

Etat de l'art des recherches et développements de méthodes parallèles dans FreeFem

Frédéric Nataf

Laboratory J.L. Lions (LJLL), CNRS, Equipe Alpines (Inria and Sorbonne University)

with Xavier Claeys, Pierre Jolivet, Frédéric Hecht, Pierre Marchand and
Pierre-Henri Tournier

GdR Ondes – 24 Mars 2022

- 1 Take Home message
- 2 The FreeFem Domain Specific Language
- 3 Parallelism in FreeFem
- 4 Conclusion

- 1 Take Home message
- 2 The FreeFem Domain Specific Language
- 3 Parallelism in FreeFem
- 4 Conclusion

- Short and expressive scripts
- Closer to the math
- Beneficial to Multigrid methods
- Beneficial to Domain decomposition methods

- 1 Take Home message
- 2 The FreeFem Domain Specific Language
- 3 Parallelism in FreeFem
- 4 Conclusion

Domain Specific Language for finite element method

```
28 //Th=square(Nbnoeuds,Nbnoeuds,[x,y+,2*x]);
29 [Th=buildmesh(c1(Nbnoeuds)+c2(Nbnoeuds));
30 plot(Th,wait=1);
31
32 //Fonction de x et de y
33 func f = 1.+100.*sin(cos(x)*y);
34
35
36 //Definition de l'espace des elements finis P1 associe
37 //au maillage Th
38 fespace Vh(Th,P1); //P1<->P2
39
40 Vh funcfh=f;
41 plot(funcfh,wait=1);
42
43 //uh et vh sont des elements de Vh
44 Vh uh,vh;
45
46 real alpha=0.;
47 //Definition du probleme variationnel
48 problem chaleur(uh,vh)=
49   int2d(Th)((dx(uh)*dx(vh)+dy(uh)*dy(vh)))+int1d(Th,2)(alpha*uh*vh)
50   -int2d(Th)(f*vh)
51   +on(1,uh=0)
52 ;
53
54 // Résolution du problème et Adaptations successives du maillage
55 int nlevels=6;
56 for(int i=0;i<nlevels;++i)
57 {
58   // adaptation du maillage à la solution approchée uh
59   if(i!=0){ Th=adaptmesh(Th,uh);
60   plot(Th,wait=1);
61   }
62   //Resolution du probleme
63   real cpu=clock();
64   chaleur;
65
66   cout << "temps de resolution du probleme "<< i << ": " << clock()-cpu << " secondes " << endl;
67
68   //On affiche le resultat
69   plot(uh,wait=1,fill=1,value=1,dim=3 );
70
71 }
72
73
```

Line: 29 freefem ⚡ Tab Size: 4 ↴ ⚡

Why use a DS(E)L (FreeFem++, Feel++, Dune, Fenics or Firedrake) instead of C/C++/Fortran/.. ?

- performances close to low-level language implementation,
- hard to beat something as simple as:

```
varf a(u, v) = int3d(mesh)([dx(u), dy(u), dz(u)]' * [dx(v), dy(v), dz(v)])  
    - int3d(mesh)(f * v) + on(boundary_mesh)(u = 0) ,
```

- access to the variational formulation is then natural and it helps.

A few facts

- 1987: First version by O. Pironneau written in Pascal on Macintosh
- Since 1992: the main developer is Frédéric Hecht

Some FreeFem features

- Integrates many state of the art libraries.
- "Natively" multiphysics (e.g. Modeling of SAW transducers via Maxwell + Elasticity)
- Automatic Mesh refinement native in 2d and via the plugin "Mmg3d" (Frey at al.) in 3D
- Interpolate between different finite element spaces defined on different meshes, clouds of points to mesh
- Extensible via dynamic plugins
- Interface to MPI
- parallel version runs on Linux, Windows, Mac since 2017
- Docker on Qarnot and Rescale cloud computing platform
- Web browser (Javascript port thanks to A. Le Hyaric (LJLL))

demo avec chaleurEllipse.edp

Outline

- 1 Take Home message
- 2 The FreeFem Domain Specific Language
- 3 Parallelism in FreeFem
- 4 Conclusion

Data Distribution for parallel computing

Domain Decomposition via Metis or Scotch interface

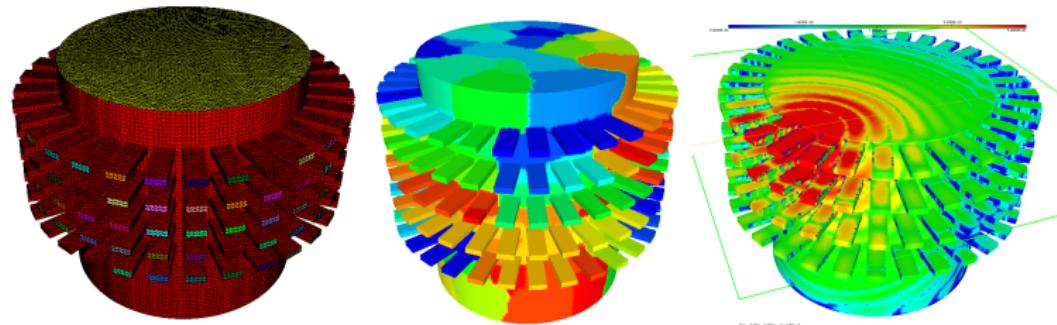


Figure: Electromagnetic chamber

Overlap is done by FreeFem based on the mesh connectivity

Overlap helps handling cross points

Strong Scalability test for 3D Maxwell

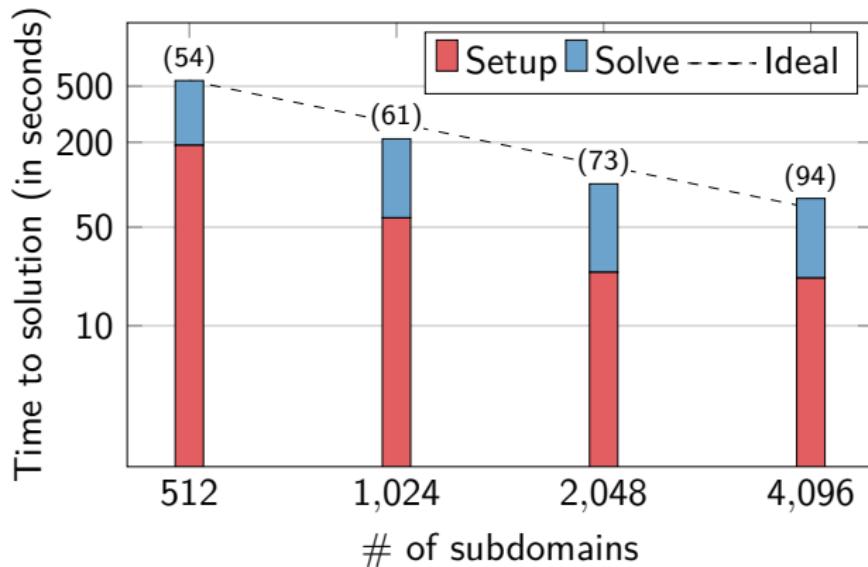


Figure: Maxwell 3D with edge elements of degree 2 - 119M d.o.f.

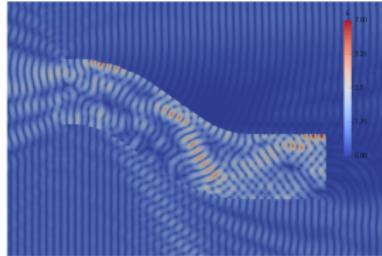
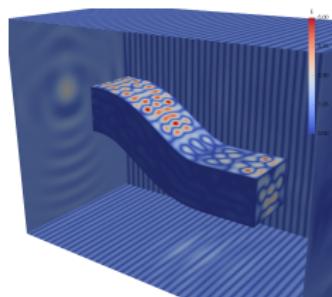
Maxwell's equations – Cobra test case in FreeFem++

Bonazzoli, Dolean, Graham, Spence, Tournier, 2018.

order 2 edge elements (Nedelec), 10 pts per wavelength

$$f = 10 \text{ GHz}: n \approx 1.07 \times 10^8 \quad f = 16 \text{ GHz}: n \approx 1.98 \times 10^8$$

f	N_{sub}	# it	inner it	Total	Setup	GMRES	inner
10GHz	1536	32	1527	515.8	383.2	132.6	61.8
10GHz	3072	33	2083	285.0	201.6	83.4	40.6
16GHz	3072	43	3610	549.2	336.8	212.4	118.6
16GHz	6144	46	4744	363.0	210.5	152.5	96.8



Helmholtz equations – overthrust 3D

5 points per wave length, P2 FE Simulations réalisées sur Occigen (CINES) noeuds Haswell

		cartesian mesh			adaptive mesh		
f	# cores	# dofs	# it	sec.	# dofs	# it	sec.
5	384	22 M	167	58	11 M	125	25
10	3072	176 M	340	121	85 M	253	59
20	12288	—	—	—	678 M	438	218

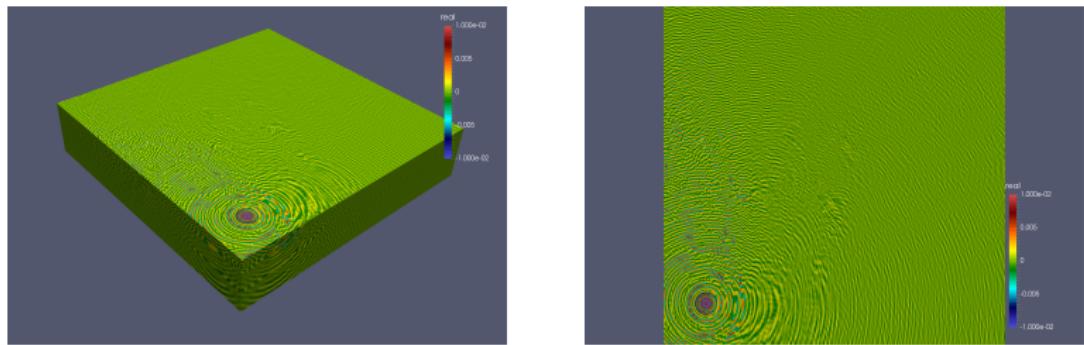


Figure: Simulations with FreeFem++ (P.H. Tournier)

Radiative transfer problem

$$(\mathbf{s} \cdot \nabla + \beta(\mathbf{x}))I(\mathbf{x}, \mathbf{s}) - \sigma_s(\mathbf{x}) \oint_{\mathcal{S}} I(\mathbf{x}, \mathbf{s}') \Phi(\mathbf{s}, \mathbf{s}') \, d\mathbf{s}' \\ - \kappa(\mathbf{x})I_b(\mathbf{x}) = 0 \quad \forall \mathbf{x} \in \Omega, \mathbf{s} \in \mathcal{S}$$

One billion unknowns in 60 seconds with 8192 MPI processes

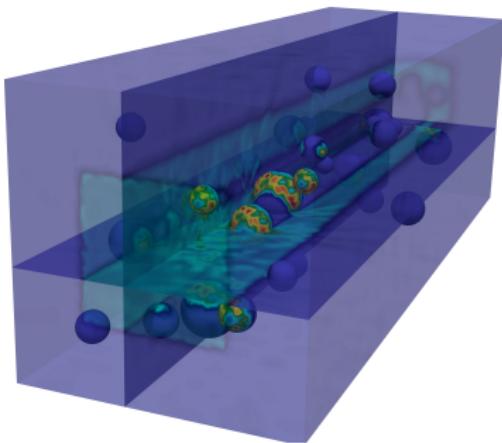


Figure: Badria, Jolivet, Rousseau, Le Corre, Digonnet and Favennec, 2018 – FreeFem++

Two approaches:

- PETSc Interface (P. Jolivet (IRIT) with some inputs from S. Zampini (KAUST))
- FFDDM (P.H. Tournier (LJLL) with some inputs from F.N.)

If existing PETSc solvers (Direct solvers e.g. MUMPS, Multigrid e.g. GAMG, Domain Decomposition methods (hpddm)) do the job = Good solution

Pros :

- Huge library with not only solvers but also time schemes, optimizers, ...

Cons :

- cannot handle Real and Complex value problems together
- Customising is tricky (PETSc library in C + MPI)

Pros :

- Handles both Real and Complex value problems at the same time
- Written in FreeFem language via prefixed MACROS
- Very good for experimenting with DD methods
- Parallel direct solver MUMPS is also provided

Cons :

- Multigrid is accessible only via the PETSc interface.
- No GPU access (but with the new ARM SoC (cf. A64FX from Fujitsu or m1 from Apple) this is maybe only a temporary issue)

Note that a FreeFem script can mix the PETSc interface with ffddm

ffddm documentation

Minimal example

```
1 macro dimension 3// EOM           // 2D or 3D
2
3 include "ffddm.idp"
4
5 load "msh3"
6
7 int[int] LL = [2,2, 1,2, 2,2];
8 mesh3 ThGlobal = cube(10, 10, 10, [x, y, z], label = LL);      // global mesh
9
10 macro grad(u) [dx(u), dy(u), dz(u)]// EOM    // three-dimensional gradient
11
12 macro Varf(varfName, meshName, VhName)
13     varf varfName(u,v) = int3d(meshName)(grad(u)'* grad(v)) + int3d(meshName)(v) + on
14 // EOM
15
16 // Domain decomposition
17 ffddmbuildDmesh( LapMesh , ThGlobal , mpiCommWorld )
18
19 macro def(i)// EOM                // scalar field definition
20 macro init(i)// EOM              // scalar field initialization
21 ffddmbuildDfespace( LapFE , LapMesh , real , def , init , P1 )
22
23 ffddmsetupOperator( Lap , LapFE , Varf )
24
25 real[int] rhsi(0);
26 ffddmbuildrhs( Lap , Varf , rhsi )
27
28 LapFEVhi def(ui);
29
30 //Direct solve
31 ui[] = Lapdirectsolve(rhsi);
32
33 Lapwritesummary
34
35 ffddmplot(LapFE,ui,"u");
```

- 1 Take Home message
- 2 The FreeFem Domain Specific Language
- 3 Parallelism in FreeFem
- 4 Conclusion

Where to find Informations

FreeFem website

<https://freefem.org/>

- Online documentation
- Sources on Github
- Liste de discussion
- FreeFem days
 - Introduction to parallel FreeFem by P. Jolivet and P.H. Tournier on Youtube <https://www.youtube.com/watch?v=-Aw2O46V2bo&feature=youtu.be>
 - Youtube channel FreeFem

Recap: Bonuses from DSL

- Short and expressive scripts
- Closer to the math
- Multigrid \Rightarrow faster algorithm
- Domain decomposition methods \Rightarrow faster algorithm

Not mentioned here

- Boundary Element Method and Domain Decomposition for BEM (Xavier Claeys, Pierre Marchand, P.H Tournier) along with matrix compression
- ParMmg parallel Mmg
- Paraview

Special thanks to GENCI for their support:
Developments have been developed, benchmarked and used
on national HPC resources at IDRIS and at CINES under
Allocations 20XX-067730.

THANK YOU
FOR YOUR ATTENTION!

Special thanks to GENCI for their support:
Developments have been developed, benchmarked and used
on national HPC resources at IDRIS and at CINES under
Allocations 20XX-067730.

THANK YOU
FOR YOUR ATTENTION!