

A programming language for the FEM: FreeFem++

F. Hecht,

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

with O. Pironneau, P-H. Tournier, P. Jolivet

<http://www.freefem.org>

<mailto:frederic.hecht@upmc.fr>



Outline

- 1 Introduction
- 2 Tools
- 3 Academic Examples
- 4 Numerics Tools
- 5 Schwarz method with overlap
- 6 Exercises
- 7 Non Linear Problem
- 8 Technical Remark on freefem++



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



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 their own script to solve non 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 runs on Mac, Unix and Windows architecture, in parallel with MPI.

To try it out go to <https://www.ljll.math.upmc.fr/lehyaric/ffjs/>

The FreeFem++ days, 2nd week of December, 2016, UPMC, Jussieu, Paris, France

Info: FreeFem++ solves a problem with $22 \cdot 10^9$ unknowns in 200 s on 12,000 proc.



1 Introduction

- History

- The main characteristics
- The Changes from sep. 2015 to dec. 2016
- Basement
- Weak form



- 1987 MacFem/PCFem the old ones (O. Pironneau in Pascal) no free.
- 1992 FreeFem rewrite in C++ (P1,P0 one mesh) O. Pironneau, D. Bernardi, F. Hecht , C. Prudhomme (mesh adaptation , bamg).
- 1996 FreeFem+ rewrite in C++ (P1,P0 more mesh) O. Pironneau, D. Bernardi, F. Hecht (algebra of fucntion).
- 1998 FreeFem++ rewrite with an other finite element kernel and an new language ; F. Hecht, O. Pironneau, K.Ohtsuka.
- 1999 FreeFem 3d (S. Del Pino) , a fist 3d version base on fictitious domaine method.
- 2008 FreeFem++ v3 use a new finite element kernel multidimensionnels: 1d,2d,3d...
- 2014 FreefEM++ v3.34 parallel version



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@1j1.math.upmc.fr> with 410 members with a flux of 5-20 messages per day.

More than 2000 true Users (more than 1000 download / month)



1 Introduction

- History
- The main characteristics
- The Changes from sep. 2015 to dec. 2016
- Basement
- Weak form



- 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, PETSc. ... sparse linear solver; eigenvalue and eigenvector computation with ARPACK.
- Online graphics with OpenGL/GLUT/VTK, C++ like syntax.
- An integrated development environment FreeFem++-cs



- Analytic description of boundaries, with specification by the user of the intersection of boundaries in 2d.
- **Automatic mesh generator**, based on the Delaunay-Voronoi 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, **Iopt**, NLOpt, stochastic
- Wide range of examples: Navier-Stokes **3d**, elasticity **3d**, fluid structure, eigenvalue problem, Schwarz' domain decomposition algorithm, residual error indicator ...



1 Introduction

- History
- The main characteristics
- The Changes from sep. 2015 to dec. 2016
- Basement
- Weak form



The Changes from feb. 2016 to dec 2016

- pass to version 5.0.2 of MUMPS
- add new stuff in arpack interface
- add plot of multi curve (array of array int 2d, 3d , with color)
- add new function in examples++-tutorial/movemeshsmooth.idp to do movemesh with some Laplacien Smoothing (see file for doc.)
- correct problem of size of the arrowhead when zooming
- add tools to extract border of 3d surface version 3.49
- add tool to compute integral with mapping on test or unknown function (in test just in 2d to day and only on varf definition)
`int1d(Th, mapu=[x+1,y], mapt=[x-1,y]) (...)
int2d(Th, mapu=[x+1,y], mapt=[x-1,y]) (...)`
- add distance plugin to computation the signed distance to a curve defined by a iso isovalue
- change fixeborder in fixedborder in buildmesh version 3.48 (7 sep 2016)
- add doc for time depend Stokes problem with matrix (O. Pironneau)
- add doc for time depend Heat problem with matrix (O. Pironneau)
- correct a very old bug
- add spline interface of gsl library in gsl plugin see [gsl.edp examples](#). version 3.47 (1/june/2016)
- add solveurs FETI and BDD in HPDDM
- add new finite element to store value a quadrature point. in plugin Element_QF
- add gluemesh of 3d mesh
- add operator `b = A * c; // where is real matrix avec b and c complexe vector.` version 3.45 (10 march 2016)
- correct very old bug in fflut, some time we miss wait due to asynchroneous stuff and we can have memory error for the same reason .
- try use lot of preinstall software
- pass to tetgen version 1.5.1-beta (the last one)
- add Th.hmin and Th.hmax methode to Mesh type version 3.44-1
- add new hpddm example for Stokes probleme (Thank to P. Jolivet)
- put the new convect 3d as the default, (newconvect do nothing now). 25/02/16
- try to add new version do 3d convect (add newconvect=1;) in test ... version 3.44 (22 feb. 2016)
- add tools to extract border after movemesh in Curvature plugin)
- restruct the hpddm and petsc plugin (P. Jolivet) version 3.43-3
- remove some spurious print when verbosity = 0
- add initialisation of array which string parameter
- add setw iomanip stuff version 3.34-2



The Changes from sep. . 2015 to feb. 2016

- change RT03d finite element interpolation (put 1 point integration by face)
- add renum=false in all trunc fonction of DDM examples
- warning change the default no renumbering to be sure trunc commute the two mesh must be the same truc(trunc(Th,I),J) == truc(trunc(Th,J),I)
- correct of the formal operator trace (bug was compute a00 + trace(A), thank to O. Pironneau)
- put hppdm source include in the distribution to have more safe compile process.
- update a little the documentation (add new finite element)
- add new finite element just for interpolate partition of unity in DDM / HPDDM method Edge23ds0, Edge13ds0,Edge03d in plugin Element_Mixte3d
- add new finite element P1bl, P1bl3d in plugin Element_P1bl
- add new finite element Edge23d, P1bl,in plugin Element_Mixte3d
- change EPSD in eigenv because trouble in some very rare case (see G. Vergez)

- correct problem for vectoral solution on implicit declaration of product of simple EF like P1b
- add tools to compute curvature of border in Cuvature plugin
- all implicit loop for lot of array : real[ini], real[int,int], real[string] ,matrix , string[string] for [i,bi: A] bi=f(i);
- change the Chinese documentation (thank to Helin GONG) version 3.42
- correct optimisation of convect in case on windows bug)
- add tool do build quadrature formule when we split triangle in 3 triangle for HCT element.
- add new quadrature formula 3d exact form degré 6 to 14 in qf11to25.cpp the name of qfV quadrature are qfVp1 .. qfVp14
- correct optimisation in convect for navier stokes problem.. for this kind of expression + int2d(Th) (-alpha*convect([up1,up2],-dt,up1)*v1 -alpha*convect([up1,up2],-dt,up2)*v2) in this case now only on characteristics is computed
- correct trouble in configure this /usr/bin/machine
- map&key to know if a key exist in a map int[string], version 3.41
- add tools to get mesh of fespace Vh in 2d and 3d mesh th2= Vh2d.Th ; mesh3 th3= Vh3d.Th ; version 3.40 (14 sep. 2015)
- add new C1 Finite element HCT see examples++-load/bilabHCT.edp



1 Introduction

- History
- The main characteristics
- The Changes from sep. 2015 to dec. 2016
- Basement
- Weak form



Element of syntax : the script is like in C/C++

First FreeFem++ is a compiler and after it launch the create code (a kind of byte code).

The language is polymorphe but it is not a objet oriented language.

The key words are reserved and the operator is like in C exempt: ^ & |

+ - * / ^ // $a^b = a^b$
== != < > <= >= & | // $a|b \equiv a \text{ or } b, a\&b \equiv a \text{ and } b$
= += -= /= *=

BOOLEAN: 0 \leftrightarrow `false`, \neq 0 \leftrightarrow `true` = 1

// Automatic cast for numerical value : bool, int, reel, complex , so
`func heavyside = real(x>0.);`

`for (int i=0;i<n;i++) { ... ;}`
`if (<bool exp>) { ... ;} else { ...;};`
`while (<bool exp>) { ... ;}`
`break continue` key words

weakless: all local variables are almost static (????)
bug if break before variable declaration in same block.
bug for fespace argument or fespace function argument



Element of syntax: special word for finite element

```
x,y,z                                // current coord.  
label, region                         // label of BC (border) , (interior)  
N.x, N.y, N.z,                         // normal  
int i = 0;                            // an integer  
real a=2.5;                           // a reel  
bool b=(a<3.);  
real[int] array(10) ;                  // a real array of 10 value  
mesh Th; mesh3 Th3;                   // a 2d mesh and a 3d mesh  
fespace Vh(Th,P2);                   // Def. of 2d finite element space;  
fespace Vh3(Th3,P1);                 // Def. of 3d finite element space;  
Vh u=x;                             // a finite element function or array  
Vh3<complex> uc = x+ 1i *y;        // complex valued FE  
u(.5,.6,.7);                        // value of FE function u at point (.5,.6,.7)  
u[];                                 // the array of DoF value assoc. to FE function u  
u[][5];                             // 6th value of the array (numbering begin  
                                // at 0 like in C)
```



Element of syntax: weak form, matrix and vector 2/4

```
fespace V3h(Th, [P2,P2,P1]);
V3h [u1,u2,p]=[x,y,z]; // a vectorial finite element
// function or array
// remark u1[] <==> u2[] <==> p[] same array of unknown.
macro div(u,v) (dx(u)+dy(v)) // EOM a macro
// (like #define in C )
macro Grad(u) [dx(u),dy(u)] // the macro end with //
varf a([u1,u2,p],[v1,v2,q])=
int2d(Th)(Grad(u1)'*Grad(v1)+Grad(u2)'*Grad(v2)
-div(u1,u2)*q -div(v1,v2)*p)
+on(1,2,u1=g1,u2=g2);

matrix A=a(V3h,V3h,solver=UMFPACK);
real[int] b=a(0,V3h);
u2[] =A^-1*b; // or you can put also u1[]= or p[].
```



Element of syntax: matrix and vector tools

```
func Heaveside=(x>0);           // a formal line function
func real g(int i, real a) { .....; return i+a;}
A = A + A'; A = A'*A           // matrix operation (only 1/1)
A = [ [ A,0],[0,A' ] ];        // Block matrix.
int[int] I(15),J(15);          // two array for renumbering
                                // the aim is to transform a matrix into a sparse matrix
matrix B;
B = A;                         // copie matrix A
B=A(I,J);                      // B(i,j) = A(I(i),J(j))
B=A(I^-1,J^-1);                // B(I(i),J(j))= A(i,j)
B.resize(10,20);                // resize the sparse matrix
                                // and remove out of bound terms
int[int] I(1),J(1); real[int] C(1);
[I,J,C]=A;                     // get of the sparse term of the matrix A
                                // (the array are resized)
A=[I,J,C];                    // set a new matrix
matrix D=[diagofA] ;            // set a diagonal matrix D
                                // from the array diagofA.
real[int] a=2:12;              // set a[i]=i+2; i=0 to 10.
```



Element of syntax formal computation on array

a formal array is [exp1, exp1, ..., expn]

the Hermitian transposition is [exp1, exp1, ..., expn]'

```
complex a=1,b=2,c=3i;
func va=[ a,b,c];                                //      is a formal array in []
a =[ 1,2,3i]*va ; cout << a << endl;          //      Hermitian product
matrix<complex> A=va*[ 1,2,3i]'; cout << A << endl;
a =[ 1,2,3i]*va*2.;
a =(va+[ 1,2,3i])*va*2.;
va./va;                                         //      term to term /
va.*va;                                         //      term to term *
trace(va*[ 1,2,3i]') ;                         //
(va*[ 1,2,3i]')[1][2] ;                         //      get coef
det([[1,2],[-2,1]]);                            //      just for matrix 1x1 et 2x2
usefull macro to def your edp.
macro grad(u) [dx(u),dy(u)] // 
macro div(u1,u2) (dx(u1)+dy(u2)) //
```



List of Plugin

```
ls /usr/local/lib/ff++/3.50/lib/
```

BEC.dylib	MUMPS_seq.dylib	ff-cmaes.dylib	mmg3d-v4.0.dylib
BernadiRaugel.dylib	MetricKuate.dylib	fflapack.dylib	msh3.dylib
BinaryIO.dylib	MetricPk.dylib	ffnewuoa.dylib	mshmet.dylib
ClosePoints.dylib	Morley.dylib	ffrandom.dylib	myfunction.dylib
Curvature.dylib	NewSolver.dylib	freeyams.dylib	myfunction2.dylib
DxWriter.dylib	PARDISO.dylib	funcTemplate.dylib	pcm2rnm.dylib
Element_HCT.dylib	SuperLu.dylib	gmsh.dylib	pipe.dylib
Element_Mixte.dylib	UMFPACK64.dylib	gsl.dylib	ppm2rnm.dylib
Element_Mixte3d.dylib	VTK_writer.dylib	ilut.dylib	qfl1to25.dylib
Element_P1bl.dylib	VTK_writer_3d.dylib	iohdf5.dylib	scotch.dylib
Element_P1dc1.dylib	addNewType.dylib	iovtk.dylib	shell.dylib
Element_P3.dylib	aniso.dylib	isoline.dylib	splitedges.dylib
Element_P3dc.dylib	bfstream.dylib	isolineP1.dylib	splitmesh3.dylib
Element_P4.dylib	dfft.dylib	lapack.dylib	splitmesh6.dylib
Element_P4dc.dylib	distance.dylib	lgbmo.dylib	symmetrizeCSR.dylib
Element_PkEdge.dylib	exactpartition.dylib	mat_dervieux.dylib	tetgen.dylib
Element_QF.dylib	ff-AiryBiry.dylib	mat_psi.dylib	thresholdings.dylib
FreeFemQA.dylib	ff-Ipopt.dylib	medit.dylib	
MUMPS.dylib	ff-NLopt.dylib	metis.dylib	



Important Plugin

- qf11to25 add more quadrature formulae in 1d , 2d, and tools to build own quadrature
- Element_*,Morlay,BernadiRaugel add new kind of 2d finite element
- SuperLu,UMFPACK64,SuperLu,MUMPS_seq add sequentiel sparse solver
- metis,scotch mesh Partitioning
- ffrandom true random number generator: srandomdev,srandom, random
- gsl the gsl lib interface (lot of special function)
- shell,pipe directory and file interface, pipe interface
- dfft interface with fftw3 library for FFT.
- **msh3,tetgen** 3d mesh tools and tetgen interface
- lapack a small lacpack,interface of full linear solver, full eigen value problem.
- ff-Ipopt interface with Ipopt optimisation software
- ppm2rnm interface with ppm library to read ppm bitmap.
- isoline build a border from isoline.
- distance build a the signed distance approximation to an isoline in 2d and 3d.
- freeyams, mesh met, mmg3d-v4, medit interface of library of P. Frey to adapt mesh in 3d.



Important Plugin with MPI

- HPDDM a new parallel linear solver see diffusion-2d.edp example in examples++-hpddm
- PETSc a new version of PETSc interface
- MUMPS a new version of MUMPS interface
- MPICG parallel version of CG, and GMRES
- mpi-cmaes parallel version of stochastic optimization algorithm.
- hips_FreeFem,parms_FreeFem,MUMPS_FreeFem old parallel linear solver interface.



1 Introduction

- History
- The main characteristics
- The Changes from sep. 2015 to dec. 2016
- Basement
- Weak form



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:

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

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

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

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



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  $\gamma_2$ 
plot(u,fill=1,wait=1,value=0,wait=1);
```

Run:fish.edp Run:fish3d.edp



- All boundary (internal or not) was defined through a label number and this number is defined in the mesh data structure. The support of this label number is an edge in 2d and a face in 3d and **so FreeFem++ never use label on vertices.**
- To define integration you can use the region (resp. label) number if you compute integrate in domain (resp. boundary). There is no way to compute 1d integral in 3d domain.
- Today there are no Finite Element defined on surface.
- You can put list of label or region in integer array (`int[int]`).



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



2 Tools

- Remarks on weak form and boundary conditions
- Mesh generation
- Build mesh from image
- 3d mesh
- Mesh tools
- Anisotropic Mesh adaptation



Remark on varf

The functions appearing in the variational form are formal and local to the varf definition, the only important think is the order in the parameter list, like in

```
varf vb1([u1,u2],[q]) = int2d(Th) ( (dy(u1)+dy(u2)) *q)
                                +int2d(Th) (1*q) + on(1,u1=2);
varf vb2([v1,v2],[p]) = int2d(Th) ( (dy(v1)+dy(v2)) *p)
                                +int2d(Th) (1*p) ;
```

To build matrix A from the bilinear part the the variational form a of type varf do simply

```
matrix B1 = vb1(Vh,Wh [, ...] );
matrix<complex> C1 = vb1(Vh,Wh [, ...] );
// where the fespace have the correct number of comp.
// Vh is "fespace" for the unknown fields with 2 comp.
// ex fespace Vh(Th,[P2,P2]); or fespace Vh(Th,RT);
// Wh is "fespace" for the test fields with 1 comp.
```

To build a vector, put $u1 = u2 = 0$ by setting 0 of on unknown part.

```
real[int] b = vb2(0,Wh);
complex[int] c = vb2(0,Wh);
```

Remark: In this case the mesh use to defined , \int, u, v can be different.



The boundary condition terms

First FreeFem++ use only the label number of edge (2d) or faces (3d).

- An "on" scalar form (for Dirichlet) : `on(1, u = g)`

The meaning is for all degree of freedom i (DoF) of this associated boundary, the diagonal term of the matrix $a_{ii} = tgv$ with the *terrible giant value* tgv ($=10^{30}$ by default) and the right hand side $b[i] = "(\Pi_h g)[i]" \times tgv$, where the " $(\Pi_h g)[i]$ " is the boundary DoF value given by the interpolation of g .

- An "on" vectorial form (for Dirichlet) : `on(1,u1=g1,u2=g2)` If you have vectorial finite element like RT0, the 2 components are coupled, and so you have :
 $b[i] = "(\Pi_h(g1,g2))[i]" \times tgv$, where Π_h is the vectorial finite element interpolant.
- a linear form on Γ (for Neumann in 2d)
`-int1d(Th) (f*w) or -int1d(Th, 3) (f*w)`
- a bilinear form on Γ or Γ_2 (for Robin in 2d)
`int1d(Th) (K*v*w) or int1d(Th, 2) (K*v*w) .`
- a linear form on Γ (for Neumann in 3d)
`-int2d(Th) (f*w) or -int2d(Th, 3) (f*w)`



2 Tools

- Remarks on weak form and boundary conditions
- **Mesh generation**
- Build mesh from image
- 3d mesh
- Mesh tools
- Anisotropic Mesh adaptation



Build Mesh 2d

First a 10×10 grid mesh of unit square $[0, 1]^2$

```
int[int] labs=[10,20,30,40];           // bot., right, top, left
mesh Th1 = square(10,10,label=labs,region=0,[x,y]);           //
plot(Th1,wait=1);
int[int] old2newlabs=[10,11, 30,31];           // 10 -> 11, 30 -> 31
Th1=change(Th1,label=old2newlabs);           //
// do Change in 2d or in 3d. region=a, fregion=f ,
// flabel=f
```

a L shape domain $[0, 1]^2 \setminus [\frac{1}{2}, 1]^2$

```
mesh Th = trunc(Th1, (x<0.5) | (y < 0.5),label=1);           //
plot(Th,cmm="Th");
mesh Thh = movemesh(Th, [-x,y]);
mesh Th3 = Th+movemesh(Th, [-x,y]);           // glumesh ...
plot(Th3,cmm="Th3");
```

Run:mesh1.edp



Build Mesh 2d, border

a Circle with or without an hole ;

Remark; by default the domain is a left of the border (if the number of segments positive).

```
border Co(t=0,2*pi) { x=cos(t); y=sin(t); label=1; }
border Ci(t=0,2*pi) { x=cos(t)/2; y=sin(t)/2; label=2; }
plot(Co(30)+Ci(15),wait=1);
mesh Thf=buildmesh(Co(30)+Ci(15));           //      without hole
                                                //      two region:
cout << " The two Region of Thf : " << Thf(0,0).region << " " <<
Thf(0,0.9).region << endl;
plot(Thf,wait=1);
mesh Thh=buildmesh(Co(30)+Ci(-15));           //      without hole
plot(T
```

Get an external mesh

```
mesh Th2("april-fish.msh");
build with emc2, bamg, modulef, etc...
```

Run:mesh-circles.edp



Build Mesh 2d, more complicate with implicit loop

a L shape domain $]0, 1[^2 \setminus [\frac{1}{2}, 1[^2$ with 6 multi-borders.

```
int nn=30; real dd=0.5;
real[int,int] XX=[[0,0],[1,0],[1,dd],[dd,dd],[dd,1],[0,1]];
int[int] NN=[nn,nn*dd,nn*(1-dd),nn*(1-dd),nn*dd,nn];
border bb(t=0,1;i)
{
    // i is the index of the multi border loop
    int ii = (i+1)%XX.n; real t1 = 1-t;
    x = XX(i,0)*t1 + XX(ii,0)*t;
    y = XX(i,1)*t1 + XX(ii,1)*t;
    label = 1; ;
}
plot(bb(NN),wait=1);
mesh Th=buildmesh(bb(NN));
plot(Th,wait=1);
Run:mesh-multi.edp
```



2 Tools

- Remarks on weak form and boundary conditions
- Mesh generation
- **Build mesh from image**
- 3d mesh
- Mesh tools
- Anisotropic Mesh adaptation



Build mesh from image 1/2

```
load "ppm2rnm" load "isoline" load "shell"
string lac="lac-oxford", lacjpg =lac+".jpg", lacpgm =lac+".pgm";
if(stat(lacpgm)<0) exec("convert "+lacjpg+" "+lacpgm);
real[int,int] Curves(3,1); int[int] be(1); int nc;
{
    real[int,int] ff1(lacpgm);                                // read image
    int nx = ff1.n, ny=ff1.m;           // grey value in 0 to 1 (dark)
    mesh Th=square(nx-1,ny-1,[(nx-1)*(x),(ny-1)*(1-y)]);
    fespace Vh(Th,P1); Vh f1; f1[] = ff1; // array to fe function.
    real iso =0.3;           // try some value to get correct iso
    real[int] viso=[iso];
    nc=isoline(Th,f1,iso=iso,close=0,Curves,beginend=be,
               smoothing=.1, ratio=0.5);
    for(int i=0; i<min(3,nc);++i)
    { int il=be(2*i),i2=be(2*i+1)-1;
        plot(f1,viso=viso,[Curves(0,il:i2),Curves(1,il:i2)],
              wait=1,cmm=i); } }
```



Build mesh from image 2/2

```
int[int] iii=[1,2];           // chose to componante ...
int[int] NC=[-300,-300];      // 2 componante
border G(t=0,1;i) {
    P=Curve(Curves,be(2*iii[i]),be(2*iii[i]+1)-1,t);
    label= iii[i];
}
plot(G(NC),wait=1);
mesh Th=buildmesh(G(NC));
plot(Th,wait=1);
real scale = sqrt(AreaLac/Th.area);
Th=movemesh(Th, [x*scale,y*scale]);
Run:lac.edp      (the Powai lac close to IIT Bombay )
```



2 Tools

- Remarks on weak form and boundary conditions
- Mesh generation
- Build mesh from image
- 3d mesh
- Mesh tools
- Anisotropic Mesh adaptation



A cube with cube or buildlayer (simple)

```
load "msh3" buildlayer
mesh3 Th;
int nn=10;
if(1) { int[int] ll=[1,1,1,1,1,2];           // front,right,back,left,down,up
         Th=cube(nn,nn,nn,label=ll); }
else
{ int[int] rup=[0,2],                      // label: upper face 0-> 2 (region -> label)
    rdown=[0,1],                         // label: lower face 0-> 1 (region -> label)
    rmid=[1,1 ,2,1 ,3,1 ,4,1 ],          // 4 Vert. 2d label -> 3d label
    rtet= [0,0];
    real zmin=0,zmax=1;
    Th=buildlayers(square(nn,nn,),nn,
                     zbound=[zmin,zmax],region=rtet,
                     labelmid=rmid,  labelup = rup,
                     labeledown = rdown)
}
Th= trunc(Th, ((x<0.5) | (y< 0.5) | (z<0.5)),label=3);           // remove 1/2 cube
plot("cube",Th);
Run:Cube.edp
```



3D layer mesh of a Lac with buildlayer

```
load "msh3"//      buildlayer
load "medit"//     medit
int nn=5;
border cc(t=0,2*pi){x=cos(t);y=sin(t);label=1;};
mesh Th2= buildmesh(cc(100));
fespace Vh2(Th2,P2);
Vh2 ux,uz,p2;
int[int] rup=[0,2], rdown=[0,1], rmid=[1,1];
func zmin= 2-sqrt(4-(x*x+y*y));    func zmax= 2-sqrt(3.);
//    we get nn*coef layers
mesh3 Th=buildlayers(Th2,nn,
                      coef= max((zmax-zmin)/zmax,1./nn),
                      zbound=[zmin,zmax],
                      labelmid=rmid, labelup = rup,
                      labeldown = rdown);           //    label def
medit("lac",Th);
Run:Lac.edp Run:3d-leman.edp
```



a 3d axi Mesh with buildlayer

```
func f=2*((.1+((x/3))*(x-1)*(x-1)/1+x/100))^(1/3.)-(.1)^(1/3.));
real yf=f(1.2,0);
border up(t=1.2,0.) { x=t; y=f; label=0; }
border axe2(t=0.2,1.15) { x=t; y=0; label=0; }
border hole(t=pi,0) { x= 0.15 + 0.05*cos(t); y= 0.05*sin(t);
    label=1; }
border axe1(t=0,0.1) { x=t; y=0; label=0; }
border queue(t=0,1) { x= 1.15 + 0.05*t; y = yf*t; label =0; }
int np= 100;
func bord= up(np)+axe1(np/10)+hole(np/10)+axe2(8*np/10)
    + queue(np/10);
plot( bord);                                     // plot the border ...
mesh Th2=buildmesh(bord);                      // the 2d mesh axi mesh
plot(Th2,wait=1);
int[int] l23=[0,0,1,1];
Th=buildlayers(Th2,coef= max(.15,y/max(f,0.05)), 50
, zbound=[0,2*pi],transfo=[x,y*cos(z),y*sin(z)]
, facemerge=1,labelmid=l23);
```

Run:3daximesh.edp



boundary mesh of a Sphere

```
load "tetgen"
mesh Th=square(10,20,[x*pi-pi/2,2*y*pi]);           // ]-pi/2,-pi/2[ x]0,2pi[
func f1 =cos(x)*cos(y); func f2 =cos(x)*sin(y); func f3 = sin(x);
               //      the partiel derivative of the parametrization DF
func f1x=sin(x)*cos(y);    func f1y=-cos(x)*sin(y);
func f2x=-sin(x)*sin(y);   func f2y=cos(x)*cos(y);
func f3x=cos(x);          func f3y=0;
                           //      M = DF^t DF
func m11=f1x^2+f2x^2+f3x^2;  func m21=f1x*f1y+f2x*f2y+f3x*f3y;
func m22=f1y^2+f2y^2+f3y^2;
func perio=[[4,y],[2,y],[1,x],[3,x]];
real hh=0.1/R; real vv= 1/square(hh);
Th=adaptmesh(Th,m11*vv,m21*vv,m22*vv,IsMetric=1,periodic=perio);
int[int] ref=[0,L]; //      the label of the Sphere to L ( 0 -> L)
mesh3 ThS= movemesh23(Th,transfo=[f1*R,f2*R,f3*R],orientation=1,
label=ref);
```

Run:Sphere.edp Run:sphere6.edp



Build 3d Mesh from boundary mesh

```
include "MeshSurface.idp"          // tool for 3d surfaces meshes
mesh3 Th;
try { Th=readmesh3("Th-hex-sph.mesh"); }           // try to read
catch(...) {           // catch a reading error so build the mesh...
    real hs = 0.2;           // mesh size on sphere
    int[int] NN=[11,9,10];
    real [int,int] BB=[[-1.1,1.1],[-.9,.9],[-1,1]]; // Mesh Box
    int [int,int] LL=[[1,2],[3,4],[5,6]];           // Label Box
    mesh3 ThHS = SurfaceHex(NN,BB,LL,1)+Sphere(0.5,hs,7,1);
                                              // surface meshes
    real voltet=(hs^3)/6.;           // volume mesh control.
    real[int] domaine = [0,0,0,1,voltet,0,0,0.7,2,voltet];
    Th = tetg(ThHS,switch="pqaAAYYQ",
              nbofregions=2,regionlist=domaine);
    savemesh(Th,"Th-hex-sph.mesh"); }           // save for next run
```



Build a bottle boundary mesh I/II

```
load "msh3" load "medit" load "gsl" load "tetgen"
...
mesh3 Ths; // the surface
{ mesh3 Th3c,Th3bottom, Th3top;
 { real[int,int] srneck=
 [ [Htube-0.001,Htube,Htube+Hneck*0.1, Htube+Hneck*0.3 , Htube+Hneck*0.7 ,
Htube+Hneck*0.9, Htube+Hneck+0.1],
 [Rext ,Rext ,Rext , Rext*.7+Rneck*.3 , Rext*.1 + Rneck*.9,
Rneck , Rneck ]];
gslspline rneck(gslinterpcpline,srneck); // Curve of neck of the bottle
}
// adap mesh of get pretty mesh
mesh Th2c= square(Hbot/hh,2*pi*Rext/hh,[x*Hbot,y*2*pi]);
fespace V2x(Th2c,P1);
func E1 = rneck(x)*cos(y); func E2 = rneck(x)*sin(y); func E3 = x;
...
Th2c=adaptmesh(Th2c,em11,em21,em22,IsMetric=1,periodic=perio,nbvx=100000); }
Th2c=change(Th2c,fregion=labcyl);
Th3c = movemesh23(Th2c,transfo=[E1 , E2 , E3]); // maillage exterieur
}
```



Build a bottle boundary mesh II/II

```
//      extraction of the border of the bottle
int[int] databoder(1);    int ncb= getborder(Th3c,databoder);
int ktop= Th3c(databoder[databoder[1]]).z < Th3c(databoder[databoder[0]]).z;
int kbot=1-ktop;           //      other border
...
macro DefBorder(bname,kk,Th3,bb,ll)
  int n#bname= bb[kk+1]-bb[kk];  border bname(t=bb[kk], bb[kk+1])
  {   real iv = int(t); if( iv == bb[kk+1]) iv = bb[kk];
      iv = bb[iv];x= Th3(iv).x ; y= Th3(iv).y ;label = ll;  }           //      EOM
DefBorder(btop,ktop,Th3c,databoder,1) DefBorder(bbot,kbot,Th3c,databoder,1)

Th3bottom=movemesh23(change(buildmesh(bbot (nbbot),fixeborder=1),fregion=labbottom)
                      ,transfo=[x,y,Zbot],orientation=-1);
Th3top=movemesh23(change(buildmesh(btop (-nbtop),fixeborder=1),fregion=labtop)
                     ,transfo=[x,y,Ztop],orientation=1);
Ths = Th3c + Th3bottom + Th3top; }

...
real[int] domaine = [0,0,Htube,1,hh^3/6.];
mesh3 Th=tetg(Ths,switch="pqaAYY",regionlist=domaine); medit ("Th",Th);
```

Run:bottle.edp



2 Tools

- Remarks on weak form and boundary conditions
- Mesh generation
- Build mesh from image
- 3d mesh
- **Mesh tools**
- Anisotropic Mesh adaptation



- change to change label and region numbering in 2d and 3d.
- movemesh checkmovemesh movemesh23 movemesh3
- triangulate (2d) , tetgconvexhull (3d) build mesh mesh for a set of point
- emptymesh (2d) built a empty mesh for Lagrange multiplier
- freeyams to optimize surface mesh
- mmg3d to optimize volume mesh with constant surface mesh
- mshmet to compute metric
- isoline to extract isoline (2d)
- trunc to remove peace of mesh and split all element (2d,3d)
- splitmesh to split 2d mesh in no regular way.



2 Tools

- Remarks on weak form and boundary conditions
- Mesh generation
- Build mesh from image
- 3d mesh
- Mesh tools
- Anisotropic Mesh adaptation



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

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

We introduce the metric $\mathcal{M}(x)$ as a field of $d \times d$ symmetric positive definite matrices, and the length ℓ of Γ 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} .

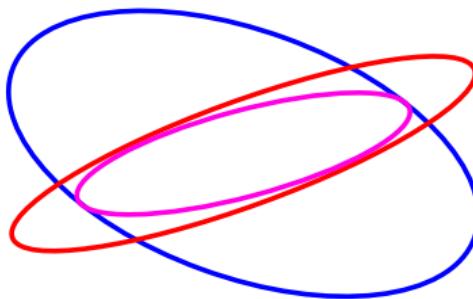


Metrix intersection

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:

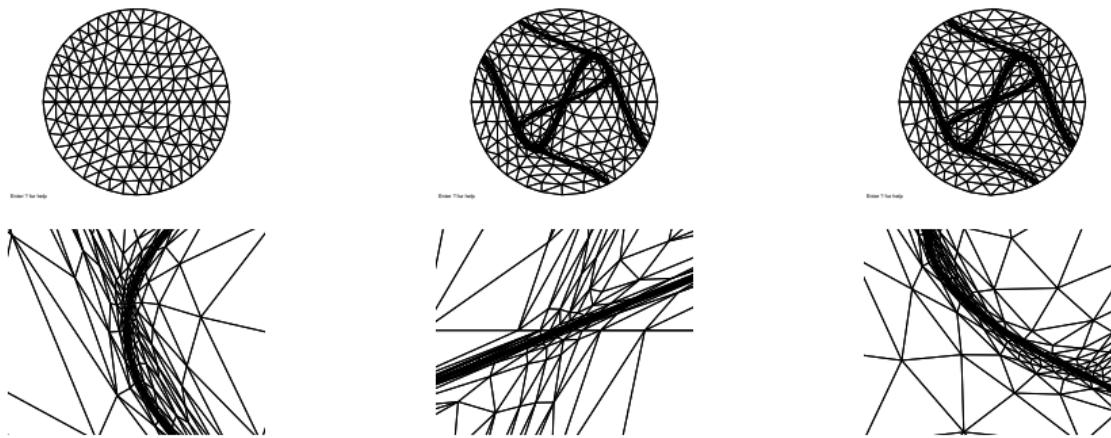
$$\mathcal{B}(\mathcal{M}_v) \subset \mathcal{B}(\mathcal{M}_u) \cap \mathcal{B}(\mathcal{M}_v)$$



Example of mesh

$$u = (10x^3 + y^3) + \tanh(500(\sin(5y) - 2x)));$$

$$v = (10y^3 + x^3) + \tanh(5000(\sin(5y) - 2*)));$$

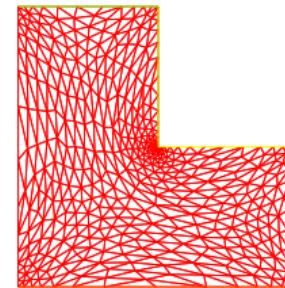
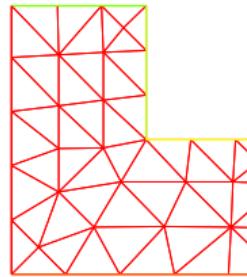


Run:Adapt-uv.edp

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

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

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



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}_ℓ for the P_ℓ Lagrange finite element, Optimal is given by (with only acute triangle) (thank J-M. Mirebeau)

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

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

$$D^{(3)}u(x,y) = (ax^3 + 3bx^2y + 3cxy^2 + dy^3)/3!,$$

Run:adapt.edp

Run:AdaptP3.edp



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



3 Academic Examples

- Laplace/Poisson
- 3d Poisson equation with mesh adaptation
- Linear elasticity equation
- Stokes equation
- Optimize Time depend schema



We find p in H^1 , such that:

$$-\Delta p = f \text{ in } \Omega, \quad \partial_n p = g_n \text{ on } \Gamma.$$

This problem is defined through a constant and mathematically the problem is well pose iff $\int_{\Omega} f + \int_{\Gamma} g_n = 0$ and p is in H^1/\mathbb{R} . So we can make a small regularization to remove the problem of constant by find $p_\varepsilon \in H^1$ such that

$$\varepsilon p_\varepsilon - \Delta p_\varepsilon = f \text{ in } \Omega, \quad \partial_n p_\varepsilon = g_n \text{ on } \Gamma$$

and the last problem is trivial to be approximate in FreeFem++:

```
Vh p,q; real eps=1e-8; // warning eps
                           // must be small but no too small.
solve Laplace(p,q)
= int2d(Th) ( p*q*eps + grad(p)' *grad(q) )
  - int2d(Th) ( f*q ) - int1d(Th) (gn*q) ;
```

Remark: it is hard to put Dirichlet boundary condition on only one point so set the constant due to label definition.



Laplace equation (mixte formulation) I/III

Now we solve $-\Delta p = f$ in Ω , $p = g_d$ on Γ_d , $\partial_n p = g_n$ on Γ_n .

Γ_d, Γ_n is a partition of $\partial\Omega$.

with $\vec{u} = \nabla p$ the problem becomes:

Find \vec{u}, p such that:

$$-\nabla \cdot \vec{u} = f, \quad \vec{u} - \nabla p = 0 \quad \text{in } \Omega, \quad p = g_d \quad \text{on } \Gamma_d, \quad \partial_n p = g_n \quad \text{on } \Gamma_n \quad (3)$$

Mixte variational formulation is: find $\vec{u} \in H_{div}(\Omega)$, $p \in L^2(\Omega)$, $\vec{u} \cdot n = g_n$ on Γ_n such that

$$\int_{\Omega} q \nabla \cdot \vec{u} + \int_{\Omega} p \nabla \cdot \vec{v} + \vec{u} \cdot \vec{v} = \int_{\Omega} -f q + \int_{\Gamma_d} g_d \vec{v} \cdot \vec{n}, \quad \forall (\vec{v}, q) \in H_{div} \times L^2, \text{ and } \vec{v} \cdot n = 0 \text{ on } \Gamma_n$$



Laplace equation (mixed formulation) II/III

frfr

```
mesh Th=square(10,10); fespace Vh(Th,RT0), Ph(Th,P0);
func gd = 1.; func gn = 1.; func f = 1.;
Vh [u1,u2],[v1,v2];
Ph p,q;
solve laplaceMixte([u1,u2,p],[v1,v2,q],solver=UMFPACK)
= int2d(Th)( p*q*0e-10 + u1*v1 + u2*v2
              + p*(dx(v1)+dy(v2)) + (dx(u1)+dy(u2))*q )
+ int2d(Th)( f*q)
- int1d(Th,1,2,3)( gd*(v1*N.x+v2*N.y)) // int on  $\Gamma_d$ 
+ on(4,u1=gn*N.x,u2=gn*N.y); // mean  $u.n = (g_n).n$ 
```

Run:LaplaceRT.edp



Laplace equation (Galerkin discontinuous formulation) III/III

```
solve  $-\Delta u = f$  on  $\Omega$  and  $u = g$  on  $\Gamma$ 
macro dn(u) (N.x*dx(u)+N.y*dy(u)) // def the normal derivative
mesh Th = square(10,10); // unite square
fespace Vh(Th,P2dc); // discontinuous P2 finite element
// if pena = 0 => Vh must be P2 otherwise penalization
real pena=0; // to add penalization
func f=1; func g=0;
Vh u,v;

problem A(u,v,solver=UMFPACK) = // 
    int2d(Th) (dx(u)*dx(v)+dy(u)*dy(v))
+ intalledges(Th) ( // loop on all edge of all triangle
    ( jump(v)*average(dn(u)) - jump(u)*average(dn(v))
        + pena*jump(u)*jump(v) ) / nTonEdge )
- int2d(Th) (f*v)
- int1d(Th) (g*dn(v) + pena*g*v) ;
A; // solve DG
```

Run:LapDG2.edp



A mathematical Poisson Problem with full Neumann BC. with 1D lagrange multiplier

The variationnal form is find $(u, \lambda) \in V_h \times \mathbb{R}$ such that

$$\forall (v, \mu) \in V_h \times \mathbb{R} \quad a(u, v) + b(u, \mu) + b(v, \lambda) = l(v), \quad \text{where } b(u, \mu) = \mu \int_{\Omega} u$$

```
mesh Th=square(10,10);      fespace Vh(Th,P1);      //      P1 FE space
int n = Vh.ndof, n1 = n+1; func f=1+x-y;
macro Grad(u) [dx(u),dy(u)]                                         // EOM
varf va(uh,vh) = int2d(Th)(Grad(uh)'*Grad(vh)) ;
varf vL(uh,vh) = int2d(Th)(f*vh) ;
varf vb(uh,vh)= int2d(Th)(1.*vh);
matrix A=va(Vh,Vh);
real[int] b=vL(0,Vh), B = vb(0,Vh);
real[int] bb(n1),x(n1),b1(1),l(1); b1=0;
matrix AA = [ [ A , B ] , [ B' , 0 ] ] ; bb = [ b, b1];
set(AA,solver=UMFPACK);          // set the type of linear solver.
x = AA^-1*bb;      [uh[],l] = x;           // solve the linear systeme
plot(uh,wait=1);                      // set the value
```

Run:Laplace-lagrange-mult.edp



3 Academic Examples

- Laplace/Poisson
- 3d Poisson equation with mesh adaptation
- Linear elasticity equation
- Stokes equation
- Optimize Time depend schema



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];
mesh3 Th3=buildlayers(square(nn,nn,region=0,label=l1111),
    nn, zbound=[0,1], labelmid=l11,labelup = 101,labeldown = 101); // label numbering
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



3 Academic Examples

- Laplace/Poisson
- 3d Poisson equation with mesh adaptation
- Linear elasticity equation
- Stokes equation
- Optimize Time depend schema



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 \mathbf{u} field such that:

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

Where $\varepsilon(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + {}^t \nabla \mathbf{u})$ and $\sigma(\mathbf{u}) = \mathbf{A}\varepsilon(\mathbf{u})$ with \mathbf{A} the linear positif operator on symmetric $d \times d$ matrix corresponding to the material propriety. Denote

$$V_g = \{\mathbf{v} \in H^1(\Omega)^d / \mathbf{v}|_{\Gamma_d} = \mathbf{g}\}.$$

The Basic displacement variational formulation is: find $\mathbf{u} \in V_0(\Omega)$, such that:

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



Linear elasticity equation, in FreeFem++

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

```
load "medit"    include "cube.idp"
int[int] Nxyz=[20,5,5];
real [int,int] Bxyz=[[0.,5.],[0.,1.],[0.,1.]];
int [int,int] Lxyz=[[1,2],[2,2],[2,2]];
mesh3 Th=Cube(Nxyz,Bxyz,Lxyz);
//      Alu ...
real rhoAlu = 2600, alu11= 1.11e11 , alu12 = 0.61e11 ;
real alu44= (alu11-alu12)*0.5;
func Aalu = [ [alu11, alu12,alu12, 0. ,0. ,0. ],
              [alu12, alu11,alu12, 0. ,0. ,0. ],
              [alu12, alu12,alu11, 0. ,0. ,0. ],
              [0. , 0. , 0. , alu44,0. ,0. ],
              [0. , 0. , 0. , 0. ,alu44,0. ],
              [0. , 0. , 0. , 0. ,0. ,alu44] ];
real gravity = -9.81;
```



Linear elasticity equation, in FreeFem++

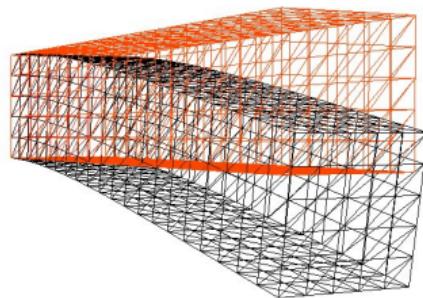
```
fespace Vh(Th, [P1,P1,P1]);
Vh [u1,u2,u3], [v1,v2,v3];
macro Strain(u1,u2,u3)
[ dx(u1), dy(u2), dz(u3), (dz(u2) +dy(u3)), (dz(u1)+dx(u3)), (dy(u1)+dx(u2)) ]
// EOM
solve Lame([u1,u2,u3],[v1,v2,v3])=
int3d(Th)( Strain(v1,v2,v3)'*(Aalu*Strain(u1,u2,u3)) )
- int3d(Th) ( rhoAlu*gravity*v3)
+ on(1,u1=0,u2=0,u3=0)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

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

out amplification= 3997.95

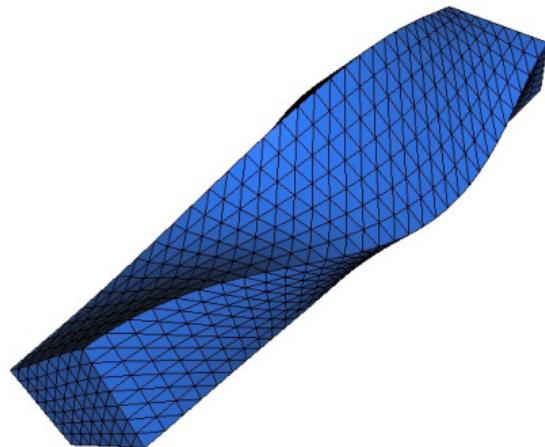


Run:beam-3d.edp

Run:beam-EV-3d.edp

Run:free-cyl-3d.edp

Run:beam-3d-Adapt.edp



3 Academic Examples

- Laplace/Poisson
- 3d Poisson equation with mesh adaptation
- Linear elasticity equation
- **Stokes equation**
- Optimize Time depend schema



Stokes equation

The Stokes equation is find a velocity field $\mathbf{u} = (u_1, \dots, u_d)$ and the pressure p on domain Ω of \mathbb{R}^d , such that

$$\begin{aligned}-\Delta \mathbf{u} + \nabla p &= 0 && \text{in } \Omega \\ \nabla \cdot \mathbf{u} &= 0 && \text{in } \Omega \\ \mathbf{u} &= \mathbf{u}_\Gamma && \text{on } \Gamma\end{aligned}$$

where \mathbf{u}_Γ is a given velocity on boundary Γ .

The classical variational formulation is: Find $\mathbf{u} \in H^1(\Omega)^d$ with $\mathbf{u}|_\Gamma = \mathbf{u}_\Gamma$, and $p \in L^2(\Omega)/\mathbb{R}$ such that

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

or now find $p \in L^2(\Omega)$ such than (with $\varepsilon = 10^{-10}$)

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



Stokes equation in FreeFem++

```
... build mesh .... Th (3d) T2d ( 2d)
fespace VVh(Th, [P2,P2,P2,P1]);           // Taylor Hood FE.
macro Grad(u) [dx(u),dy(u),dz(u)]        // EOM
macro div(u1,u2,u3) (dx(u1)+dy(u2)+dz(u3)) // EOM
VVh [u1,u2,u3,p],[v1,v2,v3,q];
solve vStokes([u1,u2,u3,p],[v1,v2,v3,q]) =
int3d(Th)(
    Grad(u1)'*Grad(v1)
    + Grad(u2)'*Grad(v2)
    + Grad(u3)'*Grad(v3)
    - div(u1,u2,u3)*q - div(v1,v2,v3)*p
    - 1e-10*q*p )
+ on(1,u1=0,u2=0,u3=0) + on(2,u1=1,u2=0,u3=0);
```

[Run:Stokes-2d.edp](#)

[Run:Stokes-bug.edp](#)

[Run:Stokes-Pipe.edp](#)

[Run:Stokes-UzawaCahouetChabart-bug.edp](#)

[Run:Stokes3d.edp](#)



3 Academic Examples

- Laplace/Poisson
- 3d Poisson equation with mesh adaptation
- Linear elasticity equation
- Stokes equation
- Optimize Time depend schema



Fast method for Time depend Problem / formulation

First, it is possible to define variational forms, and use this forms to build matrix and vector to make very fast script (4 times faster here).

For example solve the Thermal Conduction problem of section 3.4. We must solve the temperature equation in Ω in a time interval $(0, T)$.

$$\begin{aligned} \partial_t u - \nabla \cdot (\kappa \nabla u) &= 0 \text{ in } \Omega \times (0, T), \\ u(x, y, 0) &= u_0 + xu_1 \\ u &= 30 \text{ on } \Gamma_{24} \times (0, T), \quad \kappa \frac{\partial u}{\partial n} + \alpha(u - u_e) = 0 \text{ on } \Gamma \times (0, T). \end{aligned} \quad (6)$$

The variational formulation is in $L^2(0, T; H^1(\Omega))$; we shall seek u^n satisfying

$$\forall w \in V_0; \quad \int_{\Omega} \frac{u^n - u^{n-1}}{\delta t} w + \kappa \nabla u^n \nabla w + \int_{\Gamma} \alpha(u^n - u_{ue}) w = 0$$

where $V_0 = \{w \in H^1(\Omega) / w|_{\Gamma_{24}} = 0\}$.



Fast method for Time depend Problem algorithm

So the to code the method with the matrices $A = (A_{ij})$, $M = (M_{ij})$, and the vectors $u^n, b^n, b', b'', b_{cl}$ (notation if w is a vector then w_i is a component of the vector).

$$u^n = A^{-1}b^n, \quad b' = b_0 + Mu^{n-1}, \quad b'' = \frac{1}{\varepsilon} b_{cl}, \quad b_i^n = \begin{cases} b''_i & \text{if } i \in \Gamma_{24} \\ b'_i & \text{else} \end{cases}$$

Where with $\frac{1}{\varepsilon} = \text{tgv} = 10^{30}$:

$$A_{ij} = \begin{cases} \frac{1}{\varepsilon} & \text{if } i \in \Gamma_{24}, \text{and } j = i \\ \int_{\Omega} w_j w_i / dt + k(\nabla w_j \cdot \nabla w_i) + \int_{\Gamma_{13}} \alpha w_j w_i & \text{else} \end{cases}$$

$$M_{ij} = \begin{cases} \frac{1}{\varepsilon} & \text{if } i \in \Gamma_{24}, \text{and } j = i \\ \int_{\Omega} w_j w_i / dt & \text{else} \end{cases}$$

$$b_{0,i} = \int_{\Gamma_{13}} \alpha u_{ue} w_i$$

$$b_{cl} = u^0 \quad \text{the initial data}$$



Fast The Time depend Problem/ edp

...

```
Vh u0=fu0,u=u0;
```

Create three variational formulation, and build the matrices A, M .

```
varf vthermic (u,v)= int2d(Th) (u*v/dt  
+ k*(dx(u) * dx(v) + dy(u) * dy(v)))  
+ int1d(Th,1,3) (alpha*u*v) + on(2,4,u=1);
```

```
varf vthermic0(u,v) = int1d(Th,1,3) (alpha*ue*v);
```

```
varf vMass (u,v)= int2d(Th) ( u*v/dt) + on(2,4,u=1);
```

```
real tgv = 1e30;
```

```
matrix A= vthermic(Vh,Vh,tgv=tgv,solver=CG);
```

```
matrix M= vMass (Vh,Vh);
```



Fast The Time depend Problem/ edp

Now, to build the right hand size we need 4 vectors.

```
real[int] b0 = vthermic0(0,Vh); // constant part of RHS
real[int] bcn = vthermic(0,Vh); // tgv on Dirichlet part
// we have for the node i : i ∈ Γ24 ⇔ bcn[i] ≠ 0
real[int] bcl=tgv*u0[]; // the Dirichlet B.C. part
```

The Fast algorithm:

```
for(real t=0;t<T;t+=dt) {
    real[int] b = b0 ; // for the RHS
    b += M*u[]; // add the time dependent part
    b = bcn ? bcl : b; // do ∀i: b[i] = bcn[i] ? bcl[i] : b[i];
    u[] = A^-1*b; // Solve linear problem
    plot(u);
}
```

Run:Heat.edp



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



4 Numerics Tools

- Connectivity
- Input/Output
- Tricks
- Eigenvalue
- Optimization Tools
- MPI/Parallel



```
mesh Th=square(5,5);
fespace Wh(Th,P2);
cout << " nb of DoF      : " << Wh.ndof << endl;
cout << " nb of DoF / K : " << Wh.ndofK << endl;
int k= 2, kdf= Wh.ndofK ;;                                //      element 2
cout << " df of element " << k << ":" ;
for (int i=0;i<kdf;i++) cout << Wh(k,i) << " ";
cout << endl;
```

Remark on local numbering of Dof by element is

for each sub finite element Pk in [P2, P2, P1] get fist DoF on vertex, second Dof on edge (opposite to vertex), second on K.

Run:Mesh-info.edp



4 Numerics Tools

- Connectivity
- Input/Output
- Tricks
- Eigenvalue
- Optimization Tools
- MPI/Parallel



uses `cout`, `cin`, `endl`, `<<`, `>>`.

To write to (resp. read from) a file,

declare a new variable `ofstream ofile("filename");`

or

`ofstream ofile("filename", append);` (resp. `ifstream ifile("filename");`)

or

`ofstream ofile("filename", append|binary);` (resp. `ifstream ifile("filename", binary);`)

and use `ofile` (resp. `ofile`) as `cout` (resp. `cin`).

You can use pipe to transfer data to another code here (gnuplot), see `pipe.edp` example:

Run:`pipe.edp`

Run:`io.edp`



4 Numerics Tools

- Connectivity
- Input/Output
- Tricks
- Eigenvalue
- Optimization Tools
- MPI/Parallel



What is simple to do with freefem++ :

- Evaluate variational form with Boundary condition or not.
- Do interpolation
- Do linear algebra
- Solve sparse problem.



Freefem++ Trick: extract Dof list of border

? Question how the list Degree of Freedom (DoF) of border.

Idea Take a function negative function, and increasing on the border, and sort do a simultaneous sort this value and DoF numbering. and use the macro ExtractDofsonBorder(labs, Wh, dof labs, orient) defined in "ExtractDofsonBorder.idp",

Run:ListOfDofOnBorder.edp

Computation of error estimate $\eta_K = \sqrt{\int_K \text{blabla}} = \sqrt{\int_{\Omega} w_k \text{blabla}}$ where w_k is the basic function of fespace Ph(Th, P0).

```
varf vetaK(unused,wK) = int2d(Th)( blabla * wK);  
Ph etaK; etaK[] = vetaK(0,Ph); etaK=sqrt(etaK);
```



to Change Default sparse solver add following line:

```
load += "MUMPS_seq"
```

if MUMPS-seq is available in file \\${HOME}/.freefem++.pref

Diff How to compute, differential: use of macro

$$J(u) = \int_{\Omega} F(u); \quad \text{macro } F(u) = \sqrt{1 + \nabla u \cdot \nabla u}$$

$$dJ(u)(v) = \int_{\Omega} dF(u, v); \quad \text{macro } dF(u, v) = \frac{\nabla u \cdot \nabla v}{\sqrt{1 + \nabla u \cdot \nabla u}}$$

$$ddJ(u)(v, w) = \int_{\Omega} ddF(u, v, w);$$

$$\text{macro } ddF(u, v, w) = \frac{\nabla w \cdot \nabla v}{\sqrt{1 + \nabla u \cdot \nabla u}} - \frac{(\nabla u \cdot \nabla v)(\nabla w \cdot \nabla v)}{\sqrt{1 + \nabla u \cdot \nabla u}^3}$$



4 Numerics Tools

- Connectivity
- Input/Output
- Tricks
- Eigenvalue
- Optimization Tools
- MPI/Parallel



Eigenvalue/ Eigenvector example

The problem, Find the first λ, u_λ such that:

$$a(u_\lambda, v) = \int_{\Omega} \nabla u_\lambda \cdot \nabla v = \lambda \int_{\Omega} u_\lambda v = \lambda b(u_\lambda, v)$$

the boundary condition is made with exact penalization: we put $1e30 = tgv$ on the diagonal term of the lock degree of freedom. So take Dirichlet boundary condition only with a variational form and not on b variational form , because we compute eigenvalue of

$$\frac{1}{\lambda}v = A^{-1}Bv$$

Otherwise we get spurious mode.

Arpack interface:

```
int k=EigenValue(A,B,sym=true,value=ev,vector=eV);
```



Eigenvalue/ Eigenvector example code

```
...
fespace Vh(Th,P1);
macro Grad(u) [dx(u),dy(u),dz(u)] // EOM
varf a(u1,u2)= int3d(Th)( Grad(u1)'*Grad(u2) + on(1,u1=0) ;
varf b([u1],[u2]) = int3d(Th)( u1*u2 ) ; // no BC
matrix A= a(Vh,Vh,solver=UMFPACK),
B= b(Vh,Vh,solver=CG,eps=1e-20);

int nev=40; // number of computed eigenvalue close to 0
real[int] ev(nev); // to store nev eigenvalue
Vh[int] eV(nev); // to store nev eigenvector
int k=EigenValue(A,B,sym=true,value=ev,vector=eV);
k=min(k,nev);
for (int i=0;i<k;i++)
plot(eV[i],cmm="ev "+i+" v =" + ev[i],wait=1,value=1);
```

Run:Lap3dEigenValue.edp

Run:LapEigenValue.edp



4 Numerics Tools

- Connectivity
- Input/Output
- Tricks
- Eigenvalue
- Optimization Tools
- MPI/Parallel



Ipopt optimizer

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 :

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

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 $c_i^{\text{lb}} < c_i^{\text{ub}}$.



Ipopt Data, next

```
func real J(real[int] &X) {...}                                // Fitness Function,  
func real[int] gradJ(real[int] &X) {...}                          // Gradient  
  
func real[int] C(real[int] &X) {...}                                // Constraints  
func matrix jacC(real[int] &X) {...}                            // Constraints jacobian  
  
matrix jacCBuffer;                                              // just declare, no need to define yet  
func matrix jacC(real[int] &X)  
{  
    ...                                                               // fill jacCBuffer  
    return jacCBuffer;  
}
```

The hessian returning function is somewhat different because it has to be the hessian of the lagrangian function : $(x, \sigma_f, \lambda) \mapsto \sigma_f \nabla^2 f(x) + \sum_{i=1}^m \lambda_i \nabla^2 c_i(x)$ where $\lambda \in \mathbb{R}^m$ and $\sigma \in \mathbb{R}$. Your hessian function should then have

the following prototype :

```
matrix hessianLBuffer;                                         // just to keep it in mind  
func matrix hessianL(real[int] &X, real sigma, real[int] &lambda) {...}
```



Ipopt Call

```
real[int] Xi = ... ;                                // starting point
IPOPT(J,gradJ,hessianL,C,jacC,Xi, ... );

IPOPT(J,gradJ,C,jacC,Xi,...);                      // IPOPT with BFGS
IPOPT(J,gradJ,hessianJ,Xi,...);                    // Newton IPOPT
                                                       // without constraints
IPOPT(J,gradJ,Xi, ... );                           // BFGS, no constraints
IPOPT(J,gradJ,Xi, ... );                           // BFGS, no constraints
IPOPT([b,A],CC,ui1[],lb=lb1[],clb=cl[]...);     // affine case
...
...
```



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=lmi;
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



```
load "ff-NLopt"
...
if(kas==1)
    mini = nloptAUGLAG(J,start,grad=dJ,lb=lo,ub=up,IConst=IneqC,
                         gradIConst=dIneqC,subOpt="LBFGS",stopMaxFEval=10000,
                         stopAbsFTol=starttol);
else if(kas==2)
    mini = nloptMMA(J,start,grad=dJ,lb=lo,ub=up,stopMaxFEval=10000,
                      stopAbsFTol=starttol);
else if(kas==3)
    mini = nloptAUGLAG(J,start,grad=dJ,IConst=IneqC,gradIConst=dIneqC,
                         EConst=BC,gradEConst=dBC,
                         subOpt="LBFGS",stopMaxFEval=200,stopRelXTol=1e-2);
else if(kas==4)
    mini = nloptSLSQP(J,start,grad=dJ,IConst=IneqC,gradIConst=dIneqC,
                        EConst=BC,gradEConst=dBC,
                        stopMaxFEval=10000,stopAbsFTol=starttol);
```

Run:VarIneq2.edp



This algorithm works with a normal multivariate distribution in the parameters space and try to adapt its covariance matrix using the information provides by the successive function evaluations. Syntaxe: `cmaes (J, u[], ...)` ()

From <http://www.lri.fr/~hansen/javadoc/fr/inria/optimization/cmaes/package-summary.html>



Stochastic Exemple

```
load "ff-cmaes"

real mini = cmaes(J,start,stopMaxFunEval=10000*(al+1),
                   stopTolX=1.e-4/(10*(al+1)),
                   initialStdDev=(0.025/(pow(100.,al))));

SSPToFEF(best1[],best2[],start);
```

Run:cmaes-VarIneq.edp

```
load "mpi-cmaes"

real mini = cmaesMPI(J,start,stopMaxFunEval=10000*(al+1),
                      stopTolX=1.e-4/(10*(al+1)),
                      initialStdDev=(0.025/(pow(100.,al))));

SSPToFEF(best1[],best2[],start);
```

remark, the FreeFem `mpicommworld` is used by default. The user can specify his own MPI communicator with the named parameter "`comm=`", see the MPI section of this manual for more informations about communicators in FreeFem++.



4 Numerics Tools

- Connectivity
- Input/Output
- Tricks
- Eigenvalue
- Optimization Tools
- MPI/Parallel



Full MPI interface I/III

```
int[int] proc1=[1,2,3],proc2=[0,4];
mpiGroup grp(procs);           //      set MPI_Group to proc 1,2,3 in MPI_COMM_WORLD
mpiGroup grp1(comm,proc1);     //      set MPI_Group to proc 1,2,3 in comm
mpiGroup grp2(grp,proc2);     //      set MPI_Group to grp union proc1

mpiComm comm=mpiCommWorld;        //      set a MPI_Comm to MPI_COMM_WORLD
mpiComm ncomm(mpiCommWorld,grp); //      set the MPI_Comm form grp
                                  //      MPI_COMM_WORLD
mpiComm ncomm(comm,color,key);   //      MPI_Comm_split(MPI_Comm comm,
                                  //      int color, int key, MPI_Comm *ncomm)
mpiComm nicomm(processor(local_comm,local_leader),
                processor(peer_comm,peer_leader),tag);
//      build MPI_INTERCOMM_CREATE(local_comm, local_leader, peer_comm,
//                                  //      remote_leader, tag, &ncomm)
mpiComm ncomm(intercomm,hight) ; //      build using
                                  //      MPI_Intercomm_merge( intercomm, high, &ncomm)
mpiRequest rq;                  //      defined an MPI_Request
mpiRequest[int] arq(10);        //      defined an array of 10 MPI_Request

mpiSize(comm) ;                 //      return the size of comm (int)
mpiRank(comm) ;                 //      return the rank in comm (int)
```



Full MPI interface II/III

```
processor(i)           //      return processor i with no Request in MPI_COMM_WORLD
processor(MPIAnySource)          //      return processor any source
                                //      with no Request in MPI_COMM_WORLD
processor(i,comm)           //      return processor i with no Request in comm
processor(comm,i)           //      return processor i with no Request in comm
processor(i,rq,comm)         //      return processor i with Request rq in comm
processor(i,rq)             //      return processor i with Request rq in
                                //      MPI_COMM_WORLD
processorblock(i)           //      return processor i in MPI_COMM_WORLD
                            //      in block mode for synchronously communication
processorblock(MPIAnySource)          //      return processor any source
                                //      in MPI_COMM_WORLD in block mode for synchronously communication
processorblock(i,comm)         //      return processor i in comm in block mode

int status;                  //      to get the MPI status of send / recv
processor(10) « a « b;       //      send a,b asynchronously to the process 1,
processor(10) » a » b;       //      receive a,b synchronously from the process 10,
broadcast(processor(10,comm),a); //      broadcast from processor
                                //      of com to other comm processor
status=Send( processor(10,comm) , a); //      send synchronously
                                //      to the process 10 the data a
status=Recv( processor(10,comm) , a); //      receive synchronously
                                //      from the process 10 the data a;
```



Full MPI interface II/III

```
status=Isend( processor(10,comm) , a);           // send asynchronously to
                                                // the process 10 , the data a without request
status=Isend( processor(10,rq,comm) , a);        // send asynchronously to to
                                                // the process 10, the data a with request
status=Irecv( processor(10,rq) , a) ;            // receive synchronously from
broadcast(processor(comm,a));                   // Broadcast to all process of comm

mpiBarrier(comm) ;                            // do a MPI_Barrier on communicator comm,
mpiWait(rq);                                // wait on of Request,
mpiWaitAll(arq);                            // wait add of Request array,
mpiWtime() ;                                 // return MPIWtime in second (real),
mpiWTick() ;                                // return MPIWTick in second (real),
mpiAlltoall(a,b[,comm]) ;
mpiAllgather(a,b[,comm]) ;
mpiGather(a,b,processor(..) ) ;
mpiScatter(a,b,processor(..)) ;
mpiReduce(a,b,processor(..),mpiMAX) ;
mpiAllReduce(a,b,comm, mpiMAX) ;
mpiReduceScatter(a,b,comm, mpiMAX) ;
....
```



A first way to break complexity

- ① Build matrix in parallel by assembling per 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



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



5 Schwarz method with overlap

- Poisson equation with Schwarz method
- Transfer Part
- parallel GMRES
- A simple Coarse grid solver
- Numerical experiment
- HPDDM / see talk of Frédéric Nataf



To solve the following Poisson problem on domain Ω with boundary Γ 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 Ω , 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 Ω_i the sub domain which is the support of π_i function and also denote Γ_i the boundary of Ω_i .

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

$$\forall i = 1.., N_p : \quad -\Delta u_i^\ell = f, \text{ in } \Omega_i, \quad \text{and } u_i^\ell = u^\ell \text{ on } \Gamma_i \quad (7)$$

$$u^{\ell+1} = \sum_{i=1}^{N_p} \pi_i u_i^\ell \quad (8)$$



Some Remark

We never use finite element space associated to the full domain Ω because it is 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

$$(u^{\ell+1})_{|\Omega_i} = \sum_{j \in J_i} (\pi_j u_j^\ell)_{|\Omega_i} \quad (9)$$

We denote $u_h^\ell|_i$ the restriction of u_h^ℓ on V_{hi} , so the discrete problem on Ω_i of problem (7) is find $u_{hi}^\ell \in V_{hi}$ such that:

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

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

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



5 Schwarz method with overlap

- Poisson equation with Schwarz method
- Transfer Part
- parallel GMRES
- A simple Coarse grid solver
- Numerical experiment
- HPDDM / see talk of Frédéric Nataf



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;++)
    Irecv(processor(jpart[j],comm,rq[j]), Ri[j][]);
  for (int j=0;j<n;++)
    Isend(processor(jpart[j],comm,rq[j+n]), Si[j][]);
  for (int j=0;j<n*2;++)
    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; }
```



5 Schwarz method with overlap

- Poisson equation with Schwarz method
- Transfer Part
- parallel GMRES
- A simple Coarse grid solver
- Numerical experiment
- HPDDM / see talk of Frédéric Nataf



parallel GMRES

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 Ω_i .

```
func real[int] Si(real[int]& U) {  
    real[int] V(U.n); b= onG .* U;  
    b = onG ? b : Bi;  
    V = Ai^-1*b; // (7)  
    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; }
```



5 Schwarz method with overlap

- Poisson equation with Schwarz method
- Transfer Part
- parallel GMRES
- A simple Coarse grid solver
- Numerical experiment
- HPDDM / see talk of Frédéric Nataf



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
}
```

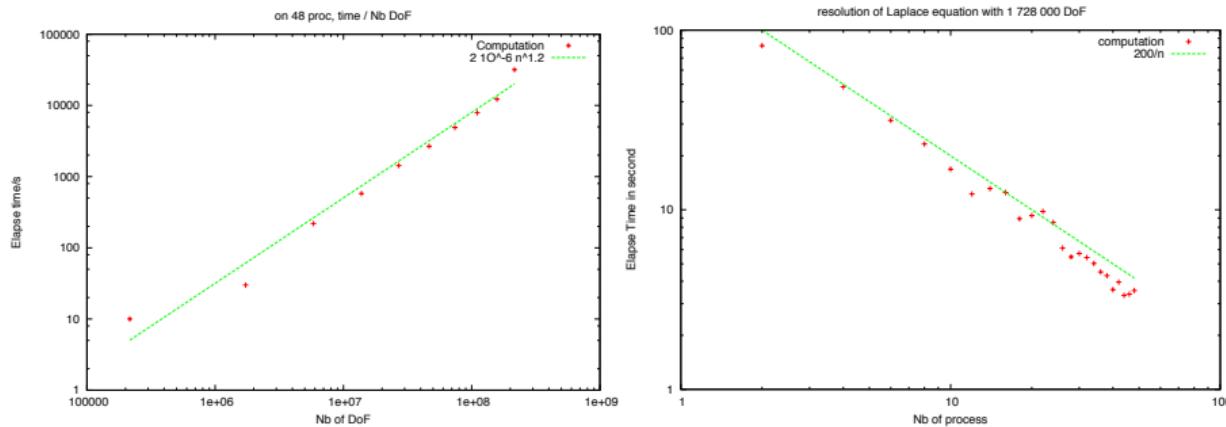
Limitation: if the initial problem, data have oscillation, you must use homogenization technic on coarse problem, or use the F. Nataf and co, preconditionner.



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:



5 Schwarz method with overlap

- Poisson equation with Schwarz method
- Transfer Part
- parallel GMRES
- A simple Coarse grid solver
- Numerical experiment
- HPDDM / see talk of Frédéric Nataf



A Parallel Numerical experiment on laptop

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

$$-\Delta u = 1, \text{in } \Omega; \quad u = 0, \text{ on } \partial\Omega. \quad (10)$$

With a cartesian meshes \mathcal{T}_{hn} of Ω 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



5 Schwarz method with overlap

- Poisson equation with Schwarz method
- Transfer Part
- parallel GMRES
- A simple Coarse grid solver
- Numerical experiment
- HPDDM / see talk of Frédéric Nataf



Run:diffusion-3d.edp

Run:diffusion-3d-PETSc.edp

Run:elasticity-3d.edp

Run:elasticity-3d-PETSc.edp

Run:Stokes-3d.edp

Run:Stokes-3d-PETSc.edp

Run:maxwell-3d.edp



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



6 Exercices

- An exercice: Oven problem
- An exercice: Min surface problem
- Heat equation with thermic resistance
- Benchmark: Navier-Stokes

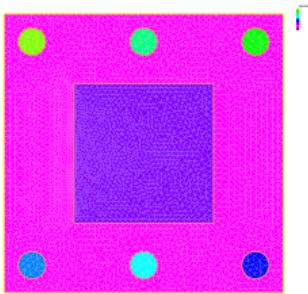


An exercice: Oven problem

Find the power on the 6 resistors of an oven such that the temperature is close as possible to a given temperature in the region 6.

The equation are the stationary Head equation in 2d with classical Fourier boundary condition. the mesh of the domain :

let call the u_p the solution of



"oven.msh"

Run:
mesh-
oven.edp

$$\begin{aligned}-\nabla \cdot K \nabla u_p &= \sum_{i=0}^5 p_i * \chi_i \text{ in } \Omega \\ u + K \nabla u_p \cdot n &= 0 \text{ on } \Gamma = \partial\Omega\end{aligned}$$

where χ_i is the characteristics function of the resistance i , $K = 10$ in region 6, $K = 1$ over where.

The problem is find the array p such that

$$p = \operatorname{argmin} \int_{\Omega_6} (u_p - 100)^2 dx$$

Some remark

build the mesh with multi border trick.

```
Xh[int] ur(6); // to store the 6 FE. functions Xh  
FreeFem++ as only linear solver on sparse matrix by default, but in the lapack  
plugin you have access to full matrix solver (see examples++-load/lapack.edp )  
so a way to solve a full matrix problem is for example :
```

```
real[int,int] AP(6,6); // a full matrix  
real[int] B(6),PR(6); // to array (vector of size 6)
```

... bla bla to compute AP and B

```
matrix A=AP; // full matrix to sparse of or use of  
lapack  
set(A,solver=CG); // set linear solver to the C.G.  
PR=A^-1*B; // solve the linear system.
```

The file name of the mesh is oven.msh, and the region numbers are 0 to 5 for the resistor, 6 for Ω_6 and 7 for the rest of Ω and the label of Γ is 1.



My solution, build the 6 basics function u_{e_i}

```
int nbresistor=6;           mesh Th("oven.msh");
real[int] pr(nbresitor+2), K(nbresitor+2);
K=1;           K[regi]=10;                                // def K
int regi=nbresitor, rege=nbresitor+1, lext=1;

macro Grad(u) [dx(u),dy(u)]                                // EOM
fespace Xh(Th,P2);   Xh u,v; int iter=0;
problem Chaleur(u,v,init=iter)
= int2d(Th) ( Grad(u)'*Grad(v)* K[region]) +
int1d(Th,lext) (u*v)
+ int2d(Th) (pr[region]*v) ;

Xh[int] ur(nbresitor);                                     // to store the 6  $u_{e_i}$ 
for(iter=0;iter<nbresitor;++iter)
{ pr=0;pr[iter]=1;
Chaleur;
ur[iter][]=u[];
plot(ur[iter],fill=1,wait=1); }
```



Computation of the optimal value

```
real[int,int] AP(nbresistor,nbresistor);
real[int] B(nbresistor),PR(nbresistor);

Xh    ui = 100;
for(int i=0;i<nbresistor;++i)
{
    B[i]=int2d(Th,regi)(ur[i]*ui);
    for(int j=0;j<6;++j)
        AP(i,j)= int2d(Th,regi)(ur[i]*ur[j]);
}
matrix A=AP; set(A,solver=UMFPACK);
PR=A^-1*B;
cout << " P R = " << PR << endl;
u[] =0;
for (int i=0;i<nbresistor;++i)
    u[] += PR[i]*ur[i][];
```

Run:oven-cimpa.edp



6 Exercices

- An exercice: Oven problem
- An exercice: Min surface problem
- Heat equation with thermic resistance
- Benchmark: Navier-Stokes



An exercice: Min surface problem

The geometrical problem: Find a function $u : C^1(\Omega) \mapsto \mathbb{R}$ where u is given on $\Gamma = \partial\Omega$, (e.i. $u|_{\Gamma} = g$) such that the area of the surface S parametrize by $(x, y) \in \Omega \mapsto (x, y, u(x, y))$ is minimal.

So the problem is $\arg \min J(u)$ where

$$\arg \min J(u) = \int_{\Omega} \left\| \begin{pmatrix} 1 \\ 0 \\ \partial_x u \\ \partial_y u \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\| d\Omega = \int_{\Omega} \sqrt{1 + (\partial_x u)^2 + (\partial_y u)^2} d\Omega$$

So the Euler-Lagrange equation associated to the minimization is:

$$\forall v / v|_{\Gamma} = 0 \quad : \quad DJ(u)v = - \int_{\Omega} \frac{(\partial_x v \partial_x u + \partial_y v \partial_y u)}{\sqrt{1 + (\partial_x u)^2 + (\partial_y u)^2}} d\Omega = 0$$

So find the solution for $\Omega =]0, \pi[^2$ and $g(x, y) = \cos(2 * x) * \cos(2 * y)$. by using the Non Linear Conjugate gradient NLCG like in the example: algo.edp in examples++-tutorial, or IPOPT interface.



Example of use of NLCG function:

```

func real J(real[int] & xx)           // the functional to minimized
{   real s=0;
    ...
    return s; }
func real[int] DJ(real[int] &xx)        // the grad of functional
{ ....
    return xx; };                      // return of an existing variable ok
...
NLCG(DJ,x,eps=1.e-6,nbiter=20,precon=matId);

```

Two useful operators on array real[int]

```

real[int] a(10),b(10);
...
a = b ? 1. : 0;                     // a[i] = 1 if b[i] else a[i]=0.  ∀i

```

To see the 3D plot of the surface

```
plot(u,dim=3);
```

My solution First the functional

```
func g=cos(2*x)*cos(2*y); // valeur au bord
mesh Th=square(20,20,[x*pi,y*pi]); // mesh definition of Ω
fespace Vh(Th,P1);

func real J(real[int] & xx) // the functionnal to minimise
{ Vh u;u[]={xx}; // to set FE.function u from xx array
  return int2d(Th) ( sqrt(1 +dx(u)*dx(u) + dy(u)*dy(u)) );
}

func real[int] dJ(real[int] & xx) // the grad of the J
{ Vh u;u[]={xx}; // to set FE. function u from xx array
  varf au(uh,vh) = int2d(Th) ( ( dx(u)*dx(vh) + dy(u)*dy(vh) )
    / sqrt(1. +dx(u)*dx(u) + dy(u)*dy(u)) )
    + on(1,2,3,4,uh=0);
  return xx= au(0,Vh); } // warning no return of local array
```



Solution 1:

```
Vh u=G;  
verbosity=5; // to see the residual  
int conv=NLCG(dJ,u[],nbiter=500,eps=1e-5);  
cout << " the surface =" << J(u[]) << endl;  
// so see the surface un 3D  
plot(u,dim=3);
```

Run:minimal-surf.edp



6 Exercices

- An exercice: Oven problem
- An exercice: Min surface problem
- Heat equation with thermic resistance
- Benchmark: Navier-Stokes



Heat equation with thermic resistance

let Ω be a domain cut with internal boundary Γ_0 in 2 sub-domain $\Omega_i, (i = 1, 2)$
We have Heat equation (Poisson) on Ω , but on Γ_0 we have a jump $[u]$ on the
temperature u proportional to the temperature flux which is continuous

So the equation to solve is:

Find u such that $u|_{\Omega_i} \in H(\Omega_i)$ for $i = 1, 2$ and

$$-\nabla \kappa \nabla u = f_i, \quad \text{in } \Omega_i$$

$$\alpha[u] - \kappa \nabla u \cdot n = 0, \quad [\kappa \nabla u \cdot n] = 0, \quad \text{on } \Gamma_0$$

+ external boundary condition on $\partial\Omega$.

For the test take:

$L = 3$, $\Omega =]-L, L[\times]0, 1[$, $\Gamma_0 = \{\sin(\pi y)/5, y \in [0, 1]\}$, take $\kappa = i$ in Ω_i .

The external boundary condition on $\partial\Omega$ are: $\kappa \nabla u \cdot n = 0$ on upper and lower boundary,
 $u = 0$ at the left part, $u = 1$ at the right part.



Heat equation with thermic resistance

Method 1: Solve 2 coupled problems and use the block matrix tools to defined the linear system of the problem.

Method 2: We suppose the Γ_0 move with time, and Γ_0 is not discretize in the mesh.



6 Exercices

- An exercice: Oven problem
- An exercice: Min surface problem
- Heat equation with thermic resistance
- Benchmark: Navier-Stokes



An exercice: Navier-Stokes

Try to make the 2d benchmark of :

<http://www.mathematik.tu-dortmund.de/lsiii/cms/papers/SchaeferTurek1996.pdf>

The mesh can be set:

```
int n=15;                                     // parameter ...
real D=0.1, H=0.41;
real cx0 = 0.2, cy0 = 0.2;                     // center of cyl.
real xa = 0.15, ya=0.2, xe = 0.25,ye =0.2;    // point for
pressure..
```



```
border fr1(t=0,2.2){x=t; y=0; label=1;}
border fr2(t=0,H){x=2.2; y=t; label=2;}
border fr3(t=2.2,0){x=t; y=H; label=1;}
border fr4(t=H,0){x=0; y=t; label=1;}
border fr5(t=2*pi,0){x=cx0+D*sin(t)/2; y=cy0+D*cos(t)/2; label=3;}
```



```
mesh Th=buildmesh(fr1(5*n)+fr2(n)+fr3(5*n)+fr4(n)+fr5(-n*3));
plot(Th, wait=1);
```



Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



Newton methods

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

- ① u^0 a initial guest
- ② do
 - ① find w^n solution of $DF(u^n)w^n = F(u^n)$
 - ② $u^{n+1} = u^n - w^n$
 - ③ if($\|w^n\| < \varepsilon$) break;

The Optimize Newton Method if $F = C + L + N$, where C is the constant part, L is Linear part and N is Non linear part of F . we have $DF = L + DN$ and $DF(u^n)u^{n+1} = DF(u^n)u^n - F(u^n) = DN(u^n)u^n - N(u^n) - C$. So the change in algorithm are:

- ② find u^{n+1} solution of $DF(u^n)u^{n+1} = DN(u^n)u^n - N(u^n) - C$
- ③ if($\|u^{n+1} - u^n\| < \varepsilon$) break;



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



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

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

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

Run:cavityNewton.edp

Run:NSNewtonCyl-100-mpi.edp



$$\frac{\partial u}{\partial t} + (u \cdot \nabla) u - \nu \Delta u + \nabla p = 0, \quad \nabla \cdot u = 0$$

with the same boundary conditions and with initial conditions $u = 0$.

This is implemented by using the interpolation operator for the term $\frac{\partial u}{\partial t} + (u \cdot \nabla) u$, giving a discretization in time

$$\begin{aligned} \frac{1}{\tau}(u^{n+1} - u^n \circ X^n) - \nu \Delta u^{n+1} + \nabla p^{n+1} &= 0, \\ \nabla \cdot u^{n+1} &= 0 \end{aligned} \tag{11}$$

The term $X^n(x) \approx x - \tau u^n(x)$ will be computed by the interpolation operator or convect operator.

Or better we use an order 2 schema, BDF1

$$\frac{\partial u}{\partial t} + (u \cdot \nabla) u \approx \frac{(3u^{n+1} - 4u^n \circ X_1^n + u^{n-1} \circ X_2^n)}{2\tau}$$

with $u^* = 2u^n - u^{n-1}$, and $X_1^n(x) \approx x - \tau u^*(x)$, $X_2^n(x) \approx x - 2\tau u^*(x)$

Run: NSCarCyl-100-mpi.edp



The ff++ NSI 3d code

```
real alpha =1./dt;
varf vNS([uu1,uu2,uu3,p],[v1,v2,v3,q]) =
int3d(Th)( alpha*(uu1*v1+uu2*v2+uu3*v3)
+ nu*(Grad(uu1)'*Grad(v1)+Grad(uu2)'*Grad(v2) +Grad(uu3)'*Grad(v3))
- div(uu1,uu2,uu3)*q - div(v1,v2,v3)*p + 1e-10*q*p )
+ on(1,2,3,4,5,uu1=0,uu2=0,uu3=0)
+ on(6,uu1=4*(1-x)*(x)*(y)*(1-y),uu2=0,uu3=0)
+ int3d(Th)( alpha*
    u1(X1,X2,X3)*v1 + u2(X1,X2,X3)*v2 + u3(X1,X2,X3)*v3 ) );
A = vNS(VVh,VVh);    set(A,solver=UMFPACK);           // build and factorize matrix
real t=0;
for(int i=0;i<50;++i)
{ t += dt;   X1[] = XYZ[] - u1[] * dt;                  // set  $\chi = [X1, X2, X3]$  vector
  b=vNS(0,VVh);                                         // build NS rhs
  u1[] = A^-1 * b;                                       // solve the linear system
  ux= u1(x,0.5,y);  uz= u3(x,0.5,y);  p2= p(x,0.5,y);
  plot([ux,uz],p2,cmm=" cut y = 0.5, time =" +t,wait=0); }
```

Run: NSI3d.edp



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



Variational Inequality

To solve just make a change of variable $u = u^+ - u^-$, $u > 0$ and $v = u^+ + u^-$, and we get a classical VI problem on u and the Poisson on v .

So we can use the algorithm of Primal-Dual Active set strategy as a semi smooth Newton Method HinterMuller , K. Ito, K. Kunisch SIAM J. Optim. V 13, I 3, 2002.

In this case , we just do all implementation by hand in FreeFem++ language

Run:VI-2-membrane-adap.edp



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- **Ground water**
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



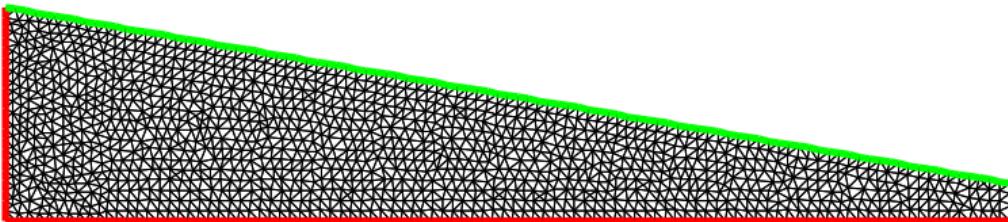
A Free Boundary problem , (phreatic water)

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 Γa
border b(t=0,h1){x=L;y=t;label=2;}; // the source Γb
border f(t=L,0){x=t;y=t*(h1-h)/L+h;label=3;}; // Γf
border d(t=h,0){x=0;y=t;label=4;}; // Left impermeable Γd
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:

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

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.* Run: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)

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

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 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
}
Run:freeboundary.edp
```



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



Bose Einstein Condensate

Just a direct use of Ipopt interface (2day of works)

The problem is find a complex field u on domain \mathcal{D} such that:

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

to code that in FreeFem++

use

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

Run:BEC.edp



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- **Hyper elasticity equation**
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



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

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

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

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

The problem is find

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



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

Run:ElasticLaw2d.edp

Run:CiarletGemona.edp



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



A true industrial numerical problem (2d) :

- ① 3 EDP: Dielectric, Elasticity, Piezoelectric , Linear, harmonic approximation
- ② α -periodic B.C. (New simple idea)
- ③ semi-infinite Dielectric domain (classical Fourier/Floquet transforme)
- ④ semi-infinite Piezoelectric domain (Hard)

In 9 month, we build with P. Ventura (100%, me 10%) a numerical simulator form scratch (8 months for the validation), The only thing to add to freefem++ is a interface with lapack to compute eigenvector of full 8×8 matrix.

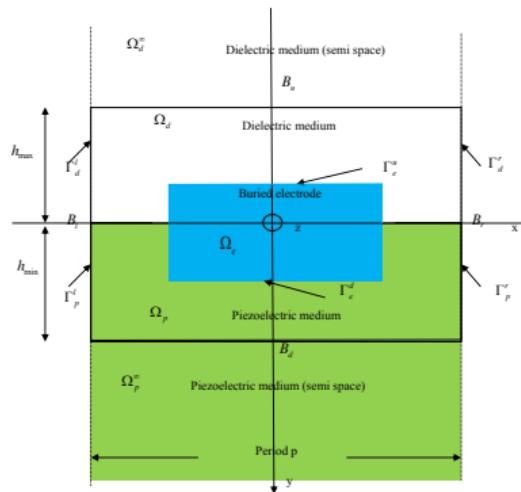
A good message : [Les calculs des paramètres physiques des transducteurs dans la bande d'arrêt et les évaluations de capacité statiques sont très satisfaisants par rapport aux résultats expérimentaux !](#)



The physic

In the dielectric medium Ω_d ,

$$\mathbf{D} = \epsilon_d \mathbf{E} \quad (12)$$



In the elastic medium Ω_e

$$\mathbf{T} = \mathbf{C}_e : \mathbf{S} \quad (13)$$

With \mathbf{C}_e is the elastic tensor for the elastic metallic domain. In the piezo-electric domain

$$\left\{ \begin{array}{l} \mathbf{T} = \mathbf{C}_p^E : \mathbf{S} - e \mathbf{E} \\ \mathbf{D} = e^T \mathbf{S} + \epsilon \mathbf{S} \mathbf{E} \end{array} \right. \quad (14)$$

The material domain Ω_m obeys Newton's second law:

$$\nabla \cdot \mathbf{T} = \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} \quad (15)$$

The quasi static Maxwell's equation is assumed for the whole domain Ω :

$$\nabla \cdot \mathbf{D} = 0 \quad (16)$$

By using the divergence relationship and the Green's integral formula, it results the general weak formulation of the periodic γ -harmonic problem:



The variational form

Find (\mathbf{u}, ϕ) in $V_\gamma^3(\Omega_m) \times V_\gamma(\Omega)$ (verifying the equipotential boundary condition in the electrode), such that for all (\mathbf{v}, ψ) in $V_\gamma^3(\Omega_m) \times V_\gamma^3(\Omega)$, satisfying the zero equi potential boundary condition), we have:

$$\begin{aligned} & \int_{\Omega_m} \overline{\mathbf{S}(\mathbf{v})} : \mathbf{T}(\mathbf{u}) \, d\Omega - \omega^2 \int_{\Omega_m} \rho \bar{\mathbf{v}} \cdot \mathbf{u} \, d\Omega \\ & \quad - \int_{\Omega} \overline{\mathbf{E}(\psi)} \cdot (\mathbf{e}\mathbf{S}(\mathbf{u}) + \varepsilon \mathbf{E}(\phi)) \, d\Omega \\ & \quad - \int_{\Gamma_d} \bar{\mathbf{v}} \cdot (\mathbf{T}(\mathbf{u}) \cdot \mathbf{n}) \, d\Gamma - \int_{\Gamma_u \cup \Gamma_d} \bar{\psi} (\mathbf{D}(\phi) \cdot \mathbf{n}) \, d\Gamma = 0 \quad (17) \end{aligned}$$

With, $V_\gamma(\Omega)$ is the mathematical space of $L^2(\Omega)$ with the derivative in $L^2(\Omega)$ satisfying γ -harmonic periodic boundary conditions.



The γ -harmonic periodic boundary trick

Let us first define $\varphi_\gamma(x) = e^{-j2\pi\gamma \frac{x}{p}}$, $\varphi_\gamma(x)$ is a γ -harmonic periodic function satisfying:

$$\varphi_\gamma(x+p) = e^{-j2\pi\gamma} \varphi_\gamma(x) \quad (18)$$

We just do the change of variable:

$$\begin{cases} \mathbf{u}(x, y) = \varphi_\gamma(x) \mathbf{u}^\diamond(x, y) \\ \phi(x, y) = \varphi_\gamma(x) \phi^\diamond(x, y) \end{cases} \quad (19)$$

Where $\mathbf{u}^\diamond(x)$ and $\phi^\diamond(x)$ are p -periodic functions.

The main idea is to define a new differential operator ∇_γ by:

$$\nabla_\gamma \mathbf{u}^\diamond = \nabla(\varphi_\gamma \mathbf{u}^\diamond) = \varphi_\gamma \nabla \mathbf{u}^\diamond + \varphi_\gamma' \mathbf{u}^\diamond \quad (20)$$

Because the physical fields \mathbf{E} , \mathbf{D} , \mathbf{T} , and \mathbf{S} are expressed using partial derivative of \mathbf{u} , and, ϕ , it is possible to define the operators $\mathbf{E}_\gamma(\phi^\diamond) = \mathbf{E}(\varphi_\gamma \phi^\diamond)$, $\mathbf{D}_\gamma(\phi^\diamond) = \mathbf{D}(\varphi_\gamma \phi^\diamond)$, $\mathbf{T}_\gamma(\mathbf{u}^\diamond) = \mathbf{T}(\varphi_\gamma \mathbf{u}^\diamond)$,



The new variational form with period BC.

Find $(\mathbf{u}^\diamond, \phi^\diamond)$ in $V_1^3(\Omega_m) \times V_1(\Omega)$ (verifying the equipotential boundary condition), such that for all $(\mathbf{v}^\diamond, \psi^\diamond)$ in $V_1^3(\Omega_m) \times V_1^3(\Omega)$, satisfying the zero equipotential boundary condition), we have:

$$\begin{aligned} & \int_{\Omega_m} \overline{\mathbf{S}_\gamma(\mathbf{v}^\diamond)} : \mathbf{T}_\gamma(\mathbf{u}^\diamond) \, d\Omega - \omega^2 \int_{\Omega_m} \rho \overline{\mathbf{v}^\diamond} \cdot \overline{\mathbf{u}^\diamond} \, d\Omega \\ & \quad - \int_{\Omega} \overline{\mathbf{E}_\gamma(\psi^\diamond)} \cdot (\mathbf{e} \mathbf{S}_\gamma(\mathbf{u}^\diamond) + \varepsilon \mathbf{E}_\gamma(\phi^\diamond)) \, d\Omega \\ & \quad - \int_{\Gamma_d} \overline{\varphi_\gamma \mathbf{v}^\diamond} \cdot (\mathbf{T}_\gamma(\mathbf{u}^\diamond) \mathbf{n}) \, d\Gamma - \int_{\Gamma_u \cup \Gamma_d} \overline{\varphi_\gamma \psi^\diamond} (\mathbf{D}_\gamma(\phi^\diamond) \cdot \mathbf{n}) \, d\Gamma = 0 \quad (21) \end{aligned}$$

Where, $V_1(\Omega)$ is the mathematical space of $L^2(\Omega)$ with derivative in $L^2(\Omega)$ satisfying p-periodic boundary conditions.



We have to modelized the following term:

$$-\int_{\Gamma_d} \overline{\varphi_\gamma \mathbf{v}^\diamond} \cdot (\mathbf{T}_\gamma(\mathbf{u}^\diamond) \mathbf{n}) d\Gamma - \int_{\Gamma_u \cup \Gamma_d} \overline{\varphi_\gamma \psi^\diamond} (\mathbf{D}_\gamma(\phi^\diamond) \cdot \mathbf{n}) d\Gamma, \quad (22)$$

also called border terms.

First from (22), let us look at the boundary integral A_{Γ_u} , at the interface Γ_u of the semi-infinite dielectric semi-space.

$$A_{\Gamma_u} = \int_{\Gamma_u} \overline{\varphi_\gamma \psi^\diamond} \mathbf{D}_\gamma(\phi^\diamond) \cdot \mathbf{n} d\Gamma \quad (23)$$



BEM computation

The elementary coefficients to compute are for all finite element basic functions w^\diamond_i introduce in (??), only for node $i \in \mathcal{N}_u$ the set of node on Γ_u .

$$\forall (i, j) \in \mathcal{N}_u^2, \quad (A_{\Gamma_u})_{ij} = -\varepsilon_d \int_{\Gamma_u} \overline{\varphi_\gamma w^\diamond_i} \partial_n (\varphi_\gamma w^\diamond_j) d\Gamma \quad (24)$$

According [2], it is possible to expand, at the interface Γ_u the γ -harmonic periodic $\varphi_\gamma w^\diamond_j$ into the Floquet's basis function

$$f_m(x, y) = e^{-2\pi(j(m+\gamma)x - |m+\gamma|((y-y_u))/p)} = \varphi_\gamma(x) f^\diamond_m(x, y). \quad (25)$$

where y_u is the y coordinate of Γ_u .

$$\varphi_\gamma(x) w^\diamond_j(x, y) = \sum_{m=-\infty}^{+\infty} c_m^j f_m(x, y) \quad (26)$$

With the $L^2(\Gamma_u)$ orthogonality of Fourier's basis f^\diamond_m , we have:

$$c_m^j = \frac{1}{p} \int_{\Gamma_u} w^\diamond_j \overline{f^\diamond_m} d\Gamma, \quad (27)$$



BEM computation,

and on Γ_u the normal derivative $\partial_n f_m(x, y) = \partial_y f_m(x, y)$ satisfies:

$$\partial_n f_m = -g_m f_m, \quad \text{with } g_m = \frac{2\pi}{p} |\gamma + m| \quad (28)$$

Leading to the relationship:

$$\partial_n (\varphi_\gamma w^\diamond_j) = - \sum_{m=-\infty}^{+\infty} c_m^j g_m f_m \quad (29)$$

Finally the term $(A_{\Gamma_u})_{ij}$ is

$$(A_{\Gamma_u})_{ij} = \frac{\varepsilon_d}{p} \sum_{m=-\infty}^{+\infty} g_m \int_{\Gamma_u} \overline{w^\diamond_i} f^\diamond_m d\Gamma \int_{\Gamma_u} w^\diamond_j \overline{f^\diamond}_m d\Gamma \quad (30)$$

Run:BEM.edp



7 No Linear Problem

- Newton Method
- Navier-Stokes
- Variational Inequality
- Ground water
- Bose Einstein Condensate
- Hyper elasticity equation
- Periodic Surface Acoustic Waves Transducer Analysis
- Phase change with Natural Convection



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.1j11.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 $\mathbf{u} = (u_1, u_2)$, the pressure p and temperature θ :

$$\left\{ \begin{array}{lll} \mathbf{u} & \text{given} & \text{in } \Omega_s \\ \partial_t \mathbf{u} + (\mathbf{u} \nabla) \mathbf{u} + \nabla \cdot \mu \nabla \mathbf{u} + \nabla p & = -c_T \mathbf{e}_2 & \text{in } \Omega_f \\ \nabla \cdot \mathbf{u} & = 0 & \text{in } \Omega_f \\ \partial_t \theta + (\mathbf{u} \nabla) \theta + \nabla \cdot k_T \nabla \theta & = \partial_t S(T) & \text{in } \Omega \end{array} \right. \quad (31)$$

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 :

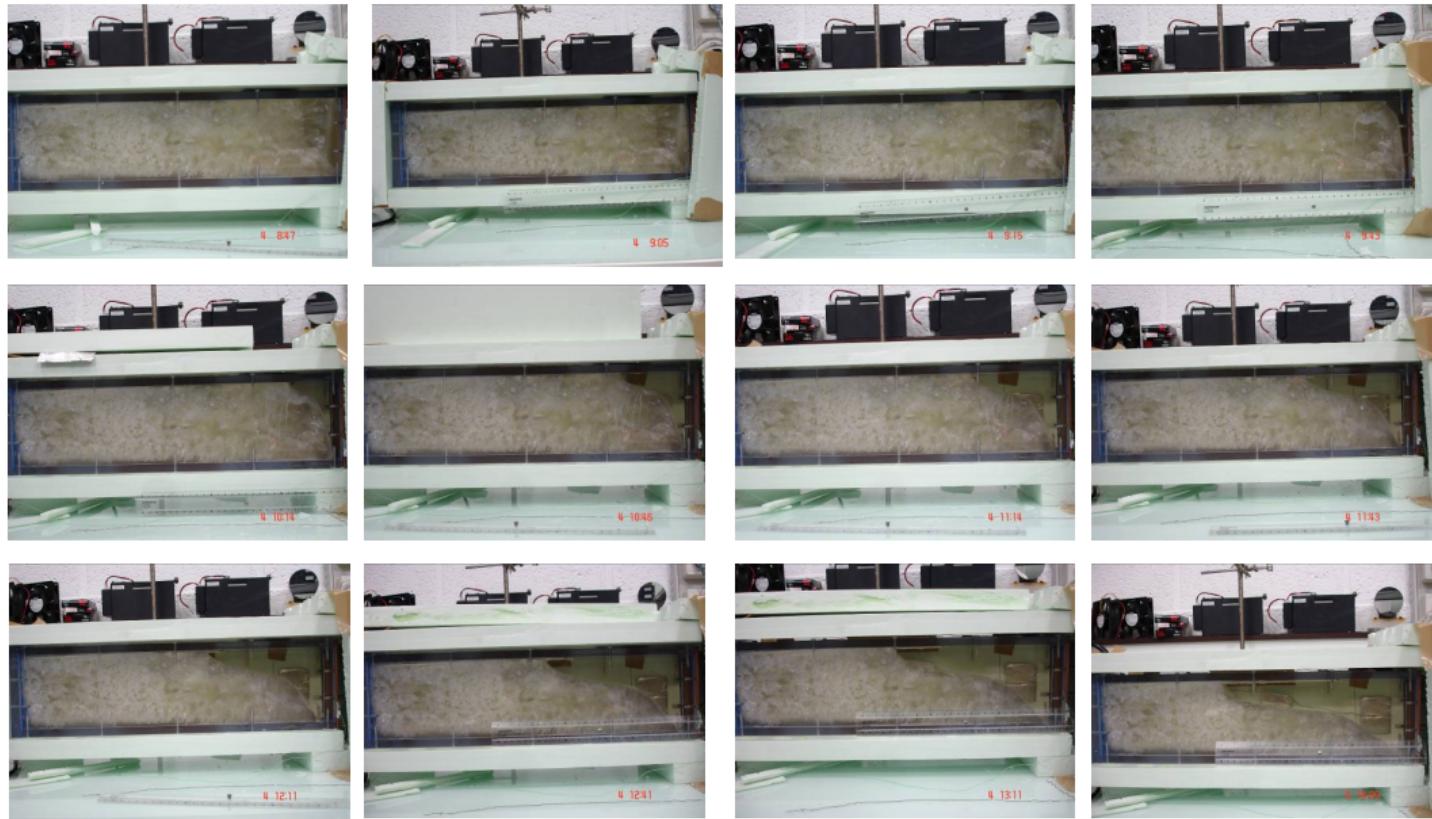
$$\mu = \begin{cases} \theta < \theta_f & \sim 10^6 \\ \theta \geq \theta_m & \sim \frac{1}{Re} \end{cases},$$

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

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



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 u_1, u_2, p, θ is defined by

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

We do mesh adaptation at 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 means, we adapt with all variables plus the 2 melting phases at 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 = 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*T*w*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*w*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

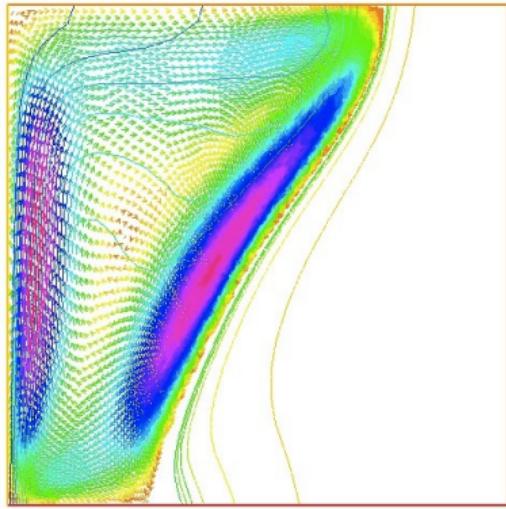
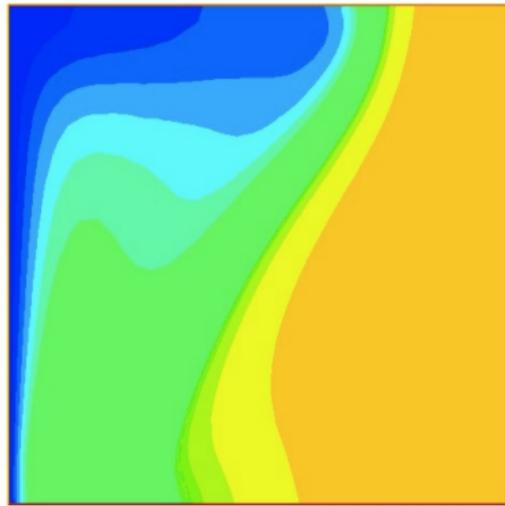
take case 2 from

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 $cmT = cT = -R_a/P_r$, $kT = k_T = 1/P_r$, $\text{eps} = 10^{-6}$, time step $\delta t = 10^{-1}$, $c dt = 1/\delta t$, at time $t = 80$ and we get a good agreement with the article.



Phase change with Natural Convection



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

Outline

1 Introduction

2 Tools

3 Academic Examples

4 Numerics Tools

5 Schwarz method with overlap

6 Exercises

7 No Linear Problem

8 Technical Remark on freefem++



8 Technical Remark on freefem++

- compilation process
- Plugin
- Plugin to read image
- Plugin of link code through a pipe
- FreeFem++ et C++ type



compilation process on Windows

① Download and install MINGW32 see

<http://sourceforge.net/projects/mingw/files/Installer/mingw-get-inst/mingw-get-inst-20120426/>

② Under mingw32 install wget and unzip

- mingw-get install msys-wget
- mingw-get.exe install msys-unzip

③ To install freeglut of win32 for the graphics part

```
wget http://files.transmissionzero.co.uk/software/development/GLUT/freeglut-MinGW.zip
unzip freeglut-MinGW-2.8.0-1.mp.zip
cp freeglut/include/* /c/MinGW/include/GL/.
cp freeglut/lib*.a /c/MinGW/lib/.
cp freeglut/freeglut.dll /bin
```

④ install a good blas (OpenBlas) <http://xianyi.github.com/OpenBLAS/>

⑤ install MPI for // version HPC Pack 2008 SDK and HPC Pack 2008 R2 Service Pack 2

⑥ install inno setup to build installer: <http://www.xs4all.nl/~mlaan2/ispack/isetup-5.4.0.exe>

⑦ GSL for gsl interface <http://sourceforge.net/projects/mingw-cross/files/%5BLIB%5D%20GSL/mingw32-gsl-1.14-1/mingw32-gsl-1.14-1.zip/download>

Finaly, the configure argument are:

```
./configure '--enable-download' 'FC=mingw32-gfortran' 'F77=mingw32-gfortran' 'CC=mingw32-gcc'
'CXX=mingw32-g++' '-with-blas=/home/hecht/blas-x86/libgoto2.dll' 'CXXFLAGS=-I/home/hecht/blas-x86'
'--enable-generic' '--with-wget=wget' 'MPIRUN=/c/Program Files/Microsoft HPC Pack 2008 R2/Bin/mpieexec.exe'
```



8 Technical Remark on freefem++

- compilation process
- **Plugin**
- Plugin to read image
- Plugin of link code through a pipe
- FreeFem++ et C++ type



Dynamics Load facility

Or How to add your C++ function in FreeFem++.

First, like in cooking, the first true difficulty is how to use the kitchen.

I suppose you can compile the first example for the examples++-load

```
numerical11:FH-Seville hecht# ff-c++ myppm2rnm.cpp
```

```
...
```

```
add tools to read pgm image
```



8 Technical Remark on freefem++

- compilation process
- Plugin
- **Plugin to read image**
- Plugin of link code through a pipe
- FreeFem++ et C++ type



The interesting code

```
#include "ff++.hpp"
typedef KNM<double> * pRnm; // real[int,int] array variable type
typedef KN<double> * pRn; // real[int] array variable type
typedef string ** string; // the ff++ string variable type
                           // the function to read image

pRnm read_image( pRnm const & a,const pstring & b); // the function to set 2d array from 1d array

pRn seta( pRn const & a,const pRnm & b)
{ *a=*b;
  KN_<double> aa=*a;
  return a; }

void Init(){ // the link with FreeFem++
  // add ff++ operator "<-" constructor of real[int,int] form a string
  TheOperators->Add("<-",
    new OneOperator2_<KNM<double> *,KNM<double> *,string*>(&read_image) );
  // add ff++ an affection "=" of real[int] form a real[int,int]
  TheOperators->Add("=", // the link with FreeFem++
    new OneOperator2_<KN<double> *,KN<double> *,KNM<double>*>(seta));
}
LOADFUNC(Init); // to call Init Function at load time
```

Remark, **TheOperators** is the ff++ variable to store all world operator, **Global** is to store function.



8 Technical Remark on freefem++

- compilation process
- Plugin
- Plugin to read image
- Plugin of link code through a pipe
- FreeFem++ et C++ type



How to extend

A true simple example How to make dynamic gnuplot

Idea: use a pipe to speak with gnuplot the C code :

```
FILE * gp = popen("gnuplot");
for( double f=0; f < 3.14; f += 0.01)
    fprintf(gp,"plot sin(x+%f) \n", f);
```

To do this add a new constructor of ofstream in freefem++



A way to pass info between to code

Make a pipe, under unix (with a use of pstream tools)

```
#include "ff++.hpp"
#include "pstream.h"
typedef redi::pstream pstream;
typedef std::string string;
static pstream ** pstream_init(pstream **const & p, string * const & a)
{ *p = new pstream(a->c_str());
  return p; }

void inittt()
{
    // add new pointer type * pstream
Dcl_TypeandPtr<pstream*>(0,0,::InitializePtr<pstream*>,::DeletePtr<pstream*>);
    // add cast operation to make std iostream read and write
atype<iostream*>()->AddCast( new E_F1_funcT<iostream*,pstream**>(UnRef<iostream*>));
atype<ostream*>()->AddCast( new E_F1_funcT<ostream*,pstream**>(UnRef<ostream*>));
    // the constructor from a string .
TheOperators->Add("<-", new OneOperator2_<pstream**,pstream**,string*>(pstream_init) );
    // add new keyword type pstream
zzzfff->Add("pstream",atype< pstream ** >());
}
LOADFUNC(inittt);
t

MBP-FH:plugin hecht$ ff-c++ pipe.cpp
/usr/local/bin/g++ -c -g -m64 -fPIC -DNDEBUG -O3 -DBAMG_LONG_LONG -DNCHECKPTR -fPIC -I/usr/local/lib/ff++/3.20/include
/usr/local/bin/g++ -bundle -undefined dynamic_lookup -g -m64 -fPIC -DNDEBUG -O3 -DBAMG_LONG_LONG -DNCHECKPTR -fPIC 'pipe'
```

a small test : Run:gnuplot.edp



8 Technical Remark on freefem++

- compilation process
- Plugin
- Plugin to read image
- Plugin of link code through a pipe
- FreeFem++ et C++ type



FreeFem++ et C++ type

The tools to add a operator with 2 arguments:

```
OneOperator2_<returntype ,typearg1 ,typearg2>(& thefunction );
returntype thefunction(typearg1 const &, typearg2 const &)
```

To get the C++ type of all freefem++ type, method, operator, just do in examples++-tutorialdirectory

```
c++filt -t < lestables
Cmatrix 293 Matrice_Creuse<std::complex<double> >
R3 293 Fem2D::R3
bool 293 bool*
complex 293 std::complex<double>*
element 293 (anonymous namespace)::lgElement
func 294 C_F0
    ifstream 293 std::basic_istream<char, std::char_traits<char> >**
int 293 long*
matrix 293 Matrice_Creuse<double>
mesh 293 Fem2D::Mesh**
mesh3 293 Fem2D::Mesh3**
ofstream 293 std::basic_ostream<char, std::char_traits<char> >**
problem 294 Problem
real 293 double*
solve 294 Solve
string 293 std::basic_string<char, std::char_traits<char>, std::allocator<char> >**
varf 294 C_args
vertex 293 (anonymous namespace)::lgVertex
```



FreeFem++ Triangle/Tet capabylity

```
// soit T un Element de sommets A,B,C ∈ ℝ2
// -----
// number of vertices of triangle (here 3)
//      the vertex i of T (i ∈ 0,1,2
//      mesure of T
//      edge vector
//      gradient of 2 base fonction
//      length of i edge oppose of i
//      label of T (region number)
//      The barycentre of T in 3d
Element::nv ;
const Element::Vertex & V = T[i];
double a = T.mesure() ;
Rd AB = T.Edge(2);
Rd hC = T.H(2) ;
R l = T.lenEdge(i);
(Label) T ;
R2 G(T(R2(1./3,1./3)));
```



FreeFem++ Mesh/Mesh3 capability

```
Mesh Th("filename");                                // read the mesh in "filename"
Th.nt ;                                              // number of element (triangle or tet)
Th.nv ;                                              // number of vertices
Th.neb or Th.nbe ;                                 // number of border element (2d) or (3d)
Th.area;                                            // area of the domain (2d)
Th.peri;                                            // length of the border
typedef Mesh::Rd Rd;
Mesh2::Element & K = Th[i];
Rd A=K[0];                                         // coor of vertex 0 of triangle K
Rd G=K(R2(1./3,1./3)):                           // the barycentre de K.
Rd DLambda[3];
K.Gradlambda(DLambda);                            // compute the 3  $\nabla \lambda_i^K$  for  $i = 0, 1, 2$ 
Mesh::Vertex & V = Th(j);                         // vertex j , int j $\in [0, nv[$ 
Mesh::BorderElement & BE=th.be(l);                // border element l $\in [0, nbe[$ 
Rd B=BE[1];                                         // coord of vertex 1 on Seg BE
Rd M=BE(0.5);                                      // middle of BE.
int j = Th(i,k);                                // global number of vertex k $\in [0, 3[$  of tria. i $\in [0, nt[$ 
Mesh::Vertex & W=Th[i][k];                        // vertex k $\in [0, 3[$  of triangle i $\in [0, nt[$ 

int ii = Th(K) ;                                  // number of triangle K
int jj = Th(V) ;                                  // number of triangle V
int ll = Th(BE) ;                                // number of Seg de bord BE
assert( i == ii && j == jj) ;                      // check.
```



Freefem++ v3.47 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 anisotropic mesh adaptation
- automate the parallel tool (in progress)

Thank for you attention.

