Ongoing developments for variational forms: coupled problems, parallelization

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

FreeFEM Days, 14th Edition

December 8, 2022

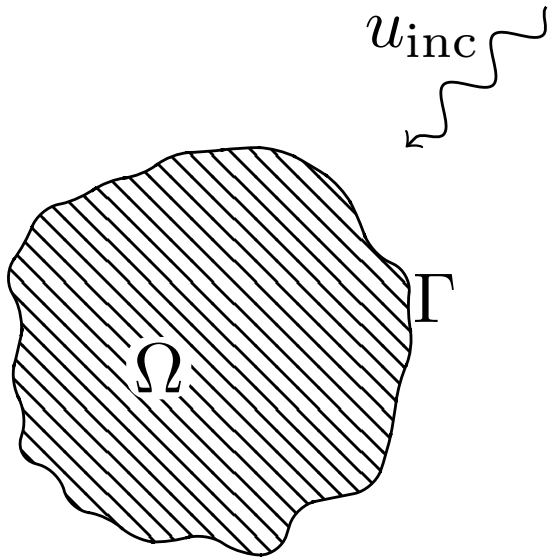
Outline

Quick recap on the Boundary Element Method

Model problem

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

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

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

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

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

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

Hierarchical matrices

Hierarchical block structure, Low-rank approximation

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

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

Low-rank approximation ?
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

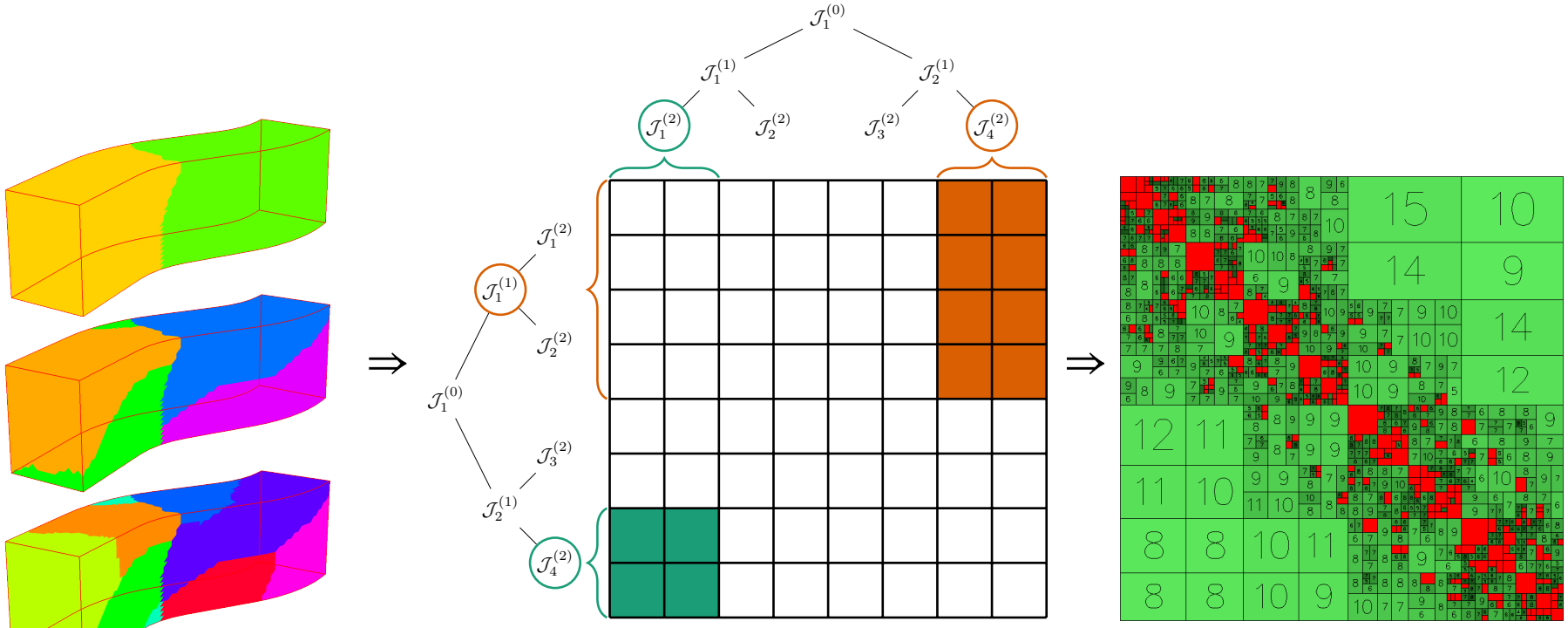
$\Rightarrow$ Build a low-rank approximation $\mathbf{B} \approx \sum_{j=1}^{r} \mathbf{u}_j \mathbf{v}_j^T$ ($\sim$ approx. truncated SVD) for each admissible block $\mathbf{B}$, using e.g. *Partially pivoted Adaptive Cross Approximation*, which needs only $2r$ rows/columns

Hierarchical matrices

Summary

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

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

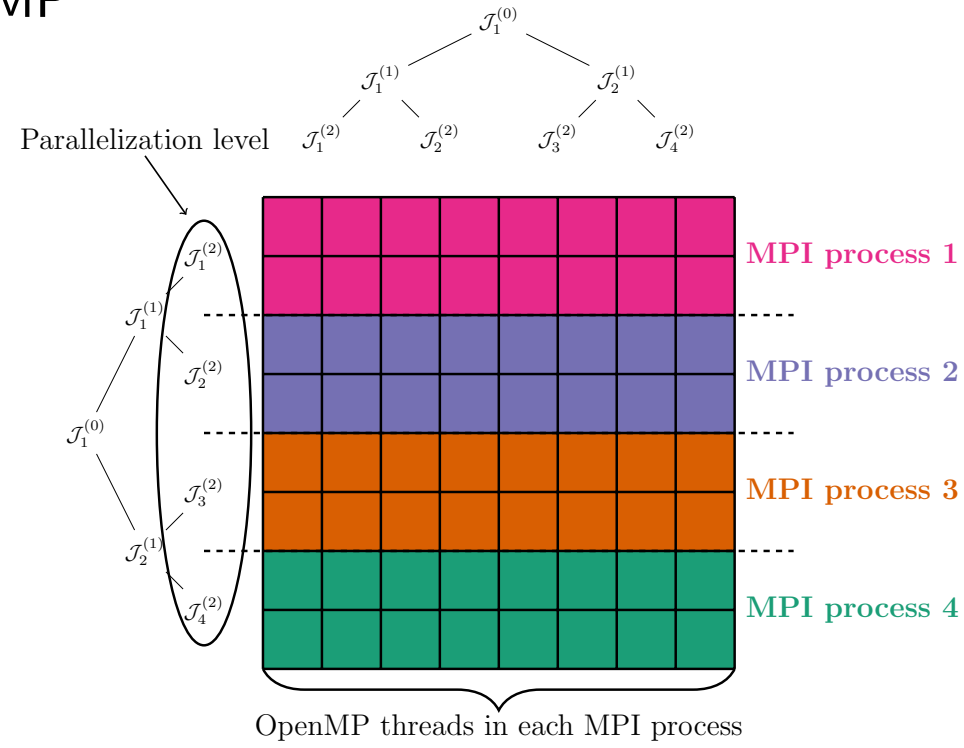


cluster tree | block tree | $\mathscr{H}$-matrix (ranks shown)

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

Define the type of operator

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

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

NEW Maxwell EFIE:

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

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);
```

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

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

Solve the linear system with GMRES, with Jacobi preconditioner:

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

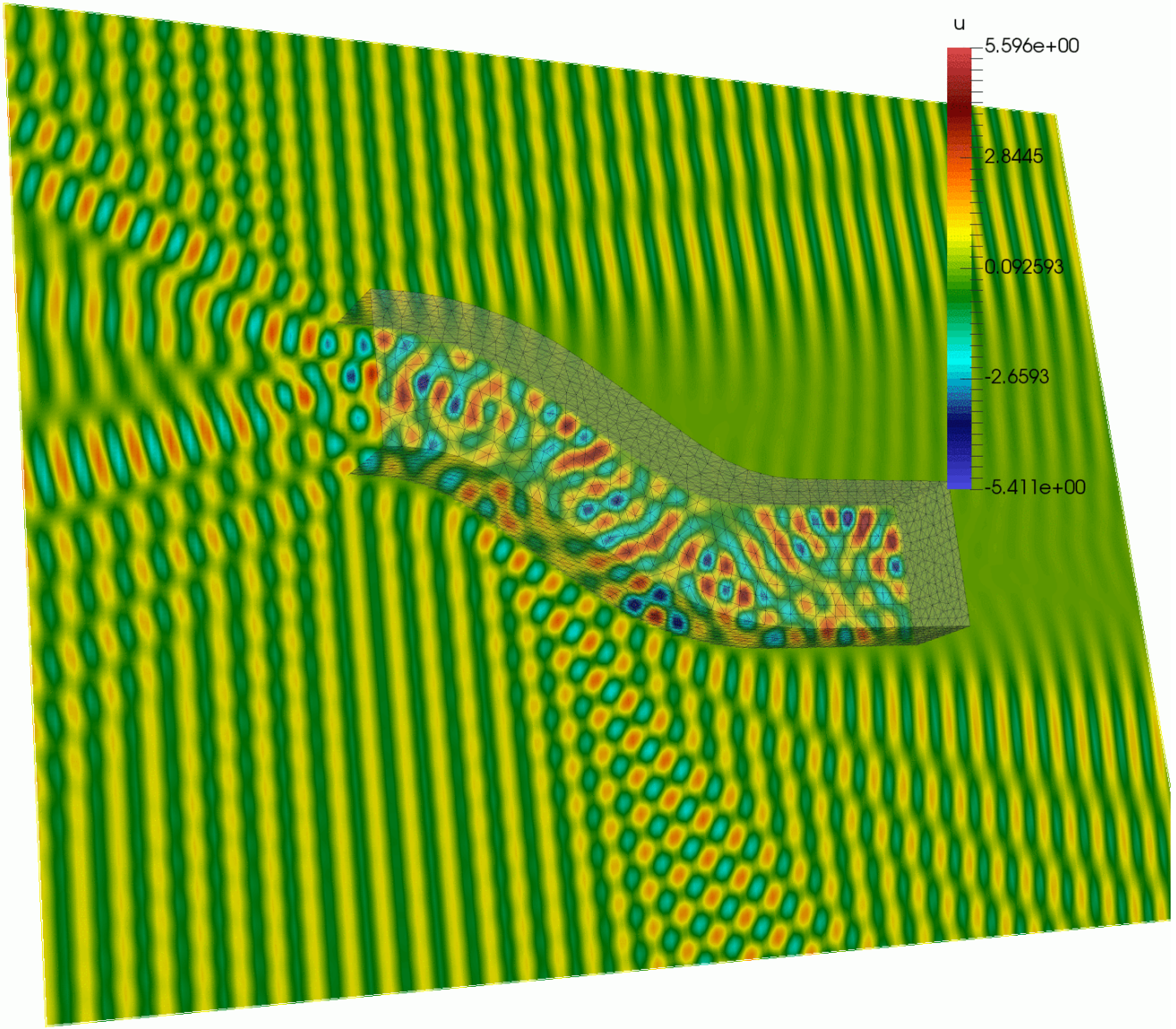
Reconstruct the solution on an output mesh using the Potential:

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

meshS ThOut = square3(50,50);
fespace UhOut(ThOut,P1);
UhOut<complex> u;

HMatrix<complex> HP = vpot(Uh,UhOut);
// reconstruct the field on every node of ThOut:
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

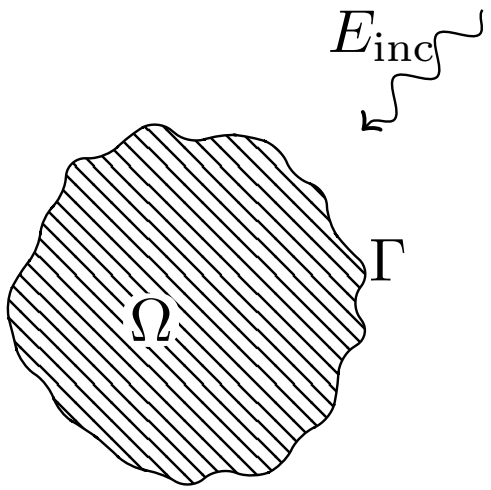


Time-harmonic Maxwell's equations

Scattered field of a perfect electric conductor (PEC) object

The scattered fields $\boldsymbol{E}$ and $\boldsymbol{H}$ for an homogeneous medium verify:

$$\begin{aligned}
\mathrm{curl}\boldsymbol{E} + \imath\omega\mu_0\boldsymbol{H} &= 0, \quad \text{in } \mathscr{R}^3\backslash\Omega \\
\mathrm{curl}\boldsymbol{H} + \imath\omega\epsilon_0\boldsymbol{E} &= 0, \quad \text{in } \mathscr{R}^3\backslash\Omega \\
\boldsymbol{E}\wedge\boldsymbol{n} &= -\boldsymbol{E}_{inc}\wedge\boldsymbol{n}, \quad \text{on } \Gamma \\
\lim_{r\to+\infty} r\left\|\sqrt{\epsilon_0}\boldsymbol{E} - \sqrt{\mu_0}\boldsymbol{H}\nabla\frac{\boldsymbol{r}}{|\boldsymbol{r}|}\right\| &= 0.
\end{aligned}$$



Time convention: $\exp(\imath\omega t)$.

We introduce the total magnetic trace:

$$\boldsymbol{j} = \imath\,\kappa\,Z_0\,\boldsymbol{n}\wedge(H + H_{\mathrm{inc}}).$$

vectorial BEM equation: Maxwell

EFIE equation: BEMTool

The total magnetic trace:

$$\boldsymbol{j} = \imath\,\kappa\, Z_0\, \boldsymbol{n} \wedge (H + H_{\text{inc}})$$

Electric field integral equation (EFIE)

$$\int_{\Gamma\times\Gamma} \mathscr{G}_\kappa(\boldsymbol{x}-\boldsymbol{y})\big(\boldsymbol{j}(\boldsymbol{x})\cdot\boldsymbol{v}(\boldsymbol{y}) - \kappa^{-2}\,\mathrm{div}_\Gamma \boldsymbol{j}(\boldsymbol{x})\,\mathrm{div}_\Gamma \boldsymbol{v}(\boldsymbol{y})\big)\,d\sigma(\boldsymbol{x})\,d\sigma(\boldsymbol{y})$$
$$= -\int_\Gamma \boldsymbol{E}_{inc}(\boldsymbol{x})\cdot\boldsymbol{v}(\boldsymbol{x})\,d\sigma(\boldsymbol{x})$$

Green kernel: $\mathscr{G}_\kappa(\boldsymbol{x}) = \exp(\imath\kappa|\boldsymbol{x}|)/(4\imath\pi|\boldsymbol{x}|)$

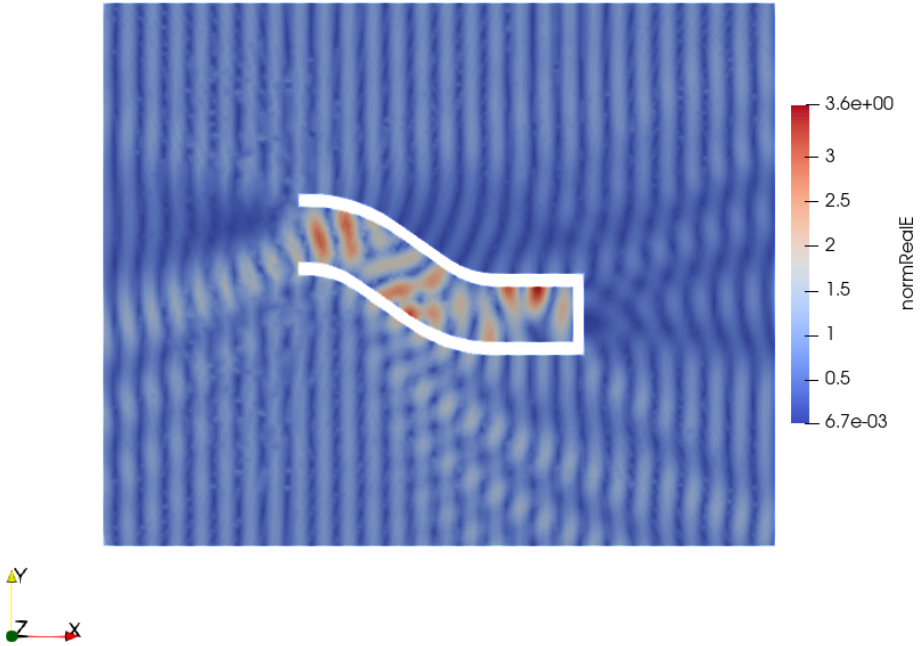
The scattered field $\boldsymbol{E}$ is obtained with the Stratton-Chu Formula:

$$\boldsymbol{E}(\boldsymbol{y}) = \int_\Gamma \mathscr{G}_\kappa(|\boldsymbol{y}-\boldsymbol{x}|)\boldsymbol{j}(\boldsymbol{x})d\sigma(\boldsymbol{x}) + \frac{1}{\kappa^2}\nabla_y\left(\int_\Gamma \mathscr{G}_\kappa(|\boldsymbol{y}-\boldsymbol{x}|)\,\mathrm{div}_\Gamma \boldsymbol{j}(\boldsymbol{x})\,d\sigma(\boldsymbol{x})\right).$$

vectorial BEM equation: Maxwell

Cobracavity

- Thickness arround the cavity.
- $f = 5Ghz$ Hz, RT0, 21204 dofs
- PEC material
- EFIE :
- assembly : 426 s
- 84 % compression
- Block Jacobi preconditioner, 8 mpi.
- solution j: 2743 gmres it (410s)
- potentiel E : 44 s
- total time 881s



Composite FE Space

Composite FE Space

Definition and example 1/2

- FE space is defined with a Finite element (P1, P2, ...) and a mesh.
- Vectorial FE Space :

```
fespace Xh(Th,[P2,P2,P1]); // depends on one mesh
fespace XhPeriodic(Th,[P2,P2,P1],periodic=[[1,x],[3,x]]);
XhPeriodic [u1,u2,p];  // periodic for all components
```

- Definition of composite FE space :

$$X_h = U_h^1(T_h^1, FE_1) \times U_h^2(T_h^2, FE_2) \cdots \times U_h^n(T_h^n, FE_n),$$

where T_h^i and T_h^j can be different meshes ($\times$ = cartesian product).

- Rewritting Vectorial FE space:

```
fespace Uh1(Th,P2);
fespace Uh2(Th,P2,periodic=[[1,x],[3,x]]);
fespace Ph(Th,P1);

fespace Xh(<Uh1,Uh2,Ph>);  // Xh = Uh1 × Uh2 × Ph
```

Composite FE Space

Definition and example 2/2

- Definition of composite FE space:

$$X_h = U_h^1(T_h^1, FE_1) \times U_h^2(T_h^2, FE_2) \cdots U_h^n(T_h^n, FE_n),$$

where T_h^i and T_h^j can be different meshes ($\times$ = cartesian product).

- Rewritting Vectorial FE space:

```
fespace Uh1(Th,P2);
fespace Uh2(Th,P2,periodic=[[1,x],[3,x]]);
fespace Ph(Th,P1);

fespace Xh(< Uh1,Uh2,Ph> ); // Xh = Uh1 × Uh2 × Ph
```

- FEM-BEM coupling:

```
fespace Xfem(Th,P1);   // Th is a 2d volumic mesh
fespace Xbem(ThL,P1); // ThL is a curve mesh (meshL)

fespace Xh(<Xfem,Xbem>); // Xh = Xfem × Xbem
```

Composite FE space

problem/solve keyword

- Current way of defining a problem:

```
// definition of the FE spaces
  fespace Uh1(Th1,P2); fespace Uh2(Th2,P1);
  fespace Vh1(Th3,P2); fespace Vh2(Th4,P1);
  // definition of FE functions
  Uh1 u1; Uh2 u2;
  Vh1 v1; Vh2 v2;

  problem myPB([u1,u2],[v1,v2]) = int2d(Th1) (...)
                                + int2d(Th2) (...)
                              - int1d(Th1)(...) + on  (...);
  myPB;
```

Limitations:

- All FE spaces are defined on the *same* mesh (Th1=Th2$\cdots$=Th4).
- Uh1 and Vh1 (resp. Uh2 and Vh2) are the *same* FE space.
- *same* = computer memory

$\Rightarrow$ Rewrite of problem/solve function in the kernel with composite FE space to bypass these limitations.

Composite FE space

Remark limitations

- In the current, we can bypass this limitations by using the keyword varf.
- Mixed variational form: Uh1=Vh1, Uh2=Vh2 are different FE spaces.

```
// definition of the FE spaces
fespace Uh1(Th1,P2);
fespace Uh2(Th2,P1); // Th1 and Th2 are different mesh
// definition of FE functions
Uh1 u1,v1;
Uh2 u2,v2;

varf vfA11(u1,v1) = int2d(Th1)( dx(u1)*dx(v1) + dy(u1)*dy(v1) )
+ int2d(Th1)( f1*v1) + on (1,u1=0);
matrix A11 = vfA11(Uh1,Uh1); // Interaction between Uh1 and Uh1
matrix A21 = vfA21(Uh1,Uh2); // Interaction between Uh1 and Uh2
matrix A12 = vfA12(Uh2,Uh1); // Interaction between Uh2 and Uh1
matrix A22 = vfA22(Uh2,Uh2); // Interaction between Uh2 and Uh2

matrix A = [[A11,A12], [A21,A22]];  // global matrix
real[int] rhs = [rhs1,rhs2]; // where real[int] rhs1=vfA11(0,Vh1);
real[int] sol = A^-1*rhs;    // solution of the system
```

Limitations:

- A variational form for each block of the system for mixed variationnal form.

⇒ The goal of composite FE space is to write one varf in this case.

Composite FE space

problem/solve keyword

- New formulation with composite FE spaces:

```
  // definition of the FE spaces
  fespace Uh(Th1,[P2,P2]);
  fespace Ph(Th2,P1);

  // definition of FE functions
  Uh [u1,u2];
  Uh [v1,v2];
  Ph p;
  Ph q;

// use angle bracket "<" ">" for composite
// "[" "]" defined a FE function
  problem myCompositePB(<[u1,u2],[p]>,<[v1,v2],[q]>) = ...

  myCompositePB;

  plot( [u1,u2] ); // u1 and u2 contains the solution of the
      problem
```

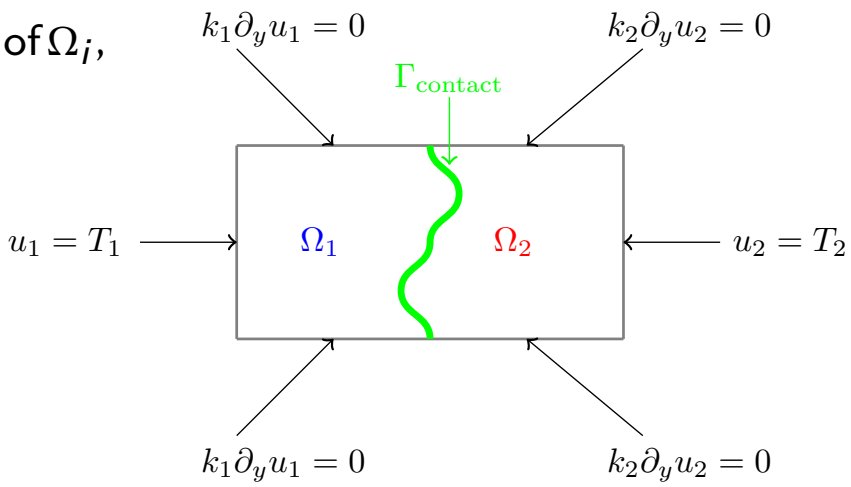
- allows for different meshes, and even different mesh types (mesh, meshS, meshL, ...)
- Uh1 and Vh1 (resp. Uh2 and Vh2) are the same FE space for the moment.

Example composite FE Space

Thermic contact

Consider the following thermic contact problem:

$$\begin{cases} \nabla(k_i \nabla u_i) = f_i \quad \text{in} \Omega_i, \\ k_i \nabla u_i \cdot \boldsymbol{n} = 0, \text{on upper and lower boundary of} \Omega_i, \\ \qquad u_1 = T_1 \text{ for } x = 0, \\ \qquad u_2 = T_2 \text{ for } x = L. \end{cases}$$



- Γ_{contact} is the boundary between the two domains:

$$k_1 \nabla u_1 . n_1 = -k_2 \nabla u_2 . n_2 \quad \text{in } \Gamma_{\text{contact}}.$$

- hypothesis on the Thermal flux in the contact boundary: $k_1 \nabla u_1 . n_1 = c\,(u_2 - u_1)$

- variational form:

$$\forall (v_1, v_2), \begin{cases} \displaystyle\int_{\Omega_1} k_1 \nabla u_1 . \nabla v_1 \, dX + c \int_{\Gamma_{\text{contact}}} (u_1 - u_2) v_1 \, d\sigma(X) = 0, \\ \displaystyle\int_{\Omega_2} k_2 \nabla u_2 . \nabla v_2 \, dX + c \int_{\Gamma_{\text{contact}}} (u_2 - u_1) v_2 \, d\sigma(X) = 0. \end{cases}$$

Example composite FE Space

Thermic contact

- variational form:

$$\forall (v_1, v_2), \begin{cases} \displaystyle\int_{\Omega_1} k_1 \nabla u_1 . \nabla v_1 \, dX + c \int_{\Gamma_{\text{contact}}} (u_1 - u_2) v_1 \, d\sigma(X) = 0, \\ \displaystyle\int_{\Omega_2} k_2 \nabla u_2 . \nabla v_2 \, dX + c \int_{\Gamma_{\text{contact}}} (u_2 - u_1) v_2 \, d\sigma(X) = 0. \end{cases}$$

```
fespace Vh1(Th1,P1); // definition of FE space
fespace Vh2(Th2,P1);
Vh1 u1,v2; // definition of the FE function
Vh2 u2,v2;

// use angle bracket "<" ">" for composite
problem compositeThermic(<[u1],[u2]>,<[v1],[v2]>) =
int2d(Th1)(k1*dx(u1)*dx(v1) + k1*dy(u1)*dy(v1))
+ int2d(Th2)(k2*dx(u2)*dx(v2) + k2*dy(u2)*dy(v2))
+ int1d(Th1,10,mortar=1)(c*(u1-u2)*v1)
+ int1d(Th2,10,mortar=1)(c*(u2-u1)*v2)
+ on(3,u1=T0)
+ on(4,u2=T1);

compositeThermic;
// u1 and u2 contains the solution of the problem
```

Example composite FE Space

Thermic contact

conclusion for problem/solve:

- FE space composite allow to avoid penalization in Ω_2 for u_1.
- FE space composite allow to used non conformal mesh for Ω_1 and Ω_2.
  $\Rightarrow$ The only relation between Ω_1 and Ω_2 is due to the coupled interface problem.
- FEspace composite allow to have different kind of physics in different domain.

$\Rightarrow$ Point (2) and (3) are true also for varf.

Example composite FE Space

Stokes example

Consider the following Stokes problem:

$$\begin{cases} -\Delta u + \nabla p = f & \text{in } \Omega \\ -\text{div} u = 0 & \text{in } \Omega \\ u = u_{\text{D}} & \text{on } \Gamma_D \end{cases}$$

- The variationnal form is given by:

$$\forall (v, q), \begin{cases} \int_\Omega \nabla u . \nabla v \, dX - \int_\Omega p div(v) dX = \int_\Omega f \cdot v \, dX, \\ -\int_\Omega div(u) q dX - \int_\Omega \epsilon p \, q = 0. \end{cases}$$

- (u, p) are solution $\Rightarrow$ $(u, p + C))$ is also a solution.
- stabilized term (blue term) allow to have only one solution $\epsilon = 1e - 10$.

Composite FE Space

Stokes example

- variational form:

$$\forall (v,q), \begin{cases} \int_\Omega \nabla u.\nabla v \, dX - \int_\Omega p div(v) dX = \int_\Omega f \cdot v \, dX, \\ \quad - \int_\Omega div(u) q dX - \int_\Omega \epsilon p \, q = 0. \end{cases}$$

- stabilized term (blue term) allow to have only one solution $\epsilon = 1e-10$.
- The choice of the fespace for u and p must be chosen carefully (LBB condition)

$$\forall u \in U_h, \quad \sup_{p \in P_h} \frac{\int_\Omega \nabla p.v dX}{\|p\|_{P_h}} \geq \beta \|u\|_{U_h},$$

- $T_h^u = T_h^p$ and P1(u)-P1(p) FE doesn't work.
- $T_h^u = T_h^p$ and P2-P1 FE (Taylor-Hood Element)

Composite FE Space

Stokes Example : driven cavity

- The driven cavity is a standard test. It is a box full of liquid with its lid moving horizontally at speed one.
- upper boundary $u = (1,0)$ and otherwise $u = (0,0)$.
- we have also added a hole inside with zero velocity at the boundary.

composite_stokes_cavite_entrainee.edp

Conclusion:

- Composite FE space allow to use different mesh for the same domain.

Composite FE Space

Stokes Example : periodic in one direction.

$$\begin{cases} -\Delta u + \nabla p = f & \text{in } \Omega \\ -\text{div} u = 0 & \text{in } \Omega \end{cases}$$

- In this test, we suppose that we know the solution:
- $u = (sin(x) * cos(y), -cos(x) * sin(y))$
- $p = 2 * cos(x) * cos(y)$
- $f = (0, -4 * cos(x) * sin(y))$

composite_stokes_periodic_problem.edp

Conclusion:

- Composite FE space allow to use a periodic condition in one direction.

FEM-BEM coupling

Model problem

Consider the following Helmholtz problem:

$$\begin{cases} -\Delta u - n(\boldsymbol{x})k^2 u = f & \text{in } \Omega_{\text{int}} \\ -\Delta u_{\text{ext}} - n_{\text{ext}} k^2 u_{\text{ext}} = 0 & \text{in } \Omega_{\text{ext}} \\ u = u_{\text{ext}} & \text{on } \Gamma \\ \nabla u \cdot \boldsymbol{n} = \nabla u_{\text{ext}} \cdot \boldsymbol{n} & \text{on } \Gamma \\ + \text{ radiation condition} \end{cases}$$

Variational form in Ω_{int}: $\forall v$,

$$\int_{\Omega_{\text{int}}} \nabla u \nabla v - \int_{\Omega_{\text{int}}} n k^2 u v - \int_{\Gamma} \nabla u \cdot \boldsymbol{n} v = \int_{\Omega_{\text{int}}} f v$$

FEM-BEM coupling

Bielak-MacCamy formulation

$$\int_{\Omega_{\text{int}}} \nabla u \nabla v - \int_{\Omega_{\text{int}}} nk^2 uv - \int_{\Gamma} \nabla u_{\text{ext}} \cdot \boldsymbol{n} v = \int_{\Omega_{\text{int}}} fv$$

Bielak-MacCamy indirect coupling: find ansatz ϕ s.t. $u_{\text{ext}} = SL(\phi)$ in Ω_{ext}

We have

$$\begin{cases} \gamma_D(u) = \gamma_D(u_{\text{ext}}) & = \gamma_D \circ SL(\phi) \\ \nabla u \cdot \boldsymbol{n} = \nabla u_{\text{ext}} \cdot \boldsymbol{n} & = -\gamma_N \circ SL(\phi) \end{cases}$$

with γ_D and γ_N the Dirichlet and Neumann trace operators.

Thus, the coupled problem is: find (u, ϕ) such that $\forall (v, \psi)$

$$\begin{cases} \displaystyle\int_{\Omega_{\text{int}}} \nabla u \nabla v - \int_{\Omega_{\text{int}}} nk^2 uv + \int_{\Gamma} \gamma_N \circ SL(\phi) v = \int_{\Omega_{\text{int}}} fv \\ \displaystyle\int_{\Gamma} \gamma_D(u)\psi - \int_{\Gamma} \gamma_D \circ SL(\phi)\psi = 0 \end{cases}$$

FEM-BEM coupling

Bielak-MacCamy formulation

$$\begin{cases} \int_{\Omega_{\text{int}}} \nabla u \nabla v - \int_{\Omega_{\text{int}}} nk^2 uv + \int_\Gamma \gamma_N \circ SL(\phi) v = \int_{\Omega_{\text{int}}} fv \\ \int_\Gamma \gamma_D(u)\psi - \int_\Gamma \gamma_D \circ SL(\phi)\psi = 0 \end{cases}$$

In the current version of Freefem, no mixed formulation in FreeFEM: assemble blocks separately

⇒ With Composite FE spaces: only one coupled variational form

```
varf Lenses([uf,ub],[vf,vb]) =
  int2d(Th)((-k^2*uf*vf+Grad(uf)'*Grad(vf))) // Fem
+ int1dx1d(ThL)(ThL)(BEM(-1*BemKernel("SL",k=k),ub,vb)) // -SL
+ int1d(ThL)(uf*vb) // Mass
+ int1d(ThL)(0.5*ub*vf) + int1dx1d(ThL)(ThL)(BEM(BemKernel(
  "TDL",k=k),ub,vf)); // TDL
```

Note: the bilinear form associated to the *transpose double layer operator* is

$$a_{K'}(\phi, v) = \frac{1}{2}\int_\Gamma \phi v + \int_\Gamma K'(\phi) v := \int_\Gamma (\gamma_N \circ SL)(\phi) v$$

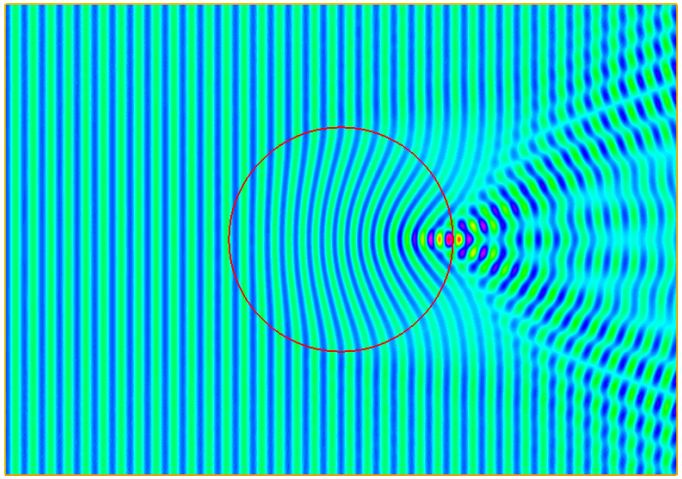
FEM-BEM coupling

Application: gradient index lenses

Wave focusing and bending using gradient index lenses

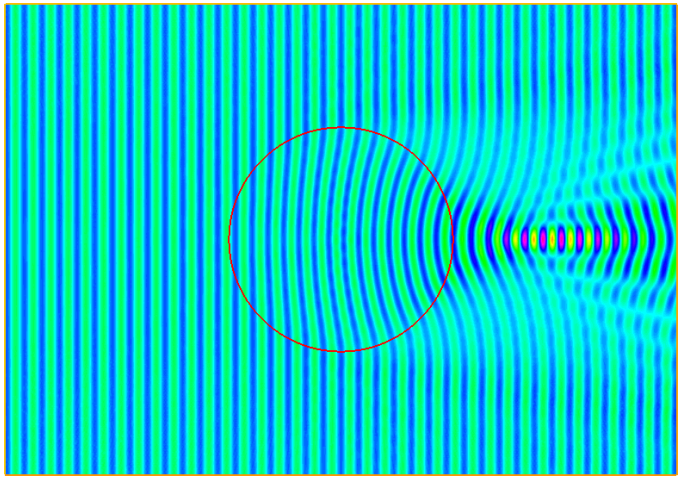
Maxwell's fish-eye lens (1854)

$$n(r) = \frac{2}{1 + \left(\frac{r}{R}\right)^2}$$



Luneburg lens (1944)

$$n(r) = \sqrt{2 - \left(\frac{r}{R}\right)^2}$$



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

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

- 35 lines for the definition of the matrix and the rhs.
- Matrix system is

$$\begin{pmatrix} F & TDL \\ mass & -SL \end{pmatrix} \begin{pmatrix} u_{\text{fem}} \\ u_{\text{bem}} \end{pmatrix} = \begin{pmatrix} F_{\text{rhs}} \\ 0 \end{pmatrix} \quad \text{wheresol} = \begin{pmatrix} u_{\text{fem}} \\ u_{\text{bem}} \end{pmatrix}.$$

```
fespace Uh(Th,P1);  // Th is a 2d volumic mesh
fespace UhL(ThL,P1); // ThL is a meshL
fespace CUh(<Uh,UhL>); // CUh = Uh × UhL

varf Lenses([ufem,ubem],[vfem,vbem]) =
int2d(Th)((-ind*k^2*ufem*vfem+Grad(ufem)'*Grad(vfem))) // F
+ int1dx1d(ThL)(ThL)(BEM(BemKernel("TDL",k=k),ubem,vfem)) // TDL
      + int1d(ThL)(0.5*ubem*vfem) // TDL
+ int1d(ThL)(ufem*vbem)      // mass
+ int1dx1d(ThL)(ThL)(BEM(-1*BemKernel("SL",k=k),ubem,vbem)) // -SL
+ int2d(Th)(finc*vfem) + on(waveguide,ufem=0) ; // RHS

matrix<complex> A = Lenses(CUh,CUh);
complex[int]  rhs = Lenses(0,CUh);
```

Composite FE space

Variational form

```
fespace Uh(Th,P1);  // Th is a 2d volumic mesh
fespace UhL(ThL,P1); // ThL is a meshL
fespace CUh(<Uh,UhL>); // CUh = Uh × UhL
...
varf Lenses([ufem,ubem],[vfem,vbem]) = ...

matrix<complex> A = Lenses(CUh,CUh);
complex[int]  rhs = Lenses(0,CUh);

complex[int] sol = A^-1*rhs;
```

- In the version in development, FE function of composite FE space is not defined:

```
CUh [ufem,ubem];  // compile error
```

- As $CUh = Uh \times UhL$, we have $sol = \begin{pmatrix} u_{\text{fem}} \\ u_{\text{bem}} \end{pmatrix}$.

```
Uh<complex>  ufem;  // ufem
UhL<complex> ubem;  // ubem
// method 1:  extract solution
ufem[] = sol(0:(Uh.ndof)-1);      // extract ufem
ubem[] = sol(Uh.ndof:(sol.n-1)); // extract ubem
// method 2:  dispatch solution
[ufem[], ubem[]] = sol;
```

Ultimate ~~Dream~~ Goal:

run any sequential script in parallel on N processes $\Rightarrow$ same result, fully distributed data structures (completely hidden to the user), speedup of N

What we are working on for now:

- Hide the parallelization of the assembly and solution steps (main bottlenecks) as mush as possible
- Generalize the matrix format, using PETSc for now ; useful for handling composite operators stemming from coupled problems (e.g. sparse + $\mathscr{H}-$matrices for FEM-BEM)

Hide parallel assembly, give distributed matrix to MUMPS:

```
load "MUMPS"
solve pb(u,v,solver=sparsesolver,master=-1) = int3d(
    Th)(Grad(u)'*Grad(v)) -int3d(Th)(f*v) +on(1,u=2);
```

or

```
load "MUMPS"
varf pb(u,v) = int3d(Th)(Grad(u)'*Grad(v)) +int3d(Th
    )(f*v) +on(1,u=2);

matrix A = pb(Uh,Uh,solver=sparsesolver,master=-1);
real[int] rhs = pb(0,Uh);

u[] = A^-1*rhs;
```

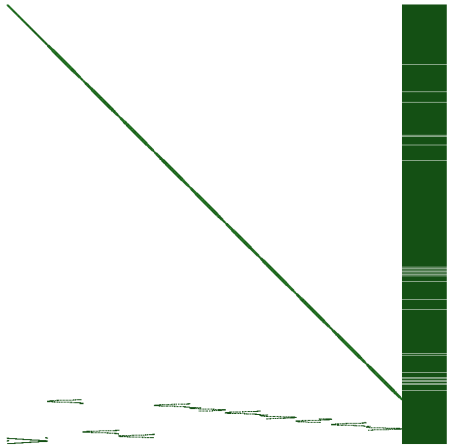
Parallel FreeFEM

PETSc matrices

Also works with composite FE spaces for coupled problems, e.g. FEM-BEM lenses:

```
load "MUMPS"
varf Lenses([ufem,ubem],[vfem,vbem]) =
int2d(Th)((-ind*k^2*ufem*vfem+Grad(ufem)'*Grad(vfem))) // F
+ int1dx1d(ThL)(ThL)(BEM(BemKernel("TDL",k=k),ubem,vfem)) // TDL
      + int1d(ThL)(0.5*ubem*vfem) // TDL
+ int1d(ThL)(u1*v2)      // mass
+ int1dx1d(ThL)(ThL)(BEM(-1*BemKernel("SL",k=k),ubem,vbem)) // -SL
+ int2d(Th)(finc*vfem) + on(waveguide,ufem=0) ; // RHS

matrix<complex> A = Lenses(CUh,CUh,solver=sparsesolver,master=-1);
complex[int] rhs = Lenses(0,CUh);

complex[int] sol = A^-1*rhs;
```

BUT for now, densifies the compressed BEM blocks ($\mathscr{H}$-matrices)
$\Rightarrow$ use PETSc instead, construct a distributed *nested* Mat, each block has its own matrix type



Parallel FreeFEM

PETSc preconditioners

We can then take advantage of the wide range of available solvers in PETSc. In particular, we can easily use *fieldsplit* preconditioners for coupled problems, each variable naturally defining its own *field*:

```
load "PETSc-complex"
varf Lenses([ufem,ubem],[vfem,vbem]) =
int2d(Th)((-ind*k^2*ufem*vfem+Grad(ufem)'*Grad(vfem))) // F
+ int1dx1d(ThL)(ThL)(BEM(BemKernel("TDL",k=k),ubem,vfem)) // TDL
      + int1d(ThL)(0.5*ubem*vfem) // TDL
+ int1d(ThL)(u1*v2)      // mass
+ int1dx1d(ThL)(ThL)(BEM(-1*BemKernel("SL",k=k),ubem,vbem)) // -SL
+ int2d(Th)(finc*vfem) + on(waveguide,ufem=0); // RHS

Mat<complex> A = Lenses(CUh,CUh);
set(A,sparams="-ksp_view␣-ksp_monitor␣-ksp_type␣fgmres
-ksp_view_final_residual␣-ksp_gmres_restart␣200␣-pc_type␣fieldsplit␣
    -fieldsplit_0_pc_type␣asm␣-fieldsplit_0_sub_pc_type␣lu
-fieldsplit_0_ksp_type␣gmres␣-fieldsplit_1_ksp_type␣gmres
-fieldsplit_1_ksp_max_it␣20");

complex[int] rhs = Lenses(0,CUh);
complex[int] sol = A^-1*rhs;
```

Current/Future developments

- use PETSc as a solver directly in problem/solve:

```
solve pb(u,v,solver=petsc,sparams="...");
```

- C++ implementation of Domain Decomposition macros (macro_ddm.idp) to hide difficulties to the user ⇒ will be able to use DD solvers such as GenEO transparently

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