Learning elastoplasticity with implicit layers

Jérémy Bleyer

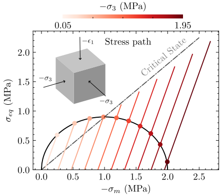
Ecole Nationale des Ponts et Chaussées
Laboratoire Navier, ENPC-IP Paris, Univ Gustave Eiffel, CNRS

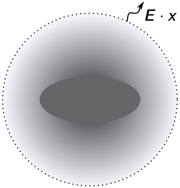
CMCS 2025
Wednesday, May 14th 2025

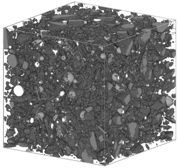
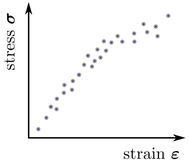
Constitutive modeling

Constitutive behavior: complements **balance equations** and **kinematic relations**

- phenomenological
- micromechanics/mean-field
- computational (FE^2, FFT, reduced models)
- data-driven/Machine-Learning techniques





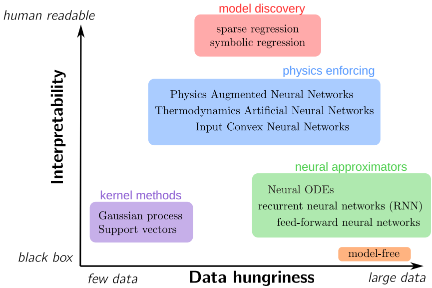
Thermodynamics: path/history-dependence, internal state variables, evolution equations

Outline

Outline

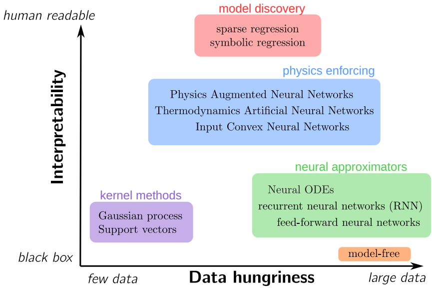
Existing approaches and challenges

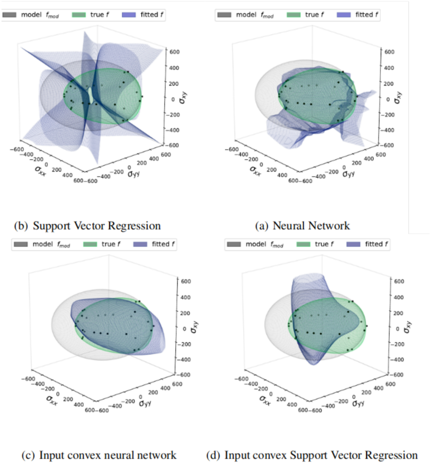
Dilemma: High **expressiveness** and **interpretability** with **good generalization** outside **limited data**



Existing approaches and challenges

Dilemma: High **expressiveness** and **interpretability** with **good generalization** outside **limited data**





Adapted from [Fuhg et al., 2022]

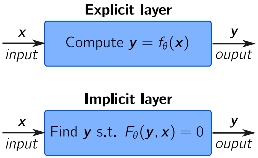
Challenges in plasticity:

- non-smooth behavior
- convex yield surfaces
- high-dimensional, path dependent

Implicit Layers: a new paradigm

What are **Implicit Layers** [Amos, Kolter, El Ghaoui, 2017-2021] ?

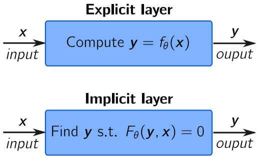
- solve equations or optimization problems instead of fixed operations
- output is defined **implicitly** as the solution to a system



Implicit Layers: a new paradigm

What are **Implicit Layers** [Amos, Kolter, El Ghaoui, 2017-2021] ?

- solve equations or optimization problems instead of fixed operations
- output is defined **implicitly** as the solution to a system



Advantages:

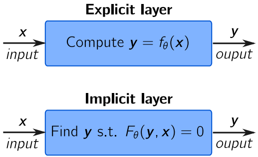
- Strong expressiveness/few parameters: *1 implicit layer is all you need* [Kolter et al., 2020]
- Backpropagation via **implicit differentiation**
- Can **explicitly** enforce **constraints** vs. loss or architecture crafting

Applications in ML: Neural ODEs, Deep Equilibrium (DEQ) models, Optimization Networks (OptNet)

Implicit Layers: a new paradigm

What are **Implicit Layers** [Amos, Kolter, El Ghaoui, 2017-2021] ?

- solve equations or optimization problems instead of fixed operations
- output is defined **implicitly** as the solution to a system



Advantages:

- Strong expressiveness/few parameters: *1 implicit layer is all you need* [Kolter et al., 2020]
- Backpropagation via **implicit differentiation**
- Can **explicitly** enforce **constraints** vs. loss or architecture crafting

Applications in ML: Neural ODEs, Deep Equilibrium (DEQ) models, Optimization Networks (OptNet)

Application to constitutive modeling

- Constitutive updates (e.g. *return-mapping*) are formulated as **non-linear root finding**
- Can enforce **variational structure**, **convexity**, etc.

Outline

Variational principle for dissipative media

Dissipative media as Generalized Standard Materials (GSM) [Halphen & Nguyen, 1983]

- state variables: $\boldsymbol{\varepsilon}, \boldsymbol{\alpha}$ with $\boldsymbol{\alpha} = p, \boldsymbol{\varepsilon}^{\mathsf{p}}, d, f$, etc.
- free energy: $\psi(\boldsymbol{\varepsilon}, \boldsymbol{\alpha})$
- pseudo-dissipation potential: $\phi(\dot{\boldsymbol{\varepsilon}}, \dot{\boldsymbol{\alpha}})$

ψ and ϕ are **convex**, **non-negative** and **zero at the origin**

Variational principle for dissipative media

Dissipative media as Generalized Standard Materials (GSM) [Halphen & Nguyen, 1983]

- state variables: $\boldsymbol{\varepsilon}, \boldsymbol{\alpha}$ with $\boldsymbol{\alpha} = p, \boldsymbol{\varepsilon}^{\mathsf{p}}, d, f$, etc.
- free energy: $\psi(\boldsymbol{\varepsilon}, \boldsymbol{\alpha})$
- pseudo-dissipation potential: $\phi(\dot{\boldsymbol{\varepsilon}}, \dot{\boldsymbol{\alpha}})$

ψ and ϕ are **convex**, **non-negative** and **zero at the origin**

In duality to state variables: **stress $\boldsymbol{\sigma}$** and **thermodynamic forces $\boldsymbol{Y}$**

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\mathsf{nd}} + \boldsymbol{\sigma}^{\mathsf{d}}$$

$$0 = \boldsymbol{Y}^{\mathsf{nd}} + \boldsymbol{Y}^{\mathsf{d}}$$

$$(\boldsymbol{\sigma}^{\mathsf{nd}}, \boldsymbol{Y}^{\mathsf{nd}}) \in \partial_{(\boldsymbol{\varepsilon},\boldsymbol{\alpha})}\psi(\boldsymbol{\varepsilon}, \boldsymbol{\alpha}) \qquad (\boldsymbol{\varepsilon}, \boldsymbol{\alpha}) \in \partial_{(\boldsymbol{\sigma}^{\mathsf{nd}}, \boldsymbol{Y}^{\mathsf{nd}})}\psi^*(\boldsymbol{\sigma}^{\mathsf{nd}}, \boldsymbol{Y}^{\mathsf{nd}})$$

$$(\boldsymbol{\sigma}^{\mathsf{d}}, \boldsymbol{Y}^{\mathsf{d}}) \in \partial_{(\dot{\boldsymbol{\varepsilon}},\dot{\boldsymbol{\alpha}})}\phi(\dot{\boldsymbol{\varepsilon}}, \dot{\boldsymbol{\alpha}}) \qquad (\dot{\boldsymbol{\varepsilon}}, \dot{\boldsymbol{\alpha}}) \in \partial_{(\boldsymbol{\sigma}^{\mathsf{d}}, \boldsymbol{Y}^{\mathsf{d}})}\phi^*(\boldsymbol{\sigma}^{\mathsf{d}}, \boldsymbol{Y}^{\mathsf{d}})$$

sub-differential for non-smooth functions
e.g. $\partial\phi^*$ is a convex set when ϕ is 1-homogeneous (rate-independent behaviour)

Incremental variational principles

Evolution between t_n and $t_n + \Delta t$ is obtained by solving [Ortiz & Stainier, 1999; Mielke, 2005]:

Primal incremental variational problem

$$(\boldsymbol{u}_{n+1}, \boldsymbol{\alpha}_{n+1}) = \underset{\boldsymbol{u}\in\mathcal{U}_{\textbf{ad}}, \boldsymbol{\alpha}}{\arg\min} \int_\Omega \left(\psi(\boldsymbol{\varepsilon}, \boldsymbol{\alpha}) + \Delta t \phi\left(\frac{\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_n}{\Delta t}, \frac{\boldsymbol{\alpha} - \boldsymbol{\alpha}_n}{\Delta t}\right)\right) \mathrm{d}\Omega - W_{\text{ext},n+1}(\boldsymbol{u})$$

Incremental variational principles

Evolution between t_n and $t_n + \Delta t$ is obtained by solving [Ortiz & Stainier, 1999; Mielke, 2005]:

Primal incremental variational problem

$$(\boldsymbol{u}_{n+1}, \boldsymbol{\alpha}_{n+1}) = \underset{\boldsymbol{u}\in\mathcal{U}_{\text{ad}},\boldsymbol{\alpha}}{\arg\min} \int_\Omega \left(\psi(\boldsymbol{\varepsilon},\boldsymbol{\alpha}) + \Delta t \phi\left(\dfrac{\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}_n}{\Delta t}, \dfrac{\boldsymbol{\alpha}-\boldsymbol{\alpha}_n}{\Delta t}\right)\right) \,\mathrm{d}\Omega - W_{\text{ext},n+1}(\boldsymbol{u})$$

e.g. **Plasticity**: $\boldsymbol{\varepsilon}$ non-dissipative, rate-independent, elasticity and hardening $\boldsymbol{\alpha} = (\boldsymbol{\varepsilon}^\text{p}, p)$:

$$\psi(\boldsymbol{\varepsilon},\boldsymbol{\alpha}) = \psi_\text{el}(\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}^\text{p}) + \psi_\text{h}(\boldsymbol{\varepsilon}^\text{p}, p)$$

yields:

$$(\boldsymbol{u}_{n+1}, \boldsymbol{\alpha}_{n+1}) = \underset{\boldsymbol{u}\in\mathcal{U}_{\text{ad}},\boldsymbol{\alpha}}{\arg\min} \int_\Omega \left(\psi_\text{el}(\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}^\text{p}) + \psi_\text{h}(\boldsymbol{\varepsilon}^\text{p}, p) + \phi\left(\boldsymbol{\varepsilon}^\text{p}-\boldsymbol{\varepsilon}^\text{p}_n, p-p_n\right)\right) \,\mathrm{d}\Omega - W_{\text{ext},n+1}(\boldsymbol{u})$$

with $\phi\left(\Delta\boldsymbol{\varepsilon}^\text{p}, \Delta p\right) = \begin{cases} \sigma_0 \Delta p & \text{s.t. } \pi_{\bar{G}}(\Delta\boldsymbol{\varepsilon}^\text{p}) \leq \Delta p \\ +\infty \end{cases}$

where $\pi_{\bar{G}}(\boldsymbol{\varepsilon}) = \sup\limits_{\boldsymbol{\sigma}\in\bar{G}} \boldsymbol{\sigma} : \boldsymbol{\varepsilon}$ is the support function of the unit plastic domain $\bar{G}$

Auxiliary problem and dual principle

In a **strain**-driven setting, we solve the **local inner** problem at **given** ε:

$$\underset{\varepsilon^{\mathsf{p}}, p}{\arg\min}\, \psi_{\mathsf{el}}(\varepsilon - \varepsilon^{\mathsf{p}}) + \psi_{\mathsf{h}}(\varepsilon^{\mathsf{p}}, p) + \phi\left(\varepsilon^{\mathsf{p}} - \varepsilon^{\mathsf{p}}_n, p - p_n\right)$$

Auxiliary problem and dual principle

In a **strain**-driven setting, we solve the **local inner** problem at **given** $\boldsymbol{\varepsilon}$:

$$\underset{\boldsymbol{\varepsilon}^{\mathsf{p}},p}{\arg\min}\, \psi_{\mathsf{el}}(\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}^{\mathsf{p}}) + \psi_{\mathsf{h}}(\boldsymbol{\varepsilon}^{\mathsf{p}}, p) + \phi\left(\boldsymbol{\varepsilon}^{\mathsf{p}}-\boldsymbol{\varepsilon}^{\mathsf{p}}_n, p-p_n\right)$$

The latter enjoys the following **dual formulation**:

$$\begin{array}{rl} \underset{\boldsymbol{\sigma},\boldsymbol{Y},R}{\arg\min} & \psi^*_{\mathsf{el}}(\boldsymbol{\sigma}) + \psi^*_{\mathsf{h}}(\boldsymbol{Y}, R) - \boldsymbol{\sigma}(\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}^{\mathsf{p}}_n) - \boldsymbol{Y}\boldsymbol{\varepsilon}^{\mathsf{p}}_n - Rp_n \\ \text{s.t.} & \boldsymbol{\sigma}-\boldsymbol{Y} \in (\sigma_0+R)\bar{G} \end{array}$$

where ψ^*_{el}, ψ^*_{h} are dual elastic and hardening potentials

Auxiliary problem and dual principle

In a **strain**-driven setting, we solve the **local inner** problem at **given** ε:

$$\underset{\varepsilon^{\mathsf{p}},p}{\arg\min}\, \psi_{\mathsf{el}}(\varepsilon - \varepsilon^{\mathsf{p}}) + \psi_{\mathsf{h}}(\varepsilon^{\mathsf{p}}, p) + \phi\left(\varepsilon^{\mathsf{p}} - \varepsilon_n^{\mathsf{p}}, p - p_n\right)$$

The latter enjoys the following **dual formulation**:

$$\begin{aligned} \underset{\boldsymbol{\sigma},\boldsymbol{Y},R}{\arg\min} \quad & \psi_{\mathsf{el}}^*(\boldsymbol{\sigma}) + \psi_{\mathsf{h}}^*(\boldsymbol{Y}, R) - \boldsymbol{\sigma}(\varepsilon - \varepsilon_n^{\mathsf{p}}) - \boldsymbol{Y}\varepsilon_n^{\mathsf{p}} - Rp_n \\ \text{s.t.} \quad & \boldsymbol{\sigma} - \boldsymbol{Y} \in (\sigma_0 + R)\bar{G} \end{aligned}$$

where ψ_{el}^*, ψ_{h}^* are dual elastic and hardening potentials

Perfect-plastic case and linear elasticity:

$$\begin{aligned} \underset{\boldsymbol{\sigma}}{\arg\min} \quad & \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{C}^{-1}\boldsymbol{\sigma} - \boldsymbol{\sigma}(\varepsilon - \varepsilon_n^{\mathsf{p}}) \\ \text{s.t.} \quad & \boldsymbol{\sigma} \in \sigma_0 G \end{aligned} \quad = \quad \begin{aligned} \underset{\boldsymbol{\sigma}}{\arg\min} \quad & \frac{1}{2}(\boldsymbol{\sigma} - \boldsymbol{\sigma}^{\text{elas}})\boldsymbol{C}^{-1}(\boldsymbol{\sigma} - \boldsymbol{\sigma}^{\text{elas}}) \\ \text{s.t.} \quad & \boldsymbol{\sigma} \in \sigma_0 G \end{aligned}$$

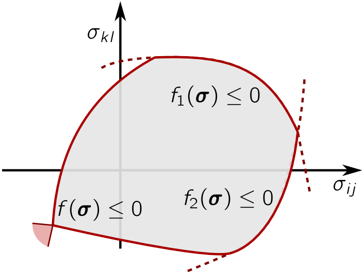
is a **closest-point projection** [Simo & Hugues, 1998] of the elastic predictor $\boldsymbol{\sigma}^{\text{elas}} = \boldsymbol{C}(\varepsilon - \varepsilon_n^{\mathsf{p}}) = \sigma_n + \boldsymbol{C}\Delta\varepsilon$ onto the yield surface $\sigma_0\bar{G}$

Return mapping and convex optimization solvers

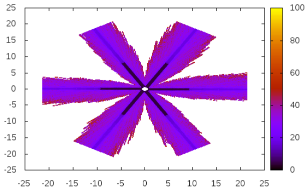
Return mapping as a **convex optimization problem**

$$\boxed{\begin{aligned}\boldsymbol{\sigma} = F(\Delta\boldsymbol{\varepsilon};\boldsymbol{\sigma}_n) = \operatorname*{arg\,min}_{\boldsymbol{\sigma}} \quad & \frac{1}{2}(\boldsymbol{\sigma}-\boldsymbol{\sigma}^{\text{elas}})^{\mathsf{T}}\boldsymbol{C}^{-1}(\boldsymbol{\sigma}-\boldsymbol{\sigma}^{\text{elas}}) \\ \text{s.t.} \quad & \boldsymbol{\sigma}\in\mathcal{G}\end{aligned}}$$

Newton method for smooth $\mathcal{G}$ or **convex optimization solvers** for non-smooth/multisurface plasticity



(a) Non-smooth plastic yield surface



(b) RM iterations to project trial stress on Hosford surface [Helfer, MFront doc.])

Conic programming solvers

Falls into the class of **conic optimization** (**non-smooth** but **structured**)

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

where $\mathcal{K}$ is made of elementary **cones** ($\mathbb{R}^{n,+}$, quadratic, SDP, etc.)
solved efficiently with **primal-dual interior-point methods** (e.g. `cvxpy`)

Conic programming solvers

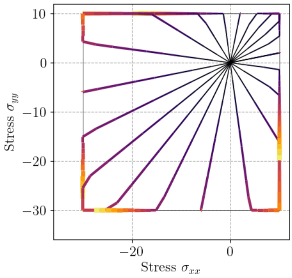
Falls into the class of **conic optimization** (**non-smooth** but **structured**)

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

where $\mathcal{K}$ is made of elementary **cones** ($\mathbb{R}^{n,+}$, quadratic, SDP, etc.)
solved efficiently with **primal-dual interior-point methods** (e.g. `cvxpy`)

e.g. **Rankine**: $f(\boldsymbol{\sigma}) \leq 0 \Leftrightarrow \begin{cases} \max \sigma_I \leq f_t \\ \min \sigma_I \geq -f_c \end{cases}$

```
import cvxpy as cp
...
def set_cvxpy_model(self):
    self.sig = cp.Variable((3,))
    self.sig_el = cp.Parameter((3,))
    obj = 0.5 * cp.quad_form(self.sig - self.sig_el, C_inv)
    Sig = to_mat(self.sig)
    cons = [cp.lambda_max(Sig) <= self.ft,
            cp.lambda_min(Sig) >= -self.fc]
    self.prob = cp.Problem(cp.Minimize(obj), cons)
```



Outline

Learning the elastoplastic return mapping

Objective: **learn** the **elastoplastic mapping** F from **data**
Inputs = strain increment $\Delta\boldsymbol{\varepsilon}$ and previous stress $\boldsymbol{\sigma}_n$
Ouptut = plastically admissible stress

Learning the elastoplastic return mapping

Objective: **learn** the **elastoplastic mapping** F from **data**
Inputs = strain increment $\Delta\boldsymbol{\varepsilon}$ and previous stress $\boldsymbol{\sigma}_n$
Ouptut = plastically admissible stress

Method: choose an approximating mapping $\mathsf{F}_{\boldsymbol{\theta}}$ and learn parameters $\boldsymbol{\theta}$

Learning the elastoplastic return mapping

Objective: **learn** the **elastoplastic mapping** F from **data**
Inputs = strain increment $\Delta\boldsymbol{\varepsilon}$ and previous stress $\boldsymbol{\sigma}_n$
Ouptut = plastically admissible stress

Method: choose an approximating mapping $\mathsf{F}_{\boldsymbol{\theta}}$ and learn parameters $\boldsymbol{\theta}$
Observation: F is a **complex activation function**

ReLU

ReLU is a **simple activation function**: $\mathrm{ReLU}(\boldsymbol{x}) = \max\{\boldsymbol{x}, 0\}$...
but can also be characterized as:

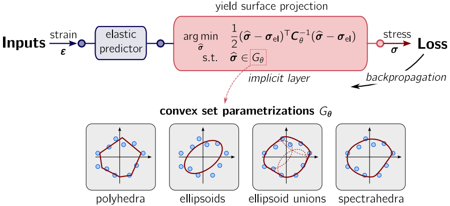
$$\begin{array}{rl} \mathrm{ReLU}(\boldsymbol{x}) = \displaystyle\min_{\boldsymbol{y}} & \dfrac{1}{2}\|\boldsymbol{y}-\boldsymbol{x}\|_2^2 \\ \text{s.t.} & \boldsymbol{y} \geq 0 \end{array}$$

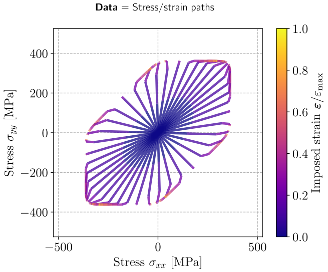
Implicit layer framework

Proposal: use **convex optimization layers** [Agrawal et al., 2019]

$$\boldsymbol{\sigma}_{\boldsymbol{\theta}} = \mathsf{F}_{\boldsymbol{\theta}}(\Delta\boldsymbol{\varepsilon};\boldsymbol{\sigma}_n) = \begin{array}[t]{rl} \displaystyle\operatorname*{arg\,min}_{\boldsymbol{\sigma}} & \dfrac{1}{2}(\boldsymbol{\sigma}-\boldsymbol{\sigma}_{\boldsymbol{\theta}}^{\text{elas}})^{\mathsf{T}}\boldsymbol{C}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{\sigma}-\boldsymbol{\sigma}_{\boldsymbol{\theta}}^{\text{elas}}) \\ \text{s.t.} & \boldsymbol{\sigma} \in G_{\boldsymbol{\theta}} \end{array}$$

- $\boldsymbol{\sigma}_{\boldsymbol{\theta}}^{\text{elas}} = \boldsymbol{\sigma}_n + \boldsymbol{C}_{\boldsymbol{\theta}}\Delta\boldsymbol{\varepsilon}$: elastic predictor
- $\boldsymbol{C}_{\boldsymbol{\theta}}$: parametrized elastic stiffness tensor
- $G_{\boldsymbol{\theta}}$ is a parametrized convex set approximating ground-truth yield surface $\mathcal{G}$





Implementation aspects

- **Learning framework** using `cvxpylayers` in `pytorch`
- **Stiffness parametrization**: use upper-triangular Cholesky factor $\boldsymbol{L}_{\boldsymbol{\theta}}$ so that $\boldsymbol{C}_{\boldsymbol{\theta}} = \boldsymbol{L}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{L}_{\boldsymbol{\theta}} \succeq 0$
- **Loss function**: stress MSE

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{N_{\textbf{train}}} \sum_{k=1}^{N_{\textbf{train}}} \|\boldsymbol{\sigma}^{(k)} - \boldsymbol{\sigma}_\theta(\Delta\boldsymbol{\varepsilon}^{(k)}, \boldsymbol{\sigma}_n^{(k)})\|_2^2$$

Implementation aspects

- **Learning framework** using `cvxpylayers` in `pytorch`
- **Stiffness parametrization**: use upper-triangular Cholesky factor $\boldsymbol{L}_{\boldsymbol{\theta}}$ so that $\boldsymbol{C}_{\boldsymbol{\theta}} = \boldsymbol{L}_{\boldsymbol{\theta}}^{\mathsf{T}} \boldsymbol{L}_{\boldsymbol{\theta}} \succeq 0$
- **Loss function**: stress MSE

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{N_{\textbf{train}}} \sum_{k=1}^{N_{\textbf{train}}} \|\boldsymbol{\sigma}^{(k)} - \boldsymbol{\sigma}_\theta(\Delta\boldsymbol{\varepsilon}^{(k)}, \boldsymbol{\sigma}_n^{(k)})\|_2^2$$

- **Gradient computations**: computing $\partial_{\boldsymbol{\theta}} \boldsymbol{\sigma}_{\boldsymbol{\theta}}$ is not trivial since there is no explicit expression $\Rightarrow$ use **implicit differentiation**
  Differentiating conic programs is covered in [Agrawal et al., 2019] and available in `cvxpy[layers]`

Implementation aspects

- **Learning framework** using `cvxpylayers` in `pytorch`
- **Stiffness parametrization**: use upper-triangular Cholesky factor $\boldsymbol{L}_{\boldsymbol{\theta}}$ so that $\boldsymbol{C}_{\boldsymbol{\theta}} = \boldsymbol{L}_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{L}_{\boldsymbol{\theta}} \succeq 0$
- **Loss function**: stress MSE

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{N_{\textbf{train}}} \sum_{k=1}^{N_{\textbf{train}}} \|\boldsymbol{\sigma}^{(k)} - \boldsymbol{\sigma}_\theta(\Delta\boldsymbol{\varepsilon}^{(k)}, \boldsymbol{\sigma}_n^{(k)})\|_2^2$$

- **Gradient computations**: computing $\partial_{\boldsymbol{\theta}}\boldsymbol{\sigma}_{\boldsymbol{\theta}}$ is not trivial since there is no explicit expression $\Rightarrow$ use **implicit differentiation**
  Differentiating conic programs is covered in [Agrawal et al., 2019] and available in `cvxpy[layers]`
- $G_{\boldsymbol{\theta}}$: **convex set parametrization** ?

  convex set regression *is a hard problem [Goyal & Rademacher, 2009]*

Parametric convex yield surfaces

Hypotheses: $G_{\theta} \in \mathbb{R}^n$, convex and contains 0 in its interior
Objective: find an **expressive** family of **approximations** of convex sets with **low number of parameters**

Parametric convex yield surfaces

Hypotheses: $G_{\boldsymbol{\theta}} \in \mathbb{R}^n$, convex and contains 0 in its interior

Objective: find an **expressive** family of **approximations** of convex sets with **low number of parameters** e.g.

- **polyhedra** with m hyperplanes:

$$\mathcal{P}_{\boldsymbol{\theta}}^m = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{\sigma} \leq 1\} \qquad \dim \boldsymbol{\theta} = nm$$

Parametric convex yield surfaces

Hypotheses: $G_{\boldsymbol{\theta}} \in \mathbb{R}^n$, convex and contains 0 in its interior

Objective: find an **expressive** family of **approximations** of convex sets with **low number of parameters** e.g.

- **polyhedra** with m hyperplanes:

$$\mathcal{P}_{\boldsymbol{\theta}}^m = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{\sigma} \leq 1\} \qquad \dim \boldsymbol{\theta} = nm$$

- **convex union of m ellipsoids** [Bleyer & de Buhan, 2013]

$$\mathcal{E}_{\boldsymbol{\theta}} = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \|\boldsymbol{Q}_{\boldsymbol{\theta}}(\boldsymbol{\sigma} - \boldsymbol{c}_{\boldsymbol{\theta}})\|_2 \leq 1\} \qquad \dim \boldsymbol{\theta} = mn(n+3)/2$$

$$\mathcal{C}_{\boldsymbol{\theta}}^m = \operatorname{conv}\left(\bigcup_{i=1}^m \mathcal{E}_{\boldsymbol{\theta}}^{(i)}\right)$$

Parametric convex yield surfaces

Hypotheses: $G_{\boldsymbol{\theta}} \in \mathbb{R}^n$, convex and contains 0 in its interior

Objective: find an **expressive** family of **approximations** of convex sets with **low number of parameters** e.g.

- **polyhedra** with m hyperplanes:

$$\mathcal{P}_{\boldsymbol{\theta}}^m = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{\sigma} \leq 1\} \qquad \dim \boldsymbol{\theta} = nm$$

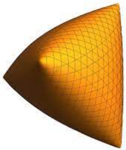
- **convex union of m ellipsoids** [Bleyer & de Buhan, 2013]

$$\mathcal{E}_{\boldsymbol{\theta}} = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \|\boldsymbol{Q}_{\boldsymbol{\theta}}(\boldsymbol{\sigma} - \boldsymbol{c}_{\boldsymbol{\theta}})\|_2 \leq 1\} \qquad \dim \boldsymbol{\theta} = mn(n+3)/2$$

$$\mathcal{C}_{\boldsymbol{\theta}}^m = \operatorname{conv}\left(\bigcup_{i=1}^m \mathcal{E}_{\boldsymbol{\theta}}^{(i)}\right)$$

- **spectrahedra** [Vinzant, 2020]:

$$\mathcal{S}_{\boldsymbol{\theta}}^m = \left\{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \sum_{i=1}^n \sigma_i \boldsymbol{A}_{\boldsymbol{\theta}}^{(i)} + \boldsymbol{I} \succeq 0\right\} \qquad \dim \boldsymbol{\theta} = nm(m+1)/2$$



Parametric convex yield surfaces

Hypotheses: $G_{\boldsymbol{\theta}} \in \mathbb{R}^n$, convex and contains 0 in its interior

Objective: find an **expressive** family of **approximations** of convex sets with **low number of parameters** e.g.

- **polyhedra** with m hyperplanes:

$$\mathcal{P}^m_{\boldsymbol{\theta}} = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \boldsymbol{A}_{\boldsymbol{\theta}}\boldsymbol{\sigma} \leq 1\} \qquad \dim \boldsymbol{\theta} = nm$$

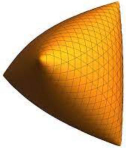
- **convex union of m ellipsoids** [Bleyer & de Buhan, 2013]

$$\mathcal{E}_{\boldsymbol{\theta}} = \{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \|\boldsymbol{Q}_{\boldsymbol{\theta}}(\boldsymbol{\sigma} - \boldsymbol{c}_{\boldsymbol{\theta}})\|_2 \leq 1\} \qquad \dim \boldsymbol{\theta} = mn(n+3)/2$$

$$\mathcal{C}^m_{\boldsymbol{\theta}} = \operatorname{conv}\left(\bigcup_{i=1}^m \mathcal{E}^{(i)}_{\boldsymbol{\theta}}\right)$$

- **spectrahedra** [Vinzant, 2020]:

$$\mathcal{S}^m_{\boldsymbol{\theta}} = \left\{\boldsymbol{\sigma} \in \mathbb{R}^n \text{ s.t. } \sum_{i=1}^n \sigma_i \boldsymbol{A}^{(i)}_{\boldsymbol{\theta}} + \boldsymbol{I} \succeq 0\right\} \qquad \dim \boldsymbol{\theta} = nm(m+1)/2$$

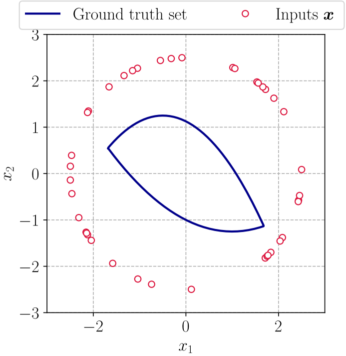


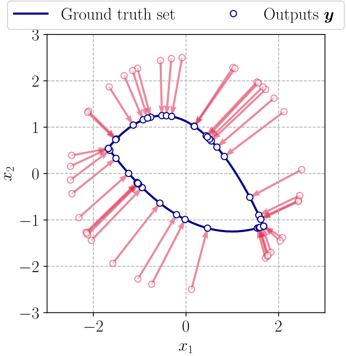
all are **expressible using convex cones** (linear, second-order, PSD resp.)

Outline

Illustrative 2D convex set learning example

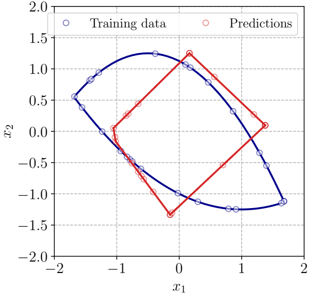
$\mathcal{G}$: intersection of two parabolas, training with 50 points, random initialization



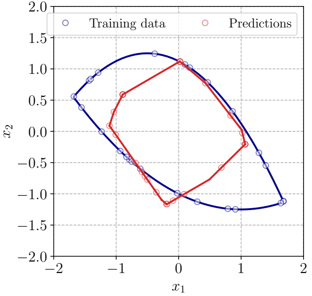


Results

Polyhedra



$m = 8$

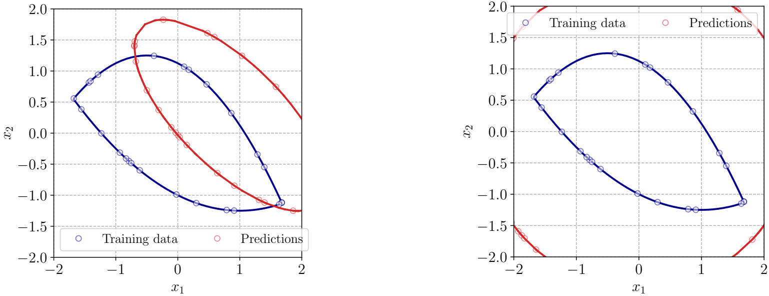


$m = 50$

not very stable learning (many inactive inequalities), poor accuracy

Results

Union of ellipsoids

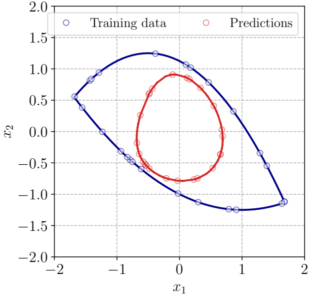


$m = 1$ $m = 8$

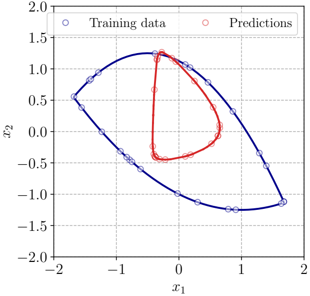
better accuracy, can still have inactive ellipsoids

Results

Spectrahedra



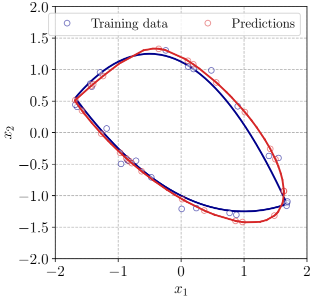
$m = 3$

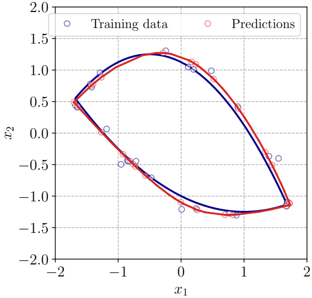


$m = 4$

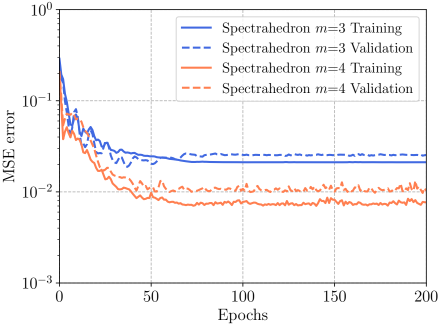
even better accuracy, much smoother learning process

Training curves and noise





10% noise, only **25 data points**

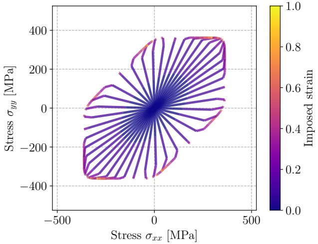


2D biaxial elasto-plasticity

Isotropic elasto-plasticity, **Hosford** yield surface:

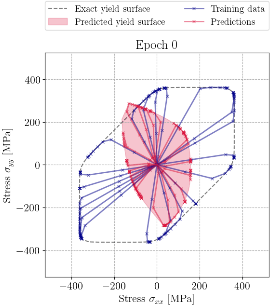
$$\left(\frac{1}{2}\left(|\sigma_I|^a + |\sigma_{II}|^a + |\sigma_I - \sigma_{II}|^a\right)\right)^{1/a} \leq \sigma_0$$

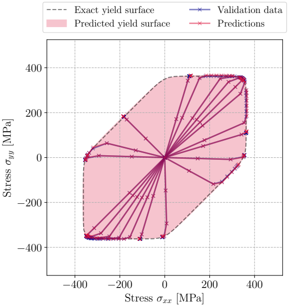
$N_{\text{path}} = 40$ **radial strain paths**
with $N_{\text{incr}} = 10$ **increments**
50% strain paths used for training



Validation on **unseen** paths

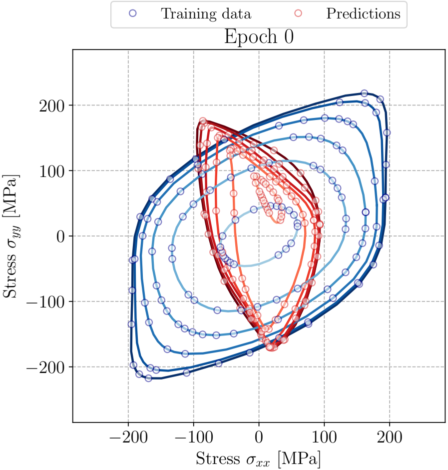
$m = 6$ **Spectrahedron**
45 training parameters





Anisotropic plasticity

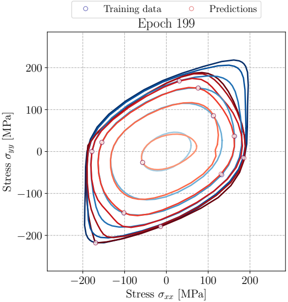
3D plane stress yield surface: **anisotropic** 2090-T3 alloy sheet, **Cazacu 2001** yield surface



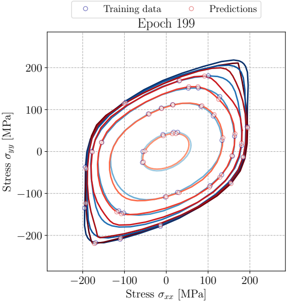
Anisotropic plasticity

3D plane stress yield surface: **anisotropic** 2090-T3 alloy sheet, **Cazacu 2001** yield surface

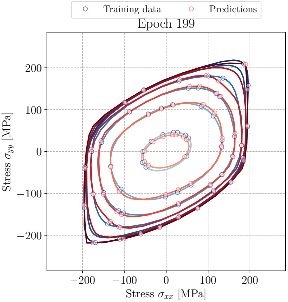
Low data regime



(a) $N_{\text{train}} = 12$ points



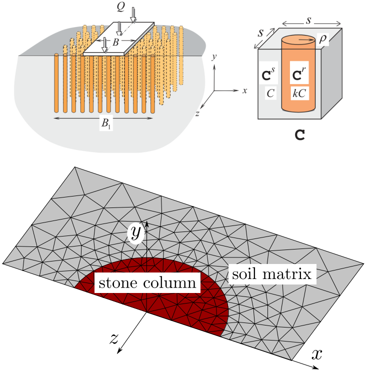
(b) $N_{\text{train}} = 36$ points

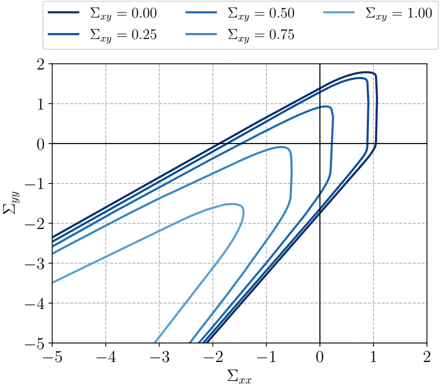


(c) $N_{\text{train}} = 60$ points

Multiscale plasticity

Multi-scale yield surface for a **soil reinforced by stone columns**
FE limit analysis simulations on RVE by [Gueguin et al., 2014]

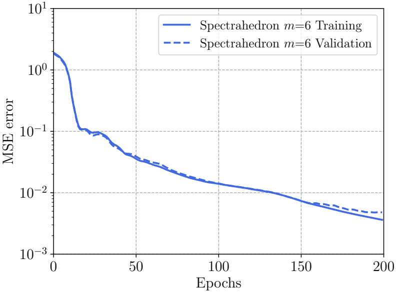


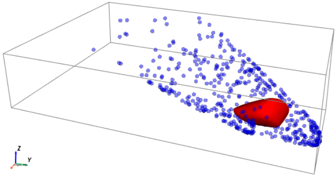


purely cohesive soil reinforced with a periodic arrangement of stone columns made of a frictional granular material

Training

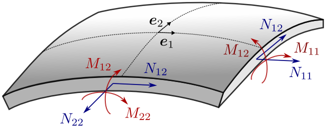
250 data points for training with $m = 6$ **spectrahedron**



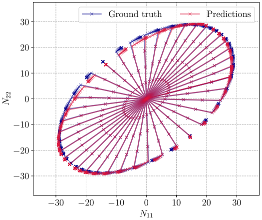


Stress-resultant shell plasticity

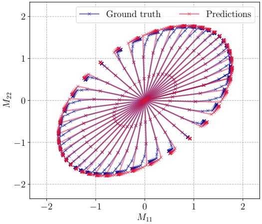
Goal: 6D plastic yield surface in normal force $\boldsymbol{N}$, bending moment $\boldsymbol{M}$ space



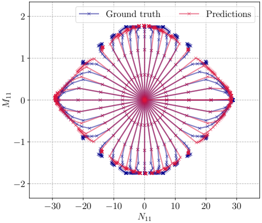
Ground-truth = through-the-thickness integration of homogeneous von Mises shell
Training on radial load paths with **random orientation**



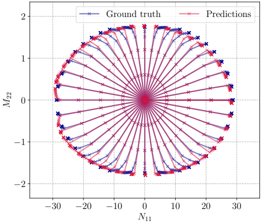
(a) $N_{11} - N_{22}$ plane



(b) $M_{11} - M_{22}$ plane



(c) $N_{11} - M_{11}$ plane



(d) $N_{11} - M_{22}$ plane

Conclusions & Outlook

Conclusions

- elastoplastic integration formulated as a **convex optimization problem**
- learnt using an implicit **convex optimization layer**
- yield surfaces very well approximated using **parametrized spectrahedra**
- strong **robustness in low-data and noisy** regimes due to **embedded mathematical structure**

Preprint available: Jeremy Bleyer, *Learning Elastoplasticity with Implicit Layers*, https://dx.doi.org/10.2139/ssrn.5210734

Perspectives

- including **hardening and viscoplasticity** by stacking implicit layers
- **non associative plasticity** ?
- material **symmetries**
- integration into **FE code**

Conclusions & Outlook

Conclusions

- elastoplastic integration formulated as a **convex optimization problem**
- learnt using an implicit **convex optimization layer**
- yield surfaces very well approximated using **parametrized spectrahedra**
- strong **robustness in low-data and noisy** regimes due to **embedded mathematical structure**

Preprint available: Jeremy Bleyer, *Learning Elastoplasticity with Implicit Layers*, https://dx.doi.org/10.2139/ssrn.5210734

Perspectives

- including **hardening and viscoplasticity** by stacking implicit layers
- **non associative plasticity** ?
- material **symmetries**
- integration into **FE code**

Thank you for your attention !