FreeFEM: a high-level DSL for multi-physics simulations

Pierre-Henri Tournier

pierre-henri.tournier@sorbonne-universite.fr

Laboratoire Jacques-Louis Lions
Sorbonne Université - Campus Pierre et Marie Curie - Paris VI
INRIA Paris - équipe ALPINES

Overview

- open source, LGPL 3 https://github.com/FreeFem
- for Unix, Windows, macOS, also works on smarthpones and supercomputers
- written in C++ with distributed MPI parallelism
- a Domain-Specific Language for finite elements, intuitive and close to the maths, with a graphical interface
  $\Rightarrow$ aims to 'bring scientific computing to all' ; allows for easy prototyping, teaching, ... without sacrificing performance
- mature software (more than 30 years of development)
- online documentation https://doc.freefem.org
- stay in touch with the community:
  - monthly developer/user online meetup
  - yearly 'FreeFEM days' (3-day in-person event in December)
  - community forum https://community.freefem.org (30+ new topics/month)
  - FreeFEM youtube channel

The team

- Core team: F. Hecht, P. Jolivet, P.-H. Tournier
- S. Legrand (SED INRIA Paris): build system, CI/CD
- F. Nataf
- O. Pironneau
- A. Le Hyaric: FreeFEM-cs, FreeFEM-js
- P. Marchand (INRIA POEMS): BEM
- G. Sadaka (LMRS): FreeVol (finite volumes)
- + a few external contributors
- ERC Synergy *PSINumScat* (2025-2031) Phase-space-inspired Numerical Methods for High Frequency Wave Scattering, J. Galkowski (UCL), E. Spence (U. Bath), P.-H. Tournier:
  - 5 Research Engineers and 1 Postdoc at LJLL throughout the project
  - A. Chabib: Research Engineer, since October 2025
  - P.-L. Bacq: Research Engineer, since June 2026

Main features

- solve PDEs in 2D, 3D, on curves or on surfaces
- anisotropic mesh adaptation
- interfaced with most sequential and parallel linear solvers
- optimization (IPopt), eigenvalue problems (Arpack, SLEPc)
- time dependent geometry / meshes
- coupling problems (multiphysics, FEM-BEM)
- 600+ examples

First script

Find u solution of

$$-\Delta u = 1 \text{ in } \Omega, \quad u = 0 \text{ on } \Gamma_D, \quad \frac{\partial u}{\partial n} = 0 \text{ on } \Gamma_N$$

First script

Variational form: find $u \in H_0^1(\Omega)$ such that

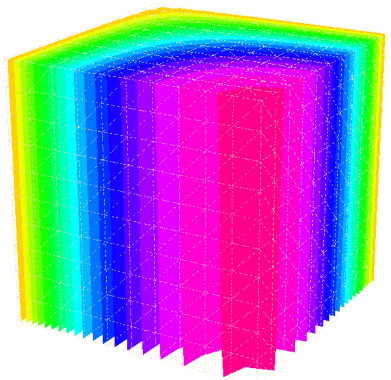
$$\int_\Omega \nabla u . \nabla v = \int_\Omega 1 v + \int_\Gamma \frac{\partial u}{\partial n} v, \quad \forall v \in H_0^1(\Omega)$$

First script

Variational form: find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u.\nabla v = \int_\Omega 1v + \int_\Gamma \frac{\partial u}{\partial n} v, \quad \forall v \in H_0^1(\Omega)$$

```
mesh3 Th = cube(10,10,10);

fespace Vh(Th,P1);  // P1 FE space
Vh uh,vh;           // Unknown and test function
func f=1;           // Right-hand side function
func g=0;           // Boundary condition function

macro Grad(u) [dx(u), dy(u), dz(u)] // EOM
// Define and solve the problem
solve laplace(uh, vh, solver=CG) =
int3d(Th)( Grad(uh) '* Grad(vh) )     // bilinear form
- int3d(Th)( f*vh )                   // linear form
+ on(1,4,uh=g);                       // boundary conditions

plot(uh, value=true, fill=true, nbiso=20);// see the result
```

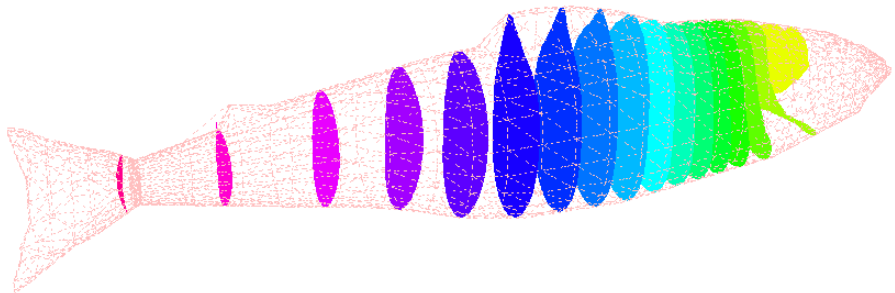


FreeFEM script files usually have the .edp extension and can be compiled and run with the command

```
FreeFem++ test.edp
```

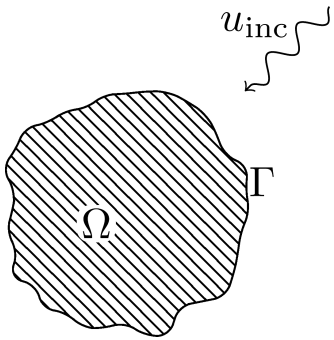
First script

```
load "gmsh"
mesh3 Th = gmshload3("poisson.msh");

fespace Vh(Th,P1);  // P1 FE space
Vh uh,vh;           // Unknown and test function
func f=1;           // Right-hand side function
func g=0;           // Boundary condition function

macro Grad(u) [dx(u), dy(u), dz(u)] // EOM
// Define and solve the problem
solve laplace(uh, vh, solver=CG) =
int3d(Th)( Grad(uh) '* Grad(vh) )    // bilinear form
- int3d(Th)( f*vh )                  // linear form
+ on(1,4,uh=g);                      // boundary conditions

plot(uh, value=true, fill=true, nbiso=20);// see the result
```



First script

Acoustic scattering of a plane wave by a 2D cavity



$$\begin{cases} -\Delta u - k^2 u = 0 & \text{in } \mathbb{R}^3 \backslash \Omega \\ u = -u_{\text{inc}} & \text{on } \Gamma \\ + \text{ radiation condition at infinity} \end{cases}$$

```
mesh Th = square(200,200);
Th = trunc(Th,max(abs(x-0.5),abs(y-0.5)) > 0.1
              || max((x-0.5),abs(y-0.5)) < 0.08,
        label=5);

func k = 40*pi;
func uinc = exp(1i*k*(cos(pi/6)*x+sin(pi/6)*y));

fespace Uh(Th,P2);
Uh<complex> u,v;

macro Grad(u) [dx(u),dy(u)] // EOM
solve Pb(u,v) = int2d(Th)(-k^2*u*v+Grad(u)'*Grad(v))
        - int1d(Th,1,2,3,4)(1i*k*u*v)
        + on(5, u=-uinc);

u = u + uinc;
plot(u,fill=1,value=1);
```

Solvers

```
mesh3 Th = cube(10,10,10);

fespace Vh(Th,P1);  // P1 FE space
Vh uh,vh;           // Unknown and test function
func f=1;           // Right-hand side function
func g=0;           // Boundary condition function

macro Grad(u) [dx(u), dy(u), dz(u)] // EOM
// Define and solve the problem
solve laplace(uh, vh, solver=CG) =
int3d(Th)( Grad(uh) '* Grad(vh) )    // bilinear form
- int3d(Th)( f*vh )                  // linear form
+ on(1,4,uh=g);                      // boundary conditions

plot(uh, value=true, fill=true, nbiso=20);  // see the result
```

Access to various linear solvers : LU, Cholesky, Crout, CG, GMRES, UMFPACK, SuperLU, MUMPS ...
Usually via loading the corresponding plugin :

```
load "MUMPS"

solve laplace(uh, vh, solver="MUMPS", sym=1) =
```

Access to underlying data structures and linear algebra

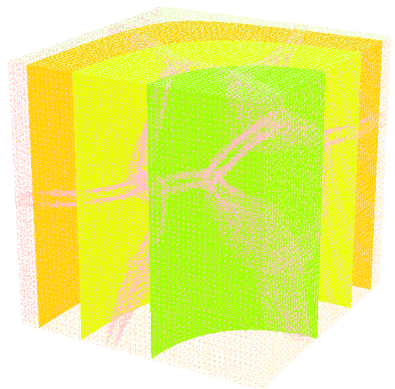
```
mesh3 Th = cube(10,10,10);

fespace Vh(Th,P1);  // P1 FE space
Vh uh,vh;           // Unknown and test function
func f=1;           // Right-hand side function
func g=0;           // Boundary condition function

macro Grad(u) [dx(u), dy(u), dz(u)] // EOM
// Define and solve the problem
varf laplace(u, v) =
int3d(Th)( Grad(u) '* Grad(v) )     // bilinear form
+ int3d(Th)( f*v )                  // linear form
+ on(1,4,u=g);                      // boundary conditions

matrix A = laplace(Vh,Vh, solver=CG);

real[int] b = laplace(0,Vh);

uh[] = A^-1*b;

plot(uh, value=true, fill=true, nbiso=20);  // see the result
```

compute algebraic L2 error :

```
real[int] err = A*uh[];
err -= b;
cout << "||A x - b|| / ||b|| = " << err.l2 / b.l2 << endl;
```

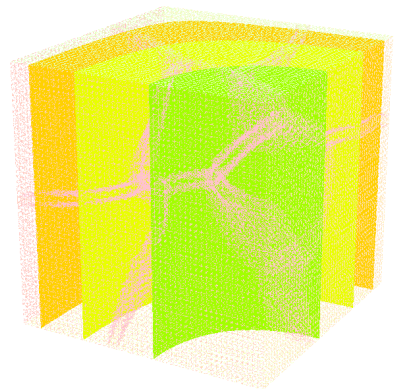
Parallel computing with PETSc

```
load "PETSc"
macro dimension()3// EOM
include "macro_ddm.idp"

mesh3 Th = cube(50,50,50);

func Pk = P1;
fespace Vh(Th,Pk);   // P1 FE space
Vh uh;               // Unknown function
func f=1;            // Right-hand side function
func g=0;            // Boundary condition function

macro Grad(u) [dx(u), dy(u), dz(u)] // EOM
// Define and solve the problem
varf laplace(u, v) =
int3d(Th)( Grad(u) '* Grad(v) )     // bilinear form
+ int3d(Th)( f*v )                  // linear form
+ on(1,4,u=g);                      // boundary conditions

Mat A;
MatCreate(Th, A, Pk);
A = laplace(Vh,Vh,sparams = "-pc_type gamg -ksp_monitor");

real[int] b = laplace(0,Vh);

uh[] = A^-1*b;

plotD(Th, uh, fill=true); // see the result
```



Parallel computing with PETSc

```
load "PETSc"
macro dimension()3// EOM
include "macro_ddm.idp"

mesh3 Th = cube(50,50,50);

func Pk = P1;
fespace Vh(Th,Pk);   // P1 FE space
Vh uh;               // Unknown function
func f=1;            // Right-hand side function
func g=0;            // Boundary condition function

macro Grad(u) [dx(u), dy(u), dz(u)] // EOM
// Define and solve the problem
varf laplace(u, v) =
int3d(Th)( Grad(u) '* Grad(v) )     // bilinear form
+ int3d(Th)( f*v )                  // linear form
+ on(1,4,u=g);                      // boundary conditions

Mat A;
MatCreate(Th, A, Pk);
A = laplace(Vh,Vh,sparams = "-pc_type gamg -ksp_monitor");

real[int] b = laplace(0,Vh);

uh[] = A^-1*b;
int[int] fforder = [1];
savevtk("lap.vtu", Th, uh, order = fforder);
```



Parallel computing with PETSc

Run it in parallel with :

```
ff-mpirun -np 8 test-PETSc.edp -ns -wg
```

Comprehensive PETSc / SLEPc interface thanks to P. Jolivet, including

- *SNESSolve* - nonlinear solvers
- *TSSolve* - timesteppers
- *TaoSolve* - optimizers
- *EPSSolve* - eigenvalue problems
- *NEPSolve* - nonlinear eigenvalue problems

FreeFEM + PETSc tutorial available at https://joliv.et/FreeFem-tutorial

C++ plugins

```
#include "ff++.hpp"
long CircumCenter(Fem2D::Mesh const *const &pTh, KN< double > *const &pcx,
                  KN< double > *const &pcy) {
  KN< double > &cx = *pcx;
  KN< double > &cy = *pcy;
  const Mesh &Th = *pTh;

  ffassert(Th.nt == cx.N( ));
  ffassert(Th.nt == cy.N( ));
  for (int k = 0; k < Th.nt; ++k) {
    const Mesh::Element &K = Th[k];
    R2 A = K[0], B = K[1], C = K[2];
    R2 a = (B + C) / 2, aa(a + R2(B, C).perp( ));
    R2 b = (A + C) / 2, bb(b + R2(A, C).perp( ));
    double ab = -det(a, aa, b);
    double abb = det(a, aa, bb);
    double s = ab + abb;
    R2 P = bb * ab / s + b * abb / s;
    cx[k] = P.x;
    cy[k] = P.y;
  }
  return 0L;
}

static void Load_Init( ) {
  Global.Add("CircumCenter", "(",
  new OneOperator3_< long, pmesh, KN<double>*, KN<double>* >(CircumCenter));
}
LOADFUNC(Load_Init)
```

C++ plugins

Compile the plugin into a dynamic library

```
ff-c++ -auto CircumCenter.cpp
```

that you can then use in your script with *load* :

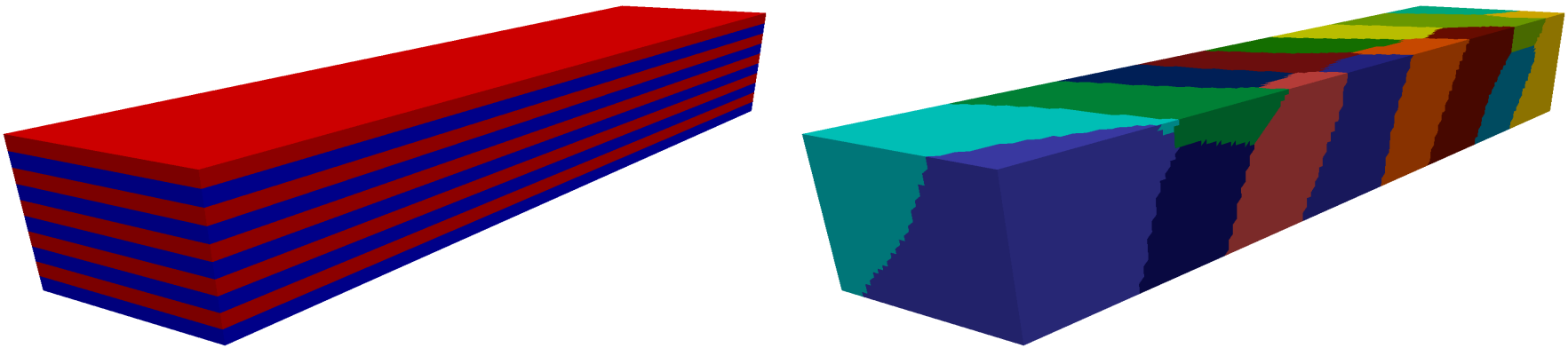
```
load "CircumCenter"

mesh Th = square(3,3);

fespace Ph(Th,P0);

Ph px,py;

CircumCenter(Th,px[],py[]);

cout << px[] << endl;
cout << py[] << endl;
```

Nearly incompressible elasticity

Robust GenEO Domain Decomposition preconditioner for saddle-point problems

Lowest order Taylor-Hood finite element $C0P2 - C0P1$ so that the pressure is continuous. In matrix form we have:

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_h \\ p_h \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_h \\ 0 \end{pmatrix}$$



Heterogeneous beam of rubber and steel. Coefficient distribution (left) and mesh partitioning by the automatic graph partitioner *Metis* (right)

Rubber is nearly incompressible $\nu_{rubber} = 0.4999$ and soft $E_{rubber} = 0.01GPa$ whereas steel is compressible $\nu_{steel} = 0.35$ and hard $E_{steel} = 200GPa$

Nearly incompressible elasticity

Weak and Strong scalability

#cores	n	$dim(V_0)$	$dim(\tilde{W}_0)$	setup(s)	#It	gmres(s)	total(s)	#It N_S^{-1}
262	15 987 380	5 383	3 319	710.7	24	631.6	1342.3	11
525	27 545 495	9 959	2 669	526.6	21	519.5	1046.1	12
1 050	64 982 431	17 837	4 587	675.2	22	665.9	1341.1	11
2 100	126 569 042	32 361	7 995	689.2	25	733.8	1423.0	10
4 200	218 337 384	59 704	13 912	593.0	27	705.4	1298.4	10
8 400	515 921 881	141 421	25 949	735.8	32	1152.5	1888.3	10
16 800	1 006 250 208	260 348	41 341	819.2	29	1717.9	2537.1	12

Weak scaling experiment

#cores	n	$dim(V_0)$	$dim(\tilde{W}_0)$	setup(s)	#It	gmres(s)	total(s)	#It N_S^{-1}
525	27 545 495	9 959	2 669	526.6	21	519.5	1046.1	12
1 050	27 545 495	15 078	4 082	265.7	21	224.7	490.4	11
2 100	27 545 495	23 172	6 453	168.8	23	131.1	299.9	10
4 200	27 545 495	37 768	11 152	103.8	23	91.3	195.1	9

Strong scaling experiment

Links with WP1

Overview

Meshes

- only simplicial meshes (segments in 1D, triangles in 2D and 3D surface, tetrahedra in 3D)
- in-house 2D unstructured mesh generator *bamg*
- in-house tools for 2D anisotropic mesh adaptation
- 3D: interface with tetgen, MMG and ParMMG (Mshmet for metric computation)

Spaces

- Arbitrary order continuous Lagrange FEs (thanks to A. Chabib)
- Raviart-Thomas or Nedelec edge elements up to third order
- DG up to degree 4 (NIPG, NCG, DG)
- can add your own finite element in the language with a C++ plugin
- FreeVol (finite volumes): G. Sadaka, F. Bouchut

Links with WP1

d7.1 benchmark

n_h	# cores	Mesh			P_1			P_3		
		# elts	build	part.	#dof	assmb.	solve	#dof	assmb.	solve
2	24	55,920	0.04 s	0.33 s	38,499	0.11 s	0.82 s	$8.48 \cdot 10^5$	28.18 s	88.76 s
4	192	$4.6 \cdot 10^5$	0.66 s	0.77 s	$2.72 \cdot 10^5$	0.13 s	1.77 s	$6.58 \cdot 10^6$	33.79 s	205.72 s
6	648	$1.57 \cdot 10^6$	2.54 s	2.42 s	$8.78 \cdot 10^5$	0.15 s	1.63 s	$2.2 \cdot 10^7$	39.81 s	293.74 s
8	1536	$3.61 \cdot 10^6$	6 s	5.66 s	$1.97 \cdot 10^6$	0.16 s	3.01 s	$5.02 \cdot 10^7$	41.57 s	325.36 s
10	3000	$7.59 \cdot 10^6$	13.06 s	10.69 s	$4.05 \cdot 10^6$	0.19 s	4.41 s	$1.05 \cdot 10^8$	47.57 s	403.42 s

Weak scaling experiment for 3D heterogeneous linear elasticity in a hollow cylinder with decreasing mesh size for P_1 and P_3 discretizations

Bottleneck at scale

Initial mesh generation and partitioning steps not scalable: we rely on a global initial mesh

Links with WP1

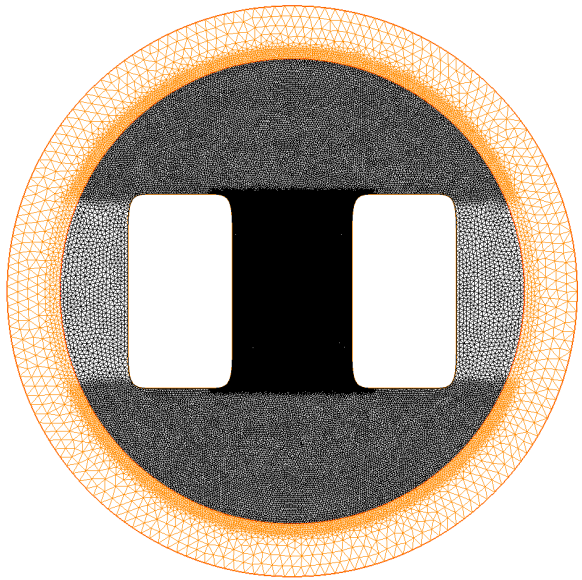
Mini-app + Roadmap

Mini-app

app-freefem-parmmg: test ParMmg performance for distributed anisotropic mesh adaptation on the Fichera corner

For waves

- Automatic generation of quasi-optimal non-uniform meshes:
  Identify trapping/visible/invisible regions using ray dynamics (RT on GPUs)
- Ongoing work of A. Chabib
- Refine a posteriori estimators by understanding error propagation using ray dynamics



Meshes

Implement curved elements (only linear elements so far)

Links with WP3

Overview

Strengths

- very strong links (developers from WP3)
- PETSc interface up-to-date with latest developments thanks to P. Jolivet
- DSL allows for quick and easy set up of test cases for various physics

Workflow

- research on new DD solvers either in PETSc/HPDDM or ffddm for prototyping
- final high-performance implementation available in PETSc/HPDDM
- e.g. multi-level methods, saddle-point solver

Links with WP3

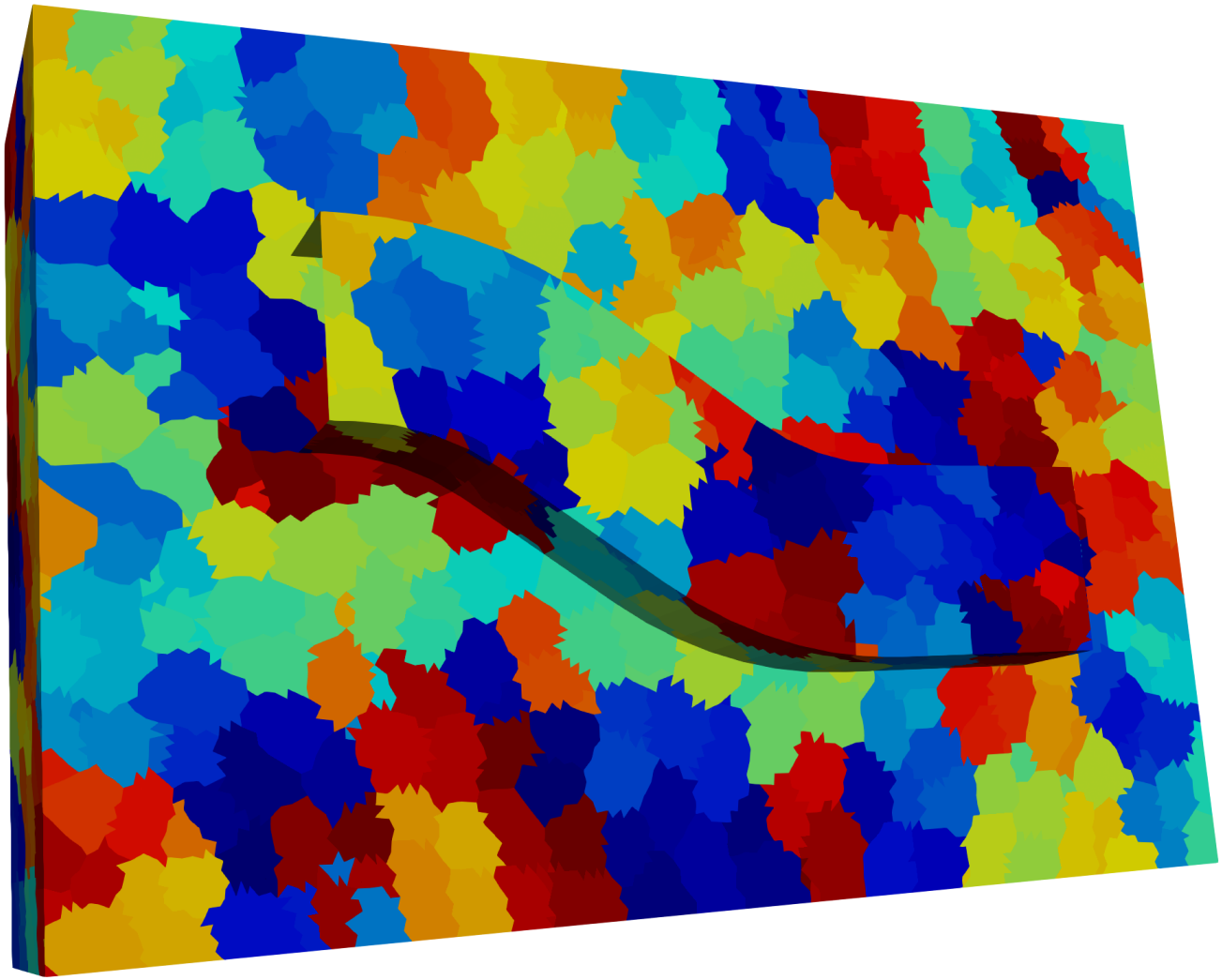
d7.1 benchmark

f	# dofs	# cores	# iter.	Computing time (s)				
				mesh part.	assembly	prec. setup	solve	VTK export
8	$3.64 \cdot 10^6$	48	7	5.97	4.33	12.05	24.18	1.95
12	$1.22 \cdot 10^7$	162	9	8.98	4.56	12.62	31.16	2.04
16	$2.89 \cdot 10^7$	384	12	13	4.86	13.34	39.63	2.14
20	$5.64 \cdot 10^7$	750	18	17.22	5.45	13.44	55.97	2.16
24	$9.73 \cdot 10^7$	1,296	25	20.59	5.66	13.94	76.63	2.24
28	$1.54 \cdot 10^8$	2,058	35	29.69	5.64	13.86	108.89	2.26
32	$2.3 \cdot 10^8$	3,072	45	42.59	5.69	14.17	147.72	2.52

Weak scaling experiment in frequency for the time-harmonic second-order Maxwell's equations in the unit cube, discretized with lowest-order Nedelec edge elements and solved with a nested two-grid optimized overlapping Schwarz preconditioner

Links with WP3

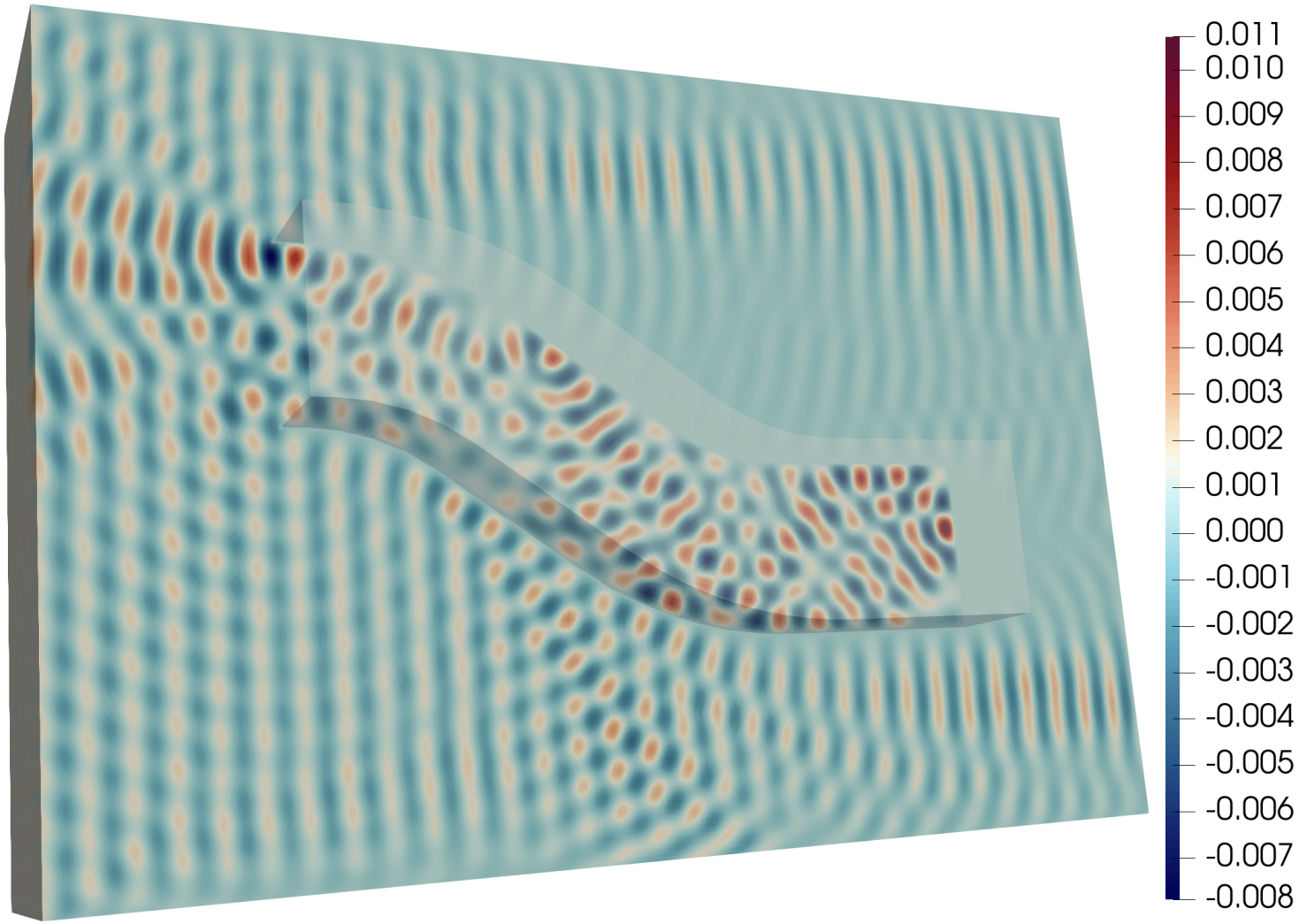
Spectral coarse spaces for waves



Partitioning of the cobra cavity domain into 2916 subdomains

Links with WP3

Spectral coarse spaces for waves



Real part of the total field for $L = 22.9\lambda$ ($k = 360m^{-1}$)

Links with WP3

Spectral coarse spaces for waves

L [λ_0]	N	n	H [λ_0]	n_s	$n_s^{\partial\Omega_s}$	1lvl It	DtN It	DtN CS	DtN CS_s	H_k It	H_k CS	H_k CS_s
5.1	32	473004	1.6	22336	2986	51	13	6400	200	17	6400	200
7.6	108	1761164	1.6	25895	3824	157	25	21600	200	38	21600	200
10.2	256	4140366	1.6	26525	4144	359	44	51200	200	63	51200	200
12.7	500	8431281	1.6	27980	4468	432	64	100000	200	106	100000	200
15.3	864	13927097	1.6	27839	5035	942	97	172800	200	>200	172800	200
17.8	1372	18879654	1.6	24589	4676	1055	120	274400	200	>200	274400	200
20.4	2048	32848020	1.6	28143	5210	3711	162	409600	200	>200	409600	200
22.9	2916	44520439	1.6	27138	5121	3398	>200	583200	200	>200	583200	200

L [λ_0]	N	n	H [λ_0]	n_s	$n_s^{\partial\Omega_s}$	1lvl It	ext It	ext CS	ext CS_s	harm It	harm CS	harm CS_s
5.1	32	473004	1.6	22336	2986	51	8	6400	200	8	6400	200
7.6	108	1761164	1.6	25895	3824	157	10	21600	200	10	21600	200
10.2	256	4140366	1.6	26525	4144	359	14	51200	200	12	51200	200
12.7	500	8431281	1.6	27980	4468	432	16	100000	200	14	100000	200
15.3	864	13927097	1.6	27839	5035	942	34	172800	200	34	172800	200
17.8	1372	18879654	1.6	24589	4676	1055	44	274400	200	46	274400	200
20.4	2048	32848020	1.6	28143	5210	3711	60	409600	200	79	409600	200
22.9	2916	44520439	1.6	27138	5121	3398	74	583200	200	107	583200	200

Weak scaling experiment for the cobra cavity test case

Links with WP3

Roadmap

Rewrite parallel data structures

- Define new parallel data structures for distributed meshes and FE spaces
- Goal: go from a sequential to a parallel script with minimal changes, e.g. by changing the type of the mesh variable from 'mesh' to 'Dmesh'
- Ongoing work of P.-L. Bacq

DD for coupled problems

- Investigate DD strategies for saddle-point problems from fluid-structure interaction
- Postdoc of L. Spies

Mixed precision for GenEO

- Adaptive strategy based on local condition number estimates supported by theoretical results
- PhD of T. Caruso

Links with WP3

Roadmap

GenEO for H(curl) problems

Improve on the GenEO-Type DD preconditioner for H(curl) problems ; works for large heterogeneties and non-trivial topologies but coarse problem too expensive

Two-level DD for BEM

- Extend the 'extended harmonic GenEO' method from F. Nataf and E. Parolin to BEM discretizations of wave problems (Helmholtz)
- Ongoing work with E. Parolin and P. Marchand

Low-rank approximation for BEM

- Interface the *theia* library from I. Chollet in Htool-DDM and FreeFEM implementing new hybrid cross approximation (HCA) techniques for low-rank compression of BEM operators
- Ongoing work with I. Chollet and P. Marchand

Links with WP5

Overview

Optimization libraries

FreeFEM is interfaced with

- IPOPT, NLopt
- TaoSolve: PETSc optimizers

Shape optimization toolboxes

- Shape and topology optimization, elastic structures, fluid mechanics
- G. Allaire, O. Pantz, C. Dapogny

Optimization book

PDE-constrained optimization within FreeFEM, F. Hecht, G. Lance and E. Trélat, 2024

- All codes available on the FreeFEM website

Mini-app

app-freefem-shape-opt-fem: shape optimisation, volume preserving, fem

Links with WP7

Overview

CI/CD

- Overhaul of the Continuous Integration pipeline using GitHub Actions
- Work done by S. Legrand

CMake

- modernize build system with Cmake for multiplatform compilation, testing and packaging
- Ongoing work of S. Legrand

Roadmap

- Write a FreeFEM debugger, allowing the user to pause the execution and inspect FreeFEM variables