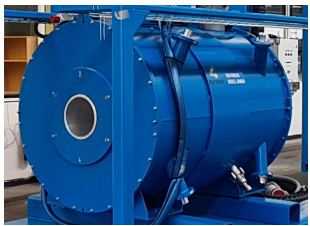
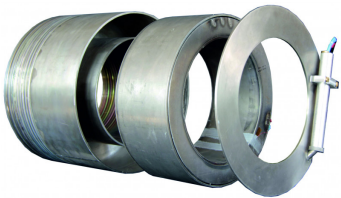
# Superconducting magnet optimisation and modelling of magnetic shields with GetDP

TECHNISCHE UNIVERSITÄT DARMSTADT





Nicolas Marsic
TEMF, TU Darmstadt

GetDP Workshop, 22–23 April 2021

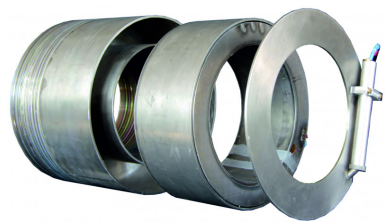
# Menu

**Superconducting confinement magnet**

- 2D magnetostatics and optimization

**Cryogenic current comparator**

- 2.5D magnetostatics



# Menu

**Superconducting confinement magnet**

- 2D magnetostatics and optimization

**Cryogenic current comparator**

- 2.5D magnetostatics

# PUMA in a nutshell I

**This project is part of the PUMA experiment**

PUMA: antiProton Unstable Matter Annihilation

# PUMA in a nutshell I

**This project is part of the PUMA experiment**

PUMA: antiProton Unstable Matter Annihilation

**What does PUMA aim at?**

Measurement of the **neutron/proton ratio** at the **surface** of heavy nuclei

# PUMA in a nutshell I

**This project is part of the PUMA experiment**

PUMA: antiProton Unstable Matter Annihilation

**What does PUMA aim at?**

Measurement of the **neutron/proton ratio** at the **surface** of heavy nuclei

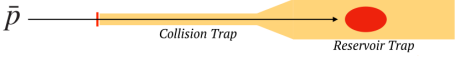
**Why antiprotons?**

High probability that antiprotons interact at the **surface** of a nucleus

# PUMA in a nutshell II

**How will it work?**

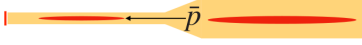
1. Store antiprotons in a reservoir trap



# PUMA in a nutshell II

**How will it work?**

1. Store antiprotons in a reservoir trap
2. Transfer some antiprotons to a collision trap



# PUMA in a nutshell II

**How will it work?**

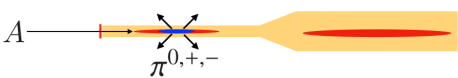
1. Store antiprotons in a reservoir trap
2. Transfer some antiprotons to a collision trap
3. Introduce the heavy nuclei in the collision trap

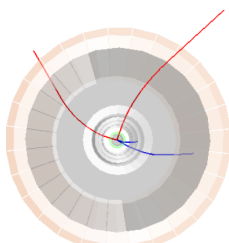
# PUMA in a nutshell II

**How will it work?**

1. Store antiprotons in a reservoir trap
2. Transfer some antiprotons to a collision trap
3. Introduce the heavy nuclei in the collision trap
4. Antiprotons interact with **surface** nucleons ⟶ emission of **pions** (subatomic particles)

The **charge of a pion** tells us if we annihilated a proton or a neutron

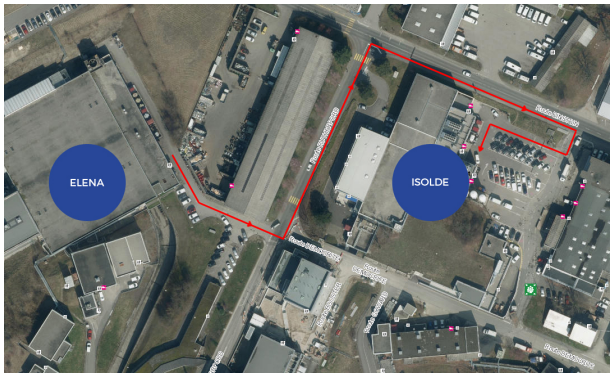


# PUMA in a nutshell III

**Where do we find heavy nuclei and antimatter?**

- At CERN :-) → PUMA has been recently accepted as a new CERN experiment (17/03/2021)
  → **No beamline** allowing collisions between antimatter and heavy nuclei




Transportable antimatter trap

# PUMA in a nutshell IV

**Which question did we try to answer in this project?**

Is it possible to build a **confinement magnet** with both

- active shielding
- passive shielding

given **constraints** on the

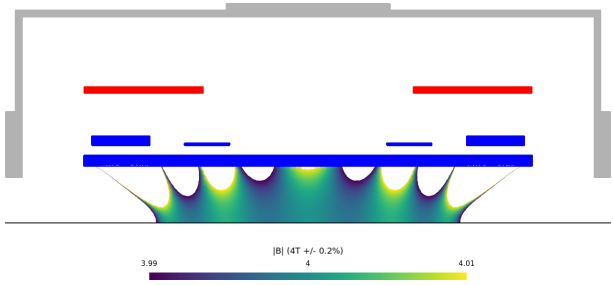
- the weight
- the geometry (bore radius, total length, . . . )
- the ***b***-field (strength, quality, far field)?

→ Lots of iterations with the team designing the experiment :-)!

# PUMA in a nutshell V

## Magnetic pre-design with GetDP

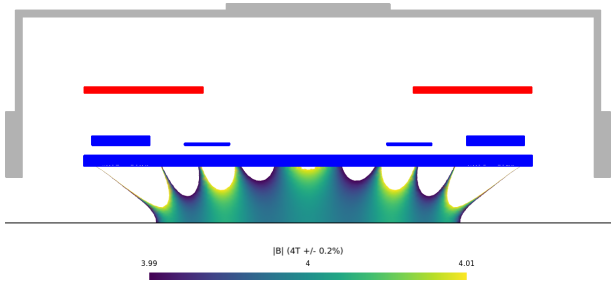
- Axisymmetric 2D magnetostatics



# PUMA in a nutshell V

**Magnetic pre-design with GetDP**

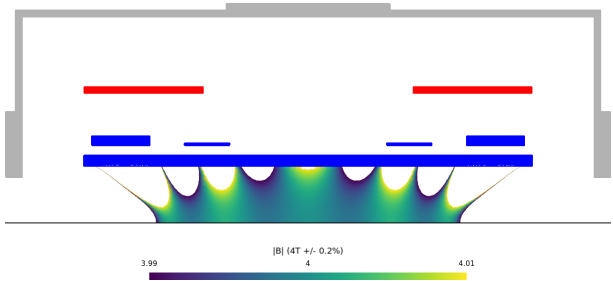
- Axisymmetric 2D magnetostatics
- Optimization problem



# PUMA in a nutshell V

**Magnetic pre-design with GetDP**

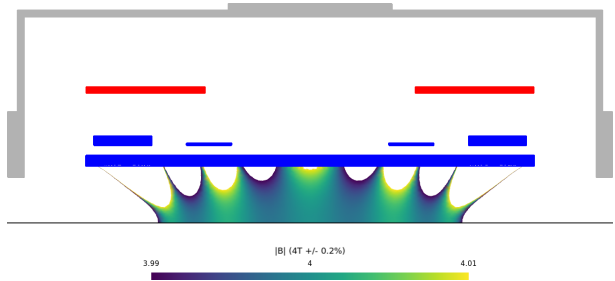
- Axisymmetric 2D magnetostatics
- Optimization problem
  - → As light as possible
  - → As compact as possible



# PUMA in a nutshell V

**Magnetic pre-design with GetDP**

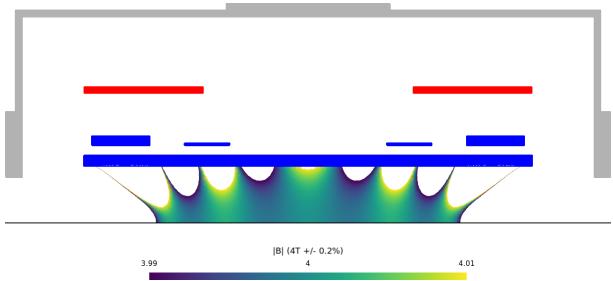
- Axisymmetric 2D magnetostatics
- Optimization problem
  - → As light as possible
  - → As compact as possible
  - → 4T with 2 homog. regions
  - → Low far field



# PUMA in a nutshell V

**Magnetic pre-design with GetDP**

- Axisymmetric 2D magnetostatics
- Optimization problem
  - → As light as possible
  - → As compact as possible
  - → 4T with 2 homog. regions
  - → Low far field
  - → Avoid shield saturation
  - → Quench margin (load line)

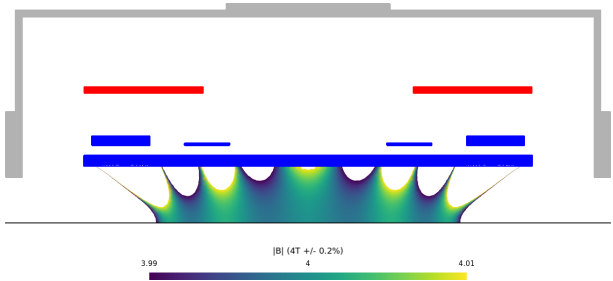


# PUMA in a nutshell V

**Magnetic pre-design with GetDP**

- Axisymmetric 2D magnetostatics
- Optimization problem
  - → As light as possible
  - → As compact as possible
  - → 4T with 2 homog. regions
  - → Low far field
  - → Avoid shield saturation
  - → Quench margin (load line)

25 parameters



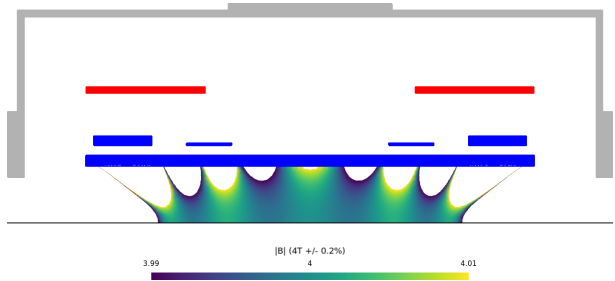
# PUMA in a nutshell V

**Magnetic pre-design with GetDP**

- Axisymmetric 2D magnetostatics
- Optimization problem
  - → As light as possible
  - → As compact as possible
  - → 4T with 2 homog. regions
  - → Low far field
  - → Avoid shield saturation
  - → Quench margin (load line)

25 parameters

- Final design by Bilfinger Noell
  - → Coils and shield: ~1t + ~2t
  - → Total: ~7t




# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$
\begin{aligned}
\tau_{\min} \leq\ & \tau \leq \tau_{\max} \\
\boldsymbol{j}_{s,\min} \leq\ & \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \\
3.992 \leq\ & \boldsymbol{b}_{\mathrm{R1}} \leq 4.008 \\
3.800 \leq\ & \boldsymbol{b}_{\mathrm{R2}} \leq 4.200 \\
& \boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4} \\
& \boldsymbol{b}_{\mathrm{sh}} \leq 1.5 \\
& Ll \leq 0.8 \\
& \boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}
\end{aligned}
$$

# Formulation of the problem

**Find $\tau$ and $j_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (}e.g.\text{ } < 2\text{m long)}$$

$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max}$$

$$3.992 \leq \boldsymbol{b}_{\mathrm{R1}} \leq 4.008$$

$$3.800 \leq \boldsymbol{b}_{\mathrm{R2}} \leq 4.200$$

$$\boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4}$$

$$\boldsymbol{b}_{\mathrm{sh}} \leq 1.5$$

$$LI \leq 0.8$$

$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\begin{aligned}
\tau_{\min} \leq\ & \tau \leq \tau_{\max} && \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)} \\
\boldsymbol{j}_{s,\min} \leq\ & \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} && \longrightarrow \text{Current density in each coil} \\
3.992 \leq\ & \boldsymbol{b}_{\mathrm{R1}} \leq 4.008 \\
3.800 \leq\ & \boldsymbol{b}_{\mathrm{R2}} \leq 4.200 \\
& \boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4} \\
& \boldsymbol{b}_{\mathrm{sh}} \leq 1.5 \\
& \mathit{LI} \leq 0.8 \\
& \boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}
\end{aligned}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\begin{aligned}
\tau_{\min} \leq\ & \tau \leq \tau_{\max} && \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)} \\
\boldsymbol{j}_{s,\min} \leq\ & \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} && \longrightarrow \text{Current density in each coil} \\
3.992 \leq\ & \boldsymbol{b}_{\mathrm{R1}} \leq 4.008 && \longrightarrow \text{High homogeneity region } (\pm 0.2\%) \\
3.800 \leq\ & \boldsymbol{b}_{\mathrm{R2}} \leq 4.200 \\
& \boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4} \\
& \boldsymbol{b}_{\mathrm{sh}} \leq 1.5 \\
& LI \leq 0.8 \\
& \boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}
\end{aligned}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)}$$
$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \longrightarrow \text{Current density in each coil}$$
$$3.992 \leq \boldsymbol{b}_{\text{R1}} \leq 4.008 \longrightarrow \text{High homogeneity region } (\pm 0.2\%)$$
$$3.800 \leq \boldsymbol{b}_{\text{R2}} \leq 4.200 \longrightarrow \text{Low homogeneity region } (\pm 5\%)$$
$$\boldsymbol{b}_{\text{3m}} \leq 2 \times 10^{-4}$$
$$\boldsymbol{b}_{\text{sh}} \leq 1.5$$
$$LI \leq 0.8$$
$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)}$$
$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \longrightarrow \text{Current density in each coil}$$
$$3.992 \leq \boldsymbol{b}_{\mathrm{R1}} \leq 4.008 \longrightarrow \text{High homogeneity region } (\pm 0.2\%)$$
$$3.800 \leq \boldsymbol{b}_{\mathrm{R2}} \leq 4.200 \longrightarrow \text{Low homogeneity region } (\pm 5\%)$$
$$\boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4} \longrightarrow \text{Field at 3m from the magnet}$$
$$\boldsymbol{b}_{\mathrm{sh}} \leq 1.5$$
$$LI \leq 0.8$$
$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)}$$
$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \longrightarrow \text{Current density in each coil}$$
$$3.992 \leq \boldsymbol{b}_{\text{R1}} \leq 4.008 \longrightarrow \text{High homogeneity region } (\pm 0.2\%)$$
$$3.800 \leq \boldsymbol{b}_{\text{R2}} \leq 4.200 \longrightarrow \text{Low homogeneity region } (\pm 5\%)$$
$$\boldsymbol{b}_{\text{3m}} \leq 2 \times 10^{-4} \longrightarrow \text{Field at 3m from the magnet}$$
$$\boldsymbol{b}_{\text{sh}} \leq 1.5 \longrightarrow \text{Avoid shield saturation}$$
$$LI \leq 0.8$$
$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)}$$
$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \longrightarrow \text{Current density in each coil}$$
$$3.992 \leq \boldsymbol{b}_{\mathrm{R1}} \leq 4.008 \longrightarrow \text{High homogeneity region } (\pm 0.2\%)$$
$$3.800 \leq \boldsymbol{b}_{\mathrm{R2}} \leq 4.200 \longrightarrow \text{Low homogeneity region } (\pm 5\%)$$
$$\boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4} \longrightarrow \text{Field at 3m from the magnet}$$
$$\boldsymbol{b}_{\mathrm{sh}} \leq 1.5 \longrightarrow \text{Avoid shield saturation}$$
$$Il \leq 0.8 \longrightarrow \text{Quench margin (load line criterion) @6K}$$
$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a}$$

# Formulation of the problem

**Find $\tau$ and $\boldsymbol{j}_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)}$$
$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \longrightarrow \text{Current density in each coil}$$
$$3.992 \leq \boldsymbol{b}_{\text{R1}} \leq 4.008 \longrightarrow \text{High homogeneity region } (\pm 0.2\%)$$
$$3.800 \leq \boldsymbol{b}_{\text{R2}} \leq 4.200 \longrightarrow \text{Low homogeneity region } (\pm 5\%)$$
$$\boldsymbol{b}_{\text{3m}} \leq 2 \times 10^{-4} \longrightarrow \text{Field at 3m from the magnet}$$
$$\boldsymbol{b}_{\text{sh}} \leq 1.5 \longrightarrow \text{Avoid shield saturation}$$
$$Ll \leq 0.8 \longrightarrow \text{Quench margin (load line criterion) @6K}$$
$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a} \longrightarrow \text{GetDP!}$$

# Formulation of the problem

**Find $\tau$ and $j_s$ that minimize $w$ such that:** [SI units]

$$
\begin{aligned}
\tau_{\min} \leq \quad \tau \leq \tau_{\max} &\longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)} \\
\boldsymbol{j}_{s,\min} \leq \quad \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} &\longrightarrow \text{Current density in each coil} \\
3.992 \leq \boldsymbol{b}_{\mathrm{R1}} \leq 4.008 &\longrightarrow \text{High homogeneity region } (\pm 0.2\%) \\
3.800 \leq \boldsymbol{b}_{\mathrm{R2}} \leq 4.200 &\longrightarrow \text{Low homogeneity region } (\pm 5\%) \\
\boldsymbol{b}_{\mathrm{3m}} \leq 2 \times 10^{-4} &\longrightarrow \text{Field at 3m from the magnet} \\
\boldsymbol{b}_{\mathrm{sh}} \leq 1.5 &\longrightarrow \text{Avoid shield saturation} \\
LI \leq 0.8 &\longrightarrow \text{Quench margin (load line criterion) @6K} \\
\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a} &\longrightarrow \text{GetDP!}
\end{aligned}
$$

**With $\boldsymbol{a} \in H_0(\mathbf{curl}, \Omega)$ such that:**

$$\int_\Omega \nu\, \mathbf{curl}\, \boldsymbol{a} \cdot \mathbf{curl}\, \boldsymbol{a}' \mathrm{d}\Omega - \int_\Omega \boldsymbol{j}_s \cdot \boldsymbol{a}' \mathrm{d}\Omega = 0 \quad \forall \boldsymbol{a}' \in H_0(\mathbf{curl}, \Omega)$$

# Formulation of the problem

**Find $\tau$ and $j_s$ that minimize $w$ such that:** [SI units]

$$\tau_{\min} \leq \tau \leq \tau_{\max} \longrightarrow \text{Geometrical parameters (\textit{e.g.} } < \text{2m long)}$$
$$\boldsymbol{j}_{s,\min} \leq \boldsymbol{j}_s \leq \boldsymbol{j}_{s,\max} \longrightarrow \text{Current density in each coil}$$
$$3.992 \leq \boldsymbol{b}_{\text{R1}} \leq 4.008 \longrightarrow \text{High homogeneity region } (\pm 0.2\%)$$
$$3.800 \leq \boldsymbol{b}_{\text{R2}} \leq 4.200 \longrightarrow \text{Low homogeneity region } (\pm 5\%)$$
$$\boldsymbol{b}_{\text{3m}} \leq 2 \times 10^{-4} \longrightarrow \text{Field at 3m from the magnet}$$
$$\boldsymbol{b}_{\text{sh}} \leq 1.5 \longrightarrow \text{Avoid shield saturation}$$
$$LI \leq 0.8 \longrightarrow \text{Quench margin (load line criterion) @6K}$$
$$\boldsymbol{b} = \mathbf{curl}\, \boldsymbol{a} \longrightarrow \text{GetDP!}$$

**With $\boldsymbol{a} \in H_0(\mathbf{curl}, \Omega)$ such that:**

$$\int_\Omega \nu\, \mathbf{curl}\, \boldsymbol{a} \cdot \mathbf{curl}\, \boldsymbol{a}' \mathrm{d}\Omega - \int_\Omega \boldsymbol{j}_s \cdot \boldsymbol{a}' \mathrm{d}\Omega = 0 \quad \forall \boldsymbol{a}' \in H_0(\mathbf{curl}, \Omega)$$

Note that we assume that each coil has its **own power source**

- ⟶ Some technological details were unknown (*e.g.* cable cross section, fill factor, …)
- ⟶ Optimizing for the same current in each coil is also possible

# GetDP: Jacobian and 2D axisymmetric problems

```
Jacobian{
  { Name Jac; Case{ { Region All; Jacobian VolAxiSqu; } } }
}
```

**VolAxiSqu**

- Cylindrical coordinate system $(x, y, z) \rightarrow (r, z, \theta)$
- together with the transformation $r \rightarrow \sqrt{\rho}$

Leads to $\lim_{\rho \rightarrow 0} \det J \rightarrow \frac{1}{2}$ instead of $\lim_{r \rightarrow 0} \det J \rightarrow 0$

[Henrotte *et al.*, 1999]

# GetDP: *bh* law

```
Function{
  mu0 = 4 * Pi * 1e-7;

  nu[NonMagnetic] = 1/mu0;
  nu[Magnetic]    = nuMag[$1]; // $1 = b
  dnu[Magnetic]   = 2 * dNudB2[$1] * SquDyadicProduct[$1];
}
```

- Differential reluctivity for Newton-Raphson: $\delta\boldsymbol{\nu} = 2\dfrac{\partial \nu}{\partial \|\boldsymbol{b}\|^2}\boldsymbol{b} \otimes \boldsymbol{b}$
- nuMag[ ] and dNudB2[ ] obtained by **linear interpolation** of a set of points $(\nu, \|\boldsymbol{b}\|^2)$ (list NuB2)
  - → `InterpolationLinear[SquNorm[$1]]{List[NuB2]}` for nuMag
  - → `dInterpolationLinear[SquNorm[$1]]{List[NuB2]}` for dNudB2

# GetDP: FunctionSpace for *a*

```
FunctionSpace{
  { Name HCurl2D; Type Form1P; // "1-form perpendicular to the domain"
    BasisFunction{
      { [...]; Function BF_PerpendicularEdge;    Entity NodesOf[All]; }
      { [...]; Function BF_PerpendicularEdge_2E; Entity EdgesOf[All]; }
    }
    Constraint{ [...] }
  }
}
```

# GetDP: Magnetostatic formulation (vector potential)

```
Formulation{
  { Name MVP; Type FemEquation;
    Quantity{ { Name a; Type Local; NameOfSpace HCurl2D; } }
    Equation{
      Galerkin{ [nu[{d a}]*Dof{d a}, {d a}];       [...] }
      Galerkin{ [-js[], {a}];                      [...] }
      // Jacobian for Newton-Raphson
      Galerkin{ JacNL[dnu[{d a}]*Dof{d a}, {d a}]; [...] }
    }
  }
}
```

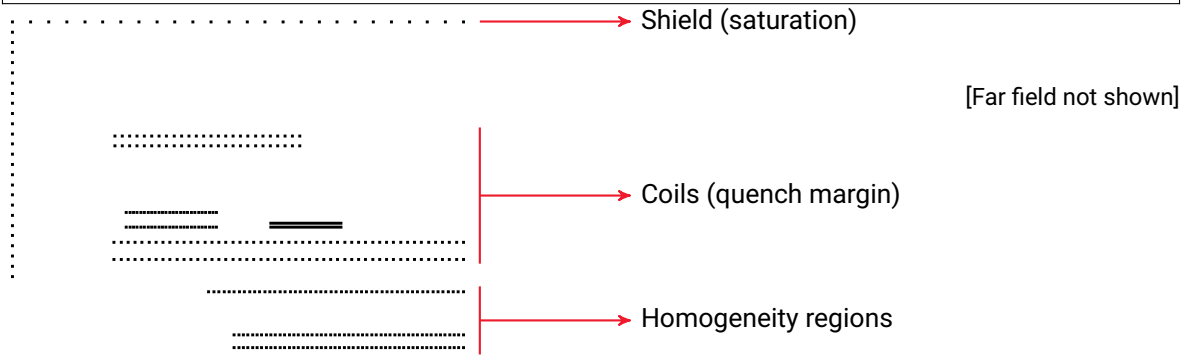
Find $\boldsymbol{a} \in H_0(\mathbf{curl}, \Omega)$ such that:

$$\int_\Omega \nu\, \mathbf{curl}\, \boldsymbol{a} \cdot \mathbf{curl}\, \boldsymbol{a}' \mathrm{d}\Omega - \int_\Omega \boldsymbol{j}_s \cdot \boldsymbol{a}' \mathrm{d}\Omega = 0 \quad \forall \boldsymbol{a}' \in H_0(\mathbf{curl}, \Omega)$$

One-to-one correspondence !

# GetDP: Post-processing

```
PostOperation MVP UsingPost MVP{
  For i In {1:MTot} // For each sampling line: bMag = Norm[{d a}]
    Print[bMag, OnLine{{/*start point*/}{/*end point*/}{/*# of samples*/},
          File Sprintf["bMag%g.txt",i], Format SimpleTable];
  EndFor
```



# Where were we?

We can now compute ***b*** :-)

**Do we have everything we need to optimize then?**

- We used a **gradient-based** optimizer: (GC)MMA
- Implementation form the `conveks` library (part of `ONELAB`) [Kuci *et al.*]

  ⟶ We need the **sensitivity** of ***b*** with respect to $\tau$ and $\boldsymbol{j}_s$

Sensitivity analysis [Kuci *et al.*, 2017]

# Sensitivity analysis I

**Finite differences**

1. We **perturb** each design variable
2. **For each** perturbed design variable, we carry out a full **nonlinear** solve
3. We extract the sensitivity with a finite difference scheme

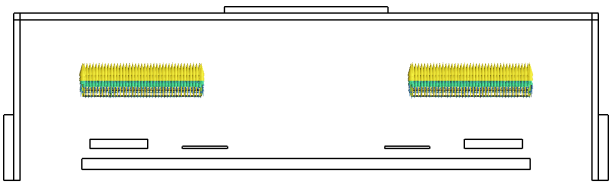
Easy to implement but slow

# Sensitivity analysis II

**Lie derivatives**

1. We define a **velocity field** for each design variable

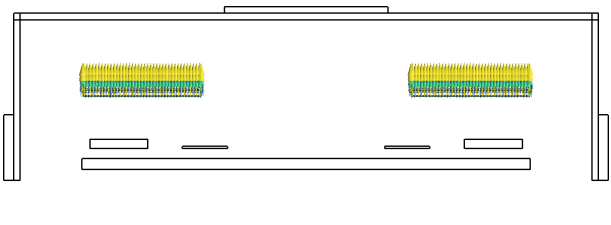
   ⟶ How does the domain move when perturbing a variable?



# Sensitivity analysis II

**Lie derivatives**

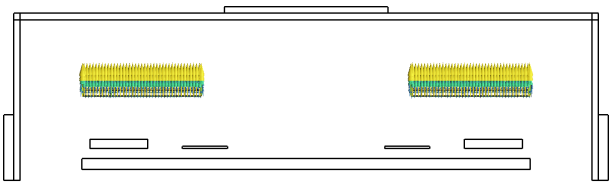
1. We define a **velocity field** for each design variable
   → How does the domain move when perturbing a variable?
2. We solve a **finite element problem** to extract the derivative



# Sensitivity analysis II

**Lie derivatives**

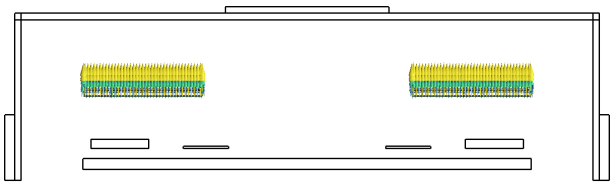
1. We define a **velocity field** for each design variable
   → How does the domain move when perturbing a variable?
2. We solve a **finite element problem** to extract the derivative
3. **Same system matrix** as the one obtained at the last Newton iteration when computing ***a***
   → LU factorization can be **reused**



# Sensitivity analysis II

**Lie derivatives**

1. We define a **velocity field** for each design variable
   → How does the domain move when perturbing a variable?
2. We solve a **finite element problem** to extract the derivative
3. **Same system matrix** as the one obtained at the last Newton iteration when computing ***a***
   → LU factorization can be **reused**



Fast sensitivity analysis!

# GetDP: sensitivity analysis I

1. We compute the velocity field $\boldsymbol{v}$ associated with each parameter $\tau_i$
   → Done by Gmsh: one `.msh` file per **perturbation** of $\tau_i$
2. New `FunctionSpace` (same as $\boldsymbol{a}$)
3. New `Formulation`

# GetDP: sensitivity analysis II

```
Resolution{
  { Name Optimization;
    System{
      { Name MVP; NameOfFormulation MVP; } // Compute b = curl a
      { Name LIE; NameOfFormulation LIE; } // Sensitivity analysis
    }
    Operation{
      // MVP //
      /////////
      InitSolution[MVP];
      IterativeLoop[NL_maxit, NL_stop, NL_relax]{ // Newton-Raphson
        GenerateJac[MVP];
        SolveJac[MVP];
      }
      PostOperation[MVP]; // Evaluate bMag
      [...]
```

# GetDP: sensitivity analysis III

```
[...]
// Sensitivity analysis //
//////////////////////////
InitSolution[LIE];

// Shape variables
For i In {0:NShapeVariables-1}
  GmshRead[Sprintf["velocity_%g.msh", i], Vi]; // Velocity field i
  Generate[LIE];                               // Only RHS relevant
  SolveAgainWithOther[LIE, MVP];               // Reuse LU of MVP
  PostOperation[Get_Lie~{i}];                  // Evaluate d(B)
EndFor
[...]
```

→ With design variables that are not related to shape ($J_s$): $\boldsymbol{v} = \mathbf{0}$

# Where were we?

**We can now carry out the sensitivity analysis :-)**

Let’s glue everything together!

# Optimizer/GetDP/Gmsh coupling I

**Overview**

```
c      = onelab.client(__file__)                          # New client
design = getDesignVariables(c.getAllParameters())         # Design variables
solver = Solver(c, design, 'magnet.geo', 'magnet.pro') # Homemade wrapper
[...]
```

# Optimizer/GetDP/Gmsh coupling I

**Overview**

```
c      = onelab.client(__file__)                          # New client
design = getDesignVariables(c.getAllParameters())         # Design variables
solver = Solver(c, design, 'magnet.geo', 'magnet.pro') # Homemade wrapper
[...]
while it <= maxIter:
  solver.setVariables(conveks.mma.getCurrentPoint()) # New var. from opt.
```

# Optimizer/GetDP/Gmsh coupling I

**Overview**

```
c      = onelab.client(__file__)                          # New client
design = getDesignVariables(c.getAllParameters())         # Design variables
solver = Solver(c, design, 'magnet.geo', 'magnet.pro') # Homemade wrapper
[...]
while it <= maxIter:
  solver.setVariables(conveks.mma.getCurrentPoint()) # New var. from opt.
  solver.mesh()     # Mesh
  solver.velocity() # Velocity fields (mesh perturbation)
  solver.solve()    # GetDP (B-field and sensitivity)
```

# Optimizer/GetDP/Gmsh coupling I

**Overview**

```
c      = onelab.client(__file__)                          # New client
design = getDesignVariables(c.getAllParameters())         # Design variables
solver = Solver(c, design, 'magnet.geo', 'magnet.pro') # Homemade wrapper
[...]
while it <= maxIter:
  solver.setVariables(conveks.mma.getCurrentPoint()) # New var. from opt.
  solver.mesh()     # Mesh
  solver.velocity() # Velocity fields (mesh perturbation)
  solver.solve()    # GetDP (B-field and sensitivity)

  obj   = solver.objective()    # Objective function (weight)
  dObj  = solver.dObjective()   # Derivatives of objective function
  cons  = solver.constraints()  # Constraints (<0: OK / >0: KO)
  dCons = solver.dConstraints() # Derivatives of constraints
```

# Optimizer/GetDP/Gmsh coupling I

**Overview**

```
c      = onelab.client(__file__)                          # New client
design = getDesignVariables(c.getAllParameters())         # Design variables
solver = Solver(c, design, 'magnet.geo', 'magnet.pro') # Homemade wrapper
[...]
while it <= maxIter:
  solver.setVariables(conveks.mma.getCurrentPoint()) # New var. from opt.
  solver.mesh()     # Mesh
  solver.velocity() # Velocity fields (mesh perturbation)
  solver.solve()    # GetDP (B-field and sensitivity)

  obj   = solver.objective()    # Objective function (weight)
  dObj  = solver.dObjective()   # Derivatives of objective function
  cons  = solver.constraints()  # Constraints (<0: OK / >0: KO)
  dCons = solver.dConstraints() # Derivatives of constraints

  conveks.mma.updateCurrentPoint(obj, dObj, cons, dCons) # Call optimizer
```

# Optimizer/GetDP/Gmsh coupling II

**A few useful methods of `Solver`**

```
def solve(self):
  """Call GetDP"""
  self.c.runSubClient('myGetDP', 'getdp magnet.pro -solve Optimization')

def mesh(self):
  """Call Gmsh (need -parametric for mesh perturbation)"""
  self.c.runSubClient('myGmsh', 'gmsh -parametric magnet.geo -')

def objective(self):
  """Objective function (computed by Gmsh, transferred via ONELAB)"""
  return self.c.getNumber('Optimization/Objective')
```

→ Same ideas for `dObjective`, `constraints` and `dConstraints`

→ Constraints handled with text files instead of ONELAB

# Optimizer/GetDP/Gmsh coupling III

**User data file**

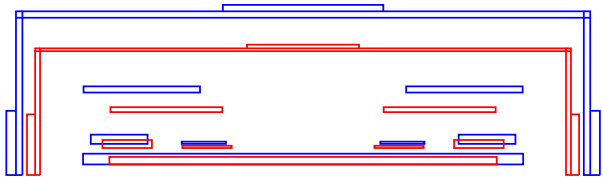
```
mm = 1e-3;
DefineConstant[
[...]
  H5 = { 20*mm, Name "0Geo/SH5", Opti "1", OptiLw "5e-3", OptiUp "*2" }
[...]
];
```

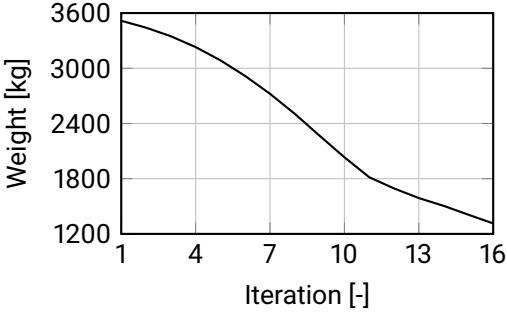
User defined attributes: `Opti`, `OptiLw` and `OptiUp`

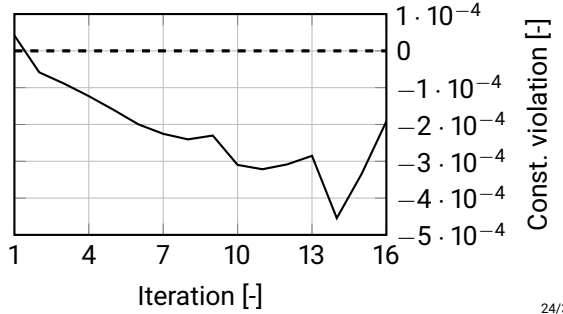
- `c.getAllParameters()`: fetches all variables and their attributes (pre- and user-defined)
- `getDesignVariables(...)`: homemade parser (regular expressions)

# Optimizer: example

**Example test case: optimization for a field magnitude of 2T instead of 4T**



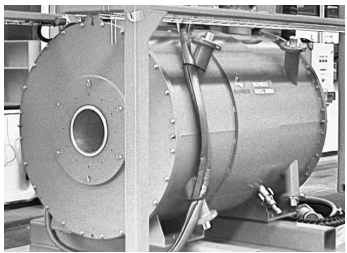


# Menu

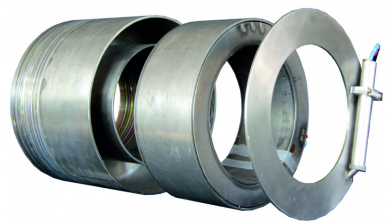
**Superconducting confinement magnet**

- 2D magnetostatics and optimization



**Cryogenic current comparator**

- 2.5D magnetostatics



# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy

# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

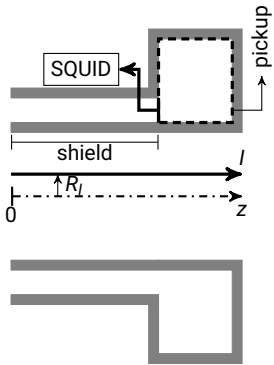
- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

Its key components are a magnetometer and a superconducting shield

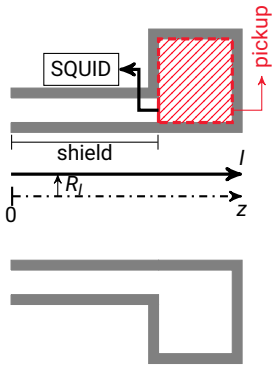


# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

Its key components are a magnetometer and a superconducting shield



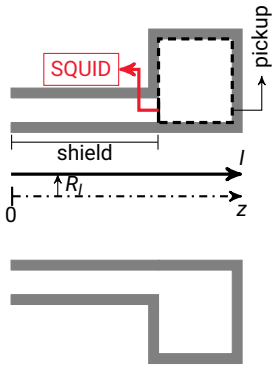

- Magnetometer
  - → Superconducting pickup coil (flux transformer)

# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

Its key components are a magnetometer and a superconducting shield



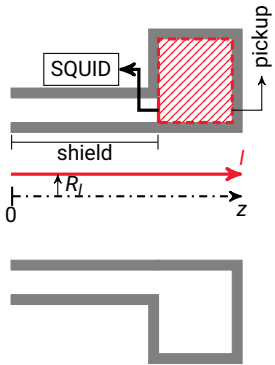

- Magnetometer
  - → Superconducting pickup coil (flux transformer)
  - → Superconducting Quantum Interference Device

# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

Its key components are a magnetometer and a superconducting shield



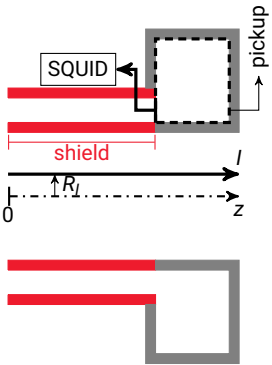

- Magnetometer
  - → Superconducting pickup coil (flux transformer)
  - → Superconducting Quantum Interference Device
  - → Measures the ***b***-field induced by the current

# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

Its key components are a magnetometer and a superconducting shield



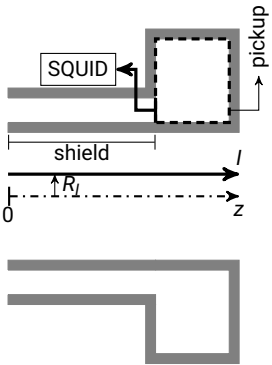

- Magnetometer
  - → Superconducting pickup coil (flux transformer)
  - → Superconducting Quantum Interference Device
  - → Measures the ***b***-field induced by the current
- Shield acts as a magnetic filter
  - → Damps all components of ***b*** but the azimuthal one
  - → We see ***b*** as if *I* was located on the symmetry axis

# CCC: cryogenic current comparator

**What is a cryogenic current comparator (CCC)?**

- One of the most sensitive instrument to measure **very low electric currents** with high accuracy
- Diagnostics device: @CERN AD (in use), @FAIR (planned), ...

Its key components are a magnetometer and a superconducting shield



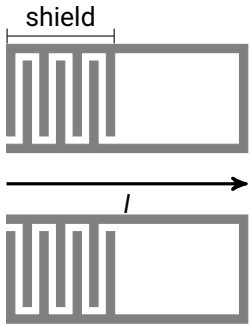

- Magnetometer
  - → Superconducting pickup coil (flux transformer)
  - → Superconducting Quantum Interference Device
  - → Measures the ***b***-field induced by the current
- Shield acts as a magnetic filter
  - → Damps all components of ***b*** but the azimuthal one
  - → We see ***b*** as if *I* was located on the symmetry axis

Good position independence property

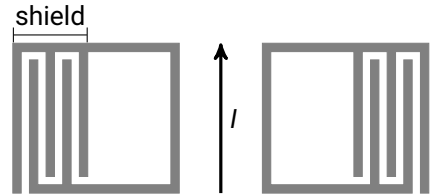
# CCC: folded coaxial shield

**For the CCC to be compact, the shield is usually folded into a meander structure**

Ring CCC




Folded coax CCC




Unlike the ring design, the damping of the folded coax is theoretically unknown*

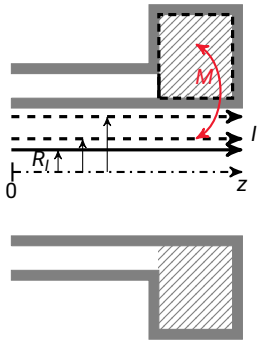
* to the best of our knowledge

→ Let's build a FE model :-)!

# Why 2.5D FEM?

**How can we determine the shield performance?**

→ Test different position of the current carrying wire and compute the mutual inductance *M*




**Can we exploit symmetries?**

→ The CCC is axisymmetric

→ But the excitation (the current carrying wire) is not

Use 2.5D modeling

# 2.5D FEM in a nutshell

## Main idea

⟶ Decompose the full 3D field into a **Fourier series** along the **azimuthal direction** $\varphi$

$$\boldsymbol{a}(r,\varphi,z) = \begin{bmatrix} a_r^0(r,z) \\ a_\varphi^0(r,z) \\ a_z^0(r,z) \end{bmatrix} + \sum_{n=1}^{\infty} \left( \begin{bmatrix} a_r^n(r,z)\cos(n\varphi) \\ a_\varphi^n(r,z)\sin(n\varphi) \\ a_z^n(r,z)\cos(n\varphi) \end{bmatrix} + \begin{bmatrix} a_r^{-n}(r,z)\sin(n\varphi) \\ a_\varphi^{-n}(r,z)\cos(n\varphi) \\ a_z^{-n}(r,z)\sin(n\varphi) \end{bmatrix} \right)$$

# 2.5D FEM in a nutshell

**Main idea**

→ Decompose the full 3D field into a **Fourier series** along the **azimuthal direction** $\varphi$

$$\boldsymbol{a}(r,\varphi,z) = \begin{bmatrix} a_r^0(r,z) \\ a_\varphi^0(r,z) \\ a_z^0(r,z) \end{bmatrix} + \sum_{n=1}^{\infty} \left( \begin{bmatrix} a_r^n(r,z)\cos(n\varphi) \\ a_\varphi^n(r,z)\sin(n\varphi) \\ a_z^n(r,z)\cos(n\varphi) \end{bmatrix} + \begin{bmatrix} a_r^{-n}(r,z)\sin(n\varphi) \\ a_\varphi^{-n}(r,z)\cos(n\varphi) \\ a_z^{-n}(r,z)\sin(n\varphi) \end{bmatrix} \right)$$

→ Make the following **ansatz** (not unique): $a_\varphi^{\star,n} = r\, a_\varphi^n$

# 2.5D FEM in a nutshell

## Main idea

→ Decompose the full 3D field into a **Fourier series** along the **azimuthal direction** $\varphi$

$$\boldsymbol{a}(r,\varphi,z) = \begin{bmatrix} a_r^0(r,z) \\ a_\varphi^0(r,z) \\ a_z^0(r,z) \end{bmatrix} + \sum_{n=1}^{\infty} \left( \begin{bmatrix} a_r^n(r,z)\cos(n\varphi) \\ a_\varphi^n(r,z)\sin(n\varphi) \\ a_z^n(r,z)\cos(n\varphi) \end{bmatrix} + \begin{bmatrix} a_r^{-n}(r,z)\sin(n\varphi) \\ a_\varphi^{-n}(r,z)\cos(n\varphi) \\ a_z^{-n}(r,z)\sin(n\varphi) \end{bmatrix} \right)$$

→ Make the following **ansatz** (not unique): $a_\varphi^{\star,n} = r\, a_\varphi^n$

→ By substitution into the magnetostatic weak formulation (vector potential), we end up with

For all $\boldsymbol{a}'_{rz} \in H_0(\mathbf{curl}, \Omega)$ and $a_\varphi^{\star,\prime} \in H_0^1(\Omega)$, find $\boldsymbol{a}_{rz}^n \in H_0(\mathbf{curl}, \Omega)$ and $a_\varphi^{\star,n} \in H_0^1(\Omega)$, such that:

$$\int_\Omega r\nu_0\, \mathbf{curl}\, \boldsymbol{a}_{rz}^n \cdot \mathbf{curl}\, \boldsymbol{a}'_{rz}\, \mathrm{d}r\mathrm{d}z + n^2 \int_\Omega r^{-1}\nu_0 \boldsymbol{a}_{rz}^n \cdot \boldsymbol{a}'_{rz}\, \mathrm{d}r\mathrm{d}z + n \int_\Omega r^{-1}\nu_0\, \mathbf{grad}\, a_\varphi^{\star,n} \cdot \boldsymbol{a}'_{rz}\, \mathrm{d}r\mathrm{d}z$$
$$+ n \int_\Omega r^{-1}\nu_0 \boldsymbol{a}_{rz} \cdot \mathbf{grad}\, a_\varphi^{\star,\prime}\, \mathrm{d}r\mathrm{d}z + n \int_\Omega r^{-1}\nu_0\, \mathbf{grad}\, a_\varphi^{\star,n} \cdot \mathbf{grad}\, a_\varphi^{\star,\prime}\, \mathrm{d}r\mathrm{d}z = \int_\Omega r\, \boldsymbol{j}_{rz}^n \cdot \boldsymbol{a}'_{rz}\, \mathrm{d}r\mathrm{d}z$$

where $\Omega$ is an **angular slice** of the 3D domain and where **curl** and **grad** are **2D** operators.

# 2.5D FEM: GetDP I

**This is easily translated into GetDP scripting language**

```
Jacobian{
  { Name Vol; Case{ { Region All; Jacobian Vol; } } } }
```

→ Standard Jacobian, but more sophisticated weak formulation

# 2.5D FEM: GetDP II

```
Formulation{
  { Name MagStaA; Type FemEquation;
    Quantity{
      { Name aRZ;   Type Local; NameOfSpace HCurl;   }
      { Name aPhiS; Type Local; NameOfSpace HGrad;   }
      { Name xi;    Type Local; NameOfSpace HGradXi; } // Gauge !
    }
    Equation{      // R[] = X[]
      Galerkin{ [     R[]*Nu0*Dof{d aRZ}, {d aRZ}];     In Omega; [...] } // Integrals tested by aRZ
      Galerkin{ [$N^2/R[]*Nu0*Dof{  aRZ},   {aRZ}];     In Omega; [...] } //  |
      Galerkin{ [$N^1/R[]*Nu0*Dof{d aPhiS}, {aRZ}];     In Omega; [...] } //  |
      Galerkin{ [$N^1/R[]*Nu0*Dof{aRZ},     {d aPhiS}]; In Omega; [...] } // Integrals tested by aPhiS
      Galerkin{ [   1/R[]*Nu0*Dof{d aPhiS}, {d aPhiS}]; In Omega; [...] } //  |
      Galerkin{ [-R[] * Ji[$N], {aRZ}];                 In Src;   [...] } // RHS

      // Gauge condition! (Coulomb)
      Galerkin{ [   +R[]*Dof{d xi}, {aRZ}];             In Omega; [...] } // Integral  tested by aRZ
      Galerkin{ [-$N/R[]*Dof{xi}, {aPhiS}];             In Omega; [...] } // Integral  tested by aPhiS
      Galerkin{ [   +R[]*Dof{aRZ}, {d xi}];             In Omega; [...] } // Integrals tested by xi
      Galerkin{ [-$N/R[]*Dof{aPhiS}, {xi}];             In Omega; [...] } //  |
    } } }
```

$$\int_\Omega r\nu_0\,\mathbf{curl}\,\mathbf{a}^n_{rz}\cdot\mathbf{curl}\,\mathbf{a}'_{rz}\,\mathrm{d}r\mathrm{d}z+\int_\Omega n^2\,r^{-1}\nu_0\mathbf{a}^n_{rz}\cdot\mathbf{a}'_{rz}\,\mathrm{d}r\mathrm{d}z+\int_\Omega n\,r^{-1}\nu_0\,\mathbf{grad}\,a^{\star,n}_\varphi\cdot\mathbf{a}'_{rz}\,\mathrm{d}r\mathrm{d}z$$
$$+\int_\Omega n\,r^{-1}\nu_0\mathbf{a}_{rz}\cdot\mathbf{grad}\,a^{\star,\prime}_\varphi\,\mathrm{d}r\mathrm{d}z+\int_\Omega n\,r^{-1}\nu_0\,\mathbf{grad}\,a^{\star,n}_\varphi\cdot\mathbf{grad}\,a^{\star,\prime}_\varphi\,\mathrm{d}r\mathrm{d}z-\int_\Omega r\mathbf{j}^n_{rz}\cdot\mathbf{a}'_{rz}\,\mathrm{d}r\mathrm{d}z=0$$

# 2.5D FEM: GetDP III

```
Resolution{
  { Name MagStaA;
    System{
      { Name A; NameOfFormulation MagStaA; }
    }
    Operation{
      Evaluate[$MTot = 0];              // Init
      For i In {0:NTot-1}               // Loop on modes
        Evaluate[$N = i];               //  --> Mode i
        Generate[A];                    //  --> Generate
        Solve[A];                       //  --> Solve
        PostOperation[ComputeM];        //  --> Compute M
        Evaluate[$MTot = $MTot + $M]; //  --> Add contribution to MTot
      EndFor
    } // NB: mode-per-mode analysis is also possible
  }
}
```

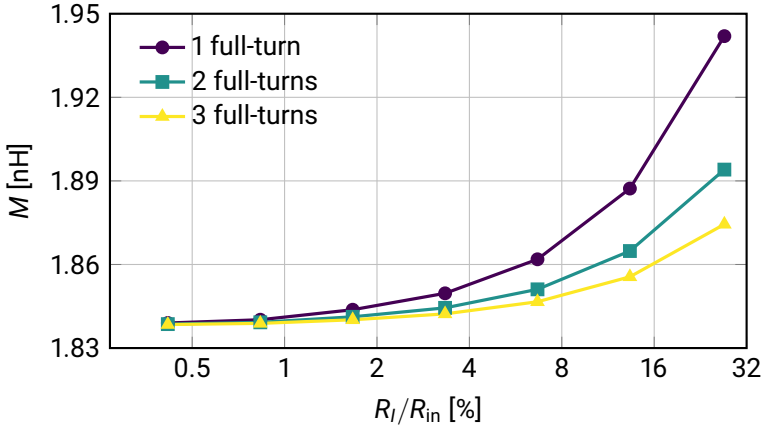
# 2.5D FEM: GetDP IV

```
PostProcessing{
  { Name MagStaA; NameOfFormulation MagStaA;
    PostQuantity{
      [...]
      { Name Phi; Value{ Integral{ [{aRZ}*Tangent[]];       [...] } } }
      { Name M;   Value{     Term{ [$Phi/I0];  Type Global; [...] } } }
    } } }
```

```
PostOperation ComputeM UsingPost MagStaA{
  Print[Phi[Loop], OnGlobal,       Format Table, StoreInVariable $Phi];
  Print[M,         OnRegion Dummy, Format Table, StoreInVariable $M,
        SendToServer "Output/02Modal M"];
}
```

# 2.5D FEM: Solution

**Test case: $M$ with 4 modes**



# CCC: mechanics I

**Large CCCs are not axisymmetric because of mechanical deformations and vibrations**

Do **deformations** and **vibrations** have a non-negligible impact on the performance of the CCC?

→ 3D elastodynamics eigenmode analysis + magnetostatics (assuming perfect screening)

# CCC: mechanics I

**Large CCCs are not axisymmetric because of mechanical deformations and vibrations**

Do **deformations** and **vibrations** have a non-negligible impact on the performance of the CCC?

→ 3D elastodynamics eigenmode analysis + magnetostatics (assuming perfect screening)

**Linear elastodynamics**

- 3D displacement field
  → One `FunctionSpace` per component
- `Resolution`
  → `EigenSolve[A, NEig, Re0,Im0]`

# CCC: mechanics I

**Large CCCs are not axisymmetric because of mechanical deformations and vibrations**

Do **deformations** and **vibrations** have a non-negligible impact on the performance of the CCC?

⟶ 3D elastodynamics eigenmode analysis + magnetostatics (assuming perfect screening)

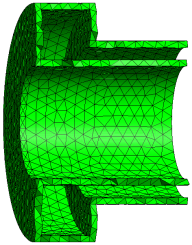
**Linear elastodynamics**

- 3D displacement field
  ⟶ One `FunctionSpace` per component
- `Resolution`
  ⟶ `EigenSolve[A, NEig, Re0,Im0]`

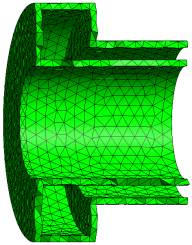
**Magnetostatics**

- 3D vector potential
  ⟶ $M$ is straightforward
- Gauge condition (spanning tree):
  ⟶ `EdgesOfTreeIn` (GetDP)
  ⟶ `Plugin(SpanningTree)` (Gmsh)

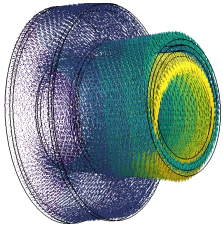
# CCC: mechanics II



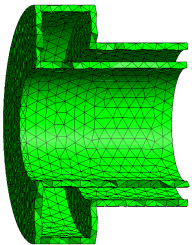
# CCC: mechanics II



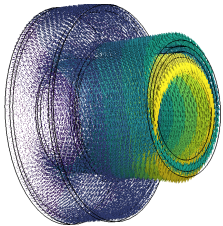
Eigenmode analysis



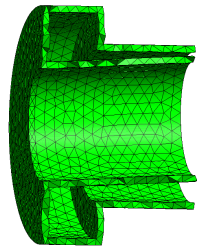
# CCC: mechanics II



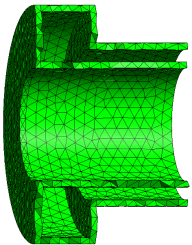
Eigenmode analysis



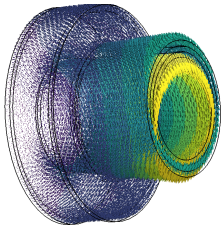
Mesh deformation



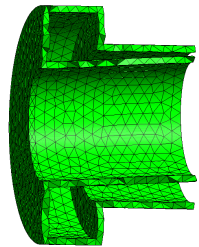
# CCC: mechanics II



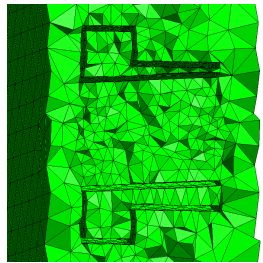
Eigenmode analysis



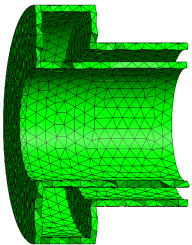
Mesh deformation



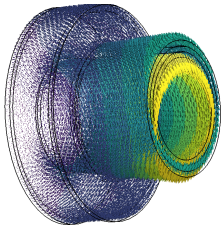
Re-meshing



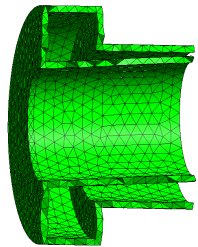
# CCC: mechanics II



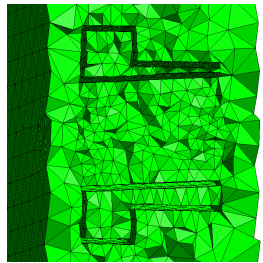
Eigenmode analysis



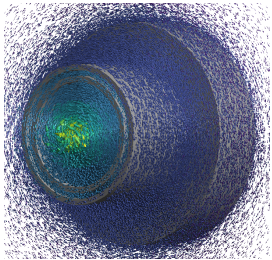
Mesh deformation



Re-meshing



MStatic



# Conclusion

**Why GetDP?**

- Straightforward implementation of a given weak formulation
- Simple integration within broader frameworks via ONELAB
- Scripting language in a single binary (or two if we count Gmsh)

**And last but not least...**

Special thanks to Erin Kuci and Erik Schnaubelt!