



Scientific Machine Learning for Modeling, Optimization, and Control

Ján Drgoňa

Associate Professor

Civil and Systems Engineering Department

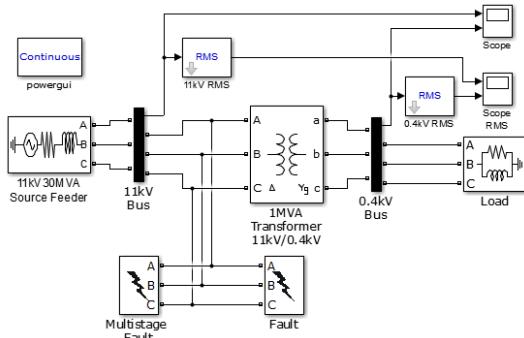
The Ralph O'Connor Sustainable Energy Institute

Optimizing Complex Energy Systems is Hard

- **Simulations** are crucial for optimal decision-making in complex energy systems
- **Need:** Improve computational efficiency and scalability of digital twins and optimization-based decision-making
- **Challenges:**
 1. Modeling and simulation of complex systems is hard
 2. Closed-loop decision-making for complex systems is hard-er
 3. Scientific computing and machine learning tools are fragmented and not easily composable



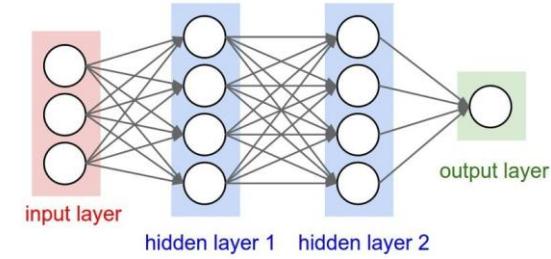
Challenge 1: Heterogenous Modeling Methods



White-box models



Gray-box models



Black-box models

Physics-based

More domain
knowledge

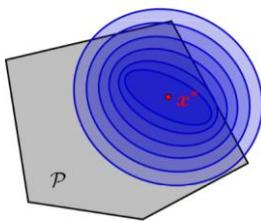
Data-driven

Less domain
knowledge

Challenge 2: Heterogenous Solution Methods

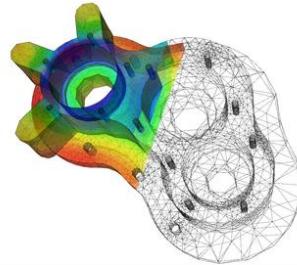
Constrained Optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & b(x) \geq 0 \\ & c(x) = 0. \end{aligned}$$



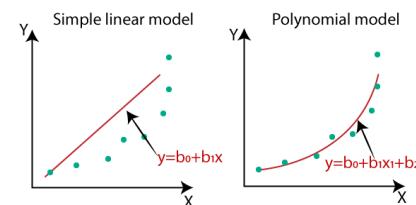
Differential Equations

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



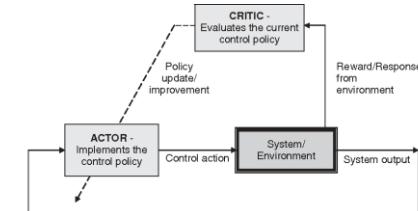
Supervised Learning

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N L(y^i, f(x^i, \theta))$$



Reinforcement Learning

$$\begin{aligned} \min_{\Theta} \sum_{i=1}^m r(x, \Theta) \\ \text{s.t. } \text{Bellman}(x, \Theta) = \mathbf{0}, \\ \text{environment}(x, \Theta) = \mathbf{0} \\ x \in \Xi \end{aligned}$$



- Requires prior knowledge of objective function and constraints

- Requires prior knowledge of the physics to be modeled

- Requires large labeled datasets

- Requires environment model to sample

More domain knowledge

Less domain knowledge

Challenge 3: Heterogenous Solution Tools

Constrained Optimization



PYOMO



JUMP



CVXPY



Differential Equations



MATLAB®



PETSc



TAO



SciPy



More domain
knowledge

Supervised Learning



flux



Less domain
knowledge



Gym

Scientific Machine Learning (SciML)

What?

- SciML systematically integrates ML methods with mathematical models and algorithms developed in various scientific and engineering domains

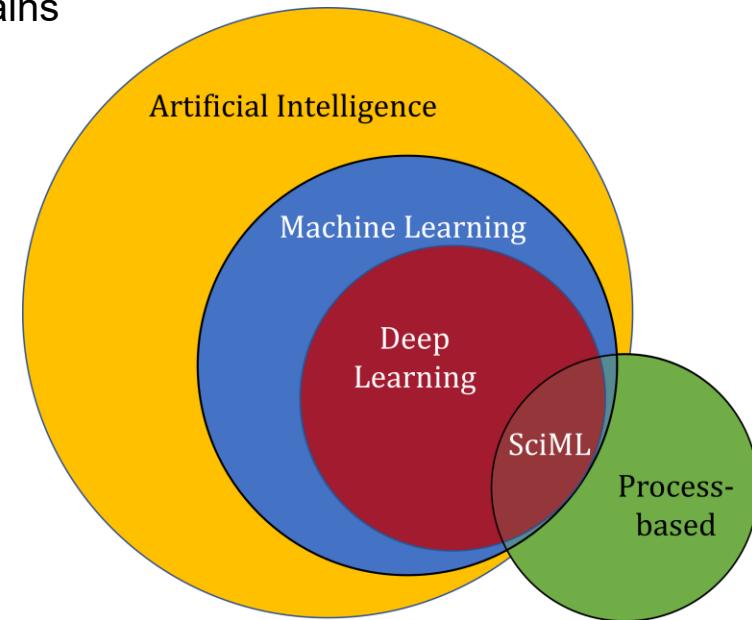
Why?

- Scientific applications are governed by fundamental principles and physical constraints
- Purely data-driven “black box” ML methods cannot satisfy underlying physics

How?

- Leverage **automatic differentiation** used in learning for modeling, optimization, and control

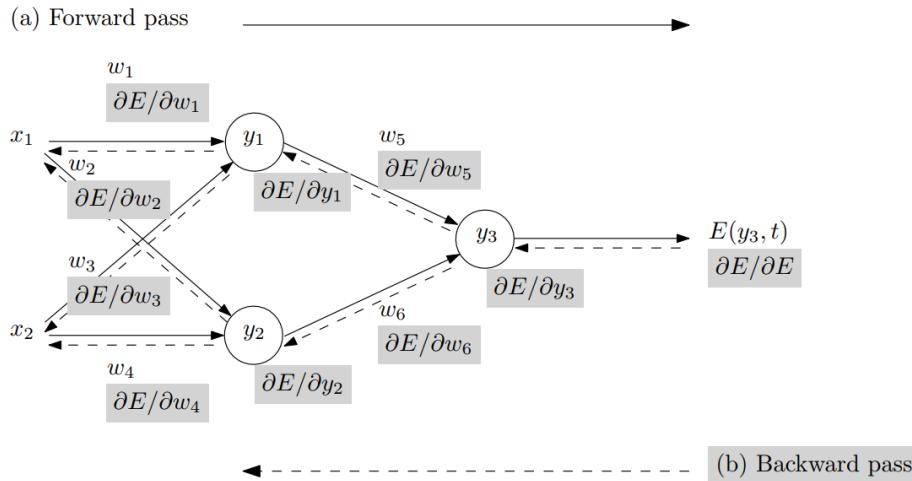
Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 422–440, 2021.



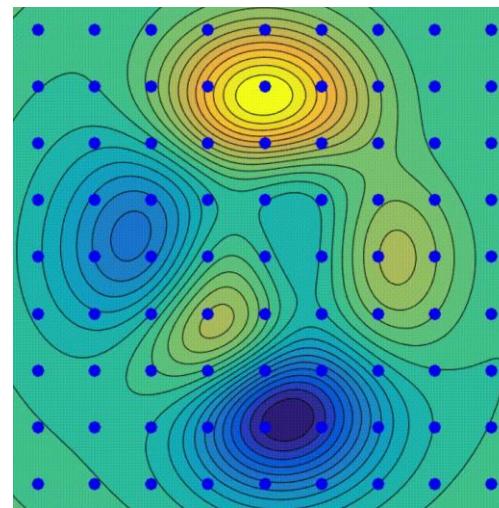
Automatic Differentiation (AD) in Machine Learning

AD enables efficient and accurate gradient computation, which is fundamental for training complex ML models using GPUs.

Backpropagation Algorithm



Gradient Descent Algorithm



Animation source: wikipedia

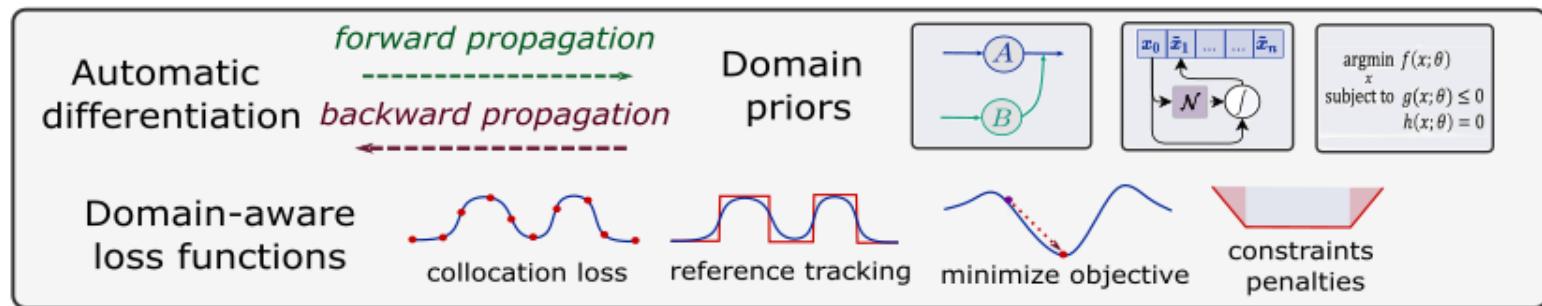
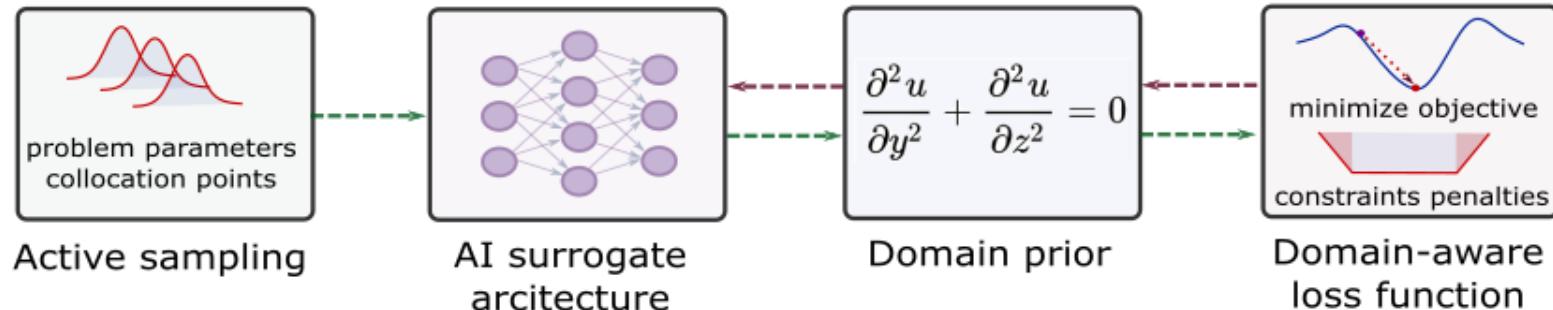
SW and HW
Innovations



Selected Scientific Machine Learning Literature

- **Differentiable Programming**
 - M. Innes, et al., *A Differentiable Programming System to Bridge Machine Learning and Scientific Computing*, 2019
- **Learning to Solve (L2S)**
 - M. Raissi, et al., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, 2019
- **Learning to Optimize (L2O)**
 - A. Agrawal, et al., *Differentiable Convex Optimization Layers*, 2019
 - P. Donti, et al., *DC3: A learning method for optimization with hard constraints*, 2021
 - J. Kotary, et al., *End-to-End Constrained Optimization Learning: A Survey*, 2021
- **Learning to Model (L2M)**
 - B. Lusch, et al., *Deep learning for universal linear embeddings of nonlinear dynamics*, 2018
 - R. T. Q. Chen, et al., *Neural Ordinary Differential Equations*, 2019
 - C. Rackauckas, et al., *Universal Differential Equations for Scientific Machine Learning*, 2021
- **Learning to Control (L2C)**
 - B. Amos, et al., *Differentiable MPC for End-to-end Planning and Control*, 2019
 - S. East, et al., *Infinite-Horizon Differentiable Model Predictive Control*, 2020
 - Y Qiao, et al., *Scalable Differentiable Physics for Learning and Control*, 2020

Components of Scientific Machine Learning

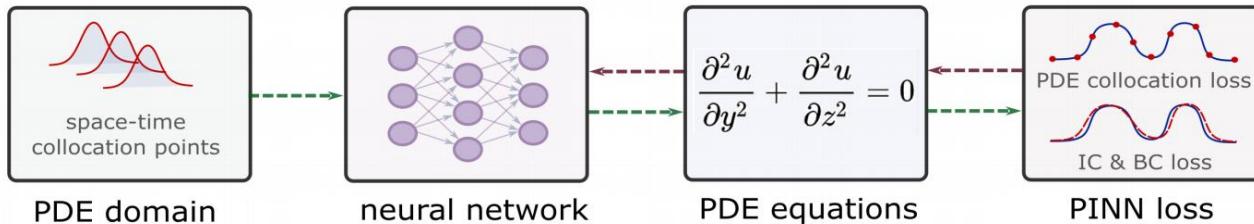


Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 2021.

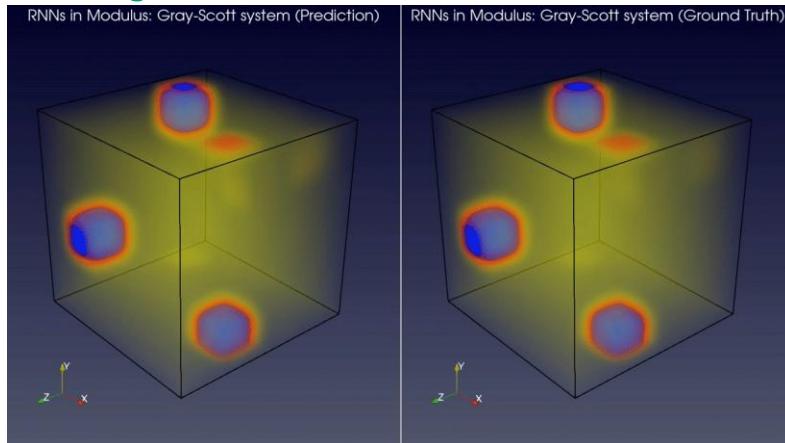
Thiyagalingam, J., Shankar, M., Fox, G. et al. Scientific machine learning benchmarks. Nature Reviews Physics 4, 413–420, 2022.

Nghiem T., Drgona J., et al. Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems, ACC, 2023.

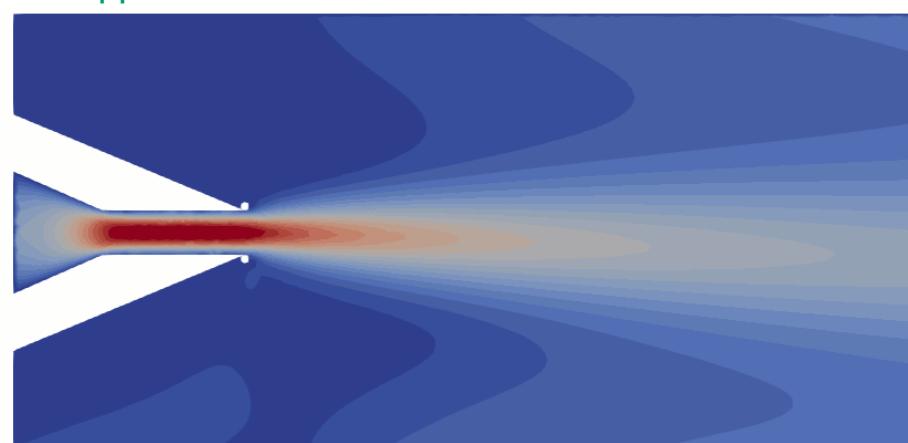
Learning to Solve Differential Equations with Physics-Informed Neural Networks (PINNs)



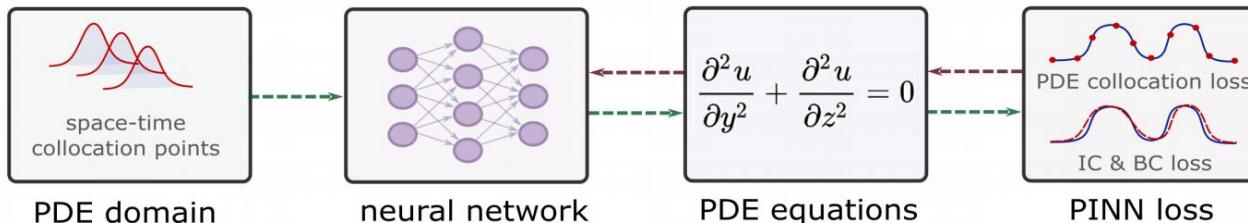
Training neural networks as PDE solutions



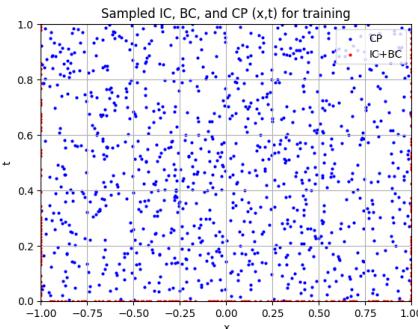
Application: Parameter estimation from data



Learning to Solve Differential Equations with Physics-Informed Neural Networks (PINNs)



Dataset: collocation points in the spatio-temporal



Architecture: PDE equations solved with **neural network** via automatic differentiation.

$$\hat{y} = NN_{\theta}(x, t)$$

$$f_{\text{PINN}}(t, x) = \left(\frac{\partial NN_{\theta}}{\partial t} - \frac{\partial^2 NN_{\theta}}{\partial x^2} \right) + e^{-t} (\sin(\pi x) - \pi^2 \sin(\pi x))$$

Loss function: minimizing PDE equation, initial and boundary condition residuals.

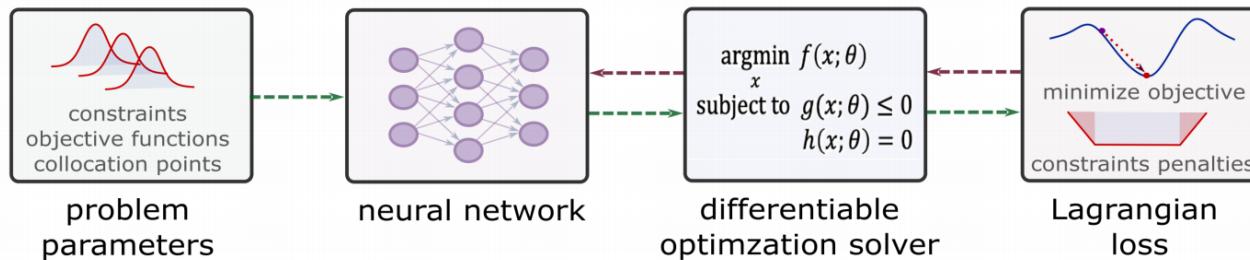
$$\ell_f = \frac{1}{N_f} \sum_{i=1}^{N_f} |f_{\text{PINN}}(t_f^i, x_f^i)|^2$$

$$\ell_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |y(t_u^i, x_u^i) - NN_{\theta}(t_u^i, x_u^i)|^2$$

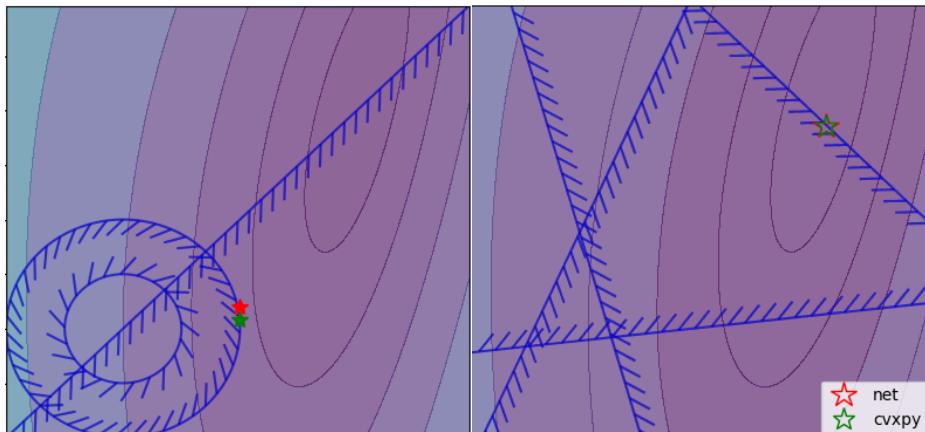
$$\ell_{\text{PINN}} = \ell_f + \ell_u$$

https://github.com/pnnl/neuromancer/blob/master/examples/PDEs/Part_2_PINN_BurgersEquation.ipynb

Learning to Optimize (L2O) with Constraints



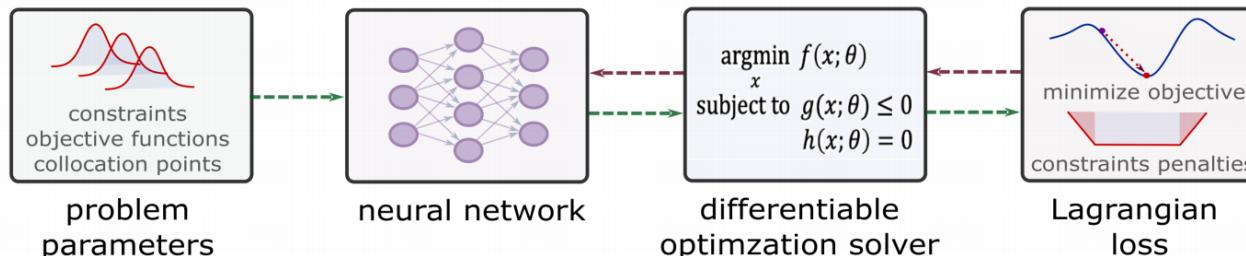
Training neural networks as optimization solutions



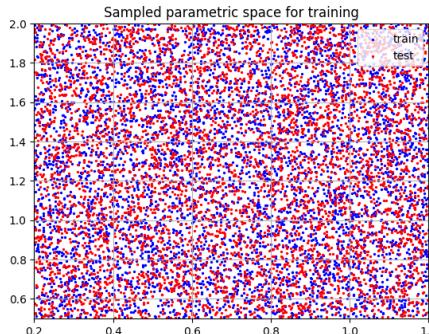
Application: solving optimal power flow



Learning to Optimize (L2O) with Constraints



Dataset: collocation points in the parametric space.



Architecture: **differentiable optimization solver** with **neural network** surrogate.

$$\begin{aligned} & \text{minimize}_{\theta} && f(x, \xi) \\ & \text{subject to} && g(x, \xi) \leq 0 \\ & && x = NN_{\theta}(\xi) \\ & && \hat{x} = \operatorname{proj}_{g(x, \xi) \leq 0}(x, \xi) \end{aligned}$$

Loss function: minimizing objective function and constraints penalties.

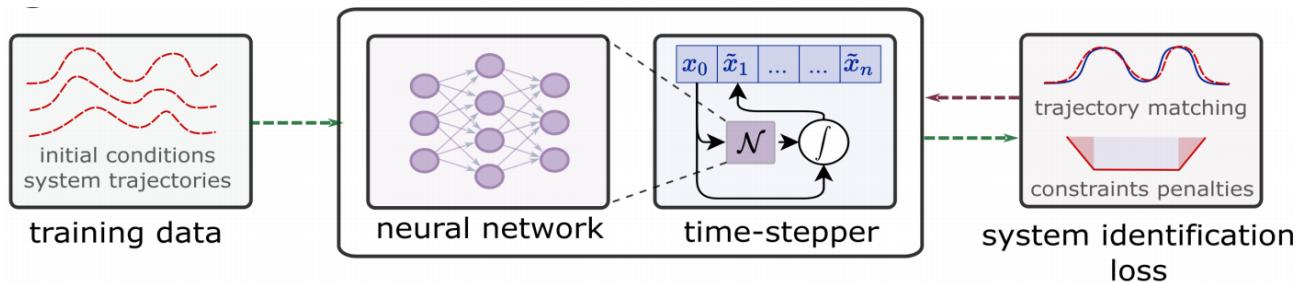
$$\ell_f = \frac{1}{m} \sum_{i=1}^m |f(x^i, \xi^i)|^2$$

$$\ell_g = \frac{1}{m} \sum_{i=1}^m |\operatorname{RELU}(g(x^i, \xi^i))|^2$$

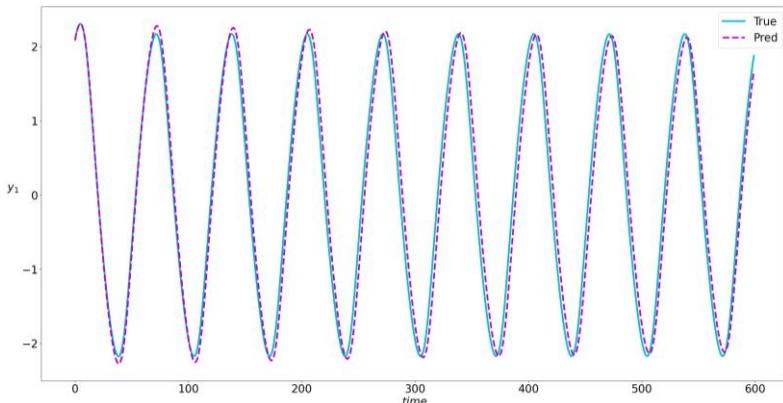
$$\ell_{L2O} = \ell_f + \ell_g$$

https://github.com/pnnl/neuromancer/blob/master/examples/parametric_programming/Part_1_basics.ipynb

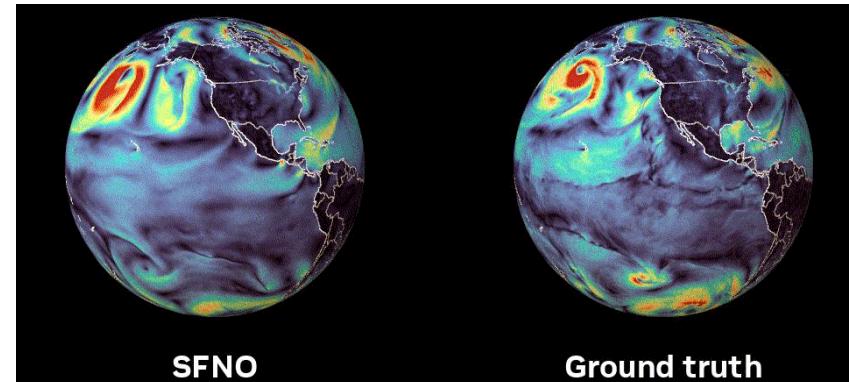
Learning to Model (L2M) Dynamical Systems



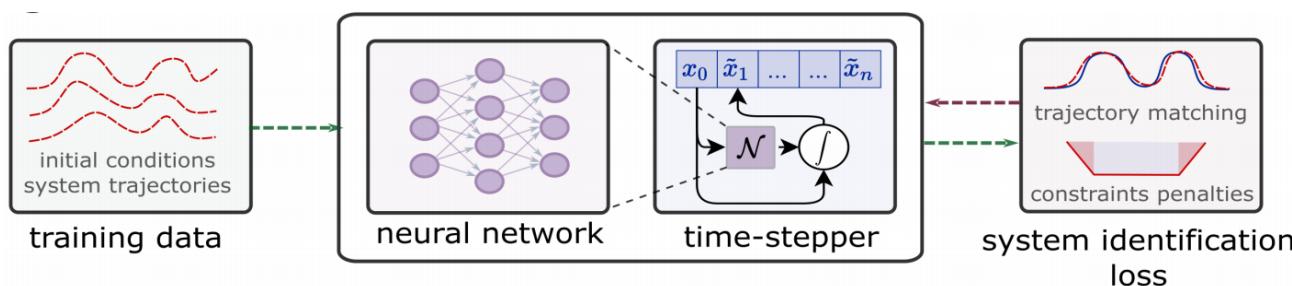
Neural models for nonlinear system identification



Application: climate modeling

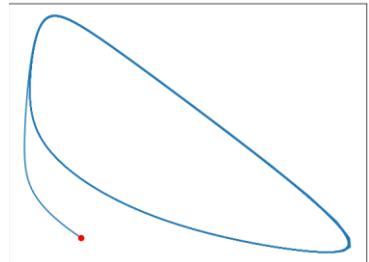


Learning to Model (L2M) Dynamical Systems



Dataset: time-series of states, inputs, and disturbances tuples.

$$\hat{X} = [\hat{x}_0^i, \dots, \hat{x}_N^i], i \in [1, \dots, m]$$



Architecture: **differentiable ODE solver** with **neural network** model.

$$x_{k+1} = \text{ODESolve}(NN_\theta(x_k))$$

Architecture: **Koopman operator** with **neural network** basis

$$\begin{aligned} \text{funct } y_k &= NN_\theta(x_k) \\ y_{k+1} &= K_\theta(y_k) \\ x_{k+1} &= NN_\theta^{-1}(y_{k+1}) \end{aligned}$$

Loss function: trajectory matching, regularizations, and constraints penalties.

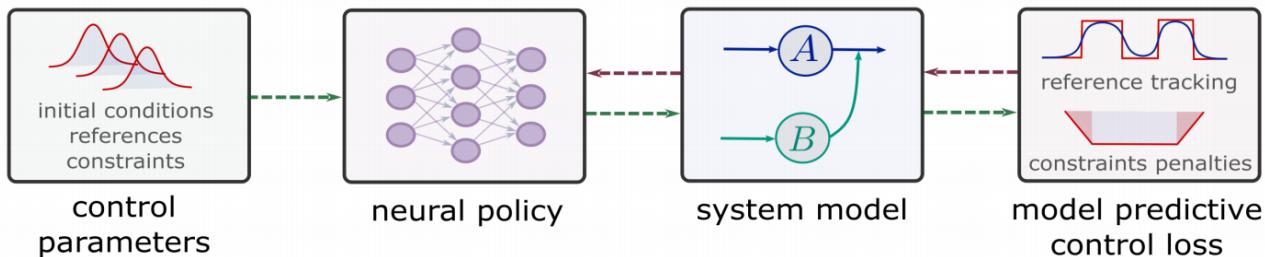
$$\ell_1 = \sum_{i=1}^m \sum_{k=1}^N Q_x \|x_k^i - \hat{x}_k^i\|_2^2$$

$$\ell_2 = \sum_{i=1}^m \sum_{k=1}^{N-1} Q_{dx} \|\Delta x_k^i - \Delta \hat{x}_k^i\|_2^2$$

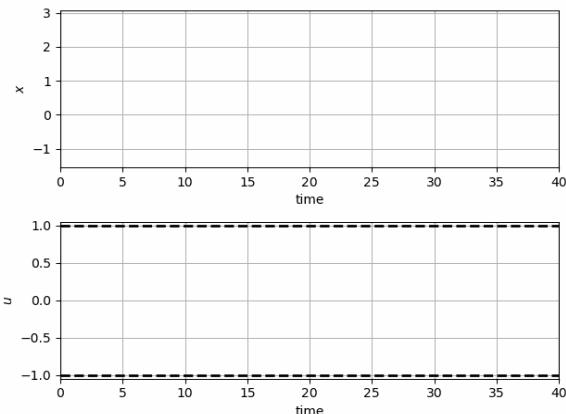
$$\ell_{L2M} = \ell_1 + \ell_2$$

https://github.com/pnnl/neuromancer/blob/master/examples/ODEs/Part_1_NODE.ipynb

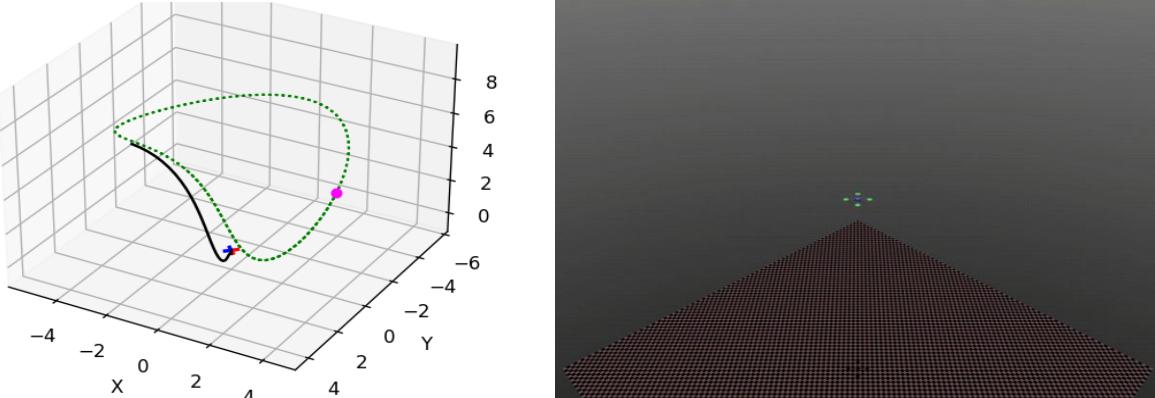
Learning to Control (L2C) via Differentiable Predictive Control (DPC)



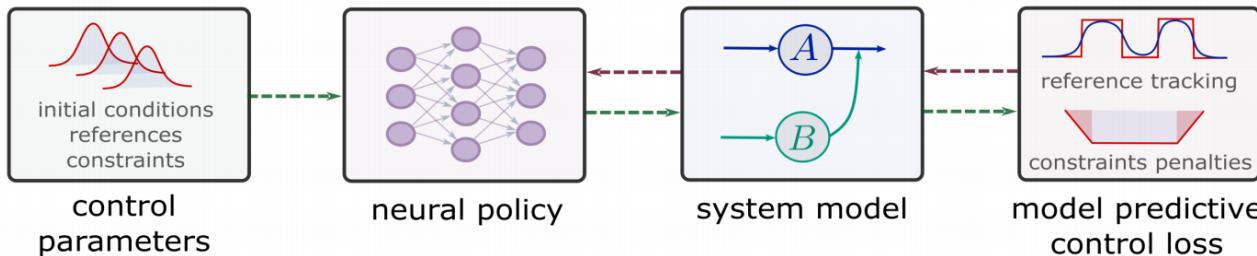
Trajectory optimization for dynamical systems



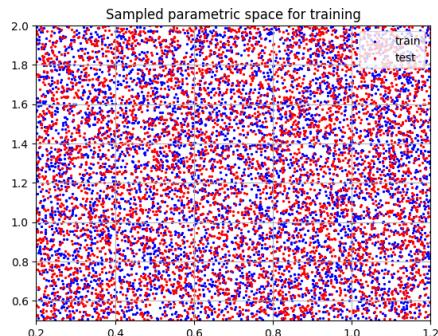
Application: autonomous systems



Learning to Control (L2C) via Differentiable Predictive Control (DPC)



Dataset: collocation points in the control parametric space.



Architecture: differentiable model with neural network

control policy.

$$x_{k+1} = \text{ODESolve}(f(x_k, u_k))$$

$$u_k = NN_\theta(x_k, \xi_k)$$

$$g(x_k, u_k, \xi_k) \leq 0$$

$$x_0 \sim \mathcal{P}_{x_0}$$

$$\xi_k \sim \mathcal{P}_\xi$$

Loss function: reference tracking, constraints and terminal penalties.

$$\ell_1 = \sum_{i=1}^m \sum_{k=1}^{N-1} Q_x ||x_k^i - r_k^i||_2^2$$

$$\ell_2 = \sum_{i=1}^m \sum_{k=1}^{N-1} Q_g ||\text{RELU}(g(x_k^i, u_k^i, \xi_k^i))||_2^2$$

$$\ell_{L2C} = \ell_1 + \ell_2$$

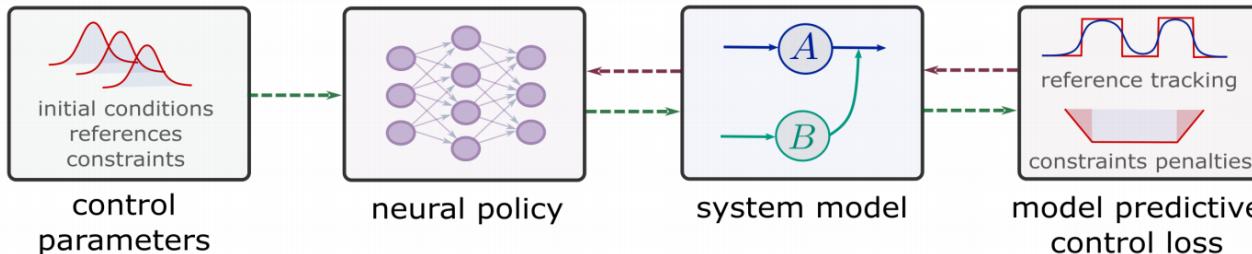
https://github.com/pnnl/neuromancer/blob/master/examples/control/Part_3_ref_tracking_ODE.ipynb

Jan Drgona, et al., *Differentiable Predictive Control: An MPC Alternative for Unknown Nonlinear Systems using Constrained Deep Learning*, Journal of Process Control, 2022

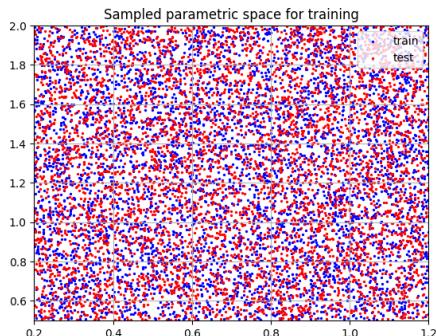
Learning to Control (L2C) via Differentiable Predictive Control (DPC)

Follow up

TuC04:
16:10-
16:35



Dataset: collocation points in the control parametric space.



Architecture: **differentiable model** with **neural network** control policy.

$$x_{k+1} = \text{ODESolve}(f(x_k, u_k))$$

$$u_k = NN_\theta(x_k, \xi_k)$$

$$g(x_k, u_k, \xi_k) \leq 0$$

$$x_0 \sim \mathcal{P}_{x_0}$$

$$\xi_k \sim \mathcal{P}_\xi$$

Loss function: reference tracking, constraints and terminal penalties.

$$\ell_1 = \sum_{i=1}^m \sum_{k=1}^{N-1} Q_x ||x_k^i - r_k^i||_2^2$$

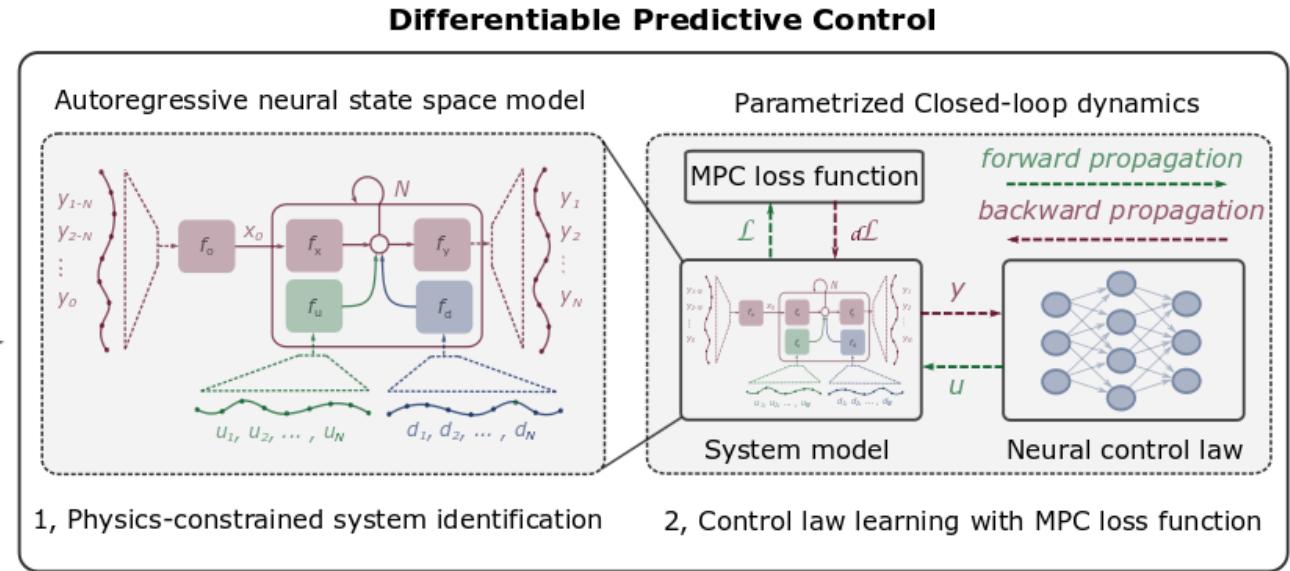
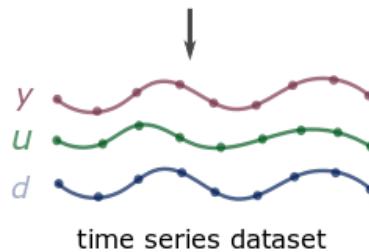
$$\ell_2 = \sum_{i=1}^m \sum_{k=1}^{N-1} Q_g ||\text{RELU}(g(x_k^i, u_k^i, \xi_k^i))||_2^2$$

$$\ell_{L2C} = \ell_1 + \ell_2$$

https://github.com/pnnl/neuromancer/blob/master/examples/control/Part_3_ref_tracking_ODE.ipynb

Jan Drgona, et al., *Differentiable Predictive Control: An MPC Alternative for Unknown Nonlinear Systems using Constrained Deep Learning*, Journal of Process Control, 2022

Learning to Control Building Energy System

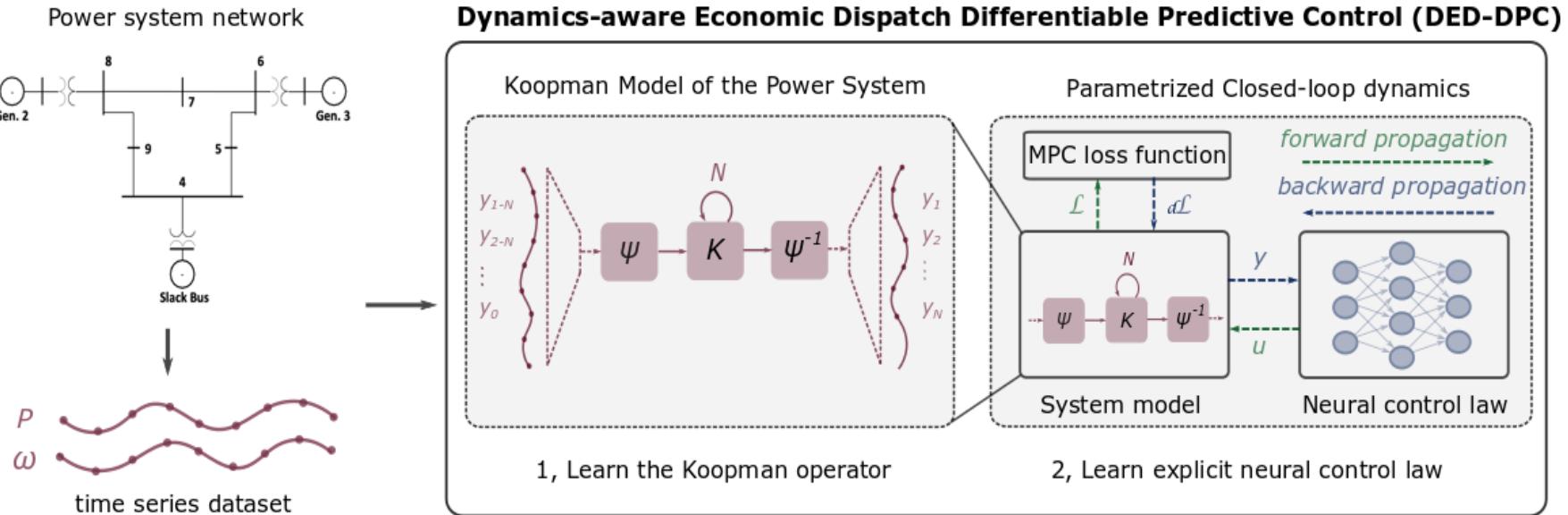


Benefits of Scientific Machine Learning

Modeling and optimal control design is roughly **10-times faster** and requires **less expertise**.

Real-time decisions are made **orders of magnitude faster** than traditional model-based approaches.

Learning to Control Power System

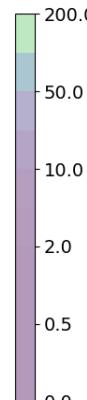
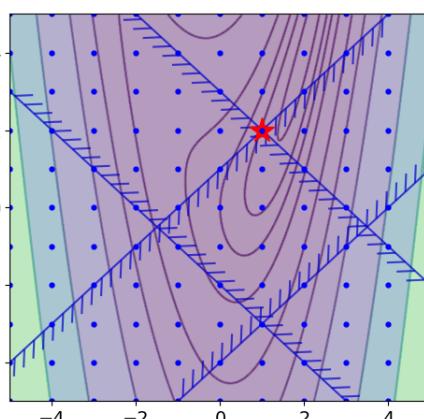
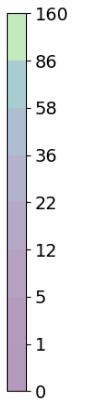
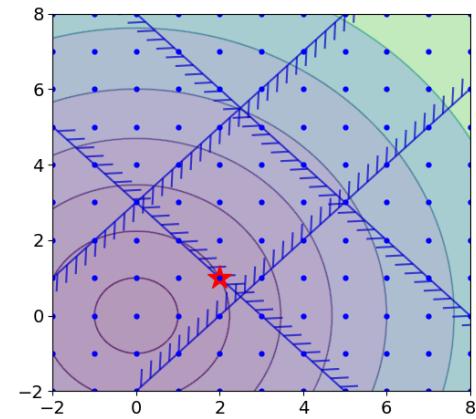


Benefits of Scientific Machine Learning

Fast prototyping by re-using code template from building control project.

Real-time decisions are made **orders of magnitude faster** than traditional model-based approaches.

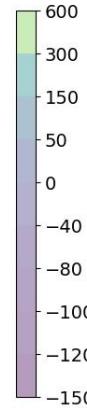
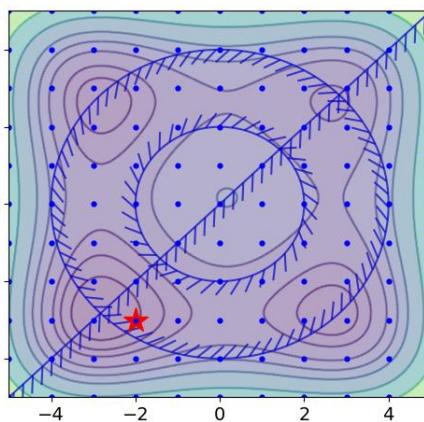
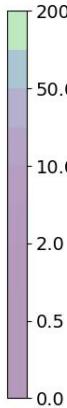
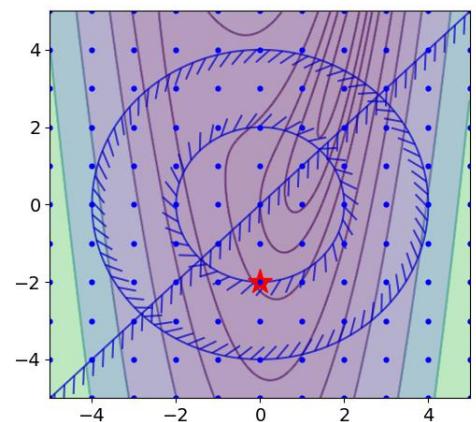
Can We Use Scientific Machine Learning for Solving Mixed-Integer Nonlinear Programming (MINLP) Problems?



$$\min_x f(x)$$

$$\text{s. t. } g(x) \leq 0$$

$$x \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}$$



Authors of Learning to Optimize for Mixed-Integer Non-Linear Programming



Bo Tang

PhD Candidate
Department of Mechanical &
Industrial Engineering
University of Toronto



Elias B. Khalil

Assistant Professor
Department of Mechanical &
Industrial Engineering
University of Toronto

SCALE AI Research Chair
Data-Driven Algorithms for Modern
Supply Chains



Ján Drgoňa

Associate Professor
Department of Civil and Systems
Engineering and the Ralph S.
O'Connor Sustainable Energy
Institute
Johns Hopkins University

Challenges of Existing Learning to Optimize Methods

1. Collecting Solutions as Training Labels for Supervised Learning is Very Expensive



Our Solution: Self-Supervised Learning Approach without requiring solutions for training.

2. Neural Networks Cannot Directly Output Integer Values



Our Solution: Differentiable Integer Correction Layers to ensure integer feasibility.

3. It is Difficult to Ensure Feasibility, Especially in Integers



Our Solution: Gradient-based Feasibility Projection to guarantee feasible integer solutions.

Selected Existing L2O methods

Ferdinando Fioretto, et al., Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. AAAI conference on AI, 2020.

Priya Donti, et al., DC3: A learning method for optimization with hard constraints, ICLR, 2021

James Kotary, et al., End-to-end constrained optimization learning: A survey. arXiv preprint arXiv:2103.16378, 2021.

He He, et al., Learning to search in branch and bound algorithms. NeurIPS, 2014.

Elias Khalil, et al., Learning to branch in mixed integer programming. AAAI Conference on AI, 2016.

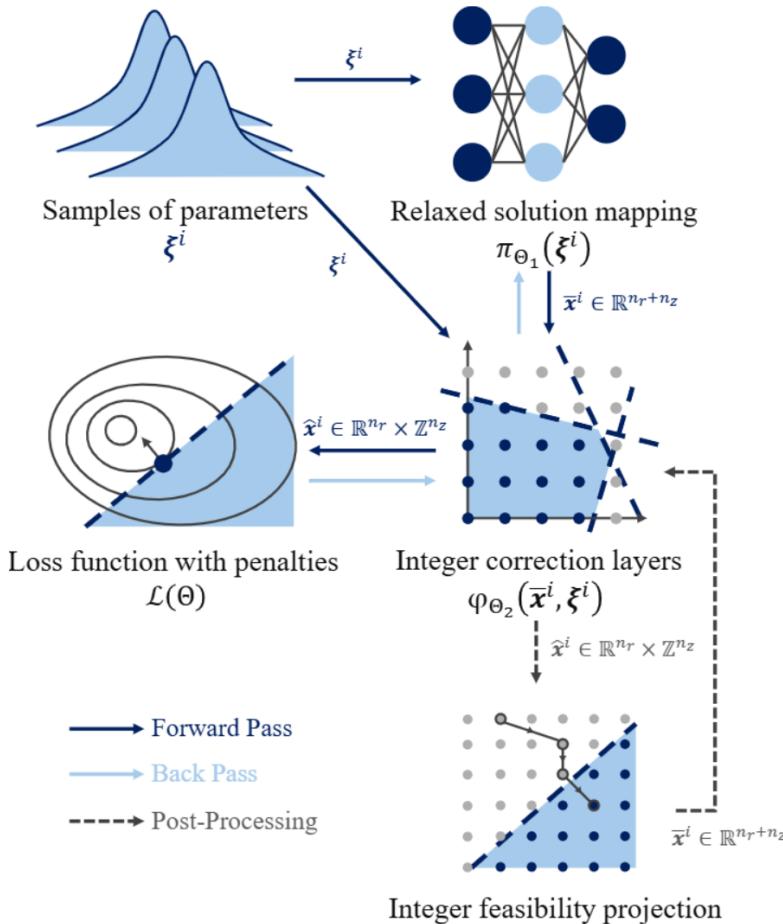
Maxime Gasse, et al., Exact combinatorial optimization with graph convolutional neural networks. NervIPS, 2019.

Timo Berthold , et al., Learning to scale mixed-integer programs. AAAI Conference on AI, 2021.

Yoshua Bengio, et al., Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research, 2021.

Dimitris Bertsimas and Bartolomeo Stellato. Online mixed-integer optimization in milliseconds. INFORMS Journal on Computing, 2022.

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

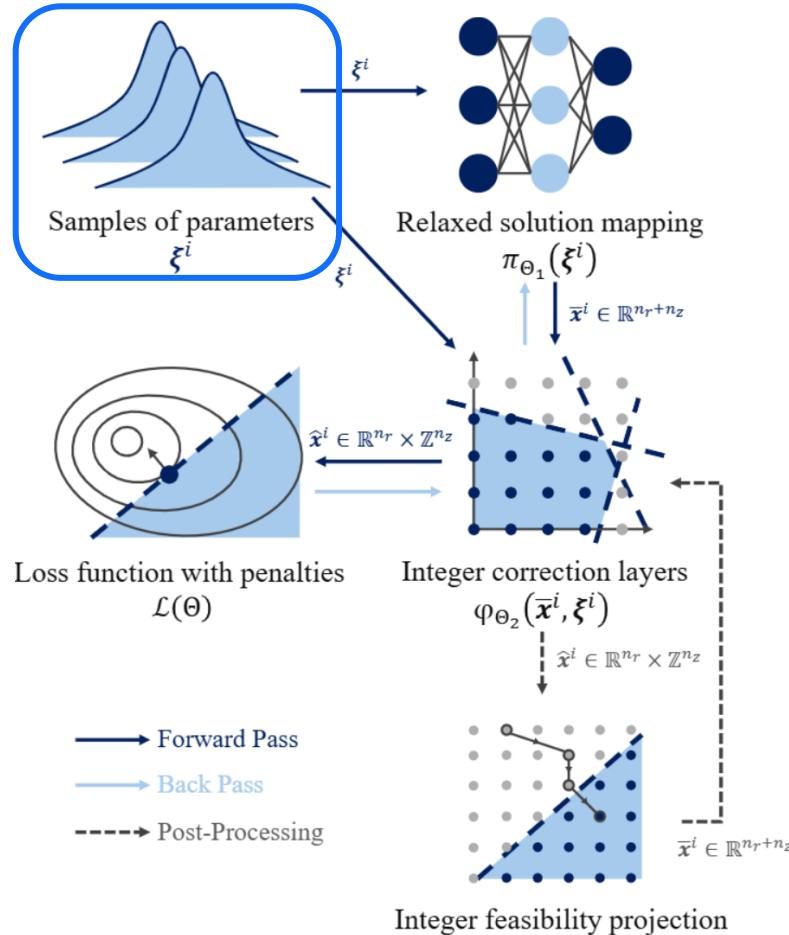
$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}[f(\hat{\mathbf{x}}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{\mathbf{x}}^i, \xi^i) \\ \text{s.t.} \quad & \mathbf{g}(\hat{\mathbf{x}}^i, \xi^i) \leq 0, \\ & \hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \psi_{\Theta}(\xi^i), \quad \forall i \in [m]. \end{aligned}$$

Penalty loss function reformulation:

$$\mathcal{L}(\Theta) = \frac{1}{m} \sum_{i=1}^m \left[f(\hat{\mathbf{x}}^i, \xi^i) + \lambda \cdot \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1 \right]$$

Amortized optimization allows scaling to
some of the largest MINLPs.
Could also be used as primal heuristics.

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

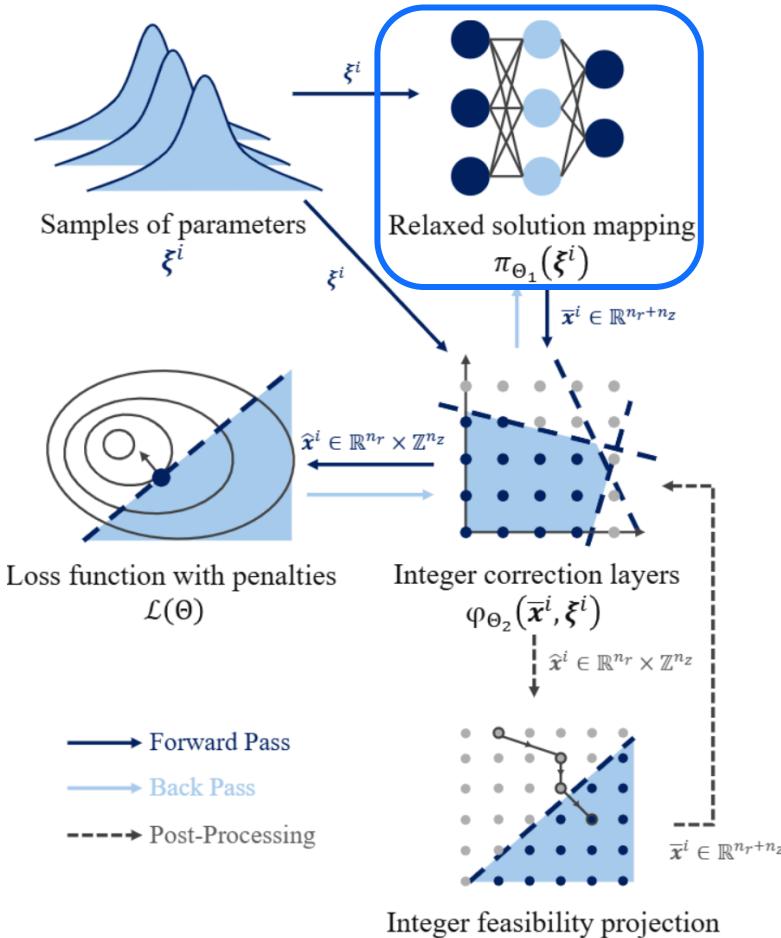
$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}[f(\hat{\mathbf{x}}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{\mathbf{x}}^i, \xi^i) \\ \text{s.t.} \quad & \mathbf{g}(\hat{\mathbf{x}}^i, \xi^i) \leq 0, \\ & \hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \psi_{\Theta}(\xi^i), \quad \forall i \in [m]. \end{aligned}$$

Penalty loss function reformulation:

$$\mathcal{L}(\Theta) = \frac{1}{m} \sum_{i=1}^m \left[f(\hat{\mathbf{x}}^i, \xi^i) + \lambda \cdot \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1 \right]$$

Parameters ξ influence the objective and constraints of the optimization problem. We sample a set of problems from a distribution.

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

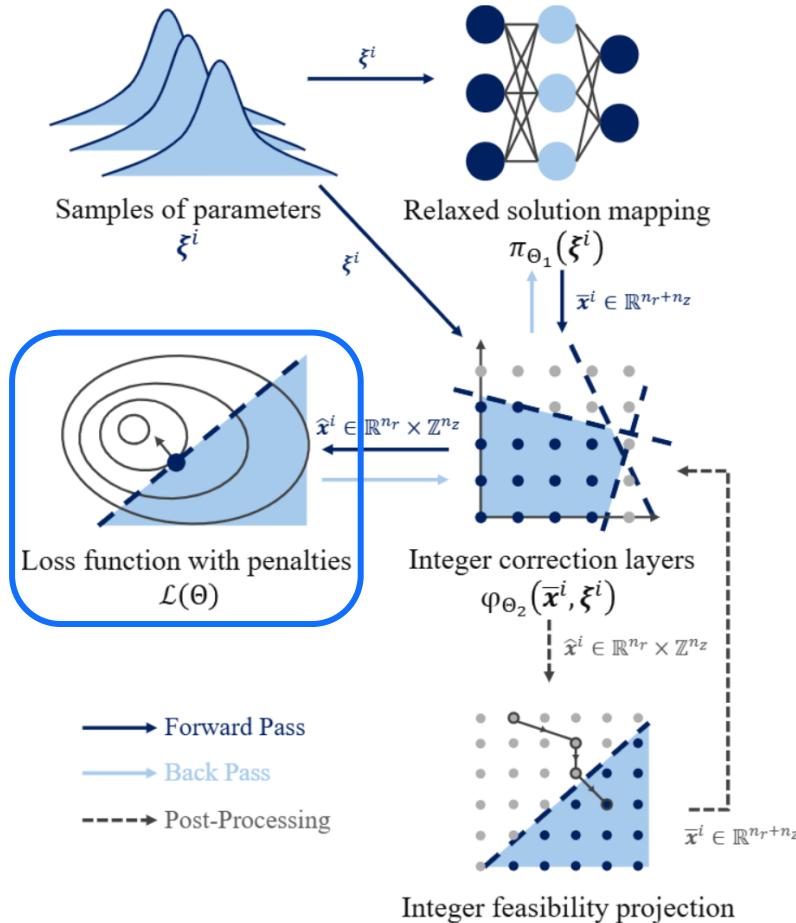
$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}[f(\hat{\mathbf{x}}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{\mathbf{x}}^i, \xi^i) \\ \text{s.t.} \quad & \mathbf{g}(\hat{\mathbf{x}}^i, \xi^i) \leq 0, \\ & \hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \psi_{\Theta}(\xi^i), \quad \forall i \in [m]. \end{aligned}$$

Penalty loss function reformulation:

$$\mathcal{L}(\Theta) = \frac{1}{m} \sum_{i=1}^m \left[f(\hat{\mathbf{x}}^i, \xi^i) + \lambda \cdot \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1 \right]$$

We aim to predict the solution $\hat{\mathbf{x}}$ with machine learning model ψ_{Θ} with inputs ξ and weights Θ .

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

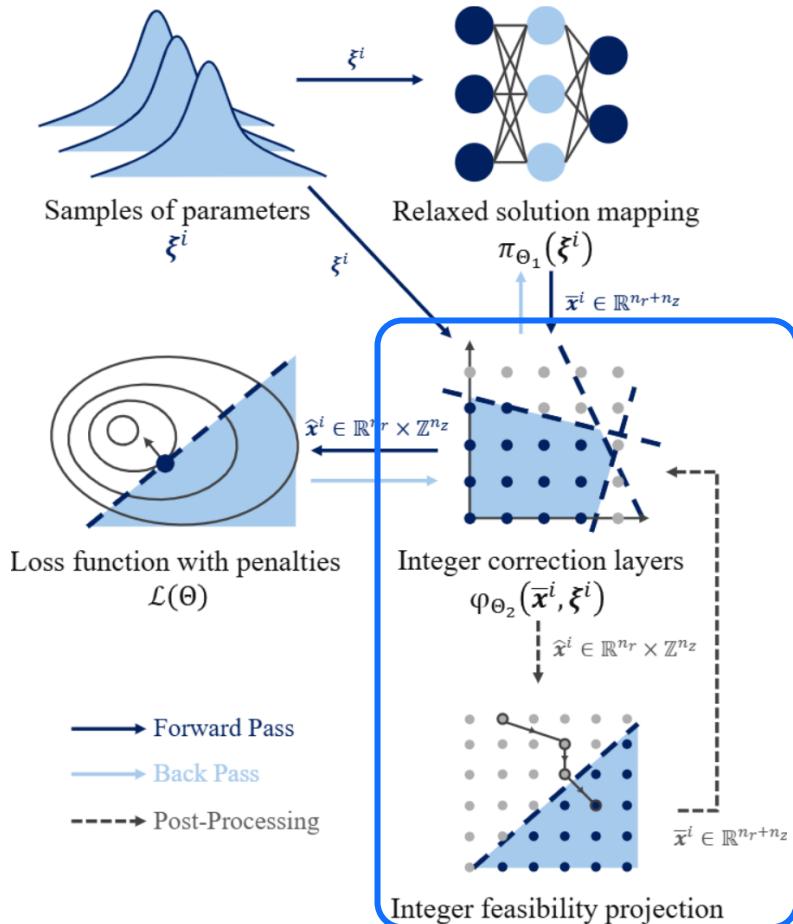
$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}[f(\hat{\mathbf{x}}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{\mathbf{x}}^i, \xi^i) \\ \text{s.t.} \quad & \mathbf{g}(\hat{\mathbf{x}}^i, \xi^i) \leq 0, \\ & \hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \psi_{\Theta}(\xi^i), \quad \forall i \in [m]. \end{aligned}$$

Penalty loss function reformulation:

$$\mathcal{L}(\Theta) = \frac{1}{m} \sum_{i=1}^m \left[f(\hat{\mathbf{x}}^i, \xi^i) + \lambda \cdot \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1 \right]$$

Until now this is standard self-supervised L2O for continuous problems.

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

$$\begin{aligned} \min_{\Theta} \quad & \mathbb{E}[f(\hat{\mathbf{x}}, \xi)] \approx \frac{1}{m} \sum_{i=1}^m f(\hat{\mathbf{x}}^i, \xi^i) \\ \text{s.t.} \quad & \mathbf{g}(\hat{\mathbf{x}}^i, \xi^i) \leq 0, \\ & \hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \psi_{\Theta}(\xi^i), \quad \forall i \in [m]. \end{aligned}$$

Penalty loss function reformulation:

$$\mathcal{L}(\Theta) = \frac{1}{m} \sum_{i=1}^m \left[f(\hat{\mathbf{x}}^i, \xi^i) + \lambda \cdot \|\mathbf{g}(\hat{\mathbf{x}}^i, \xi^i)_+\|_1 \right]$$

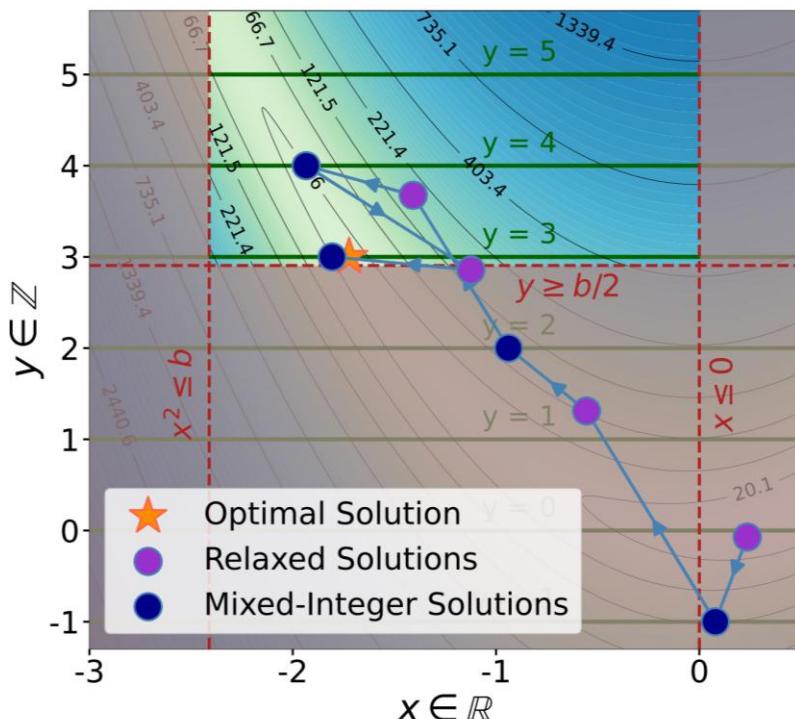
Main innovation of the paper: Integer correction layers and integer feasibility projection

Differentiable Integer Correction Layers

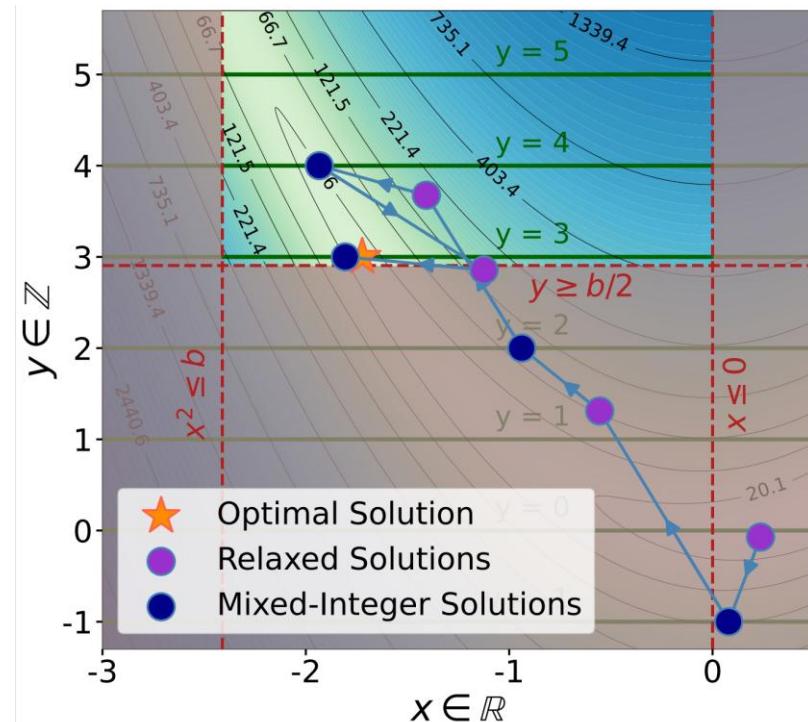
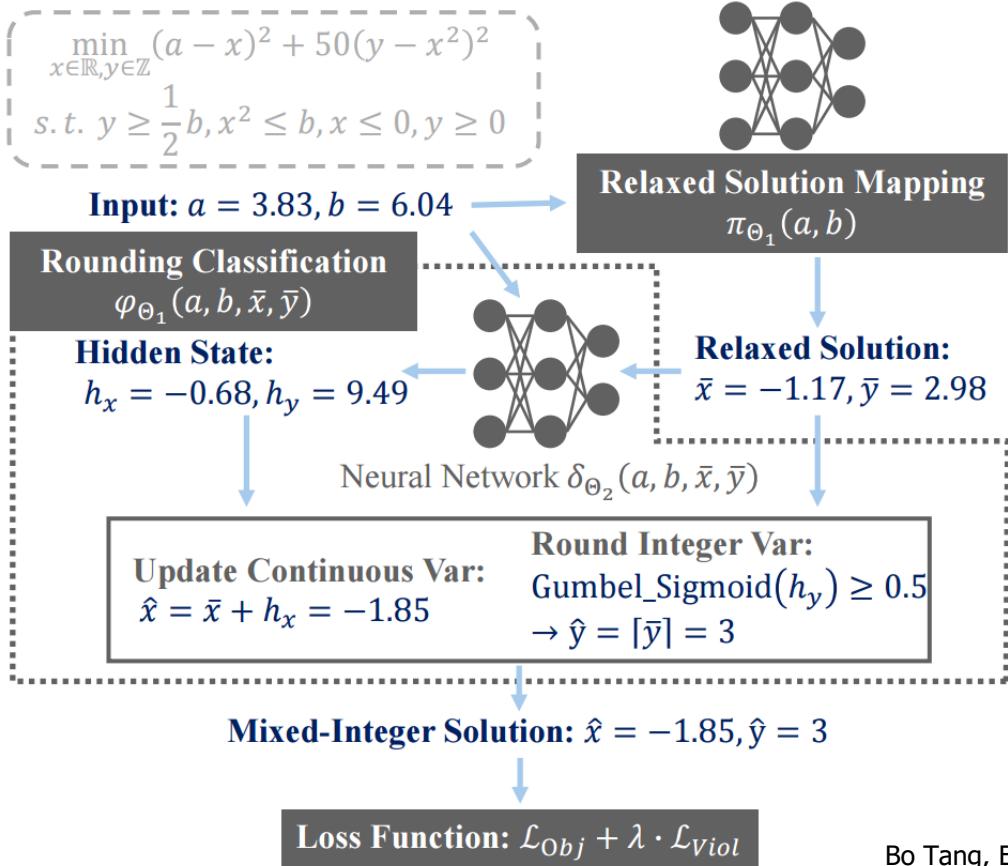
Learnable end-to-end extension of the Relaxation Enforced Neighborhood Search (RENS).

Algorithm 1 Integer Correction $\varphi_{\Theta_2}(\bar{\mathbf{x}}, \xi)$

- 1: **Input:** initial relaxed solution $\bar{\mathbf{x}}$, parameters ξ , and neural network $\delta_{\Theta_2}(\cdot)$
- 2: Obtain hidden states $\mathbf{h} \leftarrow \delta_{\Theta_2}(\bar{\mathbf{x}}, \xi)$
- 3: Update continuous variables $\hat{\mathbf{x}}_r \leftarrow \bar{\mathbf{x}}_r + \mathbf{h}_r$
- 4: Round integer variables down $\hat{\mathbf{x}}_z \leftarrow \lfloor \bar{\mathbf{x}}_z \rfloor$
- 5: **if** using *Rounding Classification* (RC) **then**
- 6: Obtain values $\mathbf{v} \leftarrow \text{Gumbel-Sigmoid}(\mathbf{h}_z)$
- 7: **else if** using *Learnable Threshold* (LT) **then**
- 8: Obtain logits $\mathbf{r} \leftarrow (\bar{\mathbf{x}}_z - \hat{\mathbf{x}}_z) - \mathbf{h}$
- 9: Obtain values $\mathbf{v} \leftarrow \text{Sigmoid}(10 \cdot \mathbf{r})$
- 10: **end if**
- 11: Obtain rounding directions $\mathbf{b} \leftarrow \mathbb{I}(\mathbf{v} > 0.5)$
- 12: Update integer variables $\hat{\mathbf{x}}_z \leftarrow \hat{\mathbf{x}}_z + \mathbf{b}$
- 13: **Output:** a mixed-integer solution $\hat{\mathbf{x}}$



Differentiable Integer Correction Layers: RC Example

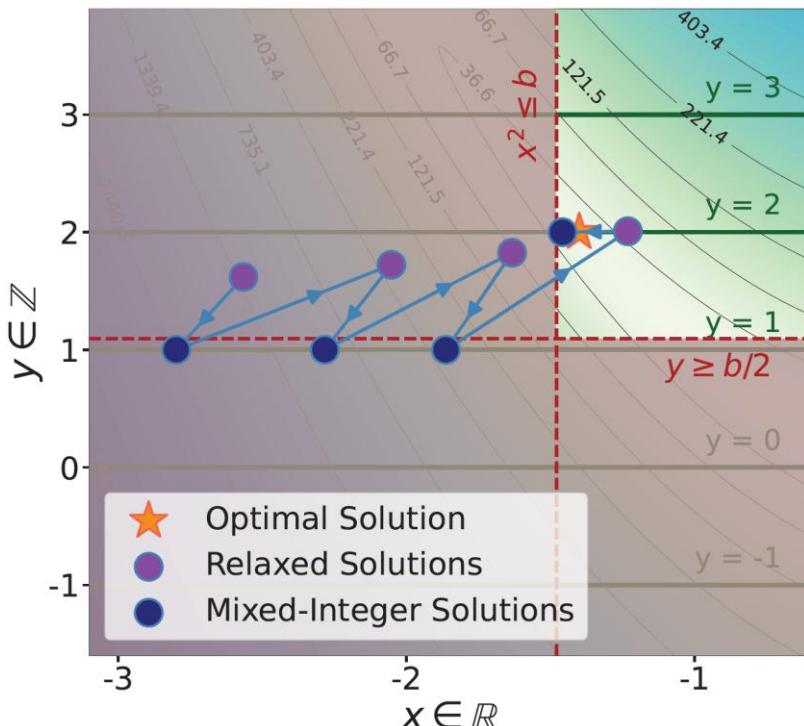


Integer Feasibility Projection

Learning-based alternative to Feasibility Pump which also alternates between rounding and projection.

Algorithm 2 Integer Feasibility Projection $\phi(\bar{\mathbf{x}}, \xi)$

```
1: Input: initial relaxed solution  $\bar{\mathbf{x}}$ , parameters  $\xi$ , integer correction layer  $\varphi_{\Theta_2}(\cdot)$ , and step size  $\eta$ 
2: while True do
3:   Obtain updated integers  $\hat{\mathbf{x}} \leftarrow \varphi_{\Theta_2}(\bar{\mathbf{x}}, \xi)$ 
4:   Compute violations  $\mathcal{V}(\hat{\mathbf{x}}, \xi) \leftarrow \|\mathbf{g}(\hat{\mathbf{x}}, \xi)_+\|_1$ 
5:   if  $\mathcal{V}(\hat{\mathbf{x}}, \xi) = 0$  then
6:     Break
7:   else
8:     Compute gradients  $\mathbf{d} \leftarrow \nabla_{\bar{\mathbf{x}}} \mathcal{V}(\hat{\mathbf{x}}, \xi)$ 
9:     Update relaxed solution  $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} - \eta \mathbf{d}$ 
10:    end if
11:  end while
12: Output: a mixed-integer solution  $\hat{\mathbf{x}}$ 
```

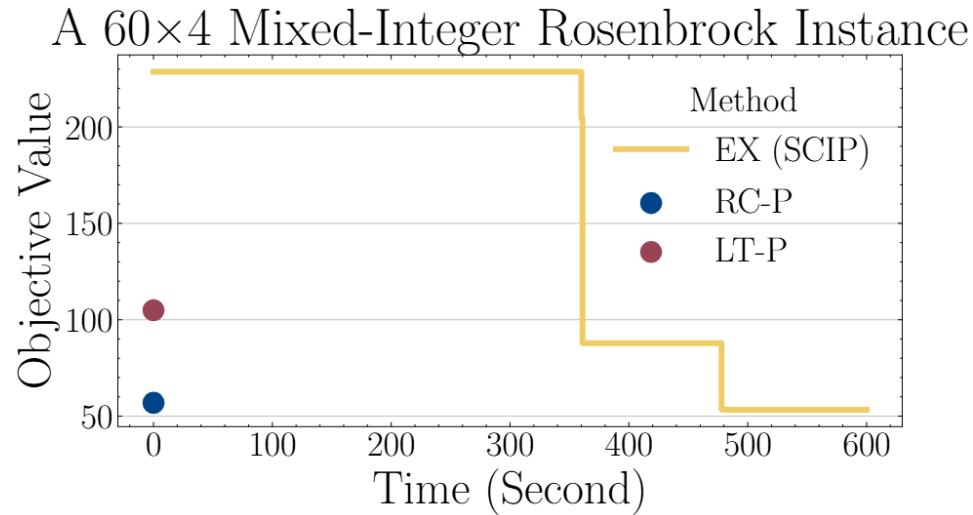
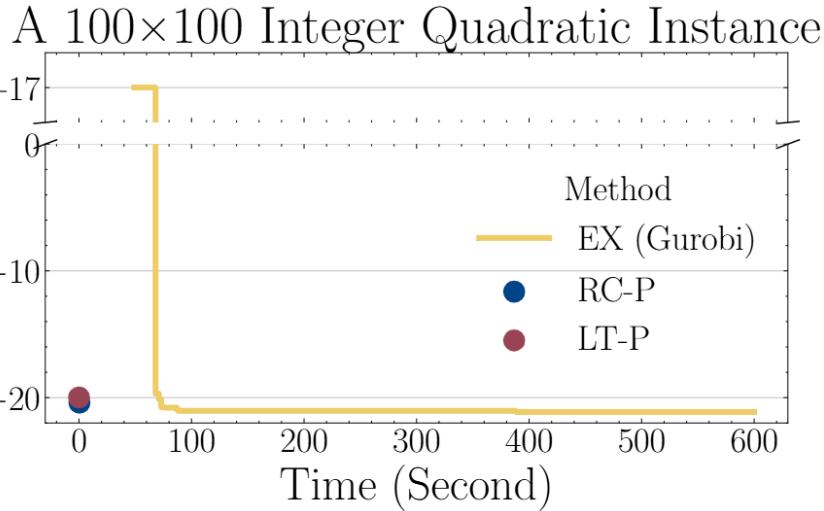


Asymptotic Guarantees of Integer Feasibility Projection

Theorem 1 (Asymptotic Convergence of Integer Feasibility Projection). *Let $\mathcal{V}(\mathbf{x}) := \|\mathbf{g}(\varphi(\mathbf{x}))_+\|_1 = \sum_{j=1}^{n_c} \max(0, g_j(\varphi(\mathbf{x})))$, where $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^{n_c}$ and $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ are continuously differentiable with gradients $\nabla \mathbf{g}$, $\nabla \varphi$ with Lipschitz constants $L_{\mathbf{g}}$, L_{φ} and bounded Jacobians $\|\nabla \mathbf{g}\| \leq G_{\mathbf{g}}$, $\|\nabla \varphi\| \leq G_{\varphi}$. Assume all gradient descent iterates $\mathbf{x}^{(k)} \in \mathcal{D} := \{\mathbf{x} \in \mathbb{R}^n : \mathcal{V}(\mathbf{x}) > 0\}$, and the number of active constraints is uniformly bounded $|I_{\mathbf{x}}| := |\{j : g_j(\varphi(\mathbf{x})) > 0\}| \leq \bar{n}_c$. Then for step size $\eta \in (0, \frac{1}{L}]$, where $L := \bar{n}_c(G_{\mathbf{g}}L_{\varphi} + G_{\varphi}L_{\mathbf{g}})$, the following hold:*

- (i) **L -smoothness:** $\mathcal{V} \in C^1(\mathcal{D})$, with $\nabla \mathcal{V}(\mathbf{x}) = \sum_{j \in I_{\mathbf{x}}} \nabla \varphi(\mathbf{x})^\top \nabla g_j(\varphi(\mathbf{x}))$, and $\nabla \mathcal{V}$ is Lipschitz continuous on compact subsets of \mathcal{D} with Lipschitz constant at most L .
- (ii) **Descent and vanishing gradient:** Gradient descent generates a non-increasing sequence $\mathcal{V}(\mathbf{x}^{(k)}) \rightarrow \mathcal{V}^* \geq 0$, and $\lim_{k \rightarrow \infty} \|\nabla \mathcal{V}(\mathbf{x}^{(k)})\| = 0$.
- (iii) **Convergence to feasibility:** If every $\mathbf{x}^* \in \mathcal{D}$ with $\mathcal{V}(\mathbf{x}^*) > 0$ satisfies $\exists j \in I_{\mathbf{x}^*}$ such that $\nabla g_j(\varphi(\mathbf{x}^*)) \neq 0$, then $\lim_{k \rightarrow \infty} \mathcal{V}(\mathbf{x}^{(k)}) = 0$.

L2O for MINLP: Comparison with SOTA Solvers

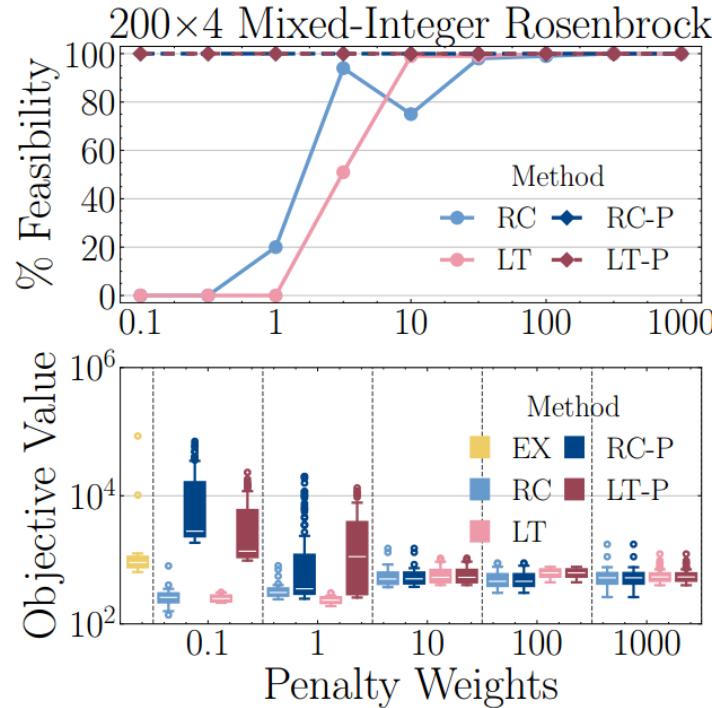
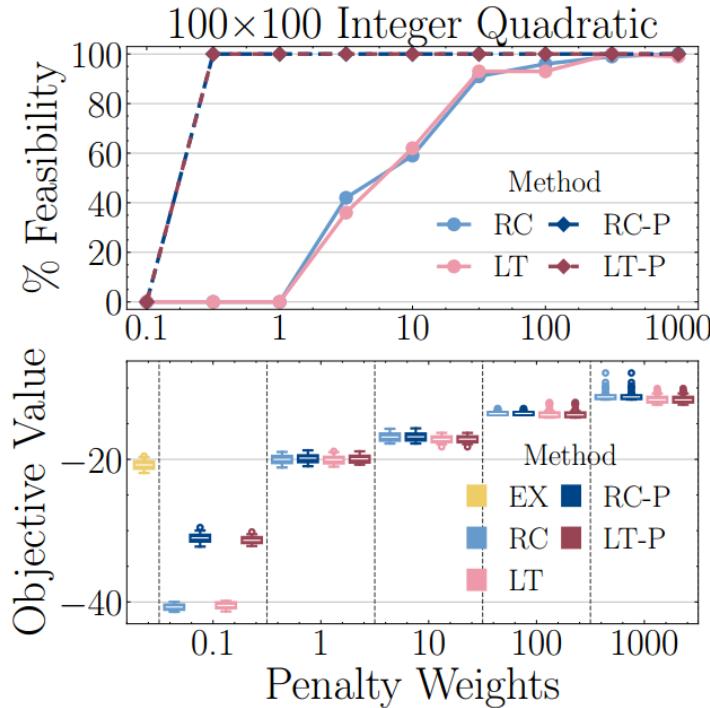


Exact solvers such as Gurobi and SCIP can find better solutions over time but are slow. In contrast, our methods achieve **high-quality feasible solutions within milliseconds**. Our methods provide **up to 5 orders of magnitude speedup**.

<https://github.com/pnnl/L2O-pMINLP>

Bo Tang, Elias B. Khalil, Ján Drgoňa, Learning to Optimize for Mixed-Integer Non-linear Programming, arXiv:2410.11061, 2024

Effect of Penalty Weight

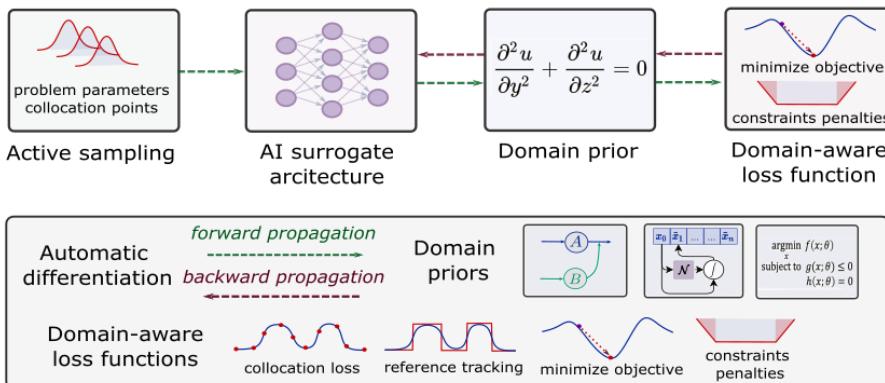


Our approach achieves **comparable or better objective values compared to exact solvers.**

<https://github.com/pnnl/L2O-pMINLP>

Summary

- **Scientific machine learning (SciML)**
methods integrating deep learning, constrained optimization, physics-based modeling, and control
 - Learning to solve (L2S)
 - Learning to optimize (L2O)
 - Learning to model (L2M)
 - Learning to control (L2C)



- **Challenges**

- Mixed-integer decisions
- Vanishing and exploding gradients
- Offline pre-training vs online adaptation
- Computational cost of uncertainty quantification and guarantees

- **Opportunities**

- Sampling efficiency via active sampling strategies
- Control of high-dimensional PDEs
- Applications in large-scale safety-critical systems
- Development of new software tools

Follow Up

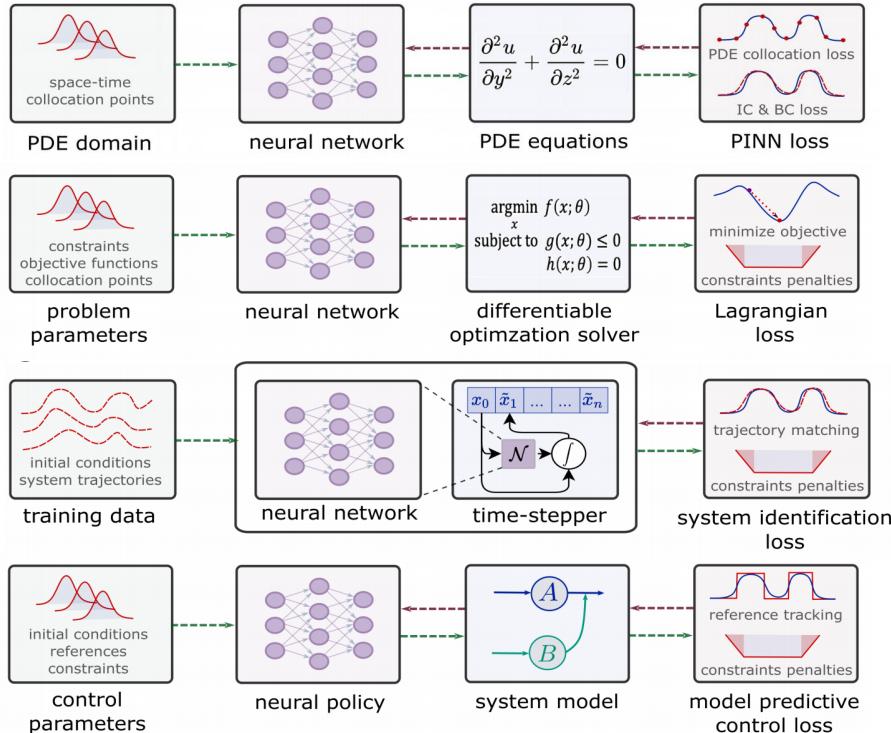
- **Safe Physics-Informed Learning in Dynamics and Control**
 - TuB03 Tutorial Session, Plaza CF, 13:30-15:00
- **Learning Interpretable Control Policies: Unifying Reinforcement Learning, Differentiable Programming and Bayesian Optimization**
 - TuC04 Tutorial Session, Governor's Sq. 15, 15:30-17:00,
- **NeuroMANCER Library tutorial – in 5 min**
 - github.com/pnnl/neuromancer

drgona.github.io

jdronga1@jh.edu

**Energy at
Hopkins**

NeuroMANCER Scientific Machine Learning Library



Open-source library in PyTorch

- Physics-informed Neural Networks
- Learning to optimize
- Neural differential equations
- Learning to control



github.com/pnnl/neuromancer

NeuroMANCER Scientific Machine Learning Library

1. Mathematical formulation

$$\begin{aligned} & \min_{\Theta} (1 - x)^2 + p(y - x^2)^2 \\ \text{s.t. } & (p/2)^2 \leq x^2 + y^2 \leq p^2, \quad x \geq y \\ & x = \pi_{\Theta}(p) \end{aligned}$$

2. Python code interface

```
import neuromancer as nm

p = nm.variable('p')
x = nm.variable('x')
y = nm.variable('y')

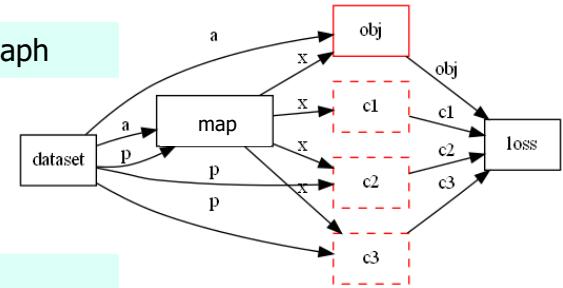
obj = ((1-x)**2 + p*(y-x**2)**2).minimize(weight=1.0, name='obj')
c1 = (p/2)**2 <= x**2 + y**2
c2 = x**2 + y**2 <= p**2
c3 = x >= y

net = nm.MLP(insize=2, outsize=2, hsizes=[80]*4)
map = nm.Node(net, input_keys=['p'], output_keys=['x', 'y'])

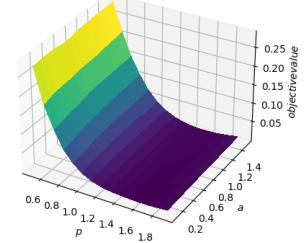
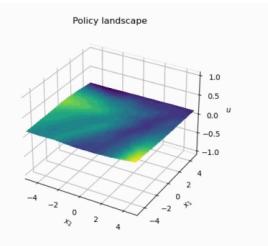
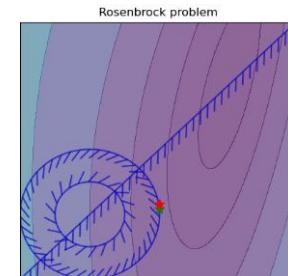
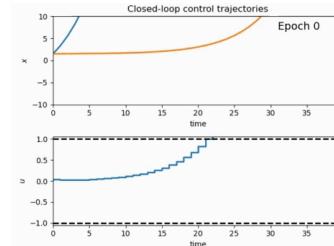
loss = nm.PenaltyLoss([obj], [c1, c2, c3])
problem = nm.Problem([map], loss)
optimizer = torch.optim.AdamW(problem.parameters())
trainer = nm.Trainer(problem, data, optimizer)
best_model = trainer.train()
```

PyTorch

3. Problem graph



4. Results



NeuroMANCER Team



Aaron Tuor



James Koch



**Madelyn
Shapiro**



**Rahul
Birmiwal**



**Bruno
Jacob**



**Draguna
Vrabie**



U.S. DEPARTMENT OF
ENERGY



Pacific Northwest
NATIONAL LABORATORY

Non-Asymptotic Guarantees of Integer Feasibility Projection

Theorem 2 (Non-Asymptotic Convergence of Integer Feasibility Projection). *Under the assumptions of Theorem 1, suppose gradient descent is applied to the function $\mathcal{V}(\mathbf{x}) = \sum_{j=1}^{n_c} \max(0, g_j(\varphi(\mathbf{x})))$, with fixed step size $\eta \in (0, \frac{1}{L}]$, where $L := \bar{n}_c(G_{\mathbf{g}}L_{\varphi} + G_{\varphi}L_{\mathbf{g}})$ is an upper bound on the Lipschitz constant of $\nabla \mathcal{V}$ over the region $\mathcal{D} := \{\mathbf{x} : \mathcal{V}(\mathbf{x}) > 0\}$. Then for any number of iterations $K \geq 1$, the minimum gradient norm over the first K iterates satisfies*

$$\min_{0 \leq k < K} \|\nabla \mathcal{V}(\mathbf{x}^{(k)})\|^2 \leq \frac{2}{\eta K} [\mathcal{V}(\mathbf{x}^{(0)}) - \mathcal{V}^*],$$

where $\mathcal{V}^* := \inf_{x \in \mathcal{D}} \mathcal{V}(\mathbf{x}) \geq 0$. In particular, to ensure $\min_{0 \leq k < K} \|\nabla \mathcal{V}(\mathbf{x}^{(k)})\| \leq \delta$, it suffices to run $K \geq \frac{2}{\eta \delta^2} (\mathcal{V}(\mathbf{x}^{(0)}) - \mathcal{V}^*)$ iterations with complexity $K = \mathcal{O}\left(\frac{1}{\delta^2}\right)$. Furthermore, if $\mathcal{V}^* = 0$, then for any $\epsilon > 0$ this implies approximate feasibility $\mathcal{V}(\mathbf{x}^{(k)}) < \epsilon$ for all $k \geq K_{\epsilon}$ for some K_{ϵ} .

The proofs of the above theorems, iteration complexity, an alternative asymptotic convergence via Łojasiewicz inequality, and other supplementary theoretical analysis can be found in Appendix C.

Benchmark Methods

Method	Description
EX (Exact Solver)	Solves problems exactly using traditional solver with 1000-sec time-limit as a benchmark.
N1 (Root Node Solution)	Finds the first feasible solution from the root node of the solver, combining various heuristics.
RC (Rounding Classification)	A neural network-based correction layer that learns a classification to determine how to round each integer variable.
LT (Learnable Thresholding)	A neural network-based correction layer that learns a threshold value to decide to round up or down for each integer variable.
RC-P (RC + Feasibility Projection)	RC combined with feasibility projection, which corrects infeasibilities while preserving integer constraints.
LT-P (LT + Feasibility Projection)	LT combined with feasibility projection, which corrects infeasibilities while preserving integer constraints.

L2O for MINLP: Empirical Evaluation

$$\min_{x \in \mathbb{Z}^n} \frac{1}{2} x^T Q x + p^T x \quad \text{subject to } Ax \leq b$$

Integer Quadratic Problems (IQPs). Each problem size is evaluated on a test set of 100 instances.

Method	RC				RC-P				LT			
	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
100×100	-13.54	-13.6	96%	0.0022	-13.54	-13.57	100%	0.005	-13.65	-13.77	93%	0.0023
200×200	-31.62	-31.71	97%	0.0021	-31.62	-31.71	100%	0.005	-31.34	-31.61	95%	0.0022
500×500	-73.31	-73.38	86%	0.0025	-73.31	-73.38	100%	0.0065	-72.36	-72.48	94%	0.0026
1000×1000	-142.7	-142.7	82%	0.0042	-142.7	-142.7	100%	0.009	-142.6	-142.6	100%	0.0047
Method	LT-P				EX				N1			
Metric	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
100×100	-13.65	-13.77	100%	0.01	-20.79	-20.78	100%	1237	1.5E+18	1.4E+18	100%	104.2
200×200	-31.34	-31.61	100%	0.0064	-	-	-	-	-	-	-	-
500×500	-72.36	-72.48	100%	0.0063	-	-	-	-	-	-	-	-
1000×1000	-142.6	-142.6	100%	0.0086	-	-	-	-	-	-	-	-

L2O for MINLP: Empirical Evaluation

$$\min_{\mathbf{x} \in \mathbb{Z}^n} \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} + \mathbf{p}^\top \sin(\mathbf{x}) \quad \text{subject to } \mathbf{A} \mathbf{x} \leq \mathbf{b}$$

Integer Non-convex Problems (INPs). Each problem size is evaluated on a test set of 100 instances.

Method	RC				RC-P				LT			
	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
Metric												
100×100	1.664	1.594	100%	0.0022	1.664	1.594	100%	0.0060	0.669	0.649	96%	0.0021
200×200	1.472	1.436	99%	0.0022	1.471	1.436	100%	0.0054	-0.356	-0.373	100%	0.0023
500×500	0.526	0.526	96%	0.0029	0.524	0.526	100%	0.0061	-1.374	-1.594	98%	0.0029
1000×1000	1.423	0.809	97%	0.0040	1.423	0.809	100%	0.0115	-3.744	-3.716	99%	0.0050
Method	LT-P				EX				N1			
Metric	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time
100×100	0.669	0.649	100%	0.0058	256.93	134.62	14%	1001	4411	155.2	14%	940.4
200×200	-0.356	-0.373	100%	0.0056	-	-	-	-	-	-	-	-
500×500	-1.374	-1.594	100%	0.0072	-	-	-	-	-	-	-	-
1000×1000	-3.744	-3.716	100%	0.0117	-	-	-	-	-	-	-	-

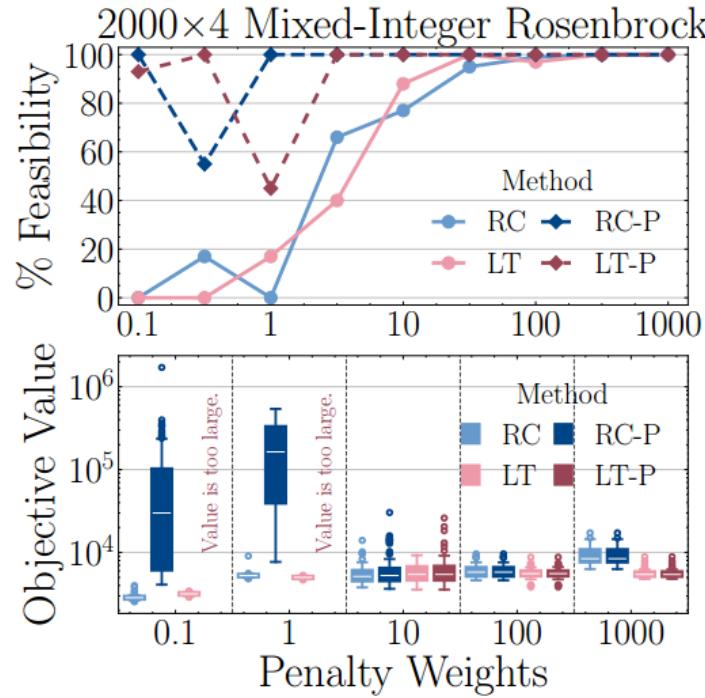
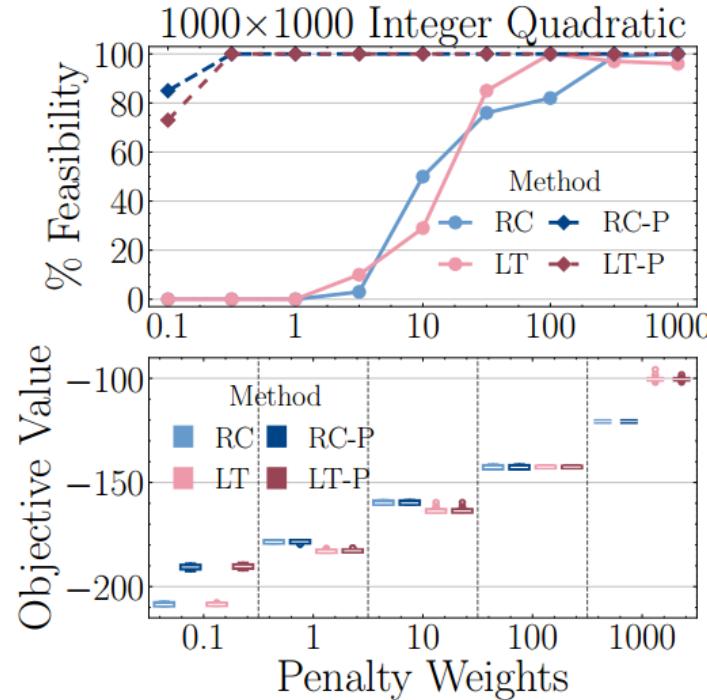
L2O for MINLP: Empirical Evaluation

$$\min_{\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{Z}^n} \|\mathbf{a} - \mathbf{x}\|_2^2 + 50\|\mathbf{y} - \mathbf{x}^2\|_2^2 \quad \text{subject to } \|\mathbf{x}\|_2^2 \leq nb, \mathbf{1}^\top \mathbf{y} \geq \frac{nb}{2}, \mathbf{p}^\top \mathbf{x} \leq 0, \mathbf{Q}^\top \mathbf{y} \leq 0,$$

Mixed-integer Rosenbrock Problems (MIRBs). Each problem size is evaluated on a test set of 100 instances.

Method	RC				RC-P				LT			
	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
20×4	59.39	48.86	100%	0.0019	59.39	48.86	100%	0.0048	62.51	63.40	100%	0.0020
200×4	503.5	461.7	99%	0.0021	504.2	461.7	100%	0.0052	622.8	626.0	100%	0.0026
50×4	5938	5792	99%	0.0033	5942	5792	100%	0.0070	5612	5558	97%	0.0030
1000×4	6.7E+4	6.7E+4	76%	0.0121	9.8E+4	7.3E+4	100%	0.0824	4.8E+4	3.5E+4	66%	0.0127
Method	LT-P				EX				N1			
Metric	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
20×4	62.51	63.40	100%	0.0055	64.67	59.16	100%	1005	87.83	77.34	100%	0.0813
200×4	622.8	626.0	100%	0.0062	8.4E+5	908.8	100%	1002	3.7E+8	957.4	100%	0.2608
2000×4	5615	5558	100%	0.0071	4.7E+10	9262	96%	1002	8.3E+12	9379	95%	71.91
20000×4	8.0E+4	4.5E+4	100%	0.0639	1.1E+15	1.0E5	78%	1040	1.2E+15	1.0E5	78%	782.1

Effect of Penalty Weight



Integer feasibility projection (RC-P and LT-P), improves feasibility even with smaller penalty weights. Without projection there is a **trade-off between feasibility and objective values** (RC and LT).