

Topology Optimization for Electromagnetic Problems using PyCFS

Walter Friza, Alice Reinbacher-Köstinger, Eniz Museljic, IGTE

Graz, April 10, 2026

Outline

1. Introduction/Recap
2. Magnetic Actuator Example
3. Setup in PyCFS

Topology Optimization

Topology Optimization

- Find a material distribution to achieve the optimization goal

Topology Optimization

- Find a material distribution to achieve the optimization goal
- Assign a design variable $0 \leq \rho \leq 1$ to every element of the mesh

Topology Optimization

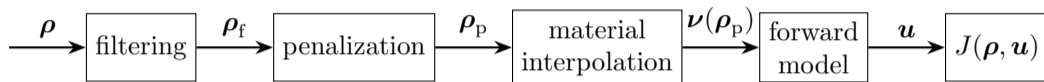
- Find a material distribution to achieve the optimization goal
- Assign a design variable $0 \leq \rho \leq 1$ to every element of the mesh
- ρ switches between materials

Topology Optimization

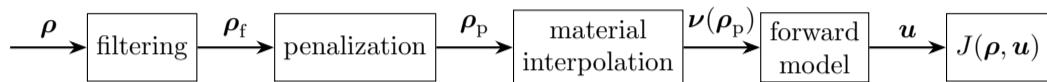
- Find a material distribution to achieve the optimization goal
- Assign a design variable $0 \leq \rho \leq 1$ to every element of the mesh
- ρ switches between materials

$$\begin{aligned} \min_{\rho} \quad & J(\rho, \mathbf{u}) \\ \text{s.t.} \quad & \mathcal{F}(\rho, \mathbf{u}) = 0 \\ & g_i(\rho, \mathbf{u}) \leq 0, i = 1, \dots, N \\ & \mathbf{0} < \rho_{\min} \leq \rho \leq \mathbf{1} \end{aligned}$$

Topology Optimization

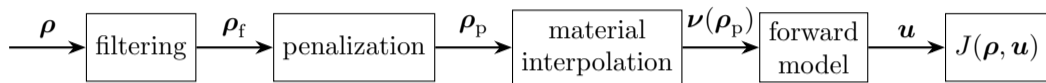


Topology Optimization



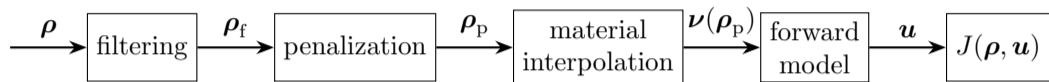
- Filtering to obtain physical meaningful solutions

Topology Optimization



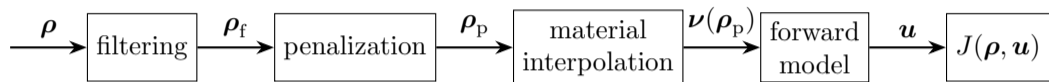
- Filtering to obtain physical meaningful solutions
- Penalization to obtain discrete solutions

Topology Optimization



- Filtering to obtain physical meaningful solutions
- Penalization to obtain discrete solutions
- Material interpolation converts the densities into physical material parameters

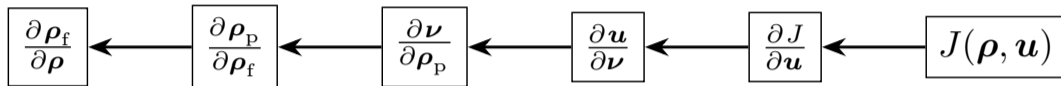
Topology Optimization



- Filtering to obtain physical meaningful solutions
- Penalization to obtain discrete solutions
- Material interpolation converts the densities into physical material parameters
- Forward model provides quantity \mathbf{u} for $J(\rho, \mathbf{u})$

Gradient Computation

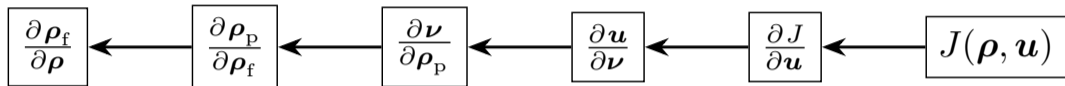
Gradient Computation



- Chain rule of derivatives

$$\frac{\partial J}{\partial \rho} = \frac{\partial J}{\partial u} \frac{\partial u}{\partial \nu} \frac{\partial \nu}{\partial \rho_p} \frac{\partial \rho_p}{\partial \rho_f} \frac{\partial \rho_f}{\partial \rho}$$

Gradient Computation

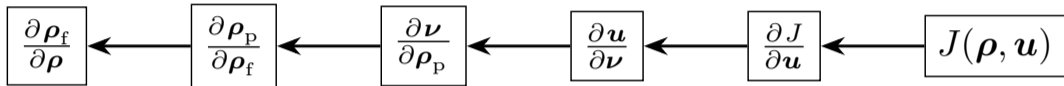


- Chain rule of derivatives

$$\frac{\partial J}{\partial \rho} = \frac{\partial J}{\partial u} \frac{\partial u}{\partial \nu} \frac{\partial \nu}{\partial \rho_p} \frac{\partial \rho_p}{\partial \rho_f} \frac{\partial \rho_f}{\partial \rho}$$

- Analytic expressions for $\frac{\partial \nu}{\partial \rho_p}$, $\frac{\partial \rho_p}{\partial \rho_f}$ and $\frac{\partial \rho_f}{\partial \rho}$ implemented in PyCFS

Gradient Computation

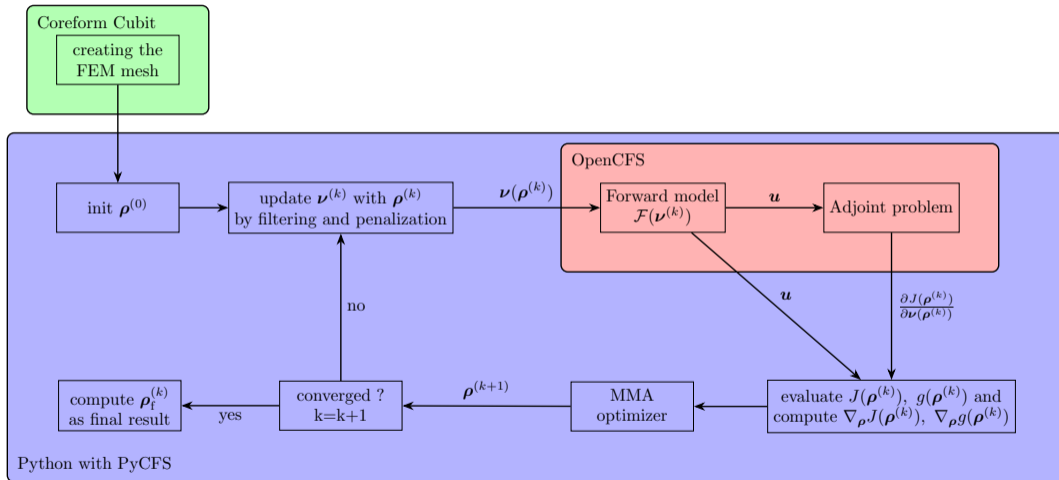


- Chain rule of derivatives

$$\frac{\partial J}{\partial \rho} = \frac{\partial J}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \nu} \frac{\partial \nu}{\partial \rho_p} \frac{\partial \rho_p}{\partial \rho_f} \frac{\partial \rho_f}{\partial \rho}$$

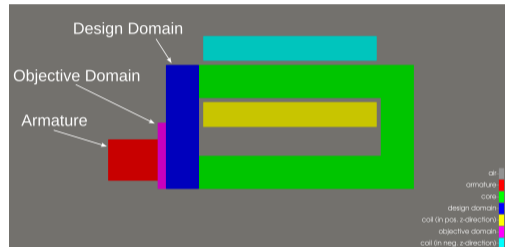
- Analytic expressions for $\frac{\partial \nu}{\partial \rho_p}$, $\frac{\partial \rho_p}{\partial \rho_f}$ and $\frac{\partial \rho_f}{\partial \rho}$ implemented in PyCFS
- $\frac{\partial J}{\partial \nu}$ Computed with the adjoint method in openCFS

Optimization with OpenCFS



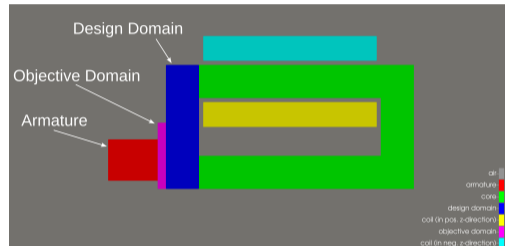
Problem Statement

Problem Statement



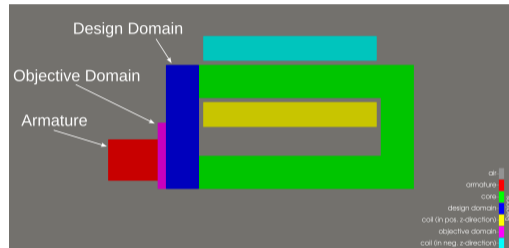
Problem Statement

- Maximize the magnetic field in the objective domain



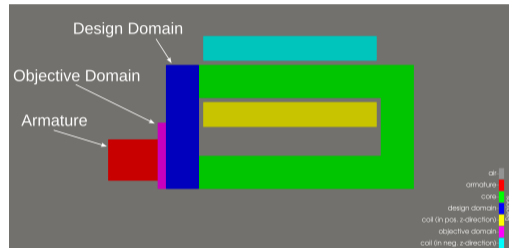
Problem Statement

- Maximize the magnetic field in the objective domain
- Material Parameter μ_r
 - armature: 2000
 - core and design domain: 1000
 - air and coil: 1



Problem Statement

- Maximize the magnetic field in the objective domain
- Material Parameter μ_r
 - armature: 2000
 - core and design domain: 1000
 - air and coil: 1
- Load parameter: $|\mathbf{J}| = 2 \text{ A/mm}^2$



Optimization Problem

$$\min_{\rho} \quad J(\rho, \mathbf{A}) = -\frac{1}{2} \sum_{e \in \Omega_o} \mathbf{B}_e^T \mathbf{B}_e$$

$$\text{s.t.} \quad \mathcal{F}(\mathbf{A}, \rho) = \mathbf{0}$$

$$g(\rho) = \frac{V(\rho)}{V_0} - V_{\text{frac}} \leq 0$$

$$\mathbf{0} < \rho_{\min} \leq \rho \leq \mathbf{1} .$$

Optimization Problem

- maximal volume fraction $V_{\text{frac}} = 0.6$

$$\min_{\rho} \quad J(\rho, \mathbf{A}) = -\frac{1}{2} \sum_{e \in \Omega_o} \mathbf{B}_e^T \mathbf{B}_e$$

$$\text{s.t.} \quad \mathcal{F}(\mathbf{A}, \rho) = \mathbf{0}$$

$$g(\rho) = \frac{V(\rho)}{V_0} - V_{\text{frac}} \leq 0$$

$$\mathbf{0} < \rho_{\min} \leq \rho \leq \mathbf{1} .$$

Optimization Problem

- maximal volume fraction $V_{\text{frac}} = 0.6$
- 6000 design variables

$$\min_{\rho} \quad J(\rho, \mathbf{A}) = -\frac{1}{2} \sum_{e \in \Omega_o} \mathbf{B}_e^T \mathbf{B}_e$$

$$\text{s.t.} \quad \mathcal{F}(\mathbf{A}, \rho) = \mathbf{0}$$

$$g(\rho) = \frac{V(\rho)}{V_0} - V_{\text{frac}} \leq 0$$

$$\mathbf{0} < \rho_{\min} \leq \rho \leq \mathbf{1} .$$

Optimization Problem

- maximal volume fraction $V_{\text{frac}} = 0.6$
- 6000 design variables
- filter size $R = 1$ mm

$$\min_{\rho} \quad J(\rho, \mathbf{A}) = -\frac{1}{2} \sum_{e \in \Omega_o} \mathbf{B}_e^T \mathbf{B}_e$$

$$\text{s.t.} \quad \mathcal{F}(\mathbf{A}, \rho) = \mathbf{0}$$

$$g(\rho) = \frac{V(\rho)}{V_0} - V_{\text{frac}} \leq 0$$

$$\mathbf{0} < \rho_{\min} \leq \rho \leq \mathbf{1} .$$

Optimization Problem

- maximal volume fraction $V_{\text{frac}} = 0.6$
- 6000 design variables
- filter size $R = 1$ mm
- SIMP with $p = 3$

$$\min_{\rho} \quad J(\rho, \mathbf{A}) = -\frac{1}{2} \sum_{e \in \Omega_o} \mathbf{B}_e^T \mathbf{B}_e$$

$$\text{s.t.} \quad \mathcal{F}(\mathbf{A}, \rho) = \mathbf{0}$$

$$g(\rho) = \frac{V(\rho)}{V_0} - V_{\text{frac}} \leq 0$$

$$\mathbf{0} < \rho_{\min} \leq \rho \leq \mathbf{1} .$$

Results



XML Setup

```
<magneticEdgeAdjT0>
  <regionList>
    <!-- specify your regions here -->
    <region name="V_core"/>
    <region name="V_air"/>
    <region name="V_design_domain" matDependIds="dep1"/>
    <region name="V_outer_coil_part"/>
    <region name="V_inner_coil_part"/>
    <region name="V_armature"/>
    <region name="V_objective_domain"/>
  </regionList>
  <matDependencyList>
    <matPermeability id="dep1">
      <grid>
        <defaultGrid quantity="magElemPermeability" definedOn="element"
          dependtype="CONST" snapToCFSTimeStep="true" sequenceStep="1" />
      </grid>
    </matPermeability>
  </matDependencyList>
</magneticEdgeAdjT0>
```

XML Setup

```
<bcAndLoads>
  <fluxParallel name="S_outer_air">
  </fluxParallel>
  <fluxParallel name="S_front">
  </fluxParallel>
  <fluxParallel name="S_back">
  </fluxParallel>
  <fluxDensity name="V_objective_domain">
    <coupling pdeName="magneticEdge">
      <quantity name="magFluxDensity"/>
    </coupling>
  </fluxDensity>
</bcAndLoads>
<storeResults>
  <elemResult type="magFluxDensityAdj">
    <allRegions/>
  </elemResult>
  <elemResult type="gradParam1">
    <regionList>
      <region name="V_design_domain" />
    </regionList>
  </elemResult>
</storeResults>
</magneticEdgeAdjT0>
```

XML Setup

```
<couplingList>
  <iterative PDEorder="magneticEdge; magneticEdgeAdjT0">
    <convergence logging="yes" maxNumIters="1" stopOnDivergence="no"/>
  </iterative>
</couplingList>
```

PyCFS Setup

```
iron_params = {
    "magElemReluctivity": 1/(MU_IRON * MU0),
    "magElemPermeability": MU_IRON * MU0,
    "magElemConductivity": 1.0e7,
}

armature_params = {
    "magElemReluctivity": 1/(MU_ARMATURE * MU0),
    "magElemPermeability": MU_ARMATURE * MU0,
    "magElemConductivity": 1.0e7,
}

air_params = {
    "magElemReluctivity": 1/(1e0 * MU0),
    "magElemPermeability": 1e0 * MU0,
    "magElemConductivity": 0.0,
}

copper_params = {
    "magElemReluctivity": 1/(1e0 * MU0),
    "magElemPermeability": 1e0 * MU0,
    "magElemConductivity": 5.96e7,
}

region_params = {
    "V_air": air_params,
    "V_objective_domain": air_params,
    "V_armature": armature_params,
    "V_inner_coil_part": copper_params,
    "V_outer_coil_part": copper_params,
    "V_design_domain": iron_params,
    "V_core": iron_params,
}
```

PyCFS Setup

```

iron_params = {
    "magElemReluctivity": 1/(MU_IRON * MU0),
    "magElemPermeability": MU_IRON * MU0,
    "magElemConductivity": 1.0e7,
}

armature_params = {
    "magElemReluctivity": 1/(MU_ARMATURE * MU0),
    "magElemPermeability": MU_ARMATURE * MU0,
    "magElemConductivity": 1.0e7,
}

air_params = {
    "magElemReluctivity": 1/(1e0 * MU0),
    "magElemPermeability": 1e0 * MU0,
    "magElemConductivity": 0.0,
}

copper_params = {
    "magElemReluctivity": 1/(1e0 * MU0),
    "magElemPermeability": 1e0 * MU0,
    "magElemConductivity": 5.96e7,
}

region_params = {
    "V_air": air_params,
    "V_objective_domain": air_params,
    "V_armature": armature_params,
    "V_inner_coil_part": copper_params,
    "V_outer_coil_part": copper_params,
    "V_design_domain": iron_params,
    "V_core": iron_params,
}

```

```

param_grad_map = {
    "magElemPermeability": 0,
}

# get simulation setup :
sim = get_setup()

# construct the topology optimizer :
penalization_method="simp"
r_filter = 1e-3
sim_topt = topt.Topt(
    sim,
    "3d",
    "V_design_domain",
    region_params,
    param_grad_map,
    penalization_method=penalization_method,
    r_filter=r_filter,
)

```

PyCFS Setup

```
def forward_problem(sim_topt, p, penalization=False):
    x = np.array([[]])
    sim_topt.update_design_parameters(p)
    sim_topt.sim(x)
    toptopt_design_volumes = sim_topt.sim.toptopt_design_volumes

    volfrac = sim_topt.compute_volume_constraint(p, penalization=penalization)
    B = sim_topt.sim.results[0][1]["magFluxDensity"]["V_objective_domain"].squeeze().flatten()
    obj_value = -0.5*np.abs(np.dot(B, B))

    dvolfrac_dp = np.array([sim_topt.compute_volume_constraint_deriv(penalization=penalization)]/sim_topt.V0)
    dg_dp = sim_topt.compute_gradient(normalize=False)*toptopt_design_volumes
    return (obj_value, -dg_dp, volfrac, dvolfrac_dp)
```

Thank you!