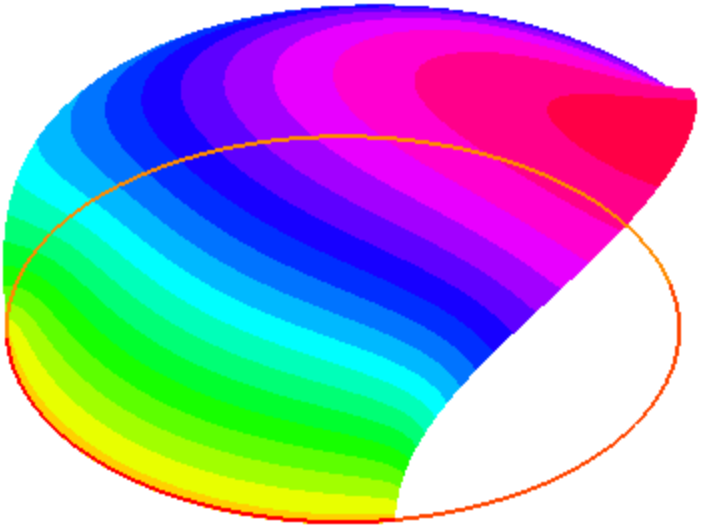
```
fespace Vh(Th, P1) ;
Vh v , u ;

func f =1;
func g = 0 ;
func l = 0 ;
func m = 0 ;

problem laplace (u , v , solver = LU, tgv = 1e30 )=
                          int2d (Th) ( dx (u) _dx ( v )+dy (u) _dy ( v ) ) // bilinear form
                         -int2d (Th) ( f _v )
                         -int1d (Th, 2 ) ( g_v )
                         -int1d (Th, 3 ) ( l _v )
                         +int1d (Th, 3 ) (u_v )
                         +on ( 1 , u=g ) ;
laplace ;
```

## 4. Valeurs données

On prend u(x, y) = sin(πx)cos(πy). Calculons f, g, m, l pour que u soit la solution.

- f On calcule le laplacien (somme des secondes dérivés) pour u donnée, et trouve que f = −2π2u
- g Gratuit: $g = u$.
- m Pour trouver m on doit trouver le gradient de u, qui est égale à (πcos(πx)cos(πy); -πsin(πx)sin(πy)) et trouver son produit scalaire avec (x; y). Finalement, on a : m = πxcos(πx)cos(πy)- πysin(πx)sin(πy).
- L Le dernier : l = u + m.
  Laplacien en 2D et 3D:





## 5. Errons

On a ajouté une boucle qui permet voir plusieurs estimations. Avec m trouvé avant je n'arrive pas à calculer les erreurs, mais voici comme il faut faire "en théorie" :

```
// ex1 . edp
border D( t=0, pi /2) {x=cos ( t ) ; y=s i n ( t ) ; l a b e l =1;};
border N( t=pi /2 , pi ) {x=cos ( t ) ; y=s i n ( t ) ; l a b e l =2;};
border R( t=pi , 2_ pi ) {x=cos ( t ) ; y=s i n ( t ) ; l a b e l =3;};
int n=5;

//Ces boucles montre evolution de l'erreur avec la taille différente de meshstep

for ( r e a l r=1; r<n ; r+=1){

int a=50;
mesh Th=bui ldmesh (D( a * r ) + N( a*r ) + R( a*r*2) ) ;
// plot (Th, wai t=1) ;
fespace Vh(Th, P1) ;
Vh v , u , un ;
func f =1;
func g = 0 ;
func l = 0 ;
func m = 0 ;

//solution exacte

func uexact=sin ( pi * x ) * cos ( pi * y ) ;
un=g ;

//Les dérivées partielles

Vh dxue=pi * cos ( pi * x ) * cos ( pi * y ) ;
Vh dyue=-pi * sin ( pi * x ) * s i n ( pi * y ) ;
solve laplace (u , v , s o l v e r=LU, tgv=1e30 )=
int2d (Th) ( dx (u) * dx ( v )+dy (u) * dy ( v ) )
                                          -int2d (Th) ( f * v )
                                         -int1d (Th, 2 ) ( g * v )
                                         -int1d (Th, 3 ) ( l * v )
                                         +int1d (Th, 3 ) (u * v )
                                            +on ( 1 , u=g ) ;

laplace ;
plot (u , wai t=1, fill = 1) ;

//Norme et erreur d'estimation

real L2 = s q r t ( int2d (Th) (u^2) ) ;
real L2er r = s q r t ( int2d (Th) ( ( u - uexact ) ^2) ) /L2 ;

real H1 = s q r t (L2^2+int2d (Th) ( dx (u)^2+dy (u) ^2) ) ;
real H1err = s q r t ( L2er r^2
                                  +int2d (Th) ( ( dx (u) - dxue ) ^2) /L2
                                 +int2d (Th) ( ( dy (u) - dyue ) ^2) /L2 ) ;
cout<<"With D="<<a _r<<" N="<<a_r<<" R="<<2 * a * r<<endl ;
```

```
cout<<"L2 error :"<<L2err<<endl ;
cout<<"H1 error :"<<H1err<<endl ;
}
```

## III. Convergence des problèmes mixtes

On considère l'équation du Stokes : $-\mu\Delta u + \nabla p = f$ avec $\nabla u \circ = 0$

L'équation de Stokes est utilisée pour des fluides visqueux qui s'écoulent lentement en un lieu étroit ou autour d'un petit objet, où les effets visqueux dominent sur les effets inertiels. Cette équation peut être dérivée à partir de Navier-Stokes en négligeant les termes de convection ($u \circ \nabla u$). Si on néglige également le terme du temps, on se retrouve dans le cas stationnaire. Les fluides correspondants ont un nombre de Reynolds très petit << 1.

La formulation variationnelle est donnée par :

$$\int_\Omega \mu \nabla u : \nabla v - p \nabla \circ v + \int_{\partial\Omega} (-\mu \nabla u + pI) n v = \int_\Omega f v$$

$$-\int_\Omega \nabla \circ u q = 0$$

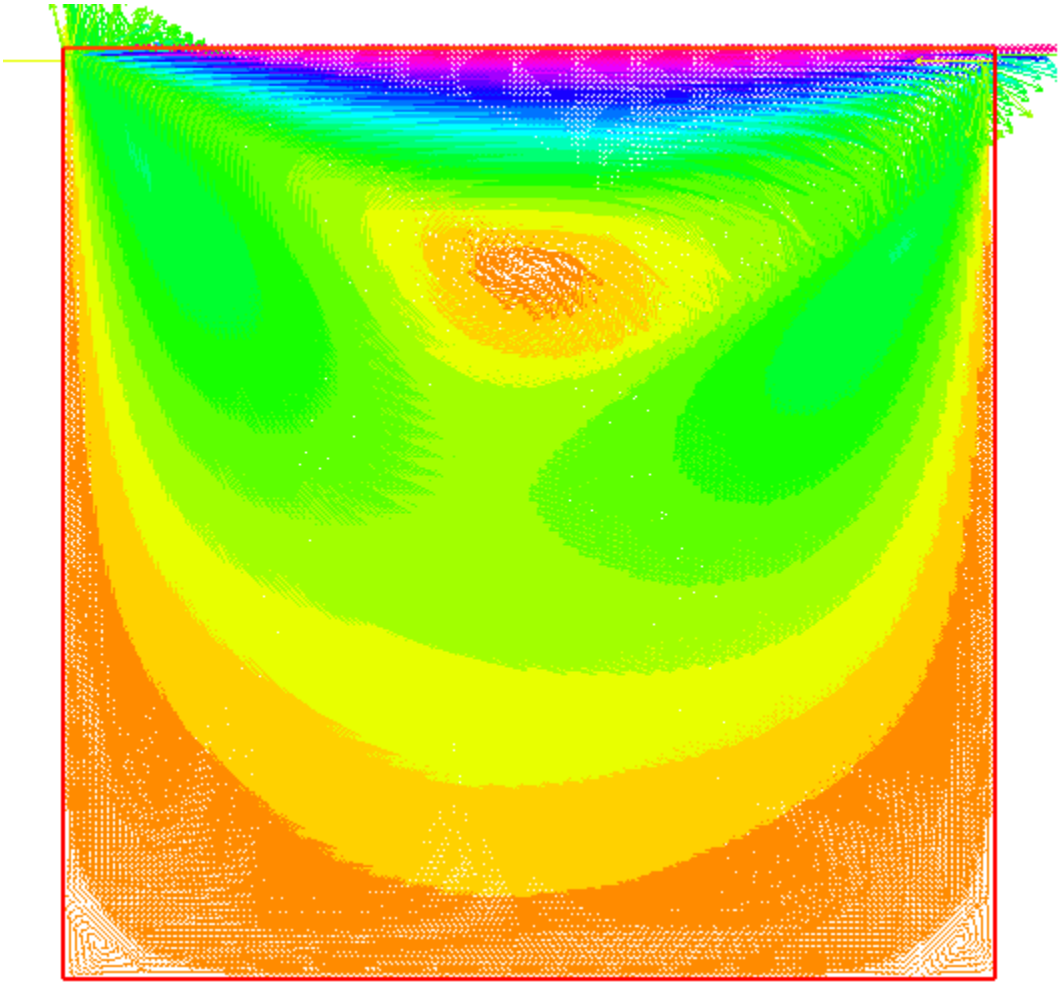
Le deuxième terme et le seconde intégral sont symétriques. Le troisième terme est appelé tenseur des contraintes.

Regardons sur le code FreeFem++:

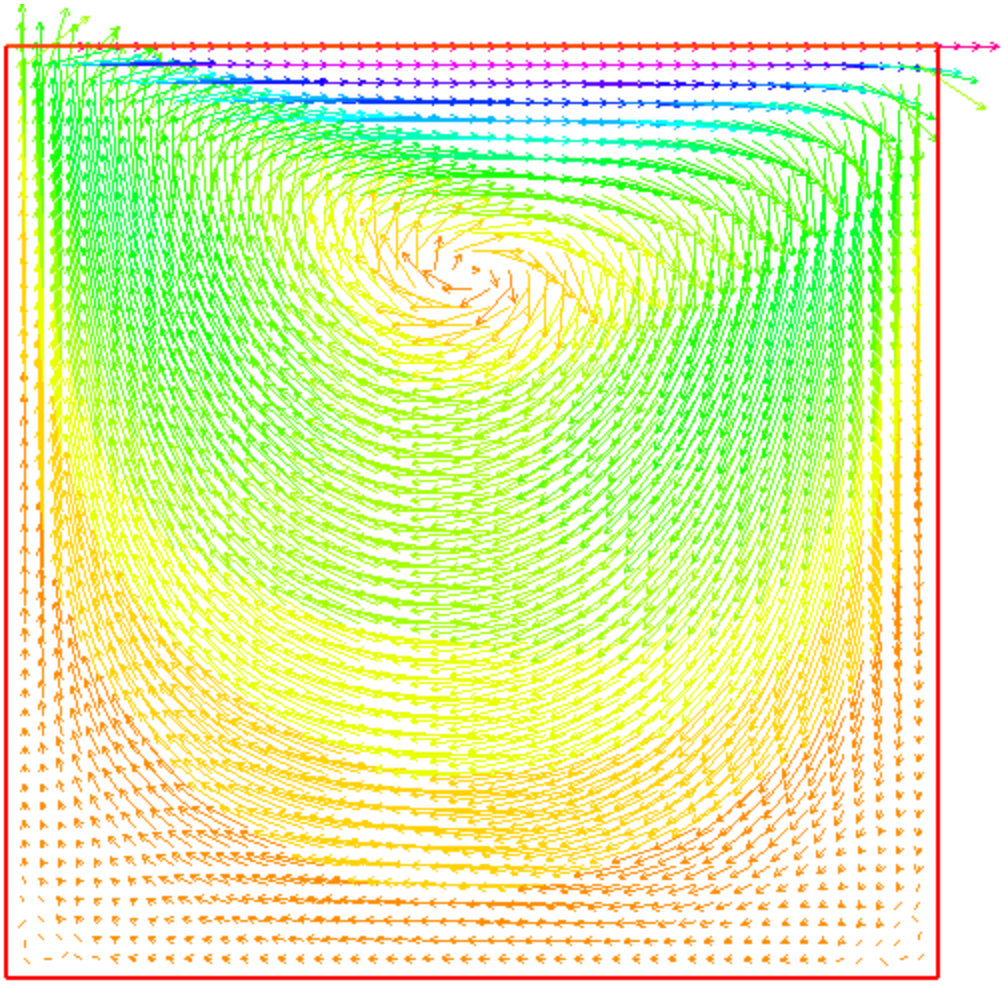
```
int [ i n t ]  l l =[ 1 , 1 , 2 , 1 ] ;
mesh Th = squar e ( 5 0 , 5 0 , label = l l ) ;
// l l : bottom , right , top , left boundary labels
fespace Vh(Th , [ P1 ,P1 ,P1 ] ) ; / / u1 , u2 , p

macro Grad (u) [ dx (u) , dy (u) ] //EOM
macro div ( u1 , u2 ) ( dx ( u1 )+dy ( u2 ) ) //EOM

Vh [ u1 , u2 , p ] , [ v1 , v2 , q ] ;
real nu=1, alpha=0;
real eps= 1e-10;
solve Pb1 ( [ u1 , u2 , p ] , [ v1 , v2 , q ] ) =
                              int2d (Th) ( (Grad ( u1 ) ' * Grad ( v1 ) )
                                    +(Grad ( u2 ) ' * Grad ( v2 ) )
                                          -p * div ( v1 , v2 )
                                          -q * div ( u1 , u2 )
                                             + eps * p * q
)

       + on ( 1 , u1 = 0 , u2=0)
       + on ( 2 , u1 = 1 , u2=0) ;

plot ( [ u1 , u2 ] , wai t=1, fill =1, value=1) ;
```

On teste les polynômes des degrés différents, les polynômes pour la vitesse doivent avoir plus grands degrés que les polynômes pour la pression, sinon on aura des oscillations (ou l'image ne sera pas générée). Regardons sur les images obtenues :

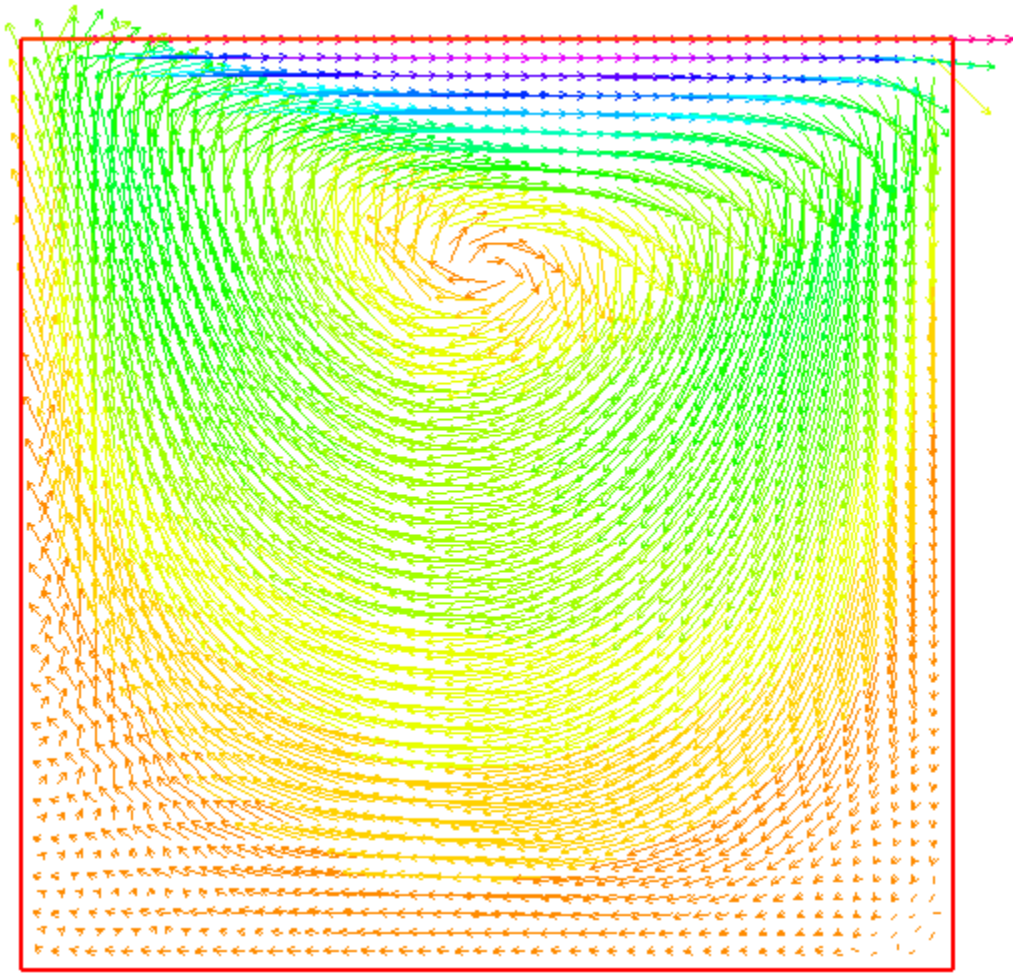
- P2/P1 :



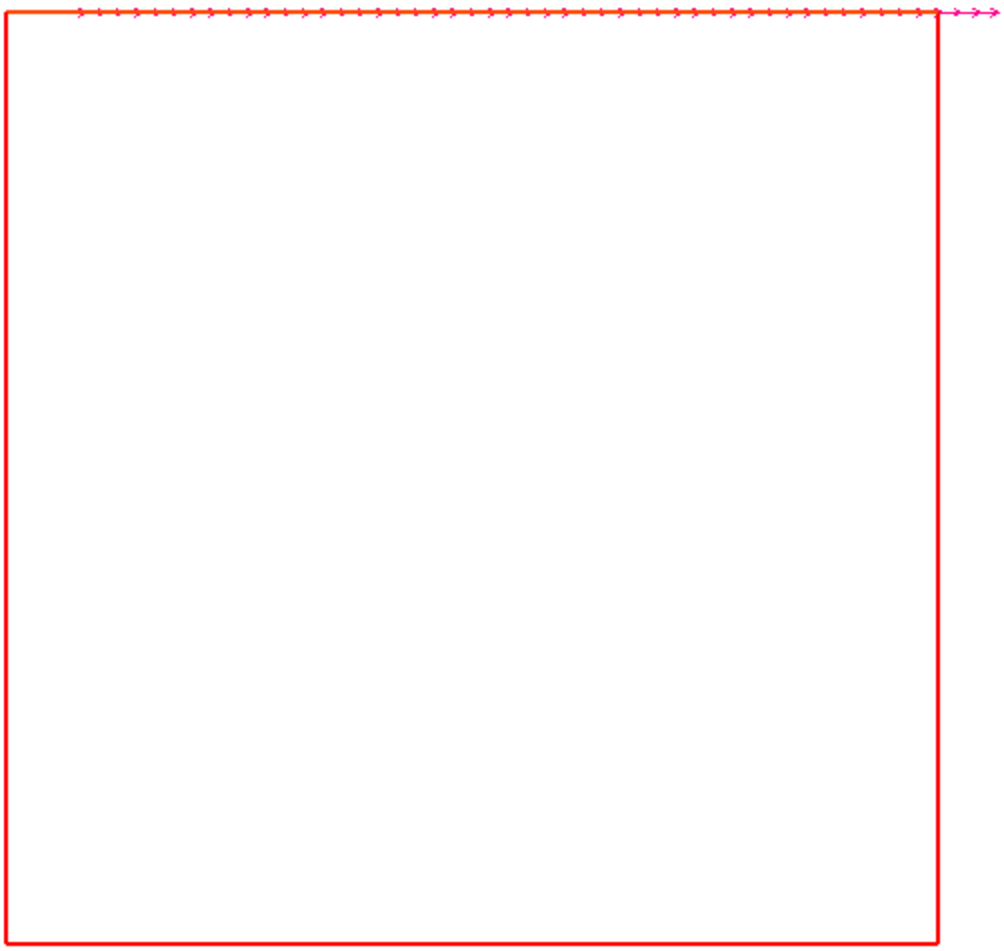
- P1b/P 1 :



- $P1 \triangleleft P1$ :



- P1/P0 :



# IV. Stabilisation

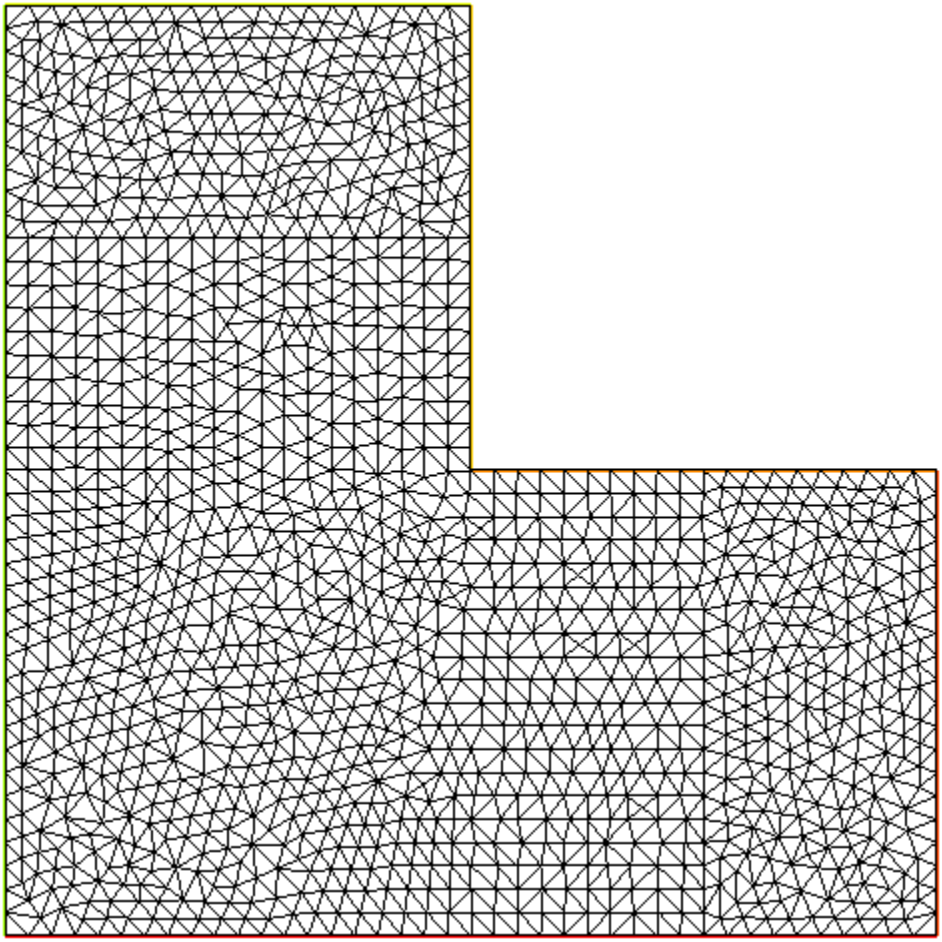
On s'intéresse aux méthodes de stabilisation pour ce type de problème :

$$-€\Delta u + \beta \circ \nabla u + \mu u = f, \nabla \circ \beta = 0, u = 0 \text{ sur } \partial\Omega$$

## 1. L-shaped

On commence par maillage :

```
border a ( t =0 ,4.0) {x=t ; y = 0 ; l a b e l =1;};
border b( t =0 ,2.0) {x=4; y = t ; l a b e l =2;};
border c ( t =0 ,2.0) {x=4-t ; y = 2 ; l a b e l =3;};
border d( t =2 . 0 , 4 . 0 ) {x=2; y = t ; l a b e l =4;};
border e ( t =2 . 0 , 4 . 0 ) {x=4-t ; y =4; l a b e l =5;};
border f ( t =0 ,4.0) {x=0; y =4-t ; l a b e l =6;};

mesh Th=bui ldmesh ( a (4 0)+b (2 0)+c (20 )+d ( 20 )+e ( 20 )+f ( 40) ) ;
```



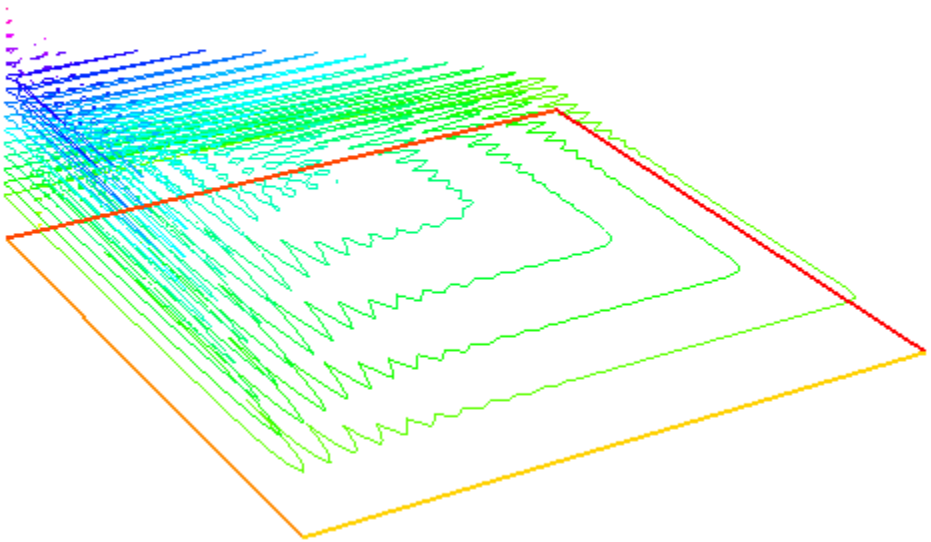
## 2. GALS et SUPG

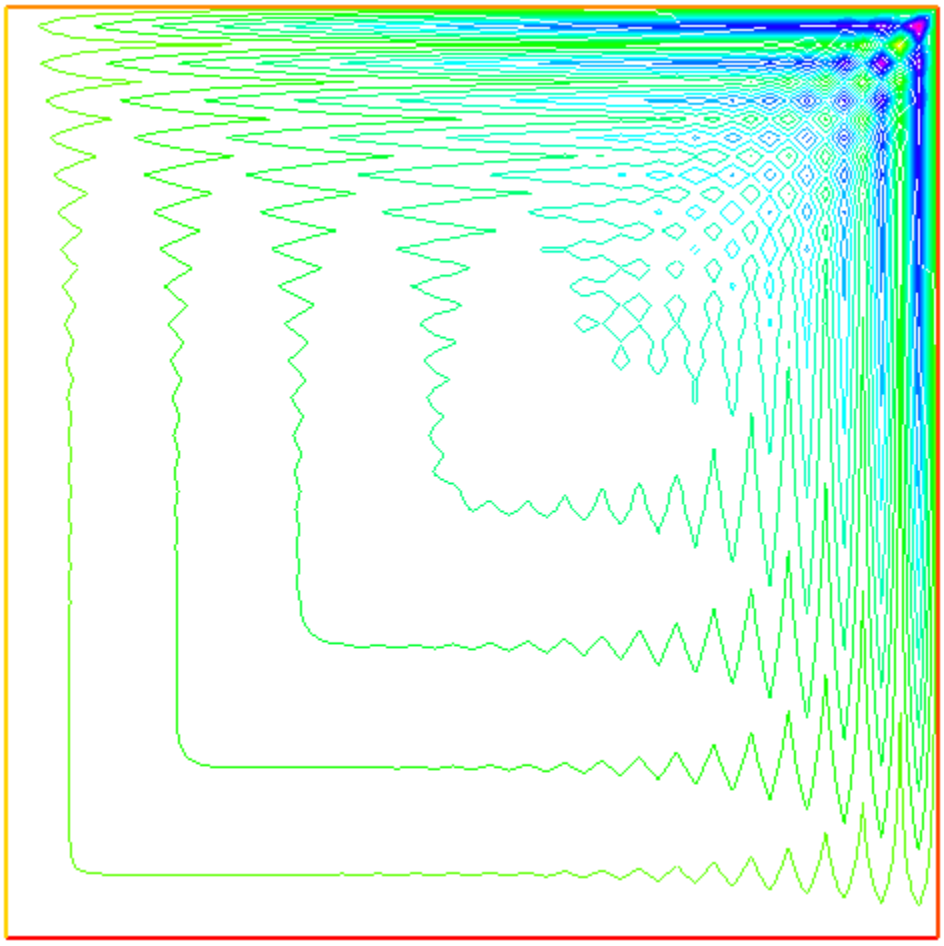
Présentation du SUPG :

```
mesh Th = squar e ( 5 0 , 5 0 ) ;

plot (Th, wai t=1) ;

fespace Vh(Th,P1) ;
Vh v , u ,w, vy , vx ;
func f =1;
int i =0;
real delta = 1 . 0 ;
real bx=1.0;
real by=1.0;
real eps=1e-3;
real mu=1.0;
fespace Nh(Th,P0) ;
Nh t = hTr iangl e / sqrt ( bx^2+by^2) ;
Vh h= hTr iangl e ;

problem cd (u , v , solver=GMRES, ini t=i ) =
                    int2d (Th) ( eps * ( dx (u) * dx ( v ) + dy (u) * dy ( v ) )
                        +bx * dx (u) * v+by * dy (u) * v+mu * u * v
      +delta *  (-eps * ( dxx (u)+dyy (u) )+bx * dx (u)+by * dy (u)+mu * u) * t *  ( bx_dx
```

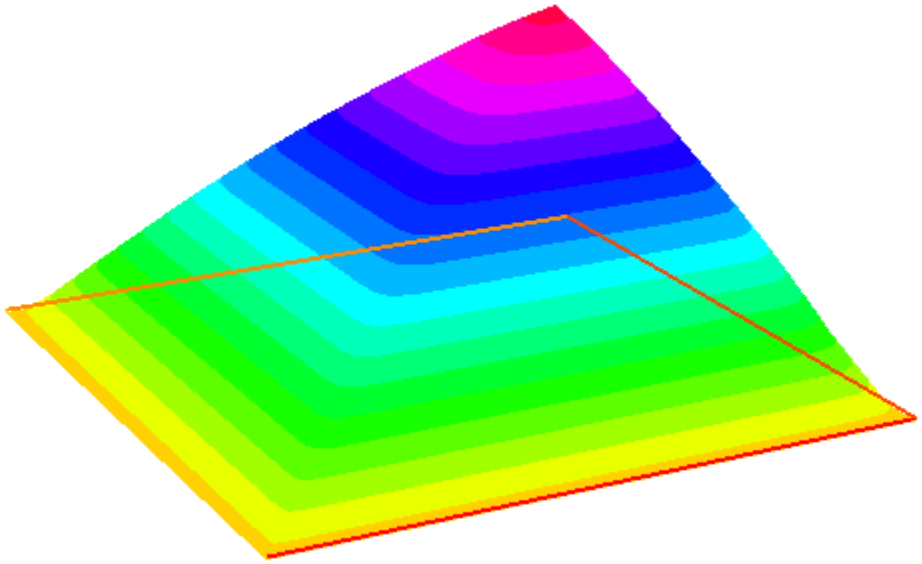
```
( v )+by_dy ( v ) ) )
                                        -int2d (Th) ( f _v )
                                    + on ( 1 , 2 , 3 , 4 , u=0) ;

cd ; // solve the problem plot (uh) ; // Resultat
plot (u , ps="CDR. pdf " , value=t rue , wai t=1) ;
```

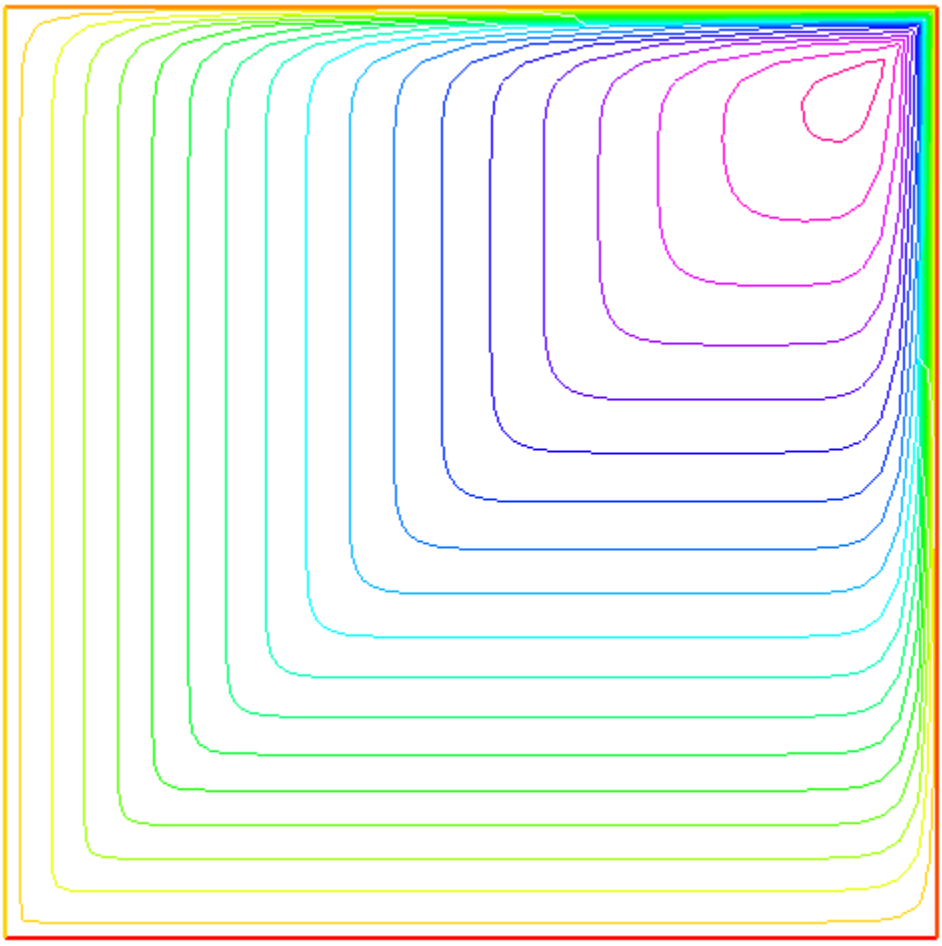
Regardons sur les résultats. Initialement on a eu des oscillations :





La stabilisation nous a permis en débarrasser :





# V. AirBus

## 1. Maillage

On pose des points sur les bords de Ω et on essaie de respecter la règle : en promenant sur les bords, Ω doit être à gauche.

```
// file avia.edp
int D = 1 ; // etiquette Dirichlet
int N = 2 ; // etiquette Neumann
int R = 3 ; //etiquette Robin
int PCBIC = 4 ; //PCB vs IC
int PCBAir=5; //PCB vs Air
int ICAir=4; //IC vs Air
int AirAi r=7; //Air vs Air

//Coordonnées des points d'extrémité pour chaque frontière
real x1=0.0 , x2=0.002 , x3=0.004 , x4=0.006 , y1=0.0 , y2=0.02 , y3=0.04 , y4=0.07 ,
y5=0.09 ,
y6=0.13;

//nous commençons par le bas, avec le dernier (troisieme) x interval
```

```
//nous montons
//nous prenons les mêmes x intervalle, mais sur le dessus


border a3 ( t = x3 , x4 ) {x=t ; y=y1 ; l a b e l =1;};
border s3 ( t = y1 , y6 ) {x=x4 ; y=t ; l a b e l =2;};
border b3 ( t = x4 , x3 ) {x=t ; y=y6 ; l a b e l =3;};

//nous marchons autour chaque horaire regtangle

border c1 ( t = y6 , y5 ) {x=x3 ; y=t ; l a b e l =7;};
border c2 ( t = x3 , x2 ) {x=t ; y=y5 ; l a b e l =4;};
border c3 ( t = y5 , y6 ) {x=x2 ; y=t ; l a b e l =5;};
border c4 ( t = x3 , x2 ) {x=t ; y=y6 ; l a b e l =3;};

border d1 ( t = y5 , y4 ) {x=x3 ; y=t ; l a b e l =4;};
border d2 ( t = x3 , x2 ) {x=t ; y=y4 ; l a b e l =4;};
border d3 ( t = y4 , y5 ) {x=x2 ; y=t ; l a b e l =4;};

border f 1 ( t = y4 , y3 ) {x=x3 ; y=t ; l a b e l =7;};
border f 2 ( t = x3 , x2 ) {x=t ; y=y3 ; l a b e l =4;};
border f 3 ( t = y3 , y4 ) {x=x2 ; y=t ; l a b e l =5;};

border g1 ( t = y3 , y2 ) {x=x3 ; y=t ; l a b e l =4;};
border g2 ( t = x3 , x2 ) {x=t ; y=y2 ; l a b e l =4;};
border g3 ( t = y2 , y3 ) {x=x2 ; y=t ; l a b e l =4;};

border h1 ( t = y2 , y1 ) {x=x3 ; y=t ; l a b e l =7;};
border h2 ( t = x2 , x3 ) {x=t ; y=y1 ; l a b e l =1;};
border h3 ( t = y1 , y2 ) {x=x2 ; y=t ; l a b e l =5;};

//lorsque le dernier rectangle est fait,
//le second fond x intervalle (x2 - x3) est effectuée par h2,
//donc nous avons seulement à ajouter le premier x
intervalle, //partie externe PCB et PCB supérieure.

border a1 ( t = x1 , x2 ) {x=t ; y=y1 ; l a b e l =1;};
border s1 ( t = y6 , y1 ) {x=x1 ; y=t ; l a b e l =2;};
border b1 ( t = x2 , x1 ) {x=t ; y=y6 ; l a b e l =2;};

mesh Th = bui ldmesh ( a3 ( 5 )+s3 ( 20 )+b3 ( 5 )
                                    +c1 (1 0)+c2 (1 0)+c3 (1 0)+c4 ( 5 )
                                    +d1 (1 0)+d2 (1 0)+d3 (1 0)
                                    +f 1 (1 0)+f 2 (1 0)+f 3 (10 )
                                    +g1 (10 )+g2 (1 0)+g3 ( 10)
                                     +h1 (1 0)+h2 ( 5 )+h3 (1 0)
                                      +a1 ( 5 )+s1 (20 )+b1 ( 5 ) ) ;

//Nous donnons le nom de chaque région

int PCE=Th( 1 , 1 ) . r e g i on ;
int air 1=Th( 3 , 0 ) . region , air 2= Th( 3 , 5 ) . region , air 3=Th( 3 , 1 0 ) . region , air
4=Th( 5 , 1 ) .
                        region ;
int IC1 = Th( 3 , 3 ) . region , IC2 = Th( 3 , 8 ) . region ;
```

```
int Q=1000000;

fespace Sh(Th,P0) ; //nous prenons une fonction discontinue
Shr eg=region ; //nous définissons la fonction discont associée à un numéro de région
plot (Th, wai t=1) ;
plot ( reg , fill =1,wai t=1, value=1,ps="region . eps ") ;

//La conductivité thermique k est évidemment différent pour chaque matériau


Sh k=2 * ( region==IC1 )+2 * ( region==IC2 ) +0.2 * ( region==PCE) +0.03 * ( region==air
1 ) +0.03* (region==air 2 ) +0.03 * ( region==air 3 ) +0.03 * ( region==air 4 ) ;

//comme nous sommes dans le cas où le temps indep , nous prenons t = 1 , si nous
mettons t = 0, alors la vitesse est 0.

// func fvel = 1-exp(-t /3) ;
real fvel = 0 . 3 ;
real D = 0 . 0 7 ;
real coef = 3/(2( ( x4-x1 ) - ( x3-x2 ) ) ) ;
real top = x-( ( ( x4-x1 )+(x3-x2 ) ) /2 + ( x2-x1 ) ) ;
real bottom= ( ( x4-x1 )-(x3-x2 ) ) / 2 ;

//maintenant, nous mettons des conditions sur la valeur de la vitesse

if (x> ( x2-x1 + x3-x2 ) ) && (x< ( x2-x1 + x4-x2 ) ) {
real vel = coef * D* (1-( top /bottom) ^2) * fvel * y ;
    }
else if (x> ( x2-x1 ) ) && (x<(x2-x1 + x3-x2 ) ) {
real vel = 0 ;
 }

fespace Vh(Th,P1) ; //pour le maillage
Vh v , u ;
problem cd (u , v , solver=LU, tgv=1e30 ) =
int2d (Th) ( k * ( dx (u) * dx ( v ) + dy (u) _dy ( v ) ) )
                                -int1d (Th, 3 ) (1100 * u * vel ) // 3 = Robin
                                        -int2d (Th) ( Q* v )
                                    +on (1 , u=0) ; //1 = Dirichlet
cd ;
```

Après avoir implémente notre code :

## 2. Navier-Stokes

```
// NS.edp

int nn=50 ;
mesh Th=squar e (nn , nn) ;

fespace Vh(Th,P1) ;
fespace Ph(Th,P1) ;
Vh u1 , u2 , v1 , v2 , up1 , up2 ;
Ph p , q ;

int i =0;
real nu=0.0001;
real dt =0.1;
real alpha=1/dt ;


macro Grad (u) [ dx (u) , dy (u) ]
macro div ( u1 , u2 ) ( dx ( u1 )+dy ( u2 ) )

func g = y * (1-y ) ;

problem Stokes ( [ u1 , u2 , p ] , [ v1 , v2 , q ] , solver=Crout ) =
int2d (Th) (Grad ( u1 ) ' * Grad ( v1 ) + Grad ( u2 ) ' * Grad ( v2 ) - div ( u1 , u2 ) * q - div (
```

```
v1 , v2 ) *
p + 1e-10 * q * p )
                                        + on ( 3 , u1=1,u2=0)
                                    + on ( 1 , 2 , 4 , u1=0,u2=0) ;

Stokes ;

for ( i =0; i <=200; i++)
{
up1=u1 ;
up2=u2 ;
Stokes ;

plot ( wai t=0, c o e f =0.2 ,p , [ u1 , u2 ] , fill =0) ;

} ;

problem Navie rStokes ( [ u1 , u2 , p ] , [ v1 , v2 , q ] , solver=Crout , ini t=i ) =
int2d (Th) (
alpha * ( u1 * v1 + u2 * v2 )
+nu * (Grad ( u1 ) '* Grad ( v1 ) + Grad ( u2 ) ' * Grad ( v2 ) - div ( u1 , u2 ) * q - div ( v1 ,
v2 ) *
p + 1e-10 * q * p )

-div ( u1 , u2 ) * q - div ( v1 , v2 ) * p + 1e-10 * q * p)
+ int2d (Th) ( -alpha * convect ( [ up1 , up2 ] ,- dt , up1 ) * v1 -alpha * conve c t ( [ up1 ,
up2 ] ,-dt
, up2 ) * v2 )
+ on ( 3 , u1=1,u2=0)
+ on ( 1 , 2 , 4 , u1=0,u2=0)

for ( i =0; i <=200; i++)
{
 up1=u1 ;
 up2=u2 ;
 NavierStokes ;
plot ( wai t=0, coef =0.2 ,p , [ u1 , u2 ] , fill =0) ;

} ;
```