

# FreeFem++, a user language to solve PDE.

F. Hecht, O. Pironneau, S. Auliac, F. Nataf, P. Jolivet, I. Danaila

For The Ninth Workshop of the INRIA-Illinois, Joint Laboratory on Petascale Computing

Laboratoire Jacques-Louis Lions Université Pierre et Marie Curie, Paris  
Projet Alpines, INRIA Rocquencourt.

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

- 1 Introduction
  - History
  - Main characteristics
  - The Kernel
- 2 Academic Examples
- 3 A first way to break complexity
- 4 Schwarz method with overlap
  - Poisson equation with Schwarz method
  - Transfer Part
  - parallel GMRES
  - A simple Coarse grid solver
  - Numerical experiment
- 5 Some tools and examples
  - Newton's Method
  - Anisotropic Mesh adaptation
  - Ipopt interface
  - Bose Einstein Condensate
  - Solid to Liquid and Natural Convection
  - Liquid to Solid with Natural Convection of water
- 6 Future/Conclusion

- 1 Introduction
  - History
  - Main characteristics
  - The Kernel
- 2 Academic Examples
- 3 A first way to break complexity
- 4 Schwarz method with overlap
- 5 Some tools and examples
- 6 Future/Conclusion

- 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@jll.math.upmc.fr> with 414 members with a flux of 5-20 messages per day.

More than 2000 true Users ( more than 100 download / month)

- 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.
- LU, Cholesky, Crout, CG, GMRES, UMFPack, SuperLU, MUMPS, HIPS , SUPERLU\_DIST, PASTIX. ... sparse linear solver ; eigenvalue and eigenvector computation with ARPACK.

- Automatic mesh generator, based on the Delaunay-Voronoi algorithm. (2d,3d (tetgen) )
- 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).
- Dynamic linking to add plugin.
- Full MPI interface
- Nonlinear Optimisation tools : CG, lpsolve, NLOpt, stochastic
- Wide range of examples : Navier-Stokes 3d, elasticity 3d, fluid structure, eigenvalue problem, Schwarz' domain decomposition algorithm, residual error indicator ...

## My early step in C++

```

typedef double R;

class Cvirt { public: virtual R operator()(R ) const =0;};

class Cfonc : public Cvirt { public:
    R (*f)(R); //      a function C
    R operator()(R x) const { return (*f)(x); }
    Cfonc( R (*ff)(R)) : f(ff) {} };

class Coper : public Cvirt { public:
    const Cvirt *g, *d; //      the 2 functions
    R (*op)(R,R); //      l'opération
    R operator()(R x) const { return (*op)((*g)(x), (*d)(x)); }
    Coper( R (*opp)(R,R), const Cvirt *gg, const Cvirt *dd):op(opp),g(gg),d(dd){}
    ~Coper(){delete g,delete d;} };

static R Add(R a,R b) {return a+b;}
static R Sub(R a,R b) {return a-b;}
static R Mul(R a,R b) {return a*b;}
static R Div(R a,R b) {return a/b;}
static R Pow(R a,R b) {return pow(a,b); }

```



A differential expression on in a PDE problem is like

$$f * [u_i | \partial_x u_i | \partial_y u_i | \dots] * [v_j | \partial_x v_j | \partial_y v_j | \dots]$$

where  $[f, |g, \dots]$  mean  $f$  or  $g$ , or  $\dots$ , and where the unknown part is  $[u_i | \partial_x u_i | \partial_y u_i | \dots] \equiv [(0, i) | (1, i) | (2, i) | \dots]$  is a pair of  $i' \times i$ , if we do the same of the test part, the differential expression is a formally sum of :

$$\sum_k f_k \times (i'_k, i_k, j'_k, j_k)$$

So we can easily code this syntax :

```
varf a(u,v) = int2d(Th) (Grad(u)'*Grad(v)) - int2d(Th) (f*v)
               + on(1, u=0);
matrix A=a(Vh,Vh, solver=UMFPACK);
real [int] b=a(0,Vh);
u[] = A^-1 * b;
```

- 1 Introduction
- 2 Academic Examples**
- 3 A first way to break complexity
- 4 Schwarz method with overlap
- 5 Some tools and examples
- 6 Future/Conclusion

# Laplace equation, weak form

Let a domain  $\Omega$  with a partition of  $\partial\Omega$  in  $\Gamma_2, \Gamma_e$ .

Find  $u$  a solution in such that :

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

Denote  $V_g = \{v \in H^1(\Omega) / v|_{\Gamma_2} = g\}$  .

The Basic variational formulation with is : find  $u \in V_2(\Omega)$  , such that

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

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

# Laplace equation in FreeFem++

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

```
mesh3 Th("fish3d.msh"); // read a mesh 3d
fespace Vh(Th,P1);      // define the P1 EF space

Vh u,v; // set test and unknow FE 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  $\gamma_2$ 
plot(u,fill=1,wait=1,value=0,wait=1);
```

Run:fish.edp      Run:fish3d.edp

- 1 Introduction
- 2 Academic Examples
- 3 A first way to break complexity**
- 4 Schwarz method with overlap
- 5 Some tools and examples
- 6 Future/Conclusion

# A first way to break complexity

Idea :

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v$$

take an equi-partition of  $\Omega$  in  $\Omega_i$  for  $i = 0$  to  $N_p - 1$  the number of processor.  
then

$$a(u, v) = \sum_{i=0}^{N_p-1} \int_{\Omega_i} \nabla u \cdot \nabla v$$

# A first way to break complexity

- 1 Build matrix in parallel by assembling par region remark with the change function you change the region numbering to build region.

```
real c = mpisize/real(Th.nt) ;  
Th=change(Th,fregion= min(mpisize-1,int(nuTriangle*c))) ;
```

- 2 Assemble the full matrix

```
varf vlaplace(uh,vh) = // definition de problem  
    int3d(Th,mpirank)( uh*vh+ dt*Grad(uh)'*grad(vh) )  
    + int3d(Th,mpirank)( dt*vh*f) + on(1,uh=g) ;  
matrix A,Ai = vlaplace(Vh,Vh,tgv=ttgv) ;  
mpiAllReduce(Ai,A,mpiCommWorld,mpiSUM) ; // assemble in //
```

- 3 Solve the linear using a good parallel solver (MUMPS)

```
load "MUMPS_FreeFem"  
uh[] = A^-1*b ; // resolution
```

Run:Heat3d.edp

Run:NSCaraCyl-100-mpi.edp

- 1 Introduction
- 2 Academic Examples
- 3 A first way to break complexity
- 4 Schwarz method with overlap**
  - Poisson equation with Schwarz method
  - Transfer Part
  - parallel GMRES
  - A simple Coarse grid solver
  - Numerical experiment
- 5 Some tools and examples
- 6 Future/Conclusion



To solve the following Poisson problem on domain  $\Omega$  with boundary  $\Gamma$  in  $L^2(\Omega)$  :

$$-\Delta u = f, \text{ in } \Omega, \text{ and } u = g \text{ on } \Gamma,$$

where  $f \in L^2(\Omega)$  and  $g \in H^{\frac{1}{2}}(\Gamma)$  are two given functions.

Let introduce  $(\pi_i)_{i=1,\dots,N_p}$  a positive regular partition of the unity of  $\Omega$ , q-e-d :

$$\pi_i \in \mathcal{C}^0(\Omega) : \quad \pi_i \geq 0 \text{ and } \sum_{i=1}^{N_p} \pi_i = 1.$$

Denote  $\Omega_i$  the sub domain which is the support of  $\pi_i$  function and also denote  $\Gamma_i$  the boundary of  $\Omega_i$ .

The parallel Schwarz method is Let  $\ell = 0$  the iterator and a initial guest  $u^0$  respecting the boundary condition (i.e.  $u^0|_{\Gamma} = g$ ).

$$\forall i = 1.., N_p : \quad -\Delta u_i^\ell = f, \text{ in } \Omega_i, \quad \text{and } u_i^\ell = u^\ell \text{ on } \Gamma_i \quad (3)$$

$$u^{\ell+1} = \sum_{i=1}^{N_p} \pi_i u_i^\ell \quad (4)$$

## Some Remark

We never use finite element space associated to the full domain  $\Omega$  because it is too expensive. So we use on each domain  $i$  we defined  $J_i = \{j \in 1, \dots, N_p / \Omega_i \cap \Omega_j \neq \emptyset\}$  and we have

$$(u^{\ell+1})|_{\Omega_i} = \sum_{j \in J_i} (\pi_j u_j^\ell)|_{\Omega_i} \quad (5)$$

We denote  $u_{h|i}^\ell$  the restriction of  $u_h^\ell$  on  $V_{hi}$ , so the discrete problem on  $\Omega_i$  of problem (3) is find  $u_{hi}^\ell \in V_{hi}$  such that :

$$\forall v_{hi} \in V_{0i} : \int_{\Omega_i} \nabla v_{hi} \cdot \nabla u_{hi}^\ell = \int_{\Omega_i} f v_{hi},$$

$$\forall k \in \mathcal{N}_{hi}^{\Gamma_i} : \sigma_i^k(u_{hi}^\ell) = \sigma_i^k(u_{h|i}^\ell)$$

where  $\mathcal{N}_{hi}^{\Gamma_i}$  is the set of the degree of freedom (Dof) on  $\partial\Omega_i$  and  $\sigma_i^k$  the Dof of  $V_{hi}$ .

## Transfer Part equation(5)

To compute  $v_i = (\pi_i u_i)|_{\Omega_i} + \sum_{j \in J_i} (\pi_j u_j)|_{\Omega_i}$  and can be write the freefem++ function Update with asynchronous send/recv (**Otherwise dead lock**).

```
func bool Update(real[int] &ui, real[int] &vi)
{
  int n= jpart.n;
  for(int j=0;j<njpart;++j)  Usend[j][]=sMj[j]*ui;
  mpiRequest[int]  rq(n*2);
  for (int j=0;j<n;++j)
    Irecv(processor(jpart[j],comm,rq[j  ]), Ri[j][]);
  for (int j=0;j<n;++j)
    Isend(processor(jpart[j],comm,rq[j+n]), Si[j][]);
  for (int j=0;j<n*2;++j)
    int k= mpiWaitAny(rq);
  vi = Pii*ui;
                                     //      set to  $(\pi_i u_i)|_{\Omega_i}$ 
                                     //      apply the unity local partition .
  for(int j=0;j<njpart;++j)
    vi += rMj[j]*Vrecv[j][];
                                     //      add  $(\pi_j u_j)|_{\Omega_i}$ 
  return true; }
```

Finally you can easily accelerate the fixe point algorithm by using a parallel GMRES algorithm after the introduction the following affine  $\mathcal{S}_i$  operator sub domain  $\Omega_i$ .

```
func real[int] Si(real[int]& U) {
  real[int] V(U.n) ; b= onG .* U;
  b = onG? b : Bi;
  V = Ai^-1*b; // (3)
  Update(V,U); // (??)
  V -= U; return V; }
```

Where the parallel MPIGMRES or MPICG algorithm is to solve  $A_i x_i = b_i, i = 1, \dots, N_p$  by just changing the dot product by reduce the local dot product of all process with the following MPI code :

```
template<class R> R ReduceSum1(R s,MPI_Comm * comm)
{
  R r=0;
  MPI_Allreduce( &s, &r, 1 ,MPI_TYPE<R>::TYPE(),
                MPI_SUM, *comm );
  return r; }
```

A simple coarse grid is we solve the problem on the coarse grid :

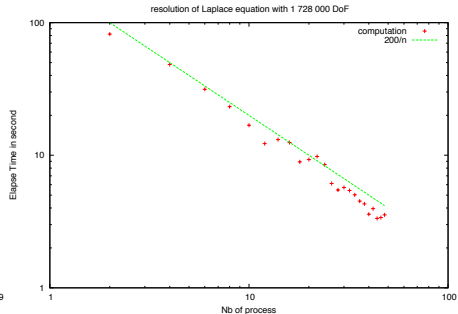
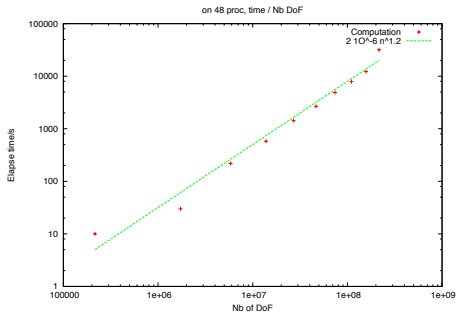
```
func bool CoarseSolve(real[int]& V, real[int]& U,
                    mpiComm& comm)
{
    if(AC.n==0 && mpiRank(comm)==0)           // first time build
        AC = vPbC(VhC,VhC,solver=sparsesolver);
    real[int] Uc(Rci.n),Bc(Uc.n);
    Uc= Rci*U;                                // Fine to Coarse
    mpiReduce(Uc,Bc,processor(0,comm),mpiSUM);
    if(mpiRank(comm)==0)
        Uc = AC^-1*Bc;                        // solve of proc 0
    broadcast(processor(0,comm),Uc);
    V = Pci*Uc;                                // Coarse to Fine
}
```

Limitation : if the initial problem, data have oscillation, you must use homogenization technic on coarse problem, or use the F. Nataf and co, preconditionner.

So we finally we get 4 algorithms

- 1 The basic schwarz algorithm  $u^{\ell+1} = \mathcal{S}(u^\ell)$ , where  $\mathcal{S}$  is one iteration of schwarz process.
- 2 Use the GMRES to find  $u$  solution of the linear system  $\mathcal{S}u - u = 0$ .
- 3 Use the GMRES to solve parallel problem  $\mathcal{A}_i u_i = b_i$ ,  $i = 1, \dots, N_p$ , with RAS preconditionneur
- 4 Use the method with two level preconditionneur RAS and Coarse.

On the SGI UV 100 of the lab :



We consider first example in an academic situation to solve Poisson Problem on the cube  $\Omega = ]0, 1[^3$

$$-\Delta u = 1, \text{ in } \Omega; \quad u = 0, \text{ on } \partial\Omega. \quad (6)$$

With a cartesian meshes  $\mathcal{T}_{hn}$  of  $\Omega$  with  $6n^3$  tetrahedron, the coarse mesh is  $\mathcal{T}_{hm}^*$ , and  $m$  is a divisor of  $n$ .

We do the validation of the algorithm on a Laptop Intel Core i7 with 4 core at 1.8 Ghz with 4Go of RAM DDR3 at 1067 Mhz,

Run:DDM-Schwarz-Lap-2dd.edp

Run:DDM-Schwarz-Lame-3d.edp

Run:DDM-Schwarz-Lame-2d.edp

Run:DDM-Schwarz-Stokes-2d.edp

- 1 Introduction
- 2 Academic Examples
- 3 A first way to break complexity
- 4 Schwarz method with overlap
- 5 Some tools and examples**
  - Newton's Method
  - Anisotropic Mesh adaptation
  - Ipopt interface
  - Bose Einstein Condensate
  - Solid to Liquid and Natural Convection
  - Liquid to Solid with Natural Convection of water



To solve  $F(u) = 0$  the Newton's algorithm is

- 1  $u^0$  a initial guest
- 2 do
  - find  $w^n$  solution of  $DF(u^n)w^n = F(u^n)$
  - $u^{n+1} = u^n - w^n$
  - if(  $\|w^n\| < \varepsilon$ ) break ;

For Navier Stokes problem the algorithm is :  $\forall v, q,$

$$F(u, p) = \int_{\Omega} (u \cdot \nabla) u \cdot v + \nu \nabla u : \nabla v - q \nabla \cdot u - p \nabla \cdot v + BC$$

$$\begin{aligned} DF(u, p)(w, w_p) &= \int_{\Omega} (w \cdot \nabla) u \cdot v + (u \cdot \nabla) w \cdot v \\ &\quad + \int_{\Omega} \nu \nabla w : \nabla v - q \nabla \cdot w - w_p \nabla \cdot v + BC0 \end{aligned}$$

Run:cavityNewtow.edp

Run:NSNewtonCyl-100-mpi.edp

Run:Hyper-Elasticity-2d.edp

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

$$|\gamma| = \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle} dt$$

We introduce the metric  $\mathcal{M}(x)$  as a field of  $d \times d$  symmetric positive definite matrices, and the length  $\ell$  of  $\Gamma$  w.r.t  $\mathcal{M}$  is :

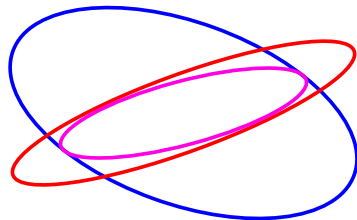
$$\ell = \int_0^1 \sqrt{\langle \gamma'(t), \mathcal{M}(\gamma(t)) \gamma'(t) \rangle} dt$$

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

# The main IDEA for mesh generation

- The difficulty is to find a tradeoff between the error estimate and the mesh generation, because these two works are strongly different.
- To do that, we propose a way based on a metric  $\mathcal{M}$  and unit mesh w.r.t  $\mathcal{M}$
- The metric is a way to control the mesh size.
- remark : The class of the mesh which can be created by the metric, is very large.

**Idea : Metric intersection** The unit ball  $\mathcal{B}_{\mathcal{M}}$  in a metric  $\mathcal{M}$  plot the maximum mesh size on all the direction, is an ellipse. If you 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 :  $\mathcal{B}_{\mathcal{M}_{uv}} \subset \mathcal{B}_{\mathcal{M}_u} \cap \mathcal{B}_{\mathcal{M}_v}$

# 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

- $L^\infty$  :  $\mathcal{M} = \frac{1}{\varepsilon} |\nabla \nabla u| = \frac{1}{\varepsilon} |\mathcal{H}|$  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  $P_\ell$  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 )

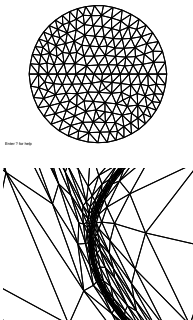
- for  $P_1$  :  $\mathcal{M}_1 = \mathcal{H}^2$  (sub optimal with acute triangle take  $\mathcal{H}$ )
- for  $P_2$  :  $\mathcal{M}_2 = 3 \sqrt{\begin{pmatrix} a & b \\ b & c \end{pmatrix}^2 + \begin{pmatrix} b & c \\ c & a \end{pmatrix}^2}$  with  
 $D^{(3)}u(x, y) = (ax^3 + 3bx^2y + 3cxy^2 + dy^3)/3!$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

# Example of adaptation process

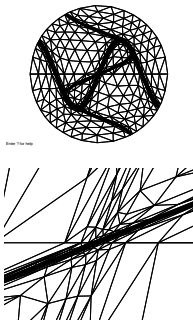
Find optimal mesh in norm  $L^\infty$  to represent :

$$u = (10 * x^3 + y^3) + \text{atan2}(0.001, (\sin(5 * y) - 2 * x))$$

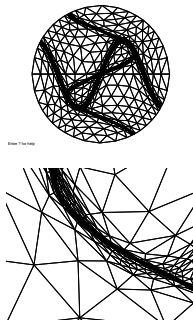
$$v = (10 * y^3 + x^3) + \text{atan2}(\textcolor{red}{0.01}, (\sin(5 * x) - 2 * y)).$$



Run:Adapt-uv.edp



Run:CornerLap.edp



Run:Laplace-Adapt-3d.edp

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 :

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

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

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 :

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

to code that in FreeFem++

use

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

Run: BEC.edp

# Solid to Liquid and Natural Convection

The starting point 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  $\theta_r$  (resp  $\theta_l$ ) at the right and left boundaries.

The model is : find the field : the velocity  $\mathbf{u} = (u_1, u_2)$ , the pressure  $p$  and temperature  $\theta$  :

$$\left\{ \begin{array}{lll} \mathbf{u} & \text{given} & \text{in } \Omega_s \\ \partial_t \mathbf{u} + (\mathbf{u} \nabla) \mathbf{u} + \nabla \cdot \mu \nabla \mathbf{u} + \nabla p & = -c_T \mathbf{e}_2 & \text{in } \Omega_f \\ \nabla \cdot \mathbf{u} & = 0 & \text{in } \Omega_f \\ \partial_t \theta + (\mathbf{u} \nabla) \theta + \nabla \cdot k_T \nabla \theta & = \partial_t S(T) & \text{in } \Omega \end{array} \right. \quad (7)$$

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



The enthalpy of the change of phase is given by the function  $S$ ;  $\mu$  is the relative viscosity,  $k_T$  the thermal diffusivity.

In  $\Omega_f = \{x \in \Omega; \theta > \theta_f\}$ , with  $\theta_m$  the melting temperature the solid has melt.

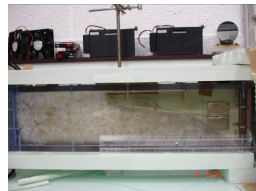
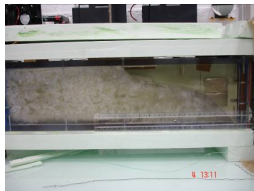
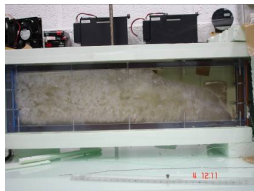
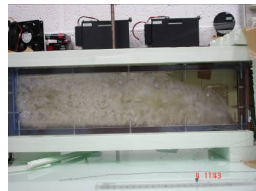
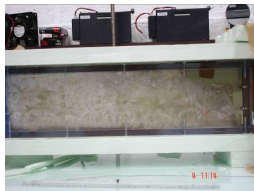
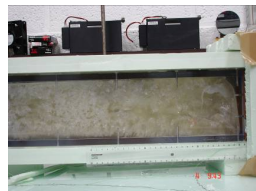
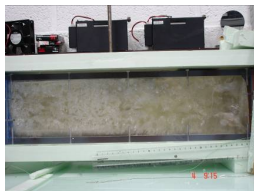
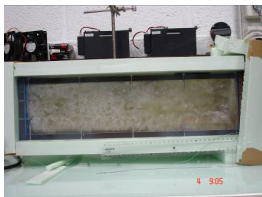
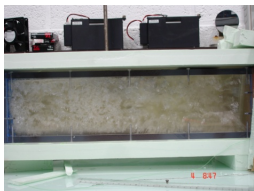
We modeled, the solid phase as a fluid with huge viscosity, so :

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

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



We apply a fixed point algorithm for the phase change part (the domain  $\Omega_f$  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  $\Omega_s$  the method also diverge.

This is a really difficult problem.

The finite element space to approximate  $u_1, u_2, p, \theta$  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  $\Omega_s$ 
{ nu = nuT; // recompute the viscosity in  $\Omega_s, \Omega_f$ 
  for(int niter=0;niter<20; ++niter) //Newton loop
  { BoussinesqNL;
    err = ulw[].linfo;
    cout << niter << "_err_NL_" << err << endl;
    ul[] -= ulw[];
    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.

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

The fusion temperature will be denoted by  $T_f$ . Using a lengthscale  $L_{ref} = H$  and a liquid reference state  $(\rho_{ref}, V_{ref}, T_{ref})$ , we can define the following scaling for the space, velocity, temperature and time variables :

$$\vec{x} = \frac{\vec{X}}{L_{ref}}, \quad \vec{u} = \frac{\vec{U}}{V_{ref}}, \quad \theta = \frac{T - T_{ref}}{T_h - T_c}, \quad t = \frac{\tau}{t_{ref}}, \quad t_{ref} = L_{ref}/V_{ref}.$$

$$f_B(\theta) = \frac{\mathcal{Ra}}{\mathcal{Pr} \mathcal{Re}^2} \theta, \quad \text{The buoyancy force} \quad (8)$$

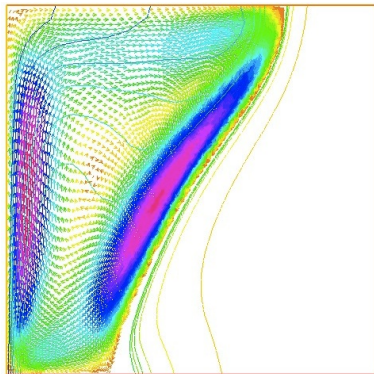
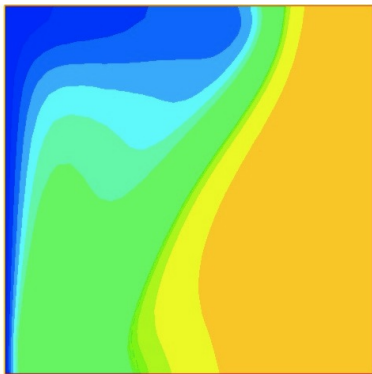
where the Reynolds, Prandtl and Rayleigh numbers, are defined as :

$$\mathcal{Re} = \frac{\rho_{ref} V_{ref} L_{ref}}{\mu_l}, \quad \mathcal{Pr} = \frac{\nu_l}{\alpha_l}, \quad \mathcal{Ra} = \frac{g \beta L_{ref}^3 (T_h - T_c)}{\nu_l \alpha_l}, \quad (9)$$

with  $\mu$  denoting the viscosity,  $\nu$  the kinematic viscosity,  $\alpha$  the thermal diffusivity,  $\beta$  the thermal expansion coefficient and  $g$  the gravitational acceleration.



# Phase change with Natural Convection



So now, a real problem, get the physical parameter of the real experiment.

Run:Orange-Newton.edp

Pure water exhibits a nonlinear density variation for  $T < 10.2^\circ\text{C}$  with a maximum at  $T_m = 4.0293^\circ\text{C}$ . We use below the following density-temperature relationship :

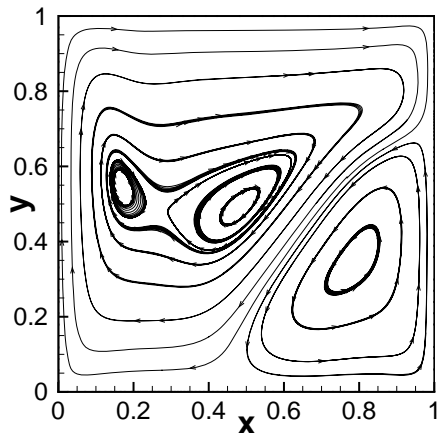
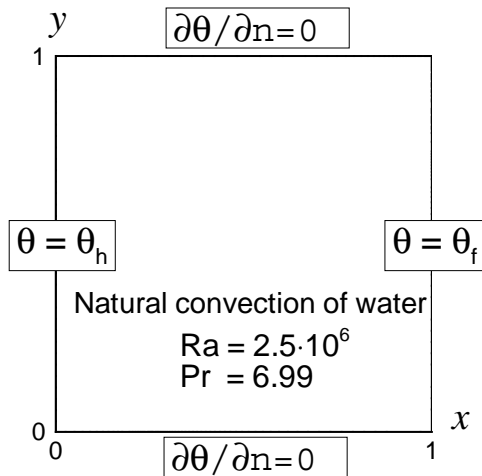
$$\rho(T) = \rho_m (1 - w |T - T_m|^q), \quad (10)$$

with  $\rho_m = 999.972 \text{ [kg/m}^3\text{]}$ ,  $w = 9.2793 \cdot 10^{-6} [(\text{C})^{-q}]$ , and  $q = 1.894816$ . Choosing the fusion temperature  $T_f = 0^\circ\text{C}$  as reference, the bouyancy term  $f_B = g(\rho_{ref} - \rho)/\rho_{ref}$  appearing becomes after scaling :

$$f_B(\theta) = \frac{\mathcal{Ra}}{\mathcal{Pr} \mathcal{Re}^2} \frac{1}{\beta(T_h - T_c)} \frac{\rho(\theta_f) - \rho(\theta)}{\rho(\theta_f)}, \quad (11)$$

where  $\beta = (1/\rho_m) (d\rho/dT)$  is the thermal expansion coefficient with the value  $\beta = 6.91 \cdot 10^{-5} \text{ [(K)}^{-1}\text{]}$ . Note that (11) compared to the classical linear form (8) non only introduces a new nonlinear term, but also the coefficient in front of this term is very large, since proportional to  $\mathcal{Ra}/(\beta(T_h - T_c))$ .

# Natural convection of water to $10^\circ$ to $0^\circ$



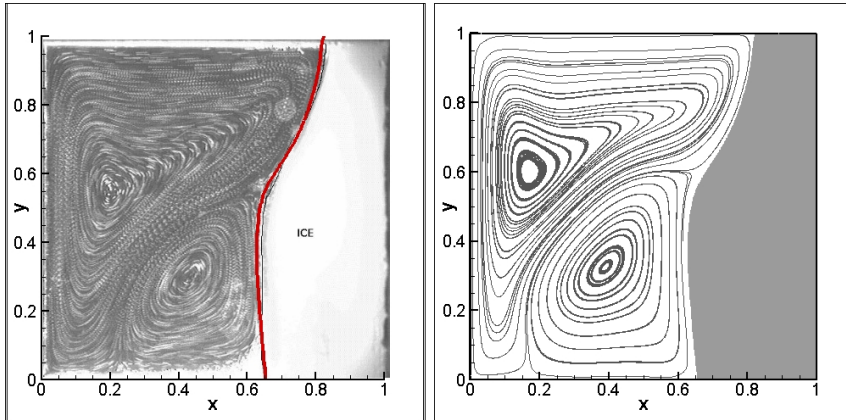
Natural convection of water in a differentially heated cavity. Problem definition and streamlines of the steady flow.

[Movie:flow](#)

[Movie:mesh](#)

[Movie:flows Region](#)

# Freezing of pure water



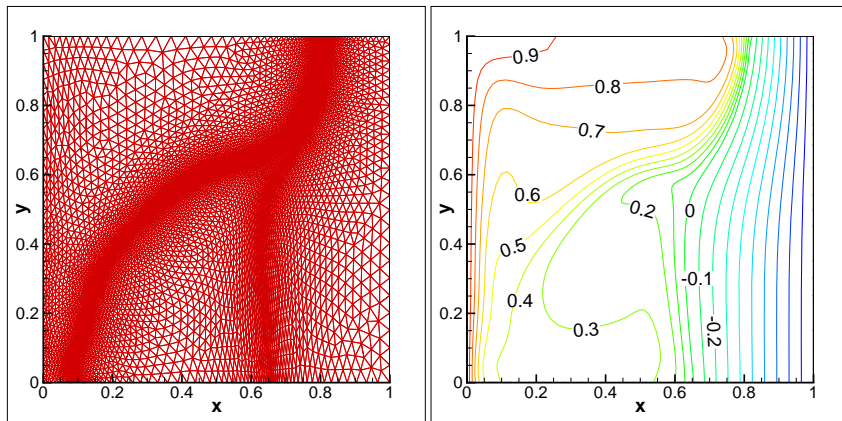
Freezing of pure water : Configuration at (physical time)  $\tau = 2340[s]$  : (a) experimental image ; the thick red line represents the solid-liquid interface computed with the present method (b)

Movie:flow

Movie:mesh

Movie:flows Region

# Freezing of pure water



Freezing of pure water. Computed configuration at (physical time)  $\tau = 2340[s]$  : (a) finite-element mesh refined along the solid-liquid interface ( $T = 0^\circ\text{C}$ ) and also along the line of maximum water density ( $T = 4^\circ\text{C}$ ) (b) temperature iso-lines.

- 1 Introduction
- 2 Academic Examples
- 3 A first way to break complexity
- 4 Schwarz method with overlap
- 5 Some tools and examples
- 6 Future/Conclusion**

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.