

## New Developments in FreeFem++

F. HECHT\*

*Received May 30, 2012*

*Communicated by Yu. A. Kuznetsov*

*Received in revised form March 1, 2012*

**Abstract** — This is a short presentation to the capability of the freefem++ software, in section 1, we recall most of the characteristics of the software, In section 2, we recall how to build a weak form of an partial differential equation (PDE) from the strong form of the PDE. In three last sections, we present different problem, tools to illustrated the software. First we do mesh adaptation problem in two and three dimension,, secondly, we solve numerically a Phase change with Natural Convection, and the finally to show the HPC possibility we show a Schwarz Domain Decomposition problem on parallel computer.

**Keywords:** Finite Element

### 1. Introduction

This paper intends to give a small presentation of the software `FreeFem++`.

A partial differential equation is a relation between a function of several variables and its (partial) derivatives. Many problems in physics, engineering, mathematics and even banking are modeled by one or several partial differential equations.

`FreeFem++` is a software to solve these equations numerically in dimensions two or three. As its name implies, it is a free software based on the Finite Element Method; it is not a package, it is an integrated product with its own high level programming language; it runs on most UNIX, WINDOWS and MacOS computers. Moreover `FreeFem++` is highly adaptive. Many phenomena involve several coupled systems, for example: fluid-structure interactions, Lorentz forces for aluminum casting and ocean-atmosphere coupling. Multiphysics problems require different finite element approximations or different polynomial degrees, possibly on different meshes. Some algorithms like Schwarz' domain decomposition method also require data interpolation on multiple meshes within one program. `FreeFem++` can handle these difficulties, i.e. *arbitrary finite element spaces on arbitrary unstructured and adapted meshes*.

---

\*UPMC, Univ. Paris 06, UMR 7598, F-75005 Paris, France. E-mail: frederic.hecht@upmc.fr  
This work was partially supported by ANR grant No. 01-2345-6789.

The characteristics of `FreeFem++` are:

- Problem description (real or complex valued) by their variational formulations, with access to the internal vectors and matrices if needed.
- Multi-variables, multi-equations, bi-dimensional and three-dimensional static or time dependent, linear or nonlinear coupled systems; however the user is required to describe the iterative procedures which reduce the problem to a set of linear problems.
- Easy geometric input by analytic description of boundaries by pieces; however this part is not a CAD system; for instance when two boundaries intersect, the user must specify the intersection points.
- Automatic mesh generator, based on the Delaunay-Voronoi algorithm; the inner point density is proportional to the density of points on the boundaries [17].
- Metric-based anisotropic mesh adaptation. The metric can be computed automatically from the Hessian of any `FreeFem++` function [20].
- High level user friendly typed input language with an algebra of analytic or finite element functions.
- Multiple finite element meshes within one application with automatic interpolation of data on different meshes and possible storage of the interpolation matrices.
- A large variety of triangular finite elements : linear, quadratic Lagrangian elements and more, discontinuous P1 and Raviart-Thomas elements, elements of a non-scalar type, the mini-element, ... but no quadrangles.
- Tools to define discontinuous Galerkin finite element formulations P0, P1dc, P2dc and keywords: `jump`, `mean`, `intalldges`.
- A large variety of linear direct and iterative solvers (LU, Cholesky, Crout, CG, GMRES, UMFPACK [13], MUMPS [2], SuperLU, ...) and eigenvalue and eigenvector solvers (ARPACK) [24], and Optimatization tools like Ipopt [33]
- Near optimal execution speed (compared with compiled C++ implementations programmed directly).
- Online graphics, generation of `.txt`, `.eps`, `.gnu`, `mesh` files for further manipulations of input and output data.
- Many examples and tutorials: elliptic, parabolic and hyperbolic problems, Navier-Stokes flows, elasticity, Fluid structure interactions, Schwarz's domain decomposition method, eigenvalue problem, residual error indicator, ...

- A parallel version using `mpi`

## 2. Weak formulation

For the first example consider a Poisson problem on a Domain  $\Omega$  with a partition of the boundary  $\partial\Omega$  in  $\Gamma_2, \Gamma_e$ : Find  $u$  such that:

$$-\Delta u = 1 \text{ in } \Omega, \quad u = 2 \text{ on } \Gamma_2, \quad \frac{\partial u}{\partial \vec{n}} = 0 \text{ on } \Gamma_e \quad (2.1)$$

Denote  $V_g = \{v \in H^1(\Omega) / v|_{\Gamma_2} = g\}$ , and the weak form of the equation is obtained by multiplying by  $v$  and integrating by parts (2.1). The variational formulation is: find  $u \in V_2(\Omega)$ , such that

$$\forall v \in V_0(\Omega), \quad \int_{\Omega} \nabla u \cdot \nabla v = \int_{\Omega} 1v + \int_{\Gamma} \frac{\partial u}{\partial n} v, \quad (2.2)$$

Note that due to  $\frac{\partial u}{\partial \vec{n}} = 0$  on  $\Gamma_2$  and  $v = 0$  the term  $\int_{\Gamma} \frac{\partial u}{\partial n} v$  disappears on  $\Gamma_2$ . Finally the finite method simply replaces  $V_g$  with a finite element space, such as

$$V_{gh} = \{v_h \in C^0(\bar{\Omega}) : \forall K \in \mathcal{T}_h, v_h|_K \in P_1 \text{ and } v_h|_{\Gamma_2} = 0\}$$

in the case of linear elements; for full detail see [10,21].

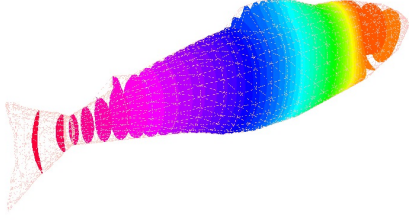
Below, we give an example of an elaborate mesh for a 3d fish and the Poisson problem; the FreeFem++ code is below and the results are on Figures 1&2

```
load "msh3" load "medit" load "tetgen"
mesh3 Ths("Y5177_Fish_cut.mesh"); // read skin fish mesh ..
real hh = 10; // the final mesh size
real[int] domaine = [0,0,0,1,hh^3/6.];
mesh3 Th=tetg(Ths,switch="pqaAYY",regionlist=domaine);
fespace Xh(Th,P2);
real x0=-200,y0=0,z0=0;
func ball = sqrt((x-x0)^2+(y-y0)^2+(z-z0)^2) <30;
macro Grad(u) [dx(u),dy(u),dz(u)] // EOM
Xh p,q;
solve laplace(p,q,solver=CG) =
  int3d(Th) ( ball*p*q+Grad(p)'*Grad(q) )
  - int3d(Th) ( 1*q ) ;
plot(p,fill=1,wait=1,value=0,nbiso=50); // see figure 1
```

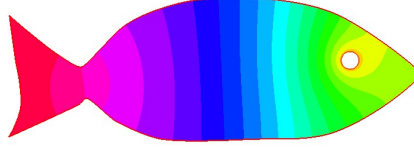
The skin mesh of the 3d fish come from the url\*. And the graphics solution of the problem is 2d and 3d is display figure 2 and 1 respectively.

---

\*[http://www-roc.inria.fr/gamma/download/counter.php?dir=FISH/&get\\_obj=Y5177\\_Fish\\_cut.mesh&acce=Y5177\\_Fish](http://www-roc.inria.fr/gamma/download/counter.php?dir=FISH/&get_obj=Y5177_Fish_cut.mesh&acce=Y5177_Fish)



**Figure 1.** 3d Poisson solution iso-surface



**Figure 2.** 2d Poisson solution iso-value

### 3. Mesh adaptation

In `FreeFem++` a lot of adaptation tools are implemented; this corresponding to the work of many people [20,17,26,25,32,19,29,9,6].

All these tools are based a Delaunay-Voronoi algorithm with a distance (or Metric) between two points  $q^1, q^2$  given by  $\sqrt{{}^t(q^1 - q^2) \mathcal{M} (q^1 - q^2)}$  where the matrix  $\mathcal{M}$  is symmetric positive definite matrix field defined on the mesh  $\mathcal{T}_h$ . Consequently the length  $\ell_{\mathcal{M}}$  of a curve  $\gamma$  for  $]0, 1[ \in \mathbb{R}^d$  with respect to  $\mathcal{M}$  is

$$\ell_{\mathcal{M}} = \int_0^1 \sqrt{{}^t \gamma'(t) \mathcal{M}(\gamma(t)) \gamma'(t)} dt \quad (3.1)$$

The computation of  $\mathcal{M}$  can be implicit or explicit; the tools to compute  $\mathcal{M}$ , in isotropic cases are scalar fields  $h$  which represent the local mesh size such that  $\mathcal{M} = \frac{1}{h^2} I_d$ , where  $I_d$  is the identity matrix.

The idea is to build a meshes with edges of equal length with respect of  $\mathcal{M}$ , i.e. the length with respect to  $\mathcal{M}$  of all the mesh edges should be close to one so as to get an equi-repartition of the error close to  $\text{err}$  in norm  $L^\infty$ . So for  $P_1$  continuous finites elements, The metric can be defined by:

$$\ell_{\mathcal{M}} = \frac{1}{\text{err}} |\partial_h^2 u_h| \quad (3.2)$$

where  $|\partial_h^2 u_h| = \sqrt{(\partial_h^2 u_h)^2}$  and  $\partial_h^2 u$  is an approximation of the Hessian matrix of  $u_h$ . `FreeFem++` does it automatically by default with `adaptmesh` function (2d), `MetricPk` (2d), or with `mshmet` (2d).

An example of  $L^\infty$  error mesh adaptation with metric for the Poisson problem in an L-shape domain  $\Omega = ]0, 1[^2 \setminus ]1/2, 1[^2$ . Find  $u \in H^1(\Omega)$  such that

$$-\Delta u = (x - y) \text{ in } \Omega, \quad \frac{\partial u}{\partial \mathbf{n}} = 0 \text{ on } \partial\Omega; \quad \int_{\Omega} u = 0 \quad (3.3)$$

we add do a small stabilization term  $\varepsilon u$  to remove the constraint  $\int_{\Omega} u$ , so the weak form is : Find  $u_{\varepsilon} \in H^1(\Omega)$  such that

$$\forall v \in H^1(\Omega), \quad \int_{\Omega} u_{\varepsilon} v + \varepsilon u_{\varepsilon} v - \int_{\Omega} (x-y)v = 0 \quad (3.4)$$

You can easily prove that  $\|u_{\varepsilon} - u\|_{H^1} < \varepsilon C$ , the discretization with  $P_1$  Lagrange finite element and  $\varepsilon = 10^{-10}$ , is:

```
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=0.01;
problem Poisson(u,v,solver=CG,eps=1.0e-6) =
  int2d(Th,qforder=2) ( u*v*1.0e-10+ dx(u)*dx(v) + dy(u)*
    dy(v) )
+ int2d(Th,qforder=2) ( (x-y)*v );
for (int i=0;i< 4;i++)
{
  Poisson;
  plot(u,wait=1);
  Th=adaptmesh(Th,u,err=error);
  plot(Th,wait=1);
  u=u;// reinterpolation of u on new mesh Th.
  error = error/2;
} ;
```

The final mesh adaption and associated solution will be show in figure 3 and 4

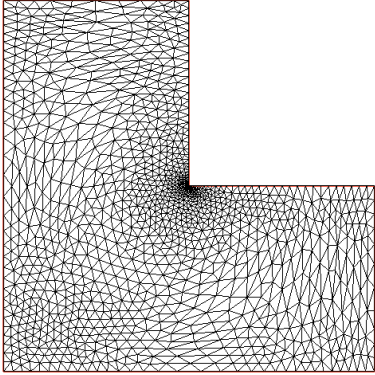


Figure 3. Mesh

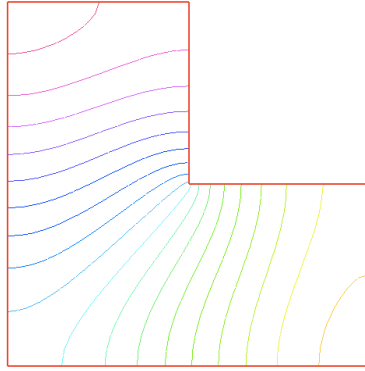
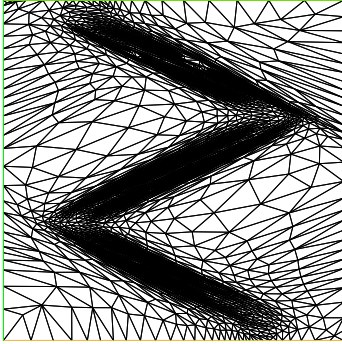


Figure 4. Solution

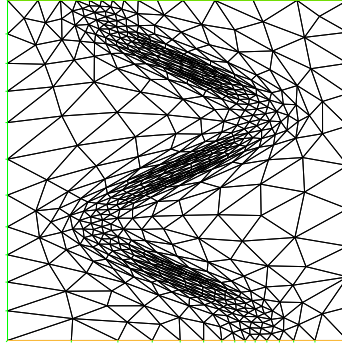
With the `MetricPk` plugin of J-M Mirebeau we can build a metric in  $\mathbb{R}^2$  for the lagrangian triangular continuous approximation of degree  $k$ ,  $P_k$ , with respect to the norm  $W^{r,p}$  with  $r = 0$  or  $1$ ; the output is an adapted mesh with  $N_t$  element

with a metric which can be optimal or sub optimal; the lost of optimality is due to constraint on acute angle in triangulation in case of  $r = 1$ . ( see [27,26] or J-M Mirebeau these details) :

```
load "MetricPk"
real Nt = 1000, r = 1, k = 3, p = 1; // def of the parameter
mesh Th = square(20, 20, [2*x-1, 2*y-1]);
func f = x^2*y + y^3 + tanh(5*(-2*x + sin(5*y)));
fespace Metric(Th, [P1, P1, P1]);
Metric [m11, m12, m22];
for(int i=0; i<5; i++){
  plot(Th, wait=true); // see figure 6 and 5
  [m11, m12, m22] = [0, 0, 0]; // resize metric array
  m11[] = MetricPk(Th, f, kDeg=k, rDeg=r, pExp=p, mass=Nt
    /Z,
    TriangulationType=0); // 0 sub optimal
                          // 1 Optimal (=>no acute angle)
  Th = adaptmesh(Th, m11, m12, m22, IsMetric=true); }
```



**Figure 5.** Optimal metric in norm  $W^{1,1}$  for  $10^3$  triangle but it is wrong mesh because the acute constraint is miss



**Figure 6.** Sub Optimal but not acute constraint for  $10^3$  triangle

In 3d we can use meshmet a plugin by J. Morice for the mshmet library of P. Frey [16]. To compute an isotropic adapted metric on can use tetgen, another plugin documented in [30]. The following listing is a full test of this idea.

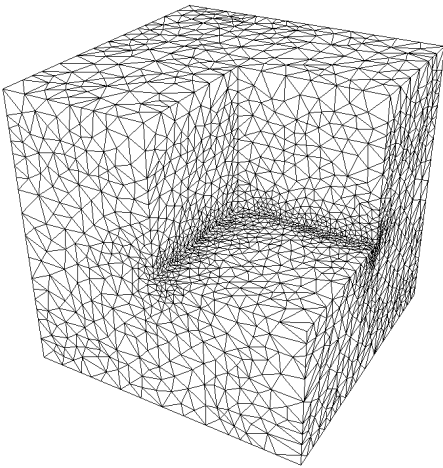
```
load "msh3" load "tetgen" load "mshmet" load "medit"
int n = 6;
int[int] l1=[1,1,1,1], l01=[0,1], l11=[1,1];
// label numbering for boundary condition
mesh3 Th3=buildlayers(square(n,n,region=0,label=l1),
  n, zbound=[0,1], labelmid=l11, labelup = l01,
  labeldown = l01);
Th3 = trunc(Th3, (x<0.5) | (y < 0.5) | (z < 0.5),
```

```

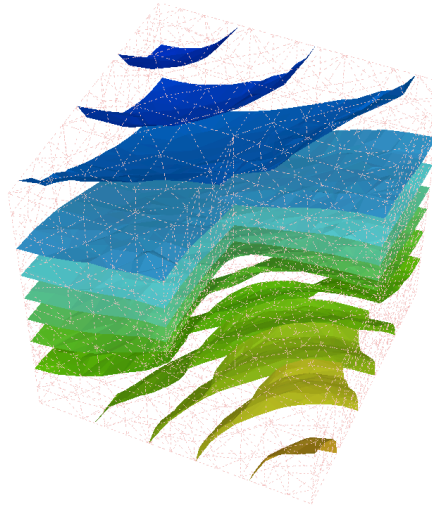
        label=1); // remove the ]0.5,1[3 cube
//end of build initial mesh
fespace Vh(Th3,P1);
Vh u,v,usol;
macro Grad(u) [dx(u),dy(u),dz(u)] // EOM
solve Poisson(u,v,solver=CG) = int3d(Th3) ( 1e-6*u*v + Grad(u)
        )'*Grad(v) )
        -int3d(Th3) ( (z-y/2-x/2)*v ) ;
real errm=1e-2; // level of error
for(int ii=0; ii<5; ii++) {
    Vh h ; h[]=mshmet(Th3,u,normalization=1,aniso=0,nbregul
        =1,
        hmin=1e-3,hmax=0.3,err=errm);
    errm*= 0.6; // change the level of error
    Th3=tetgreconstruction(Th3,switch="raAQ",sizeofvolume=h*
        h*h/6.);
    Poisson;
    plot(u,wait=1,nbiso=15); } // see figure 8

medit("U3",Th3,u); // see figure 7

```



**Figure 7.** Three 3d adapted mesh



**Figure 8.** iso surface

Finally, tools like `splitmesh` (only in 2d) can also be used; it splits triangle uniformly into sub triangles. In 3d the plugin `mmg3d-v4` moves 3d meshes or build 3d anisotropic meshes inside each element (see [15]).

#### 4. Phase change with Natural Convection

This example illustrates 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  $\vartheta_r$  (resp  $\vartheta_l$ ) at the right and left boundaries. The starting point of the problem is Brainstorming session (part I) of the third FreeFem++ days in december , 2012, this is almost the Orange Problem is describe in web page,<sup>†</sup> we present the validation part with paper [35].

The main interest of this example is to show the capability of FreeFem++ to solve complex problem with different kind of non-linearity. You can find other complex example in [5,7,23,1,8,12,11] for example, (Thank To. I. Danalia to the help in the modelization),

So the full model is: find  $\mathbf{u} = (u_1, u_2)$ , the velocity field,  $p$  the pressure field and  $\vartheta$  the temperature field in domain  $\Omega$  such that

$$\begin{cases} \mathbf{u} \text{ given} & \text{in } \Omega_s \\ \partial_t \mathbf{u} + (\mathbf{u} \nabla) \mathbf{u} + \nabla \cdot \mu \nabla \mathbf{u} + \nabla p = -c_T \mathbf{e}_2 & \text{in } \Omega_f \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega_f \\ \partial_t \vartheta + (\mathbf{u} \nabla) \vartheta + \nabla \cdot k_T \nabla \vartheta = \partial_t S(T) & \text{in } \Omega \end{cases} \quad (4.1)$$

Where  $\Omega_f$  is the fluid domain and the solid domain is  $\Omega_s = \Omega \setminus \Omega_f$ . The enthalpy of the change of phase is given by the function  $S$ ;  $\mu$  is the relative viscosity,  $k_T$  the thermal diffusivity.

In  $\Omega_f = \{x \in \Omega; \vartheta > \vartheta_f\}$ , with  $\vartheta_m$  the melting temperature the solid has melt.

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

$$\mu = \begin{cases} \vartheta < \vartheta_f \sim 10^6 \\ \vartheta \geq \vartheta_m \sim \frac{1}{\text{Re}} \end{cases},$$

This removes movement in the solid phase, and here Re is the Reynolds number of the liquid phase.

The Stefan enthalpy  $S_c$  with defined by  $S_c(\vartheta) = H(\vartheta)/S_{th}$  where  $S_{th}$  is the stefan number, and  $H$  is the Heaviside function ( 0, if ( $\vartheta < 0$ ), 1 else). with use the following smooth the enthalpy:

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

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

---

<sup>†</sup><http://www.ljll.math.upmc.fr/~hecht/ftp/ff++days/2011/Orange-problem.pdf>



Remark, if we don't make mesh adaptation the Newton method do not converge, and if we use explicit method this diverge too, and finally if we implicit the dependance in  $\Omega_s$  the method also diverge. This is a really difficult problem.

The finite element space to approximate  $u_1, u_2, p, \vartheta$  is defined by

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

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

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

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

The Newton loop and the fixed point are implemented as follows

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

The linearized problem is

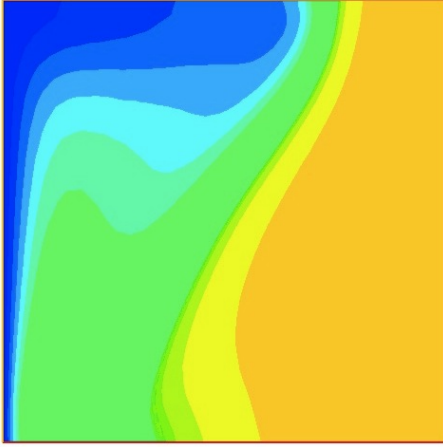
```
problem BoussinesqNL([ulw,u2w,pw,Tw],[v1,v2,q,TT])
= int2d(Th) (
  [ulw,u2w,Tw]' * [v1,v2,TT]*cdt
+ UgradV(ul,u2,ulw,u2w,Tw)' * [v1,v2,TT]
+ UgradV(ulw,u2w,ul,u2,T)' * [v1,v2,TT]
+ ( Grad(ulw,u2w)'*Grad(v1,v2)) * nu
+ ( Grad(ul,u2)'*Grad(v1,v2)) * dnu* Tw
+ cmT*Tw*v2
+ grad(Tw)'*grad(TT)*kT
- div(ulw,u2w)*q -div(v1,v2)*pw - eps*pw*q
+ dS(T)*Tw*TT*cdt
)
- int2d(Th) (
  [ul,u2,T]' * [v1,v2,TT]*cdt
+ UgradV(ul,u2,ul,u2,T)' * [v1,v2,TT]
+ ( Grad(ul,u2)'*Grad(v1,v2)) * nu
+ cmT*T*v2
+ grad(T)'*grad(TT)*kT
- div(ul,u2)*q -div(v1,v2)*p
- eps*p*q // stabilisation term
```

```

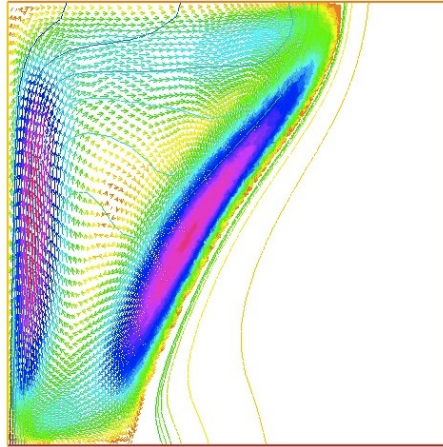
+ S(T)*TT*cdt
- [u1p,u2p,Tp]' * [v1,v2,TT]*cdt
- S(Tp)*cdt*TT
)
+ on(1,2,3,4, u1w=0,u2w=0) + on(2,Tw=0) + on(4,Tw=0) ;

```

The parameters of the computation are taken from [35] case 2.  $\vartheta_m = 0$ ,  $\text{Re} = 1$ ,  $S_{te} = 0.045$ ,  $P_r = 56.2$ ,  $R_a = 3.271 \times 10^5$ ,  $\vartheta_l = 1$ ,  $\vartheta_r = -0.1$  so in this case  $\text{cmT} = c_T = -R_a/P_r$ ,  $kT = k_T = 1/P_r$ ,  $\text{eps} = 10^{-6}$ , time step  $\delta t = 10^{-1}$ ,  $\text{cdt} = 1/\delta t$ , and with a good agreement this is figure 6 of [35] at time  $t = 80$ ; see in figures 9 and 10



**Figure 9.** Iso-value of the temperature at time  $t = 80$  of problem (4.1)



**Figure 10.** Velocity at time 80 of problem (4.1)

## 5. A Schwarz Domain Decomposition Example in Parallel

The following is an explanation of the scripts `DDM-Schwarz-*.edp` of the distribution. This is Schwarz domain decomposition in parallel with a complexity almost independent of the number of process (with a coarse grid preconditioner), thanks to F. Nataf for the all the discussion to implementation of this algorithm (see [28] for the theory).

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

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

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

Let  $(\pi_i)_{i=1,\dots,N_p}$  be a regular partition of unity of  $\Omega$ , i.e.:

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

Denote  $\Omega_i$  the sub domain which is the support of  $\pi_i$  function and also denote  $\Gamma_i$  the boundary of  $\Omega_i$ . The parallel Schwarz method is as follows.

Let  $\ell = 0$  be the iterator and  $u^0$  an initial guess respecting the boundary condition (i.e.  $u^0|_{\Gamma} = g$ ):

$$\forall i = 1.., N_p : -\Delta u_i^\ell = f, \text{ in } \Omega_i, \text{ and } u_i^\ell = u^\ell \text{ on } \Gamma_i \setminus \Gamma, u_i^\ell = g \text{ on } \Gamma_i \cap \Gamma \quad (5.1)$$

$$u^{\ell+1} = \sum_{i=1}^{N_p} \pi_i u_i^\ell \quad (5.2)$$

After discretization with the Lagrange finite element method, with a compatible mesh  $\mathcal{T}_{hi}$  for  $\Omega_i$ , there exists a global mesh  $\mathcal{T}_h$  such that  $\mathcal{T}_{hi}$  is included in  $\mathcal{T}_h$ . **Remark** We avoid using finite element spaces associated to the full domain  $\Omega$  because it is too expensive.

Now let us look at this academic example:

$$-\Delta u = 1, \text{ in } \Omega = ]0, 1[^3; \quad u = 0, \text{ on } \partial\Omega. \quad (5.3)$$

In the following test we use a GMRES version preconditioned by the Schwarz algorithm ( $P_s$ ) and with a coarse grid solver ( $P_c$ ) on a coarse mesh. To build a new preconditioner  $P$  from two preconditioner  $P_c$  and  $P_s$  we use the following approximation  $0 \sim I - AP^{-1} = (I - AP_c^{-1})(I - AP_s^{-1})$  and after a simple calculus we have  $P^{-1} = P_c^{-1} + P_s^{-1} - P_c^{-1}AP_s^{-1}$ .

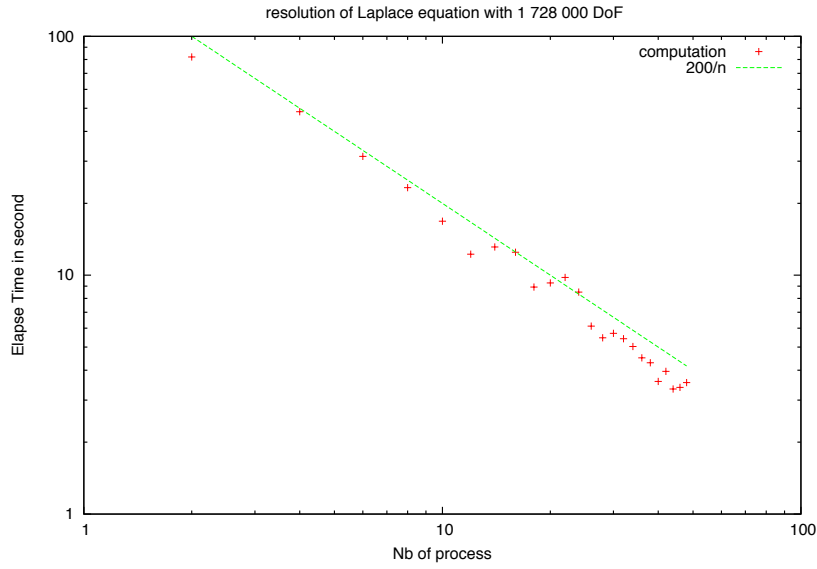
## 6. Conclusion

FreeFem++ is a continuously evolving software because it is easy to add new tools, finite elements. We have try to illustrate this with three example. In the future we expect to include more tools for try dimensional mesh anisotropic meshes adaptive, automatic different ion by operator overloding, seamless integration of parallel tools to free the user off low level programming.

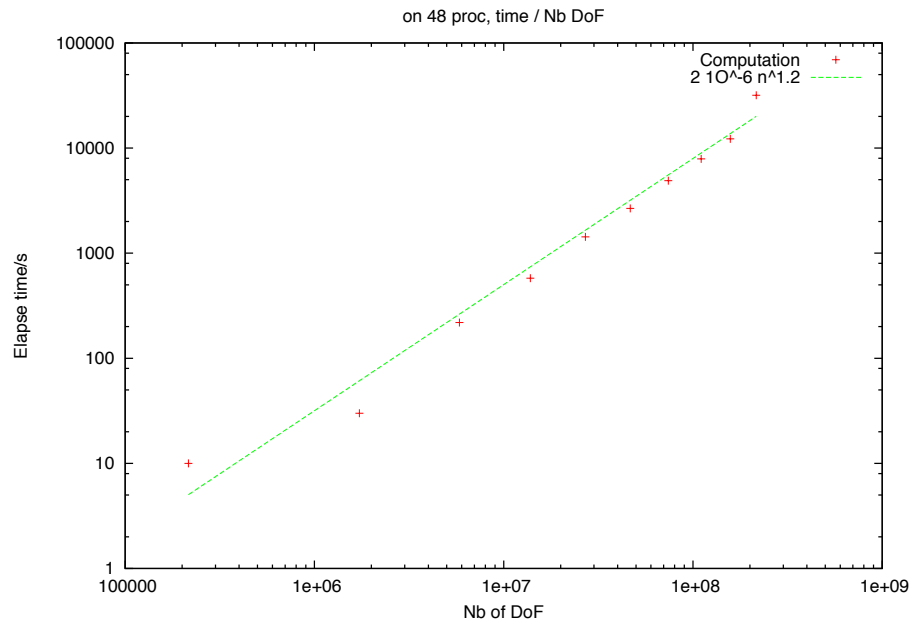
**Acknowledgements** Thanks to O. Pironneau for the fruitful discussion and the initialization for the freefem storie, to K. Ohtsuka to the help the writing of the documentation, to A Le Hyaric for the integrated version `FreeFem++-cs`, to J. Morice, and S. Auliac for the development respectively of the three dimensional meshes, and the optimization toolbox, and finally to E. Chacòn Vera for the freefem++ wiki<sup>‡</sup>

---

<sup>‡</sup><http://www.um.es/freefem/ff++/pmwiki.php>



**Figure 11.** resolution of Poisson equation with 1 728 000 DoF, scalability curve



**Figure 12.** On 48 proc, test on number of DoF from  $216 \cdot 10^3$  to  $216 \cdot 10^6$ , complexity curve

## References

1. Yves Achdou, Frédéric Hecht, and David Pommier. A posteriori error estimates for parabolic variational inequalities. *J. Sci. Comput.*, 37(3):336–366, 2008.
2. P. R. Amestoy, A. Guermouche, J.-Y. L’Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel Computing*, 32(2):136–156, 2006.
3. Z. Belhachmi and F. Hecht. Control of the effects of regularization on variational optic flow computations. *Journal of Mathematical Imaging and Vision*, pages 1–19, 2010. 10.1007/s10851-010-0239-x.
4. Christine Bernardi, Frédéric Hecht, and Fatma Zohra Nouri. A new finite-element discretization of the Stokes problem coupled with the Darcy equations. *IMA J. Numer. Anal.*, 30(1):61–93, 2010.
5. Christine Bernardi, Frédéric Hecht, and Rüdiger Verfürth. A finite element discretization of the three-dimensional Navier-Stokes equations with mixed boundary conditions. *M2AN Math. Model. Numer. Anal.*, 43(6):1185–1201, 2009.
6. Christine Bernardi, Frédéric Hecht, and Rüdiger Verfürth. A posteriori error analysis of the method of characteristics. *Math. Models Methods Appl. Sci.*, 21(6):1355–1376, 2011.
7. Christine Bernardi, Tomás Chacón Rebollo, Frédéric Hecht, and Roger Lewandowski. Automatic insertion of a turbulence model in the finite element discretization of the Navier-Stokes equations. *Math. Models Methods Appl. Sci.*, 19(7):1139–1183, 2009.
8. Christine Bernardi, Tomás Chacón Rebollo, Frédéric Hecht, and Zoubida Mghazli. Mortar finite element discretization of a model coupling Darcy and Stokes equations. *M2AN Math. Model. Numer. Anal.*, 42(3):375–410, 2008.
9. M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *Internat. J. Numer. Methods Fluids*, 25(4):475–491, 1997.
10. P. G. Ciarlet and J.-L. Lions, editors. *Handbook of numerical analysis. Vol. II.* Handbook of Numerical Analysis, II. North-Holland, Amsterdam, 1991. Finite element methods. Part 1.
11. P. G. Ciarlet and J. L. Lions, editors. *Handbook of numerical analysis. Vol. IX.* Handbook of Numerical Analysis, IX. North-Holland, Amsterdam, 2003. Numerical methods for fluids. Part 3.
12. Ionut Danaila and Frdric Hecht. A finite element method with mesh adaptivity for computing vortex states in fast-rotating bose-einstein condensates. *Journal of Computational Physics*, 229(19):6946 – 6960, 2010.
13. Timothy A. Davis. UMFPACK Version 5.4.0 User Guide. *Journal of Computing and Information Science in Engineering*, 2003.
14. Timothy A. Davis. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, June 2004.
15. Cecile Dobrzynski. MMG3D: User Guide. Rapport Technique RT-0422, INRIA, March 2012.
16. Pascal Frey. Error estimate for 2d and 3d unstructured meshes, 2010. on the web at <http://www.ann.jussieu.fr/frey/software.html>.
17. PL George and F. Hecht. *Handbook of Grid Generation*, chapter 20. CRC Press, 1999.
18. Vivette Girault and Frdric Hecht. Numerical methods for grade-two fluid models: Finite-element discretizations and algorithms. In R. Glowinski and J. Xu, editors, *Numerical Methods for Non-Newtonian Fluids*, volume 16 of *Handbook of Numerical Analysis*, pages 1 – 209. Elsevier, 2011.
19. W. G. Habashi, M. Fortin, J. Dompierre, M.-G. Vallet, and Y. Bourgault. Anisotropic mesh

- adaptation: a step towards a mesh-independent and user-independent CFD. In *Barriers and challenges in computational fluid dynamics (Hampton, VA, 1996)*, volume 6 of *ICASE/LaRC Interdiscip. Ser. Sci. Eng.*, pages 99–117. Kluwer Acad. Publ., Dordrecht, 1998.
20. F. Hecht. *BAMG: Bidimensional Anisotropic Mesh Generator*. INRIA, 1998. on the web at <http://www.freefem.org/ff++/ftp/freefem++doc.pdf>.
  21. F. Hecht. *FreeFem++, third edition, v 3.19-1*. Universite Pierre et Marie Curie, 2012. on the web at <http://www.freefem.org/ff++/ftp/freefem++doc.pdf>.
  22. Frédéric Hecht. C++ tools to construct our user-level language. *M2AN Math. Model. Numer. Anal.*, 36(5):809–836, 2002. Programming.
  23. Frédéric Hecht, Alexei Lozinski, Alain Perronnet, and Olivier Pironneau. Numerical zoom for multiscale problems with an application to flows through porous media. *Discrete Contin. Dyn. Syst.*, 23(1-2):265–280, 2009.
  24. Richard B. Lehoucq, Danny C. Sorensen, and C. Yang. *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods (Software, Environments, Tools)*. Soc for Industrial & Applied Math, 1997.
  25. A. Loseille, A. Dervieux, and F. Alauzet. Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations. *J. Comput. Phys.*, 229(8):2866–2897, 2010.
  26. Jean-Marie Mirebeau. Optimal meshes for finite elements of arbitrary order. *Constr. Approx.*, 32(2):339–383, 2010.
  27. Jean-Marie Mirebeau. Optimally adapted meshes for finite elements of arbitrary order and  $W^{1,p}$  norms. *Numer. Math.*, 120(2):271–305, 2012.
  28. F. Nataf. Optimized Schwarz methods. In *Domain decomposition methods in science and engineering XVIII*, volume 70 of *Lect. Notes Comput. Sci. Eng.*, pages 233–240. Springer, Berlin, 2009.
  29. O. Pironneau and F. Hecht. Mesh adaption for the Black and Scholes equations. *East-West J. Numer. Math.*, 8(1):25–35, 2000.
  30. Hang Si. TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. <http://tetgen.berlios.de/>.
  31. Hang Si. TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. <http://tetgen.berlios.de/>.
  32. M.-G. Vallet, C.-M. Manole, J. Dompierre, S. Dufour, and F. Guibault. Numerical comparison of some Hessian recovery techniques. *Internat. J. Numer. Methods Engrg.*, 72(8):987–1007, 2007.
  33. Andreas Wächter. Short tutorial: Getting started with ipopt in 90 minutes. In Uwe Naumann, Olaf Schenk, Horst D. Simon, and Sivan Toledo, editors, *Combinatorial Scientific Computing*, number 09061 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
  34. Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, May 2006.
  35. 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.