$\mathcal{MSR}3$ – Sparse Relaxed Regularized Regression for Linear Mixed-Effect Models

Aleksei Sholokhov

Together with James V. Burke, Peng Zheng, Damian Santomauro, and Aleksandr Aravkin

Friday 2nd June, 2023

APPLIED MATHEMATICS
UNIVERSITY of WASHINGTON

IHME

Research Objectives and Plan of the Talk

Objectives:

- Extend $\mathcal{SR}3$ relaxation to linear mixed-effects models - $\mathcal{MSR}3$ (see[1]).
- Develop theoretical foundations for it (see[2]).
- Implement it as a `scikit-learn`-compatible Python package – `pysr3` (see[3]).

Plan of the Talk:

1. Proximal Gradient Descent (PGD) for Linear Mixed-Effects Models (LMEs)
2. $\mathcal{MSR}3$ – PGD on $\mathcal{SR}3$-relaxation for LMEs
3. $\mathcal{MSR}3$-fast – practical algorithm for feature selection

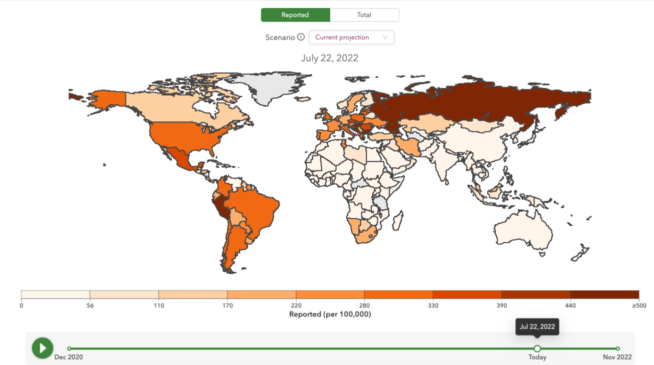
---

[1] Sholokhov, J. V. Burke, et al., A Relaxation Approach to Feature Selection for Linear Mixed Effects Models.
[2] Aravkin et al., Analysis of Relaxation Methods for Feature Selection in Mixed Effects Models.
[3] Sholokhov, Zheng, and Aravkin, "pysr3: A Python Package for Sparse Relaxed Regularized Regression".

Mixed-Effect Models

Mixed-effect models

- Used for analyzing **combined data** across a range of **groups**.
- **Borrow strength** across groups to estimate key statistics.
- Use covariates to separate the **population variability** from the **group variability**.



[4]

---

[4] Picture is taken from covid19.healthdata.org

Linear Mixed-Effect (LME) Models

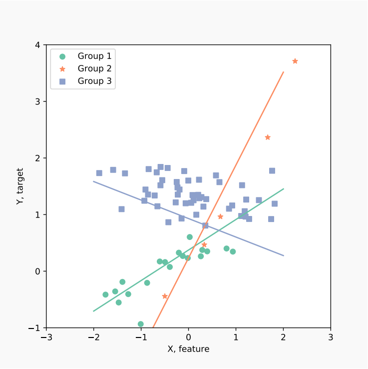
Dataset: m groups (X_i, Z_i, y_i), $\quad i = 1, \ldots m$, each has n_i observations

- $X_i \in \mathbb{R}^{n_i \times p}$ – group i design matrix for fixed features
- $Z_i \in \mathbb{R}^{n_i \times q}$ – group i design matrix for random features
- $y_i \in \mathbb{R}^{n_i}$ – group i observations
- $u_i \in \mathbb{R}^q$ – random effects
- $\Gamma \in \mathbb{R}^{q \times q}$ – covariance matrix of random effects, often $\Gamma = \mathrm{diag}\left((\gamma)\right)$
- $\Lambda_i \in R^{n_i \times n_i}$ – covariance matrix for observation noise

Model:

$$y_i = X_i\beta + \color{red}{Z_i u_i} + \varepsilon_i$$
$$\varepsilon_i \sim \mathcal{N}(0, \Lambda_i)$$
$$\color{red}{u_i \sim \mathcal{N}(0, \Gamma)}$$

Unknowns: β, u_i, γ, sometimes Λ_i.



Negative Log-Likelihood for Mixed-Effect Models

Optimization problem:

$$\mathcal{FS}-\mathcal{LME} \qquad \min_{\beta\in\mathbb{R}^p,\ \gamma\in\mathbb{R}^q_+} \mathcal{L}(\beta,\gamma)+R(\beta,\gamma) \tag{1}$$

Where $\mathcal{L}$:

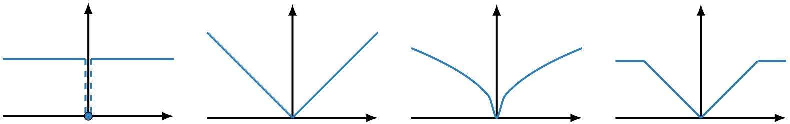
$$\begin{aligned}\mathcal{L}(\beta,\gamma) = \sum_{i=1}^{m} \frac{1}{2}(y_i - X_i\beta)^T(Z_i\Gamma Z_i^T+\Lambda_i)^{-1}(y_i-X_i\beta)+ \\ +\frac{1}{2}\log\det\left(Z_i\Gamma Z_i^T+\Lambda_i\right), \quad \Gamma = \mathrm{diag}\left((\gamma)\right)\end{aligned} \tag{2}$$

- $\mathcal{L}(\beta,\gamma)$ is smooth on its domain, quadratic w.r.t. β and $\bar{\eta}$-weakly-convex w.r.t. γ.
- $R(\beta,\gamma)$ is closed, proper, with easily computed *prox operator*

Regularization

$R(\beta,\gamma)$ is closed, proper, with easily computed *prox operator*

$$\operatorname{prox}_{\alpha R+\delta_{\mathcal{C}}}(\tilde{\beta},\tilde{\gamma}) := \operatorname*{argmin}_{(\beta,\gamma)\in\mathcal{C}} R(\beta,\gamma) + \frac{1}{2\alpha}\|(\beta,\gamma)-(\tilde{\beta},\tilde{\gamma})\|_2^2, \quad \text{where } \mathcal{C} := \mathbb{R}^p \times R_+^q \tag{3}$$



(a) ℓ_0 (b) ℓ_1 (c) $\ell_p,\ p=1/2$ (d) SCAD

Figure: Four commonly-used regularizers which promote sparsity

Proximal Gradient Descent for Feature Selection in MLE

Let

$$x := [\beta, \gamma], \quad \mathcal{C} := \mathbb{R}^p \times R_+^q \tag{4}$$

Optimization problem $\mathcal{FS} - \mathcal{LME}$:

$$\min_{x \in \mathcal{C}} \mathcal{L}(x) + R(x) \tag{5}$$

Proximal Gradient Descent for Feature Selection in MLE

Let

$$x := [\beta, \gamma], \quad \mathcal{C} := \mathbb{R}^p \times R_+^q \tag{4}$$

Optimization problem $\mathcal{FS} - \mathcal{LME}$:

$$\min_{x \in \mathcal{C}} \mathcal{L}(x) + R(x) \tag{5}$$

Algorithm: PGD for standard LMEs

$\beta^+ \leftarrow \beta_0, \quad \gamma^+ \leftarrow \gamma_0, \quad \alpha \leftarrow 1/L$ `// Initialization`

$x^+ = [\beta^+, \gamma^+]$;

while making progress **do**

  $x^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R + \delta_{\mathcal{C}}}(x^+ - \alpha \nabla_x \mathcal{L}(x^+))$ `// PGD iterations`

end

return $x^+ = [\beta^+, \gamma^+]$

Proximal Gradient Descent for Feature Selection in MLE

Algorithm: PGD for standard LMEs

$\beta^+ \leftarrow \beta_0, \quad \gamma^+ \leftarrow \gamma_0, \quad \alpha \leftarrow 1/L$ `// Initialization`

$x^+ = [\beta^+, \gamma^+]$;

while making progress **do**

  $x^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R+\delta_{\mathcal{C}}}(x^+ - \alpha\nabla_x \mathcal{L}(x^+))$ `// PGD iterations`

end

return $x^+ = [\beta^+, \gamma^+]$

Basic Assumptions for the PGD Algorithm (Theorem 10.15 from[5]):

1. R is a closed proper convex function
2. $\mathcal{L}$ is closed and proper, $\mathrm{dom}\,\mathcal{L}$ convex, $\mathrm{dom}\,R \subset \mathit{int}(\mathrm{dom}\,\mathcal{L})$, and $\mathcal{L}$ is L-smooth over $\mathit{int}(\mathrm{dom}\,\mathcal{L})$. $[\times]$.
3. The problem has an optimal solution[6] with an optimal value $\mathcal{L}^*$

Algorithm converges with backtracking[7].

[5] Beck, First-Order Methods in Optimization.

[6] Aravkin et al., Analysis of Relaxation Methods for Feature Selection in Mixed Effects Models.

[7] J. Burke and Engle, "Line Search and Trust-Region Methods for Convex-Composite Optimization".

Proximal Gradient Descent for Feature Selection in MLE

Algorithm: PGD for standard LMEs

$\beta^+ \leftarrow \beta_0, \quad \gamma^+ \leftarrow \gamma_0, \quad \alpha \leftarrow 1/L$ `// Initialization`

$x^+ = [\beta^+, \gamma^+]$;

while making progress **do**

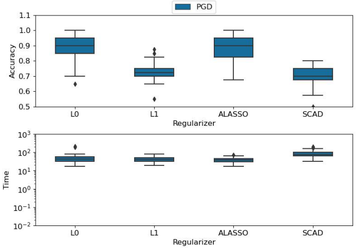
  $x^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R + \delta_{\mathcal{C}}}(x^+ - \alpha \nabla_x \mathcal{L}(x^+))$ `// PGD iterations`

end

return $x^+ = [\beta^+, \gamma^+]$

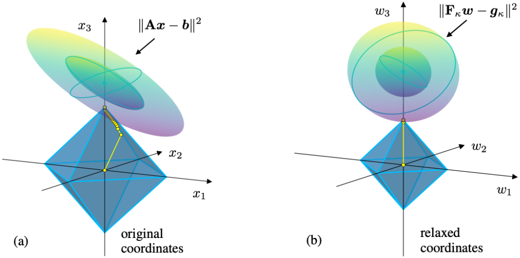
Synthetic Benchmark:

- 100 randomly-generated problems.
- $p = q = 20$.
- $\beta = \gamma = \frac{1}{2}[1, 2, 3, \dots, 10, 0 \dots, 0]$
- 9 groups from 3 to 15 observations
- $X_i \sim \mathcal{N}(0, I)^p$, $Z_i = X_i$, $\varepsilon_i \sim \mathcal{N}(0, 0.3^2 I)$
- Golden search for $\lambda \in [0, 10^5]$
- Final model is chosen to maximize BIC



Sparse Relaxed Regularized Regression ($\mathcal{SR}3$)[5]

Original		Decoupled		Value Function
$\min_x f(x) + R(x)$	$\rightarrow$	$\min_{x,w} f(x) + \frac{\eta}{2}\|x - w\|_2^2 + R(w)$	$\rightarrow$	$\min_w v_\eta(w) + R(w)$



[5] Zheng and Aravkin, "Relax-and-split method for nonconvex inverse problems".

SR3-Relaxation for Mixed-Effect Models ($\mathcal{MSR}3$)

Original problem $\mathcal{FS} - \mathcal{LME}$:

$$\min_{x \in \mathcal{C}} \mathcal{L}(x) + R(x) \tag{4}$$

SR3-Relaxation for Mixed-Effect Models ($\mathcal{MSR}3$)

Original problem $\mathcal{FS}-\mathcal{LME}$:

$$\min_{x\in\mathcal{C}} \mathcal{L}(x) + R(x) \tag{4}$$

Decoupled problem:

$$\min_{x,w\in\mathcal{C}} \mathcal{L}(x) + \frac{\eta}{2}\|x - w\|_2^2 + R(w) \tag{5}$$

where $w = [\tilde{\beta}, \tilde{\gamma}]$.

SR3-Relaxation for Mixed-Effect Models ($\mathcal{MSR}3$)

Original problem $\mathcal{FS} - \mathcal{LME}$:

$$\min_{x \in \mathcal{C}} \mathcal{L}(x) + R(x) \tag{4}$$

Decoupled problem:

$$\min_{x, w \in \mathcal{C}} \mathcal{L}(x) + \frac{\eta}{2}\|x - w\|_2^2 + R(w) \tag{5}$$

where $w = [\tilde{\beta}, \tilde{\gamma}]$.
By partially minimizing w.r.t. x we get a value function $v_\eta(w)$:

$$v_\eta(w) := \min_{x \in \mathcal{C}} \mathcal{L}(x) + \frac{\eta}{2}\|x - w\|_2^2 \tag{6}$$

so $\mathcal{SR}3$-formulation (5) becomes

$$\min_{w \in \mathcal{C}} v_\eta(w) + R(w) \tag{7}$$

SR3-Relaxation for Mixed-Effect Models ($\mathcal{MSR}3$)

Original problem $\mathcal{FS} - \mathcal{LME}$:

$$\min_{x \in \mathcal{C}} \mathcal{L}(x) + R(x) \tag{4}$$

Decoupled problem:

$$\min_{x, w \in \mathcal{C}} \mathcal{L}(x) + \frac{\eta}{2}\|x - w\|_2^2 + R(w) \tag{5}$$

where $w = [\tilde{\beta}, \tilde{\gamma}]$.
By partially minimizing w.r.t. x we get a value function $v_\eta(w)$:

$$v_\eta(w) := \min_{x \in \mathcal{C}} \mathcal{L}(x) + \frac{\eta}{2}\|x - w\|_2^2 \tag{6}$$

so $\mathcal{SR}3$-formulation (5) becomes

$$\min_{w \in \mathcal{C}} v_\eta(w) + R(w) \tag{7}$$

NB: $v_\eta(w)$ is smooth on $\mathcal{C}$ and can be evaluated using Interior Point (IP) method

Value Function of $\mathcal{MSR}3$

$\mathcal{MSR}3$-relaxation replaces the original likelihood $\mathcal{L}$ with a *value function* $v_{\eta,\mu}$:

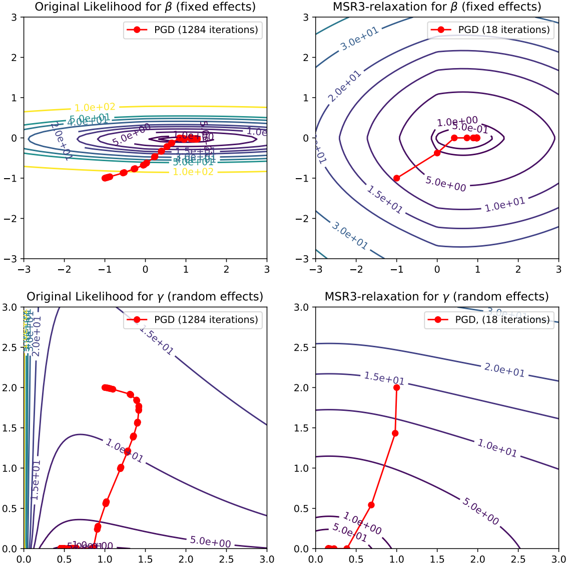
$$v_{\eta,\mu}(w) := \min_{x \in \mathcal{C}} \mathcal{L}(x) + \frac{\eta}{2}\|x - w\|_2^2 + \phi_\mu(x) \tag{8}$$

where *perspective mapping* ϕ_μ replaces $\gamma \geq 0$ with a log-barrier

$$\phi_\mu(x) = \phi_\mu(\gamma) := \begin{cases} -\mu \sum_{i=1}^{q} \ln(\gamma_i/\mu), & \mu > 0 \\ \delta_{\mathbb{R}_+^q}(\gamma), & \mu = 0 \\ +\infty, & \mu < 0 \end{cases} \tag{9}$$

Value Function of $\mathcal{MSR}3$

$$\min_{\beta,\gamma\in\mathcal{C}} \mathcal{L}(\beta,\gamma) + R(\beta,\gamma) \quad \text{vs} \quad \min_{\tilde\beta,\tilde\gamma\in\mathcal{C}} v_{\eta,\mu}(\tilde\beta,\tilde\gamma) + R(\tilde\beta,\tilde\gamma)$$



$\mathcal{MSR}3$: Algorithm

Algorithm: PGD for $\mathcal{MSR}3$

$\tilde{\beta}^+ \leftarrow \tilde{\beta}_0, \quad \tilde{\gamma}^+ \leftarrow \tilde{\gamma}_0, \quad \alpha \leftarrow 1/\eta, \quad \eta > \bar{\eta}$ // Initialization

$\tilde{w}^+ := [\tilde{\beta}^+, \tilde{\gamma}^+], \quad x^+ := [\beta, \gamma]$

while <u>making progress in $\tilde{w}$</u> **do**

  $x^+ \leftarrow$ IP on $\mathcal{L}_{\eta,\mu}(x^+, \tilde{w}^+)$ s.t. $x^+ \in \mathcal{C}$ and $\mu \to 0$ // IP Iterations

  $\nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+) \leftarrow \nabla_{\tilde{w}} \mathcal{L}_{\eta,0}(x^+, \tilde{w}^+)$ // Evaluate Gradient of VF

  $\tilde{w}^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R+\delta_{\mathcal{C}}}(\tilde{w}^+ - \alpha \nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+))$ // PGD on Value Function

end

return $\underline{\tilde{w}^+ = [\tilde{\beta}^+, \tilde{\gamma}^+]}$

$\mathcal{MSR}3$: Algorithm

Algorithm: PGD for $\mathcal{MSR}3$

$\tilde{\beta}^+ \leftarrow \tilde{\beta}_0, \quad \tilde{\gamma}^+ \leftarrow \tilde{\gamma}_0, \quad \alpha \leftarrow 1/\eta, \quad \eta > \bar{\eta}$ `// Initialization`

$\tilde{w}^+ := [\tilde{\beta}^+, \tilde{\gamma}^+], \quad x^+ := [\beta, \gamma]$

while making progress in $\tilde{w}$ **do**

  $x^+ \leftarrow$ IP on $\mathcal{L}_{\eta,\mu}(x^+, \tilde{w}^+)$ s.t. $x^+ \in \mathcal{C}$ and $\mu \to 0$ `// IP Iterations`

  $\nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+) \leftarrow \nabla_{\tilde{w}} \mathcal{L}_{\eta,0}(x^+, \tilde{w}^+)$ `// Evaluate Gradient of VF`

  $\tilde{w}^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R+\delta_{\mathcal{C}}}(\tilde{w}^+ - \alpha \nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+))$ `// PGD on Value Function`

end

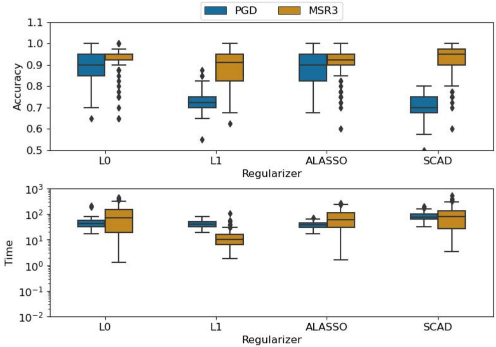
return $\tilde{w}^+ = [\tilde{\beta}^+, \tilde{\gamma}^+]$

Theoretical Results[6]:

1. When $\eta > \bar{\eta}$ the problem has an optimal solution Φ^* (**Theorem 5**)
2. $v_{\eta,\mu}$ is well-defined (**Theorem 5**) and continuously differentiable (**Theorem 10**)
3. When R is 1-coercive $\nabla v_{\eta,\mu}$ is locally $\widetilde{L}$-continuous (**Theorem 14**)
4. Algorithm converges to a stationary point of $v_{\eta,\mu}$ (**Theorem 15**)
5. As $\mu \to 0$ (**Theorem 6**) or $\eta \to \infty$ (**Theorem 7**), cluster points of solutions to $\mathcal{MSR}3$ are FOSPs for $\mathcal{FS} - \mathcal{LME}$

[6] Aleksandr Aravkin et al. Analysis of Relaxation Methods for Feature Selection in Mixed Effects Models. 2022. arXiv: 2209.10575 [stat.ME].

$\mathcal{MSR}3$: Results



$\mathcal{MSR}3$: Algorithm

Algorithm: PGD for $\mathcal{MSR}3$

$\tilde{\beta}^+ \leftarrow \tilde{\beta}_0, \quad \tilde{\gamma}^+ \leftarrow \tilde{\gamma}_0, \quad \alpha \leftarrow 1/\eta, \quad \eta > \bar{\eta}$ `// Initialization`

$\tilde{w}^+ := [\tilde{\beta}^+, \tilde{\gamma}^+], \quad x^+ := [\beta, \gamma]$

while making progress in $\tilde{w}$ **do**

- $x^+ \leftarrow$ IP on $\mathcal{L}_{\eta,\mu}(x^+, \tilde{w}^+)$ s.t. $x^+ \in \mathcal{C}$ and $\mu \to 0$ `// IP Iterations`
- $\nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+) \leftarrow \nabla_{\tilde{w}} \mathcal{L}_{\eta,0}(x^+, \tilde{w}^+)$ `// Evaluate Gradient`
- $\tilde{w}^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R+\delta_{\mathcal{C}}}(\tilde{w}^+ - \alpha \nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+))$ `// PGD on Value Function`

end

return $\tilde{w}^+ = [\tilde{\beta}^+, \tilde{\gamma}^+]$

$\mathcal{MSR}3$: Algorithm

Algorithm: PGD for $\mathcal{MSR}3$

$\tilde{\beta}^+ \leftarrow \tilde{\beta}_0, \quad \tilde{\gamma}^+ \leftarrow \tilde{\gamma}_0, \quad \alpha \leftarrow 1/\eta, \quad \eta > \bar{\eta}$ // Initialization

$\tilde{w}^+ := [\tilde{\beta}^+, \tilde{\gamma}^+], \quad x^+ := [\beta, \gamma]$

while making progress in $\tilde{w}$ **do**

  $x^+ \leftarrow$ IP on $\mathcal{L}_{\eta,\mu}(x^+, \tilde{w}^+)$ s.t. $x^+ \in \mathcal{C}$ and $\mu \to 0$ // IP Iterations

  $\nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+) \leftarrow \nabla_{\tilde{w}} \mathcal{L}_{\eta,0}(x^+, \tilde{w}^+)$ // Evaluate Gradient

  $\tilde{w}^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R+\delta_{\mathcal{C}}}(\tilde{w}^+ - \alpha \nabla_{\tilde{w}} v_{\eta,0}(\tilde{w}^+))$ // PGD on Value Function

end

return $\tilde{w}^+ = [\tilde{\beta}^+, \tilde{\gamma}^+]$

Key Observation: $\nabla_{\tilde{w}} v(\tilde{w})$ does not need to be evaluated exactly. We only need to come close enough to the central path.

$\mathcal{MSR}3$-fast: Algorithm

Algorithm: $\mathcal{MSR}3$-fast

$\tilde{\beta}^+ \leftarrow \tilde{\beta}_0, \quad \tilde{\gamma}^+ \leftarrow \tilde{\gamma}_0, \quad \alpha \leftarrow 1/\eta, \quad \eta > \bar{\eta}$ `// Initialization`

$\tilde{w}^+ := [\tilde{\beta}^+, \tilde{\gamma}^+], \quad x^+ := [\beta, \gamma]$

while making progress **do**

  while not close enough to the central path **do**

    $x^+ \leftarrow$ IP iteration on $\mathcal{L}_{\eta,\mu}(x^+, \tilde{w}^+)$ s.t. $x^+ \in \mathcal{C}$ `// IP Iterations`

  end

  Decrease μ

  $\nabla_{\tilde{w}} v_{\eta,\mu}(\tilde{w}^+) \leftarrow \nabla_{\tilde{w}} \mathcal{L}_{\eta,\mu}(x^+, \tilde{w}^+)$ `// Evaluate Gradient`

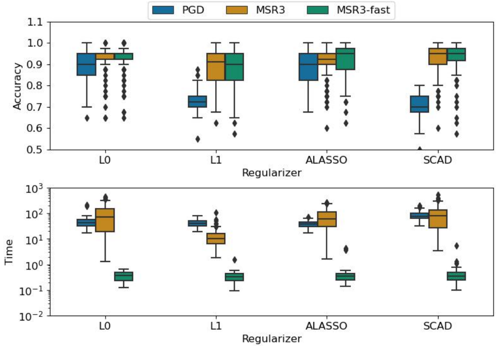
  $\tilde{w}^+ \leftarrow \mathrm{prox}_{\alpha^{-1}R+\delta_{\mathcal{C}}}(\tilde{w}^+ - \alpha \nabla_{\tilde{w}} v_{\eta,\mu}(\tilde{w}^+))$ `// PGD on Value Function`

end

return $\tilde{w}^+ = [\tilde{\beta}^+, \tilde{\gamma}^+]$

$\mathcal{MSR}3$-fast: Results

- The number of fixed effects p and random effects q is 20.
- $\beta = \gamma = \frac{1}{2}[1, 2, 3, \ldots, 10, 0 \ldots, 0]$
- 9 groups with sizes [10, 15, 4, 8, 3, 5, 18, 9, 6]
- $X_i \sim \mathcal{N}(0, I)^p$, $Z_i = X_i$, $\varepsilon_i \sim \mathcal{N}(0, 0.3^2 I)$
- Each experiment is repeated 100 times.
- Grid-search for $\eta \in [10^{-4}, 10^2]$, golden search for $\lambda \in [0, 10^5]$
- Final model is chosen to maximize BIC



+ $\mathcal{MSR}3$-relaxation improves feature selection performance of the original likelihood.
+ $\mathcal{MSR}3$-fast optimization accelerates the compute time by $\sim 10^2$.
− Initialization of η is problem-specific

Comparison to Other Libraries

Algorithm	MSR3-Fast (ℓ_1)	`glmmLasso`[7][8]	`lmmLasso`[9][10]	PGD (ℓ_1)
Accuracy, %	**88**	48	66	73
FE Accuracy, %	**86**	52	47	56
RE Accuracy, %	**91**	45	84	**91**
Time, sec	**0.19**	1.37	11.51	38.39
Iterations, num	34	50	-	7693

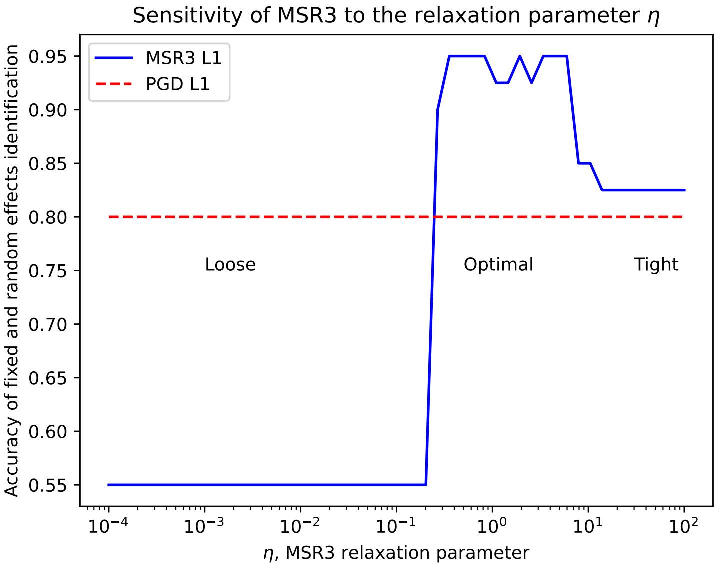
[7] https://rdrr.io/cran/glmmLasso/man/glmmLasso.html

[8] Groll and Tutz, "Variable selection for generalized linear mixed models by L 1-penalized estimation".

[9] https://rdrr.io/cran/lmmlasso/

[10] Schelldorfer, Bühlmann, and DE GEER, "Estimation for high-dimensional linear mixed-effects models using l1-penalization".

Choice of η



ℓ_0-based Covariate Selection for Bullying Study from GBD

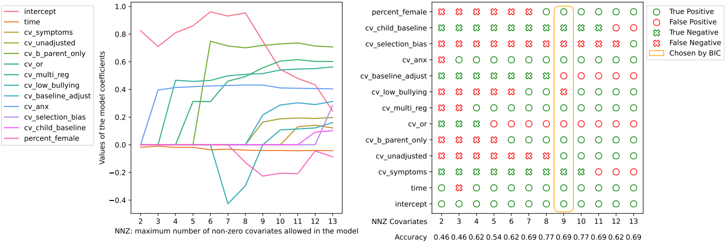
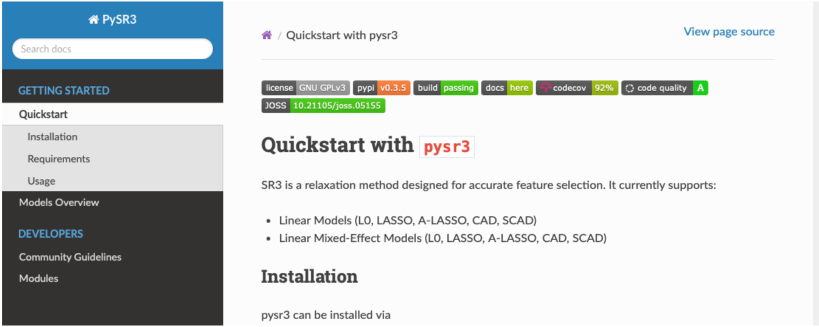


Figure: Fixed and random covariate selection for Bullying dataset[11]. The model selected 9 covariates, 7 of which were historically significant, and did not select 4 covariates, 1 of which was historically significant.

[11] Institute for Health Metrics and Evaluation (IHME). Bullying Victimization Relative Risk Bundle GBD 2020. Seattle, United States of America (USA), 2021.

Thank you!



The code is available on GitHub: https://github.com/aksholokhov/pysr3[12]

- All estimators are fully compatible to `scikit-learn` library.
- Implements SR3 for linear, generalized-linear, and linear mixed-effect models.
- Has tutorials, tests, and documentation.

[12] Aleksei Sholokhov, Peng Zheng, and Aleksandr Aravkin. "pysr3: A Python Package for Sparse Relaxed Regularized Regression". In: Journal of Open Source Software 8.84 (2023), p. 5155.

[13] Wikipedia: Navier-Stokes

[14] tibco.com

[15] COMSOL Simulations

Data-Driven Modeling of Physical Systems

First-Principle Models

- Require extensive knowledge of the phenomenon
- Require a lot of compute for simulating large-scale phenomena

$$\begin{cases} \rho\dfrac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot\boldsymbol{\sigma}(\mathbf{u},p) = \mathbf{f} & \text{in } \Omega\times(0,T) \\ \nabla\cdot\mathbf{u} = 0 & \text{in } \Omega\times(0,T) \\ \mathbf{u} = \mathbf{g} & \text{on } \Gamma_D\times(0,T) \\ \boldsymbol{\sigma}(\mathbf{u},p)\hat{\mathbf{n}} = \mathbf{h} & \text{on } \Gamma_N\times(0,T) \\ \mathbf{u}(0) = \mathbf{u}_0 & \text{in } \Omega\times\{0\} \end{cases}$$

[13] Wikipedia: Navier-Stokes

[14] tibco.com

[15] COMSOL Simulations

Data-Driven Modeling of Physical Systems

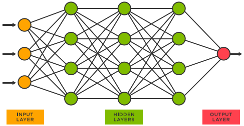
First-Principle Models

- Require extensive knowledge of the phenomenon
- Require a lot of compute for simulating large-scale phenomena

$$\begin{cases} \rho \dfrac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} - \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, p) = \mathbf{f} & \text{in } \Omega \times (0, T) \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \times (0, T) \\ \mathbf{u} = \mathbf{g} & \text{on } \Gamma_D \times (0, T) \\ \boldsymbol{\sigma}(\mathbf{u}, p)\hat{\mathbf{n}} = \mathbf{h} & \text{on } \Gamma_N \times (0, T) \\ \mathbf{u}(0) = \mathbf{u}_0 & \text{in } \Omega \times \{0\} \end{cases}$$

Data-Driven Models

- Require a lot of data
- Often struggle to extrapolate



[13] Wikipedia: Navier-Stokes

[14] tibco.com

[15] COMSOL Simulations

Data-Driven Modeling of Physical Systems

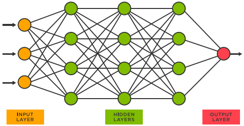
First-Principle Models

- Require extensive knowledge of the phenomenon
- Require a lot of compute for simulating large-scale phenomena

$$
\begin{cases}
\rho \dfrac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot\boldsymbol{\sigma}(\mathbf{u},p) = \mathbf{f} & \text{in } \Omega\times(0,T) \\
\nabla\cdot\mathbf{u} = 0 & \text{in } \Omega\times(0,T) \\
\mathbf{u} = \mathbf{g} & \text{on } \Gamma_D\times(0,T) \\
\boldsymbol{\sigma}(\mathbf{u},p)\hat{\mathbf{n}} = \mathbf{h} & \text{on } \Gamma_N\times(0,T) \\
\mathbf{u}(0) = \mathbf{u}_0 & \text{in } \Omega\times\{0\}
\end{cases}
$$

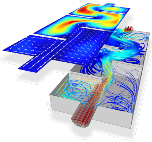
Data-Driven Models

- Require a lot of data
- Often struggle to extrapolate



Hybrid Models

- Incorporate elements of both approaches
- Supplement data with knowledge or priors



Pictures sources: [13],[14],[15]

[13] Wikipedia: Navier-Stokes

[14] tibco.com

[15] COMSOL Simulations

Incorporating Knowledge of Physics into Neural Networks

[16] Geiger and Smidt, "e3nn: Euclidean neural networks".

[17] Finzi et al., "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data".

[18] Chidester, Do, and Ma, Rotation Equivariance and Invariance in Convolutional Neural Networks.

[19] Champion et al., "Data-driven discovery of coordinates and governing equations".

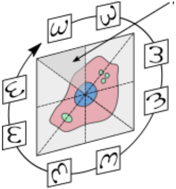
[20] Schmidt and Lipson, "Distilling free-form natural laws from experimental data".

[21] Raissi, Perdikaris, and Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations".

[22] Rackauckas et al., "Universal differential equations for scientific machine learning".

Incorporating Knowledge of Physics into Neural Networks

1. **Symmetry Based:** incorporate symmetries and conservation laws as hard constraints into a network: E3NNs[16], LieConv[17], RiCNN[18]



[16] Geiger and Smidt, "e3nn: Euclidean neural networks".

[17] Finzi et al., "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data".

[18] Chidester, Do, and Ma, Rotation Equivariance and Invariance in Convolutional Neural Networks.

[19] Champion et al., "Data-driven discovery of coordinates and governing equations".

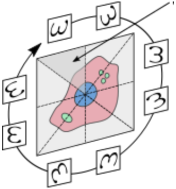
[20] Schmidt and Lipson, "Distilling free-form natural laws from experimental data".

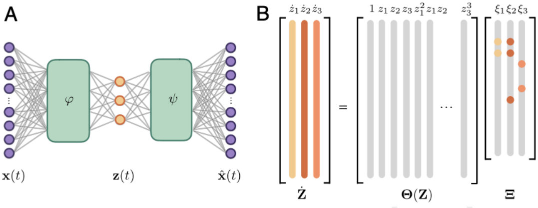
[21] Raissi, Perdikaris, and Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations".

[22] Rackauckas et al., "Universal differential equations for scientific machine learning".

Incorporating Knowledge of Physics into Neural Networks

1. **Symmetry Based:** incorporate symmetries and conservation laws as hard constraints into a network: E3NNs[16], LieConv[17], RiCNN[18]
2. **Model Discovery Based:** use a library of terms to discover simple equations: SINDy[19], Genetic Programming[20]





---

[16] Geiger and Smidt, "e3nn: Euclidean neural networks".

[17] Finzi et al., "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data".

[18] Chidester, Do, and Ma, Rotation Equivariance and Invariance in Convolutional Neural Networks.

[19] Champion et al., "Data-driven discovery of coordinates and governing equations".

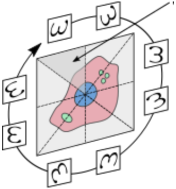
[20] Schmidt and Lipson, "Distilling free-form natural laws from experimental data".

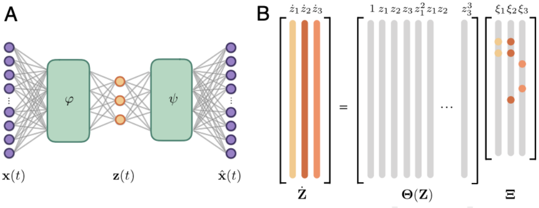
[21] Raissi, Perdikaris, and Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations".

[22] Rackauckas et al., "Universal differential equations for scientific machine learning".

Incorporating Knowledge of Physics into Neural Networks

1. **Symmetry Based:** incorporate symmetries and conservation laws as hard constraints into a network: E3NNs[16], LieConv[17], RiCNN[18]
2. **Model Discovery Based:** use a library of terms to discover simple equations: SINDy[19], Genetic Programming[20]
3. **Equations Based:** incorporate first-principle models to aid training of networks: PINNs[21], UDEs[22]





[16] Geiger and Smidt, "e3nn: Euclidean neural networks".

[17] Finzi et al., "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data".

[18] Chidester, Do, and Ma, Rotation Equivariance and Invariance in Convolutional Neural Networks.

[19] Champion et al., "Data-driven discovery of coordinates and governing equations".

[20] Schmidt and Lipson, "Distilling free-form natural laws from experimental data".

[21] Raissi, Perdikaris, and Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations".

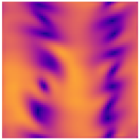
[22] Rackauckas et al., "Universal differential equations for scientific machine learning".

Reduced-Order Models (ROMs)

$x \in \mathbb{R}^n$

$$\frac{dx}{dt} = f(x)$$

x_0



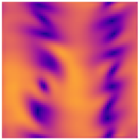
Reduced-Order Models (ROMs)

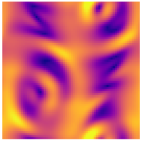
$x \in \mathbb{R}^n$

$$\frac{dx}{dt} = f(x)$$

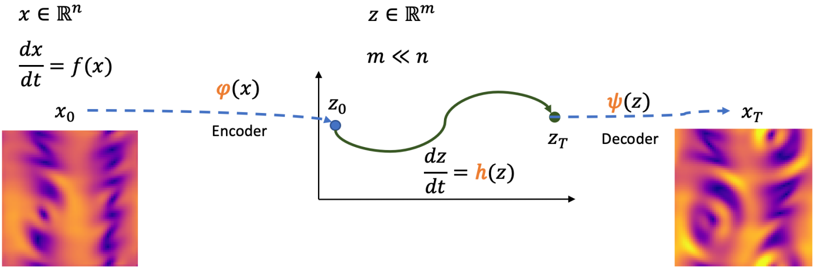
$$x_T = x_0 + \int_0^T f(x)dt$$

$$x_0 \longrightarrow x_T$$

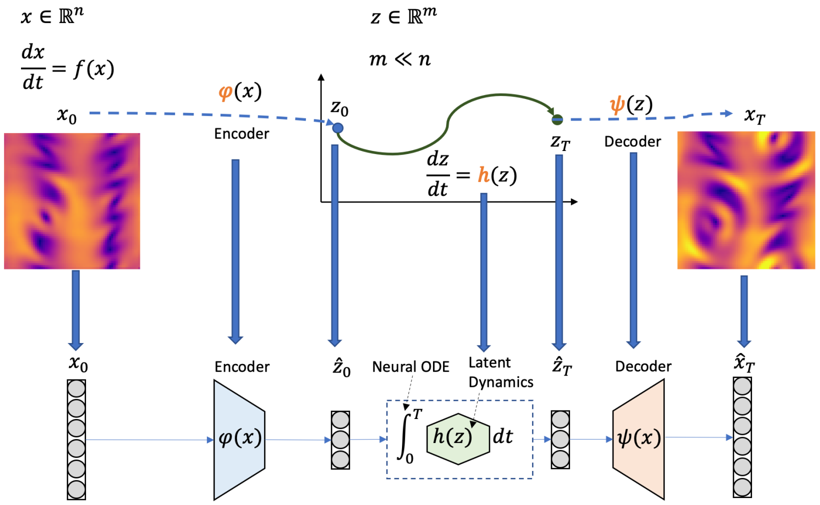




Reduced-Order Models (ROMs)



Reduced-Order Models (ROMs)



Physics-Informed Loss

Using chain rule:

$$\frac{dz}{dt} = \frac{dz}{dx}\frac{dx}{dt} = \nabla\varphi(x)^T f(x)$$

Physics-Informed Loss

Using chain rule:

$$\frac{dz}{dt} = \frac{dz}{dx}\frac{dx}{dt} = \nabla\varphi(x)^T f(x) \qquad \frac{dz}{dt} = h(\varphi(x))$$

Physics-Informed Loss

Using chain rule:

$$\frac{dz}{dt} = \frac{dz}{dx}\frac{dx}{dt} = \nabla\varphi(x)^T f(x) \qquad \frac{dz}{dt} = h(\varphi(x))$$

$$\mathcal{L}^{physics}(\tilde{x}) = \|\nabla\varphi(\tilde{x})^T f(\tilde{x}) - h(\varphi(\tilde{x}))\|_2^2 + \|\tilde{x} - \psi(\varphi(\tilde{x}))\|_2^2$$

Physics-Informed Loss = Latent Gradient Loss + Collocation Reconstruction Loss

Physics-Informed Loss

Using chain rule:

$$\frac{dz}{dt} = \frac{dz}{dx}\frac{dx}{dt} = \nabla\varphi(x)^T f(x) \qquad \frac{dz}{dt} = h(\varphi(x))$$

1 2 3

$$\mathcal{L}^{physics}(\tilde{x}) = \|\nabla\varphi(\tilde{x})^T f(\tilde{x}) - h(\varphi(\tilde{x}))\|_2^2 + \|\tilde{x} - \psi(\varphi(\tilde{x}))\|_2^2$$

Physics-Informed Loss = Latent Gradient Loss + Collocation Reconstruction Loss

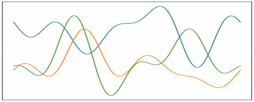
1

$$-u_t = u_{xx} + u_{xxxx} + \frac{1}{2}u_x^2 \Rightarrow \dot{x} = f(x)$$

$$u(x) = \frac{a}{1 + e^{-k(x-x_0)}} - \frac{a}{1 + e^{-k(x-x_1)}}, \quad x_0 < x_1$$

$$u(x) = \sum_{w=1}^{30} a(w)\sin(2\pi x) + b(w)\cos(2\pi x)$$

$$u(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_0)^2}{2\sigma^2}}$$



Physics-Informed Loss

Using chain rule:

$$\frac{dz}{dt} = \frac{dz}{dx}\frac{dx}{dt} = \nabla\varphi(x)^T f(x) \qquad \frac{dz}{dt} = h(\varphi(x))$$

1 2 3

$$\mathcal{L}^{physics}(\tilde{x}) = \|\nabla\varphi(\tilde{x})^T f(\tilde{x}) - h(\varphi(\tilde{x}))\|_2^2 + \|\tilde{x} - \psi(\varphi(\tilde{x}))\|_2^2$$

Physics-Informed Loss = Latent Gradient Loss + Collocation Reconstruction Loss

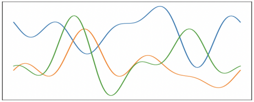
1

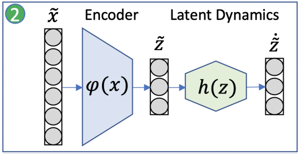
$$-u_t = u_{xx} + u_{xxxx} + \frac{1}{2}u_x^2 \Rightarrow \dot{x} = f(x)$$

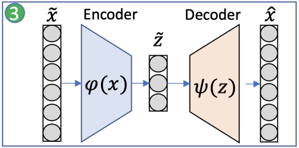
$$u(x) = \frac{a}{1 + e^{-k(x-x_0)}} - \frac{a}{1 + e^{-k(x-x_1)}}, \quad x_0 < x_1$$

$$u(x) = \sum_{w=1}^{30} a(w)\sin(2\pi x) + b(w)\cos(2\pi x)$$

$$u(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-x_0)^2}{2\sigma^2}}$$







Physics-Informed, Data-Driven, and Hybrid Models

Results: Extrapolation to Unknown Regions

Duffing Oscillator on a low-dimensional (2D) manifold:

$$\begin{aligned} \frac{dz_1}{dt} &= z_2 \\ \frac{dz_2}{dt} &= z_1 - z_1^3 \end{aligned} \tag{10}$$

Projection to a high-dimensional (128) space:

$$\boldsymbol{x} := \mathcal{A}(\boldsymbol{z}) = A\boldsymbol{z}^3, \quad A \in \mathbb{R}^{128\times 2}, \quad A_{ij} \sim_{i.i.d.} \mathcal{N}(0,1) \tag{11}$$

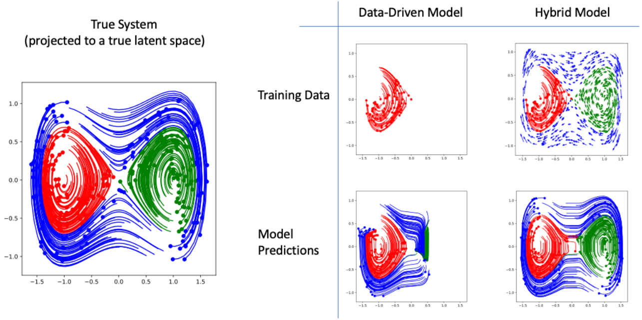
Results: Extrapolation to Unknown Regions

Duffing Oscillator on a low-dimensional (2D) manifold:

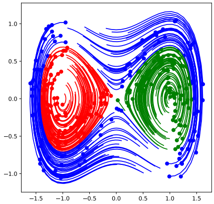
$$\begin{aligned}\frac{dz_1}{dt} &= z_2 \\ \frac{dz_2}{dt} &= z_1 - z_1^3\end{aligned} \tag{10}$$

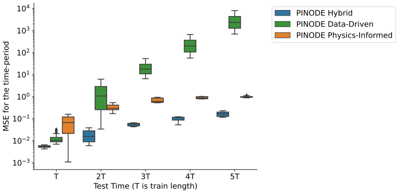
Projection to a high-dimensional (128) space:

$$\boldsymbol{x} := \mathcal{A}(\boldsymbol{z}) = A\boldsymbol{z}^3, \quad A \in \mathbb{R}^{128\times 2}, \quad A_{ij} \sim_{i.i.d.} \mathcal{N}(0,1) \tag{11}$$

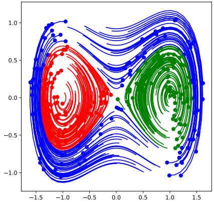


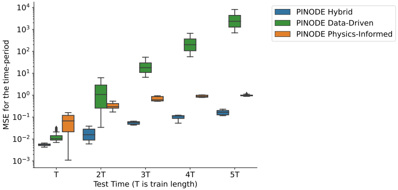
Results: Stable Long-Term Predictions

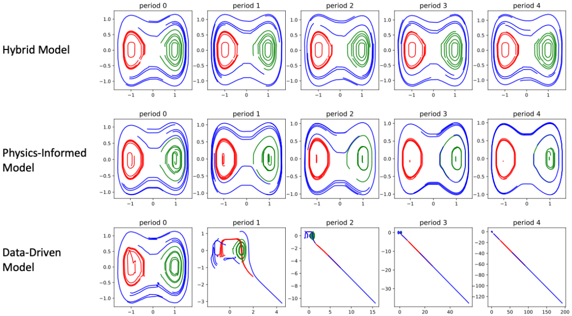




Results: Stable Long-Term Predictions







Results: Burgers' Equation

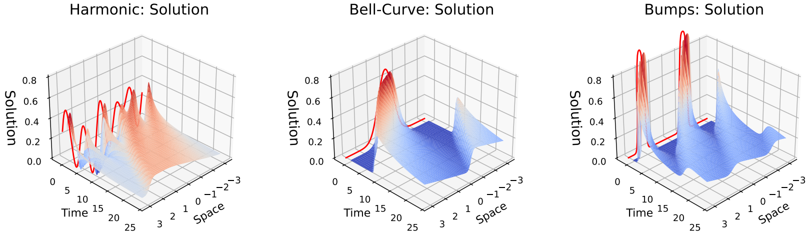
Burgers' Equation:

$$\begin{aligned} u_t + uu_x &= \nu u_{xx} \\ u(-\pi, t) &= u(\pi, t), \quad \forall t \in [0, T] \end{aligned} \tag{12}$$

Results: Burgers' Equation

Burgers' Equation:

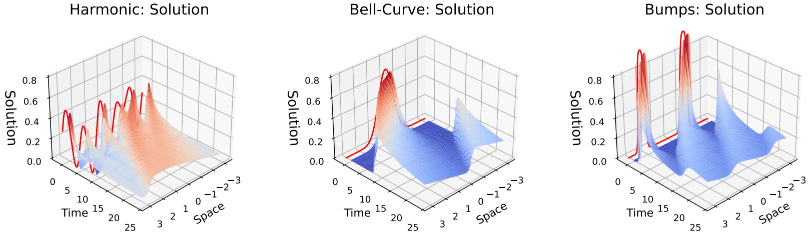
$$
\begin{aligned}
u_t + uu_x &= \nu u_{xx} \\
u(-\pi, t) &= u(\pi, t), \quad \forall t \in [0, T]
\end{aligned}
\tag{12}
$$

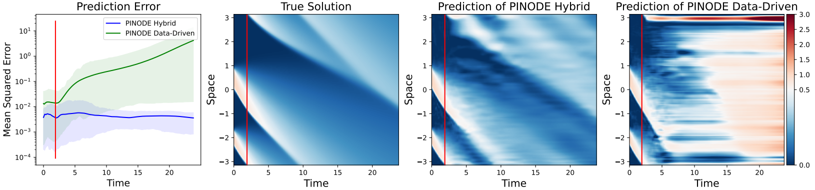


Results: Burgers' Equation

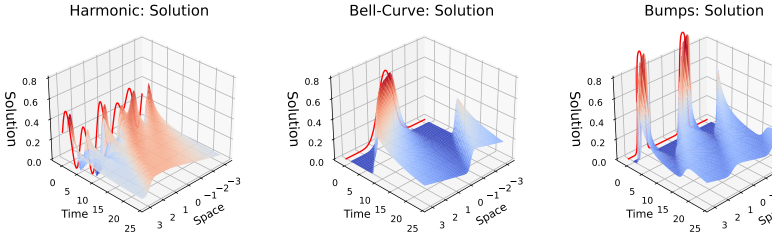
Burgers' Equation:

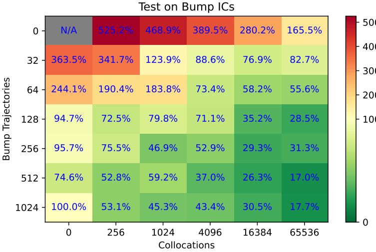
$$
\begin{aligned}
u_t + u u_x &= \nu u_{xx} \\
u(-\pi, t) &= u(\pi, t), \quad \forall t \in [0, T]
\end{aligned}
\tag{12}
$$

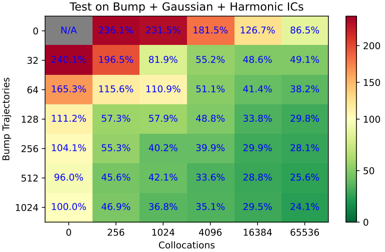




Results: Learning From Collocations







Discussion and Limitations

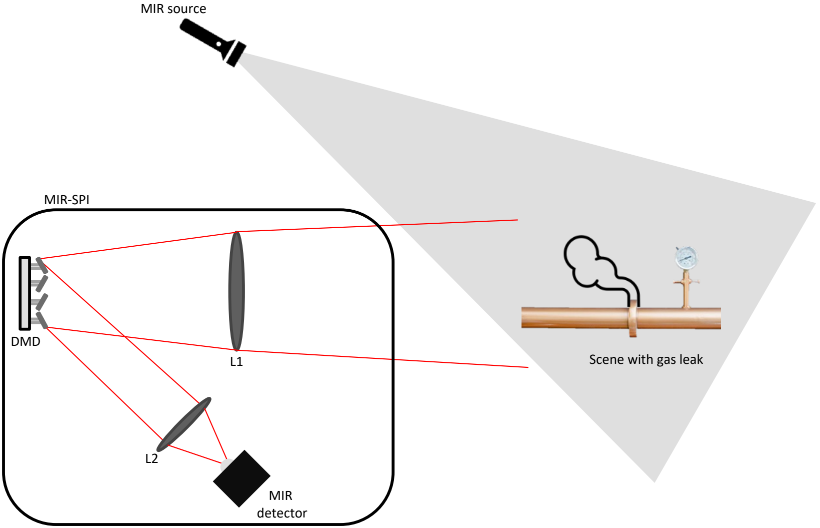
We showed that:

- Physics-informed loss improves accuracy and forecasting stability of ROMs
- Collocations can supplement data to improve model's performance for unseen initial conditions.

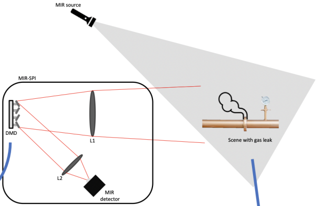
Limitations:

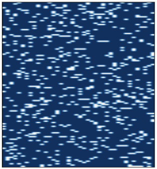
- Optimal choice of collocations is problem-specific
- Need a lot of collocations
  - *Seems* possible to overcome with smarter sampling techniques

Single-Pixel Imaging



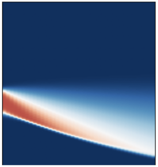
Single-Pixel Imaging





Compressive Sensing matrix **A**

$*$



Complete Observation **x**

Sum

Collected Information

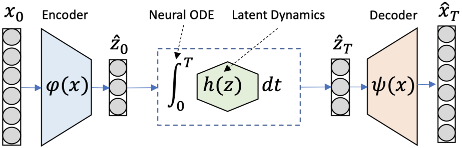
$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

Number of timesteps

Number of samples per frame

Compressive Sensing with Reduced-Order Models

Offline Step: Train a Data-Driven Reduced-Order Model



Online Step: Reconstruct Complete Observations by Optimizing in Latent Space

Reconstruction Loss

Compressive Sensing Loss

Loss for Prediction in Latent space

$$\mathcal{L}^{recon.}(z) = ||y - A\psi(z)|| + \lambda \left\| z - \left(z_0 + \int_0^T h(z)dz\right) \right\|$$

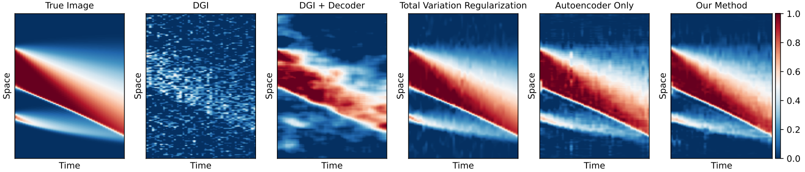
Latent-space representation of the trajectory

"What the data tells us the trajectory should be"

"What the model thinks the trajectory should be"

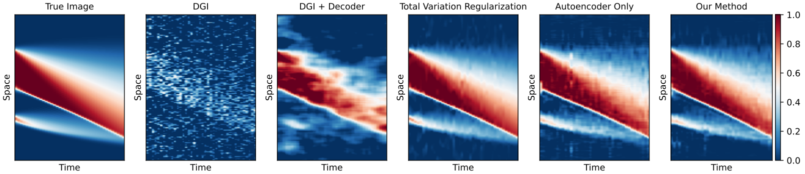
Results: Burger's Equation

When we capture 32 samples per frame:

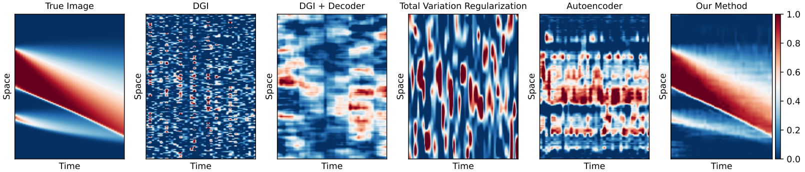


Results: Burger's Equation

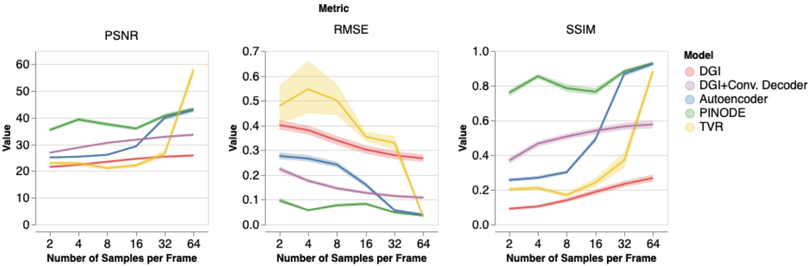
When we capture 32 samples per frame:



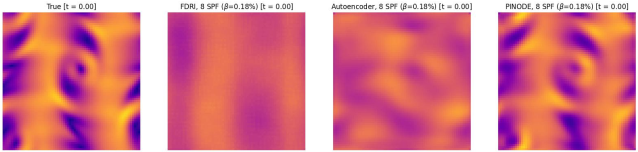
When we capture 2 samples per frame:



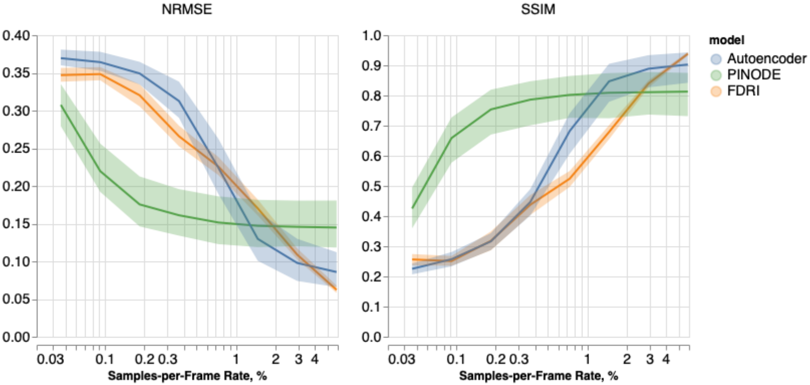
Results: Burger's Equation



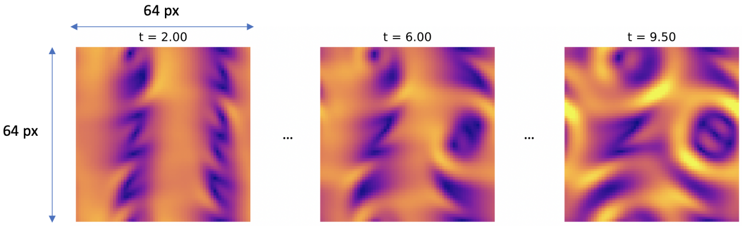
Results: Kolmogorov Flow



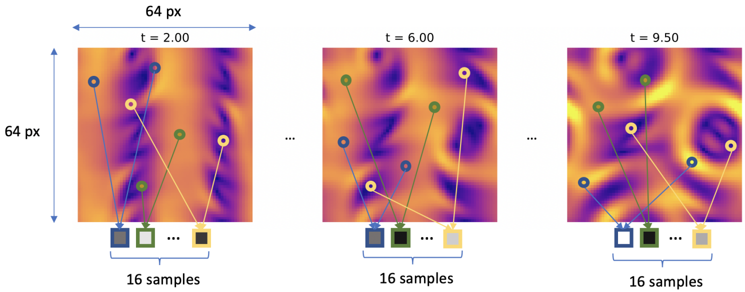
Results: Kolmogorov Flow



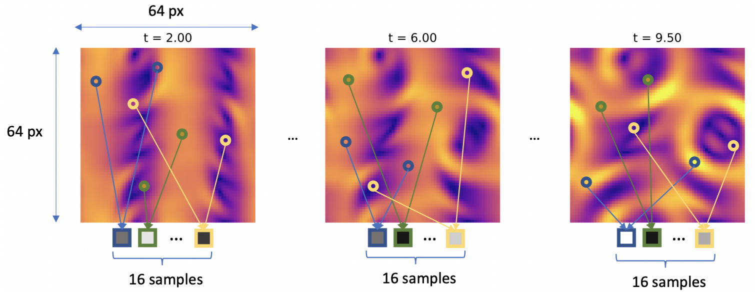
Results: Interpretation

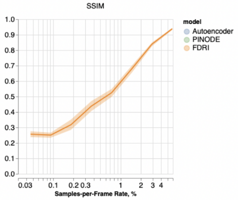


Results: Interpretation

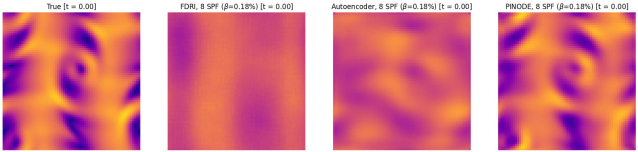


Results: Interpretation

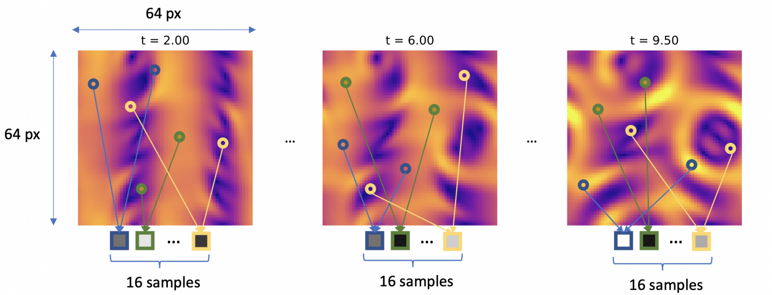


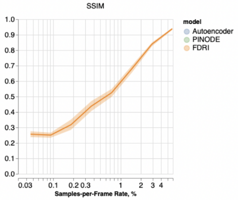


Results: Kolmogorov Flow

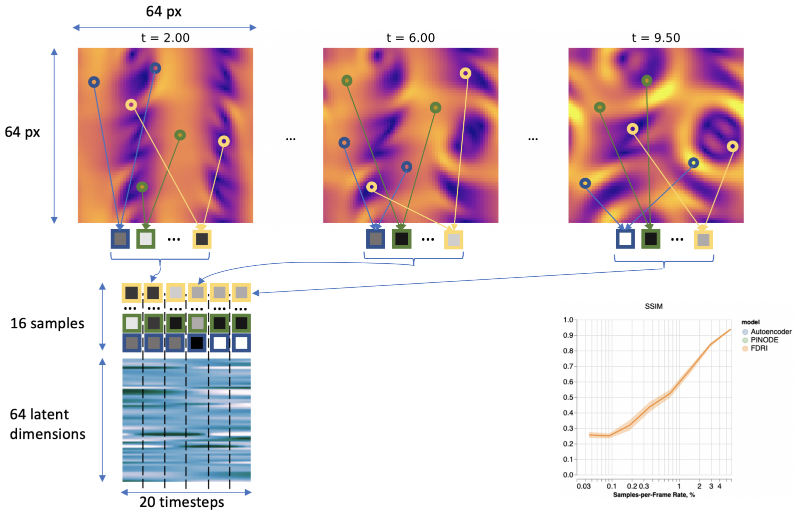


Results: Interpretation

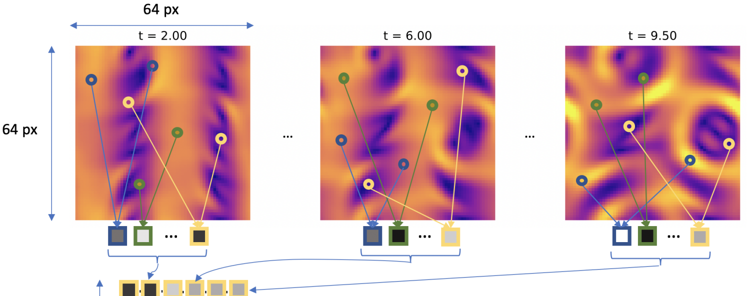


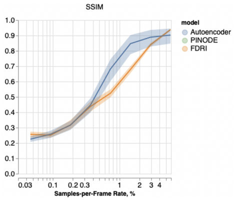


Results: Interpretation

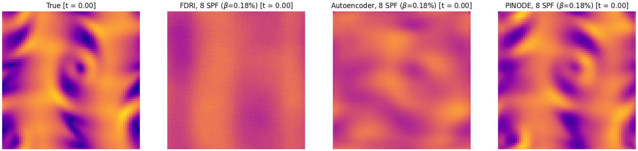


Results: Interpretation

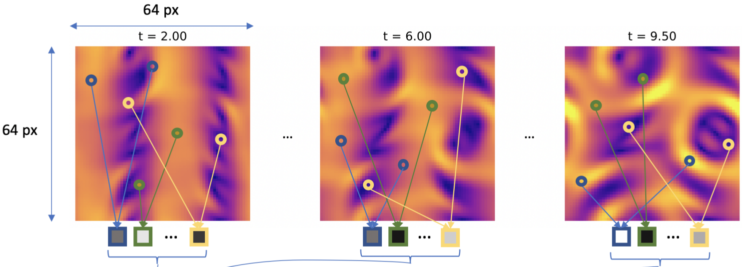


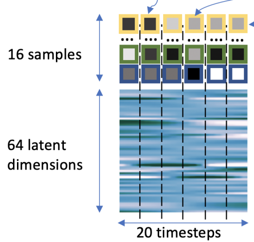


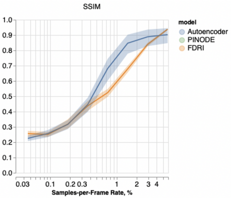
Results: Kolmogorov Flow



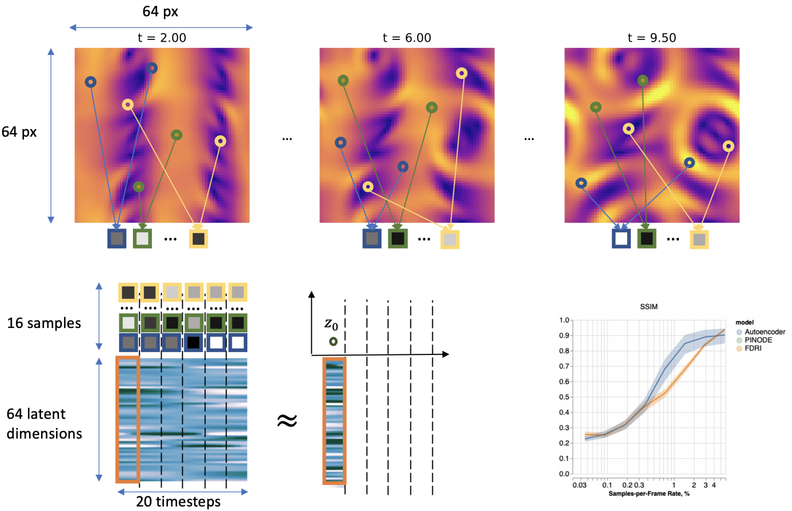
Results: Interpretation



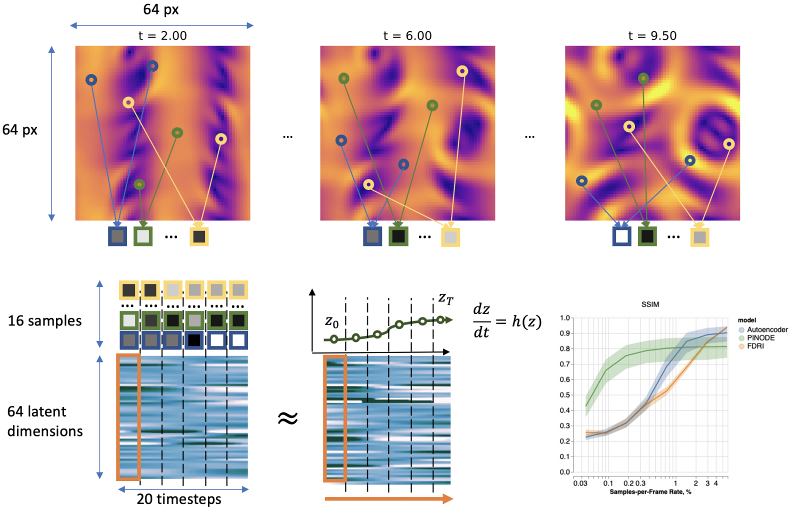




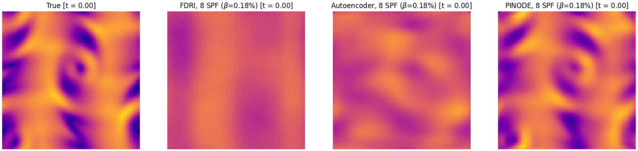
Results: Interpretation



Results: Interpretation



Results: Kolmogorov Flow



Results: Gas Monitoring

Conclusion and Acknowledgments

Designing an Algorithm

$G_{\nu,\eta}$ encodes both gradient of a Lagrangian (lines 1-2) and the complementarity condition (line 3):

$$G_{\nu,\eta}((\beta,\gamma,v),(\tilde{\beta},\tilde{\gamma})) := \begin{bmatrix} \nabla_\beta \mathcal{L}(\beta,\gamma) + \eta(\beta - \tilde{\beta}) \\ \nabla_\gamma \mathcal{L}(\beta,\gamma) + \eta(\gamma - \tilde{\gamma}) - v \\ v \bigodot \gamma - \mu \mathbf{1} \end{bmatrix} \tag{13}$$

We apply Newton method to G while geometrically decreasing μ.
Lemma: For every $(\mu,\eta) \in \mathbb{R}_+ \times \mathbb{R}_{++}$,

$$\begin{aligned} &(\hat{\beta},\hat{\gamma}) = \operatorname*{argmin}_{(\beta,\gamma)} \mathcal{L}_{\eta,\mu}((\beta,\gamma),(\tilde{\beta},\tilde{\gamma})) \\ &\iff \\ &\exists \hat{v} \in \mathbb{R}^q_+ \text{ s.t. } G_{\nu,\eta}((\beta,\gamma,\hat{v}),(\tilde{\beta},\tilde{\gamma})) = 0 \end{aligned} \tag{14}$$

If $\mu > 0$, then $\hat{v} = -\nabla\phi_\mu(\hat{\gamma})$, and if $\mu = 0$, then $\hat{v}$ is the unique KKT multiplier associated with the constraint $0 \leq \gamma$.

$\mathcal{MSR}3$-fast Algorithm

progress $\leftarrow$ **True**; iter = 0;

$\beta^+, \tilde\beta^+ \leftarrow \beta_0$; $\gamma^+, \tilde\gamma^+ \leftarrow \gamma_0$; $v^+ \leftarrow 1 \in \mathbb{R}^q$; $\mu \leftarrow \frac{{v^+}^T \gamma^+}{10q}$

while iter $<$ max_iter and $\|G_\mu(\beta^+, \gamma^+, v^+)\| >$ tol and progress **do**

- $\beta \leftarrow \beta^+$; $\gamma \leftarrow \gamma^+$; $\tilde\beta \leftarrow \tilde\beta^+$; $\tilde\gamma \leftarrow \tilde\gamma^+$
- $[dv, d\beta, d\gamma] \leftarrow \nabla G_\mu((\beta, \gamma, v), (\tilde\beta, \tilde\gamma))^{-1} G_\mu((\beta, \gamma, v), (\tilde\beta, \tilde\gamma))$
  $\alpha \leftarrow 0.99 \times \min\left(1, -\frac{\gamma_i}{d\gamma_i}, \forall i : d\gamma_i < 0\right)$
- $\beta^+ \leftarrow \beta + \alpha d\beta$; $\gamma^+ = \gamma + \alpha d\gamma$; $v^+ \leftarrow v + \alpha dv$
- **if** $\|\gamma^+ \odot v^+ - q^{-1}{\gamma^+}^T v^+ \mathbf{1}\| > 0.5 q^{-1} {v^+}^T \gamma^+$ **then** continue;
- **else**
  - $\tilde\beta^+ = \mathrm{prox}_{\alpha R}(\beta^+)$; $\tilde\gamma^+ = \mathrm{prox}_{\alpha R + \delta_{\mathbb{R}_+}}(\gamma^+)$; $\mu = \frac{1}{10}\frac{{v^+}^T\gamma^+}{q}$
- **end**
- progress $= (\|\beta^+ - \beta\| \geq$ tol or $\|\gamma^+ - \gamma\| \geq$ tol or $\|\tilde\beta^+ - \tilde\beta\| \geq$ tol or $\|\tilde\gamma^+ - \tilde\gamma\| \geq$ tol$)$
- iter += 1

end

return $\tilde\beta^+, \tilde\gamma^+$

References I

Aravkin, Aleksandr et al. Analysis of Relaxation Methods for Feature Selection in Mixed Effects Models. 2022. arXiv: 2209.10575 [stat.ME].

Beck, Amir. First-Order Methods in Optimization. MOS-SIAM Series on Optimization. SIAM, 2017. ISBN: 9781611974980. DOI: 10.1137/1.9781611974997.

Burke, J.V. and A. Engle. "Line Search and Trust-Region Methods for Convex-Composite Optimization". In: arXiv:1806.05218 (2018). DOI: https://doi.org/10.48550/arXiv.1806.05218.

Champion, Kathleen et al. "Data-driven discovery of coordinates and governing equations". In: Proceedings of the National Academy of Sciences 116.45 (2019), pp. 22445–22451.

Chidester, Benjamin, Minh N. Do, and Jian Ma. Rotation Equivariance and Invariance in Convolutional Neural Networks. 2018. arXiv: 1805.12301 [stat.ML].

Finzi, Marc et al. "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data". In: 37th International Conference on Machine Learning, ICML 2020 PartF16814 (2020), pp. 3146–3157. arXiv: 2002.12880.

Geiger, Mario and Tess Smidt. "e3nn: Euclidean neural networks". In: arXiv preprint arXiv:2207.09453 (2022).

Groll, Andreas and Gerhard Tutz. "Variable selection for generalized linear mixed models by L 1-penalized estimation". In: Statistics and Computing 24.2 (2014), pp. 137–154.

References II

Rackauckas, Christopher et al. "Universal differential equations for scientific machine learning". In: arXiv preprint arXiv:2001.04385 (2020).

Raissi, Maziar, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations". In: arXiv preprint arXiv:1711.10561 (2017).

Schelldorfer, Jürg, Peter Bühlmann, and SARA VAN DE GEER. "Estimation for high-dimensional linear mixed-effects models using l1-penalization". In: Scandinavian Journal of Statistics 38.2 (2011), pp. 197–214.

Schmidt, Michael and Hod Lipson. "Distilling free-form natural laws from experimental data". In: science 324.5923 (2009), pp. 81–85.

Sholokhov, Aleksei, James V. Burke, et al. A Relaxation Approach to Feature Selection for Linear Mixed Effects Models. 2022. arXiv: 2205.06925 [stat.ME].

Sholokhov, Aleksei, Peng Zheng, and Aleksandr Aravkin. "pysr3: A Python Package for Sparse Relaxed Regularized Regression". In: Journal of Open Source Software 8.84 (2023), p. 5155.

Zheng, Peng and Aleksandr Aravkin. "Relax-and-split method for nonconvex inverse problems". In: Inverse Problems 36.9 (2020). ISSN: 13616420. DOI: 10.1088/1361-6420/aba417.