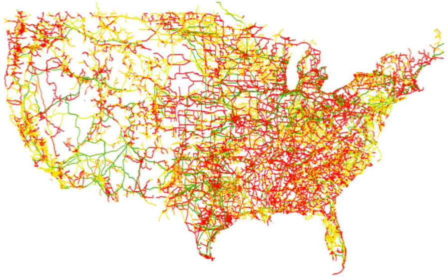
JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Scientific Machine Learning for Modeling, Optimization, and Control of Energy Systems

Ján Drgoňa
Associate Professor
Civil and Systems Engineering Department
Electrical and Computer Engineering Department (secondary)
The Ralph O'Connor Sustainable Energy Institute (ROSEI)
Data Science and AI Institute (DSAI)

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



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.

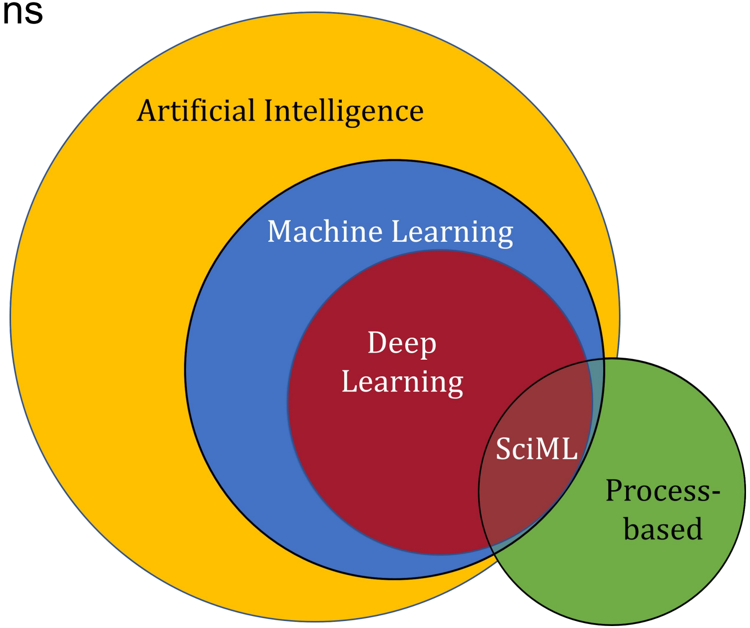
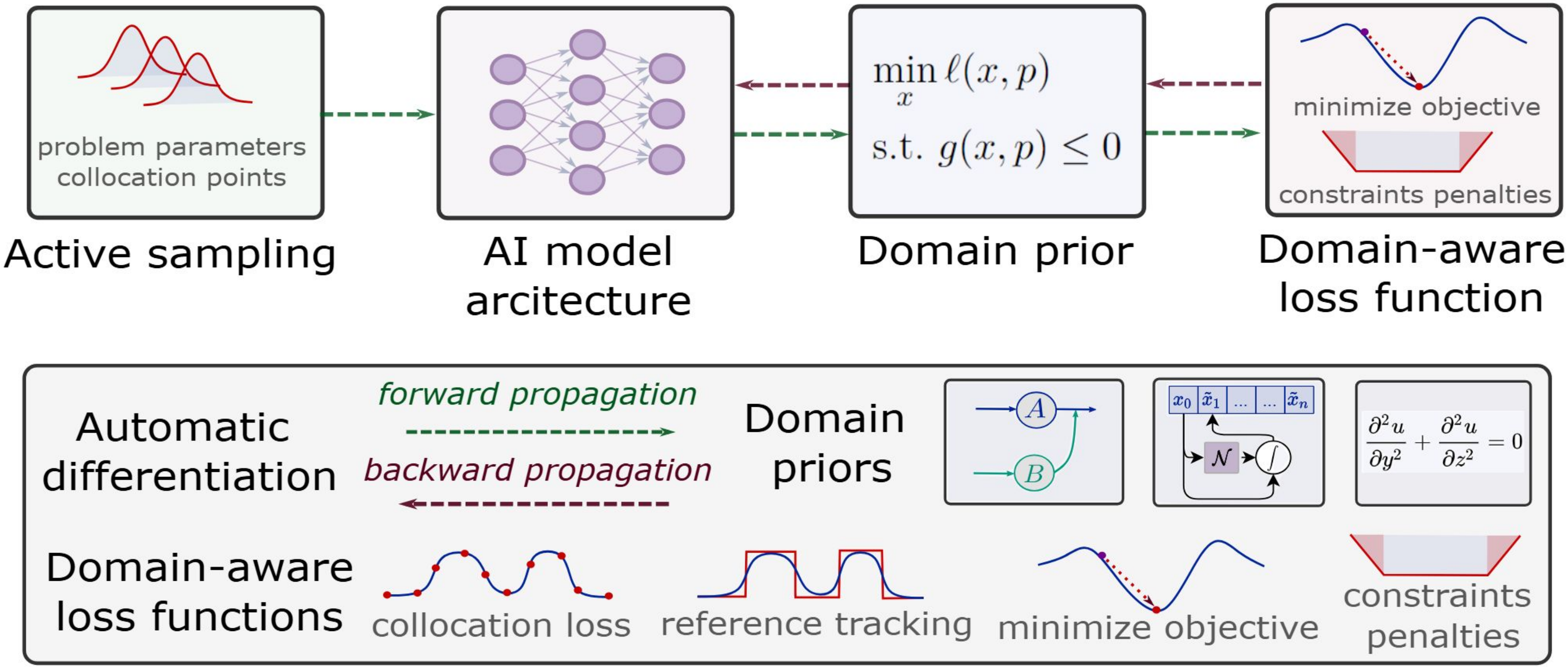


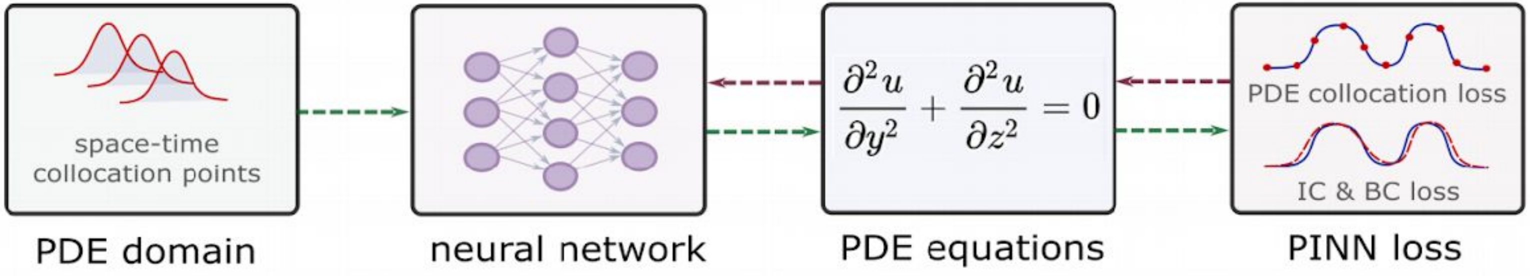
Image source: https://sciml.wur.nl/reviews/sciml/sciml.html

Components of Scientific Machine Learning

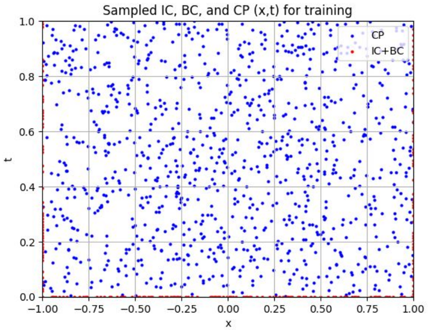


Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 2021.
Thiyagalingam, J., Shankar, M., Fox, G. et al. Scientific machine learning benchmarks. Nature Reviews Physics 4, 413–420, 2022.
Nghiem T., Drgona J., et al. Physics-Informed Machine Learning for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

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



Dataset: collocation points in the spatio-temporal coordinates.



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

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

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

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

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

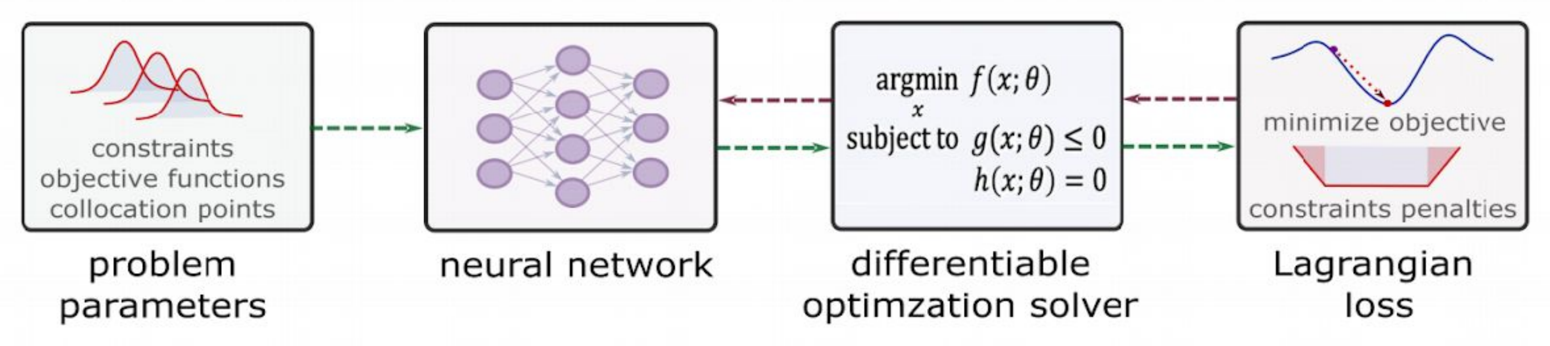
$$\ell_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |y(t_u^i, x_u^i) - NN_\theta(t_u^i, x_u^i)|^2$$

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

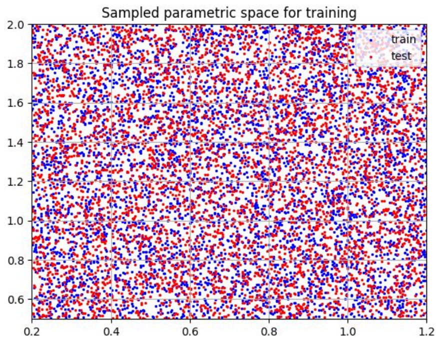
https://github.com/pnnl/neuromancer/blob/master/examples/PDEs/Part_2_PINN_BurgersEquation.ipynb

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



Dataset: collocation points in the parametric space.



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

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

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

Loss function: minimizing objective function and constraints penalties.

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

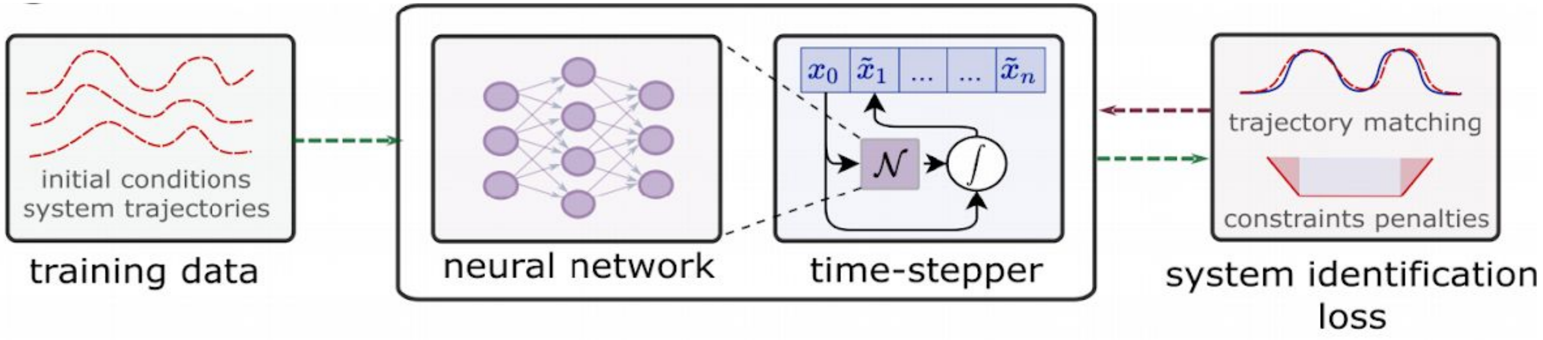
$$\ell_g = \frac{1}{m}\sum_{i=1}^{m} |\text{RELU}(g(x^i, \xi^i))|^2$$

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

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

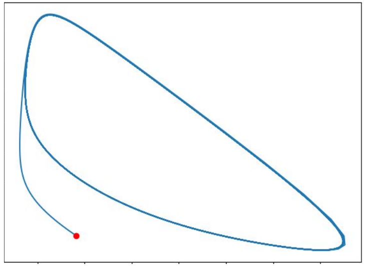
A. Agrawal, et al., *Differentiable Convex Optimization Layers*, 2019
P. Donti, et al., *DC3: A learning method for optimization with hard constraints*, 2021

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 network** model.

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

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

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

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

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

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

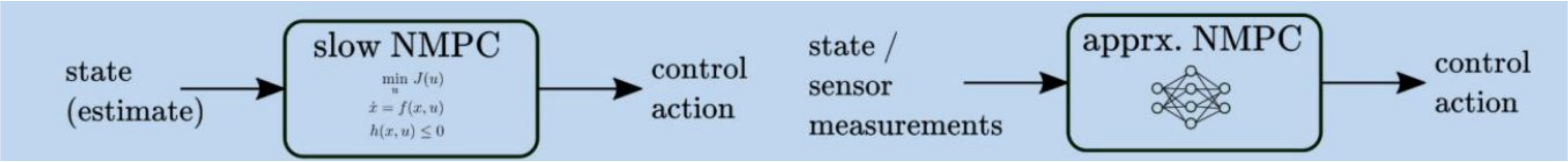
$$\ell_{L2M} = \ell_1 + \ell_2$$

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

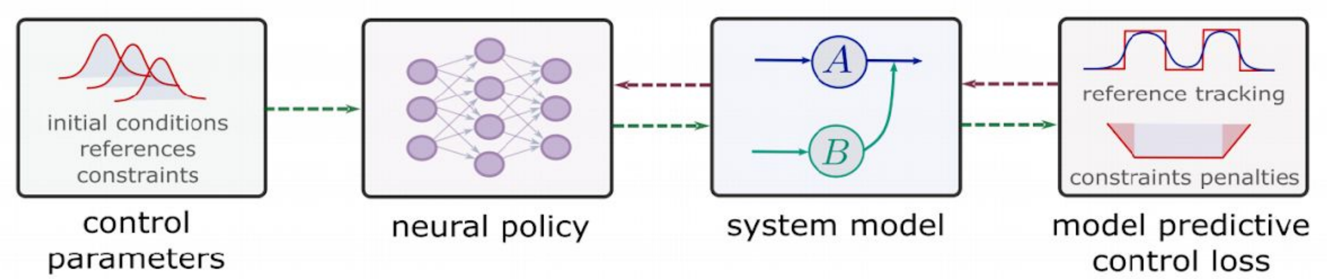
Learning to Control (L2C) Methodologies

Supervised L2C: Approximate Model Predictive Control



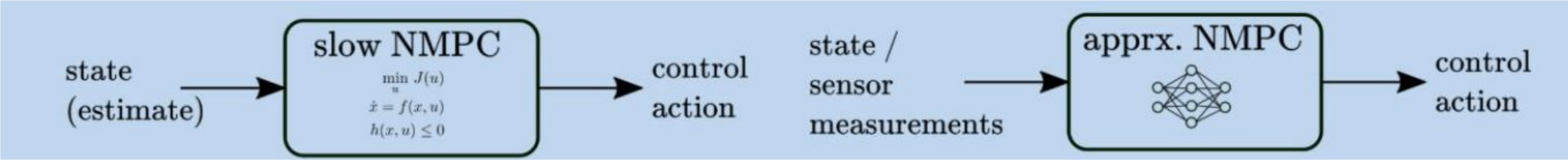
M. Hertneck, et al., "Learning an Approximate Model Predictive Controller With Guarantees," in IEEE Control Systems Letters, 2018
B. Karg and S. Lucia, "Efficient Representation and Approximation of Model Predictive Control Laws via Deep Learning," in IEEE Transactions on Cybernetics, 2020

Self-Supervised L2C: Differentiable Predictive Control (DPC)



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
Ján Drgoňa, et al, Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems, Journal of Process Control, 2022

Supervised L2C: Approximate MPC



Step 1: solve set of MPC problems to generate labeled training data

$$\min_{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}} \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + p_N(\mathbf{x}_N)$$

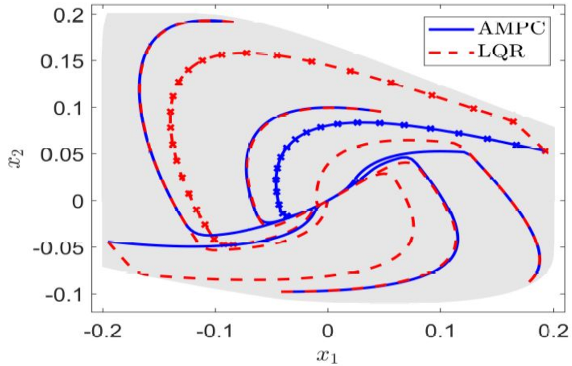
$$\begin{aligned} \text{s.t. } \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \ k \in \mathbb{N}_0^{N-1} \\ h(\mathbf{x}_k) &\leq 0 \\ g(\mathbf{u}_k) &\leq 0 \\ \mathbf{x}_0 &= \mathbf{x}(t) \end{aligned}$$

Step 2: supervised imitation learning to learn approximate MPC policy

Machine Learning

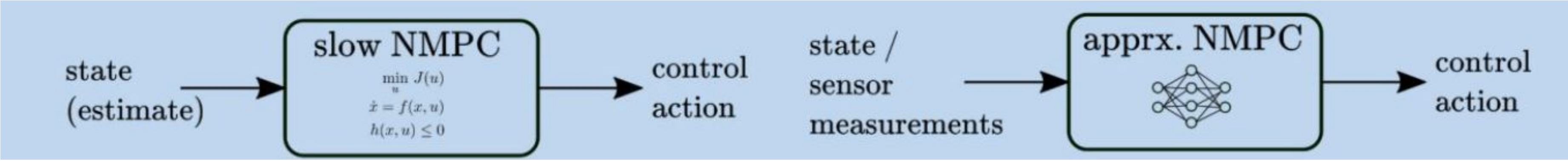
Sample robust MPC

Learn: $\pi_{\text{approx}} \approx \pi_{\text{MPC}}$



M. Hertneck, et al., "Learning an Approximate Model Predictive Controller With Guarantees," in IEEE Control Systems Letters, 2018

Supervised L2C: Approximate MPC



Step 1: solve set of MPC problems to generate labeled training data

$$\min_{\mathbf{u}_0,\ldots,\mathbf{u}_{N-1}} \sum_{k=0}^{N-1} \ell(\mathbf{x}_k, \mathbf{u}_k) + p_N(\mathbf{x}_N)$$

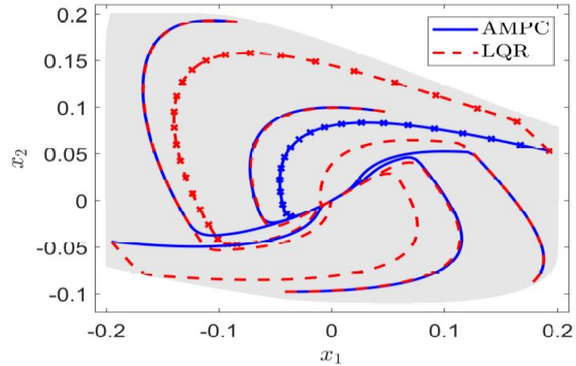
$$\begin{aligned} \text{s.t. } & \mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \;\; k \in \mathbb{N}_0^{N-1} \\ & h(\mathbf{x}_k) \le 0 \\ & g(\mathbf{u}_k) \le 0 \\ & \mathbf{x}_0 = \mathbf{x}(t) \end{aligned}$$

Step 2: supervised imitation learning to learn approximate MPC policy

Machine Learning

Sample robust MPC

Learn: $\pi_{\text{approx}} \approx \pi_{\text{MPC}}$

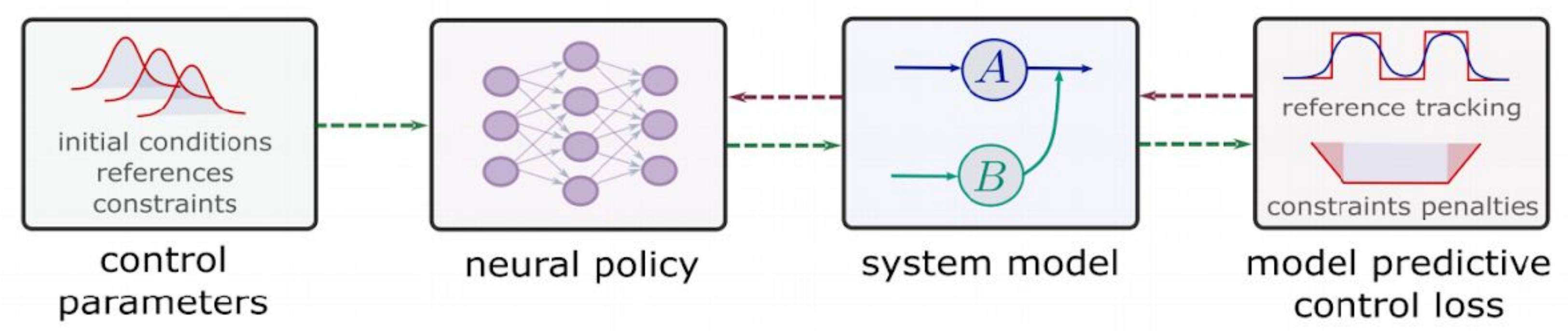


Problem 1: data generation is expensive!

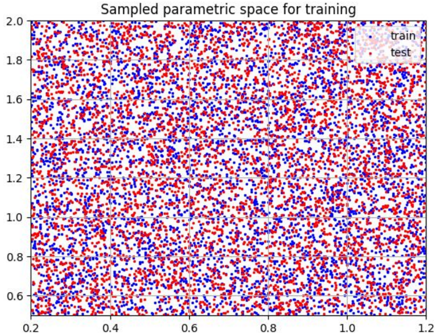
Problem 2: hard to integrate constraints into supervised learning.

M. Hertneck, et al., "Learning an Approximate Model Predictive Controller With Guarantees," in IEEE Control Systems Letters, 2018

Self-Supervised L2C: Differentiable Predictive Control (DPC)



Dataset: collocation points in the control parametric space.



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

$$x_{k+1} = \text{ODESolve}(f(x_k, u_k))$$
$$u_k = NN_\theta(x_k, \xi_k)$$
$$g(x_k, u_k, \xi_k) \le 0$$
$$x_0 \sim \mathcal{P}_{x_0}$$
$$\xi_k \sim \mathcal{P}_\xi$$

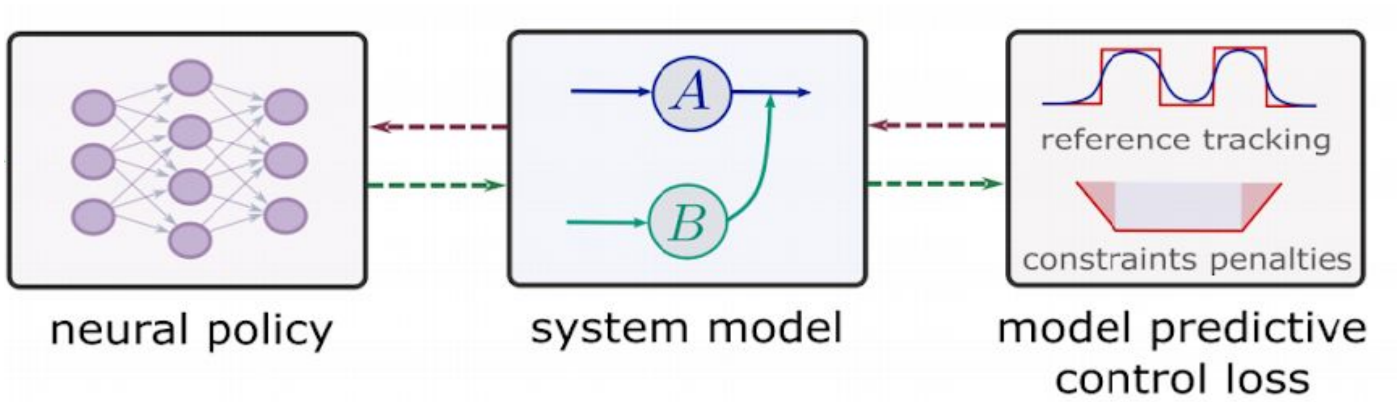
Loss function: reference tracking, constraints and terminal penalties.

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

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

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

Differentiable Closed-Loop System



Forward pass. For a single scenario, the closed-loop recursion and loss are

$$
\begin{aligned}
u_k &= \pi_\theta(x_k; \xi_k), \\
x_{k+1} &= f(x_k, u_k; \xi_k), \\
\ell_k &= \ell(x_k, u_k; \xi_k) \; + \; p_x\big(g(x_k; \xi_k)\big) \; + \; p_u\big(h(u_k; \xi_k)\big), \\
\mathcal{L} &= \frac{1}{N} \sum_{k=0}^{N-1} \ell_k \; + \; \ell_N(x_N; \xi_N).
\end{aligned}
$$

Differentiable Closed-Loop System

Backward pass (sensitivities). Let

$$A_k := \frac{\partial f}{\partial x}, \quad B_k := \frac{\partial f}{\partial u}, \quad P_k := \frac{\partial \pi_\theta}{\partial x}, \quad G_k := \frac{\partial \pi_\theta}{\partial \theta} \quad \text{(all evaluated at } (x_k, u_k; \xi_k)\text{)}.$$

Define $s_N := \frac{\partial \ell_N}{\partial x}(x_N; \xi_N)$ and stage gradients

$$\ell_{x,k} := \frac{\partial \ell}{\partial x} + \frac{\partial p_x}{\partial x}, \qquad \ell_{u,k} := \frac{\partial \ell}{\partial u} + \frac{\partial p_u}{\partial u}.$$

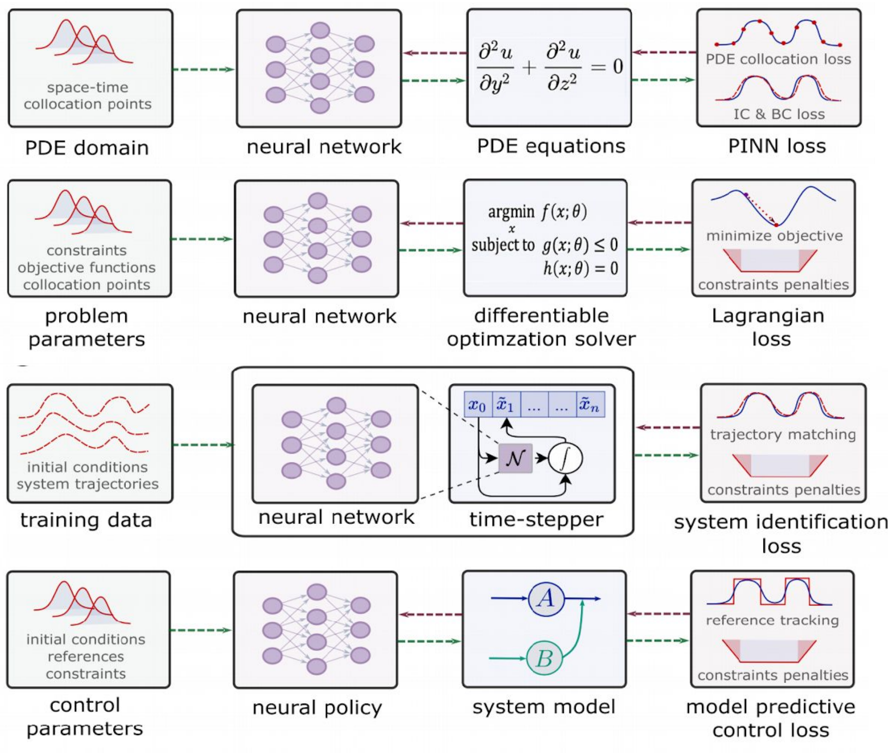
Then the reverse recursion for $k = N-1, \dots, 0$ is

$$s_k \;=\; \ell_{x,k} \;+\; A_k^\top s_{k+1} \;+\; P_k^\top\big(\ell_{u,k} + B_k^\top s_{k+1}\big),$$

and the gradient w.r.t. the policy parameters is

$$\nabla_\theta \mathcal{L} \;=\; \sum_{k=0}^{N-1} G_k^\top \big(\ell_{u,k} + B_k^\top s_{k+1}\big).$$

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

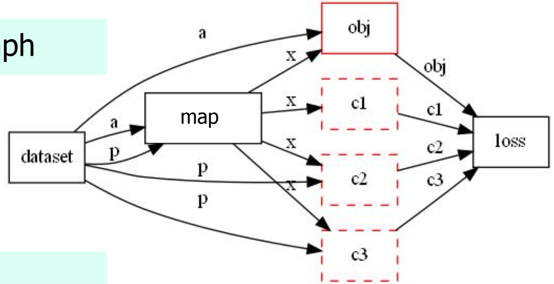
$$\min_{\Theta} \; (1-\mathbf{x})^2 + \boldsymbol{p}(\mathbf{y}-\mathbf{x}^2)^2$$
$$\text{s.t. } (\boldsymbol{p}/2)^2 \leq \mathbf{x}^2 + \mathbf{y}^2 \leq \boldsymbol{p}^2, \;\; \mathbf{x} \geq \mathbf{y}$$
$$\mathbf{x} = \boldsymbol{\pi}_{\Theta}(\boldsymbol{p})$$

2. Python code interface

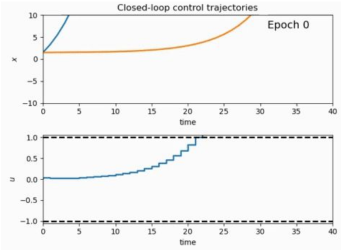
PyTorch

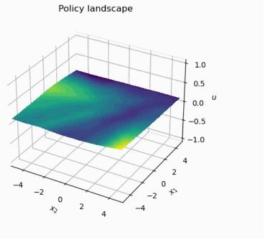
```
import neuromancer as nm

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

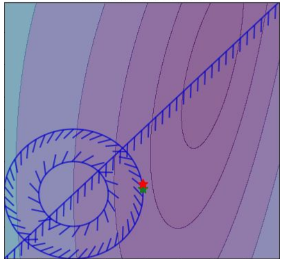


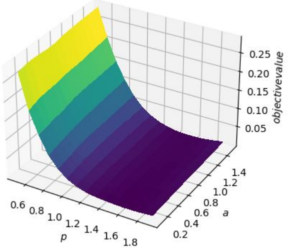
4. Results





Rosenbrock problem





NeuroMANCER Development Team


Aaron Tuor


James Koch


Madelyn Shapiro


Rahul Birmiwal

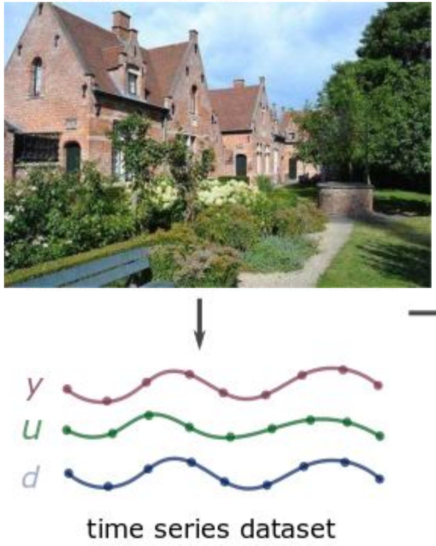

Bruno Jacob

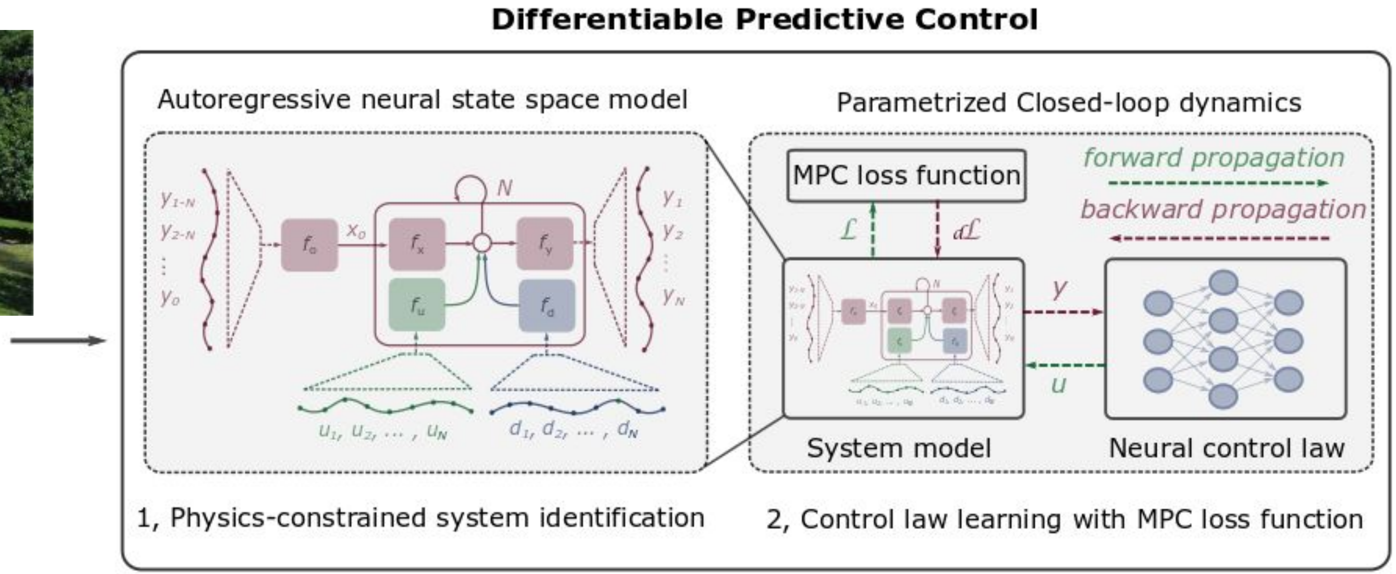

Draguna Vrabie

U.S. DEPARTMENT OF ENERGY

Pacific Northwest NATIONAL LABORATORY

Learning to Control Building Energy System





Benefits of Scientific Machine Learning

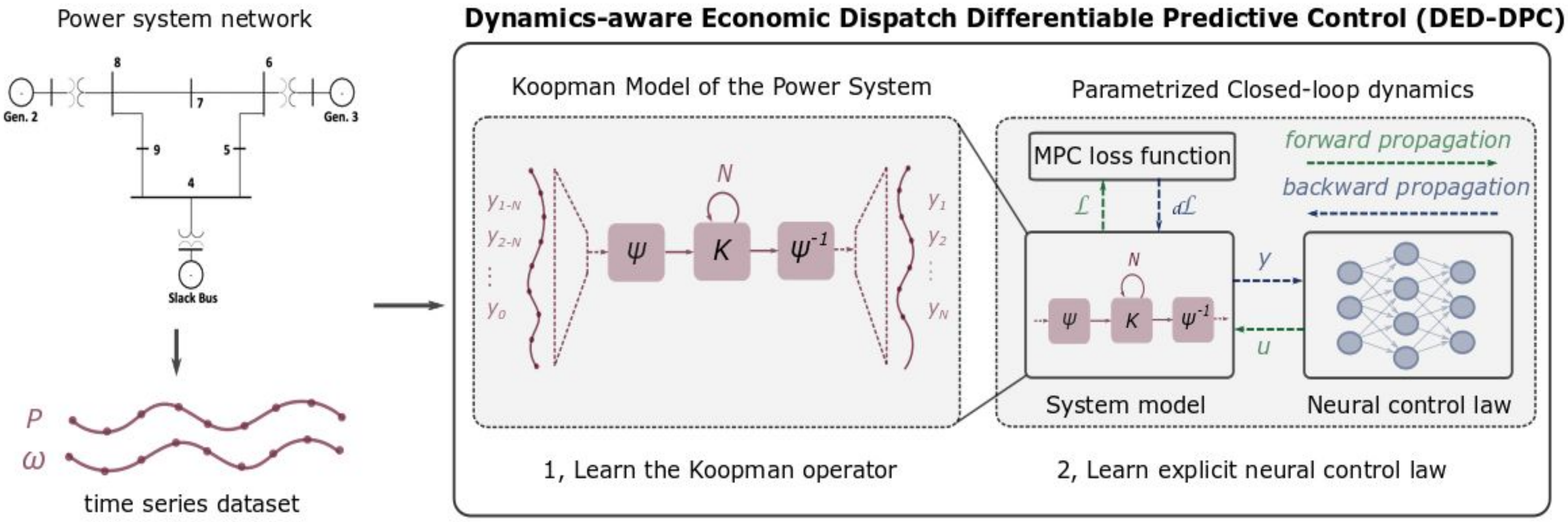
Modeling and optimal control design is roughly **10-times faster** and requires **less expertise**.

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

J. Drgona, et al., Physics-constrained deep learning of multi-zone building thermal dynamics, Energy and Buildings, 2021

J. Drgona, et al., Deep Learning Explicit Differentiable Predictive Control Laws for Buildings, IFAC NMPC 2021

Learning to Control Power System



Benefits of Scientific Machine Learning

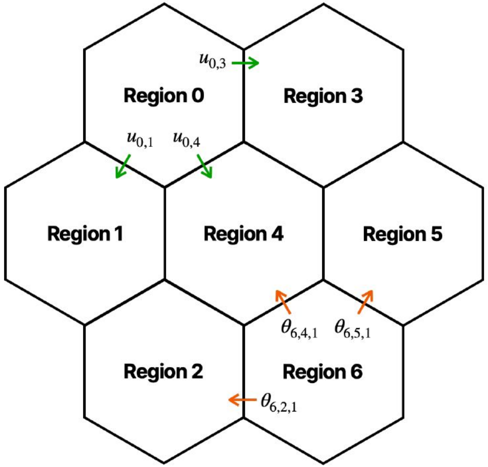
Fast prototyping by re-using code template from building control project.

Real-time decisions are made **5 orders of magnitude faster** than online model predictive control (MPC).

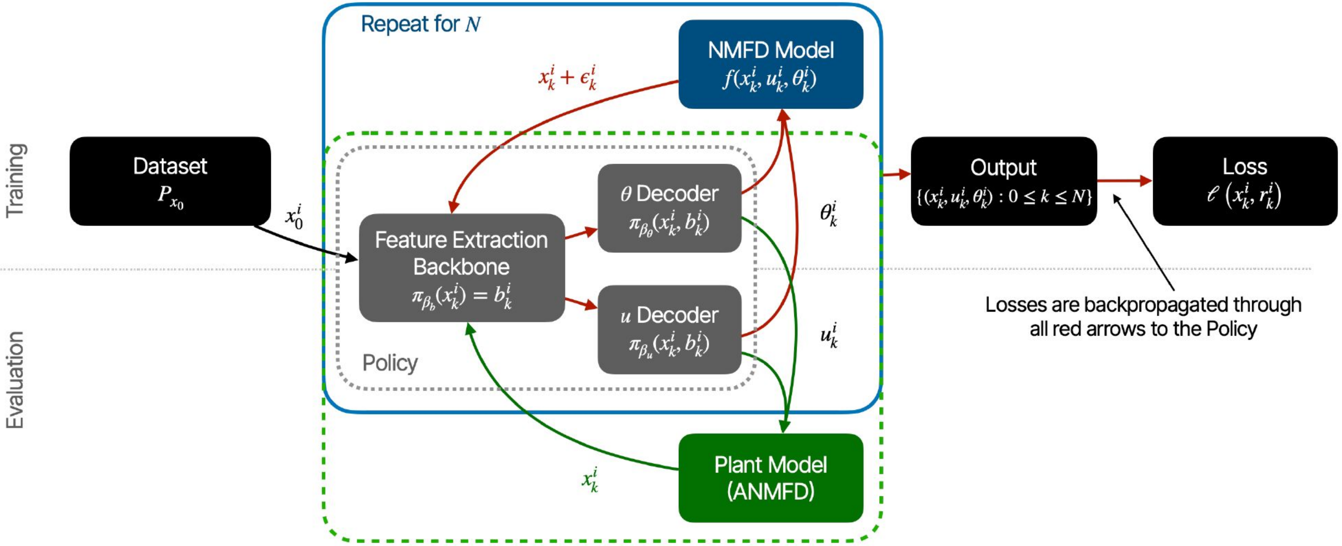
Ethan King, et al., Koopman-based Differentiable Predictive Control for the Dynamics-Aware Economic Dispatch Problem, American Control Conference 2022

Learning to Control Large-Scale Urban Road Networks

Networked Macroscopic Fundamental Diagram (NMFD).



Differentiable Predictive Control (DPC) with NMFD.



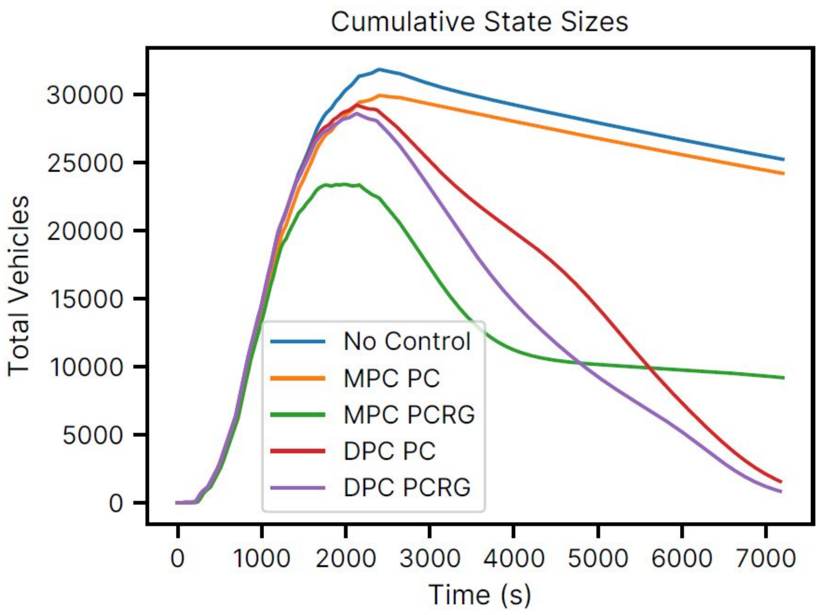
Case study

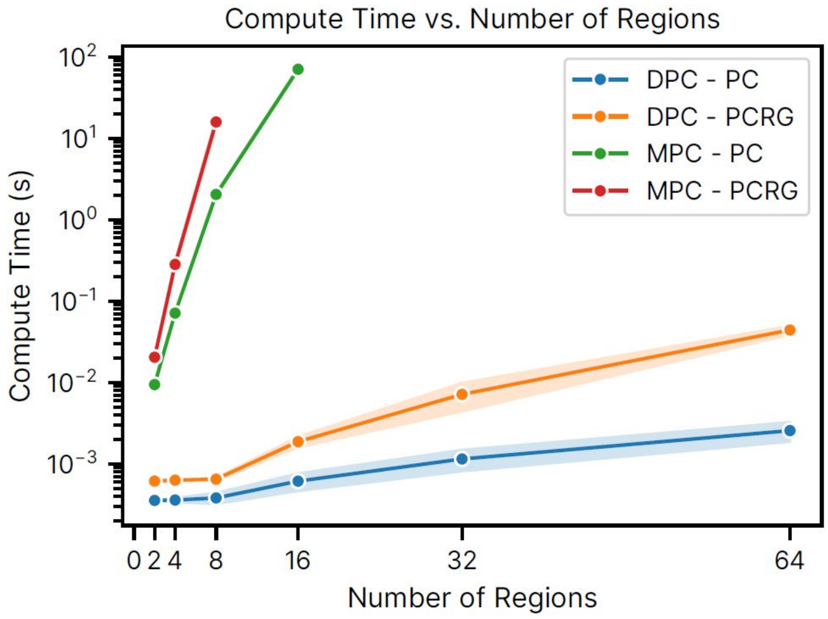
Realistic 7-region urban traffic scenario, and **synthetic 64-region** scenario modeled via NMFD.

Joint u **perimeter control (PC) and** θ **routing guidance (RG)** via DPC.

Renukanandan Tumu, Wenceslao Shaw Cortez, Ján Drgoňa, Draguna L. Vrabie, Sonja Glavaski, Differentiable Predictive Control for Large-Scale Urban Road Networks, arXiv:2406.10433, 2024

Learning to Control Large-Scale Urban Road Networks





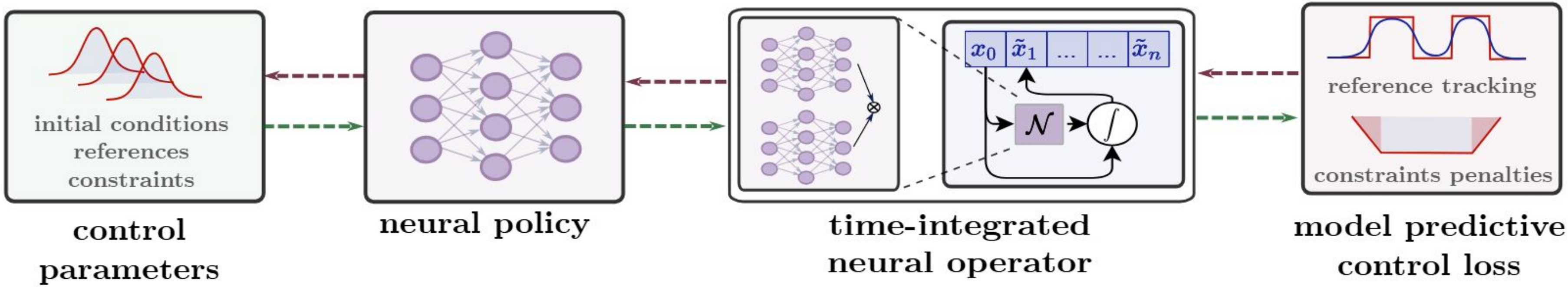
Benefits of Scientific Machine Learning

37% improvement in traveled time traffic performance. Up to **90% reduction** in total **traffic accumulation**.

Real-time decisions are made **4 orders of magnitude faster** than online model predictive control (MPC).

Renukanandan Tumu, Wenceslao Shaw Cortez, Ján Drgoňa, Draguna L. Vrabie, Sonja Glavaski, Differentiable Predictive Control for Large-Scale Urban Road Networks, arXiv:2406.10433, 2024

Learning to Control PDEs with DPC and Neural Operators



$$\min_{a(\cdot,\cdot)} \quad J[u,a] = \int_0^T \int_\Omega \ell(u(t,\mathbf{x}), a(t,\mathbf{x}), \boldsymbol{\xi}(t)) \; d\mathbf{x}\, dt + \int_\Omega \ell_T(u(T,\mathbf{x}))\, d\mathbf{x},$$

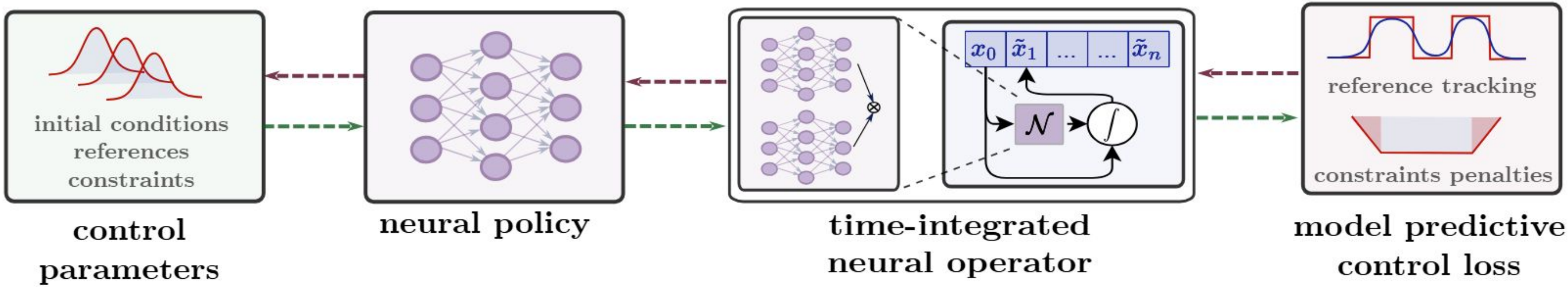
$$\text{s.t.} \quad \frac{\partial u}{\partial t} = \mathcal{F}(t, \mathbf{x}, u, \nabla u, \nabla^2 u, \ldots, a),$$

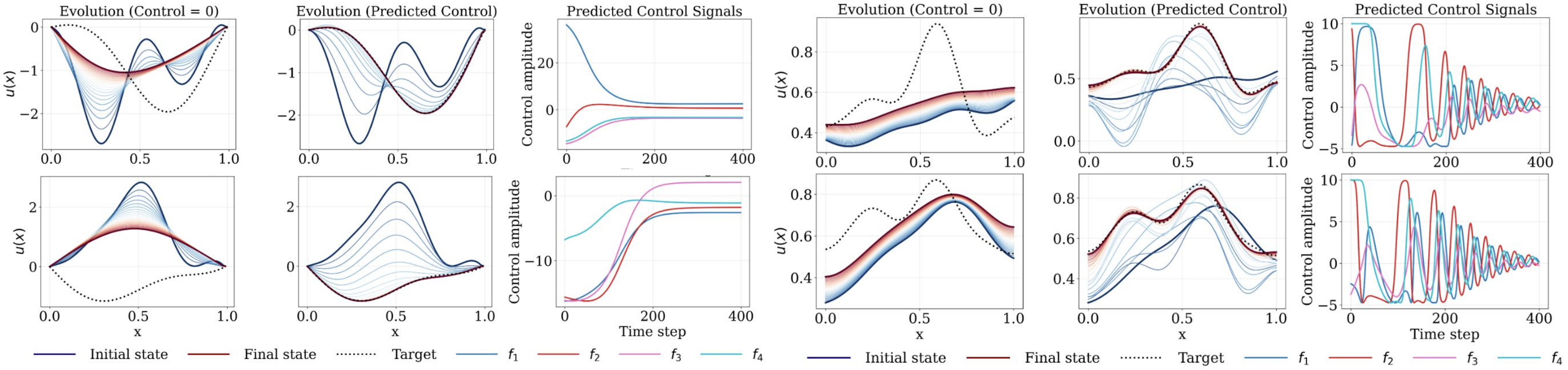
$$h(u(\mathbf{x},t), \boldsymbol{\xi}(t)) \leq 0, \quad g(a(\mathbf{x},t), \boldsymbol{\xi}(t)) \leq 0,$$

$$u(\mathbf{x},0) = u_0(\mathbf{x}), \quad \mathcal{B}(u) = 0.$$

D. R. Sharkar, J. Drgoňa, S. Goswami, "Learning to Control PDEs with Differentiable Predictive Control and Time-Integrated Neural Operators," under review, 2025

Learning to Control PDEs with DPC and Neural Operators

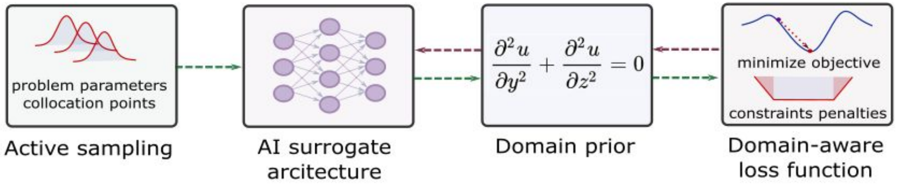


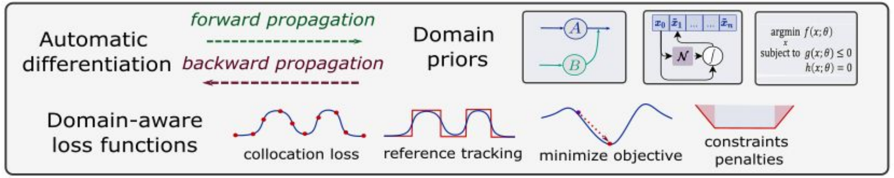


D. R. Sharkar, J. Drgoňa, S. Goswami, "Learning to Control PDEs with Differentiable Predictive Control and Time-Integrated Neural Operators," under review for L4DC, 2025

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 equality constraints
  - Gradients ill conditioning for deep graphs
  - Offline pre-training vs online adaptation
  - Computational cost of guarantees
- **Opportunities**
  - Large-scale modeling, design, and control of sustainable energy systems
  - PDE optimization and control
  - Development of new software tools