Introduction à FreeFem++

M. KALLEL
LAMSIN
Ecole Nationale d'Ingénieurs de Tunis
Tunisie

avec O. Pantz

http://www.lamsin.rnu.tn mailto:moez.kallel@ipeit.rnu.tn

PLAN

- Introduction
- Les types de données
- Les fonctions
- Les boucles et les instructions de contrôle
- Les entrées / sorties
- Définir un maillage
- Résoudre une EDP
- Visualiser les résultats
- A travers des exemples

Introduction

`FreeFem++` est un logiciel (freeware) écrit en C++ développé au Laboratoire Jacques-Louis Lions de l'Université Pierre et Marie Curie [F. Hecht, O. Pironneau], porté sous Windows, Unix (Linux) et MacOs.

Il s'exécute en ligne de commande. Il permet de créer des maillages ainsi que de résoudre des équations aux dérivées partielles par des méthodes de type éléments finis.

Pour le télécharger, consulter :

http://www.freefem.org/ff++

Caractéristiques de FreeFem++ I/II (2D)

- Plusieurs types d'éléments finis : élément linéaire et quadratique, P1 discontinu et élément de Raviart-Thomas, élément vectoriel,...
- Interpolation automatique des données à partir d'un maillage sur un autre.
- Définition du problème aux limites (complexe ou réel) directement avec la forme variationnelle.
- Formulation Galerkin Discontinu.
- Discription analytique de la frontière.
- Créer des maillages, basée sur l'algorithme de Delaunay-Voronoi.
- Lire et sauver des maillages, solutions
- Adaptation des maillages, isotrope ou anisotrope.

Caractéristiques de FreeFem++ II/II (2D)

- LU, Cholesky, Crout, CG, GMRES, UMFPack solveurs linéaires ; valeurs propores et vecteurs propres calculées avec ARPACK.
- Même syntax que C++.
- Lien avec d'autres logiciels : modulef, emc2, medit, gnuplot, ...
- Plusieurs exemples sont traités : Navier-Stokes, élasticité, structure fluide, problèmes de valeurs propores, décomposition de domaine, ...

Comment utiliser FreeFem++

1/ Editer un fichier "nomfichier.edp" avec : emacs, vi, nedit, etc. Exécuter votre script :

FreeFem++ nomfichier.edp

2/ Utiliser l'interface graphique de FreeFem++ :

FreeFem++-cs

-Editer un fichier

-run

Les types de données

Variables globales

- x, y et z : les coordonnées du point courant.
- label : le numéro de référence de la frontière dans laquelle se situe le point courant, 0 sinon.
- P : le point courant.
- N : le vecteur normal sortant unitaire au point courant s'il se situe sur une frontière.
- cin, cout et endl : les commandes d'affichage/récupération de données issues de C++, utilisées avec << et >>.
- pi : le nombre π.
- true et false : les booléens.
- i : le nombre imaginaire ($\sqrt{-1}$).

Les types basiques

Les opérateurs sont comme dans C :

`+ - * / ^  //` where `a^b=` a^b

`== != < > <= >= & | //` where `a|b=` a or b, `a&b=` a and b

`= += -= /= *=`

Les entiers : Il s'agit du type int

Les réels : Il s'agit du type real

Les complexes : Il s'agit du type complex. Quelques fonctions élémentaires associées : real, imag et conj

Les chaines de caractères : Il s'agit de string

```
string toto ''this is a string''
```

Les tableaux et les matrices

Il existe deux types de tableaux, ceux avec des indices entiers, et ceux dont les indices sont des chaines de caractères. Les éléments sont de type int, real ou complex.

```
real[int] a(n);
a[3]=2;

real [int,int] A(n,m);
A[1][2]=0;

matrix A=[
[1,2,3],
[1,2,3],
[1,2,3]];
```

Les fonctions

Les fonctions prédéfinies

Les plus courantes :
cos, sin, tan, acos, asin et atan
cosh, sinh, acosh et asinh
log, log10 et exp
sqrt et $\wedge$

Définir une fonction à une variable

```
func type nom_fct(type & var)
{
instruction 1 ;
...
...
instruction n ;
return outvar ;
}
```

Les formules

Il s'agit de fonctions dépendant des deux variables d'espace **x** et **y** et sont définies à partir des fonctions élémentaires.

```
func outvar = expression(x,y) ;
func f = x+y ;
func g = imag(sqrt(z)) ;
```

Les fonctions dépendant du maillage

Il s'agit d'évaluer une formule sur les noeuds du maillage.

```
fespace espace_name(maillage, type_elements_finis) ;
func fctoutvar = expression(x,y) ;
espace_name FEfctoutvar = fctoutvar ;
```

Les boucles et les instructions de contrôle

La boucle for :

```
for (init,cond,incr) {
...
}
```

La boucle while :

```
while (cond) {
...
}
```

Les instructions de contrôle :

```
if (cond) {
...
}
else {
...
}
```

Les entrées / sorties

Pour ouvrir un fichier en lecture :

```
ifstream name(nom_fichier) ;
```

Pour ouvrir un fichier en écriture :

```
ofstream name(nom_fichier) ;
```

Lire/écrire dans un fichier >> / <<

```
{
ofstream gnu("plot.gp") ;
for (int i=0 ;i<n ;i++)
{
 gnu <<  x[i] << " " << y[i] << endl ;
}
}
```

Définir un maillage

Maillage triangulaire régulier dans un domaine rectangulaire

On considère le domaine $]x_0, x_1[\times]y_0, y_1[$. Pour générer un maillage régulier $n \times m$:

```
mesh nom_maillage = square(n,m,[x0+(x1-x0)*x,y0+(y1-y0)*y]) ;
```

Maillage triangulaire non structuré défini à partir de ses frontières

Pour définir les frontières, on utilise la commande border :

```
border name(t=deb,fin){x=x(t) ;y=y(t) ;label=num_label} ;
```

Pour définir un maillage à partir de ses frontières, on utilise buildmesh :

```
buildmesh nom_maillage=buildmesh(a1(n1)+a2(n2)+...+ak(nk)) ;
```

Exemples de maillages

1/ Un domaine carré : $]0,1[^2$

```
mesh Th1 =  square(10,10) ;                              //    boundary label:
plot(Th1,wait=1) ;                          //    1 bottom, 2 right, 3 top, 4, left
```

2/ Un domaine sous forme L : $]0,1[^2 \backslash [\frac{1}{2},1[^2$

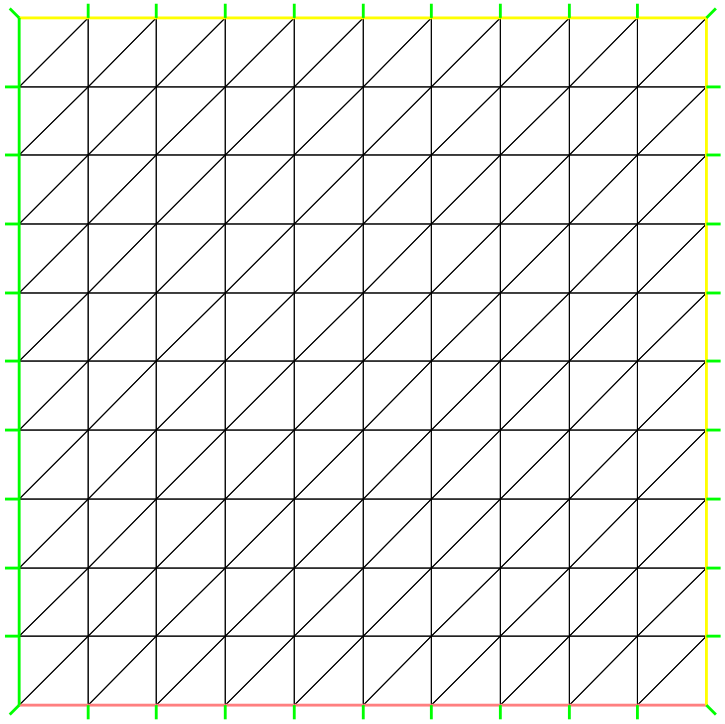
```
border a(t=0,1.0){x=t ;   y=0 ;  label=1 ;} ;
border b(t=0,0.5){x=1 ;   y=t ;  label=2 ;} ;
border c(t=0,0.5){x=1-t ; y=0.5 ;label=3 ;} ;
border d(t=0.5,1){x=0.5 ; y=t ;  label=4 ;} ;
border e(t=0.5,1){x=1-t ; y=1 ;  label=5 ;} ;
border f(t=0.0,1){x=0 ;   y=1-t ;label=6 ;} ;
plot(a(6) + b(4) + c(4) +d(4) + e(4) + f(6),wait=1) ;          //    to see the 6 borders
mesh Th2 =  buildmesh (a(6) + b(4) + c(4) +d(4) + e(4) + f(6)) ;
```

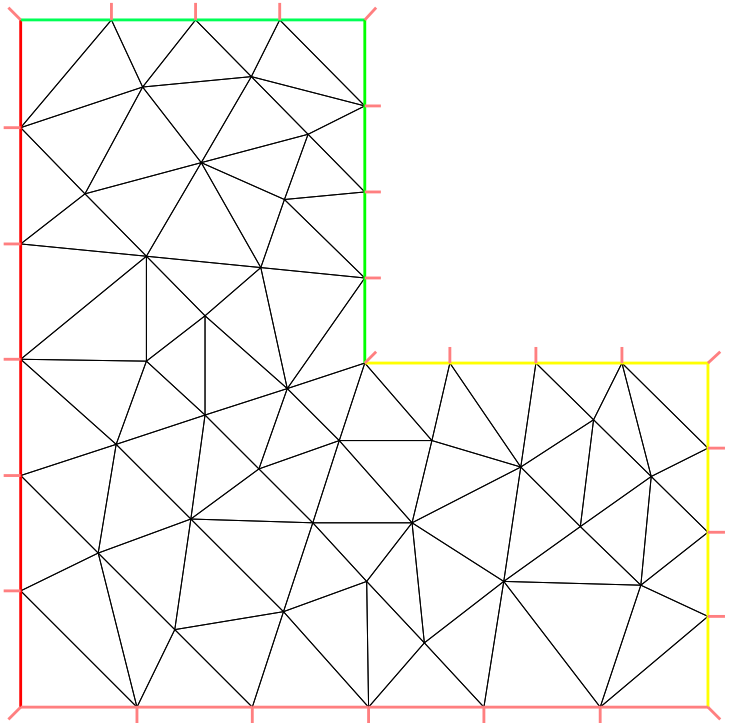
3/ Lire un maillage externe

```
mesh Th2("april-fish.msh") ;
```

avec `emc2`, `bamg`, modulef, etc...

Maillage / figure





Entrées / sorties fichiers

```
savemesh(nom_maillage,nom_fichier) ;        //    sauver le maillage (.msh)

readmesh(nom_fichier) ;        //    lire un maillage à partir d'un fichier
```

Autres fonctions sur les maillage

```
mesh mail2 = movemesh(mail1,[f1(x,y),f2(x,y)]) ; //    déformer le maillage

mesh mail2 = adaptmesh(mail1,var) ;     //    raffiner le maillage dans les
zones de forte variations
```

Lire les données d'un maillage

Th.nt (nbre de triangle), Th.nv (nbre de noeuds),
Th[i][j] (sommet j du triangle i), ...

Résoudre une EDP

Définition de l'espace d'approximation

```
fespace nom_espace(nom_maillage,type_elements_finis) ;
```

Le types d'éléments finis est un mot-clé dans la liste suivante : P0, P1, P1dc, P1b, P2, P2b, P2dc, RT, P1inc. L'espace ainsi défini est à son tour un type de données pour définir les variables de type éléments finis.

Définir le problème variationnel

```
problem pb_name(u,v,solver)=
         a(u,v) - l(v)
         + (conditions aux limites) ;
```

Pour résoudre un problème variationnel, il suffit de taper la commande :

```
pb_name ;
```

Formes bilinéaires

int1d(Th,n1,...,nk)(A*u*v) $\displaystyle\sum_{T\in\tau_h}\int_{\partial T\cap(\cup_i\Gamma_{n_i})} Auv$

int2d(Th[,k])(A*u*v) $\displaystyle\sum_{T\in\tau_h[\cap\Omega_k]}\int_{T} Auv$

intalledges(Th[,k])(A*u*v) $\displaystyle\sum_{T\in\tau_h[\cap\Omega_k]}\int_{\partial T} Auv$

Formes linéaires

int1d(Th,n1,...,nk)(A*v) $\displaystyle\sum_{T\in\tau_h}\int_{\partial T\cap(\cup_i\Gamma_{n_i})} Av$

intalledges(Th[,k])(A*v) $\displaystyle\sum_{T\in\tau_h[\cap\Omega_k]}\int_{\partial T} Av$

Visualiser les résultats

Directement avec Freefem++ Pour afficher des maillages, les courbes d'isovaleurs et les champs de vecteurs :

```
plot(var1,[var2,var3],...[liste d'options]);
wait=true/false, value=true/false, fill=num, ps="nom_fichier",...
```

Exporter vers un autre logiciel Gnuplot, Medit,...

```
{ ofstream file("exemple.bb");                    //    file for medit
file <<"2 1"<<uh[].n<<" 2"<<endl;
for (int i=0;i<uh[].n;i++){
 file << uh[][j] << endl;
}}
   //    to call gnuplot command and wait 5 second (tanks to unix command)
                                                  //    and make postscipt plot
  exec("echo 'plot \"plot.gp\" using 1:2 w l    pause 5    set term
postscript    set output \"gnuplot.eps\"    replot    quit' | gnuplot");
```

A travers des exemples

Equation de Laplace

Soit un domaine Ω avec $\partial\Omega = \Gamma_2 \cup \Gamma_e$.

Trouver u solution :

$$-\Delta u = 1 \text{ dans } \Omega, \quad u = 2 \text{ sur } \Gamma_2, \quad \frac{\partial u}{\partial \vec{n}} = 0 \text{ sur } \Gamma_e \tag{1}$$

On note $V_g = \{v \in H^1(\Omega)/v_{|\Gamma_2} = g\}$.

Formulation variationnelle : Trouver $u \in V_2(\Omega)$, t.q.

$$\int_\Omega \nabla u.\nabla v = \int_\Omega 1v + \int_{\Gamma_e} \frac{\partial u}{\partial n} v, \quad \forall v \in V_0(\Omega) \tag{2}$$

Equation de Laplace I/II

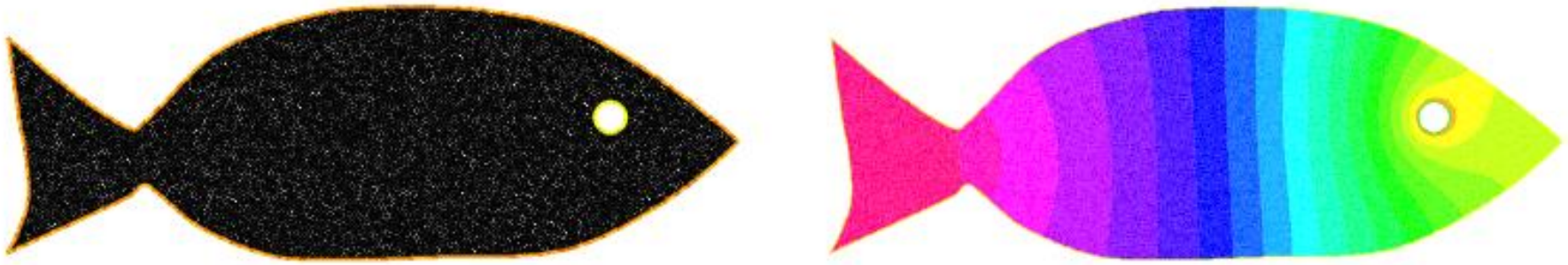
Code FreeFem++ :

```
mesh Th("april-fish.msh") ;
fespace Vh(Th,P1) ;                                  //    define the P1 EF space

 Vh u,v ;

solve laplace(u,v,solver=CG) =
    int2d(Th)(  dx(u)*dx(v)+ dy(u)*dy(v)      )
    -  int2d(Th) ( 1*v)
  + on(2,u=2) ;                                                  //    int on γ2
plot(u,fill=1,wait=1,value=0,ps="april-fish.eps") ;
```

Equation de Laplace / figure



Execute fish.edp

Equation de Laplace (formulation mixte) II/II

On cherche à résoudre $-\Delta p = f$ dans Ω et $p = g$ sur $\partial\Omega$, avec $\vec{u} = \nabla p$

Ce problème reste équivalent à :

Trouver $\vec{u}, p$ solutions dans un domaine Ω t.q. :

$$-\nabla.\vec{u} = f,\ \vec{u} - \nabla p = 0 \quad \text{dans } \Omega, \quad p = g \text{ sur } \Gamma = \partial\Omega \tag{3}$$

Formulation variationnelle Mixte :

Trouver $\vec{u} \in H_{div}(\Omega),\ p \in L^2(\Omega)$, t.q.

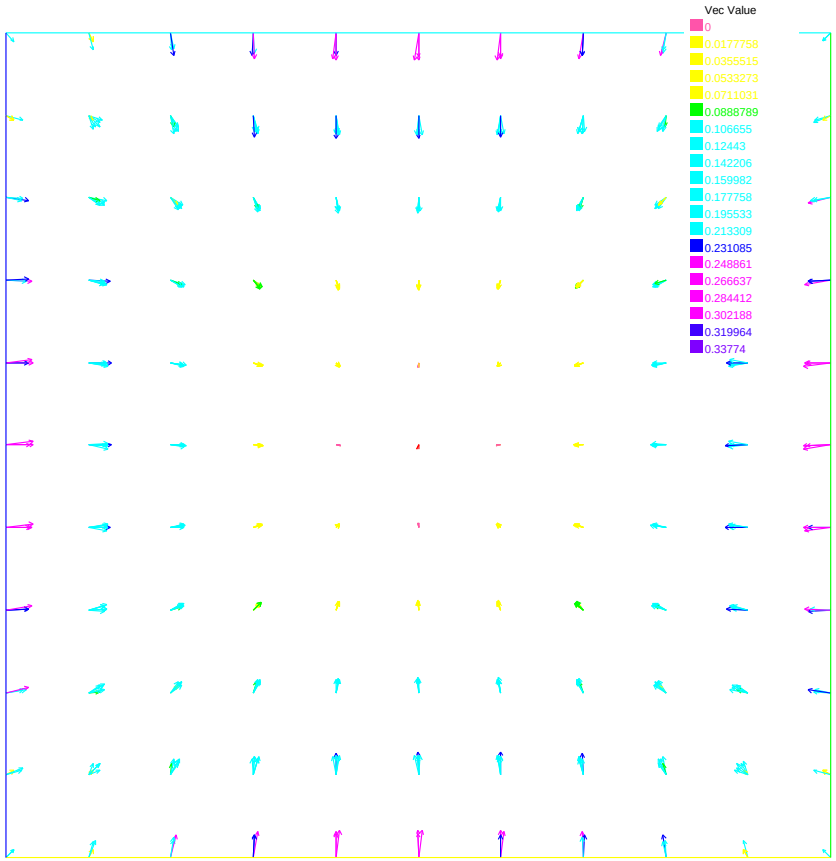
$$\int_\Omega q\nabla.\vec{u} + \int_\Omega p\nabla.\vec{v} + \vec{u}.\vec{v} = \int_\Omega -fq + \int_\Gamma g\vec{v}.\vec{n}, \quad \forall(\vec{v}, q) \in H_{div} \times L^2$$

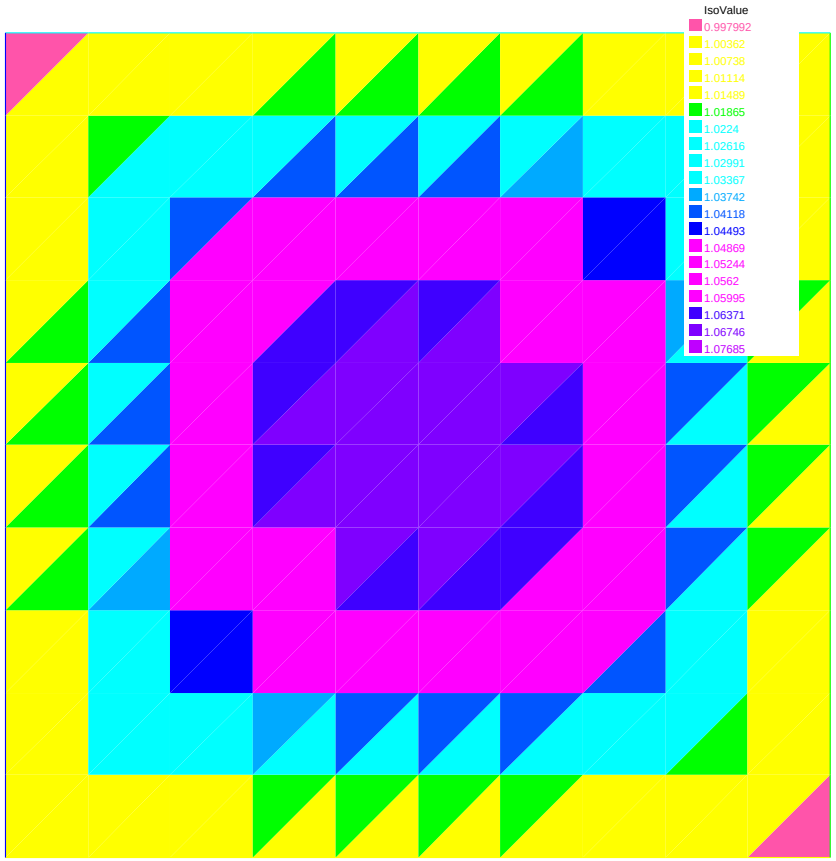
Equation de Laplace (formulation mixte) II/II

```
mesh Th=square(10,10) ;
fespace Vh(Th,RT0) ;                    fespace Ph(Th,P0) ;
Vh [u1,u2],[v1,v2] ;                      Ph p,q ;
func f=1. ;
func g=1 ;
problem laplaceMixte([u1,u2,p],[v1,v2,q],solver=LU) =                          //
   int2d(Th)(  p*q*1e-10 + u1*v1 + u2*v2
               + p*(dx(v1)+dy(v2)) + (dx(u1)+dy(u2))*q )
 - int2d(Th) ( -f*q)
 - int1d(Th)( (v1*N.x +v2*N.y)*g) ;                          //    int on gamma
 laplaceMixte ;                                  //    the problem is now solved
 plot([u1,u2],coef=0.1,wait=1,ps="lapRTuv.eps",value=true) ;
 plot(p,fill=1,wait=1,ps="laRTp.eps",value=true) ;
```

Execute LaplaceRT.edp

Equation de Laplace (formulation mixte) / figure



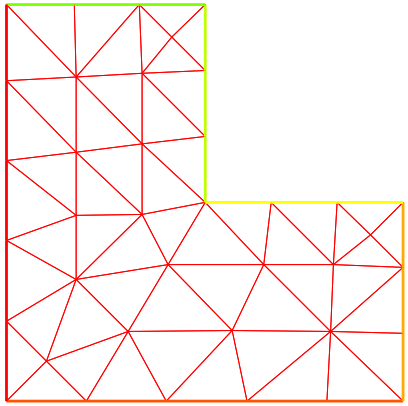


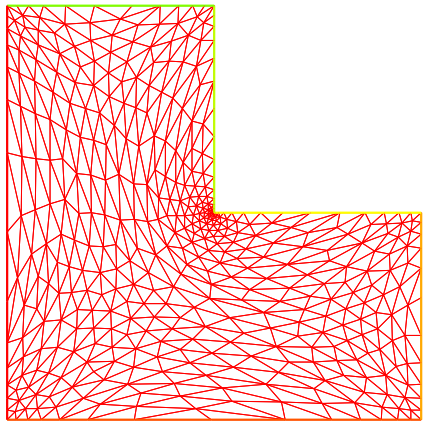
Présence d'une singularité (adaptation de maillage)

Le domaine est $\Omega =]0,1[^2 \backslash [\frac{1}{2},1]^2$ et l'EDP est

$$\text{Trouver } u \in H_0^1(\Omega) \text{ t.q.} \quad -\Delta u = 1 \text{ dans } \Omega,$$

La solution u a une singularité au voisinage des angles, et on cherche à capter numérique cette singularité.





Exemple d'un maillage adapté

programme FreeFem++ (singularités des coins)

```
border a(t=0,1.0){x=t ;   y=0 ;  label=1 ;} ;
border b(t=0,0.5){x=1 ;   y=t ;  label=2 ;} ;
border c(t=0,0.5){x=1-t ; y=0.5 ;label=3 ;} ;
border d(t=0.5,1){x=0.5 ; y=t ;  label=4 ;} ;
border e(t=0.5,1){x=1-t ; y=1 ;  label=5 ;} ;
border f(t=0.0,1){x=0 ;   y=1-t ;label=6 ;} ;


mesh Th =  buildmesh (a(6) + b(4) + c(4) +d(4) + e(4) + f(6)) ;
fespace Vh(Th,P1) ;                  Vh u,v ;          real error=0.01 ;
problem Probem1(u,v,solver=CG,eps=1.0e-6) =
     int2d(Th)(  dx(u)*dx(v) + dy(u)*dy(v))  -   int2d(Th)( v)
     + on(1,2,3,4,5,6,u=0) ;
int i ;
for (i=0 ;i< 7 ;i++)
{  Probem1 ;                                                  //    solving the pde problem
   Th=adaptmesh(Th,u,err=error) ;                          //    the adaptation with Hessian of u
   plot(Th,wait=1) ;            u=u ;      error = error/ (1000^(1./7.)) ;   } ;
```

Un problème de frontière libre

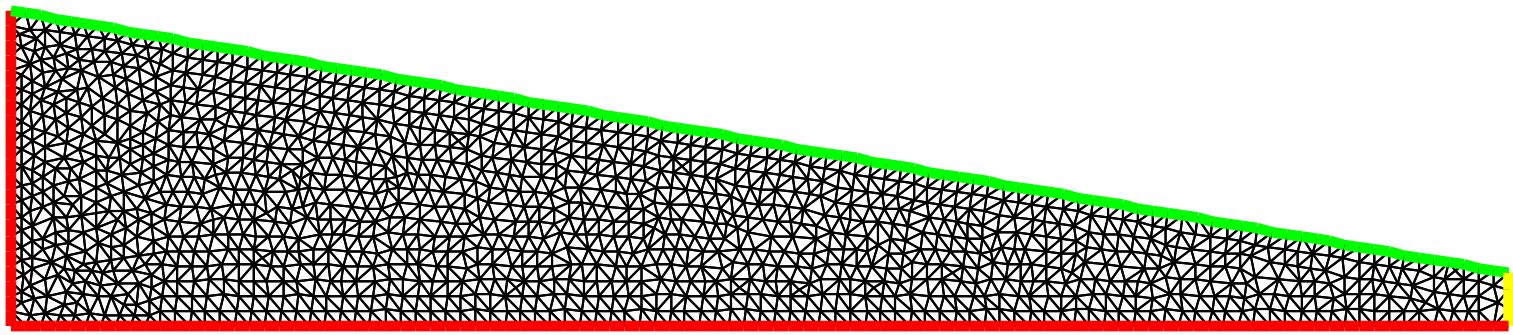
Soit un domaine Ω :

```
real L=10 ;                                                    //    Width
real h=2.1 ;                                                   //    Left height
real h1=0.35 ;                                                 //    Right height

border a(t=0,L){x=t ;y=0 ;label=1 ;} ;    //    bottom impermeable Γ_a
border b(t=0,h1){x=L ;y=t ;label=2 ;} ;    //    right, the source Γ_b
border f(t=L,0){x=t ;y=t*(h1-h)/L+h ;label=3 ;} ; //    the free surface Γ_f
border d(t=h,0){x=0 ;y=t ;label=4 ;} ;    //    Left impermeable Γ_d

int n=10 ;
mesh Th=buildmesh (a(L*n)+b(h1*n)+f(sqrt(L^2+(h-h1)^2)*n)+d(h*n)) ;
plot(Th,ps="dTh.eps") ;
```

Maillage initial



Le problème est de trouver p et Ω t.q. :

$$\begin{cases} -\Delta p = 0 & \text{dans } \Omega \\ p = y & \text{sur } \Gamma_b \\ \dfrac{\partial p}{\partial n} = 0 & \text{sur } \Gamma_d \cup \Gamma_a \\ \dfrac{\partial p}{\partial n} = \frac{q}{K} n_x & \text{sur } \Gamma_f \quad (Neumann) \\ p = y & \text{sur } \Gamma_f \quad (Dirichlet) \end{cases}$$

où, $q = 0.02$, $K = 0.5$ et la vitesse $u = -\nabla p$.

algorithme

Nous allons utiliser la méthode de point fixe : soit, $k = 0$, $\Omega^k = \Omega$.

On résout le problème : Trouver p dans $V = H^1(\Omega^k)$, t.q. $p = y$ sur $\Gamma_b^k \cup \Gamma_f^k$

$$\int_{\Omega^k} \nabla p \nabla p' = 0, \quad \forall p' \in V \text{ avec } p' = 0 \text{ sur } \Gamma_b^k \cup \Gamma_f^k$$

Avec la condition de type Newmann, on transforme le domaine par $\mathcal{F}(x, y) = [x, y - v(x)]$ où v est solution de : $v \in V$, t.q. $v = 0$ sur Γ_a^k

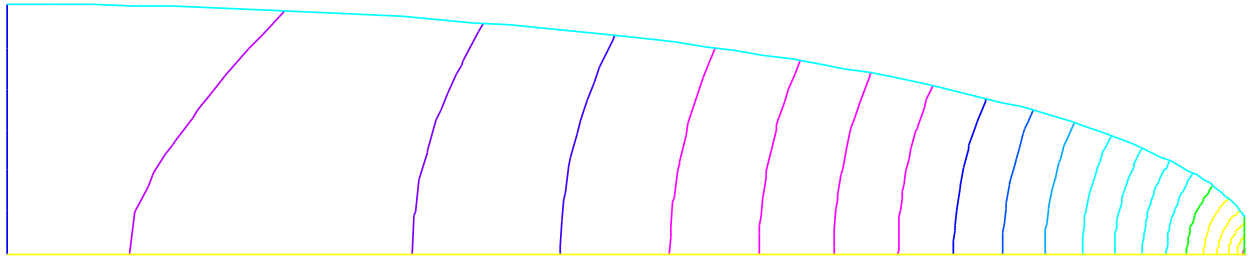
$$\int_{\Omega^k} \nabla v \nabla v' = \int_{\Gamma_f^k} (\frac{\partial p}{\partial n} - \frac{q}{K} n_x) v', \quad \forall v' \in V \text{ avec } v' = 0 \text{ sur } \Gamma_a^k$$

Le nouveau domaine est : $\Omega^{k+1} = \mathcal{F}(\Omega^k)$

```
problem Pp(p,pp,solver=CG) = int2d(Th)( dx(p)*dx(pp)+dy(p)*dy(pp))
  + on(b,f,p=y) ;
problem Pv(v,vv,solver=CG) = int2d(Th)( dx(v)*dx(vv)+dy(v)*dy(vv))
  +  on (a, v=0) + int1d(Th,f)(vv*((Q/K)*N.y- (dx(p)*N.x+dy(p)*N.y))) ;
while(errv>1e-6)
{   j++ ;  Pp ;  Pv ;    errv=int1d(Th,f)(v*v) ;
     coef = 1 ;
    //    Here french cooking if overlapping see the example
    Th=movemesh(Th,[x,y-coef*v]) ;                          //    deformation
}
```

Execute freeboundary.edp

avec du/du = residu



Un problème de décomposition de domaine

Résoudre

$$-\Delta u = f, \quad \text{in } \Omega = \Omega_1 \cup \Omega_2 \quad u|_\Gamma = 0$$

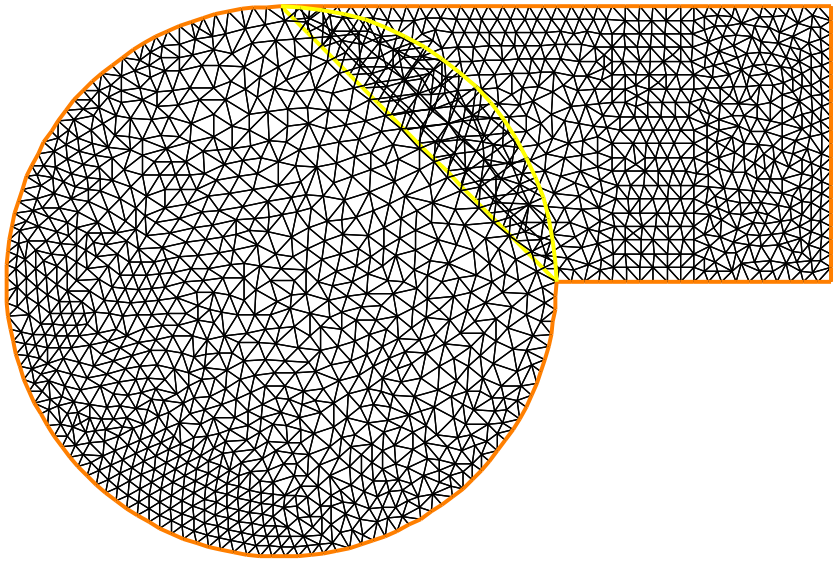
par l'algorithme de Schwarz :

$$-\Delta u_1^{m+1} = f \text{ in } \Omega_1 \quad u_1^{m+1}|_{\Gamma_1} = u_2^m$$

$$-\Delta u_2^{m+1} = f \text{ in } \Omega_2 \quad u_2^{m+1}|_{\Gamma_2} = u_1^m$$

où Γ_i est la frontière de Ω_i avec les conditions $\Omega_1 \cap \Omega_2 \neq \emptyset$ et que $u_i = 0$ pour l'itération 1.

Ici, Ω_1 est un quadrangle et Ω_2 est un disque.



Schwarz-overlap.edp / Maillage

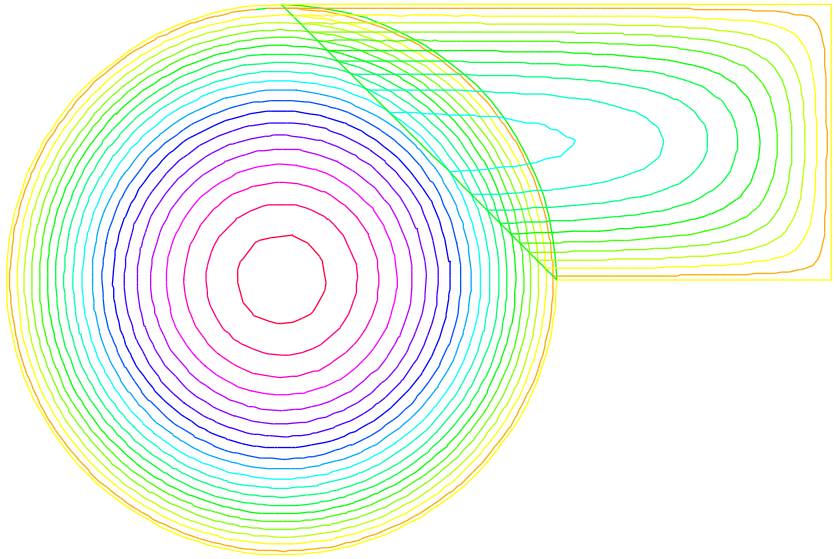
```
int inside = 2;                                          //    inside boundary
int outside = 1;                                         //    outside boundary
border a(t=1,2){x=t;y=0;label=outside;};
border b(t=0,1){x=2;y=t;label=outside;};
border c(t=2,0){x=t;y=1;label=outside;};
border d(t=1,0){x = 1-t; y = t;label=inside;};
border e(t=0, pi/2){ x= cos(t); y = sin(t);label=inside;};
border e1(t=pi/2, 2*pi){ x= cos(t); y = sin(t);label=outside;};
int n=4;
mesh th = buildmesh( a(5*n) + b(5*n) + c(10*n) + d(5*n));
mesh TH = buildmesh( e(5*n) + e1(25*n) );
plot(th,TH,wait=1);                                      //    to see the 2 meshes
```

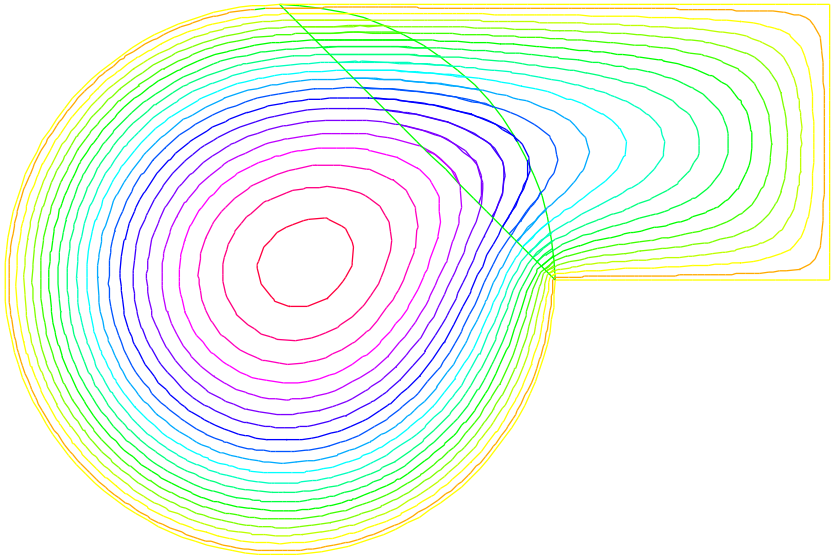
Schwarz-overlap.edp

```
fespace vh(th,P1) ;
fespace VH(TH,P1) ;
vh u=0,v ; VH U,V ;
int i=0 ;

problem PB(U,V,init=i,solver=Cholesky) =
    int2d(TH)( dx(U)*dx(V)+dy(U)*dy(V) )
  + int2d(TH)( -V) + on(inside,U = u)  + on(outside,U= 0 ) ;
problem pb(u,v,init=i,solver=Cholesky) =
    int2d(th)( dx(u)*dx(v)+dy(u)*dy(v) )
  + int2d(th)( -v) + on(inside ,u = U) + on(outside,u = 0 ) ;
for ( i=0 ;i< 10 ; i++)
{
   PB ;   pb ;      plot(U,u,wait=true) ;
} ;
```

Schwarz-overlap.edp





Isovaleurs de la solution à l'itération 0 et à l'iteration 9

Equations de Stokes

Le problème de Stokes consiste à trouver une vitesse $\boldsymbol{u} = (u_1, .., u_d)$ et une préssion p dans un domaine Ω de $\mathbb{R}^d$, vérifiant :

$$\begin{aligned} -\Delta \boldsymbol{u} + \nabla p &= 0 \quad \text{dans} \quad \Omega \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{dans} \quad \Omega \\ \boldsymbol{u} &= \boldsymbol{u}_\Gamma \quad \text{sur} \quad \Gamma \end{aligned}$$

où, $\boldsymbol{u}_\Gamma$ est la vitesse sur le bord Γ.

Une formulation variationnelle de ce problème est : Trouver $\boldsymbol{u} \in H^1(\Omega)^d$ avec $\boldsymbol{u}_{|\Gamma} = u_\Gamma$, et $p \in L^2(\Omega)/\mathbb{R}$ t.q.

$$\forall \boldsymbol{v} \in H_0^1(\Omega)^d, \ \forall q \in L^2(\Omega)/\mathbb{R}, \qquad \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - p \nabla . v - q \nabla . u = 0$$

Numériquement, nous cherchons $(u, p) \in H_0^1(\Omega)^d \times L^2(\Omega)$ t.q.
(avec $\varepsilon = 10^{-10}$)

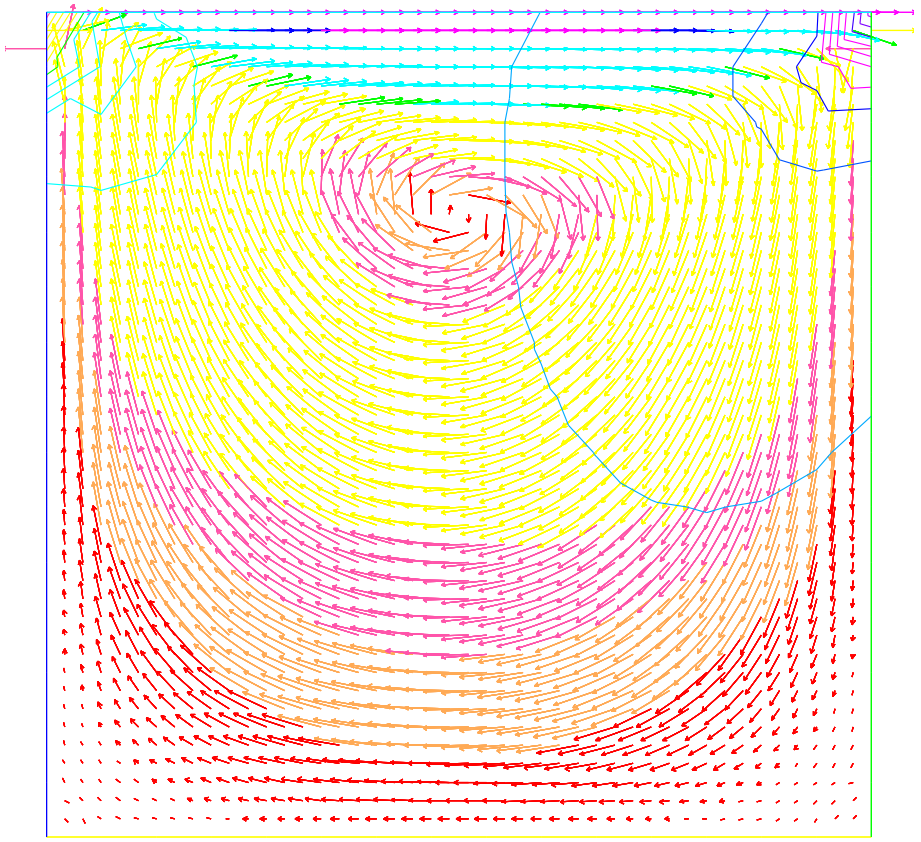
$$\forall \boldsymbol{v} \in H_0^1(\Omega)^d, \ \forall q \in L^2(\Omega), \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - p \nabla . v - q \nabla . u + \varepsilon p q = 0$$

Stokes en FreeFem++

```
mesh Th=square(15,15) ;

fespace Uh(Th,P2) ; Uh u1,u2,v1,v2 ;
fespace Ph(Th,P1) ; Ph p,q ;

solve stokes([u1,u2,p],[v1,v2,q]) = 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)+dy(v2))
                                              - q*(dx(u1)+dy(u2)))
                                          + on(1,2,4,u1=0,u2=0)
                                          + on(3,u1=1,u2=0) ;
plot([u1,u2],p,ps="stokes_exple.eps") ;
```



Merci pour votre attention ?