J. Numer. Math., Vol. 20, No. 3-4, pp. 251–265 (2012)
DOI 10.1515/jnum-2012-0013

New development in freefem++

F. HECHT*

Received July 2, 2012

Abstract — This is a short presentation of the freefem++ software. In Section 1, we recall most of the characteristics of the software, In Section 2, we recall how to to build the weak form of a partial differential equation (PDE) from the strong form. In the 3 last sections, we present different examples and tools to illustrated the power of the software. First we deal with mesh adaptation for problems in two and three dimension, second, we solve numerically a problem with phase change and natural convection, and the finally to show the possibilities for HPC we solve a Laplace equation by a Schwarz domain decomposition problem on parallel computer.

Keywords: finite element, mesh adaptation, Schwarz domain decomposition, parallel computing, freefem++

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

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

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*.

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, etc., but no quadrangles.
- Tools to define discontinuous Galerkin finite element formulations `P0`, `P1dc`, `P2dc` and keywords: `jump`, `mean`, `intalledges`.
- A large variety of linear direct and iterative solvers (LU, Cholesky, Crout, CG, GMRES, UMFPACK [13], MUMPS [2], SuperLU, etc.) and eigenvalue and eigenvector solvers (ARPARK) [24], and optimization tools like Ipopt [32].

- 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, as well as Schwarz's domain decomposition method, eigenvalue problem, residual error indicator, etc.
- A parallel version using MPI.

2. Weak formulation

For the first example consider a Poisson problem on a domain Ω with a partition of the boundary $\partial\Omega$ in Γ_2, Γ_e: Find u such that

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

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

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

Note that due to $\partial u/\partial \vec{n} = 0$ on Γ_2 and $v = 0$ the term $\int_\Gamma (\partial u/\partial n)v$ disappears on Γ_2. Finally the finite method simply replaces V_g with a finite element space, such as

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

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 Figs. 1 and 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.];
```

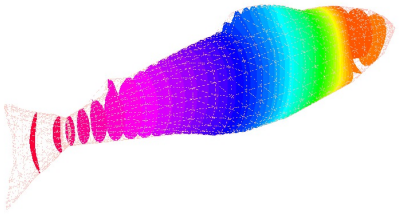

Figure 1. 3D Poisson solution iso-surface.

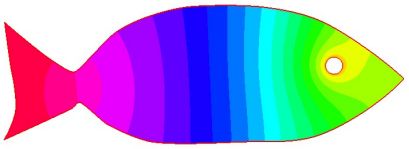

Figure 2. 2D Poisson solution iso-value.

```
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 Fig.1
```

The skin mesh of the 3D fish come from the URL http://www-roc.inria.fr/gamma/download/counter.php?dir=FISH&get_obj=Y5177_Fish_cut.mesh&acces=Y5177_Fish. The graphics solution of the problem in 2D and 3D are displayed in Figs. 2 and 1, respectively.

3. Mesh adaptation

In freefem++ a lot of adaptation tools are implemented; this corresponding to the work of many people [6, 9, 17, 19, 20, 25, 26, 29, 31].

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

$$\ell_{\mathscr{M}} = \int_0^1 \sqrt{{}^t\gamma'(t)\mathscr{M}(\gamma(t))\gamma'(t)}\,\mathrm{d}t. \tag{3.1}$$

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

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

$$\ell_{\mathscr{M}} = \frac{1}{\text{err}} |\partial_h^2 u_h| \tag{3.2}$$

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^∞ 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

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

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

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

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;
};
```

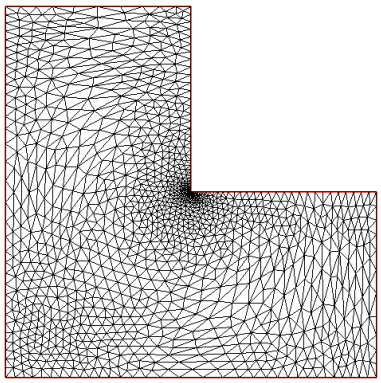

Figure 3. Mesh.

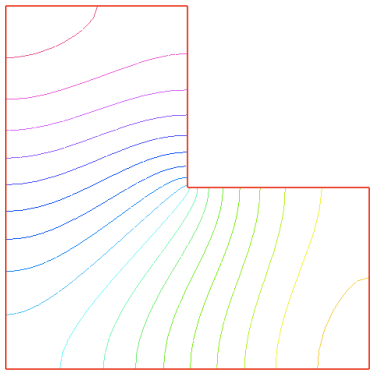

Figure 4. Solution.

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

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 Nt 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 [26, 27] 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 Figs. 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); }
```

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.

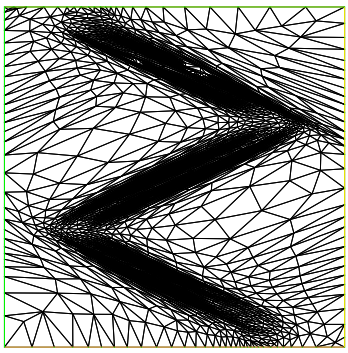

Figure 5. Optimal metric in norm $W^{1,1}$ for 10^3 triangle but it is wrong mesh because the acute constraint is miss.

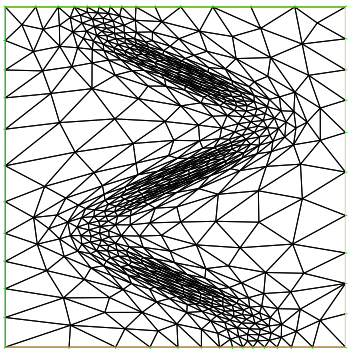

Figure 6. Suboptimal but not acute constraint for 10^3 triangle.

```
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 Fig.8

medit("U3",Th3,u); // see Fig.7
```

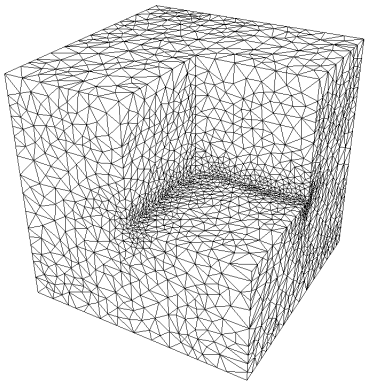

Figure 7. 3D adapted mesh.

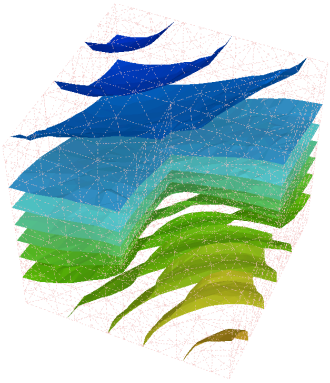

Figure 8. Iso surface.

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 ϑ_r (resp ϑ_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 described in URL http://www.ljll.math.upmc.fr/~hecht/ftp/ff++days/2011/Orange-problem.pdf, we present the validation part with paper [34].

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 [1, 5, 7, 8, 11, 12, 23] for example, (thanks to I. Danalia to the help in the modelization).

So the full model is: Find $\boldsymbol{u} = (u_1, u_2)$, the velocity field, p the pressure field and ϑ the temperature flied in domain Ω such that

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

Where Ω_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, μ is the relative vis-

cosity, k_T the thermal diffusivity.

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

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

$$\mu = \begin{cases} \vartheta < \vartheta_f \sim 10^6 \\ \vartheta \geqslant \vartheta_m \sim \dfrac{1}{\mathrm{Re}}. \end{cases}$$

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_{the} 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 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.

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 dependence in Ω_s the method also diverge. This is a really difficult problem.

The finite element space to approximate $u1, u2, p, \vartheta$ 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 fixed point on Ω_s
```

```
{ nu = nuT; // recompute the viscosity in Ω_s,Ω_f
  for (int niter=0; niter<20; ++niter) // newton loop
  { BoussinesqNL;
    err = u1w[].linfty;
    cout << niter << " err NL " << err << endl;
    u1[] -= u1w[];
    if(err < tolNewton) break; } // convergence...
}
```

The linearized problem is:

```
problem BoussinesqNL([u1w,u2w,pw,Tw],[v1,v2,q,TT])
= int2d(Th) (
     [u1w,u2w,Tw]'*[v1,v2,TT]*cdt
     + UgradV(u1,u2,u1w,u2w,Tw)' * [v1,v2,TT]
     + UgradV(u1w,u2w,u1,u2,T)' * [v1,v2,TT]
     + ( Grad(u1w,u2w)'*Grad(v1,v2)) * nu
     + ( Grad(u1,u2)'*Grad(v1,v2)) * dnu* Tw
     + cmT*Tw*v2
     + grad(Tw)'*grad(TT)*kT
     - div(u1w,u2w)*q -div(v1,v2)*pw - eps*pw*q
     + dS(T)*Tw*TT*cdt
     )
   - int2d(Th) (
     [u1,u2,T]'*[v1,v2,TT]*cdt
     + UgradV(u1,u2,u1,u2,T)' * [v1,v2,TT]
     + ( Grad(u1,u2)'*Grad(v1,v2)) * nu
     + cmT*T*v2
     +  grad(T)'*grad(TT)*kT
     - div(u1,u2)*q -div(v1,v2)*p
     - eps*p*q  // stabilization 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 [34] case 2, $\vartheta_m = 0$, $\mathrm{Re} = 1$, $S_{te} = 0.045$, $P_r = 56.2$, $R_a = 3.27 \cdot 10^5$, $\vartheta_l = 1$, $\vartheta_r = -0.1$ so in this case $\mathrm{cmT} = c_T = -R_a/P_r$, $\mathrm{kT} = k_T = 1/P_r$, $\mathrm{eps} = 10^{-6}$, time step $\delta t = 10^{-1}$, $\mathrm{cdt} = 1/\delta t$, and with a good agreement this is Fig. 6 of [34] at time $t = 80$ (see Figs. 9 and 10).

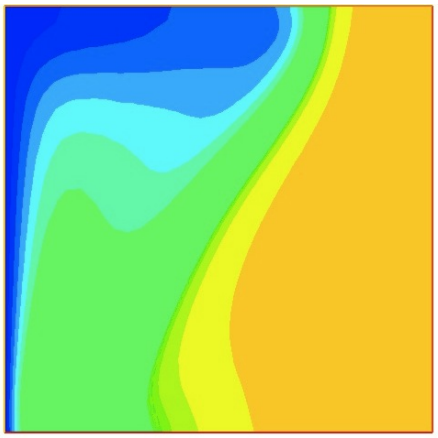

Figure 9. Iso-value of the temperature at time $t = 80$ of problem (4.1).

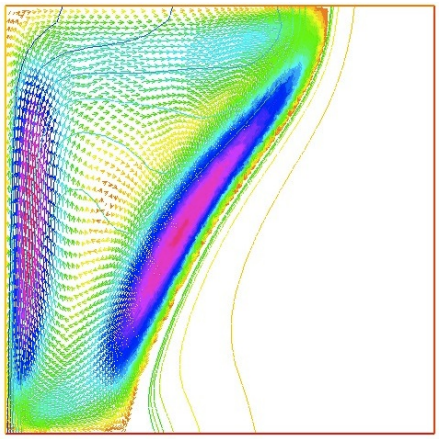

Figure 10. Velocity at time 80 of problem (4.1).

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 Ω with boundary Γ in $L^2(\Omega)$:

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

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

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

$$\pi_i \in \mathscr{C}^0(\Omega): \quad \pi_i \geqslant 0, \quad \sum_{i=1}^{N_p} \pi_i = 1.$$

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 as follows.

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

$$-\Delta u_i^\ell = f \quad \text{in } \Omega_i, \quad u_i^\ell = u^\ell \quad \text{on } \Gamma_i \setminus \Gamma, \quad u_i^\ell = g \quad \text{on } \Gamma_i \cap \Gamma \tag{5.1}$$

$$u^{\ell+1} = \textstyle\sum_{i=1}^{N_p} \pi_i u_i^\ell \qquad \forall\, i = 1, \ldots, N_p. \tag{5.2}$$

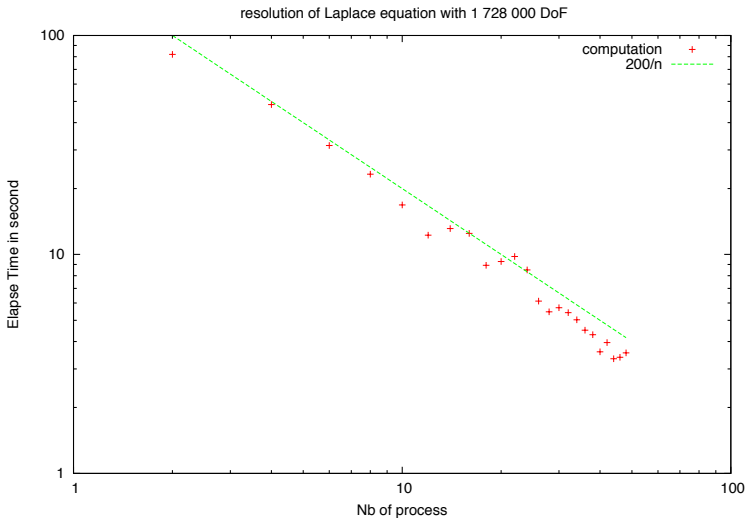

Figure 11. Resolution of Poisson equation with 1 728 000 DoF, scalability curve.

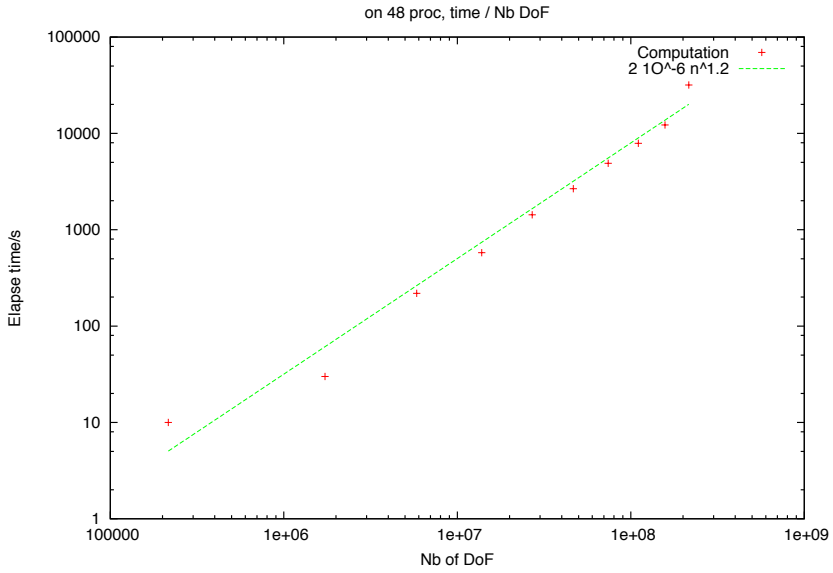

Figure 12. On 48 processors, test on number of DoF from $216\,10^3$ to $216\,10^6$, complexity curve.

After discretization with the Lagrange finite element method, with a compatible mesh $\mathscr{T}_{hi}$ for Ω_i, there exists a global mesh $\mathscr{T}_h$ such that $\mathscr{T}_{hi}$ is included in $\mathscr{T}_h$.

Remark 5.1. We avoid using finite element spaces associated to the full domain Ω because it is too expensive.

Now let us look at this academic example:

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

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 lP 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 tried to illustrate this with three examples. In the future we expect to include more tools for three dimensional anisotropic meshes adaptive, automatic differentiation by operator overloading, seamless integration of parallel tools to free the user off low level programming.

Acknowledgments. Our thanks to Olivier Pironneau for the fruitful discussions and the initialization of freefem in the nineties, to Kohji Ohtsuka for helping with the documentation, to Antoine Le Hyaric for the integrated version freefem++-cs, to Jacques Morice, and Sylvain Auliac for the development respectively of the three dimensional meshes, and the optimization toolbox, and finally to Eliseo Chacòn Vera for the freefem++ wiki (see URL http://www.um.es/freefem/ff++/pmwiki.php).

References

1. Y. Achdou, F. Hecht, and D. Pommier, A posteriori error estimates for parabolic variational inequalities. *J. Sci. Comput.* (2008) **37**, No. 3, 336 – 366.
2. P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet, Hybrid scheduling for the parallel solution of linear systems. *Parallel Comput.* (2006) **32**, No. 2, 136 – 156.
3. Z. Belhachmi and F. Hecht, Control of the effects of regularization on variational optic flow computations. *J. Math. Imag. Vision* (2010) 1 – 19. 10.1007/s10851-010-0239-x.

4. C. Bernardi, F. Hecht, and F. Z. Nouri, A new finite-element discretization of the Stokes problem coupled with the Darcy equations. *IMA J. Numer. Anal.* (2010) **30**, No. 1, 61 – 93.

5. C. Bernardi, F. Hecht, and R. Verfürth, A finite element discretization of the three-dimensional Navier–Stokes equations with mixed boundary conditions. *M2AN Math. Model. Numer. Anal.* (2009) **43**, No. 6, 1185 – 1201.

6. C. Bernardi, F. Hecht, and R. Verfürth, A posteriori error analysis of the method of characteristics. *Math. Models Methods Appl. Sci.* (2011) **21**, No. 6, 1355 – 1376.

7. C. Bernardi, T. C. Rebollo, F. Hecht, and R. Lewandowski, Automatic insertion of a turbulence model in the finite element discretization of the Navier–Stokes equations. *Math. Models Methods Appl. Sci.* (2009) **19**, No. 7, 1139 – 1183.

8. C. Bernardi, T. C. Rebollo, F. Hecht, and Z. Mghazli, Mortar finite element discretization of a model coupling Darcy and Stokes equations. *M2AN Math. Model. Numer. Anal.* (2008) **42**, No. 3, 375 – 410.

9. M. J. Castro-D´ıaz, F. Hecht, B. Mohammadi, and O. Pironneau, Anisotropic unstructured mesh adaption for flow simulations. *Int. J. Numer. Methods Fluids* (1997) **25**, No. 4, 475 – 491.

10. P. G. Ciarlet and J.-L. Lions, editors, *Handbook of Numerical Analysis, Vol. II. Finite Element Methods, Part 1*. North-Holland, Amsterdam, 1991.

11. P. G. Ciarlet and J.-L. Lions, editors, *Handbook of Numerical Analysis, Vol. IX. Numerical Methods for Fluids, Part 3*. North-Holland, Amsterdam, 2003.

12. I. Danaila and F. Hecht, A finite element method with mesh adaptivity for computing vortex states in fast-rotating Bose–Einstein condensates. *J. Comp. Phys.* (2010) **229** No. 19, 6946 – 6960.

13. T. A. Davis, UMFPACK version 5.4.0 user guide. *J. Comput. Inform. Sci. Engrg.*, 2003.

14. T. A. Davis, Algorithm 832: Umfpack v4.3 – an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software* (2004), **30**, No. 2, 196 – 199.

15. C. Dobrzynski, MMG3D: User Guide. *Rapport Tech. RT-0422*, INRIA, 2012.

16. P. Frey, Error estimate for 2D and 3D unstructured meshes, 2010. URL http://www.ann.jussieu.fr/frey/software.html.

17. P. L. George and F. Hecht, *Handbook of Grid Generation*, Chapter 20. CRC Press, 1999.

18. V. Girault and F. Hecht, Numerical methods for grade-two fluid models: Finite-element discretizations and algorithms. In: *Numerical Methods for Non-Newtonian Fluids, Vol. 16*. (Eds. R. Glowinski and J. Xu), *Handbook of Numerical Analysis*, Elsevier, 2011, pp. 1 – 209.

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*, *ICASE/LaRC Interdiscip. Ser. Sci. Engrg., Vol. 6*. Kluwer Acad. Publ., Dordrecht, 1998, pp. 99 – 117.

20. F. Hecht, *BAMG: Bidimensional Anisotropic Mesh Generator*. INRIA, 1998. http://www.freefem.org/ff++/ftp/freefem++doc.pdf.

21. F. Hecht, *Freefem++, v.3.19-1*. Universite Pierre et Marie Curie, 2012. http://www.freefem.org/ff++/ftp/freefem++doc.pdf.

22. F. Hecht, C++ tools to construct our user-level language. *M2AN Math. Model. Numer. Anal.* (2002) **36**, No. 5, 809 – 836.

23. F. Hecht, A. Lozinski, A. Perronnet, and O. Pironneau. Numerical zoom for multiscale problems with an application to flows through porous media. *Discrete Contin. Dyn. Syst.* (2009) **23**, No. 1-2, 265 – 280.

24. R. B. Lehoucq, D. C. Sorensen, and C. Yang, *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restorted 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.* (2010) **229**, No. 8, 2866 – 2897.

26. J.-M. Mirebeau, Optimal meshes for finite elements of arbitrary order. *Constr. Approx.* (2010) **32**, No. 2, 339 – 383.

27. J.-M. Mirebeau, Optimally adapted meshes for finite elements of arbitrary order and $W^{1,p}$ norms. *Numer. Math.* (2012) **120**, No. 2, 271 – 305.

28. F. Nataf, Optimized Schwarz methods. In: *Domain Decomposition Methods in Science and Engineering, XVIII*, *Lect. Notes Comput. Sci. Eng., Vol. 70*. Springer, Berlin, 2009, pp. 233 – 240.

29. O. Pironneau and F. Hecht, Mesh adaption for the Black and Scholes equations. *East-West J. Numer. Math.* (2000) **8**, No. 1, 25 – 35.

30. H. Si, TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator. URL http://tetgen.berlios.de/.

31. M.-G. Vallet, C.-M. Manole, J. Dompierre, S. Dufour, and F. Guibault, Numerical comparison of some Hessian recovery techniques. *Int. J. Numer. Methods Engrg.* (2007) **72**, No. 8, 987 – 1007.

32. A. Wächter, Short tutorial: Getting started with Ipopt in 90 minutes. In: *Combinatorial Scientific Computing* (Eds. U. Naumann, O. Schenk, H. D. Simon, and S. Toledo) *Dagstuhl Seminar Proceedings, Vol. 09061*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2009.

33. A. Wächter and L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* (2006) **106**, No. 1, 25 – 57.

34. S. Wang, A. Faghri, and T. L. Bergman, A comprehensive numerical model for melting with natural convection. *Int. J. Heat Mass Transfer* (2010) **53**, No. 9-10, 1986 – 2000.