The Boundary Element Method in FreeFEM

Xavier Claeys Axel Fourmont Frédéric Hecht
Pierre Marchand Pierre-Henri Tournier

FreeFEM Days, 11th Edition

Paris

December 18, 2019

Outline

Let $\mathbf{B} \in \mathbb{C}^{N \times N}$ be a dense matrix

quadratic cost in storage and complexity of the matrix-vector product

Assume that $\mathbf{B}$ can be written as follows:

$$\mathbf{B} = \sum_{j=1}^{r} \mathbf{u}_j \mathbf{v}_j^T$$

where $r \leq N, \mathbf{u}_j \in \mathbb{C}^N, \mathbf{v}_j \in \mathbb{C}^N$.

if $r < \frac{N^2}{2N}$, cost is reduced to $O(rN) < O(N^2)$

$\Longrightarrow \mathbf{B}$ is *low rank*

Usually $\mathbf{B}$ is NOT low rank

Let's write its Singular Value Decomposition (SVD):

$$\mathbf{B} = \sum_{j=1}^{P} \sigma_j \mathbf{u}_j \mathbf{v}_j^T$$

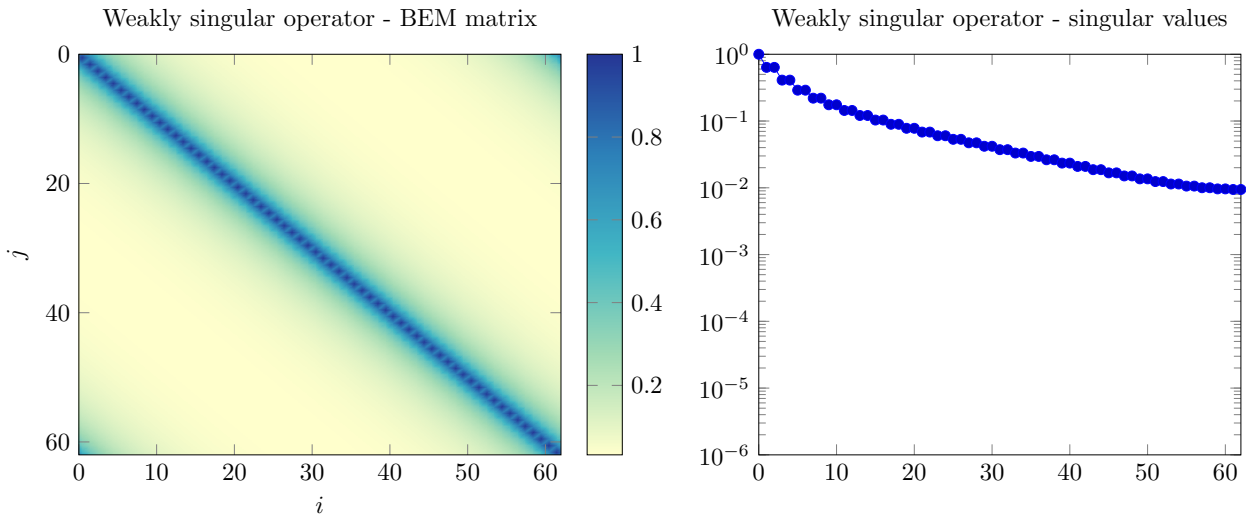
- Idea: truncate the SVD to obtain a low-rank approximation of $\mathbf{B}$

  $\Longrightarrow$ good approximation if $(\sigma_j)_{j=1}^{P}$ quickly decreases

- BUT SVD is costly ($O(N^3)$)
  AND requires all N^2 coefficients of $\mathbf{B}$ (expensive in BEM) !

$\Longrightarrow$ use only some rows and columns of $\mathbf{B}$
Partially pivoted Adaptive Cross Approximation, needs $\sim 2rN$ coefficients

BEM matrices do not have fast decreasing singular values



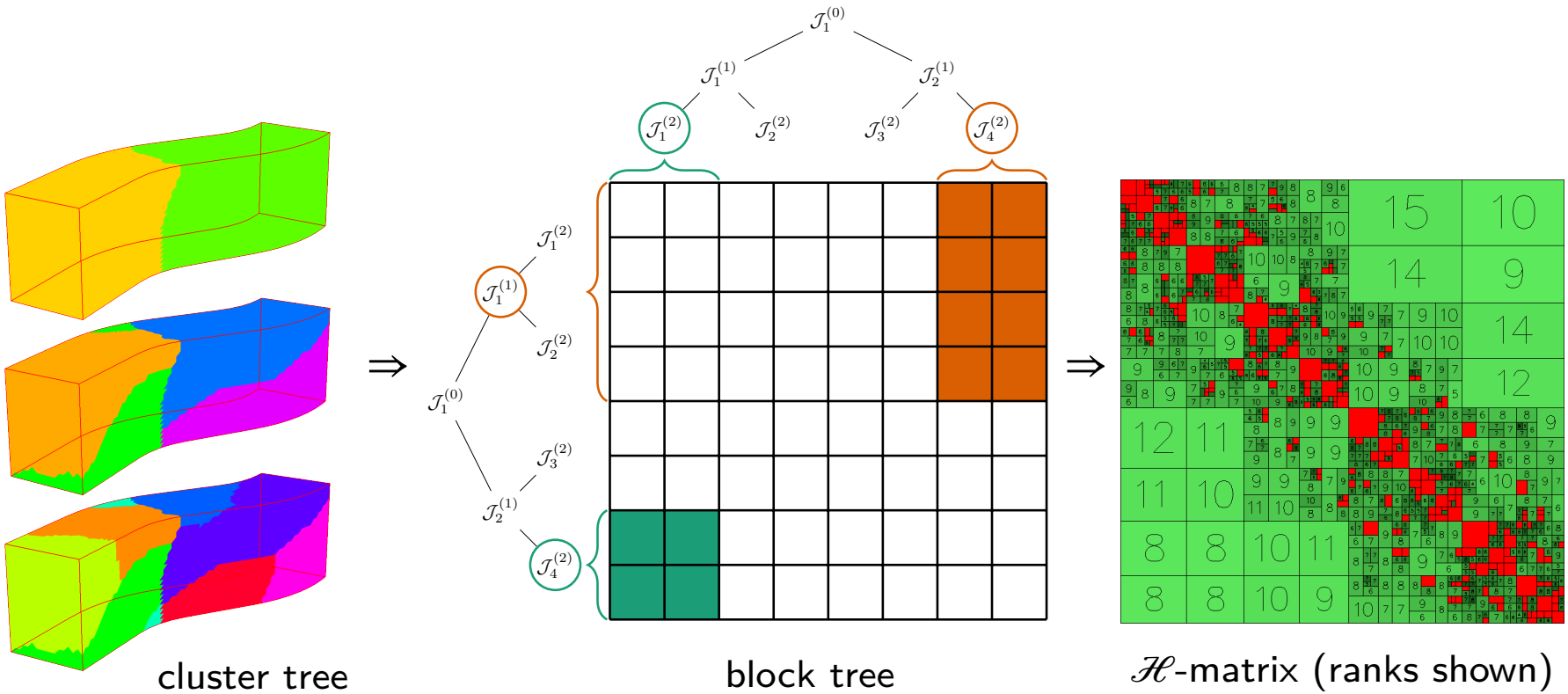
BUT *near* the diagonal : *near-field* interactions
away from the diagonal : *far-field* $\Longrightarrow$ Green function very regularizing

Idea: build a hierarchical representation of the blocks of the matrix
identify and compress admissible blocks using low-rank approximation

Hierarchical matrices

Hierarchical block structure

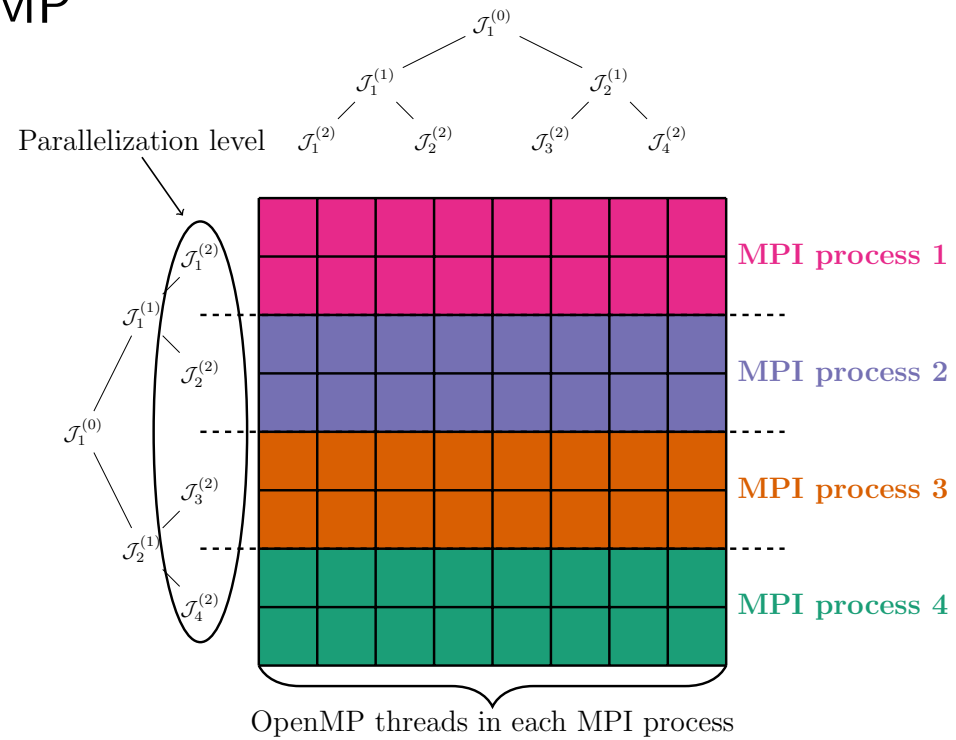
- build a hierarchical, geometric clustering of the degrees of freedom
- traverse the block tree recursively
- geometric *admissibility condition* is verified $\Longrightarrow$ compress the block



Hierarchical matrices

Htool library

- C++ library available on GitHub
  https://github.com/PierreMarchand20/htool
  by P. Marchand and P.-H. T.
- interfaces with BemTool for BEM kernels
- Parallel assembly, $\mathscr{H}$−matrix/vector and $\mathscr{H}$−matrix/matrix products using MPI and OpenMP



BEM variational forms in FreeFEM

Define the type of operator

DISCLAIMER: the DSL presented in this section is NOT live yet !! It will be live next month (January 2020)

$-\Delta u - k^2 u = 0, \quad k \in \mathbb{C}$

$k = 0$	Laplace
$k \in \mathbb{R}_+^*$	Helmholtz
$k \in \mathrm{i}\mathbb{R}_+^*$	Yukawa

```
complex k = 2*pi;
BemKernel Ker("SL",k);
BemPotential Pot("SL",k);
```

"SL"	Single Layer
"DL"	Double Layer
"HS"	Hyper Singular
"TDL"	Transpose Double Layer

BEM variational forms in FreeFEM

Define the problem

- Bilinear form on 3D surface mesh :

```
BemKernel Ker("SL",k);
varf vbem(u,v) = int2dx2d(ThS)(ThS)(BEM(Ker,u,v));
```

- Bilinear form on 2D curve mesh :

```
varf vbem(u,v) = int1dx1d(ThL)(ThL)(BEM(Ker,u,v));
```

- Assemble the HMatrix with *BemTool* and *Htool* :

```
load "htool_bem"
HMatrix<complex> H = vbem(Uh,Uh);
```

```
complex k = 2*pi;
BemKernel Ker1("HS",k);
BemKernel Ker2("DL",k);
```

Second kind formulation :

```
varf vbem(u,v) = int2dx2d(ThS)(ThS)(BEM(Ker2,u,v))
               - int2d(ThS)(0.5*u*v);
```

Combined formulation :

```
BemKernel Ker = Ker1 + 1./(1i*k) * Ker2;
varf vbem(u,v) = int2dx2d(ThS)(ThS)(BEM(Ker,u,v))
               - int2d(ThS)(0.5*u*v);
```

```
load "htool_bem"
HMatrix<complex> H = vbem(Uh,Uh);
```

$\Longrightarrow$ assemble the HMatrix in parallel using *mpisize* MPI processes
Remark: need to run the code in parallel, with *FreeFem++-mpi* or *ff-mpirun*

Default values of *Htool* parameters:

```
HMatrix<complex> H = vbem(Uh,Uh,
 compressor = "partialACA",// or "fullACA", "SVD"
 eps = 1e-3,               // target compression error
 eta = 10.,                // admissibility parameter
 minclustersize = 10,      // minimum block side size
 maxblocksize = 1000000,   // maximum n*m block size
 commworld = mpiCommWorld);// MPI communicator
```

```
fespace Uh(ThS,P1);
Uh<complex> u, b;

HMatrix<complex> H=vbem(Uh,Uh);// assemble the HMatrix

display(H); // plot H
cout << H.infos << endl; // output some stats

varf vrhs(u,v) = -int2d(ThS)(finc*v);
b[] = vrhs(0,Uh);// assemble the right-hand side
```

Access to the parallel matrix-vector product:

```
u[] = H*b[];
```

Solve the linear system with GMRES, with Jacobi preconditioner:

```
u[] = H^-1*b[];
```

```
BemPotential Pot("SL",k);
varf vpot(u,v) = int2d(ThS)(BEM(Pot,u,v));

meshS ThOut = square3(50,50);
fespace UhOut(ThOut,P1);

HMatrix<complex> P = vpot(Uh,UhOut);
```

Reconstruct the field on every node of *ThOut*
$\Longrightarrow$ matrix-vector product with P

```
UhOut<complex> uout;
uout[] = P*u[]; // u is the BEM solution
plot(uout);
```

Sound-hard scattering by a disk 1/2

```
load "htool_bem"
load "msh3"

complex k = 4*pi; // wavenumber

// incident wave
real[int] dir=[1,0];
func finc = exp(1i*k*(x*dir[0]+y*dir[1]));

border circle(t=0,2*pi){x=cos(t); y=sin(t);}
meshL Th = buildmeshL(circle(100));

fespace Uh(Th,P1);
Uh<complex> u, b;

BemKernel Ker("SL",k); // first kind formulation for Dirichlet B.C.
varf vbem(u,v) = int1dx1d(Th)(Th)(BEM(Ker,u,v));

HMatrix<complex> H = vbem(Uh,Uh);// assemble the HMatrix

varf vrhs(u,v) = -int1d(Th)(finc*v);
b[] = vrhs(0,Uh);// assemble the right-hand side

u[] = H^-1*b[]; // solve the linear system
```

Sound-hard scattering by a disk 2/2

```
BemPotential Pot("SL",k); // potential for the radiated field
varf vpot(u,v) = int1d(Th)(BEM(Pot,u,v));

// Reconstruct the field in a square around the disk:
int np = 200;
int R = 3;

border b1(t=-R, R){x=t; y=-R;}
border b2(t=-R, R){x=R; y=t;}
border b3(t=-R, R){x=-t; y=R;}
border b4(t=-R, R){x=-R; y=-t;}

mesh ThOut = buildmesh(b1(np)+b2(np)+b3(np)+b4(np)+circle(-np*pi/R));

fespace UhOut(ThOut,P1);

HMatrix<complex> P = vpot(Uh,UhOut);

UhOut<complex> v;

v[] = P*u[];

v = v + finc; // total field

UhOut vabs = abs(v);
plot(vabs);
```

DISCLAIMER

DISCLAIMER: the DSL presented in the previous section is NOT live yet !! It will be live next month (January 2020)

For now:

```
HMatrix<complex> H = assemblecomplexHESL(Uh,Uh,k,eta=10,eps=1e-2);
```

Combined formulation:

```
HMatrix<complex> H = assemblecomplexHEcombinedHSDL(Uh,Uh,k,eta=10,
    eps=1e-3,alpha=-0.5,combinedcoef=1./(1i*k));
```

Potential:

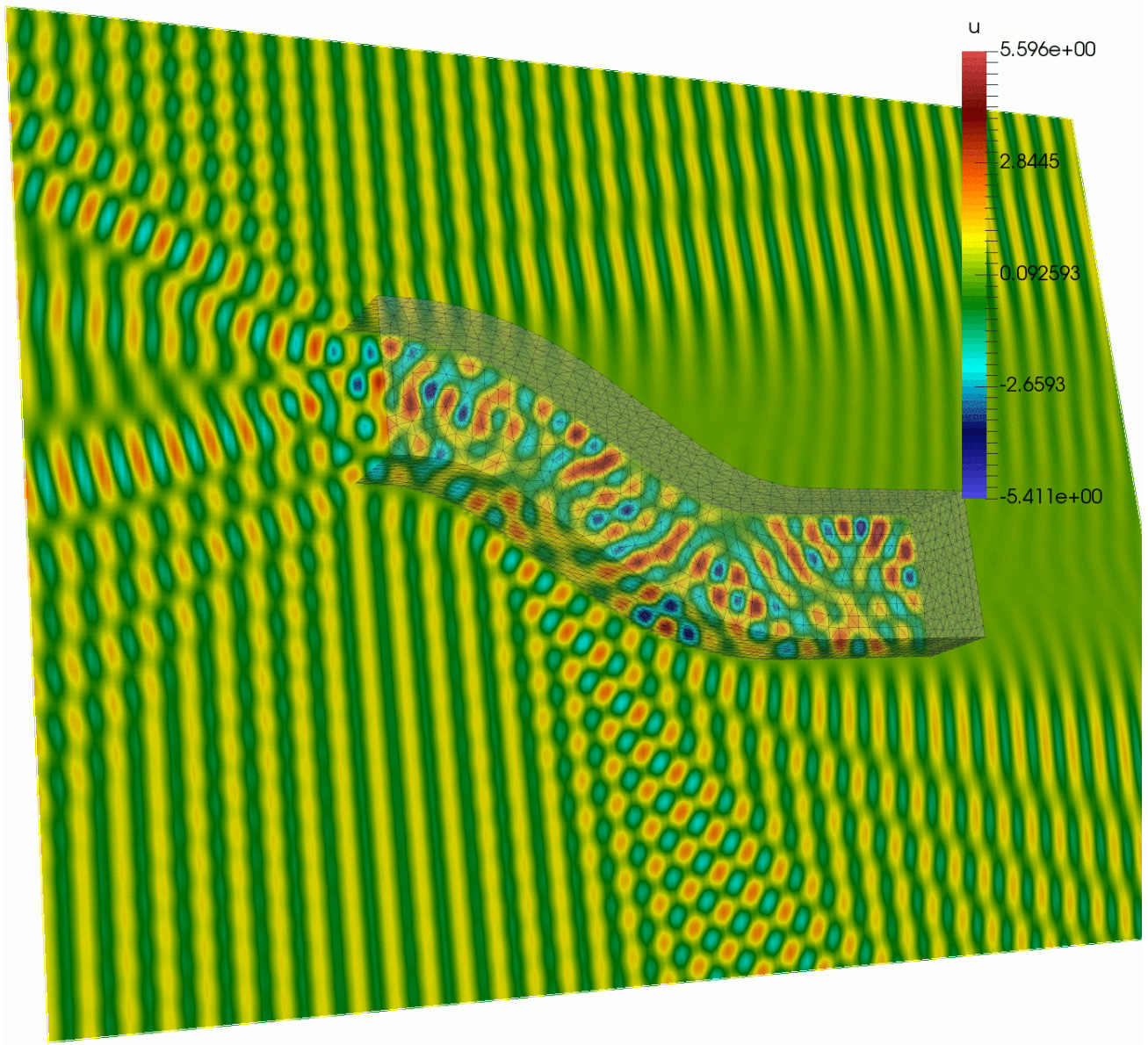
```
HMatrix<complex> HPot = assemblecomplexHESLPot(Uh,UhOut,k);
```

Cf. examples in the bem branch

Plane wave scattering by the COBRA cavity

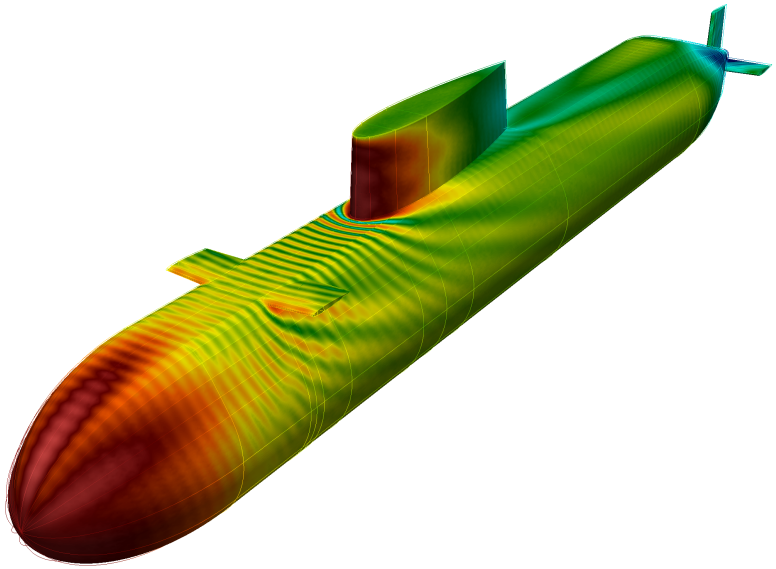
- P1, 10 points per wavelength
- 33K dofs
- Dirichlet B.C.
- First kind formulation
- 94.4% compression
- two-level DD precond, coarse mesh w/ 3.3 p.p. wavelength (3.7K dofs)

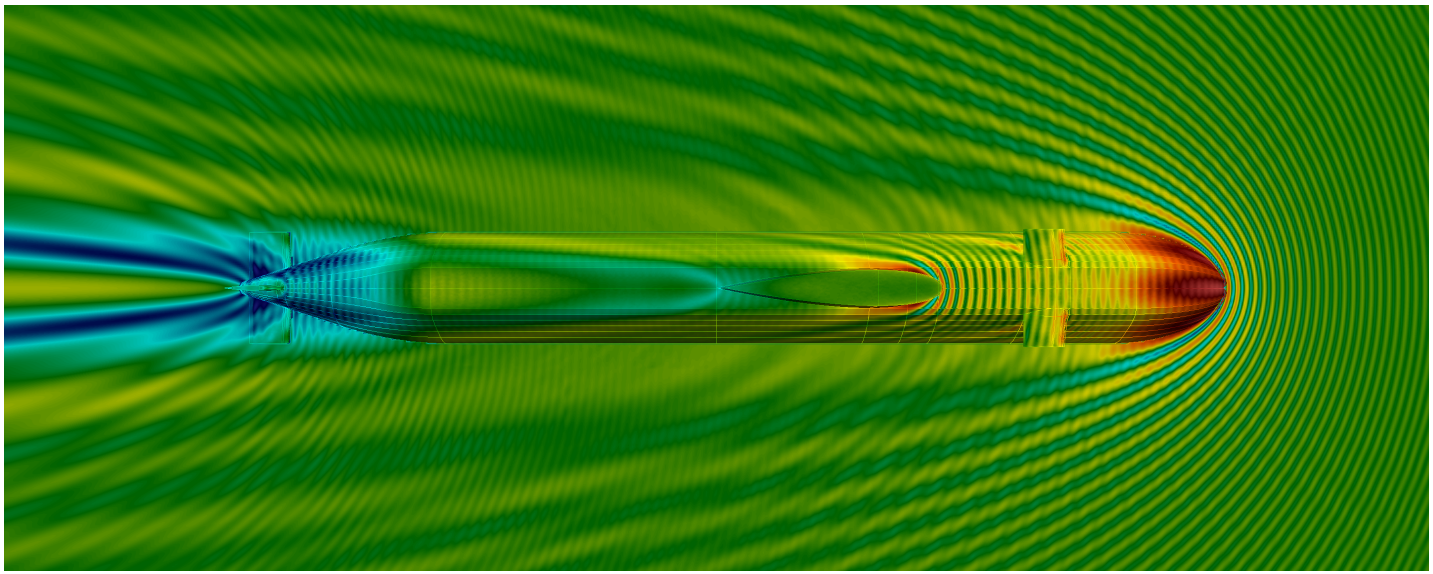
- assembly: 29.5s on 192 cores
- 7 gmres it (0.5s)
- radiation: 6.8s



Plane wave scattering by the BeTSSi submarine

- $f = 300$ Hz, P1, 166K dofs
- Neumann B.C.
- Combined field formulation
- assembly : 57s on 1792 cores
- 97.7% compression
- solution : 38 gmres it (1s)
- radiation : 8.2s





What is to come ?

- Finalize the BEM DSL in FreeFEM $\Longrightarrow$ ~ January 2020
- Extension to vectorial problems, Maxwell, elasticity, Stokes, ...
- $\mathscr{H} - LU$ factorization in Htool
- Domain decomposition preconditioners for BEM
  - already some ffddm and PETSc examples
  - cf. Thesis of Pierre Marchand
- FEM-BEM coupling !