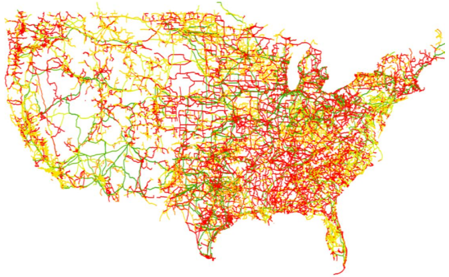
JOHNS HOPKINS
WHITING SCHOOL *of* ENGINEERING

Neuromancer: Differentiable Programming Library for Data-Driven Modeling and Control

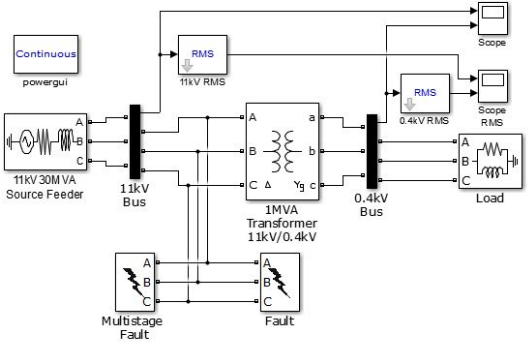
Ján Drgoňa
Associate Professor @ CaSE & ROSEI

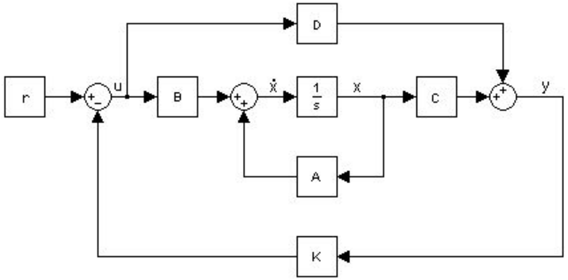
Optimizing Complex Energy Systems is Hard

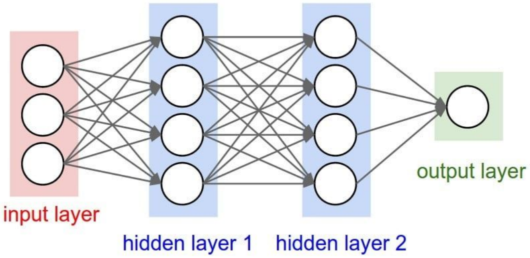
- ***Simulations*** are crucial for optimal decision-making in complex energy systems
- ***Need:*** Improve computational efficiency and scalability of digital twins
- ***Challenges:***
  1. Modeling and simulation of complex systems is hard
  2. Optimal control and closed-loop decision-making for complex 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 ⟷ Data-driven

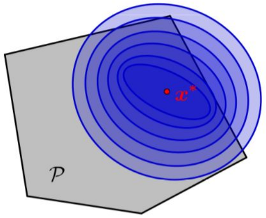
More domain knowledge

Less domain knowledge

Challenge 2: Heterogenous Solution Methods

Constrained Optimization

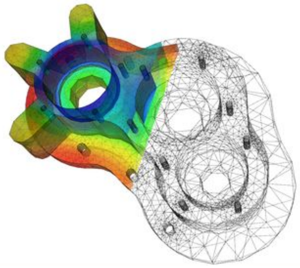
$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & b(x) \geq 0 \\ & c(x) = 0. \end{aligned}$$



- Requires prior knowledge of objective function and constraints

Differential Equations

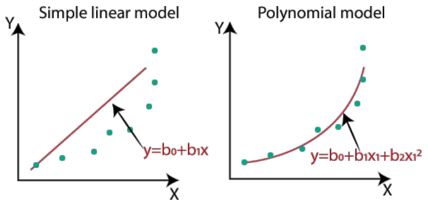
$$\frac{dy}{dx} = f(x)$$



- Requires prior knowledge of the physics to be modeled

Supervised Learning

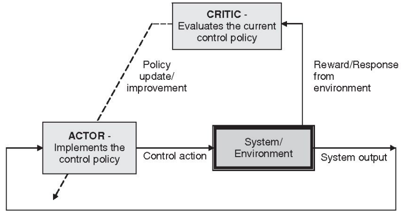
$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(y^i, f(x^i, \theta))$$



- Requires large labeled datasets

Reinforcement Learning

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



- Requires environment model to sample

More domain knowledge

Less domain knowledge

Challenge 3: Heterogenous Solution Tools

Constrained Optimization

PYOMO
JUMP
GEKKO DYNAMIC OPTIMIZATION
CVXPY
CasADi
GUROBI OPTIMIZATION

Differential Equations

Ansys
MATLAB®
PETSc
TAO
SciPy
mfem

Supervised Learning

PyTorch
TensorFlow
flux
JAX

Reinforcement Learning

Gym

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.

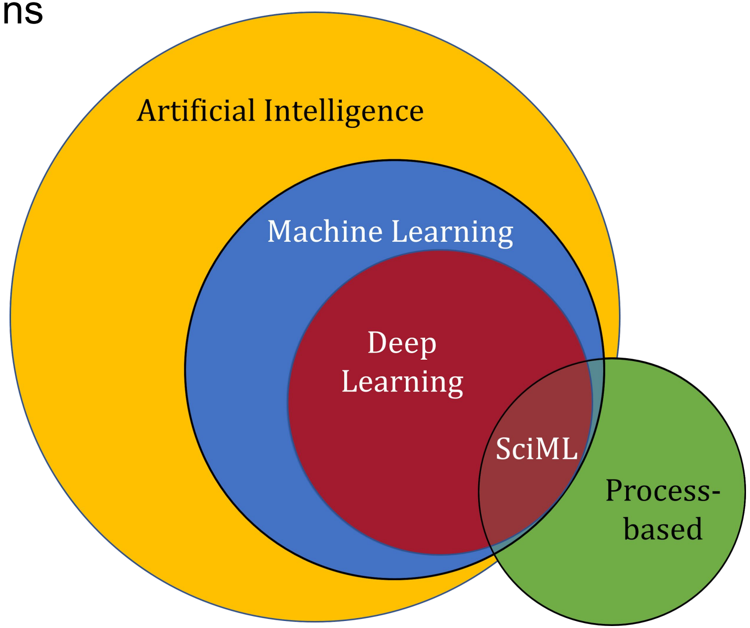
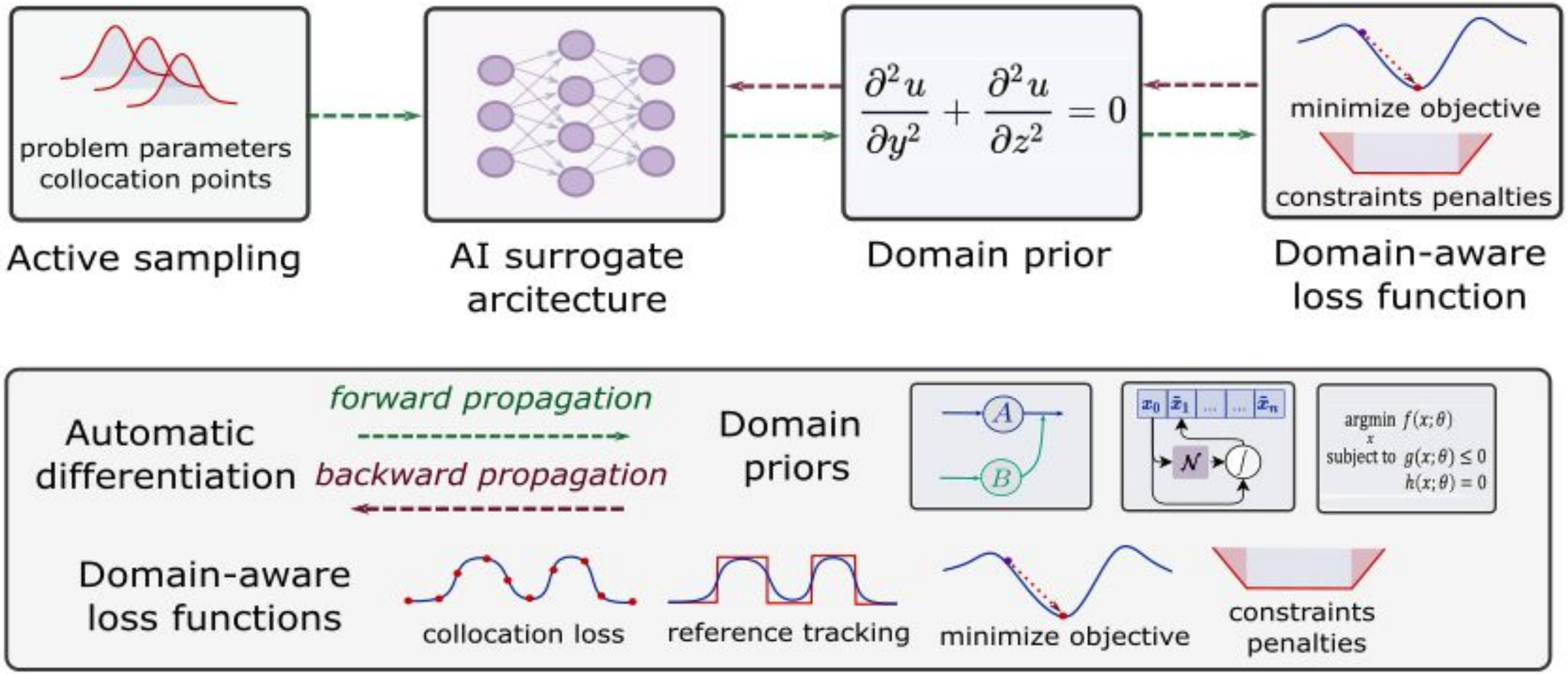


Image source: https://sciml.wur.nl/reviews/sciml/sciml.html

Selected Scientific Machine Learning Literature

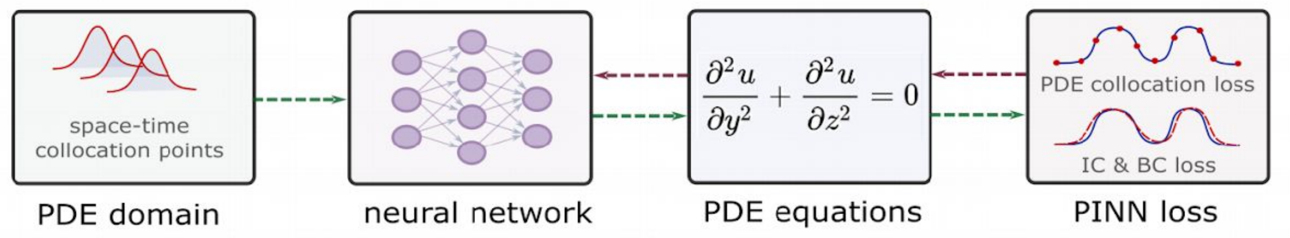
- **Differentiable Programming**
  - M. Innes, et al., *A Differentiable Programming System to Bridge Machine Learning and Scientific Computing, 2019*
- **Learning to Solve (L2S)**
  - M. Raissi, et al., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems 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

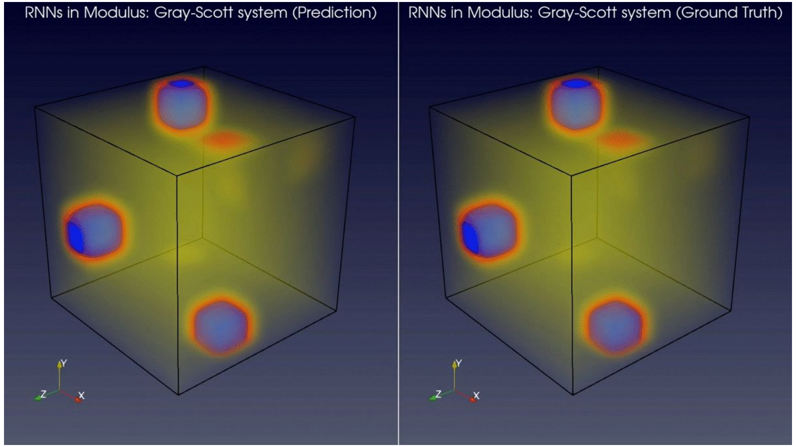


Karniadakis, G.E., Kevrekidis, I.G., Lu, L. et al. Physics-informed machine learning. Nat Rev Phys 3, 2021.
Thiyagalingam, J., Shankar, M., Fox, G. et al. Scientific machine learning benchmarks. Nature Reviews Physics 4, 413–420, 2022.
Nghiem T., Drgona J., et al. Physics-Informed Machine Learning for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

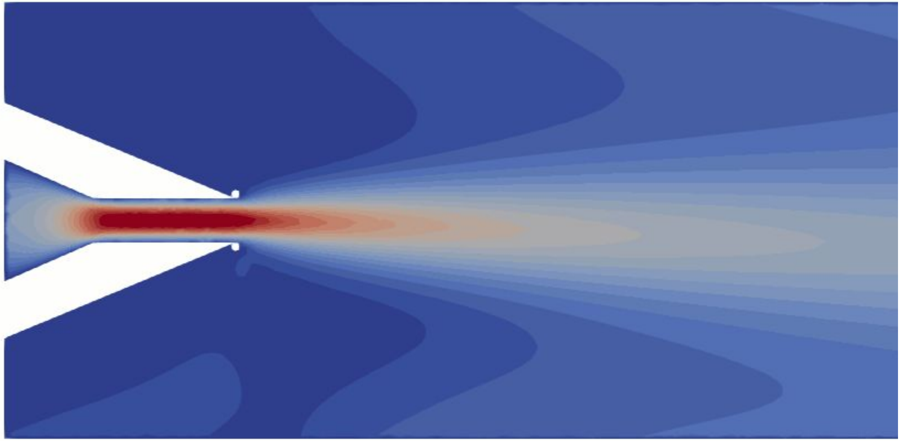
Learning to Solve Differential Equations with Physics-Informed Neural Networks (PINNs)



Training neural networks as PDE solutions



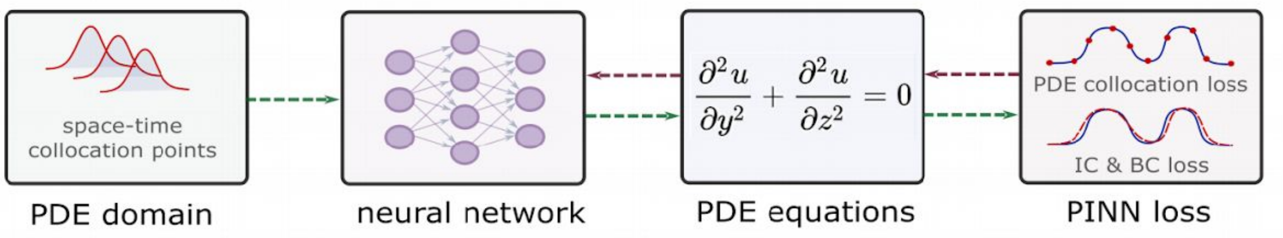
Application: Parameter estimation from data



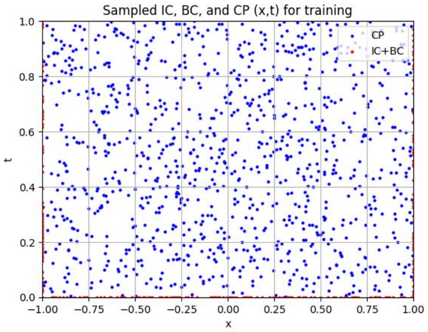
M. Raissi, et al., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics*, 2019

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.

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

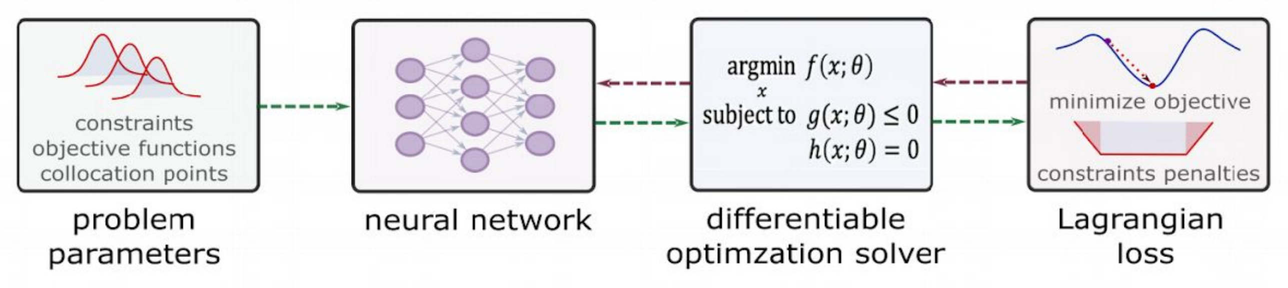
$$\ell_u = \frac{1}{N_u} \sum_{i=1}^{N_u} |y(t_u^i, x_u^i) - NN_\theta(t_u^i, x_u^i)|^2$$

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

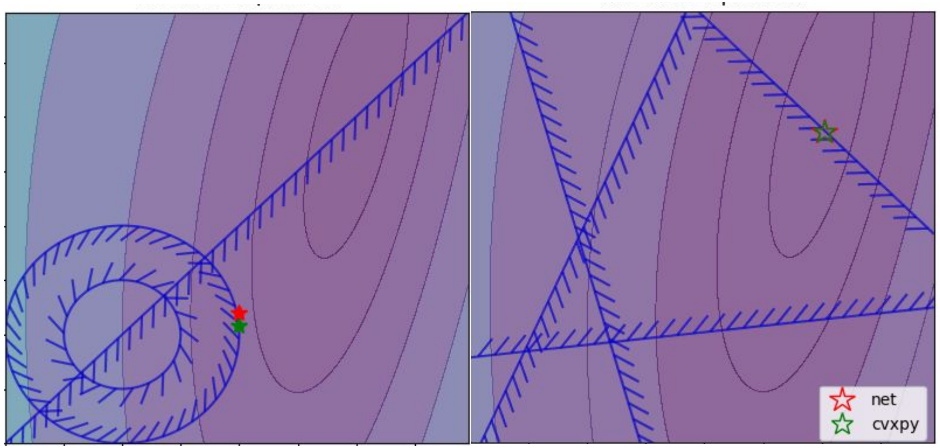
https://github.com/pnnl/neuromancer/blob/master/examples/PDEs/Part_2_PINN_BurgersEquation.ipynb

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



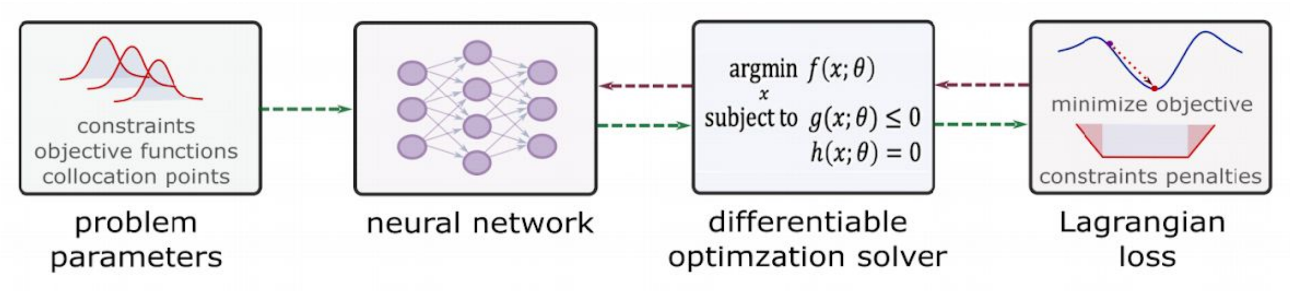
Training neural networks as optimization solutions



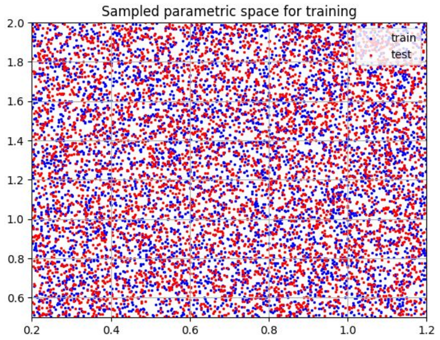
Application: solving optimal power flow

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.

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

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

Loss function: minimizing objective function and constraints penalties.

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

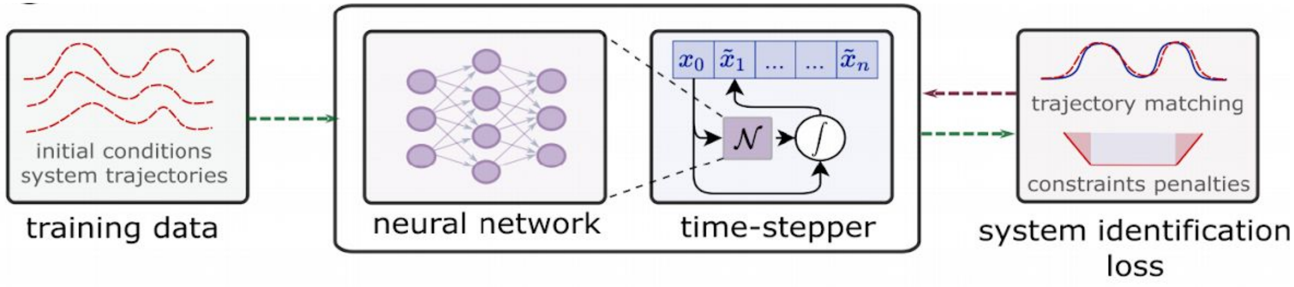
$$\ell_g = \frac{1}{m}\sum_{i=1}^{m} |\text{RELU}(g(x^i,\xi^i))|^2$$

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

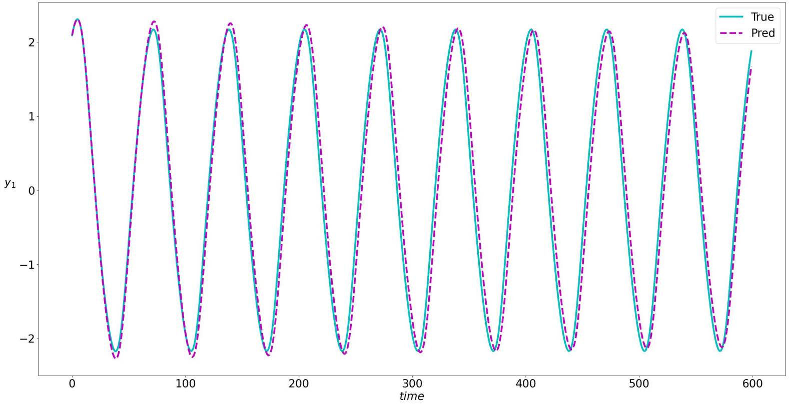
https://github.com/pnnl/neuromancer/blob/master/examples/parametric_programming/Part_1_basics.ipynb

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

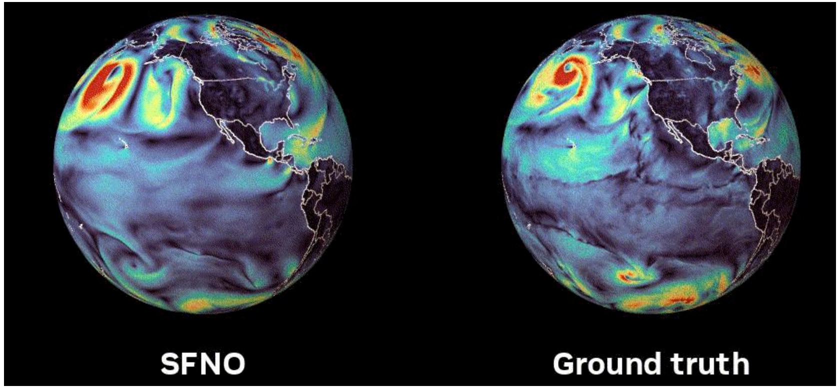
Learning to Model (L2M) Dynamical Systems



Neural models for nonlinear system identification



Application: climate modeling

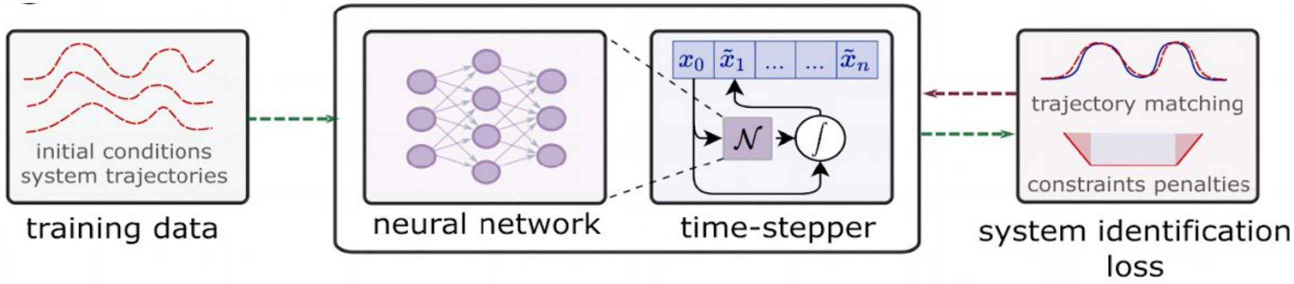


R. T. Q. Chen, et al., *Neural ordinary differential equations. NeurIPS, 2018*

C. Rackauckas, et al., *Universal Differential Equations for Scientific Machine Learning*, 2021

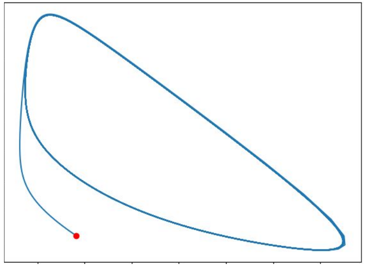
Image: NVIDIA FourCastNet

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**

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

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

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

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

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

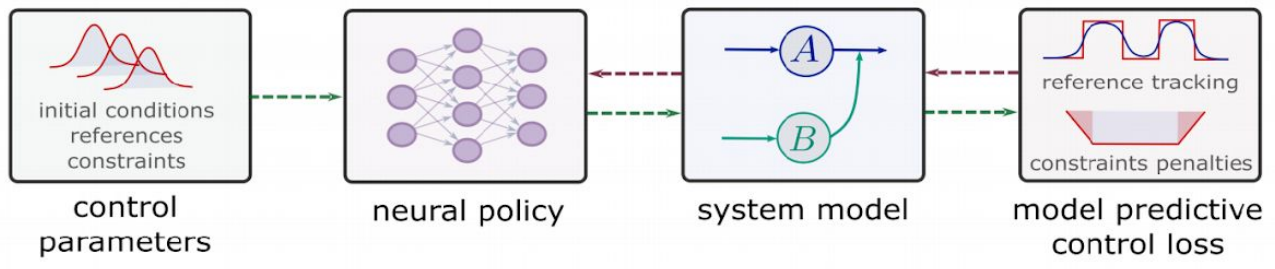
$$\ell_2 = \sum_{i=1}^{m} \sum_{k=1}^{N-1} Q_{dx} ||\Delta x_k^i - \Delta \hat{x}_k^i||_2^2$$

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

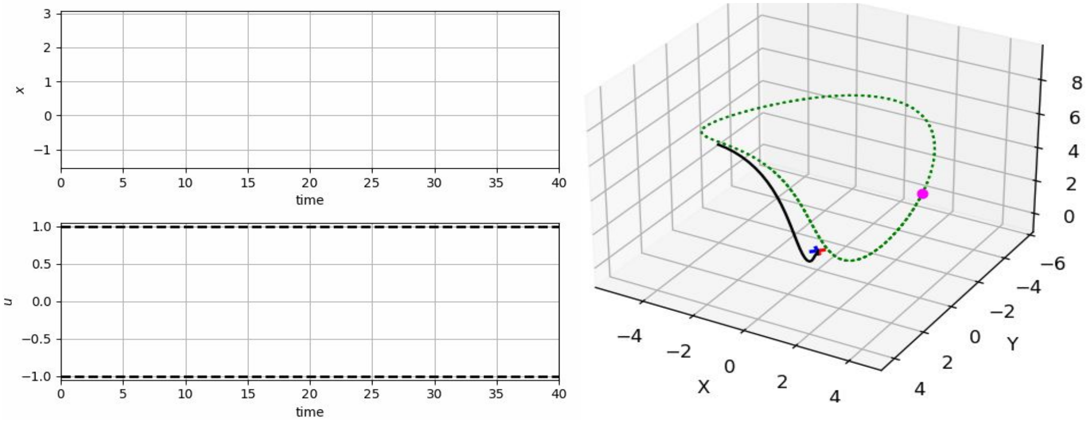
https://github.com/pnnl/neuromancer/blob/master/examples/ODEs/Part_1_NODE.ipynb

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

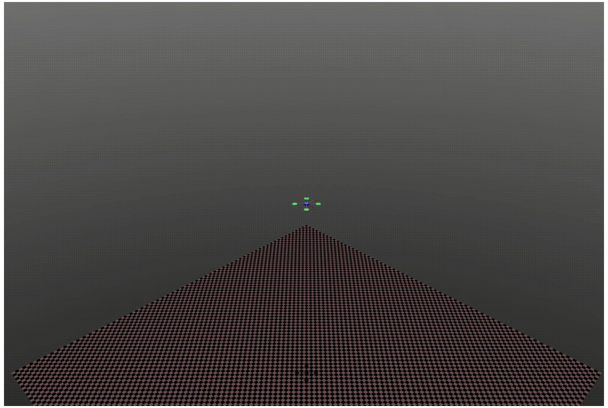
Learning to Control (L2C) with Differentiable System Models



Trajectory optimization for dynamical systems

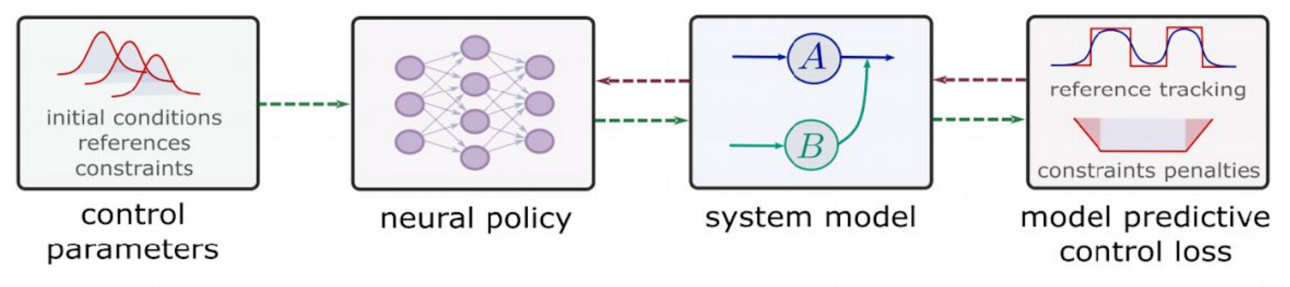


Application: autonomous systems

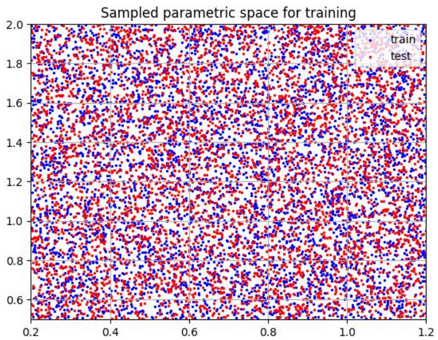


J. Drgoňa, A. Tuor and D. Vrabie, "Learning Constrained Parametric Differentiable Predictive Control Policies With Guarantees," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2024

Learning to Control (L2C) with Differentiable System Models



Dataset: collocation points in the control parametric space.



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

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

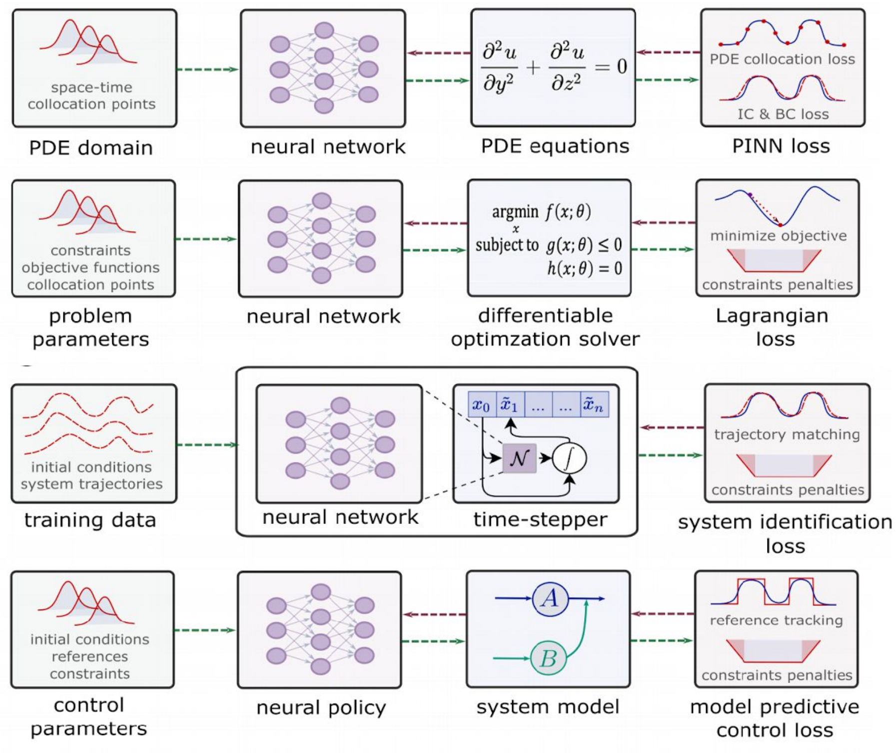
Loss function: reference tracking, constraints and terminal penalties.

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

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

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

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

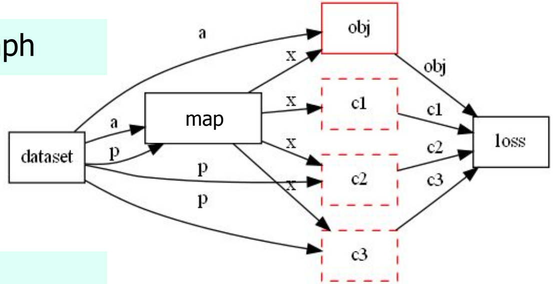
$$\min_{\Theta} \; (1-\mathbf{x})^2 + \boldsymbol{p}(\mathbf{y}-\mathbf{x}^2)^2$$
$$\text{s.t. } (\boldsymbol{p}/2)^2 \leq \mathbf{x}^2 + \mathbf{y}^2 \leq \boldsymbol{p}^2, \;\; \mathbf{x} \geq \mathbf{y}$$
$$\mathbf{x} = \pi_{\Theta}(\boldsymbol{p})$$

2. Python code interface

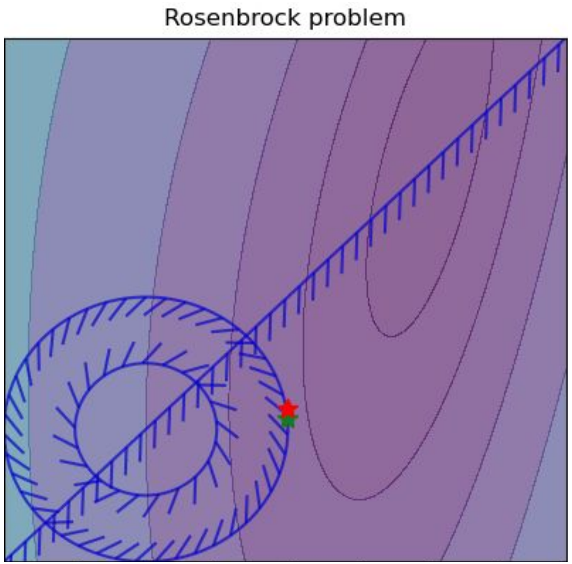
```python
import neuromancer as nm

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



Summary

- **Scientific machine learning (SciML)** methods integrating deep learning, constrained optimization, physics-based modeling, and control
  - Learning to optimize (L2O)
  - Learning to control (L2C)
  - Learning to model (L2M)
  - Learning to solve (L2S)
- **Energy systems applications**
  - Buildings
  - Power systems
  - Wind farms
  - Energy storage
  - Transportation networks

drgona.github.io

jdrgona1@jh.edu

github.com/pnnl/neuromancer

Acknowledgements

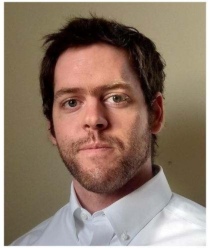
Aaron Tuor

James Koch

Madelyn Shapiro

Bruno Jacob

Rahul Birmiwal



Ethan King

Sonja Glavaski

Sayak Mukherjee

Wenceslao Shaw Cortez

Soumya Vasisht

Shrirang Abhyankar

Mahantesh Halappanavar

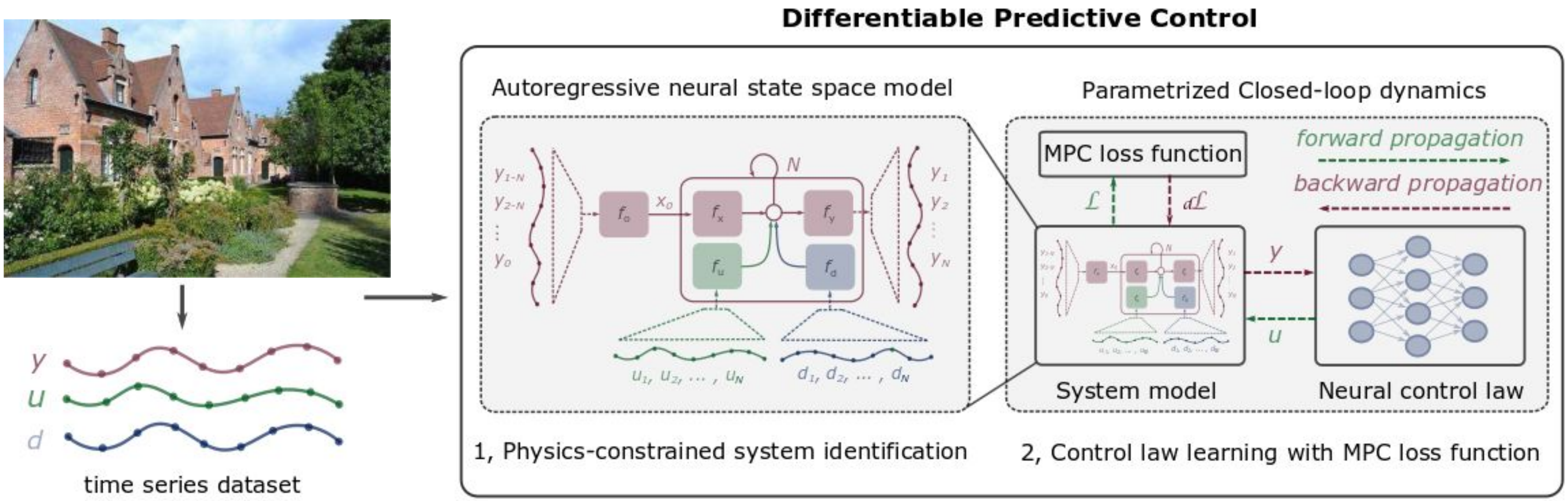
Panos Stinis

Draguna Vrabie

U.S. DEPARTMENT OF ENERGY

Pacific Northwest NATIONAL LABORATORY

Learning to Control Building Energy System



Benefits of Scientific Machine Learning-based Digital Twins

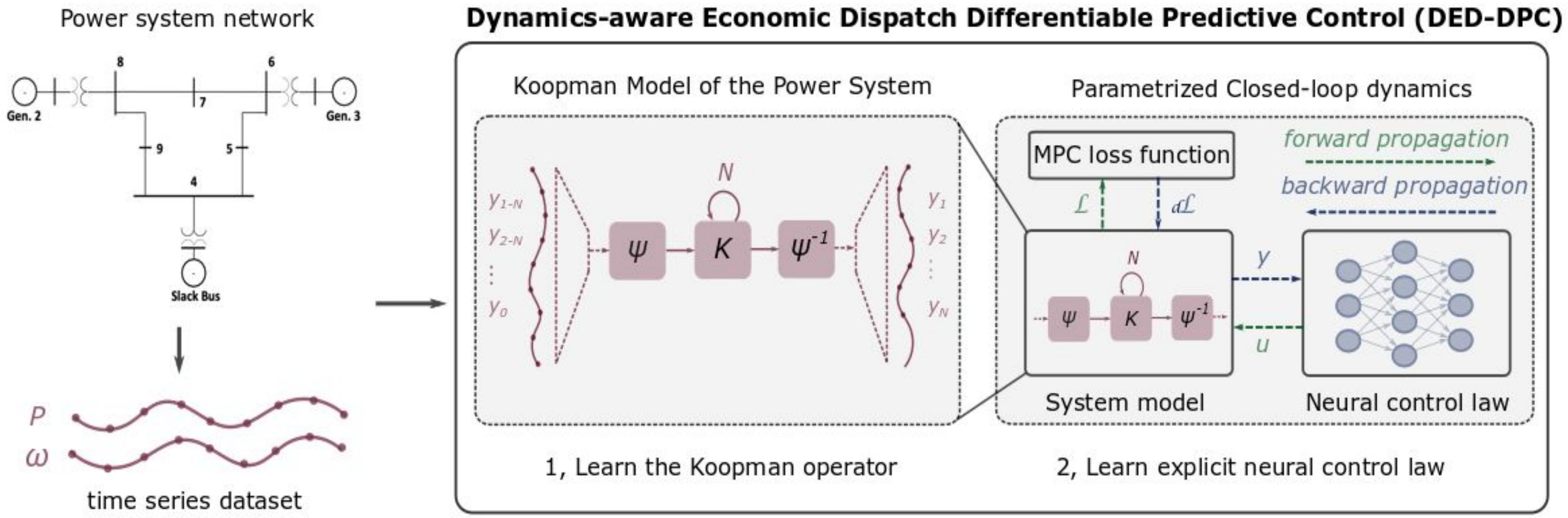
Modeling and optimal control design is roughly **10-times faster** and requires **less expertise**.

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-based Digital Twins

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.

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