Utilisation de FreeFem++ pour analyser des résultats de calculs

F. Hecht

LJLL, Université Pierre et Marie Curie, Paris — projet Alpines, Inria de Paris

with P-E. Emeriau, G. Vergez, I. Danaila

march 23th 2017

http://www.freefem.org mailto:frederic.hecht@upmc.fr

Outline

Outline

Introduction

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

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

FreeFem++ is a freeware and this run on Mac, Unix and Window architecture, in parallel with MPI.
To try of cell phone `https://www.ljll.math.upmc.fr/lehyaric/ffjs/`

The 9^{th} FreeFem++ days, 1st week of December, 2017, UPMC, Jussieu, Paris, France
Info: FreeFem++ solve a problem with $22\,10^9$ unknowns in 200 s on 12,000 proc.

Outline

- 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.

- Analytic description of boundaries, with specification by the user of the intersection of boundaries in 2d.
- Automatic mesh generator, based on the Delaunay-Voronoï algorithm. (2d,3d (tetgen))
- load and save Mesh, solution
- Mesh adaptation based on metric, possibly anisotropic, with optional automatic computation of the metric from the Hessian of a solution. (2d,3d mmg3d).
- Link with other soft: parview, gmsh , vtk, medit, gnuplot
- Dynamic linking to add plugin.
- Full MPI interface
- Nonlinear Optimisation tools: CG, Ipopt, NLOpt, stochastic
- Wide range of examples: Navier-Stokes 3d, elasticity 3d, fluid structure, eigenvalue problem, Schwarz' domain decomposition algorithm, residual error indicator ...

Outline

Laplace equation, weak form

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

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

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

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

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

Poisson equation in a fish with FreeFem++

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

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

 Vh u,v;                    //   set test and unknown function in Vh.
macro Grad(u) [dx(u),dy(u),dz(u)]                    //   EOM Grad def
solve laplace(u,v,solver=CG) =
   int3d(Th)(  Grad(u)'*Grad(v)    )
   - int3d(Th) ( 1*v)
   + on(2,u=2);                                          //    BC on Γ2
plot(u,fill=1,wait=1,value=0,wait=1);
```

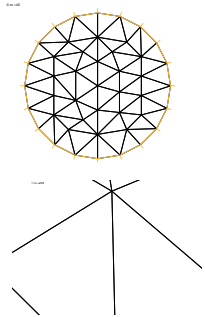
Run:fish.edp Run:fish3d.edp

2. Academic Examples
   - Weak form
   - Anisotropic Mesh adaptation
   - Incompressible Navier-Stokes
   - Monolitic Fuild Structure interaction

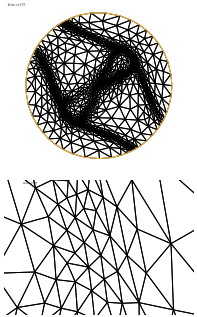
Example of adaptation process

Find optimal mesh in norm L^∞ to represent:

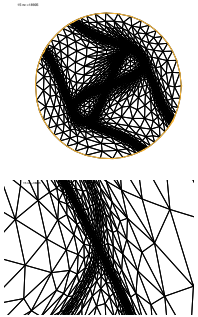
$$u = 10x^3 + y^3 + tanh(50\ (sin(5y) - 2x)\ );$$
$$v = 10y^3 + x^3 + tanh(500(sin(5x) - 2y)\ );$$



Run:Adapt-uv.edp



Run:CornerLap.edp



Run:LaplaceDiracP2.edp

Example of adaptation process in 3d with mmg3

Let a domain $\Omega =]0,1[^3 \backslash [\frac{1}{2},1[^3$ The border of $\partial\Omega$ is split in 2 part

- Γ_2, if $x = 1$, $y = 1$, or $z = 1$
- Γ_1, else.

Find u a solution in such that:

$$\begin{aligned} -\Delta u &= 1 && \text{in } \Omega, \\ \frac{\partial u}{\partial \vec{n}} &= 0 && \text{on } \Gamma_2, \\ u &= 0 && \text{on } \Gamma_1. \end{aligned}$$

Thank to mmg3 to do 3d mesh adaptation.

Run:Laplace-Adapt-3d.edp

Run:Laplace-Adapt-aniso-3d.edp

2. Academic Examples
   - Weak form
   - Anisotropic Mesh adaptation
   - Incompressible Navier-Stokes
   - Monolitic Fuild Structure interaction

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

1. u^0 a initial guest
2. do
   - find w^n solution of $DF(u^n)w^n = F(u^n) + BC0$
   - $u^{n+1} = u^n - w^n$ or $DF(u^n)u^{n+1} = DF(u^n)u^n - F(u^n) + BC$
   - if($||w^n|| < \varepsilon$) break;

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

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

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

Run:cavityNewton.edp

Run:NSNewtonCyl-100-mpi.edp

Run:NSCaraCyl-100-mpi.edp

The generalise Stokes problem is find u, p solution of

$$Au + Bp = f, \quad {}^tBu = 0$$

with $A \equiv (\alpha Id + \nu\Delta)$ and $B \equiv \nabla$. remark,if A est symmetric positive the you can use a conjugate gradient to solve the the following problem

$${}^tBA^{-1}B\,p = {}^tBA^{-1}f$$

Now in a periodic domain, all differential operators commute and the Uzawa algorithm comes to solving the linear operator $-\nabla.((\alpha Id - \nu\Delta)^{-1}\nabla$, where Id is the identity operator. So the preconditioner suggested is $-\alpha\Delta^{-1} + \nu Id$.
the term $\frac{\partial u}{\partial t} + (u \cdot \nabla)u$ is the total derivative and discretization in time

$$\begin{array}{rl} \frac{1}{\tau}(u^{n+1} - u^n \circ X^n) - \nu\Delta u^{n+1} + \nabla p^{n+1} & = 0, \\ \nabla \cdot u^{n+1} & = 0 \end{array} \tag{3}$$

The term $X^n(x) \approx x - \tau u^n(x)$ will be computed with convect operator.
Run:NSUzawaCahouetChabart.edp Run:NSUzawaCahouetChabart-3d-aorte.edp

2. Academic Examples
   - Weak form
   - Anisotropic Mesh adaptation
   - Incompressible Navier-Stokes
   - Monolitic Fuild Structure interaction

Monolitic Fuild Structure interaction

Beam in a flow Movie:turekcomp

Bouncing Ball Movie:bouncingBall

Bouncing Ball in fluid Movie:bouncingBallInLiquid

Outline

Bose Einstein Condensate

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

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

to code that in FreeFem++
use

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

The idea to mixte Ipopt and adapt mesh is play this stop criterion, and finally use freefem++ to analyse the result.

Run:BEC.edp

3. Bose Einstein Condensate
   - Search all local min
   - Read image and extract an isoline
   - Best Fit
   - Delaunay mesh
   - Analyse of a Condensate

Search all local min

The function `findalllocalmin` find all the le local min and use a greedy algorithm to to the local attraction zone, by adding the triangle through the minimal vertices.

```
mesh Th=square(50,50,[x*2-1,y*2-1]);
load "isoline"
fespace Vh(Th,P1), Ph(Th,P0);
int k =2;
Vh u= sin(k*pi*x)*sin(k*pi*y);
plot(u, wait=1);
Ph r;
int[int] lm=findalllocalmin(Th,u[],r[]);
// lm array gives the vertex number of all the local min
// r is function P0 defined the attraction zone of the local min
// (local min number)
plot(r,u,fill=1,wait=1);
// to see where is the minimuns
Ph mx= Th(lm[real(r)]).x -x, my= Th(lm[real(r)]).y -y;
plot([mx,my],u,wait=1,fill=0);
```

Run:findalllocalmin.edp

Run:findalllocalminbec.edp

Read image and extract an isoline

```
load "ppm2rnm" load "isoline" load "shell"
string lac="metz", lacjpg =lac+".jpg", lacpgm =lac+".pgm";
if(stat(lacpgm)<0)  exec("convert␣"+lacjpg+"␣"+lacpgm);
  real[int,int] ff1(lacpgm); // read pgm thank to ppm2rnm
  int nx = ff1.n, ny=ff1.m; // grey value beetween 0 to 1 (dark)
  mesh Th=square(nx-1,ny-1,[(nx-1)*(x),(ny-1)*(1-y)]);
  fespace Vh(Th,P1);
```

Extract the isoline

```
real[int,int] xys(3,1); int[int] be(1); // curve k :  p=be[2k],
   be[2k+1]-1 :  xys(.,k)
int nc;// nb of curve
nc=isoline(Th,f1,xys,iso=0.8,close=0,Curves,beginend=be);
```

Use of curve

```
int[int] iii=[0,3]; /
border G(t=0,1;i) {  P=Curve(Curves,be[2*iii[i]],be[2*iii[i]+1)-1,
   t];  label= iii[i];}
```

Run:metz.edp

Just use `ipopt` to find the arg min of $J(\alpha) = \int_\Omega (u - \phi_\alpha)^2)$ where $alpha$ is the set of parameters.

```
real[int]  data0 = [ ux , x0, y0 , s0] ;  // start point

func real J(real[int] & dat){
   alpha=dat;
   return int2d(Th)(square(u-phialpha)) ;}

func real[int]  dJ(real[int] & dat){
   alpha=dat;
    dat[0]=int2d(Th)(-2*(u-phialpha)*d0phialpha) ;
    dat[1]=int2d(Th)(-2*(u-phialpha)*d1phialpha) ;
    dat[2]=int2d(Th)(-2*(u-phialpha)*d2phialpha) ;
    dat[3]=int2d(Th)(-2*(u-phialpha)*d3phialpha) ;
   return dat;  }
 real[int] data=data0;
 verbosity=0;
 int r = IPOPT(J,dJ,data,printlevel=0);
```

Run:fit-ipopt.edp

Best Fit

Just use ipopt to find the arg min of $J(\alpha) = \int_\Omega (u - \phi_a lpha)^2)$ where α is the set of parameters.

```
real[int]  data0 = [ ux , x0, y0 , s0] ;  // start point

func real J(real[int] & dat){
   alpha=dat;
   return int2d(Th)(square(u-phialpha)) ;}

func real[int]  dJ(real[int] & dat){
   alpha=dat;
    dat[0]=int2d(Th)(-2*(u-phialpha)*d0phialpha) ;
    dat[1]=int2d(Th)(-2*(u-phialpha)*d1phialpha) ;
    dat[2]=int2d(Th)(-2*(u-phialpha)*d2phialpha) ;
    dat[3]=int2d(Th)(-2*(u-phialpha)*d3phialpha) ;
   return dat;  }
 real[int] data=data0;
 verbosity=0;
 int r = IPOPT(J,dJ,data,printlevel=0);
```

Run:fit-ipopt.edp

Best Fit axisymmetric

On the domain with no vortex, all just do the L2 projection on axisymmetric space with a laplace regularisation

```
Ph pok=data7(6,real(r)); // domain with no hole
func r = sqrt(x*x+y*y);
Vh pr = r;
real R = pr[].max;
mesh Th1d=square(200,1,[x*R,y]);
fespace V1d(Th1d,P1,periodic=[[1,x],[3,x]]);// dat axi
mesh The = trunc(Thg,pok==1);  // mesh
   Vh u2=u*u;
varf vM1d(u,v) = int1d(Th1d,1)(dx(u)*dx(v))
     + int2d(The,  mapu=[r,0] ,  mapt=[r,0]  )(u*v);
matrix M=vM1d(V1d,V1d,solver=CG);
varf vb1d(u,v) = int2d(The,mapt=[r,0])(u2*v);
real[int] b1d=vb1d(0,V1d);  V1d u1dt;
u1dt[]=M^-1*b1d;
Vh u20 = u1dt(r,0); // Axi -> 2d
plot(u20,u2,wait=1,dim=3);
```

Run:fit-axi.edp

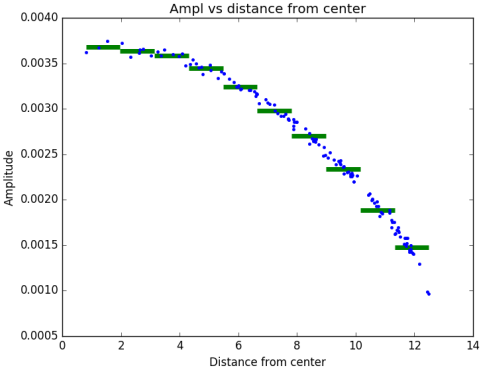
Delaunay mesh

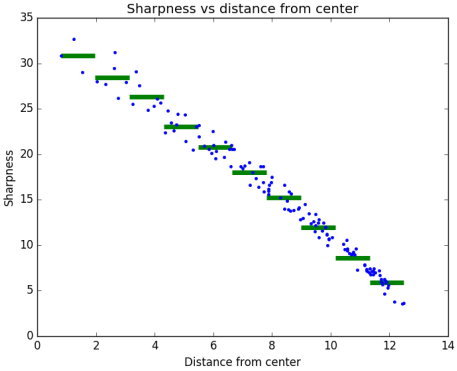
```
mesh Thc=triangulate(data7(0,:),data7(1,:));

fespace Eh(Thc,P0edge);// Element P0 / Edge
varf vedge(u,v) = intalledges(Thc,qforder=1)((nTonEdge==2)*v/
   nTonEdge);
real[int] eih=vedge(0,Eh);
int ei=0;
for(int e=0; e < eih.n; ++e) if(eih[e])  eih[ei++]=eih[e];
eih.resize(ei);
real moy =  eih.sum/ eih.n ;
// Statistic
real[int] dd = eih;dd-= moy;
real variance = dd.l2 /  dd.n;
cout << " moy eih =" << moy<< " standart deviation " << sqrt(
   variance) <<endl;
for(int i=1 ; i<10; ++i)
cout <<" quantile "<<i/10.<< " = " << eih.quantile(i/10.) <<endl;
```

Run:analyssolbec.edp

3. Bose Einstein Condensate
   - Search all local min
   - Read image and extract an isoline
   - Best Fit
   - Delaunay mesh
   - Analyse of a Condensate

Analyse of a Condensate





Outline

Conclusion/Future

Freefem++ v3 is

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

The the future we try to do:

- Clean the installation procedure, and the linear solver part,
- automate the parallel tool
- 3d anisotrope mesh adaptation (a true integration of MMG3d V5)
- Add Integral Formulation and fast multi pole method (X. Claeys).
- Build more graphic with VTK, paraview , ... (in progress)
- Add Finite volume facility for hyperbolic PDE (just begin C.F. FreeVol Projet)

Thank for you attention.