Automating convex optimization problems in FEniCSx

Jérémy Bleyer

Laboratoire Navier, Ecole des Ponts ParisTech

$p\partial$esoft 2024
1-3 July 2024, Cambridge, UK

Convex variational problems

Differentiable case

$$\inf_{u\in V} J(u)$$

variational equality:
$D_u J(u,v) = 0 \quad \forall v \in V$

Convex variational problems

Differentiable case

$$\inf_{u\in V} J(u)$$

variational equality:
$D_uJ(u,v) = 0 \quad \forall v \in V$

Cone-constrained case

$$\begin{aligned} \inf_{u\in V} \quad & J(u) \\ \text{s.t.} \quad & u \in \mathcal{K} \end{aligned}$$

variational inequality:
$D_uJ(u,v) \succeq_{\mathcal{K}^*} 0 \quad \forall v \in V$

Non-smooth case

$$\inf_{u\in V} \quad J(u)$$

variational inequality:
$D_uJ(u,v) \ni 0 \quad \forall v \in V$

VI arise in presence of **inequality constraints** or **non-smooth** objective functions

Convex variational problems

Differentiable case

$$\inf_{u\in V} J(u)$$

variational equality:
$D_uJ(u,v) = 0 \quad \forall v \in V$

Cone-constrained case

$$\begin{aligned} \inf_{u\in V} \quad & J(u) \\ \text{s.t.} \quad & u \in \mathcal{K} \end{aligned}$$

variational inequality:
$D_uJ(u,v) \succeq_{\mathcal{K}^*} 0 \quad \forall v \in V$

Non-smooth case

$$\inf_{u\in V} \quad J(u)$$

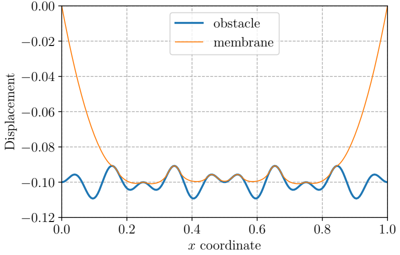
variational inequality:
$D_uJ(u,v) \ni 0 \quad \forall v \in V$

VI arise in presence of **inequality constraints** or **non-smooth** objective functions

Obstacle problem

$$\begin{aligned} \inf_{u\in V} \quad & \int_\Omega \frac{1}{2}\|\nabla u\|_2^2 \,d\Omega - \int_\Omega fu \,d\Omega \\ \text{s.t.} \quad & u \geq g \text{ on } \Omega \end{aligned}$$

`PETSc TAO` bound-constrained solvers



Convex variational problems

Differentiable case

$$\inf_{u \in V} J(u)$$

variational equality:
$D_u J(u, v) = 0 \quad \forall v \in V$

Cone-constrained case

$$\begin{array}{ll} \inf\limits_{u \in V} & J(u) \\ \text{s.t.} & u \in \mathcal{K} \end{array}$$

variational inequality:
$D_u J(u, v) \succeq_{\mathcal{K}^*} 0 \quad \forall v \in V$

Non-smooth case

$$\inf_{u \in V} \quad J(u)$$

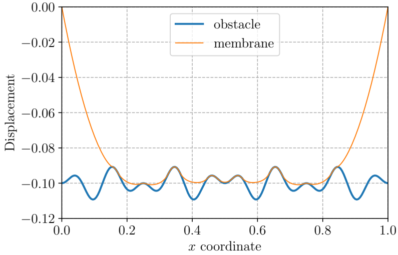
variational inequality:
$D_u J(u, v) \ni 0 \quad \forall v \in V$

VI arise in presence of **inequality constraints** or **non-smooth** objective functions

Obstacle problem

$$\begin{array}{ll} \inf\limits_{u \in V} & \displaystyle\int_\Omega \frac{1}{2}\|\nabla u\|_2^2 \,\mathrm{d}\Omega - \int_\Omega f u \,\mathrm{d}\Omega \\ \text{s.t.} & u \geq g \text{ on } \Omega \end{array}$$

`PETSc TAO` bound-constrained solvers



Various applications: finance, power systems, supply chain, robotics, image processing

In **mechanics**: unilateral conditions, friction, plasticity, damage, shape optimization, etc.

Non-smooth optimization as conic programming

Linear programming

$$\begin{array}{rl} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{array}$$

Non-smooth optimization as conic programming

Conic programming

$$\begin{array}{rl} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \mathcal{K} \end{array}$$

Non-smooth optimization as conic programming

Conic programming

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \mathcal{K} \end{array}$$

where $\mathcal{K}$ is a product of elementary cones e.g.:

- positive orthants $\mathbb{R}^m_+$;
- Lorentz quadratic cones: $\mathcal{Q}_m = \{\mathsf{z} = (z_0, \bar{\mathsf{z}}) \in \mathbb{R}^+ \times \mathbb{R}^{m-1} \text{ s.t. } \|\bar{\mathsf{z}}\|_2 \leq z_0\}$
- semi-definite cones $\mathcal{S}^+_m$, the cone of semi-definite positive $m \times m$ symmetric matrices;
- power cones, exponential cones, etc.

Non-smooth optimization as conic programming

Conic programming

$$\begin{array}{rl} \min_{\boldsymbol{x}} & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \mathcal{K} \end{array}$$

where $\mathcal{K}$ is a product of elementary cones e.g.:

- positive orthants $\mathbb{R}^m_+$;
- Lorentz quadratic cones: $\mathcal{Q}_m = \{z = (z_0, \bar{z}) \in \mathbb{R}^+ \times \mathbb{R}^{m-1} \text{ s.t. } \|\bar{z}\|_2 \leq z_0\}$
- semi-definite cones $\mathcal{S}^+_m$, the cone of semi-definite positive $m \times m$ symmetric matrices;
- power cones, exponential cones, etc.

Solvers

interior-point algorithms, very efficient and robust (20-30 iterations)

The magic cone family [Juditsky & Nemirovski, 2021]

very large modelling power of **convex** functions and constraints

Conic-representable functions

Conic-representable function/constraint:

$$\begin{aligned} F(\boldsymbol{x}) = \min_{\boldsymbol{y}} \quad & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}^\mathsf{T}\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{b}_l \leq \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \leq \boldsymbol{b}_u \\ & \boldsymbol{y} \in \mathcal{K}^1 \times \ldots \times \mathcal{K}^p \end{aligned}$$

Conic-representable functions

Conic-representable function/constraint:

$$\begin{aligned} F(\boldsymbol{x}) = \min_{\boldsymbol{y}} \quad & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}^\mathsf{T}\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{b}_l \leq \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \leq \boldsymbol{b}_u \\ & \boldsymbol{y} \in \mathcal{K}^1 \times \ldots \times \mathcal{K}^p \end{aligned}$$

Conic-representable variational problem:

$$J(u) = \sum_{i=1}^{n} \int_\Omega F_i(\ell_i(u)) \,\mathrm{d}\Omega$$

where F_i are *conic-representable* and ℓ_i are UFL-representable linear operators

Conic-representable functions

Conic-representable function/constraint:

$$\begin{aligned} F(\boldsymbol{x}) = \min_{\boldsymbol{y}} \quad & \boldsymbol{c}^\mathsf{T}\boldsymbol{x} + \boldsymbol{d}^\mathsf{T}\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{b}_l \leq \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} \leq \boldsymbol{b}_u \\ & \boldsymbol{y} \in \mathcal{K}^1 \times \ldots \times \mathcal{K}^p \end{aligned}$$

Conic-representable variational problem:

$$J(u) = \sum_{i=1}^{n} \int_\Omega F_i(\ell_i(u))\, \mathrm{d}\Omega$$

where F_i are *conic-representable* and ℓ_i are UFL-representable linear operators

Choice of a **quadrature rule**: $J(u) = \displaystyle\int_\Omega F(\ell(u))\, \mathrm{d}\Omega \approx \sum_{g=1}^{N_g} \omega_g F(\boldsymbol{L}_g \boldsymbol{u})$

$$\Rightarrow \qquad \begin{aligned} \min_{\boldsymbol{u}} J(\boldsymbol{u}) = \min_{\boldsymbol{u}, \boldsymbol{y}_g} \quad & \sum_{g=1}^{N_g} \omega_g (\boldsymbol{c}^\mathsf{T}\boldsymbol{L}_g\boldsymbol{u} + \boldsymbol{d}^\mathsf{T}\boldsymbol{y}_g) \\ \text{s.t.} \quad & \boldsymbol{b}_l \leq \boldsymbol{A}\boldsymbol{L}_g\boldsymbol{u} + \boldsymbol{B}\boldsymbol{y}_g \leq \boldsymbol{b}_u \\ & \boldsymbol{y}_g \in \mathcal{K}^1 \times \ldots \times \mathcal{K}^p \end{aligned}$$

The dolfinx_optim package

$$\min_{\boldsymbol{u}\in V} \int_\Omega F(\boldsymbol{u})\,\mathrm{d}\Omega \quad \overset{\text{FE discretization}}{\underset{\text{canonicalization}}{\Longrightarrow}} \quad \begin{array}{ll} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \mathcal{K} \end{array}$$

- **Domain-Specific Language** based on UFL for convex functions and their composition
- Mosek interior-point solver
- pre-defined **convex primitives**
  - AbsValue, LinearTerm, QuadraticTerm, QuadOverLin, etc.
  - vectors: L1Norm, L2Norm, LinfNorm, LpNorm, etc.
  - matrices: SpectralNorm, NuclearNorm, FroebeniusNorm, LambdaMax, etc.
- **composability** through convex-preserving transformations

The dolfinx_optim package

FEniCS PROJECT

$$\min_{u\in V} \int_\Omega F(u)\,\mathrm{d}\Omega$$

FE discretization $\Longrightarrow$ canonicalization

MOSEK

$$\begin{array}{ll}\min\limits_{x} & c^\top x \\ \text{s.t.} & Ax = b \\ & x \in \mathcal{K}\end{array}$$

- **Domain-Specific Language** based on UFL for convex functions and their composition
- `Mosek` interior-point solver
- pre-defined **convex primitives**
  - `AbsValue`, `LinearTerm`, `QuadraticTerm`, `QuadOverLin`, etc.
  - vectors: `L1Norm`, `L2Norm`, `LinfNorm`, `LpNorm`, etc.
  - matrices: `SpectralNorm`, `NuclearNorm`, `FroebeniusNorm`, `LambdaMax`, etc.
- **composability** through convex-preserving transformations

```
prob = MosekProblem(domain, name="Obstacle problem")
u = prob.add_var(V, bc=bc, lx=g)

prob.add_obj_func(-ufl.dot(f, u) * ufl.dx)

J = QuadraticTerm(ufl.grad(u), degree)
prob.add_convex_term(J)

prob.optimize()
```

Obstacle problem

$$\begin{array}{ll}\inf\limits_{u\in V} & \displaystyle\int_\Omega \frac{1}{2}\|\nabla u\|_2^2\,\mathrm{d}\Omega - \int_\Omega f u\,\mathrm{d}\Omega \\ \text{s.t.} & u \geq g \text{ on } \Omega\end{array}$$

Transformations

Convexity-preserving operations:

- sum $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$
- supremum $\sup\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$
- partial minimization
- Legendre-Fenchel transform

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}} \boldsymbol{s}^\mathsf{T}\boldsymbol{x} - f(\boldsymbol{x})$$

- inf-convolution

$$(f \Box g)(\boldsymbol{x}) = \begin{array}{rl} \inf\limits_{\boldsymbol{x}_1, \boldsymbol{x}_2} & f(\boldsymbol{x}_1) + g(\boldsymbol{x}_2) \\ \text{s.t.} & \boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2 \end{array}$$

- perspective
- ...

Transformations

Convexity-preserving operations:

- sum $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$
- supremum $\sup\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$
- partial minimization
- Legendre-Fenchel transform

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}} \boldsymbol{s}^{\mathsf{T}}\boldsymbol{x} - f(\boldsymbol{x})$$

- inf-convolution

$$\begin{aligned}(f\Box g)(\boldsymbol{x}) = \inf_{\boldsymbol{x}_1,\boldsymbol{x}_2} \quad & f(\boldsymbol{x}_1) + g(\boldsymbol{x}_2) \\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2\end{aligned}$$

- perspective
- ...

e.g. **Perspective** function
$\text{persp}_f(t, \boldsymbol{x}) = tf(\boldsymbol{x}/t)$

$$\begin{aligned}tf(\boldsymbol{x}/t) = \min_{\boldsymbol{y}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T t\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{b}_l \leq \boldsymbol{A}\boldsymbol{x}/t + \boldsymbol{B}\boldsymbol{y} \leq \boldsymbol{b}_u \\ & \boldsymbol{y} \in \mathcal{K}_1 \times \ldots \times \mathcal{K}_p\end{aligned}$$

Transformations

Convexity-preserving operations:

- sum $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$
- supremum $\sup\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$
- partial minimization
- Legendre-Fenchel transform

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}} \boldsymbol{s}^\mathsf{T}\boldsymbol{x} - f(\boldsymbol{x})$$

- inf-convolution

$$\begin{aligned}(f\square g)(\boldsymbol{x}) = \inf_{\boldsymbol{x}_1,\boldsymbol{x}_2} \quad & f(\boldsymbol{x}_1) + g(\boldsymbol{x}_2)\\ \text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2\end{aligned}$$

- perspective
- ...

e.g. **Perspective** function
$\text{persp}_f(t, \boldsymbol{x}) = tf(\boldsymbol{x}/t)$

$$\begin{aligned}tf(\boldsymbol{x}/t) = \min_{\widetilde{\boldsymbol{y}}} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\widetilde{\boldsymbol{y}}\\ \text{s.t.} \quad & t\boldsymbol{b}_l \leq \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\widetilde{\boldsymbol{y}} \leq t\boldsymbol{b}_u\\ & \widetilde{\boldsymbol{y}} \in \mathcal{K}_1 \times \ldots \times \mathcal{K}_p\end{aligned}$$

Transformations

Convexity-preserving operations:

- sum $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$
- supremum $\sup\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$
- partial minimization
- Legendre-Fenchel transform

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}} \boldsymbol{s}^\mathsf{T}\boldsymbol{x} - f(\boldsymbol{x})$$

- inf-convolution

$$\begin{array}{rll} (f \square g)(\boldsymbol{x}) = & \displaystyle\inf_{\boldsymbol{x}_1,\boldsymbol{x}_2} & f(\boldsymbol{x}_1) + g(\boldsymbol{x}_2) \\ & \text{s.t.} & \boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2 \end{array}$$

- perspective
- ...

e.g. **Perspective** function
$\text{persp}_f(t, \boldsymbol{x}) = tf(\boldsymbol{x}/t)$

$$\begin{array}{rll} tf(\boldsymbol{x}/t) = & \displaystyle\min_{\widetilde{\boldsymbol{y}}} & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\widetilde{\boldsymbol{y}} \\ & \text{s.t.} & 0 \leq -t\boldsymbol{b}_l + \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\widetilde{\boldsymbol{y}} \\ & & \quad - t\boldsymbol{b}_u + \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\widetilde{\boldsymbol{y}} \leq 0 \\ & & \widetilde{\boldsymbol{y}} \in \mathcal{K}_1 \times \ldots \times \mathcal{K}_p \end{array}$$

Transformations

Convexity-preserving operations:

- sum $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$
- supremum $\sup\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$
- partial minimization
- Legendre-Fenchel transform

$$f^*(\boldsymbol{s}) = \sup_{\boldsymbol{x}} \boldsymbol{s}^\mathsf{T}\boldsymbol{x} - f(\boldsymbol{x})$$

- inf-convolution

$$\begin{array}{rl} (f\Box g)(\boldsymbol{x}) = \displaystyle\inf_{\boldsymbol{x}_1,\boldsymbol{x}_2} & f(\boldsymbol{x}_1) + g(\boldsymbol{x}_2) \\ \text{s.t.} & \boldsymbol{x} = \boldsymbol{x}_1 + \boldsymbol{x}_2 \end{array}$$

- perspective
- ...

e.g. **Perspective** function
$\text{persp}_f(t,\boldsymbol{x}) = tf(\boldsymbol{x}/t)$

$$\begin{array}{rl} tf(\boldsymbol{x}/t) = \displaystyle\min_{\widetilde{\boldsymbol{y}}} & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d}^T\widetilde{\boldsymbol{y}} \\ \text{s.t.} & 0 \leq -t\boldsymbol{b}_l + \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\widetilde{\boldsymbol{y}} \\ & \quad -t\boldsymbol{b}_u + \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\widetilde{\boldsymbol{y}} \leq 0 \\ & \widetilde{\boldsymbol{y}} \in \mathcal{K}_1 \times \ldots \times \mathcal{K}_p \end{array}$$

Data transformation

$$\boldsymbol{x} \mapsto \begin{Bmatrix} t \\ \boldsymbol{x} \end{Bmatrix} \qquad \boldsymbol{c} \mapsto \begin{Bmatrix} 0 \\ \boldsymbol{c} \end{Bmatrix}$$

$$\boldsymbol{A} \mapsto \begin{bmatrix} -\boldsymbol{b}_l & \boldsymbol{A} \\ -\boldsymbol{b}_u & \boldsymbol{A} \end{bmatrix} \qquad \boldsymbol{B} \mapsto \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B} \end{bmatrix}$$

$$\boldsymbol{b}_l \mapsto \begin{Bmatrix} 0 \\ \text{None} \end{Bmatrix} \qquad \boldsymbol{b}_u \mapsto \begin{Bmatrix} \text{None} \\ 0 \end{Bmatrix}$$

Viscoplastic fluids around us

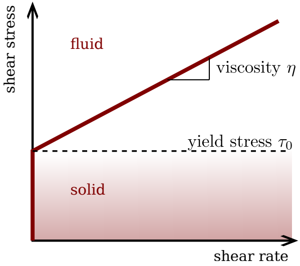
cosmetics

food

construction, geophysics

Formulation

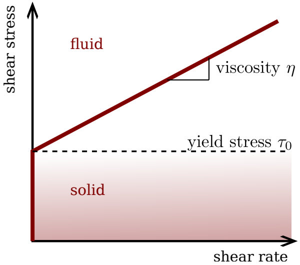
Viscoplastic fluids = a specific class of **non-Newtonian fluids** with a solid-like behaviour



- flow like a simple fluid above a **critical stress**
- remains at rest, like a solid, below

Formulation

Viscoplastic fluids = a specific class of **non-Newtonian fluids** with a solid-like behaviour



- flow like a simple fluid above a **critical stress**
- remains at rest, like a solid, below

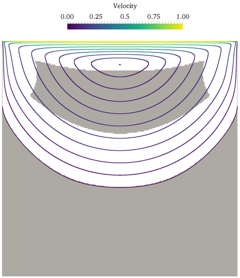
Primal variational principle: **smooth** + **non-smooth** term

$$\begin{aligned} \min_{\boldsymbol{u},\boldsymbol{d}} \quad & \int_\Omega \left(\frac{\eta}{2}\|\boldsymbol{d}\|^2 + \sqrt{2}\tau_0\|\boldsymbol{d}\|\right) \mathrm{d}\Omega - \int_\Omega \boldsymbol{f}\cdot\boldsymbol{u}\,\mathrm{d}\Omega \\ \text{s.t.} \quad & \boldsymbol{d} = \tfrac{1}{2}(\nabla\boldsymbol{u} + \nabla^\mathsf{T}\boldsymbol{u}) \\ & \operatorname{div}\boldsymbol{u} = 0 \end{aligned}$$

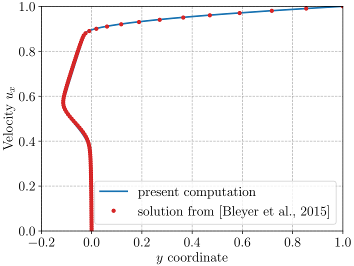
Viscoplastic fluid implementation

```
prob = MosekProblem(domain, "Viscoplastic fluid")

u = prob.add_var(V, bc=bc)

# mass conservation condition
Vp = fem.functionspace(domain, ("P", 1))
p = ufl.TestFunction(Vp)
prob.add_eq_constraint(p * ufl.div(u) * ufl.dx)

def strain(v):
    D = ufl.sym(ufl.grad(v))
    return ufl.as_vector([D[0, 0], D[1, 1], ufl.sqrt(2) * D[0, 1]])

visc = QuadraticTerm(strain(u), 2)
plast = L2Norm(strain(u), 2)

# add viscous term mu*||strain||_2^2
prob.add_convex_term(2 * mu * visc)
# add plastic term sqrt(2)*tau0*||strain||_2
prob.add_convex_term(np.sqrt(2) * tau0 * plast)

prob.optimize()
```

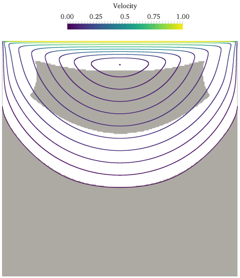
Viscoplastic fluid implementation



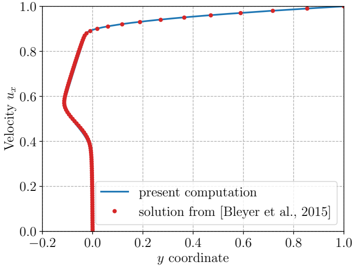
$\mathtt{Bi} = 20$

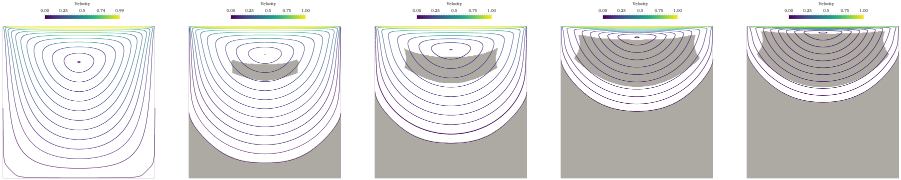


Viscoplastic fluid implementation



Bi = 20





Bi = 0 Bi = 2 Bi = 5 Bi = 50 Bi = 200

Variational cartoon/texture decomposition

Image $y = u$ (cartoon) $+ v$ (texture)

Y.Meyer's model (TV + G-norm) [Meyer, 2001]:

$$\begin{array}{ll} \inf\limits_{u,v} & \displaystyle\int_\Omega \|\nabla u\|_2 \,\mathrm{d}\Omega + \alpha\|v\|_G \\ \text{s.t.} & y = u + v \end{array}$$

$$\text{where} \quad \|v\|_G = \inf_{\boldsymbol{g}\in L^\infty(\Omega;\mathbb{R}^2)} \{\|\sqrt{g_1^2+g_2^2}\|_\infty \text{ s.t. } v = \operatorname{div}\boldsymbol{g}\}$$

reformulated as [Weiss et al., 2009]:

$$\begin{array}{ll} \inf\limits_{u,\boldsymbol{g}} & \displaystyle\int_\Omega \|\nabla u\|_2 \,\mathrm{d}\Omega \\ \text{s.t.} & y = u + \operatorname{div}(\boldsymbol{g}) \\ & \|\sqrt{g_1^2+g_2^2}\|_\infty \leq \alpha \end{array}$$

L_2 ad $L_{\infty,2}$-norms are **conic-representable** $\Rightarrow$ SOCP problem

Variational cartoon/texture decomposition

Image y : represented by a `DG0` field on a `512x512` finite-element mesh
$u, \boldsymbol{g} \in \mathtt{CR} \times \mathtt{RT}$

```
prob = MosekProblem(domain, "Cartoon/texture decomposition")
Vu = fem.functionspace(domain, ("CR", 1))
Vg = fem.functionspace(domain, ("RT", 1))

u, g = prob.add_var([Vu, Vg], name=["Cartoon", "Texture"])



lamb_ = ufl.TestFunction(Vu)
constraint = ufl.dot(lamb_, u + ufl.div(g)) * ufl.dx
rhs = ufl.dot(lamb_, y) * ufl.dx
prob.add_eq_constraint(constraint, b=rhs)

tv_norm = L2Norm(ufl.grad(u), 0)
prob.add_convex_term(tv_norm)

g_norm = L2Ball(g / alpha, 2)
prob.add_convex_term(g_norm)

prob.optimize()
```

Variational cartoon/texture decomposition

Image y : represented by a `DG0` field on a `512x512` finite-element mesh
$u, \boldsymbol{g} \in \mathtt{CR} \times \mathtt{RT}$

Barbara image

Limit analysis

Goal: find the maximum collapse load $F^+ = \lambda^+ F$ that a structure can sustain under a convex plasticity domain G

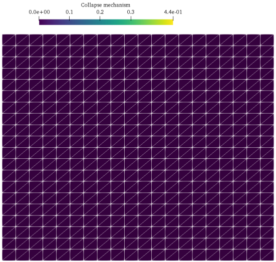
Plastic dissipation minimization principle:

$$\lambda^+ = \min_{\boldsymbol{u}\in\mathcal{U}_{\text{ad}}} \int_\Omega \pi_G(\boldsymbol{\varepsilon})\,\mathrm{d}\Omega \quad \text{s.t.} \quad \int_\Omega \boldsymbol{f}\cdot\boldsymbol{u}\,\mathrm{d}\Omega + \int_{\Omega_{\mathbf{N}}} \boldsymbol{T}\cdot\boldsymbol{u}\,\mathrm{d}S = 1$$

$$\pi_G(\boldsymbol{\varepsilon}) = \sup_{\boldsymbol{\sigma}\in G} \boldsymbol{\sigma}:\boldsymbol{\varepsilon}$$

e.g. Mohr-Coulomb 3D criterion: $\pi_G(\boldsymbol{\varepsilon}) = \begin{cases} c \operatorname{cotan}\phi \operatorname{tr}\boldsymbol{\varepsilon} & \text{if } \operatorname{tr}(\boldsymbol{\varepsilon}) \geq \sin\phi \sum_I |\varepsilon_I| \\ +\infty & \text{otherwise} \end{cases}$

```
class MohrCoulomb(ConvexTerm):
    """SDP implementation of Mohr-Coulomb criterion."""
    def conic_repr(self, X):
        Y1 = self.add_var((3,3), cone=SDP(3))
        Y2 = self.add_var((3,3), cone=SDP(3))
        a = (1 - ufl.sin(phi)) / (1 + ufl.sin(phi))
        self.add_eq_constraint(X - to_vect(Y1) + to_vect(Y2))
        self.add_eq_constraint(ufl.tr(Y2) - a * ufl.tr(Y1))
        self.add_linear_term(2 * c * ufl.cos(phi) / (1 + ufl.sin(phi))
       * ufl.tr(Y1))
```

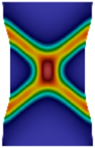


Conclusions

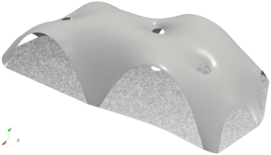
Project available at:

`https://github.com/bleyerj/dolfinx_optim`

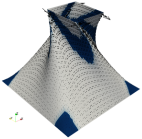
plasticity



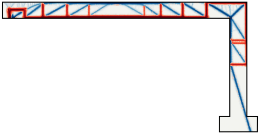
form-finding



membranes



topology optimization



Future works

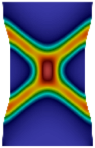
- facet-based convex terms (DG schemes)
- other solvers (custom ?)
- sensitivity analysis ?

Conclusions

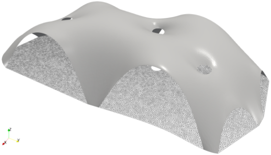
Project available at:

`https://github.com/bleyerj/dolfinx_optim`

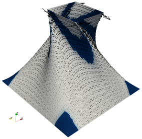
plasticity



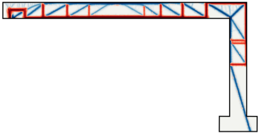
form-finding



membranes



topology optimization



Future works

- facet-based convex terms (DG schemes)
- other solvers (custom ?)
- sensitivity analysis ?

Thank you for your attention !