Learning to Optimize for Mixed-Integer Non-Linear Programming

Ján Drgoňa
Associate Professor @ CaSE & ROSEI

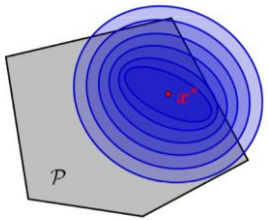
Optimizing Complex Energy Systems is Hard

- ***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 Solution Methods

Constrained Optimization

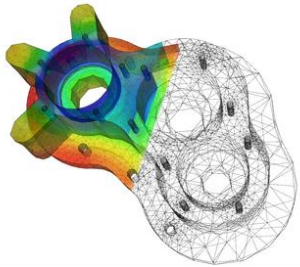
$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & b(x) \geq 0 \\ & c(x) = 0. \end{aligned}$$



- Requires prior knowledge of objective function and constraints

Differential Equations

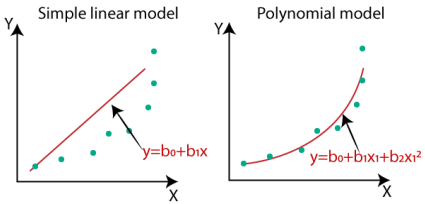
$$\frac{dy}{dx} = f(x)$$



- Requires prior knowledge of the physics to be modeled

Supervised Learning

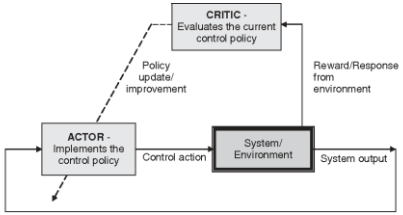
$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta))$$



- Requires large labeled datasets

Reinforcement Learning

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



- Requires environment model to sample

More domain knowledge

Less domain knowledge

Challenge 2: Heterogenous Solution Tools

Constrained Optimization

Differential Equations

Supervised Learning

Reinforcement Learning

More domain knowledge

Less domain knowledge

Scientific Machine Learning (SciML)

What?

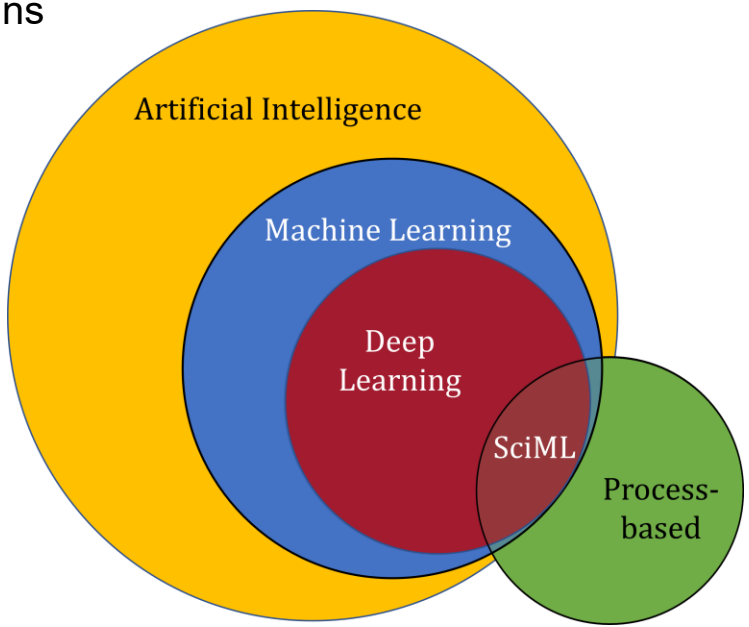
- SciML systematically integrates ML methods with mathematical models and algorithms developed in various scientific and engineering domains

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

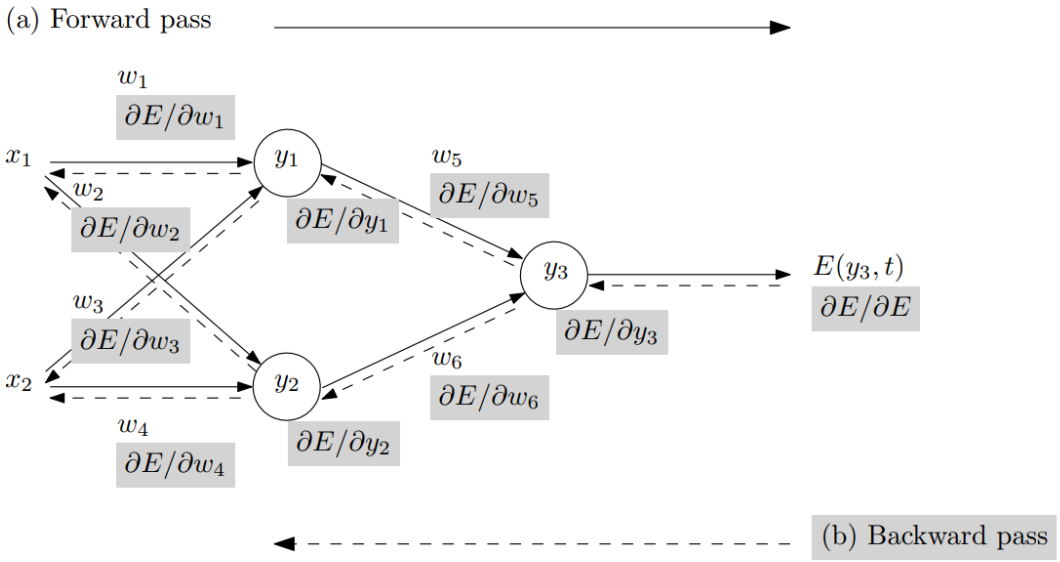


Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 422–440, 2021.

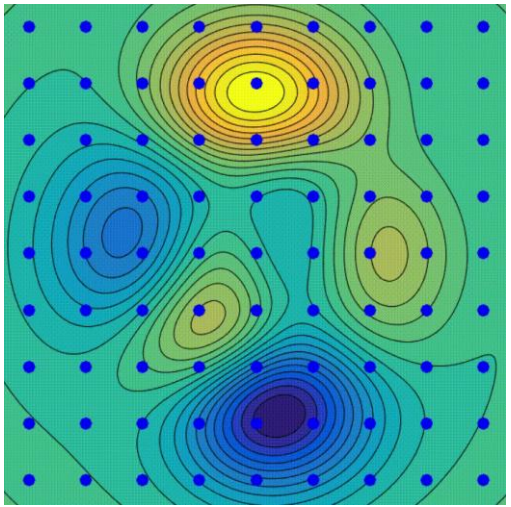
Image source: https://sciml.wur.nl/reviews/sciml/sciml.html

Automatic Differentiation in Machine Learning

Forward and Backward propagation through computational graph



Gradient descent algorithm



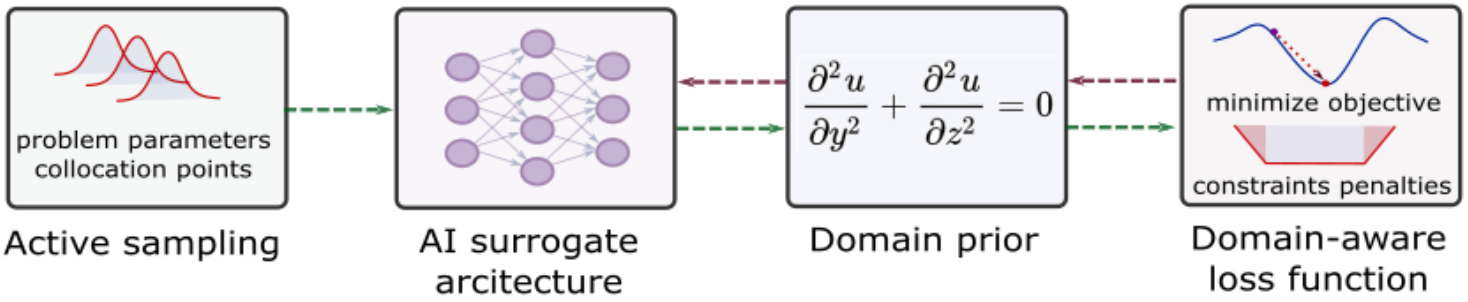
Animation source: wikipedia

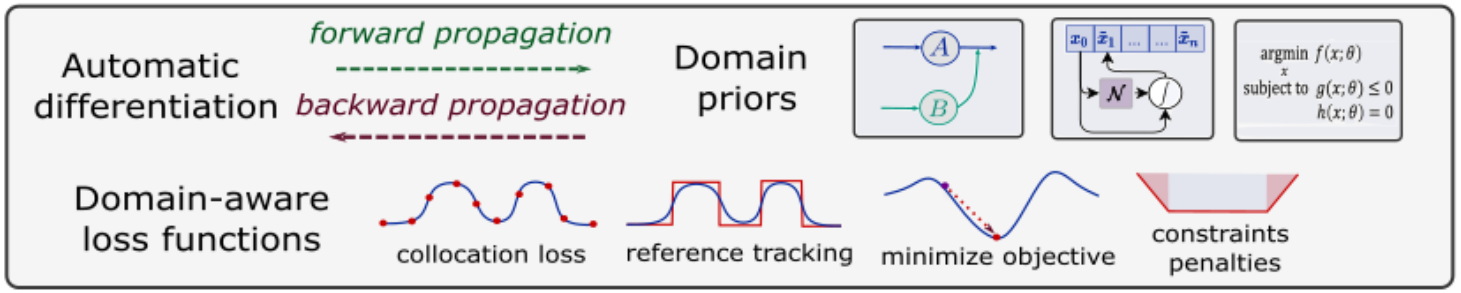
Baydin, Atilim Gunes et al. Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research, 2015

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

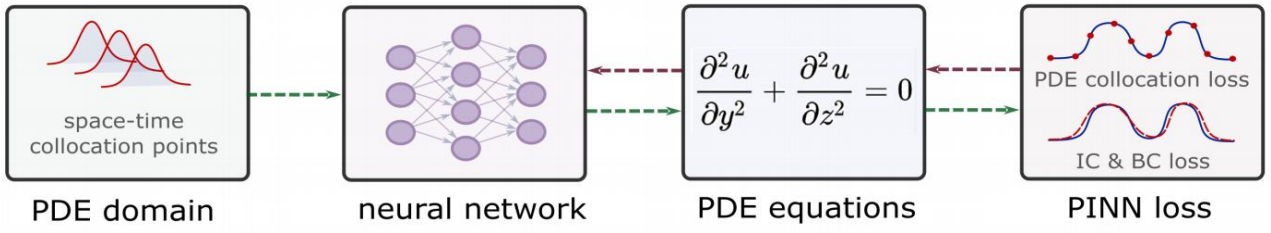
Components of Scientific Machine Learning



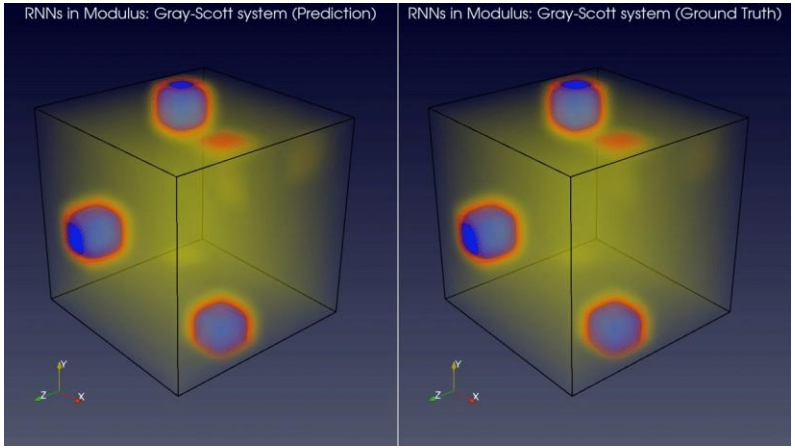


Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 2021.
Thiyagalingam, J., Shankar, M., Fox, G. et al. Scientific machine learning benchmarks. Nature Reviews Physics 4, 413–420, 2022.
Nghiem T., Drgona J., et al. Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems, ACC, 2023.

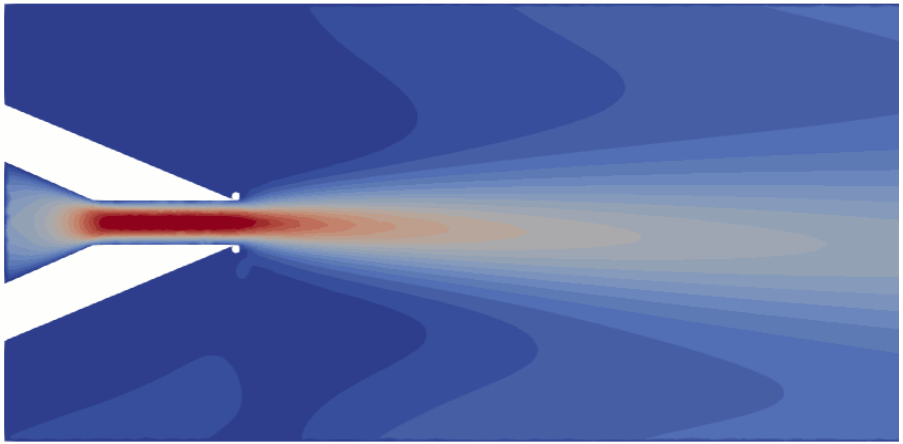
Learning to Solve Differential Equations with Physics-Informed Neural Networks (PINNs)



Training neural networks as PDE solutions



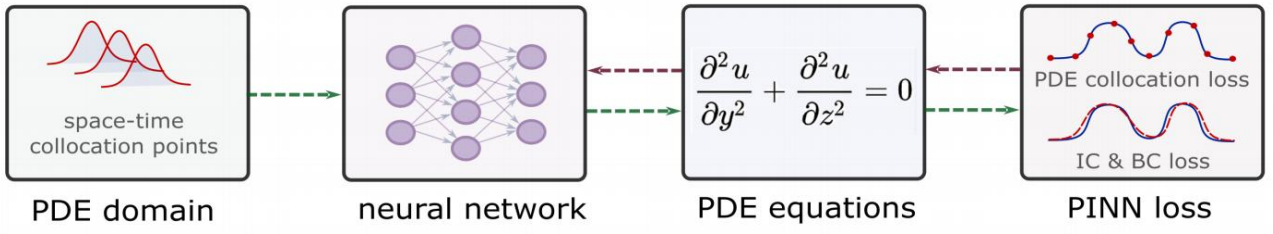
Application: Parameter estimation from data



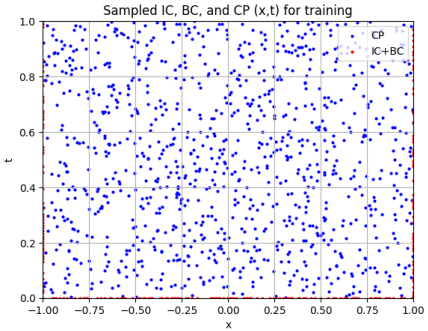
M. Raissi, et al., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics,* 2019

Images: NVIDIA Modulus

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.

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

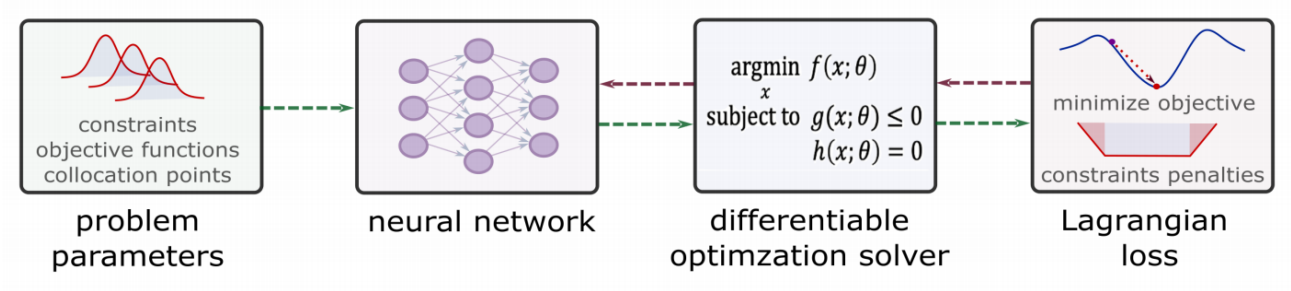
$$\ell_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |y(t_u^i, x_u^i) - NN_\theta(t_u^i, x_u^i)|^2$$

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

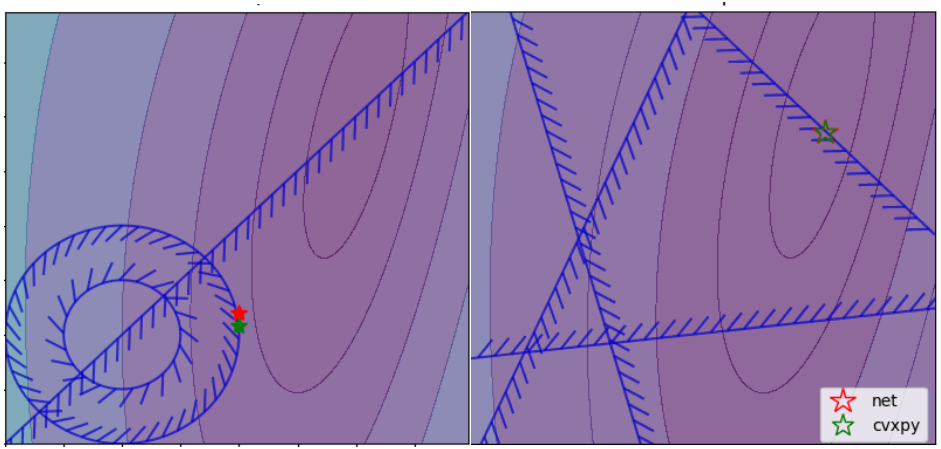
https://github.com/pnnl/neuromancer/blob/master/examples/PDEs/Part_2_PINN_BurgersEquation.ipynb

M. Raissi, et al., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics,* 2019

Learning to Optimize (L2O) with Constraints



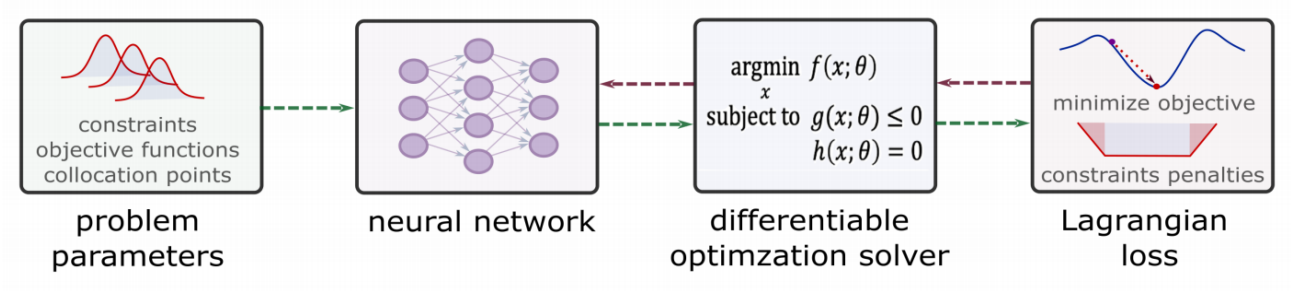
Training neural networks as optimization solutions



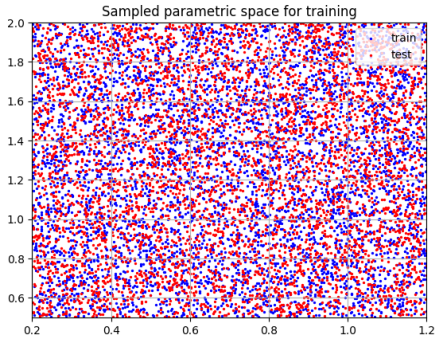
Application: solving optimal power flow

James Kotary, et al., End-to-End Constrained Optimization Learning: A Survey, IJCAI, 2021

Learning to Optimize (L2O) with Constraints



Dataset: collocation points in the parametric space.



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

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

$$\hat{x} = \text{proj}_{g(x,\xi)\le 0}(x,\xi)$$

Loss function: minimizing objective function and constraints penalties.

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

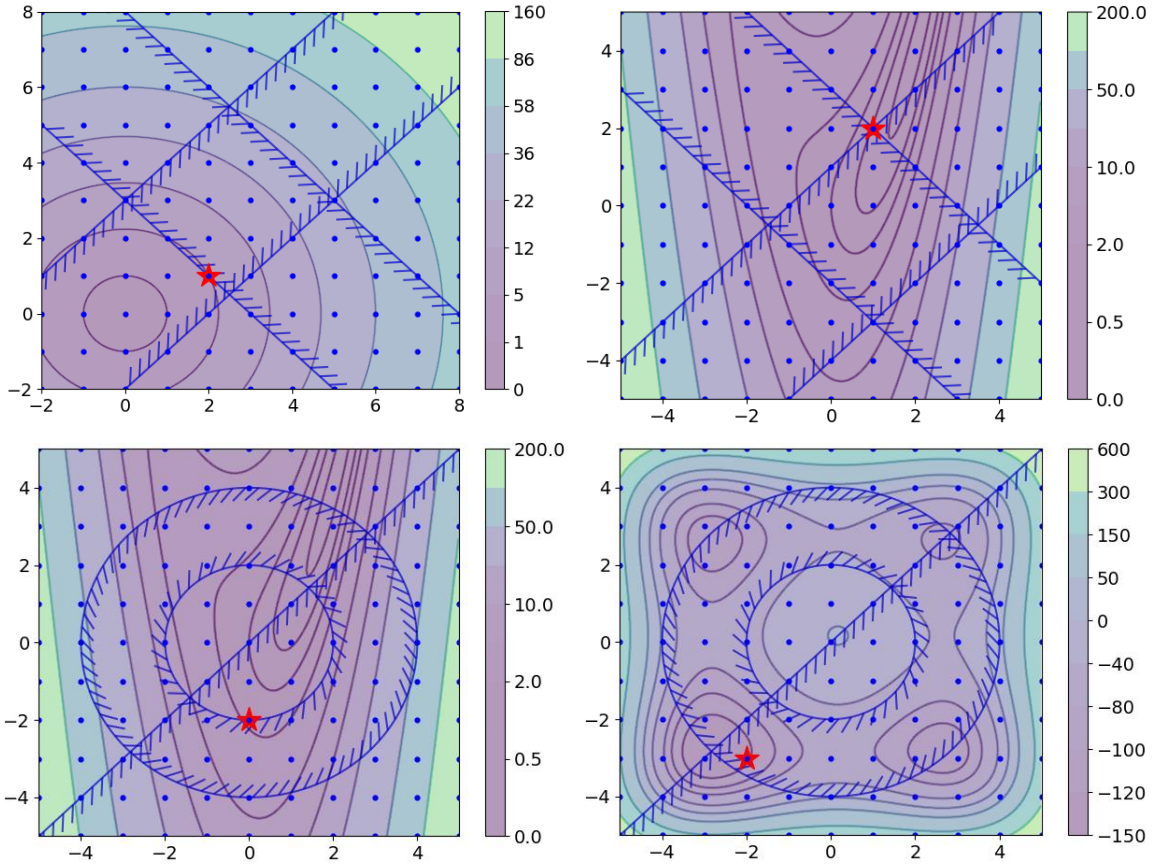
$$\ell_g = \frac{1}{m}\sum_{i=1}^{m} |\texttt{RELU}(g(x^i,\xi^i))|^2$$

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

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

A. Agrawal, et al., *Differentiable Convex Optimization Layers*, 2019
P. Donti, et al., *DC3: A learning method for optimization with hard constraints*, 2021

Can We Use Learning to Optimize (L2O) for Solving Mixed-Integer Nonlinear Programming (MINLP) Problems?



$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

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

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

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. NeruIPS, 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.

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

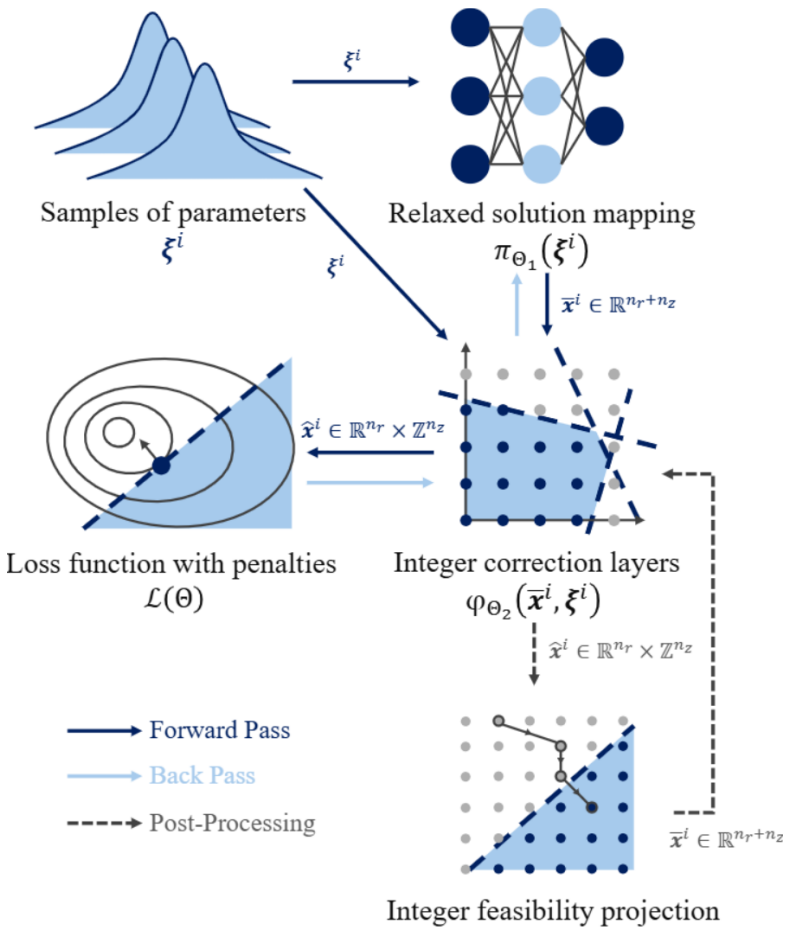
Assistant Professor
Department of Mechanical & Industrial Engineering
University of Toronto

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

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

$$\min_{\Theta} \quad \mathbb{E}\big[f(\hat{\mathbf{x}}, \boldsymbol{\xi})\big] \approx \frac{1}{m}\sum_{i=1}^{m} f(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i)$$

$$\text{s.t.} \quad \mathbf{g}(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i) \leq 0,$$

$$\hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \boldsymbol{\psi}_{\Theta}(\boldsymbol{\xi}^i), \quad \forall i \in [m].$$

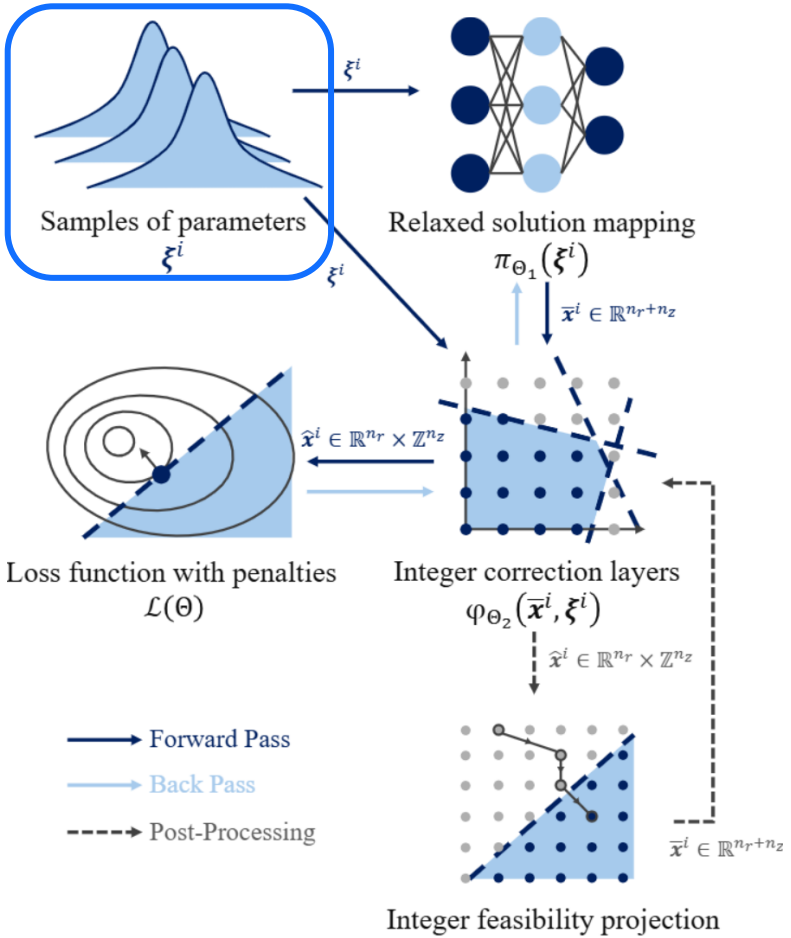
Penalty loss function reformulation:

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

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

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

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

$$\min_{\Theta} \quad \mathbb{E}\big[f(\hat{\mathbf{x}}, \boldsymbol{\xi})\big] \approx \frac{1}{m}\sum_{i=1}^{m} f(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i)$$

$$\text{s.t.} \quad \mathbf{g}(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i) \leq 0,$$

$$\hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \boldsymbol{\psi}_{\Theta}(\boldsymbol{\xi}^i), \quad \forall i \in [m].$$

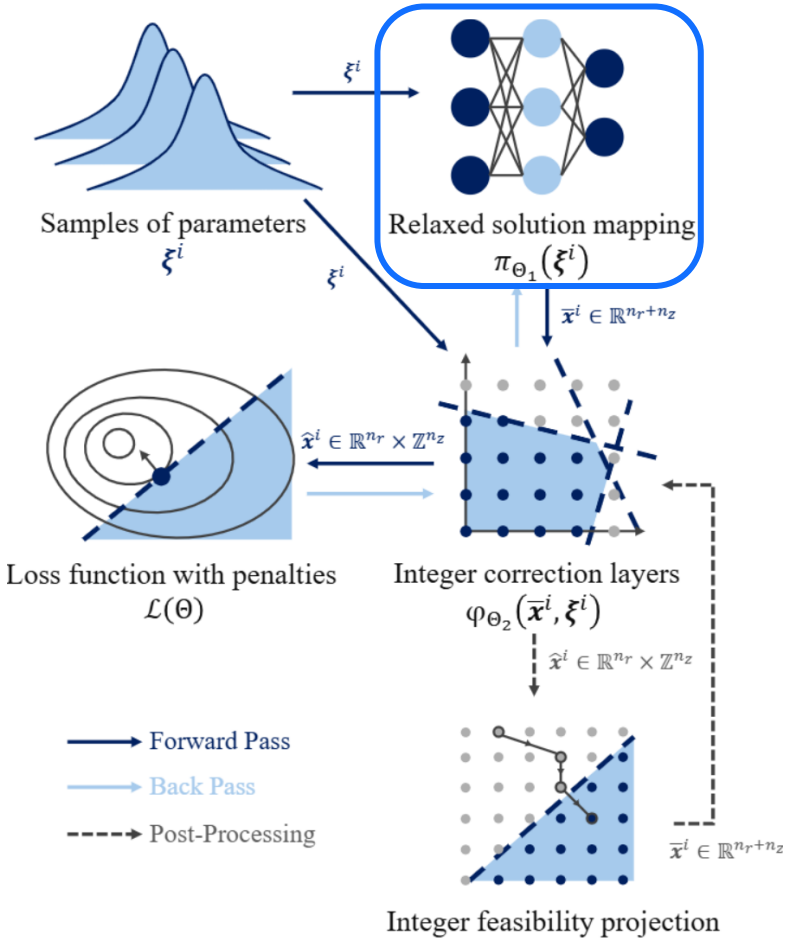
Penalty loss function reformulation:

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

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

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

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

$$\min_{\Theta} \quad \mathbb{E}\big[f(\hat{\mathbf{x}}, \boldsymbol{\xi})\big] \approx \frac{1}{m}\sum_{i=1}^{m} f(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i)$$

$$\text{s.t.} \quad \mathbf{g}(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i) \le 0,$$

$$\hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \boldsymbol{\psi}_{\Theta}(\boldsymbol{\xi}^i), \quad \forall i \in [m].$$

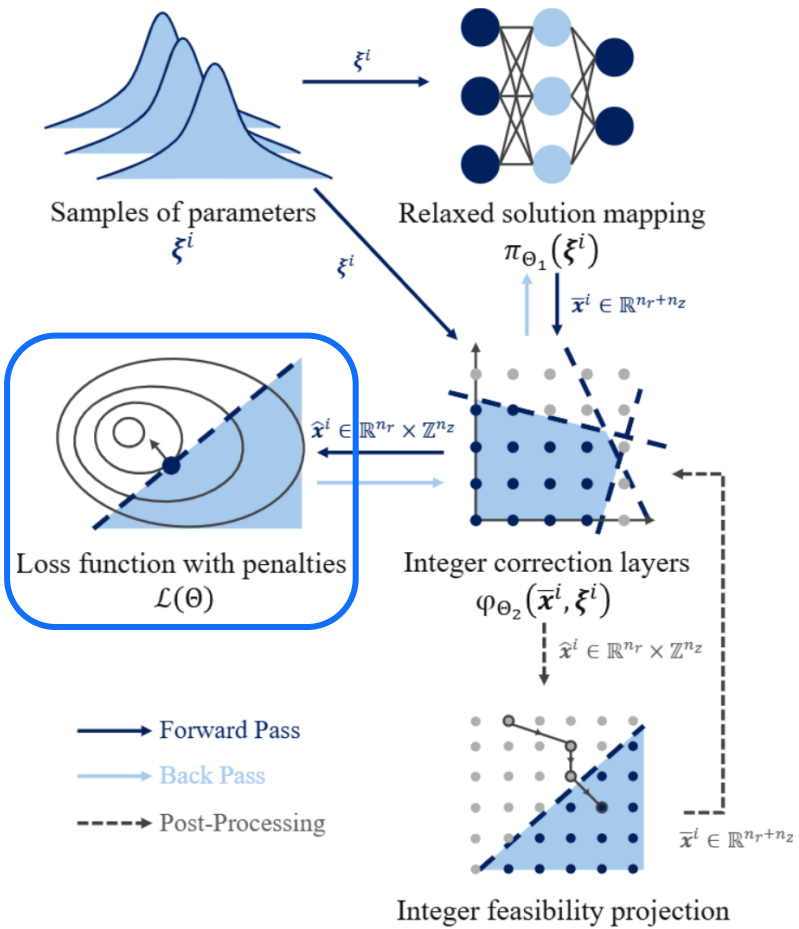
Penalty loss function reformulation:

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

We aim to predict the solution $\hat{\boldsymbol{x}}$ with machine learning model $\psi_{\boldsymbol{\Theta}}$ with inputs $\boldsymbol{\xi}$ and weights $\boldsymbol{\Theta}$.

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

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

$$\min_{\Theta} \quad \mathbb{E}\big[f(\hat{\mathbf{x}}, \boldsymbol{\xi})\big] \approx \frac{1}{m}\sum_{i=1}^{m} f(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i)$$

$$\text{s.t.} \quad \mathbf{g}(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i) \leq 0,$$

$$\hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \boldsymbol{\psi}_{\Theta}(\boldsymbol{\xi}^i), \quad \forall i \in [m].$$

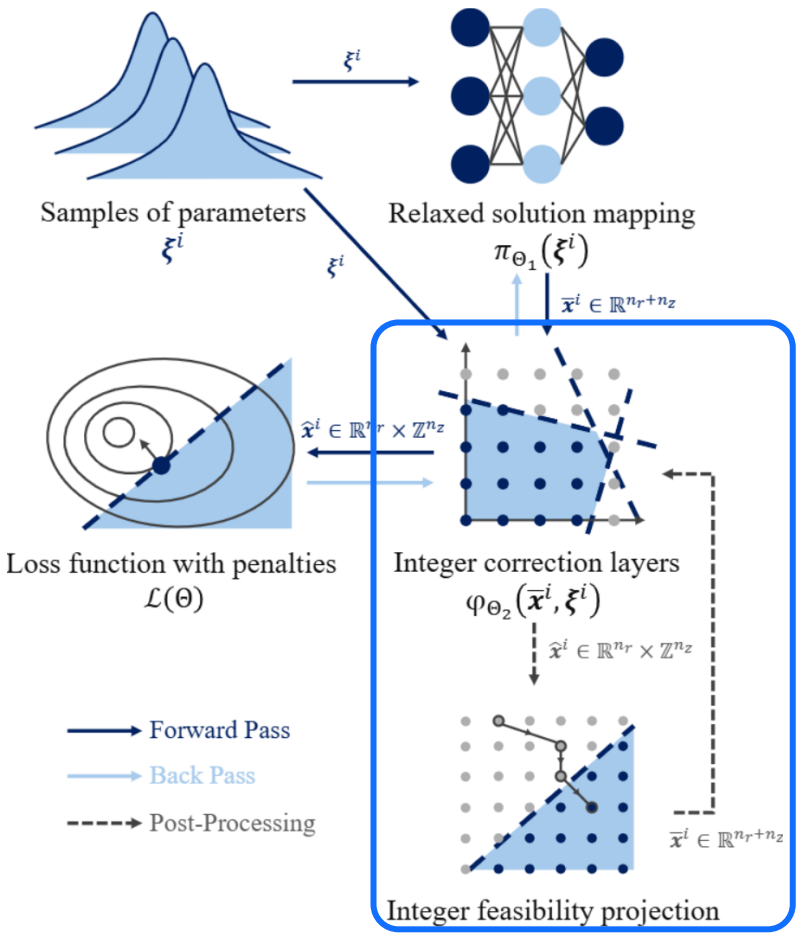
Penalty loss function reformulation:

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

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

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

Learning to Optimize for Mixed-Integer Nonlinear Programming



Learning problem formulation:

$$\min_{\Theta} \quad \mathbb{E}\big[f(\hat{\mathbf{x}}, \boldsymbol{\xi})\big] \approx \frac{1}{m}\sum_{i=1}^{m} f(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i)$$

$$\text{s.t.} \quad \mathbf{g}(\hat{\mathbf{x}}^i, \boldsymbol{\xi}^i) \le 0,$$

$$\hat{\mathbf{x}}^i \in \mathbb{R}^{n_r} \times \mathbb{Z}^{n_z}, \quad \hat{\mathbf{x}}^i = \boldsymbol{\psi}_{\Theta}(\boldsymbol{\xi}^i), \quad \forall i \in [m].$$

Penalty loss function reformulation:

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

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

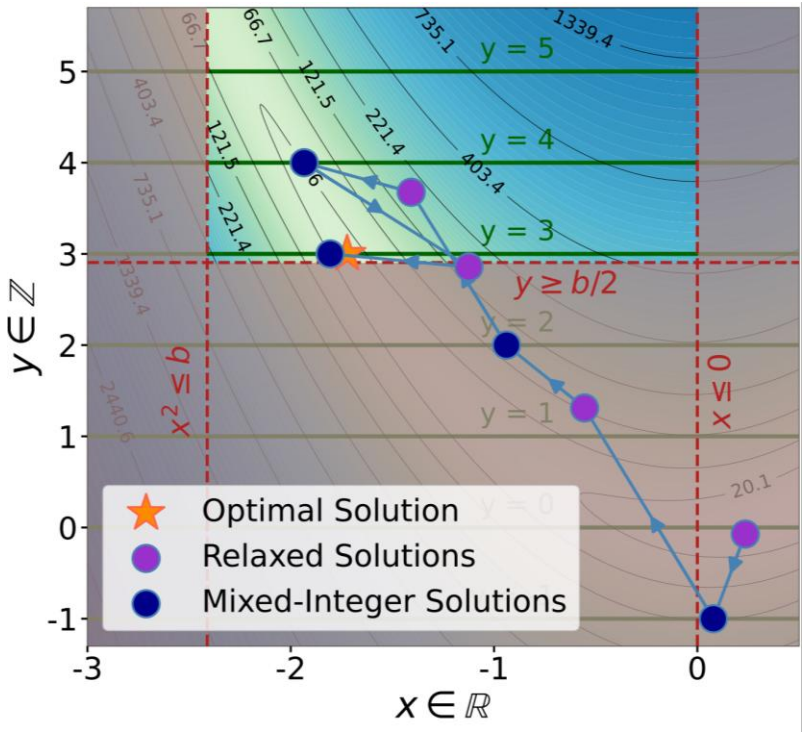
Bo Tang, Elias B. Khalil, Ján Drgoňa, Learning to Optimize for Mixed-Integer Non-linear Programming, arXiv:2410.11061, 2024

Differentiable Integer Correction Layers

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

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

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



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

Differentiable Integer Correction Layers: RC Example

$$\min_{x\in\mathbb{R},y\in\mathbb{Z}} (a-x)^2 + 50(y-x^2)^2$$
$$s.t.\ y \geq \frac{1}{2}b, x^2 \leq b, x \leq 0, y \geq 0$$

Input: $a = 3.83, b = 6.04$

Relaxed Solution Mapping $\pi_{\Theta_1}(a, b)$

Relaxed Solution: $\bar{x} = -1.17, \bar{y} = 2.98$

Rounding Classification $\varphi_{\Theta_1}(a, b, \bar{x}, \bar{y})$

Hidden State: $h_x = -0.68, h_y = 9.49$

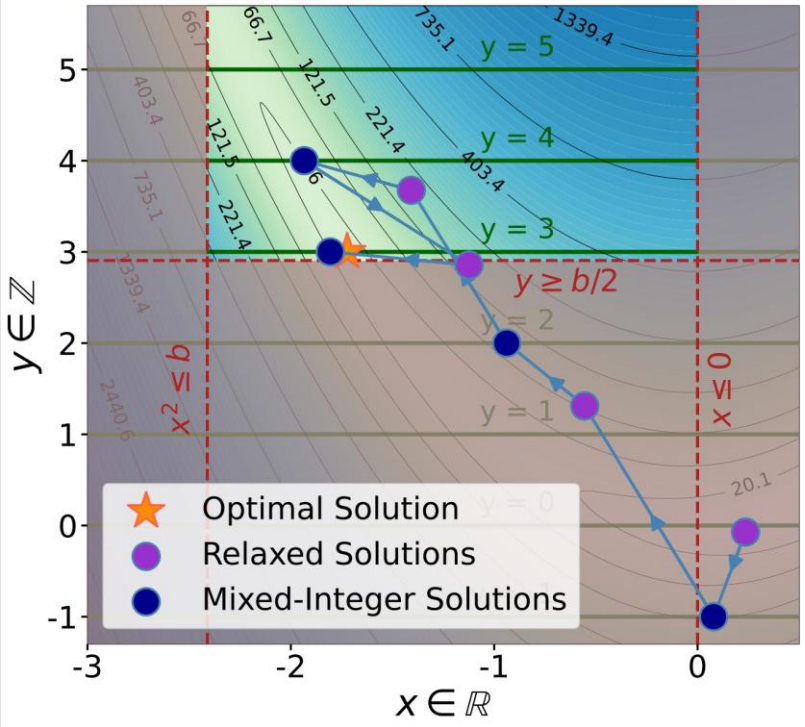
Neural Network $\delta_{\Theta_2}(a, b, \bar{x}, \bar{y})$

Update Continuous Var: $\hat{x} = \bar{x} + h_x = -1.85$

Round Integer Var: $\text{Gumbel\_Sigmoid}(h_y) \geq 0.5 \rightarrow \hat{y} = \lceil \bar{y} \rceil = 3$

Mixed-Integer Solution: $\hat{x} = -1.85, \hat{y} = 3$

Loss Function: $\mathcal{L}_{Obj} + \lambda \cdot \mathcal{L}_{Viol}$



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

Differentiable Integer Correction Layers: LT Example

$$\min_{x\in\mathbb{R},y\in\mathbb{Z}} (a-x)^2 + 50(y-x^2)^2$$
$$s.t.\ y \geq \frac{1}{2}b, x^2 \leq b, x \leq 0, y \geq 0$$

Input: $a = 3.83, b = 6.04$

Relaxed Solution Mapping $\pi_{\Theta_1}(a, b)$

Learnable Threshold $\varphi_{\Theta_1}(a, b, \bar{x}, \bar{y})$

Relaxed Solution: $\bar{x} = -1.14, \bar{y} = 3.09$

Hidden State: $h_x = 0.14, h_y = -1.84$

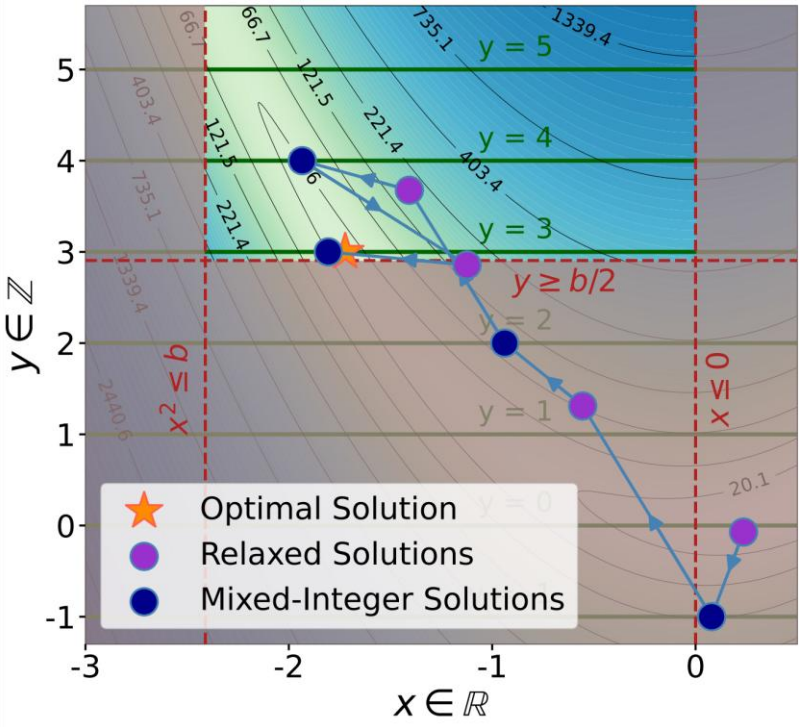
Neural Network $\delta_{\Theta_2}(a, b, \bar{x}, \bar{y})$

Update Continuous Var: $\hat{x} = \bar{x} + h_x = -1.83$

Round Integer Var: $\text{Sigmoid}(\bar{y} - \lfloor\bar{y}\rfloor - h_y) > 0.5 \rightarrow \hat{y} = \lfloor\bar{y}\rfloor = 3$

Mixed-Integer Solution: $\hat{x} = -1.83, \hat{y} = 3$

Loss Function: $\mathcal{L}_{Obj} + \lambda \cdot \mathcal{L}_{Viol}$



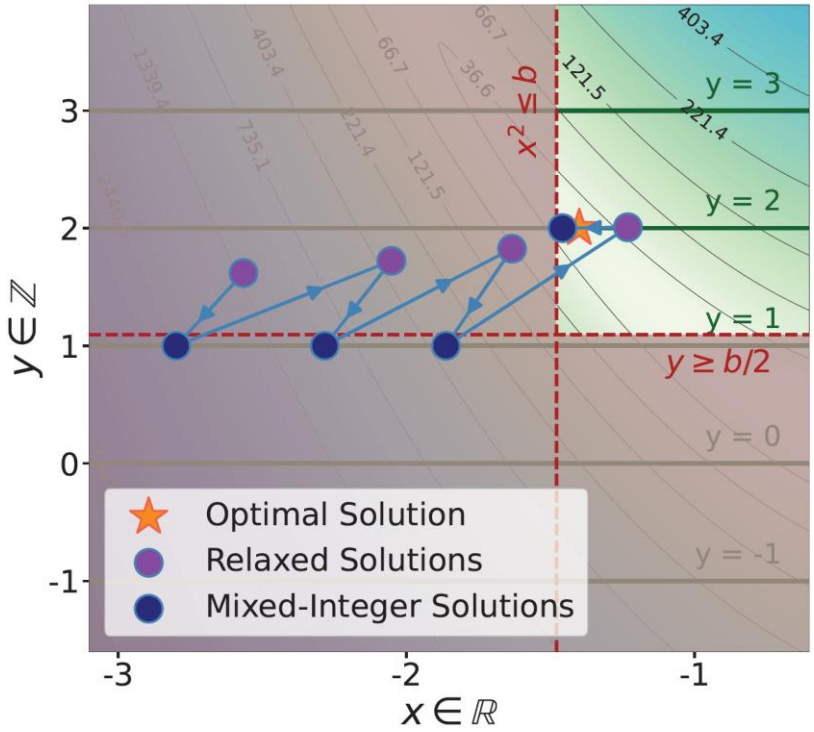
Bo Tang, Elias B. Khalil, Ján Drgoňa, Learning to Optimize for Mixed-Integer Non-linear Programming, arXiv:2410.11061, 2024

Integer Feasibility Projection

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

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

Input: initial relaxed solution $\bar{\mathbf{x}}$, parameters $\boldsymbol{\xi}$, integer correction layer $\varphi_{\Theta_2}(\cdot)$, and step size η
while True **do**
  Obtain updated integers $\hat{\mathbf{x}} \leftarrow \varphi_{\Theta_2}(\bar{\mathbf{x}}, \boldsymbol{\xi})$
  Compute violations $\mathcal{V}(\hat{\mathbf{x}}, \boldsymbol{\xi}) \leftarrow \|\mathbf{g}(\hat{\mathbf{x}}, \boldsymbol{\xi})_+\|_1$
  if $\mathcal{V}(\hat{\mathbf{x}}, \boldsymbol{\xi}) = 0$ **then**
    Break
  else
    Compute gradients $\mathbf{d} \leftarrow \nabla_{\bar{\mathbf{x}}} \mathcal{V}(\hat{\mathbf{x}}, \boldsymbol{\xi})$
    Update relaxed solution $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{x}} - \eta \mathbf{d}$
  end if
end while
Output: a mixed-integer solution $\hat{\mathbf{x}}$



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

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* $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^{n_c}$ *and* $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ *are continuously differentiable with gradients* $\nabla\mathbf{g}$, $\nabla\varphi$ *with Lipschitz constants* $L_{\mathbf{g}}$, L_φ *and bounded Jacobians* $\|\nabla\mathbf{g}\| \le G_{\mathbf{g}}$, $\|\nabla\varphi\| \le G_\varphi$. *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* $|I_{\mathbf{x}}| := |\{j : g_j(\varphi(\mathbf{x})) > 0\}| \le \bar{n}_c$. *Then for step size* $\eta \in \left(0, \frac{1}{L}\right]$, *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* $\mathcal{V}(\mathbf{x}^{(k)}) \to \mathcal{V}^\star \ge 0$, *and* $\lim_{k\to\infty} \|\nabla\mathcal{V}(\mathbf{x}^{(k)})\| = 0$.

(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* $\lim_{k\to\infty} \mathcal{V}(\mathbf{x}^{(k)}) = 0$.

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

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* $\eta \in \left(0, \frac{1}{L}\right]$, *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* $\mathcal{D} := \{x : \mathcal{V}(\mathbf{x}) > 0\}$. *Then for any number of iterations* $K \geq 1$, *the minimum gradient norm over the first* K *iterates satisfies*

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

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

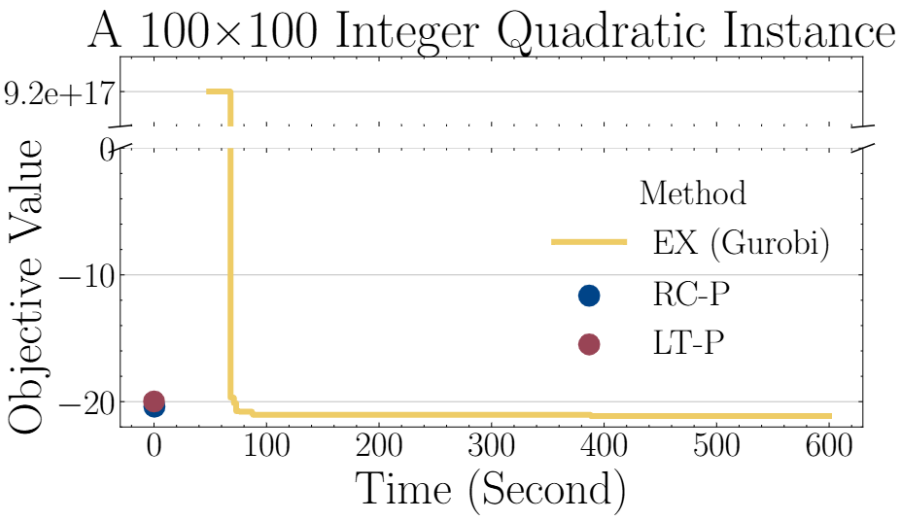
Bo Tang, Elias B. Khalil, Ján Drgoňa, Learning to Optimize for Mixed-Integer Non-linear Programming, arXiv:2410.11061, 2024

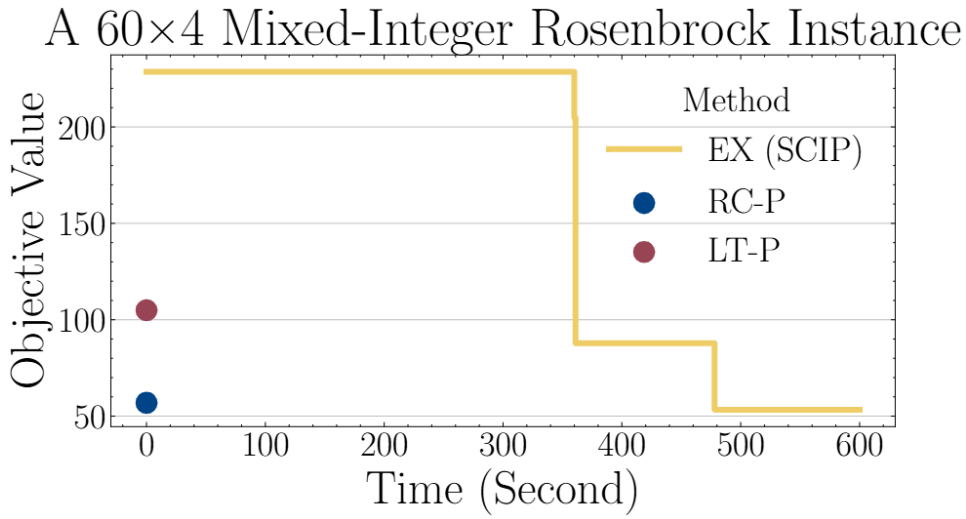
Benchmark Methods

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.

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

L2O for MINLP: Comparison with SOTA Solvers





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

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

L2O for MINLP: Empirical Evaluation

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

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

Method	RC				RC-P				LT			
Metric	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
100×100	-13.54	-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	-	-	-	-	-	-	-	-

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

L2O for MINLP: Empirical Evaluation

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

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

Method	RC				RC-P				LT			
Metric	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)	Obj Mean	Obj Median	% Feasible	Time (Sec)
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 (Sec)
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	-	-	-	-	-	-	-	-

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

L2O for MINLP: Empirical Evaluation

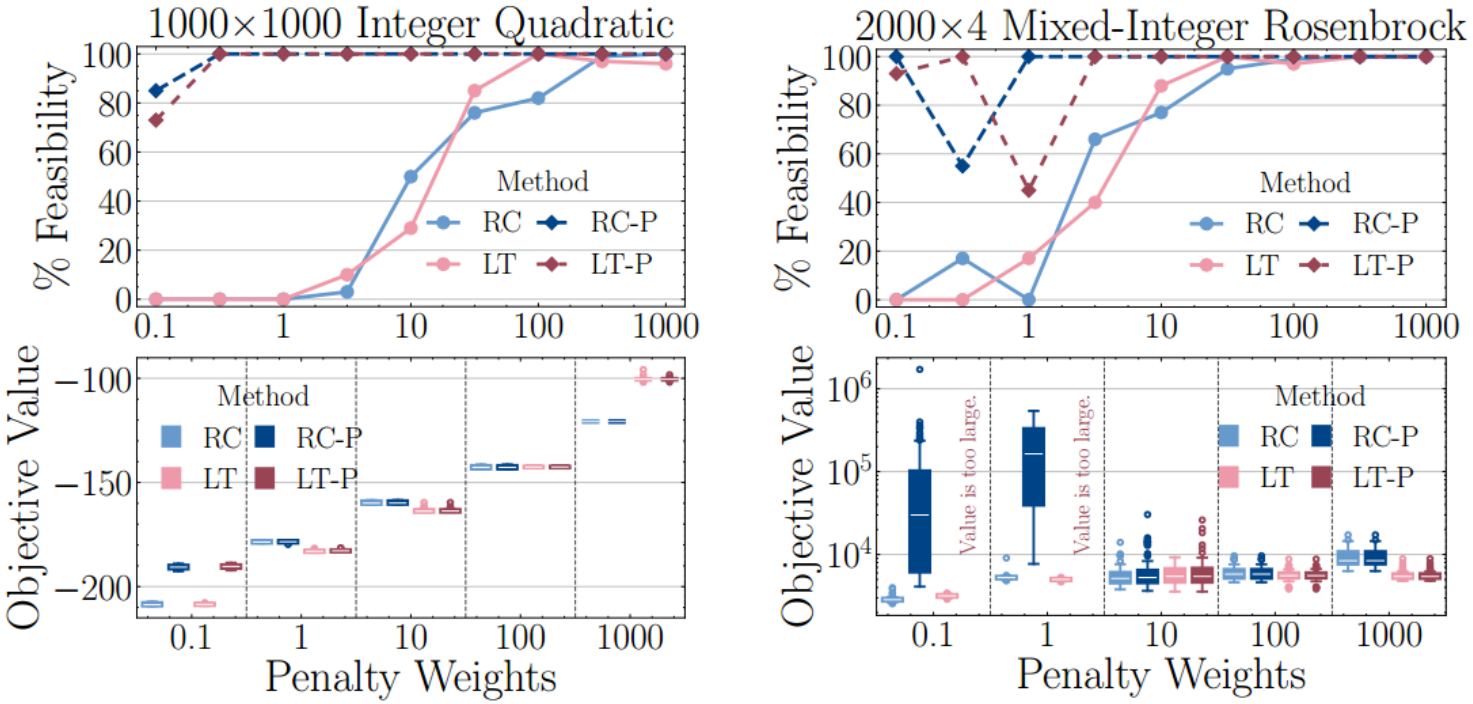
$$\min_{\mathbf{x}\in\mathbb{R}^n,\mathbf{y}\in\mathbb{Z}^n} \|\mathbf{a}-\mathbf{x}\|_2^2 + 50\|\mathbf{y}-\mathbf{x}^2\|_2^2 \quad \text{subject to } \|\mathbf{x}\|_2^2 \le nb,\ \mathbf{1}^\intercal\mathbf{y} \ge \frac{nb}{2},\ \mathbf{p}^\intercal\mathbf{x} \le 0,\ \mathbf{Q}^\intercal\mathbf{y} \le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

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

Method	RC				RC-P				LT			
Metric	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

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

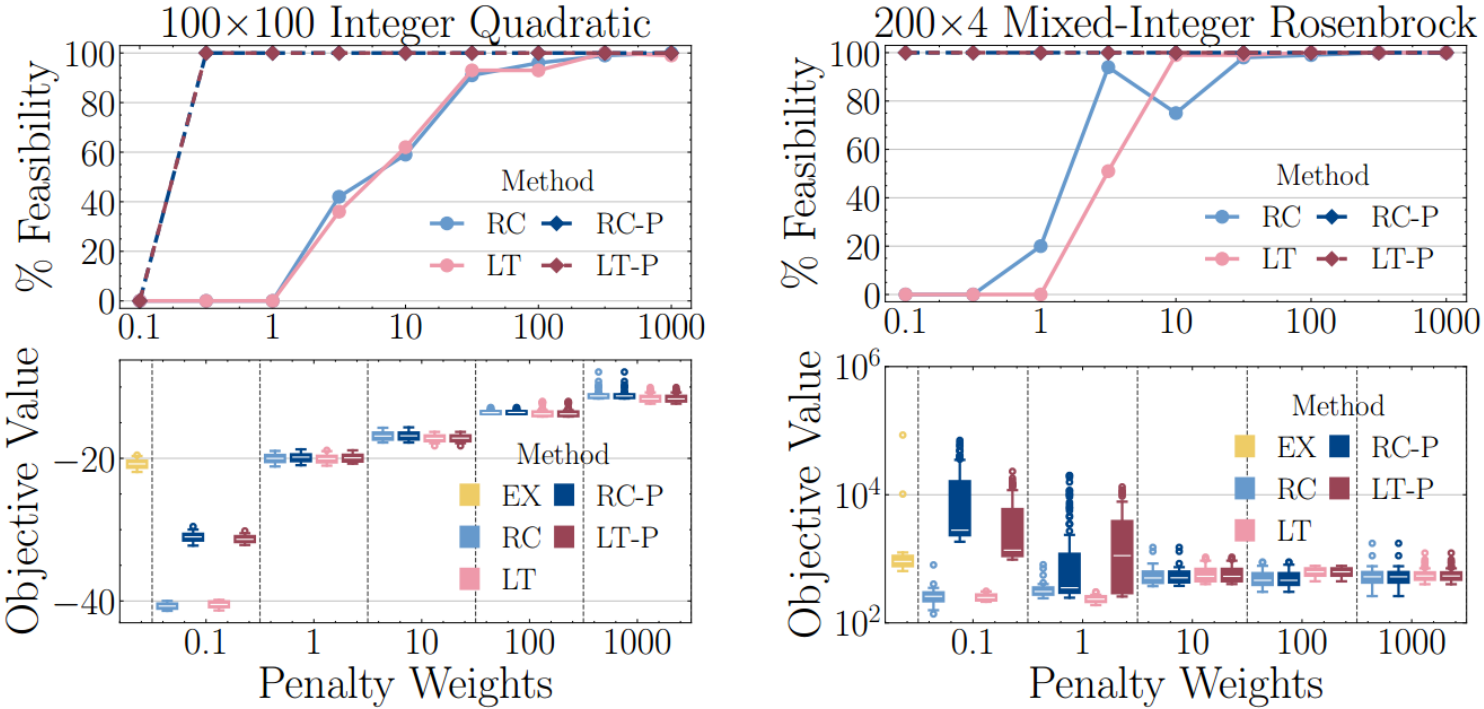
Effect of Penalty Weight



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).

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

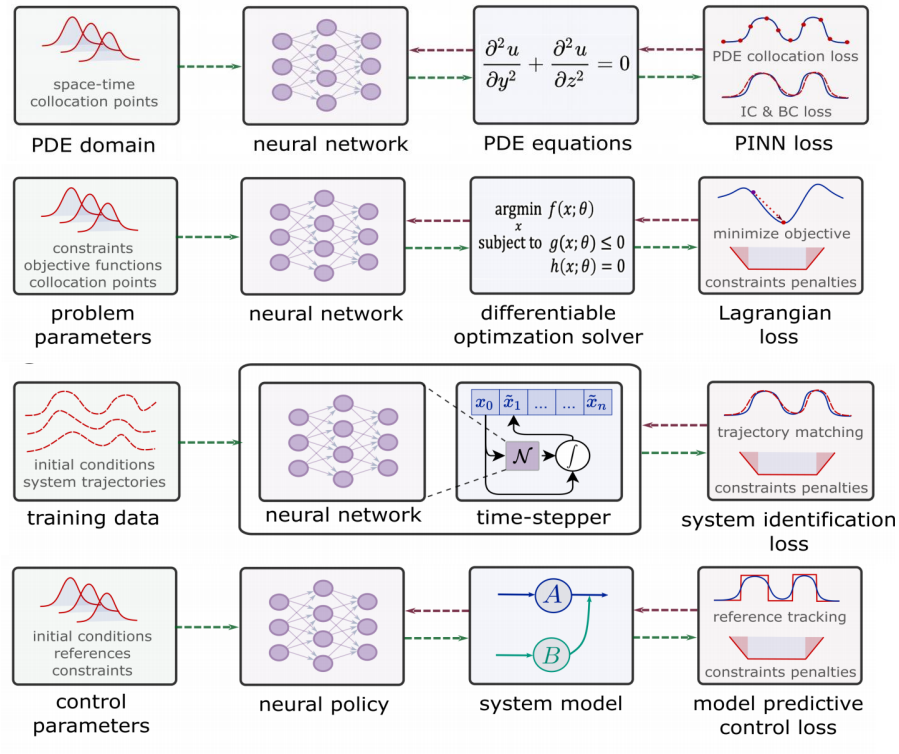
Effect of Penalty Weight



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

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

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

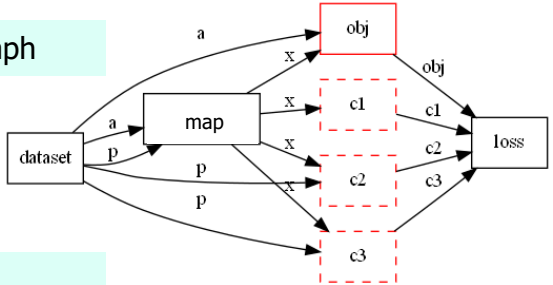
$$\min_{\Theta} (1-\mathbf{x})^2 + \boldsymbol{p}(\mathbf{y}-\mathbf{x}^2)^2$$
$$\text{s.t. } (\boldsymbol{p}/2)^2 \leq \mathbf{x}^2 + \mathbf{y}^2 \leq \boldsymbol{p}^2, \;\; \mathbf{x} \geq \mathbf{y}$$
$$\mathbf{x} = \boldsymbol{\pi}_{\Theta}(\boldsymbol{p})$$

2. Python code interface

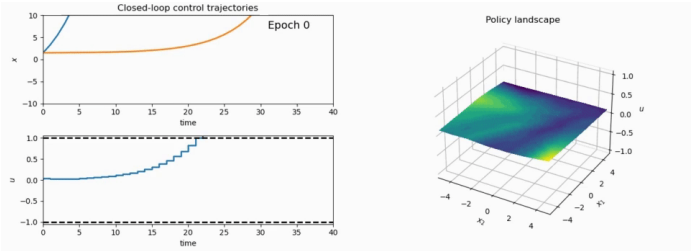
PyTorch

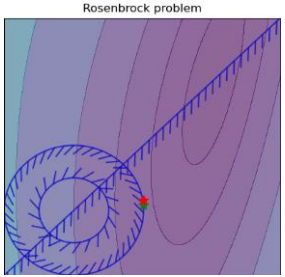
```
import neuromancer as nm

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()
```

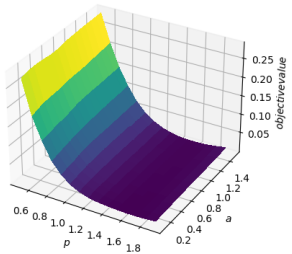
3. Problem graph



4. Results







Summary

- **General learning-to-optimize (L2O) method for parametric MINLPs**
  - Novel differentiable Integer Correction Layers enabling neural networks to predict hard integer values
  - Novel Integer Feasibility Projection scheme for mixed integer inequality constraints
  - Theoretical feasibility guarantees
  - Experimental results scaling to tens of thousands of decision variables
  - Open-source code

Energy at Hopkins

Paper

GitHub

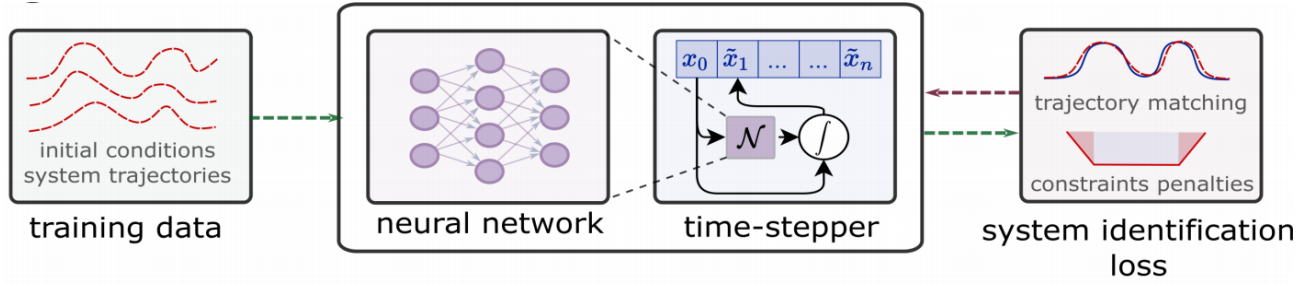
https://github.com/pnnl/L2O-pMINLP

drgona.github.io

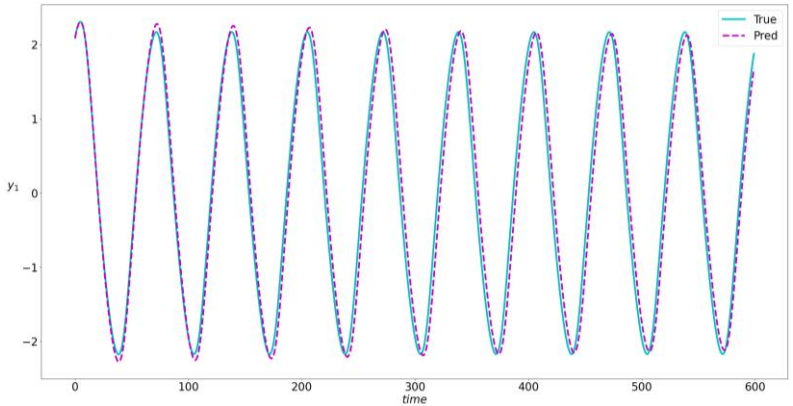
jdrgona1@jh.edu

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

Learning to Model (L2M) Dynamical Systems



Neural models for nonlinear system identification



Application: climate modeling

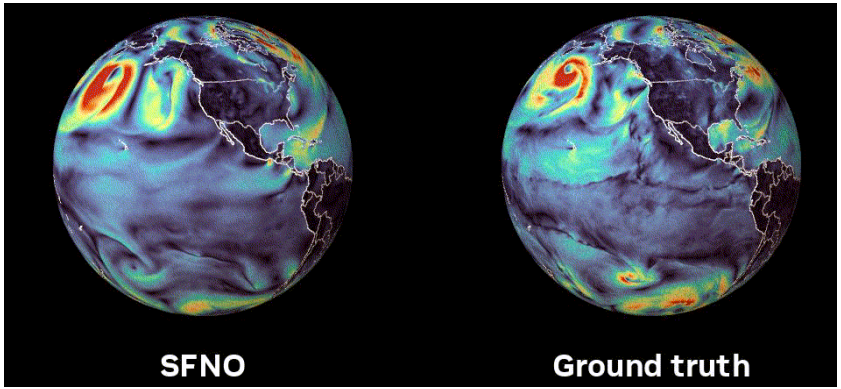
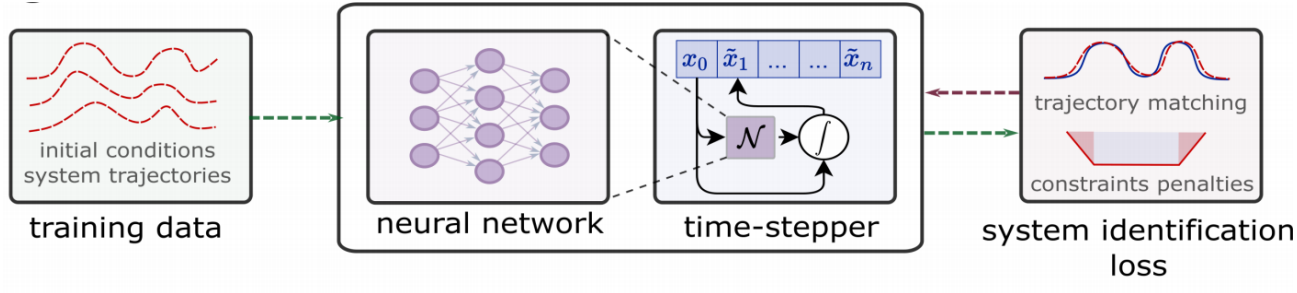


Image: NVIDIA FourCastNet

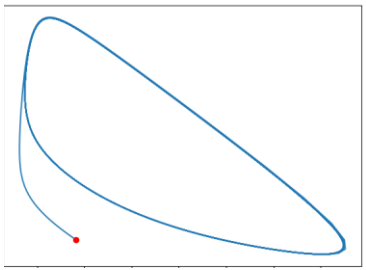
R. T. Q. Chen, et al., *Neural ordinary differential equations. NeurIPS, 2018*
C. Rackauckas, et al., *Universal Differential Equations for Scientific Machine Learning*, 2021

Learning to Model (L2M) Dynamical Systems



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

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



Architecture: differentiable ODE solver with neural

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

Architecture: Koopman operator with neural network basis funct

$$y_k = NN_\theta(x_k)$$
$$y_{k+1} = K_\theta(y_k)$$
$$x_{k+1} = NN_\theta^{-1}(y_{k+1})$$

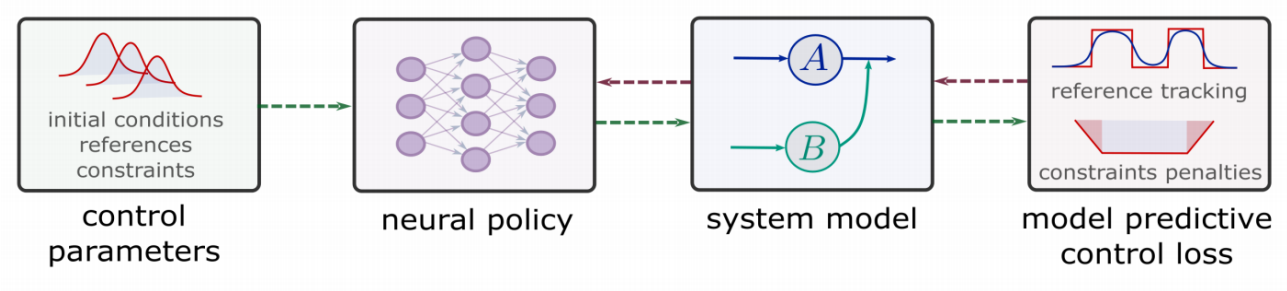
Loss function: trajectory matching, regularizations, and constraints penalties.

$$\ell_1 = \sum_{i=1}^{m}\sum_{k=1}^{N} Q_x ||x_k^i - \hat{x}_k^i||_2^2$$
$$\ell_2 = \sum_{i=1}^{m}\sum_{k=1}^{N-1} Q_{dx} ||\Delta x_k^i - \Delta \hat{x}_k^i||_2^2$$
$$\ell_{L2M} = \ell_1 + \ell_2$$

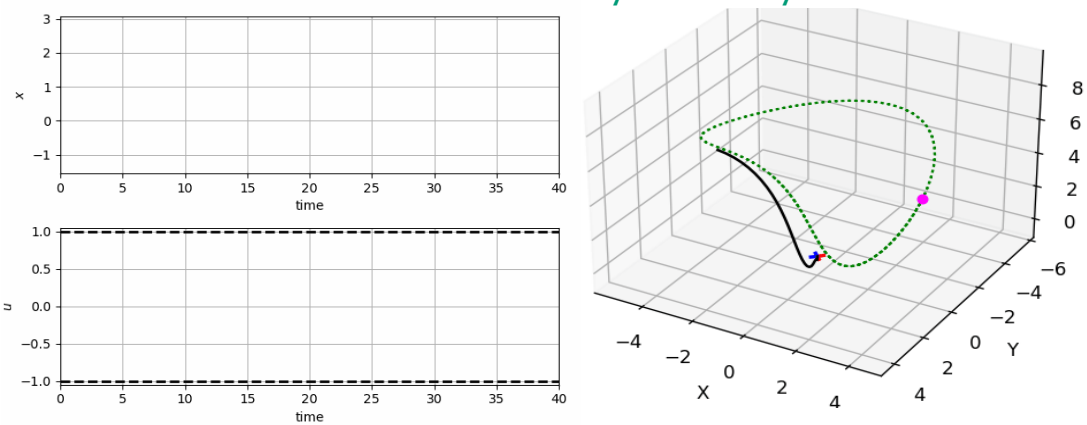
https://github.com/pnnl/neuromancer/blob/master/examples/ODEs/Part_1_NODE.ipynb

R. T. Q. Chen, et al., *Neural Ordinary Differential Equations*, 2019
B. Lusch, et al., *Deep learning for universal linear embeddings of nonlinear dynamics*, 2018

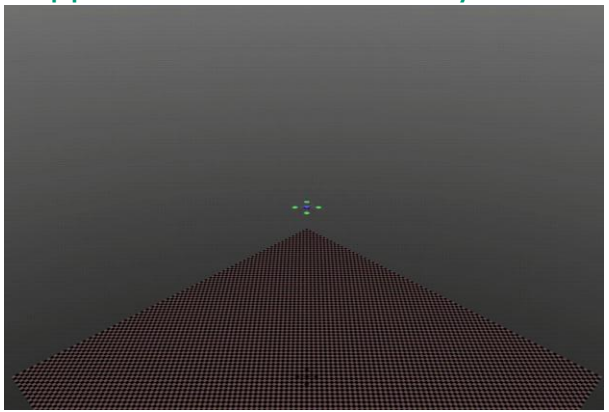
Learning to Control (L2C) with Differentiable System Models



Trajectory optimization for dynamical systems

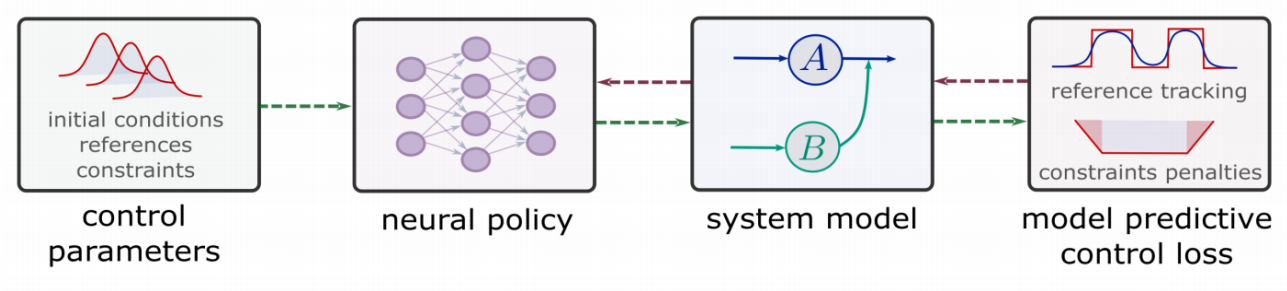


Application: autonomous systems

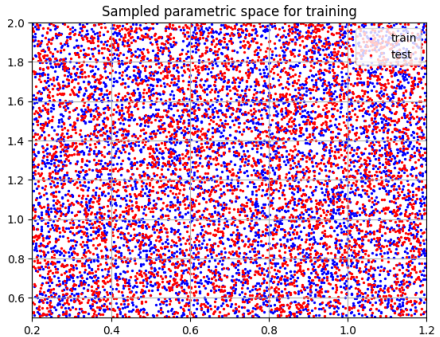


J. Drgoňa, A. Tuor and D. Vrabie, "Learning Constrained Parametric Differentiable Predictive Control Policies With Guarantees," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2024

Learning to Control (L2C) with Differentiable System Models



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.

$$\ell_1 = \sum_{i=1}^{m} \sum_{k=1}^{N-1} Q_x ||x_k^i - r_k^i||_2^2$$
$$\ell_2 = \sum_{i=1}^{m} \sum_{k=1}^{N-1} Q_g ||\texttt{RELU}(g(x_k^i, u_k^i, \xi_k^i)||_2^2$$
$$\ell_{L2C} = \ell_1 + \ell_2$$

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