The Boundary Element Method in FreeFEM

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

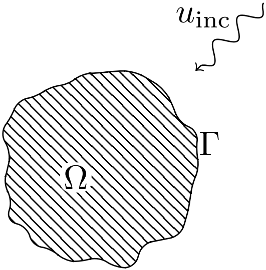
CANUM 2022

June 17, 2022

Outline

Volume form of the problem:

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



Green kernel: $\mathcal{G}(\boldsymbol{x}) = \exp(\imath k|\boldsymbol{x}|)/(4\pi|\boldsymbol{x}|)$

Single Layer Potential $\text{SL} : \forall q \in H^{-1/2}(\Gamma)$,

$$\text{SL}(q)(\boldsymbol{x}) = \int_\Gamma \mathcal{G}(\boldsymbol{x} - \boldsymbol{y}) q(\boldsymbol{y}) d\sigma(\boldsymbol{y}), \quad \forall \boldsymbol{x} \in \mathbb{R}^3 \backslash \Gamma$$

SL produces solutions of the PDE which satisfy the necessary conditions at infinity (here the Helmholtz equation and the Sommerfeld radiation condition)

$\Longrightarrow$ look for $p \in H^{-1/2}(\Gamma)$ such that $\text{SL}(p)(\boldsymbol{x}) = u(\boldsymbol{x})$ with $u = -u_{\text{inc}}$ on Γ

A variational formulation of the integral equation can be obtained by imposing the Dirichlet condition in a weak manner: find $p : \Gamma \to \mathbb{C}$ such that

$$\int_{\Gamma \times \Gamma} \frac{\exp(\imath k|\boldsymbol{x} - \boldsymbol{y}|)}{4\pi|\boldsymbol{x} - \boldsymbol{y}|} p(\boldsymbol{y}) q(\boldsymbol{x}) d\sigma(\boldsymbol{x}, \boldsymbol{y}) = -\int_\Gamma u_{\text{inc}}(\boldsymbol{x}) q(\boldsymbol{x}) d\sigma(\boldsymbol{x}) \quad \forall q : \Gamma \to \mathbb{C}$$

The building blocks for all existing integral formulations consist in four operators:

- **Single layer operator**

$$p,q \mapsto \mathscr{SL}(p,q) = \int_{\Gamma\times\Gamma} p(\boldsymbol{x})q(\boldsymbol{y})\mathscr{G}(\boldsymbol{x}-\boldsymbol{y})d\sigma(\boldsymbol{x},\boldsymbol{y})$$

- **Double layer operator**

$$p,q \mapsto \mathscr{DL}(p,q) = \int_{\Gamma\times\Gamma} p(\boldsymbol{x})q(\boldsymbol{y})\frac{\partial}{\partial \boldsymbol{n}(\boldsymbol{y})}\mathscr{G}(\boldsymbol{x}-\boldsymbol{y})d\sigma(\boldsymbol{x},\boldsymbol{y})$$

- **Transpose double layer operator**

$$p,q \mapsto \mathscr{TDL}(p,q) = \int_{\Gamma\times\Gamma} p(\boldsymbol{x})q(\boldsymbol{y})\frac{\partial}{\partial \boldsymbol{n}(\boldsymbol{x})}\mathscr{G}(\boldsymbol{x}-\boldsymbol{y})d\sigma(\boldsymbol{x},\boldsymbol{y})$$

- **Hypersingular operator**

$$p,q \mapsto \mathscr{HS}(p,q) = \int_{\Gamma\times\Gamma} p(\boldsymbol{x})q(\boldsymbol{y})\frac{\partial}{\partial \boldsymbol{n}(\boldsymbol{x})}\frac{\partial}{\partial \boldsymbol{n}(\boldsymbol{y})}\mathscr{G}(\boldsymbol{x}-\boldsymbol{y})d\sigma(\boldsymbol{x},\boldsymbol{y})$$

Quick recap on the Boundary Element Method

BEMTool library

BEMTool is a general purpose BEM library written by Xavier Claeys (LJLL).
It is written in C++ and handles:

- Laplace, Yukawa, Helmholtz, Maxwell
- both in 2D and in 3D
- 1D, 2D and 3D triangulations (not necessarily flat)
- $\mathbb{P}_k$-Lagrange $k = 0, 1, 2$ and surface $\mathbb{RT}_0$

BEMTool is interfaced with FreeFEM.

It is available on GitHub https://github.com/xclaeys/BemTool

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 **B** is NOT low rank

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

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

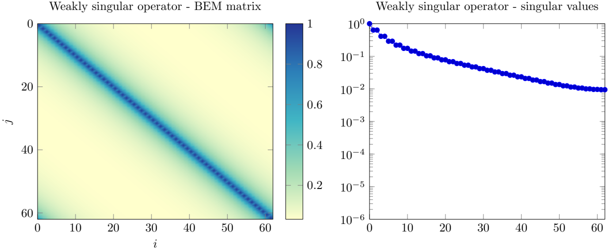
- Idea: truncate the SVD to obtain a low-rank approximation of **B**

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

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

$\Longrightarrow$ use only some rows and columns of **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

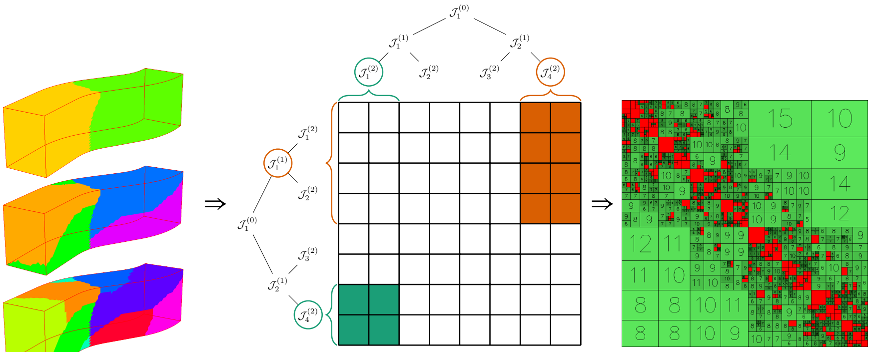
Idea: build a hierarchical representation of the blocks of the matrix
identify and compress admissible blocks using low-rank approximation

Hierarchical matrices

Hierarchical block structure

- build a hierarchical, geometric clustering of the degrees of freedom
- traverse the block tree recursively
- geometric *admissibility condition*:

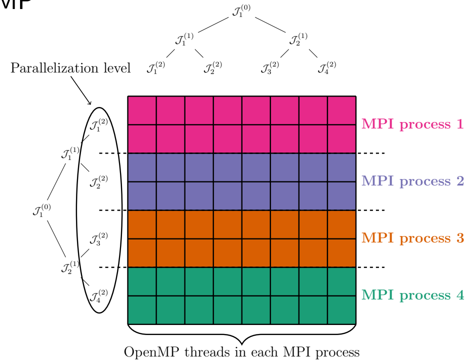
$$\max(\mathrm{diam}(X), \mathrm{diam}(Y)) \le \eta \; \mathrm{dist}(X, Y) \implies \text{compress the block}$$



cluster tree ⇒ block tree ⇒ $\mathscr{H}$-matrix (ranks shown)

Hierarchical matrices
Htool library

- C++ library available on GitHub
  https://github.com/PierreMarchand20/htool
  by Pierre 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

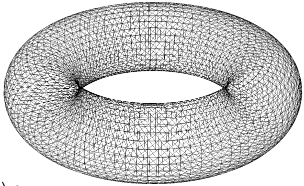


BEM variational forms in FreeFEM

a reminder on surface and line meshes

build a 2D surface mesh:

```
func torex=(R+r*cos(y*pi*2))*cos(x*pi*2);
func torey=(R+r*cos(y*pi*2))*sin(x*pi*2);
func torez=r*sin(y*pi*2);
meshS ThS=square3(nx,ny,[torex,torey,torez],removeduplicate=true);
```



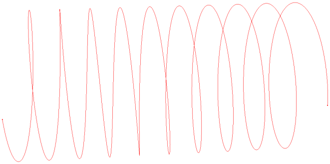
```
mesh Th = square(10,10);
meshS ThS = movemesh23(Th, transfo=[x,y,cos(x)^2+sin(y)^2]);

mesh3 Th3 = cube(10,10,10);
meshS ThS = extract(Th3);

int[int] labs = [1,2,3,4];
meshS ThS = extract(Th3, label=labs);
```

build a 1D line mesh:

```
border b(t = 0, 20*pi){x=t/pi/5; y=cos(t); z=sin(t);}
meshL ThL = buildmeshL(b(1000));
```



```
mesh Th = square(10,10);
meshL ThL = extract(Th);

int[int] labs = [1,2];
meshL ThL = extract(Th, label=labs);
```

You can find all available operations on surface and line meshes in the FreeFEM documentation

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

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

NEW Maxwell EFIE:

$k \in \mathbb{R}_+^*$ and surface $\mathbb{RT}_0$ space (RT0S)

Maxwell_cube_EFIE.edp

Operators

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

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

Potentials

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

"SL"	Single Layer
"DL"	Double Layer

BEM variational forms in FreeFEM

Define the problem

- Bilinear form on 3D surface mesh :

```
BemKernel Ker("SL", k=2*pi);
varf vbem(u,v) = int2dx2d(ThS)(ThS)(BEM(Ker,u,v));
```

or directly:

```
varf vbem(u,v) =
int2dx2d(ThS)(ThS)(BEM(BemKernel("SL",k=2*pi),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 "bem"
HMatrix<complex> H = vbem(Uh,Uh);
```

```
complex k=2*pi;
BemKernel Ker1("HS", k=k);
BemKernel Ker2("DL", k=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 = 1./(1i*k) * Ker1 + Ker2;
varf vbem(u,v) = int2dx2d(ThS)(ThS)(BEM(Ker,u,v))
               - int2d(ThS)(0.5*u*v);
```

Helmholtz_circle_Dirichlet.edp Helmholtz_circle_Neumann.edp

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

⟹ 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"
 eta = 10.,                // admissibility parameter
 eps = 1e-3,               // target compression error
 minclustersize = 10,      // minimum block side size
 maxblocksize = 1000000,   // maximum n*m block size
 commworld = mpiCommWorld);// MPI communicator
```

You can also change default values using global variables `htoolEpsilon`, `htoolEta`, ...

BEM variational forms in FreeFEM

Solve the problem

```
fespace Uh(ThS,P1);
Uh<complex> p, 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:

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

Solve the linear system with GMRES, with Jacobi preconditioner:

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

```
BemPotential Pot("SL", k=2*pi);
varf vpot(u,v) = int2d(ThS)(POT(Pot,u,v));
```

or directly:

```
varf vpot(u,v) = int2d(ThS)(POT(BemPotential("SL", k=2*pi),u,v));
```

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

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

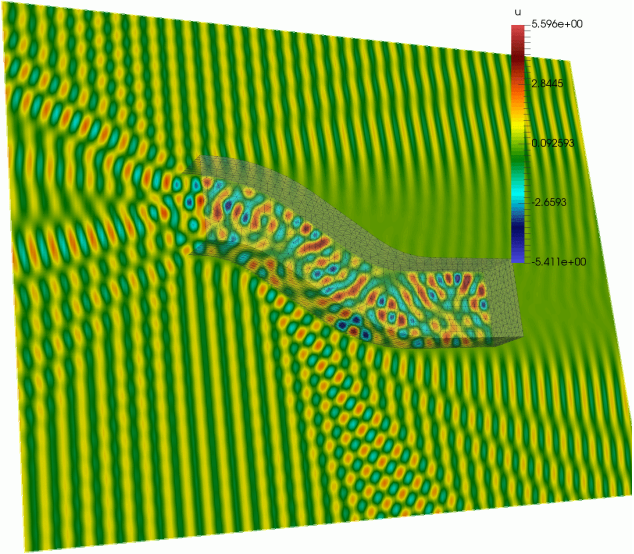
Reconstruct the field on every node of *ThOut*
$\Longrightarrow$ matrix-vector product with HP

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

Plane wave scattering by the COBRA cavity

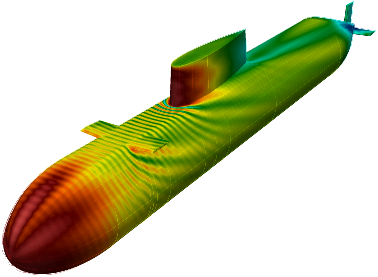
- P1, 10 points per wavelength
- 33K dofs
- Dirichlet B.C.
- First kind formulation
- 94.4% compression
- two-level DD precond, coarse mesh w/ 3.3 p.p. wavelength (3.7K dofs)

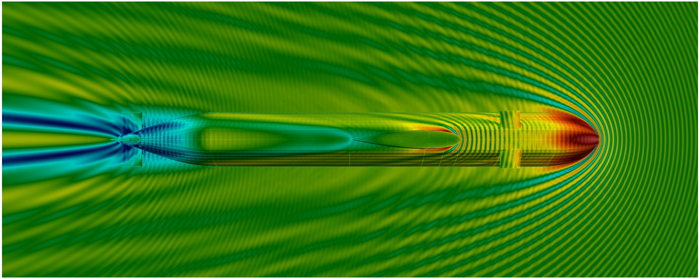
- assembly: 29.5s on 192 cores
- 7 gmres it (0.5s)
- radiation: 6.8s



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





Current/Future developments

- be able to define high-level composite operators in FreeFEM with different types of building blocks (sparse/dense/hierarchical matrices, functions, ...) to handle coupling problems, FEM-BEM variational formulations, ...

  Helmholtz-2d-FEM-BEM-coupling-MUMPS.edp

  helmholtz-coupled-2d-PETSc-complex.edp

- future developments in Htool
  - $\mathscr{H}$-LU factorization
  - more compression techniques for low-rank blocks (Block ACA, HCA, randomized SVD, ...)