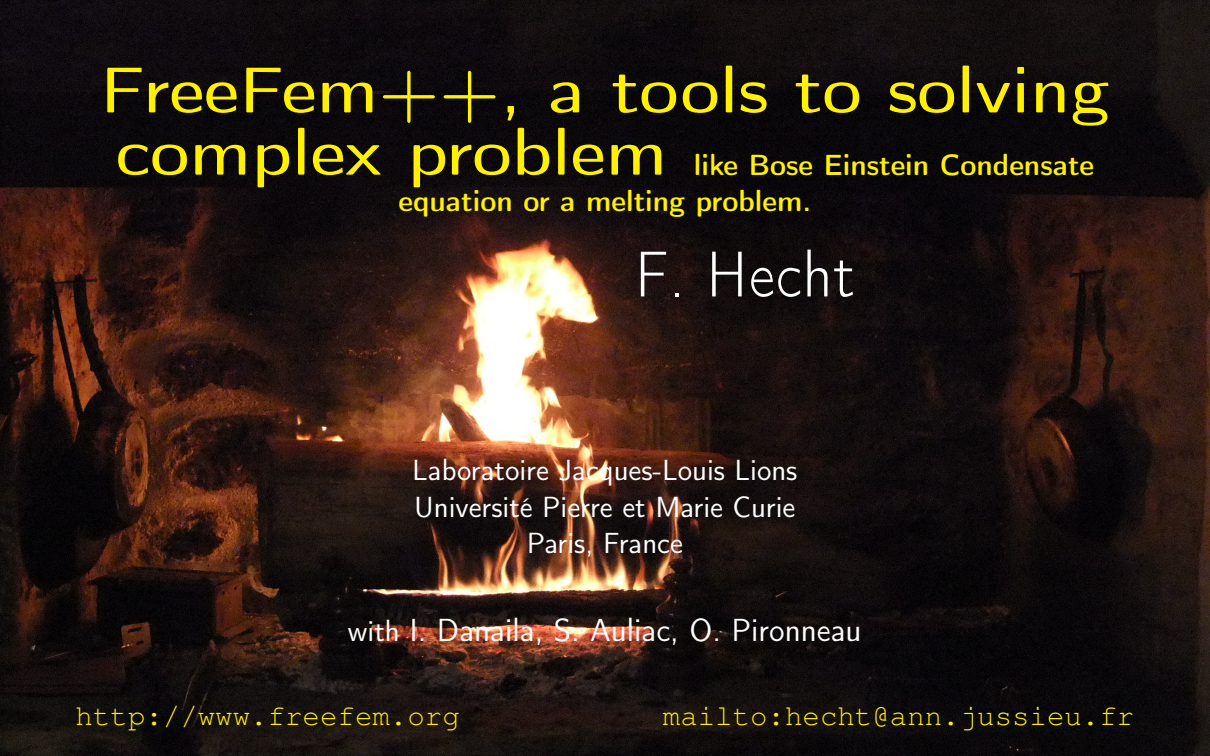
FreeFem++, a tools to solving complex problem like Bose Einstein Condensate equation or a melting problem.



F. Hecht

Laboratoire Jacques-Louis Lions
Université Pierre et Marie Curie
Paris, France

with I. Danaila, S. Auliac, O. Pironneau

http://www.freefem.org mailto:hecht@ann.jussieu.fr

Outline

News : FreeFem++ pass 10^9 unknowns

The message from Pierre Jolivet (june 2012)

```
Machine: Curie Thin Node@CEA
Financement: PRACE project HPC-PDE (number 2011050840)
Nombre de coeurs: 6144     Mémoire par coeurs: 4 Go
Eléments finis: P3  Dimension: 2D
Précision: 1e-08
Nombres d'inconnues: 1 224 387 085
Méthode de résolution: GMRES préconditionné par une méthode de
décomposition de domaine à deux niveaux
Nombre d'itérations: 18
Temps de résolution: 2 minutes
Type de problème: équation de diffusion avec coefficients très
hétérogènes (5 ordres de grandeur de variation)
```

The FreeFem++ days, the December 6th and 7th 2012, UPMC, Jussieu, Paris, France

Introduction

`FreeFem++` is a software to solve numerically partial differential equations (PDE) in $\mathbb{R}^2$) and in $\mathbb{R}^3$) with finite elements methods. We used a user language to set and control the problem. The `FreeFem++` language allows for a quick specification of linear PDE's, with the variational formulation of a linear steady state problem and the user can write they own script to solve no linear problem and time depend problem. You can solve coupled problem or problem with moving domain or eigenvalue problem, do mesh adaptation , compute error indicator, etc ...

By the way, FreeFem++ is build to play with abstract linear, bilinear form on Finite Element Space and interpolation operator.

FreeFem++ is a freeware and this run on Mac, Unix and Window architecture, in parallel with MPI.

History

1987 MacFem/PCFem les ancêtres (O. Pironneau en Pascal) payant.

1992 FreeFem réécriture de C++ (P1,P0 un maillage) O. Pironneau, D. Bernardi, F. Hecht , C. Prudhomme (adaptation Maillage, bamg).

1996 FreeFem+ réécriture de C++ (P1,P0 plusieurs maillages) O. Pironneau, D. Bernardi, F. Hecht (algèbre de fonction).

1998 FreeFem++ réécriture avec autre noyau élément fini, et un autre langage utilisateur ; F. Hecht, O. Pironneau, K.Ohtsuka.

1999 FreeFem 3d (S. Del Pino) , Une première version de freefem en 3d avec des méthodes de domaine fictif.

2008 FreeFem++ v3 réécriture du noyau élément fini pour prendre en compte les cas multidimensionnels : 1d,2d,3d...

For who, for what !

For what

1. R&D
2. Academic Research ,
3. Teaching of FEM, PDE, Weak form and variational form
4. Algorithmes prototyping
5. Numerical experimentation
6. Scientific computing and Parallel computing

For who : the researcher, engineer, professor, student...

The mailing list mailto:Freefempp@ljll.math.upmc.fr with 377 members with a flux of 5-20 messages per day.

More than 1000 true Users (more than 200 download / month)

The main characteristics I/II (2D)(3D)

- Wide range of finite elements : continuous P1,P2 elements, discontinuous P0, P1, RT0,RT1,BDM1, elements ,Edge element, vectorial element, mini-element, ...
- Automatic interpolation of data from a mesh to an other one (with matrix construction if need), so a finite element function is view as a function of (x, y, z) or as an array.
- Definition of the problem (complex or real value) with the variational form with access to the vectors and the matrix.
- Discontinuous Galerkin formulation (only in 2d to day).
- LU, Cholesky, Crout, CG, GMRES, UMFPack, SuperLU, MUMPS, HIPS , SUPERLU_DIST, PASTIX. ... sparse linear solver ; eigenvalue and eigenvector computation with ARPACK.
- Online graphics with OpenGL/GLUT/VTK, C++ like syntax.
- An integrated development environment FreeFem++-cs

The main characteristics II/II (2D)(3D)

- Analytic description of boundaries, with specification by the user of the intersection of boundaries in 2d.
- Automatic mesh generator, based on the Delaunay-Voronoï algorithm. (2d,3d (tetgen))
- load and save Mesh, solution
- Mesh adaptation based on metric, possibly anisotropic (only in 2d), with optional automatic computation of the metric from the Hessian of a solution. (2d,3d).
- Link with other soft : parview, gmsh , vtk, medit, gnuplot
- Dynamic linking to add plugin.
- Full MPI interface
- Nonlinear Optimisation tools : CG, Ipopt, NLOpt, stochastic
- Wide range of examples : Navier-Stokes 3d, elasticity 3d, Shell, fluid structure, eigenvalue problem, Schwarz' domain decomposition algorithm, residual error indicator ...

Laplace equation, weak form

Let a domain Ω with a partition of $\partial\Omega$ in Γ_2, Γ_e.
Find u a solution in such that :

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

Denote $V_g = \{v \in H^1(\Omega)/v_{|\Gamma_2} = g\}$.
The Basic variationnal formulation with is : find $u \in V_2(\Omega)$, such that

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

The finite element method is just : replace V_g with a finite element space, and the `FreeFem++` code :

Poisson equation in a fish with FreeFem++

The finite element method is just : replace V_g with a finite element space, and the FreeFem++ code :

```
mesh3 Th("fish-3d.msh");                          //    read a mesh 3d
fespace Vh(Th,P1);                          //    define the P1 EF space

 Vh u,v;                    //   set test and unknown function in Vh.
macro Grad(u) [dx(u),dy(u),dz(u)]                  //   EOM Grad def
solve laplace(u,v,solver=CG) =
   int3d(Th)(  Grad(u)'*Grad(v)    )
   - int3d(Th) ( 1*v)
   + on(2,u=2);                                    //    int on γ2
plot(u,fill=1,wait=1,value=0,wait=1);
```

Run:fish.edp Run:fish3d.edp

Mesh adaptation : Metric / unit Mesh

In Euclidean geometry the length $|\gamma|$ of a curve γ of $\mathbb{R}^d$ parametrized by $\gamma(t)_{t=0..1}$ is

$$|\gamma| = \int_0^1 \sqrt{(\gamma'(t), \gamma'(t))_{\mathbb{R}^d}}\, dt$$

We introduce the metric as a field $x \mapsto \mathcal{M}_x$ of $d \times d$ symmetric definite > 0 matrices, and the length ℓ of Γ w.r.t $\mathcal{M}$ is :

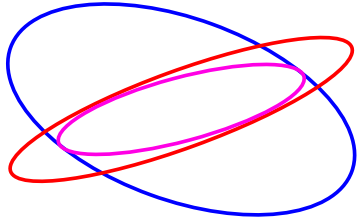
$$\ell = \int_0^1 \sqrt{(\gamma'(t), \mathcal{M}_{\gamma(t)}\gamma'(t))_{\mathbb{R}^d}}\, dt$$

The key-idea is to construct a mesh where the lengths of the edges are close to 1 accordingly to $\mathcal{M}$.

Mesh adaptation : Metrix intersection

The unit ball $\mathcal{BM}$ in a metric $\mathcal{M}$ plot the maximum mesh size on all the direction, is a ellipse.
If you we have two unknowns u and v, we just compute the metric $\mathcal{M}_u$ and $\mathcal{M}_v$, find a metric $\mathcal{M}_{uv}$ call intersection with the biggest ellipse such that :

$$\mathcal{B}(\mathcal{M}_v) \subset \mathcal{B}(\mathcal{M}_u) \cap \mathcal{B}(\mathcal{M}_v)$$



Build of the metric form the solution u

Optimal metric norm for interpolation error (function `adaptmesh` in freefem++) for P_1 continuous Lagrange finite element

- L^∞ : $\mathcal{M} = \frac{1}{\varepsilon}|\nabla\nabla u| = \frac{1}{\varepsilon}|\mathcal{H}|$ where $\mathcal{H} = \nabla\nabla u$
- L^p : $\mathcal{M} = \frac{1}{\varepsilon}|det(\mathcal{H})|^{\frac{1}{2p+2}}\,|\mathcal{H}|$ (result of F. Alauzet, A. Dervieux)

In Norm $W^{1,p}$, the optimal metric $\mathcal{M}_\ell$ for the P_ℓ Lagrange finite element, Optimal is given by (with only acute triangle) (thank J-M. Mirebeau)

$$\mathcal{M}_{\ell,p} = \frac{1}{\varepsilon}(det\mathcal{M}_\ell)^{\frac{1}{\ell p+2}}\,\mathcal{M}_\ell$$

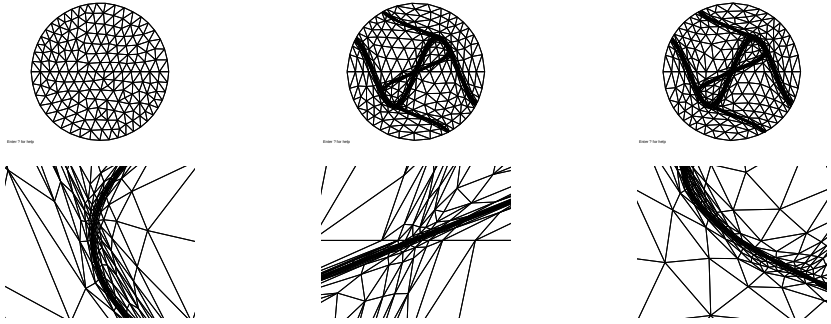
and (see `MetricPk` plugin and function)

- for P_1 : $\mathcal{M}_1 = \mathcal{H}^2$ (sub optimal with acute triangle take $\mathcal{H}$)
- for P_2 : $\mathcal{M}_2 = 3\sqrt{\begin{pmatrix} a & b \\ b & c \end{pmatrix}^2 + \begin{pmatrix} b & c \\ c & a \end{pmatrix}^2}$ with $D^{(3)}u(x,y) = (ax^3 + 3bx^2y + 3cxy^2 + dy^3)/3!$,

Mesh adaptation : Example of mesh

$$u = (10x^3 + y^3) + atan2(0.001, (sin(5y) - 2x))$$

$$v = (10y^3 + x^3) + atan2(0.01, (sin(5x) - 2y)).$$



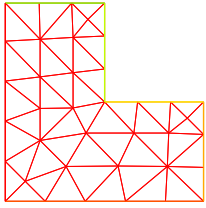
A corner singularity

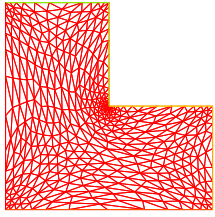
adaptation with metric

The domain is an L-shaped polygon $\Omega =]0,1[^2 \setminus [\frac{1}{2},1]^2$ and the PDE is

$$\text{Find } u \in H_0^1(\Omega) \text{ such that } \quad -\Delta u = 1 \text{ in } \Omega,$$

The solution has a singularity at the reentrant angle and we wish to capture it numerically.





example of Mesh adaptation

FreeFem++ corner singularity program

```
int[int] lab=[1,1,1,1];
mesh Th = square(6,6,label=lab);
Th=trunc(Th,x<0.5 | y<0.5, label=1);

fespace Vh(Th,P1);          Vh u,v;          real error=0.1;
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,u=0);
for (int i=0;i< 7;i++)
{  Probem1;                                    //    solving the pde
   Th=adaptmesh(Th,u,err=error,nbvx=100000);
                              //    the adaptation with Hessian of u
   plot(Th,u,wait=1,fill=1);            u=u;
   error = error/ (1000.^(1./7.));   };
```

Run:CornerLap.edp

Poisson equation with 3d mesh adaptation

```
load "msh3" load "tetgen" load "mshmet" load "medit"
int nn  = 6;
int[int] l1111=[1,1,1,1],l01=[0,1],l11=[1,1];                          //   label numbering
mesh3 Th3=buildlayers(square(nn,nn,region=0,label=l1111),
      nn,  zbound=[0,1], labelmid=l11,labelup = l01,labeldown = l01);
Th3=trunc(Th3,(x<0.5)|(y < 0.5)|(z < 0.5) ,label=1);                    //   remove ]0.5,1[^3

fespace Vh(Th3,P1);       Vh u,v;                          //   FE. space definition
macro Grad(u) [dx(u),dy(u),dz(u)]                                      //    EOM
problem Poisson(u,v,solver=CG) =
  int3d(Th3)( Grad(u)'*Grad(v) )  -int3d(Th3)( 1*v ) + on(1,u=0);

real errm=1e-2;                                                //    level of error
for(int ii=0; ii<5; ii++)
{
  Poisson;         Vh h;
  h[]=mshmet(Th3,u,normalization=1,aniso=0,nbregul=1,hmin=1e-3,
             hmax=0.3,err=errm);
  errm*= 0.8;                                  //    change the level of error
  Th3=tetgreconstruction(Th3,switch="raAQ"
        ,sizeofvolume=h*h*h/6.);
  medit("U-adap-iso-"+ii,Th3,u,wait=1);
}
```

Run:Laplace-Adapt-3d.edp

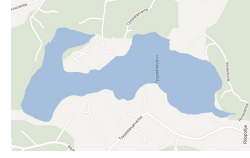
Build mesh from image 1/2

```
load "ppm2rnm" load "isoline"
real[int,int] Curves(3,1);
int[int] be(1);  int nc;                              //    nb of curve
{
  real[int,int] ff1("Lac-tyyppalanjarvi.pgm");
  int nx = ff1.n, ny=ff1.m;                    //    grey value [0,1]
  mesh Th=square(nx-1,ny-1,
                 [(nx-1)*(x),(ny-1)*(1-y)]);
  fespace Vh(Th,P1); Vh f1; f1[]=ff1;
  nc=isoline(Th,f1,iso=0.75,close=0,
             Curves,beginend=be,
             smoothing=.1,ratio=0.5);
}
```

Build mesh from image 2/2

```
border Curve0(t=0,1)                    //    the extern boundary
{ int c =0;                                     //    component 0
  int i0 = be[2*c], i1 = be[2*c+1]-1;
  P=Curve(xy,i0,i1,t);                          //    Curve 0
  label=1;
}
...

plot(Curve1(100));                              //    show curve.
mesh Th= buildmesh(Curve1(-100));                       //
plot(Th,wait=1);                                        //
```



Run:lac.edp (a local lac Jyväskylä)

Ipopt

The IPOPT optimizer in a FreeFem++ script is done with the `IPOPT` function included in the `ff-Ipopt` dynamic library. IPOPT is designed to solve constrained minimization problem in the form :

$$\begin{array}{ll} \text{find} & x_0 = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} f(x) \\ \text{s.t.} & \left\{ \begin{array}{ll} \forall i \leq n,\ x_i^{\mathrm{lb}} \leq x_i \leq x_i^{\mathrm{ub}} & \text{(simple bounds)} \\ \forall i \leq m,\ c_i^{\mathrm{lb}} \leq c_i(x) \leq c_i^{\mathrm{ub}} & \text{(constraints functions)} \end{array} \right. \end{array}$$

Where ub and lb stand for "upper bound" and "lower bound". If for some $i, 1 \leq i \leq m$ we have $c_i^{\mathrm{lb}} = c_i^{\mathrm{ub}}$, it means that c_i is an equality constraint, and an inequality one if $c_i^{\mathrm{lb}} < c_i^{\mathrm{ub}}$.

Ipopt interface

```
load  "ff-Ipopt"
varf vP([u1,u2],[v1,v2])
      =   int2d(Th)(Grad(u1)'*Grad(v1)+ Grad(u2)'*Grad(v2))
        - int2d(Th)(f1*v1+f2*v2);

matrix A = vP(Vh,Vh);                                   //   Fitness function matrix...
real[int] b = vP(0,Vh);                                        //   and linear form
int[int] II1=[0],II2=[1];                                    //   Constraints matrix
matrix C1 =  interpolate (Wh,Vh, U2Vc=II1);
matrix C2 =  interpolate (Wh,Vh, U2Vc=II2);
matrix CC = -1*C1 + C2;                                                  //    u2 - u1 >0
Wh cl=0;                              //   constraints lower bounds (no upper bounds)
varf vGamma([u1,u2],[v1,v2]) = on(1,2,3,4,u1=1,u2=1);
real[int] onGamma=vGamma(0,Vh);
Vh [ub1,ub2]=[g1,g2];
Vh [lb1,lb2]=[g1,g2];
ub1[] = onGamma ? ub1[] : 1e19 ;                                 //   Unbounded in interior
lb1[] = onGamma ? lb1[] : -1e19 ;
Vh [uzi,uzi2]=[uz,uz2],[lzi,lzi2]=[lz,lz2],[ui1,ui2]=[u1,u2];;
Wh lmi=lm;
IPOPT([b,A],CC,ui1[],lb=lb1[],clb=cl[],ub=ub1[],warmstart=iter>1,uz=uzi[],lz=lzi[],lm=lmi
```

Run:IpoptLap.edp Run:IpoptVI2.edp Run:IpoptMinSurfVol.edp

Bose Einstein Condensate

Just a direct use of `Ipopt` interface (2 day of works)
The problem is find a complex field u on domain $\mathcal{D}$ such that :

$$u = \underset{||u||=1}{\operatorname{argmin}} \int_{\mathcal{D}} \frac{1}{2}|\nabla u|^2 + V_{trap}|u|^2 + \frac{g}{2}|u|^4 - \Omega i\overline{u}\left(\begin{pmatrix} -y \\ x \end{pmatrix}.\nabla\right)u$$

to code that in FreeFem++
use

- Ipopt interface (`https://projects.coin-or.org/Ipopt`)
- Adaptation de maillage

Run:BEC.edp

Phase change with Natural Convection The starting point of the

problem is Brainstorming session (part I) of the third FreeFem++ days in december 2011, this is almost the Orange Problem is describe in web page `http://www.ljll.math.upmc.fr/~hecht/ftp/ff++days/2011/Orange-problem.pdf`.
The coupling of natural convection modeled by the Boussinesq approximation and liquid to solid phase change in $\Omega =]0,1[^2$, No slip condition for the fluid are applied at the boundary and adiabatic condition on upper and lower boundary and given temperature θ_r (resp θ_l) at the right and left boundaries.
The model is : find the field : the velocity $\boldsymbol{u} = (u_1, u_2)$, the pressure p and temperature θ :

$$\left\{\begin{array}{rll} \boldsymbol{u} & \text{given} & \text{in } \Omega_s \\ \partial_t \boldsymbol{u} + (\boldsymbol{u}\nabla)\boldsymbol{u} + \nabla.\mu\nabla\boldsymbol{u} + \nabla p & = -c_T \boldsymbol{e}_2 & \text{in } \Omega_f \\ \nabla.\boldsymbol{u} & = 0 & \text{in } \Omega_f \\ \partial_t\theta + (\boldsymbol{u}\nabla)\theta + \nabla.k_T\nabla\theta & = \partial_t S(T) & \text{in } \Omega \end{array}\right. \tag{3}$$

Where Ω_f is the fluid domain and the solid domain is $\Omega_s = \Omega \setminus \Omega_f$.

Phase change with Natural Convection

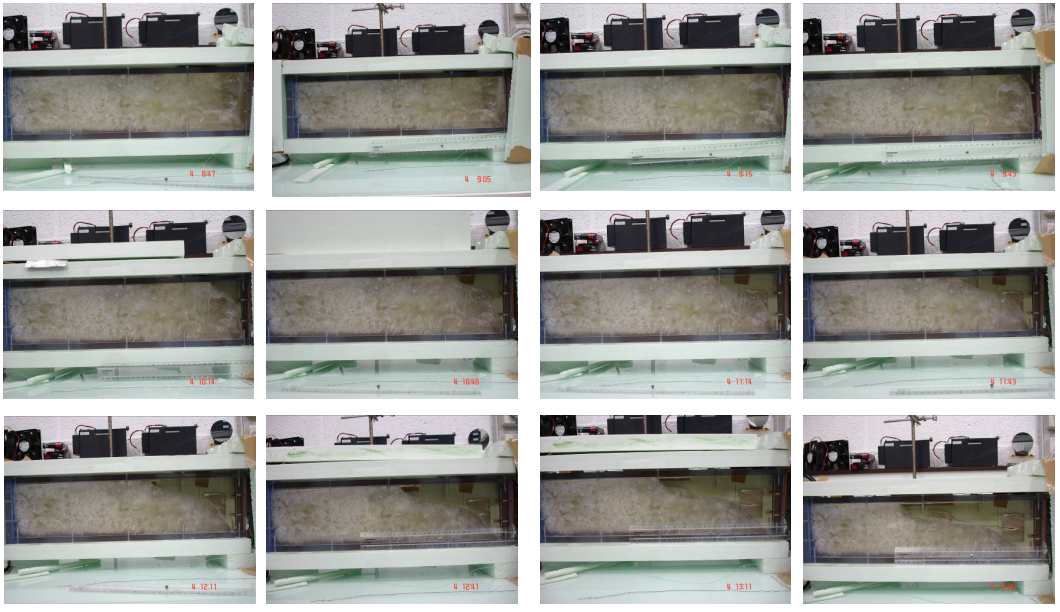
The enthalpy of the change of phase is given by the function S ; μ is the relative viscosity, k_T the thermal diffusivity.
In $\Omega_f = \{x \in \Omega; \theta > \theta_f\}$, with θ_m the melting temperature the solid has melt.
We modeled, the solid phase as a fluid with huge viscosity, so :

$$\mu = \begin{cases} \theta < \theta_f & \sim & 10^6 \\ \theta \geq \theta_m & \sim & \frac{1}{\mathrm{Re}} \end{cases},$$

The Stefan enthalpy S_c with defined by $S_c(\theta) = H(\theta)/S_{th}$ where S_{the} is the stefan number, and H is the Heaviside function with use the following smooth the enthalpy :

$$S(\theta) = \frac{\tanh(50(\theta - \theta_m)))}{2S_{te}}.$$

The true device



the Algorithm

We apply a fixed point algorithm for the phase change part (the domain Ω_f is fixed at each iteration) and a full no-linear Euler implicit scheme with a fixed domain for the rest. We use a Newton method to solve the non-linearity.

- if we don't make mesh adaptation, the Newton method do not converge
- if we use explicit method diverge too,
- if we implicit the dependance in Ω_s the method also diverge.

This is a really difficult problem.

the Algorithm, implementation

The finite element space to approximate $u1, u2, p, \theta$ is defined by

```
fespace Wh(Th,[P2,P2,P1,P1]);
```

We do mesh adaptation a each time step, with the following code :

```
Ph ph = S(T), pph=S(Tp);
Th= adaptmesh(Th,T,Tp,ph,pph,[u1,u2],err=errh,
              hmax=hmax,hmin=hmax/100,ratio = 1.2);
```

This mean, we adapt with all variable plus the 2 melting phase a time $n+1$ and n and we smooth the metric with a ratio of 1.2 to account for the movement of the melting front.

The Newton loop

the fixed point are implemented as follows

```
real err=1e100,errp ;
 for(int kk=0;kk<2;++kk) // 2 step of fixe point on Ω_s
 { nu = nuT; // recompute the viscosity in Ω_s,Ω_f
   for(int niter=0;niter<20; ++ niter) // newton loop
    { BoussinesqNL;
      err = u1w[].linfty;
      cout << niter << " err NL " << err <<endl;
      u1[] -= u1w[];
      if(err < tolNewton) break;  }// convergence ..
  }
```

The linearized problem

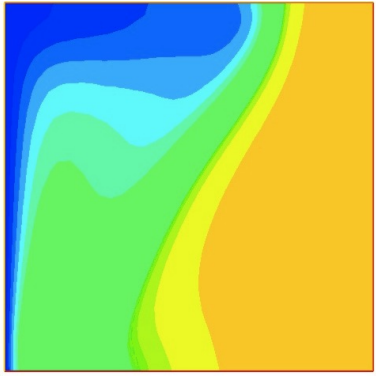
```
problem BoussinesqNL([u1w,u2w,pw,Tw],[v1,v2,q,TT])
= int2d(Th) (
      [u1w,u2w,Tw]'*[v1,v2,TT]*cdt
      + UgradV(u1,u2,u1w,u2w,Tw)' * [v1,v2,TT]
      + UgradV(u1w,u2w,u1,u2,T)' * [v1,v2,TT]
      + ( Grad(u1w,u2w)'*Grad(v1,v2)) * nu
      + ( Grad(u1,u2)'*Grad(v1,v2)) * dnu* Tw
      + cmT*Tw*v2  + grad(Tw)'*grad(TT)*kT
      - div(u1w,u2w)*q -div(v1,v2)*pw - eps*pw*q
      + dS(T)*Tw*TT*cdt )
   - int2d(Th)(
      [u1,u2,T]'*[v1,v2,TT]*cdt
      + UgradV(u1,u2,u1,u2,T)' * [v1,v2,TT]
      + ( Grad(u1,u2)'*Grad(v1,v2)) * nu
      + cmT*T*v2 - eps*p*q  +  grad(T)'*grad(TT)*kT
      - div(u1,u2)*q -div(v1,v2)*p
      +  S(T)*TT*cdt - [u1p,u2p,Tp]'*[v1,v2,TT]*cdt
      -  S(Tp)*cdt*TT)
 + on(1,2,3,4, u1w=0,u2w=0)+on(2,Tw=0)+on(4,Tw=0) ;
```

The parameters of the computation

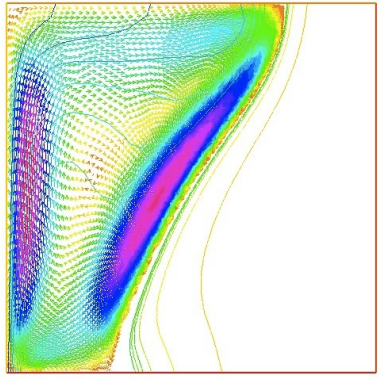
take case 2 from
Shimin Wang, Amir Faghri, and Theodore L. Bergman. A comprehensive numerical model for melting with natural convection. *International Journal of Heat and Mass Transfer*, January 2010.

$\theta_m = 0$, $\mathrm{Re} = 1$, $S_{te} = 0.045$, $P_r = 56.2$, $R_a = 3.27\ 10^5$, $\theta_l = 1, \theta_r = -0.1$ so in this case $\mathrm{cmT} = c_T = -R_a/P_r$, $\mathrm{kT} = k_T = 1/P_r$, $\mathrm{eps} = 10^{-6}$, time step $\delta t = 10^{-1}$, $\mathrm{cdt} = 1/\delta t$, at time $t = 80$ and we get a good agreement with the article.

Phase change with Natural Convection



Temperature Field at time=80s



Velocity field

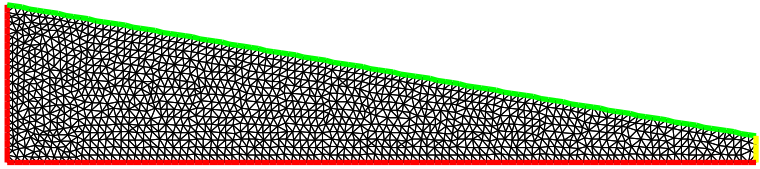
So now, a real problem, get the physical parameter of the real experiment.
Run:Orange-Newton.edp

A Free Boundary problem , (phreatic water)

Let a trapezoidal domain Ω defined in FreeFem++ :

```
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;}; //    impermeable Γ_a
border b(t=0,h1){x=L;y=t;label=2;};     //    the source Γ_b
border f(t=L,0){x=t;y=t*(h1-h)/L+h;label=3;}; //    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");
```

The initial mesh



The problem is : find p and Ω such that :

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

where the input water flux is $q = 0.02$, and $K = 0.5$. The velocity u of the water is given by $u = -\nabla p$.

algorithm

We use the following fix point method : (*with bad main B.C. Execute freeboundaryPB.edp*) let be, $k = 0$, $\Omega^k = \Omega$. First step, we forgot the Neumann BC and we solve the problem : Find p in $V = H^1(\Omega^k)$, such $p = y$ on Γ_b^k et on Γ_f^k

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

With the residual of the Neumann boundary condition we build a domain transformation $\mathcal{F}(x, y) = [x, y - v(x)]$ where v is solution of : $v \in V$, such than $v = 0$ on Γ_a^k (bottom)

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

remark : we can use the previous equation to evaluate

$$\int_{\Gamma^k} \frac{\partial p}{\partial n} v' = - \int_{\Omega^k} \nabla p \nabla v'$$

Implementation

The new domain is : $\Omega^{k+1} = \mathcal{F}(\Omega^k)$ Warning if is the movement is too large we can have triangle overlapping.

```
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

Linear Lame equation, weak form

Let a domain $\Omega \subset \mathbb{R}^d$ with a partition of $\partial\Omega$ in Γ_d, Γ_n.
Find the displacement $\boldsymbol{u}$ field such that :

$$-\nabla.\sigma(\boldsymbol{u}) = \boldsymbol{f} \text{ in } \Omega, \quad \mathbf{u = 0} \text{ on } \mathbf{\Gamma_d}, \quad \sigma(\mathbf{u}).\mathbf{n = 0} \text{ on } \mathbf{\Gamma_n} \tag{4}$$

Where $\varepsilon(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + {}^t\nabla \boldsymbol{u})$ and $\sigma(\boldsymbol{u}) = \boldsymbol{A}\varepsilon(\boldsymbol{u})$ with $\boldsymbol{A}$ the linear positif operator on symmetric $d \times d$ matrix corresponding to the material propriety. Denote
$V_{\boldsymbol{g}} = \{\boldsymbol{v} \in H^1(\Omega)^d / \boldsymbol{v}_{|\Gamma_d} = \boldsymbol{g}\}$.
The Basic displacement variational formulation is : find $\boldsymbol{u} \in V_0(\Omega)$, such that :

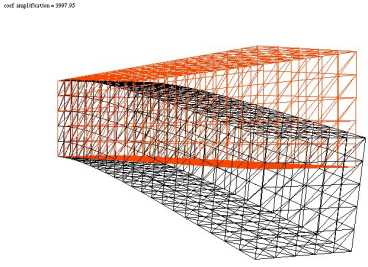
$$\int_\Omega \varepsilon(\boldsymbol{v}) : \boldsymbol{A}\varepsilon(\boldsymbol{u}) = \int_\Omega \boldsymbol{v}.\boldsymbol{f} + \int_\Gamma ((\boldsymbol{A}\varepsilon(\boldsymbol{u})).n).v, \quad \forall \boldsymbol{v} \in V_0(\Omega) \tag{5}$$

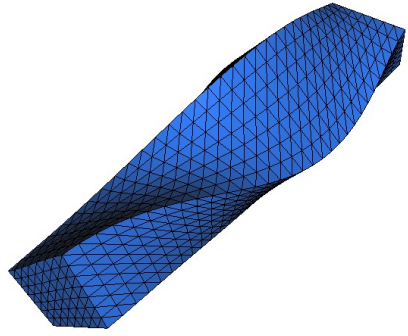
Linear elasticity equation, in FreeFem++

```
fespace Vh(Th,[P1,P1,P1]);
Vh [u1,u2,u3], [v1,v2,v3];
macro Strain(u1,u2,u3)
  [ dx(u1),  dy(u2),  dz(u3),
   (dz(u2) +dy(u3)),   (dz(u1)+dx(u3)),   (dy(u1)+dx(u2)) ]                    //     EOM
solve Lame([u1,u2,u3],[v1,v2,v3])=
  int3d(Th)(
      Strain(v1,v2,v3)'*(Aalu*Strain(u1,u2,u3))  )
- int3d(Th) ( rhoAlu*gravity*v3)
      + on(1,u1=0,u2=0,u3=0);

real coef= 0.1/u1[].linfty;  int[int] ref2=[1,0,2,0];
mesh3 Thm=movemesh3(Th,
     transfo=[x+u1*coef,y+u2*coef,z+u3*coef],
     label=ref2);
plot(Th,Thm, wait=1,cmm="coef  amplification = "+coef );
medit("Th-Thm",Th,Thm);
```

Lame equation / figure





Run:beam-3d.edp

Run:beam-EV-3d.edp

Run:beam-3d-Adapt.edp

Hyper elasticity equation

The Hyper elasticity problem is the minimization of the energy $W(I_1, I_2, I_3)$ where I_1, I_2, I_3 are the 3 invariants. For example The Ciarlet Geymonat energy model is

$$W = \kappa_1(J_1 - 3) + \kappa_2(J_2 - 3) + \kappa(J - 1) - \kappa \ln(J)$$

where $J_1 = I_1 I_3^{-\frac{1}{3}}$, $J_2 = I_2 I_3^{-\frac{2}{3}}$,$J = I_3^{\frac{1}{2}}$,
let u the deplacement, when

- $F = I_d + \nabla u$
- $C = {}^tFF$
- $I_1 = \mathrm{tr}(C)$
- $I_2 = \frac{1}{2}(\mathrm{tr}(C)^2 - \mathrm{tr}(C^2))$
- $I_3 = \det(C)$

The problem is find

$$u = \underset{u}{\mathtt{argmin}}\, W(I_1, I_2, I_3)$$

Hyper elasticity equation

```
fespace Wh(Th,[P2,P2]);
                                                //    methode de Newton ..
Wh [d1,d2]=[0,0];
Wh [w1,w2],[v1,v2];
for(int i=0;i<Nnewton;++i)
 {
    solve dWW([w1,w2],[v1,v2]) =
          int2d(Th)( ddW2d([d1,d2],[w1,w2],[v1,v2]) )
        - int2d(Th)( dW2d([d1,d2],[v1,v2]) -[v1,v2]'*[f1,f2] )
        + on(1,w1=0,w2=0);

     d1[] -= w1[];
     real err = w1[].linfty;
     if(err< epsNewton) break;
 }
```

Run:Hyper-Elasticity-2d.edp show:ElasticLaw2d.idp show:CiarletGemona.idp

Conclusion/Future

Freefem++ v3 is

- very good tool to solve non standard PDE in 2D/3D
- to try new domain decomposition domain algorithm

The the future we try to do :

- Build more graphic with VTK, paraview , ... (in progress)
- Add Finite volume facility for hyperbolic PDE (just begin C.F. FreeVol Projet)
- 3d anisotrope mesh adaptation
- automate the parallel tool

Thank for you attention.