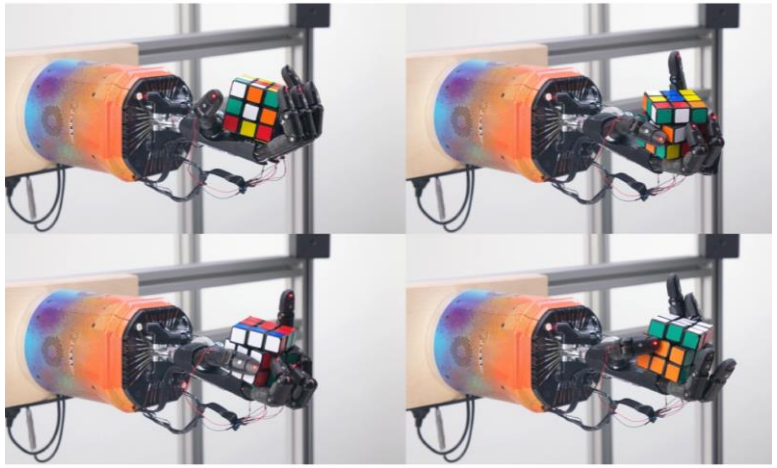
Physics-Informed Machine Learning for Modeling and Control of Dynamical Systems

Authors: Truong X. Nghiem, Jan Drgona, Colin Jones, Zoltan Nagy, Roland Schwan, Biswadip Dey, Ankush Chakrabarty, Stefano Di Cairano, Joel A. Paulson, Andrea Carron, Melanie N. Zeilinger, Wenceslao Shaw Cortez, Draguna L. Vrabie

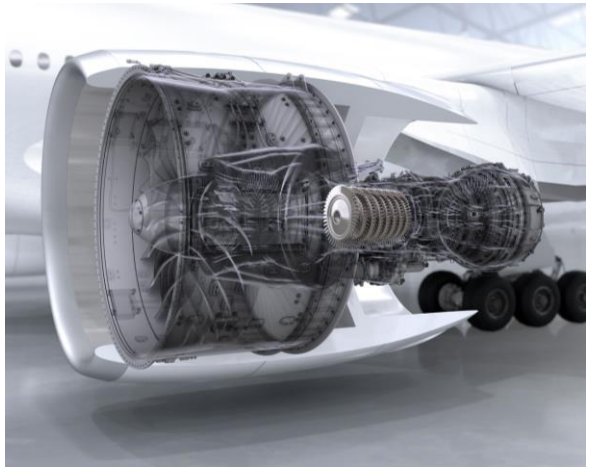
Engineering systems generate large amounts of data

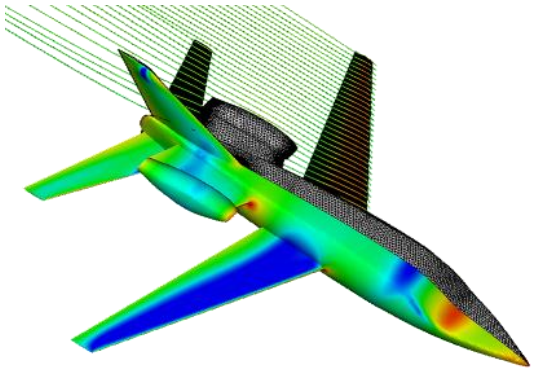
Process Control

Energy Systems



Robotics

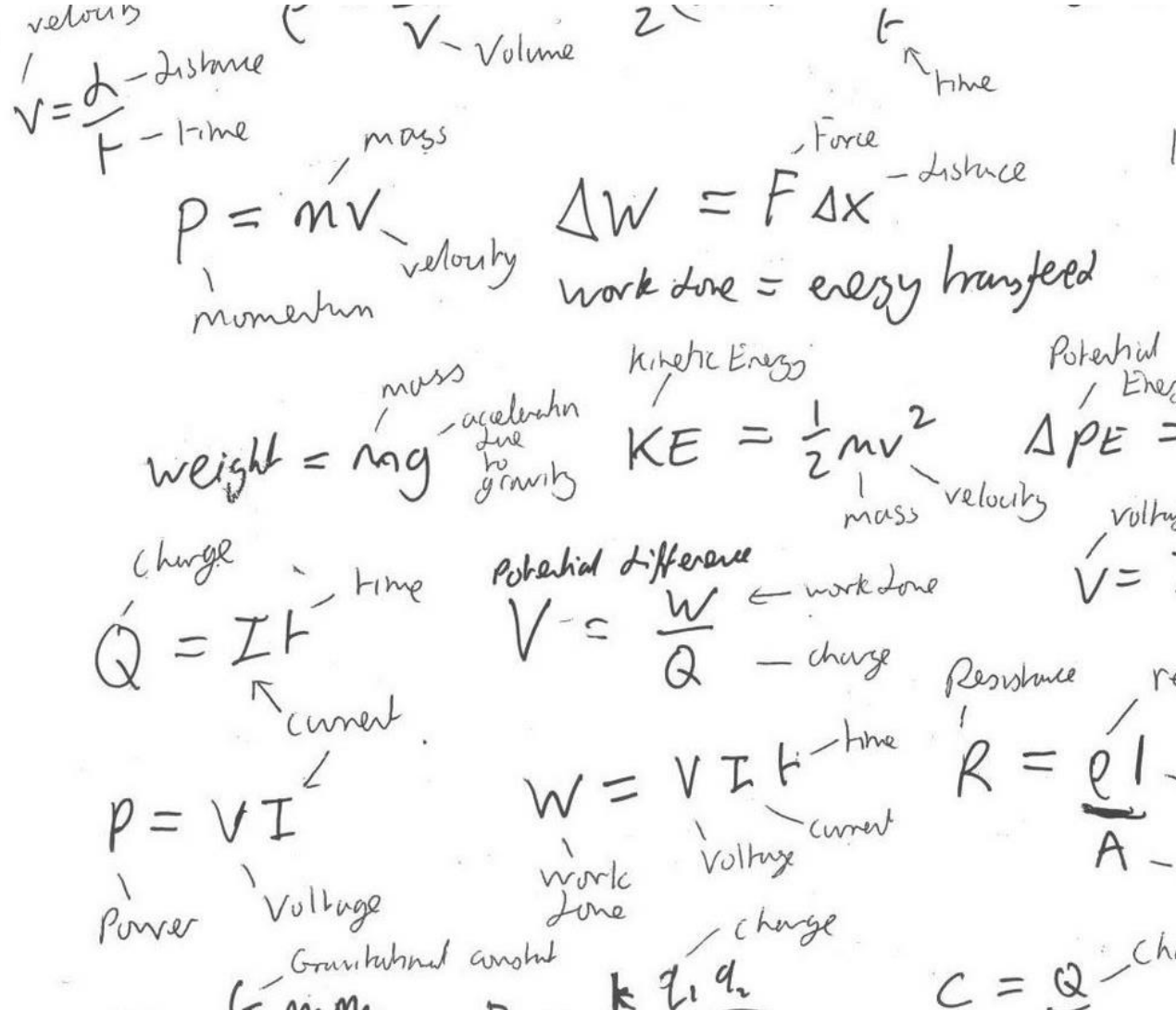




Aerospace industry

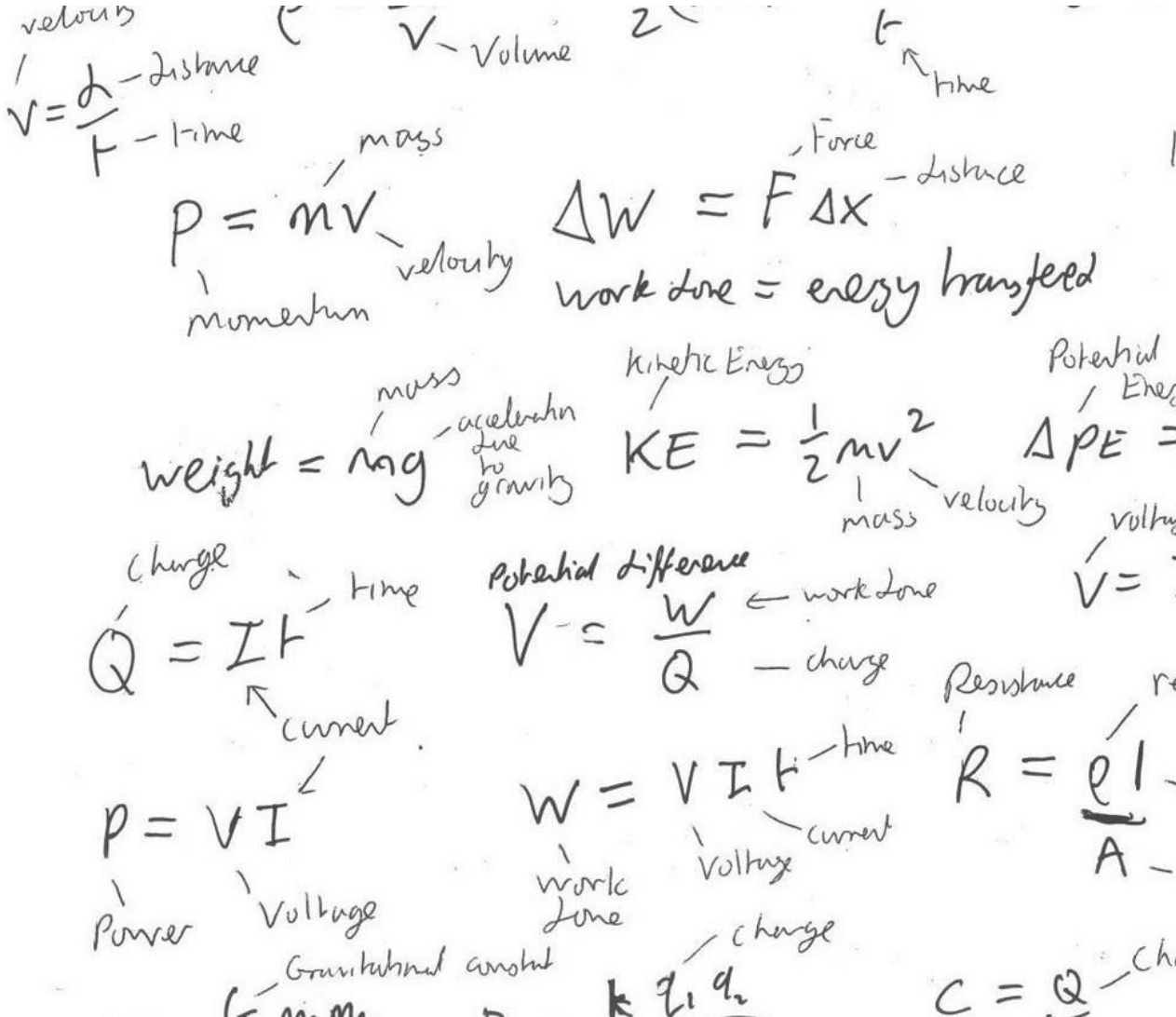
However traditional engineering applications

- are governed by fundamental principles
- have physical constraints
- Examples:
  - Conservation law,
  - Laws of motion
  - Ranges of constants (efficiencies)
  - Physical constants (gravity)



ML methods **cannot** satisfy fundamental principles and underlying physics

Significant limitation to real-world deployment



If we could integrate Physics into ML …

- Reduced data requirements for ML
- Higher precision & improved generalization
- Increased interpretability & trust

If we could integrate Physics into ML …

- Reduced data requirements for ML
- Higher precision & improved generalization
- Increased interpretability & trust
- This tutorial ☺

Physics
Informed
Machine
Learning

Also called:

- Scientific machine learning (SciML)
- Physics-constrained machine learning
- Domain-aware machine learning
- Knowledge-based machine learning

Definition: Physics Informed Machine Learning

PIML is a set of methods and tools that **systematically integrate machine learning algorithms with mathematical models** developed in various scientific and engineering domains.

… **enforcing constraints** such as symmetries, causal relationships, or conservation laws, or abstract properties such as stability, convexity, or invariance ...

PIML for Control of Dynamical Systems

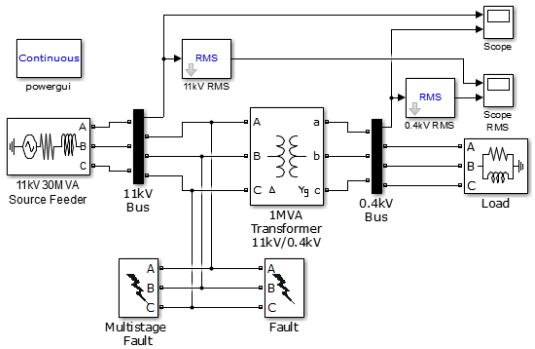
I. PIML for System Identification
II. PIML for Control
III. Analysis and Verification of ML models
IV. PIML for Digital Twins
V. Conclusions | Challenges | Opportunities

Physics-Informed Machine Learning for System Identification

Jan Drgona

Landscape of Dynamical Systems Modeling Approaches

Models



White-box

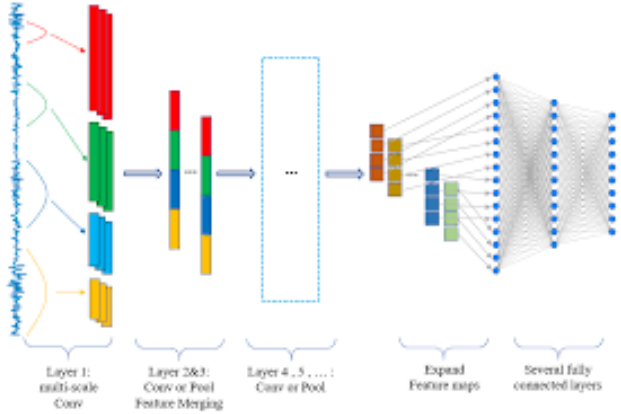
$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y$$

Gray-box

$$\frac{dx}{dt} = \alpha x - 0.75xy$$
$$\frac{dy}{dt} = NN(x, y)$$

Black Box

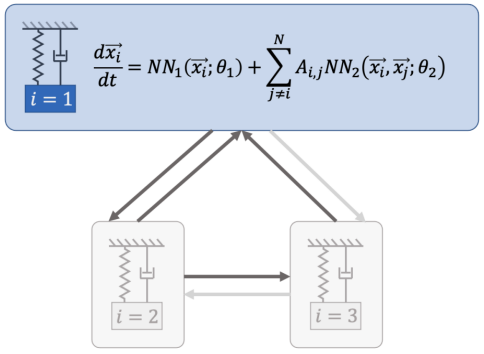
$$\frac{d\boldsymbol{X}}{dt} = NN(\boldsymbol{X})$$

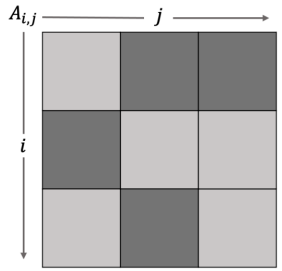


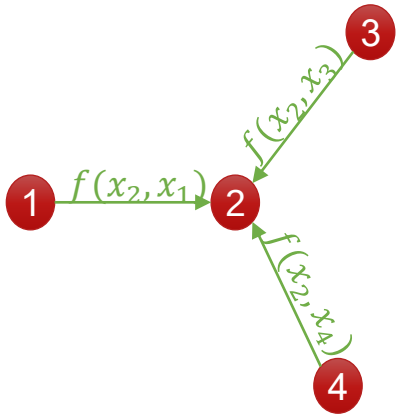
More domain knowledge

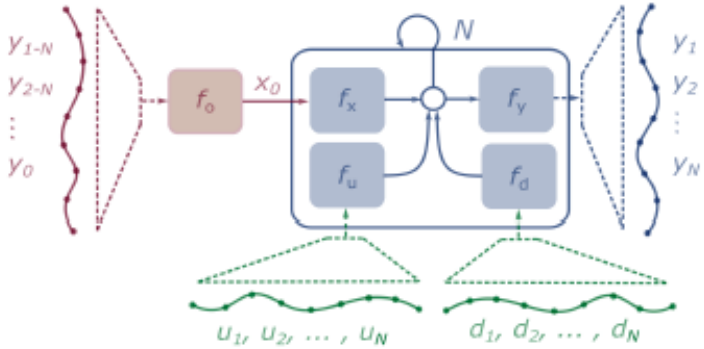
Less domain knowledge

- physics-based
- physics priors
- networked neural ODEs
- graph neural networks
- neural differential equations
- state space models







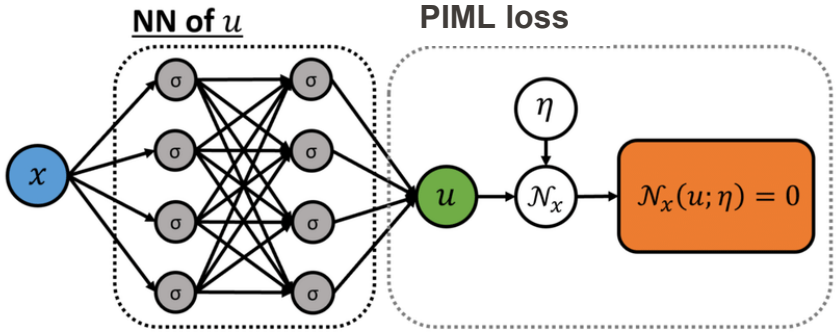


PIML Loss Functions

Dataset: time series of inputs and outputs $S = \{(\mathbf{u}_t^{(i)}, \mathbf{y}_t^{(i)}), (\mathbf{u}_{t+\Delta}^{(i)}, \mathbf{y}_{t+\Delta}^{(i)}), \ldots, (\mathbf{u}_{t+N\Delta}^{(i)}, \mathbf{y}_{t+N\Delta}^{(i)})\}$

PIML Loss:

$$\mathscr{L} = \sum_{i}^{n} \ell_{\text{data}}^{i} + \sum_{j}^{m} \ell_{\text{physics}}^{j}$$



Loss Terms:

$$\mathcal{L}_y = \frac{1}{n \cdot N \cdot n_y} \sum_{i=1}^{n} \sum_{t=1}^{N} \sum_{j=1}^{n_y} (\hat{\mathbf{y}}_{t,j}^{(i)} - \mathbf{y}_{t,j}^{(i)})^2$$

Tracking

$$\mathbf{s}^{\underline{\mathbf{y}}} = \max(0, -\hat{\mathbf{y}} + \underline{\mathbf{y}}) \qquad \mathcal{L}_y^{\text{con}} = \frac{1}{n_y} \sum_{i=1}^{n_y} (\mathbf{s}_i^{\underline{\mathbf{y}}} + \mathbf{s}_i^{\overline{\mathbf{y}}})$$
$$\mathbf{s}^{\overline{\mathbf{y}}} = \max(0, \hat{\mathbf{y}} - \overline{\mathbf{y}})$$

State constraints penalties

$$p_V(\mathbf{x}_{k+1}, \mathbf{x}_k) = ||\texttt{ReLU}(V_\phi(\mathbf{x}_{k+1}) - V_\phi(\mathbf{x_k}))||_2,$$

Lyapunov condition penalties

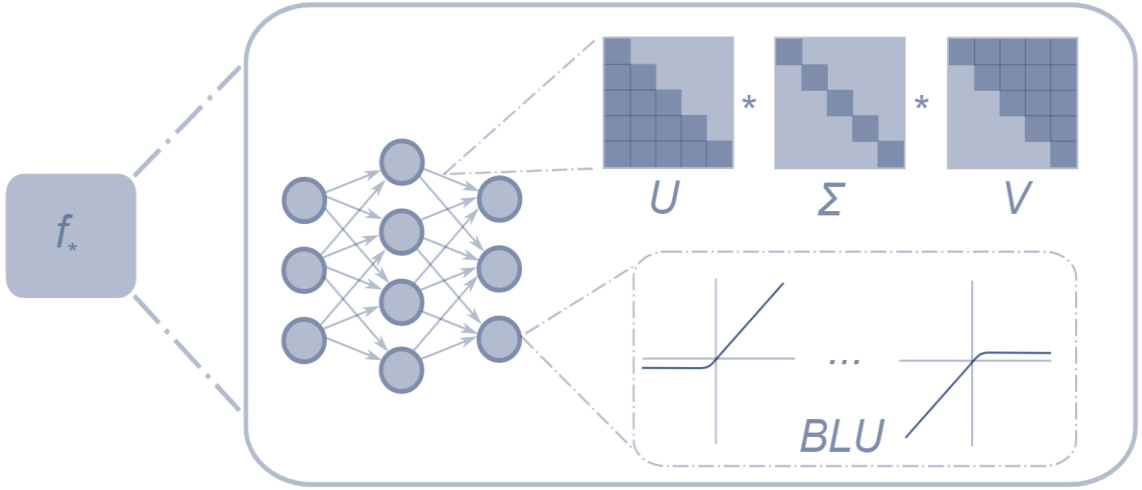
$$\| \nabla \mathbf{f}(\mathbf{z}) \|_F^2 = \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{N}(0,1)} \epsilon^\intercal \nabla \mathbf{f}(\mathbf{z}) \nabla \mathbf{f}(\mathbf{z})^\intercal \epsilon$$

Jacobian regularizations

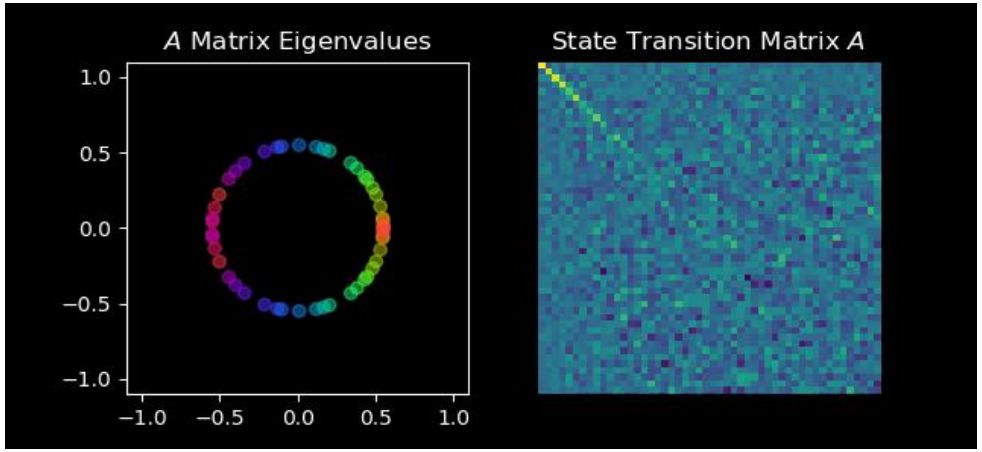
M. Raissi, et al., *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, 2019

Matrix Factorizations for PIML Models

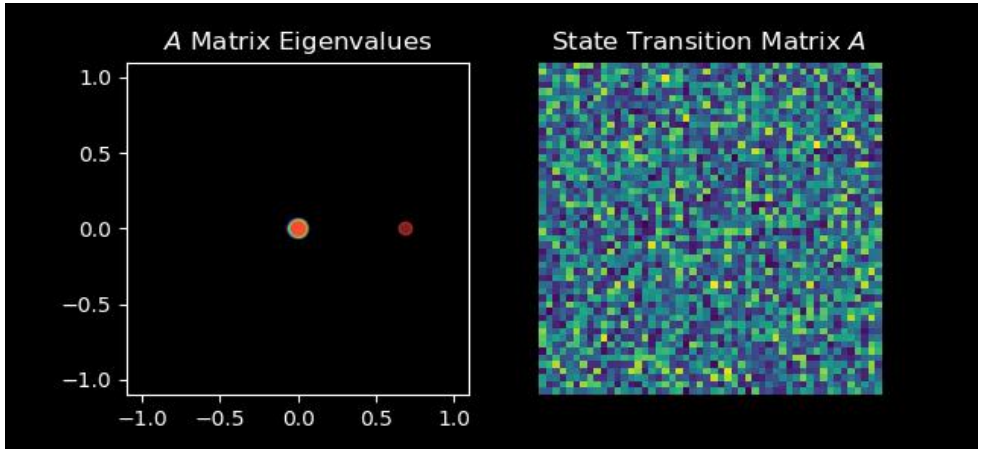
- Can be used to enforce sparsity, spectral conditions, or other linear algebra properties:
  - Perron-Frobenius Tuor et al., 2020
  - SVD Spectral Zhang et al., 2018
  - Orthogonal Mhammedi et al., 2017
  - Symplectic Haber et al., 2017
  - Anti-symmetric Chang et al., 2019
  - Schur Decomposition Kerg et al., 2019



Spectral map



Perron-Frobenius map

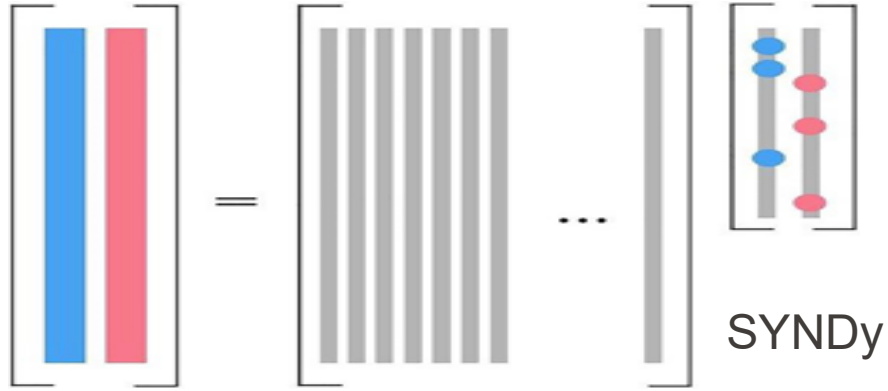


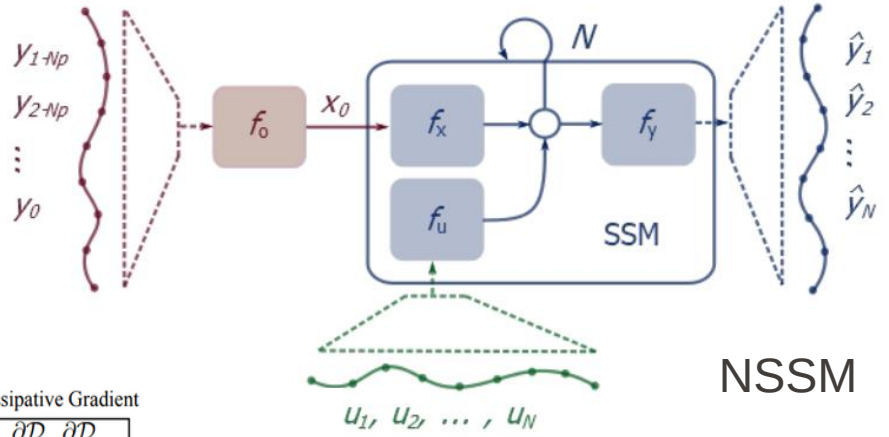
https://github.com/pnnl/slim

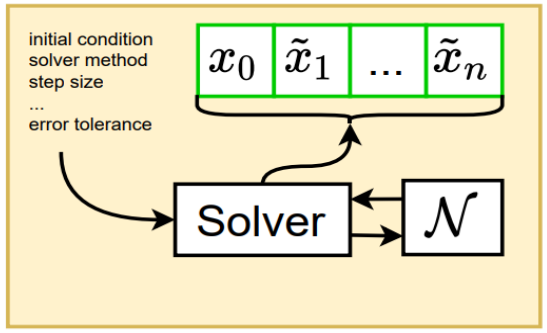
Structural Priors in Data-driven Models

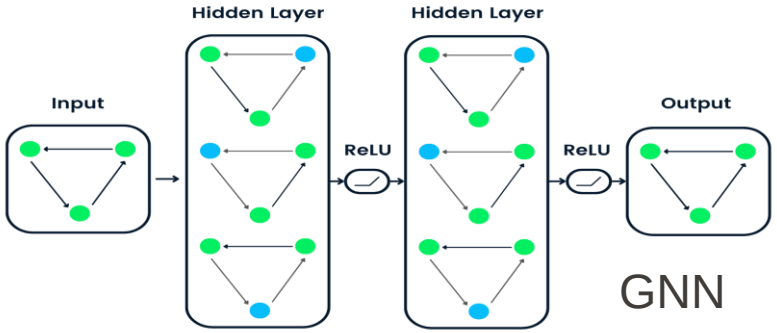
PIML Methods for Learning Nonlinear Dynamics:

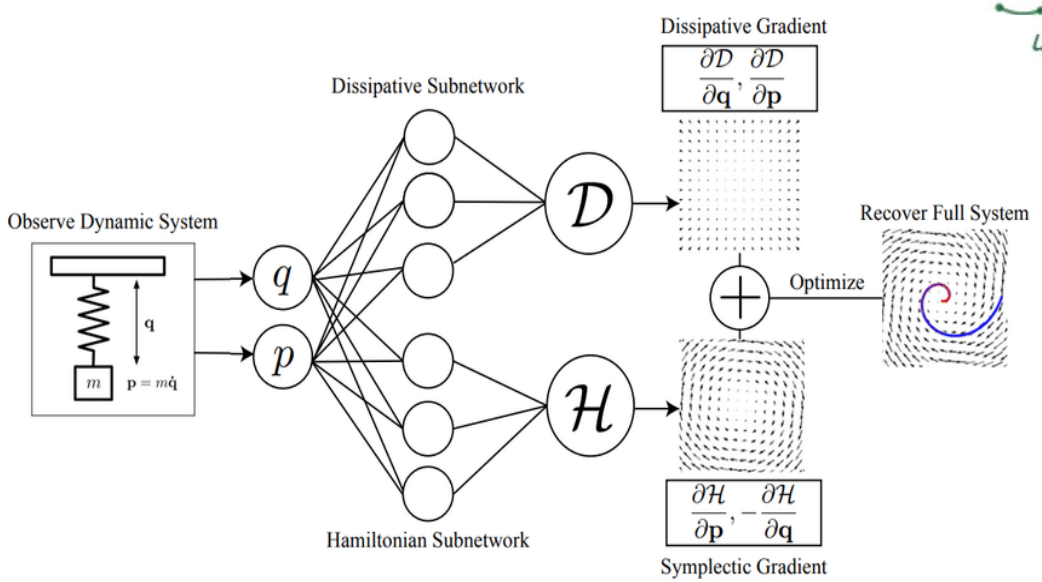
- ✓ Sparse identification of nonlinear dynamics (SYNDy) Brunton et al., 2016
- ✓ Neural State Space Models (NSSM) Skomski et al., 2021
- ✓ Graph Neural Networks (GNN) Scarselli et al., 2004
- ✓ Neural Ordinary Differential Equations (NODE) RTQ Chen et al., 2018
- ✓ Universal Differential Equations (UDE) Rackauckas et al., 2020
- ✓ Hamiltonian Neural Networks (HNN) Greydanus et al., 2019









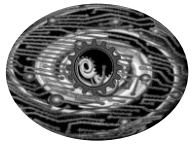


$$\frac{dx}{dt} = \alpha x - 0.75xy$$
$$\frac{dy}{dt} = NN(x, y)$$

UDE

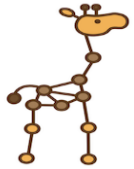
Open-Source PIML Tools for Modeling Dynamical Systems

Julia SciML: Open-Source Software for Scientific Machine Learning https://sciml.ai/



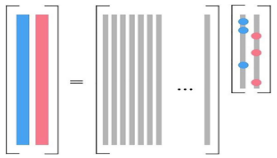
Neuromancer - PyTorch library for physics-informed system identification and control https://github.com/pnnl/neuromancer

Torchdyn - PyTorch library for neural differential equations https://github.com/DiffEqML/torchdyn



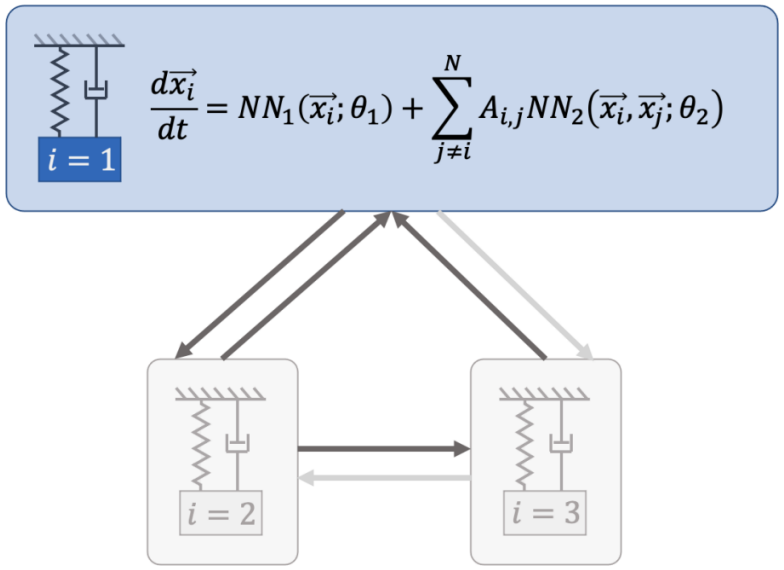
Jraph - JAX library for graph neural networks https://github.com/deepmind/jraph

Pytorch Geometric - PyTorch library for graph neural networks https://github.com/pyg-team/pytorch_geometric

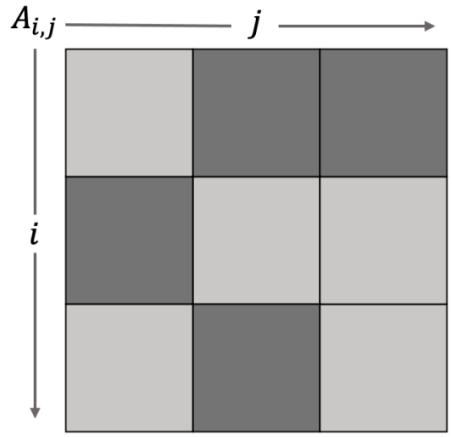


PySINDy is a sparse regression package for the SINDy method https://github.com/dynamicslab/pysindy

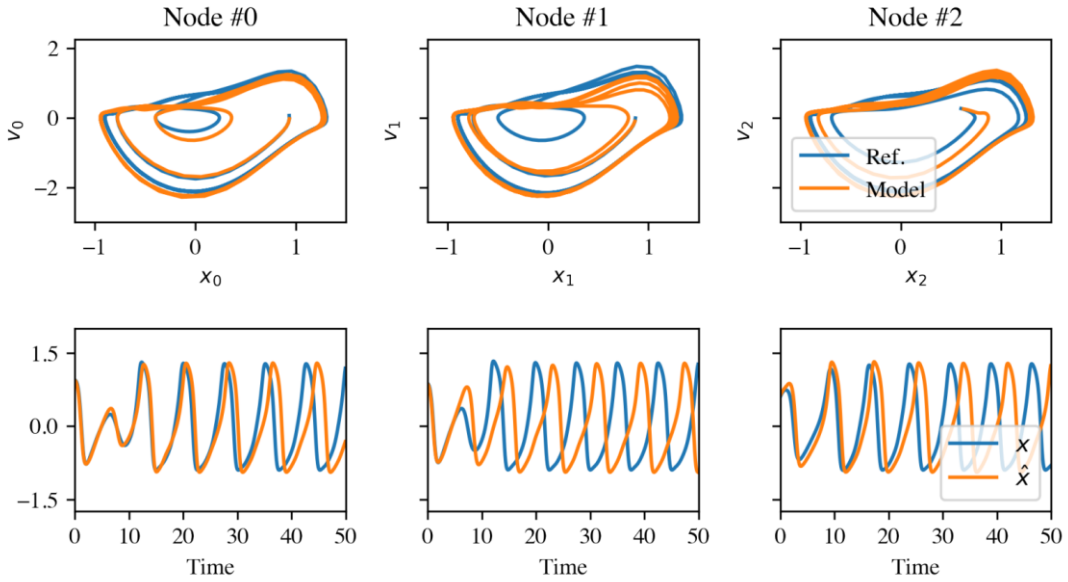
Case Study: Networked Dynamical Systems via Universal Differential Equations



Network of unknown oscillators



Coupling adjacency



Ground truth Kuramoto system:

$$\frac{d\theta_i}{dt} = \omega + \frac{K}{N}\sum_{j=1}^{N} \sin(\theta_i - \theta_j).$$

Generalization of learned dynamics on never-seen network topology.

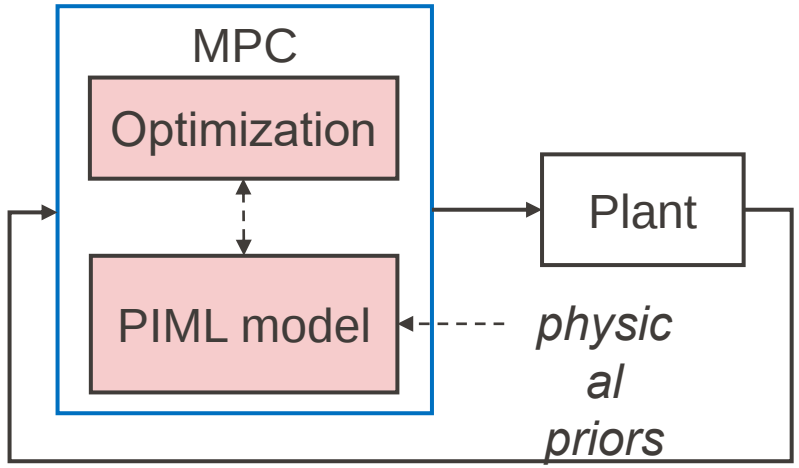
J. Koch, Z. Chen, A. Tuor, J. Drgona, D. Vrabie; Structural inference of networked dynamical systems with universal differential equations. Chaos, 1 February 2023; 33 (2): 023103.

Physics-Informed Machine Learning for Control

Wenceslao Shaw Cortez

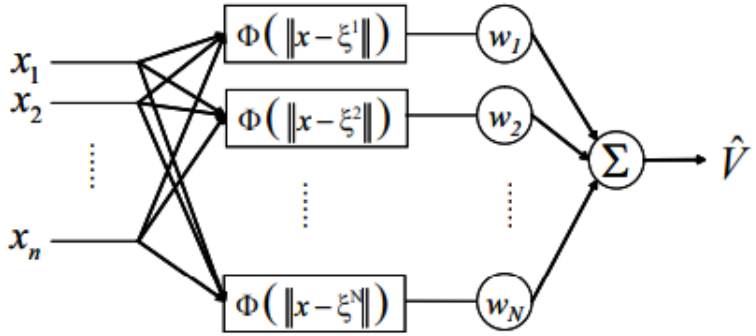
PIML in Control Loop: The Big Picture

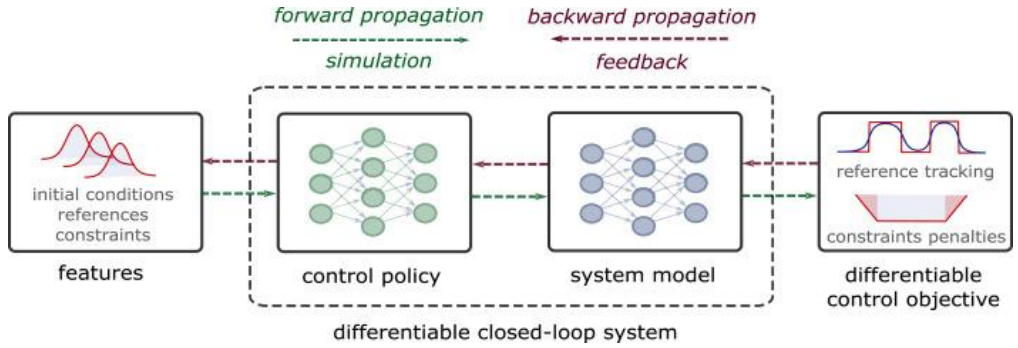
Implicit



Physical priors embedded into data-driven model

Physical priors embedded into control-oriented functions





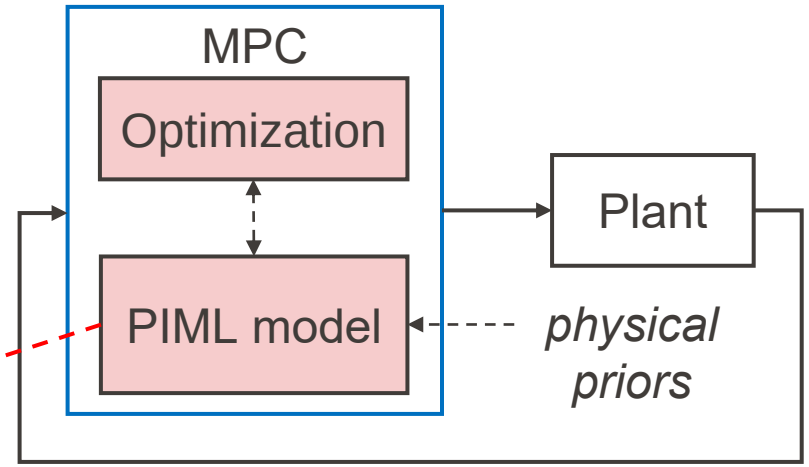
Physical priors embedded into control training

Explicit

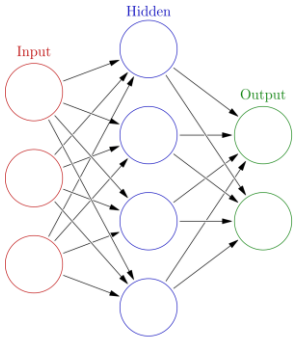
PIML Dynamic Models for MPC

- Employ PIML dynamic models in MPC
- Leverage physical priors through PIML models

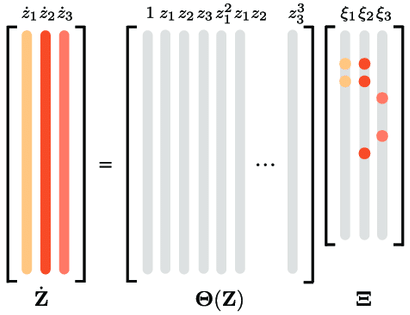
What data-driven model to use and how to incorporate it?



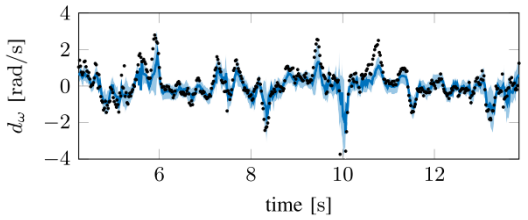
Physics-informed neural networks[1]

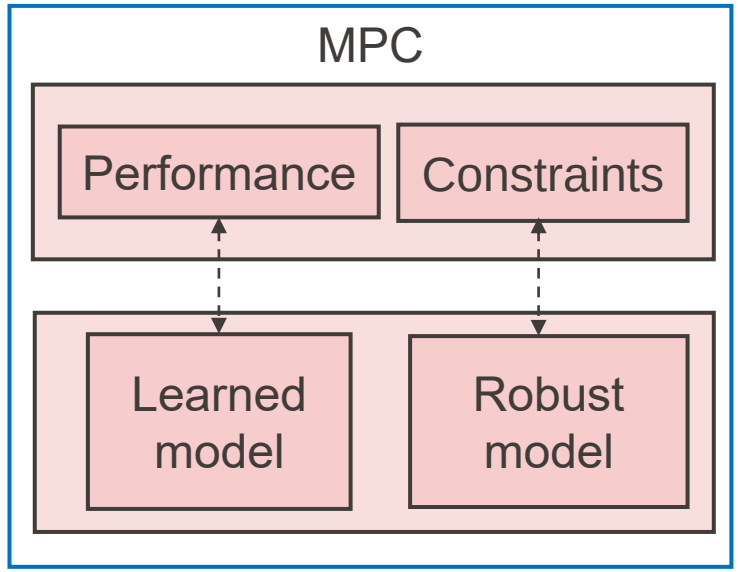


Sparse identification of nonlinear dynamics (SINDY-MPC)[2]



Gaussian Process disturbance learning[3]





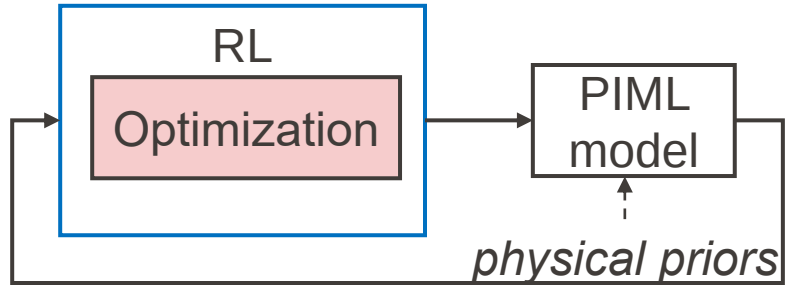
[1]J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed Neural Networks-based Model Predictive Control for Multi-link Manipulators," *IFAC-PapersOnLine*, vol. 55, no. 20, pp. 331–336, Jan. 2022

[2]E. Kaiser, J. N. Kutz, and S. L. Brunton, "Sparse identification of nonlinear dynamics for model predictive control in the low-data limit," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2219, p. 20180335, Nov. 2018.

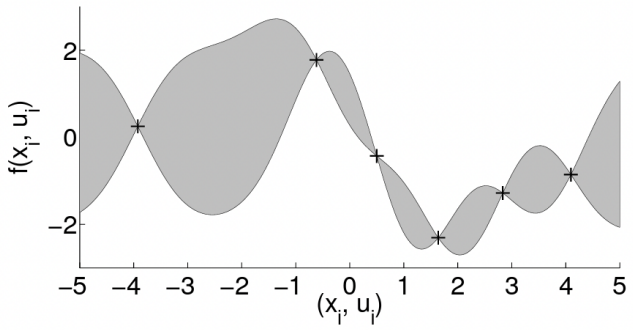
[3]L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2020

Physics-Informed Reinforcement Learning

- PIML model in model-based RL to improve accuracy & physical consistency
- Like MPC, physical priors are leveraged through PIML models



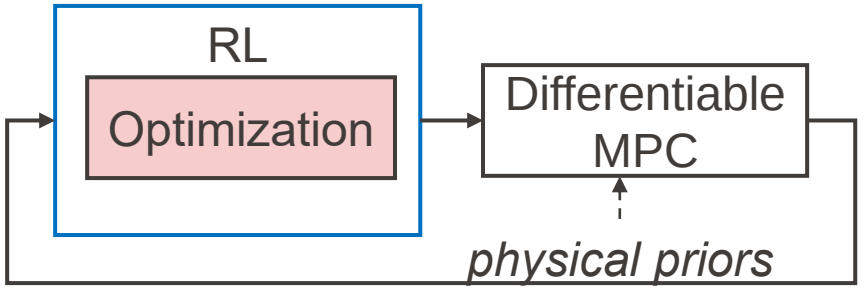
GP model reduces RL Training[4]



Physically consistent NN model[5]

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} f_{NN}(x_k, z_k) \\ f_{model}(x_k, z_k) \end{bmatrix}$$

MPC-based Value Function[6]



[4]M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Madison, 2011, p.465–472.

[5]L. D. Natale, B. Svetozarevic, P. Heer, and C. N. Jones, "Near-optimal Deep Reinforcement Learning Policies from Data for Zone Temperature Control," in *2022 IEEE 17th International Conference on Control & Automation (ICCA)*, Jun. 2022, pp. 698–703.

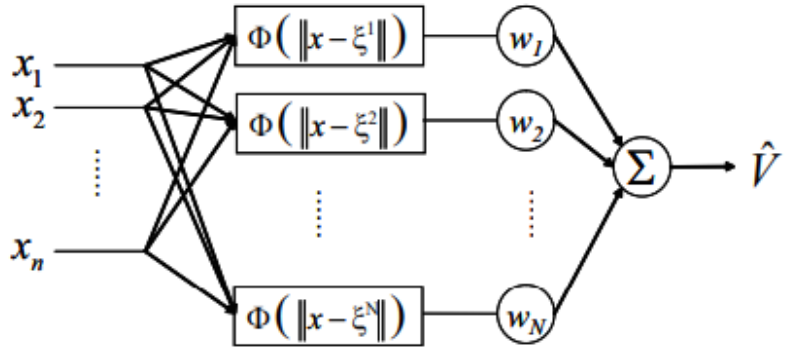
[6]M. Zanon and S. Gros, "Safe Reinforcement Learning Using Robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

Learning Lyapunov and Barrier functions

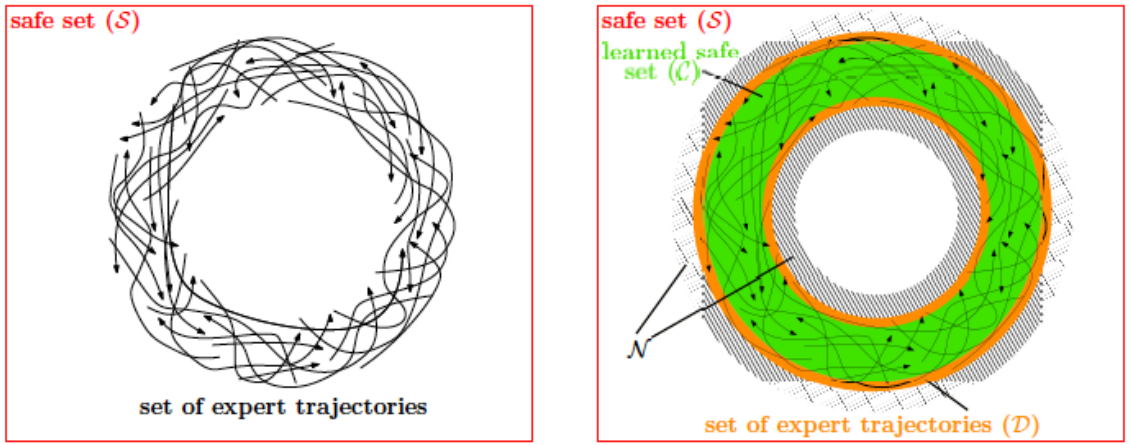
- Control Lyapunov functions (CLFs): stability
- Control barrier functions (CBFs): invariance
- How to construct CLF/CBFs?

$$u^*(x) = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \ \|u\|^2$$
$$\text{subject to:}$$
$$\nabla h^T\big(f(x) + g(x)\big)u \geq -\alpha(h(x))$$

Learning Stability from Data Samples[7]



Learning Invariance from Experts[8]

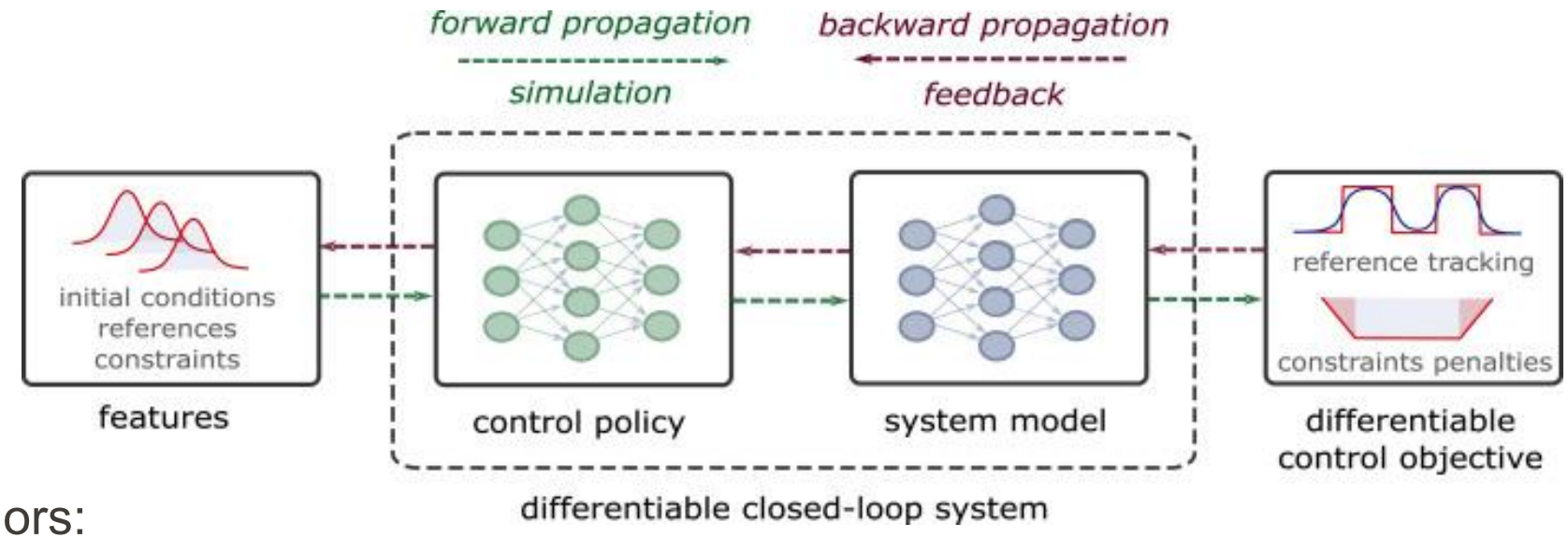


[7]T. Tsuzuki, T. Yagi, K. Yoshida, and Y. Yamashita, "A search algorithm for control Lyapunov functions using radial basis function networks," in *SICE Annual Conference 2011*, Sep. 2011, pp. 390–394.

[8]A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in 2020 59th *IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3717–3724.

Differentiable Control

- Use automatic differentiation (AD) to train NN-controllers with physics priors
- Training motivated by optimal control framework[9]



Physical Priors:

- System model, e.g., Lipschitz bounds, Dissipativity
- Constraint penalties, e.g., enforcing Lyapunov condition[10], reference tracking, state constraints

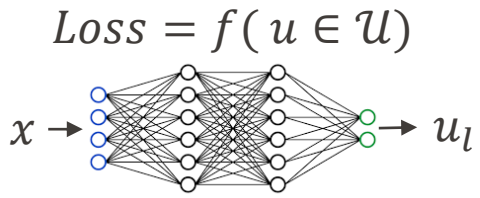
[9]J. Drgona, K. Kis, A. Tuor, D. Vrabie, and M. Klauˇco, "Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems," *Journal of Process Control*, vol. 116, pp. 80–92, 2022.

[10]S. Mukherjee, J. Drgoňa, A. Tuor, M. Halappanavar, and D. Vrabie, "Neural Lyapunov Differentiable Predictive Control," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 2097–2104.

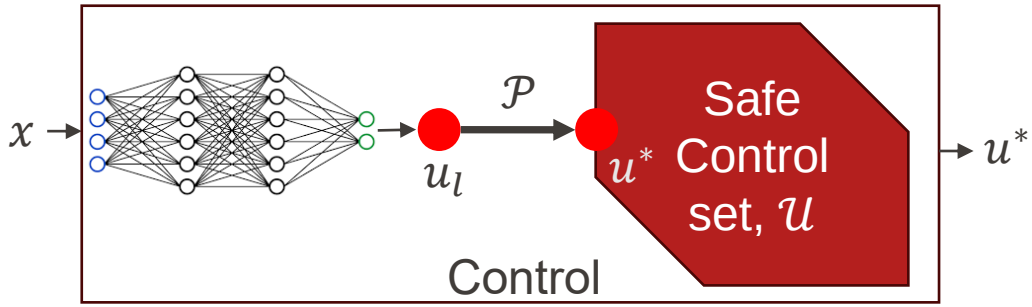
Safe Data-Driven Control

- Enforce safety/stability properties in controller training, implementation

Offline training with safety constraints

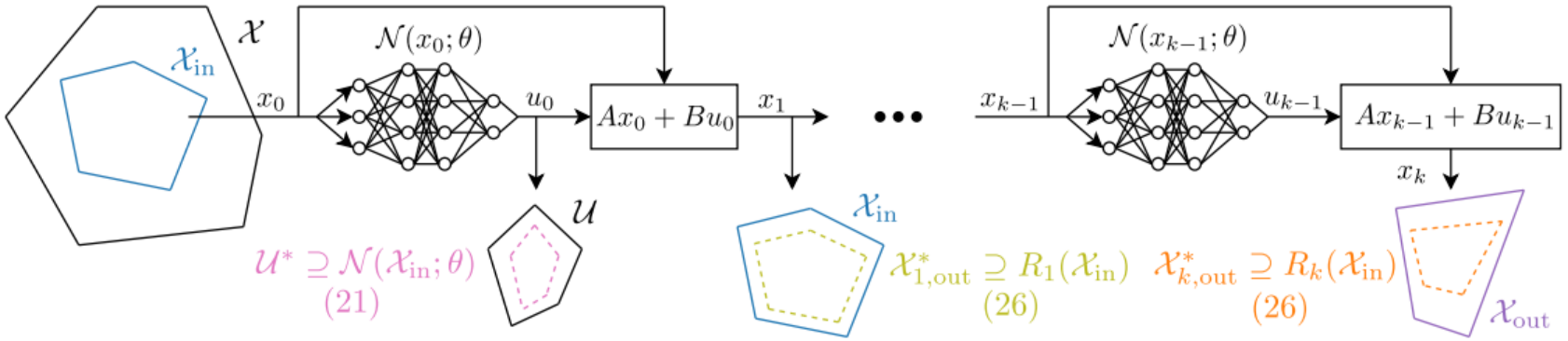


Online projection to enforce safety properties.



[11]W. Shaw Cortez, J. Drgona, A. Tuor, M. Halappanavar, and D. Vrabie, "Differentiable Predictive Control with Safety Guarantees: A Control Barrier Function Approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 932–938.

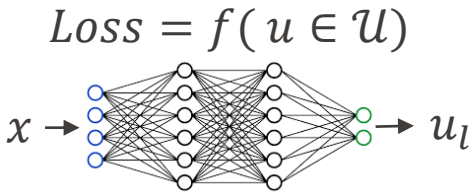
Train NN-based controllers that guarantee safety constraints, stability, etc.



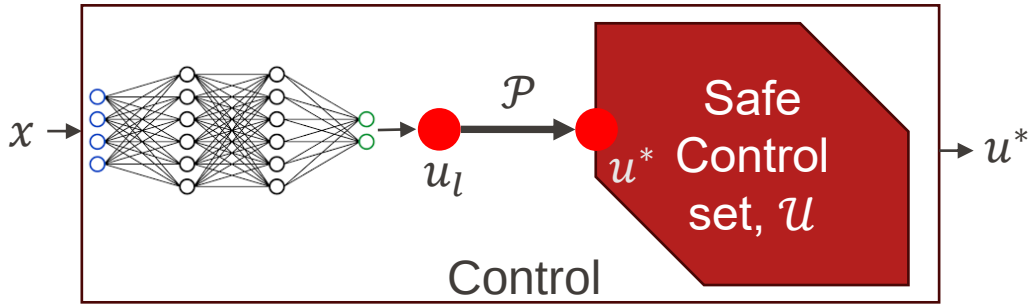
[12]B. Karg and S. Lucia, "Stability and feasibility of neural network-based controllers via output range analysis," in *2020 59th IEEE Conference on Decision and Control (CDC)*, Dec. 2020, pp. 4947–4954.

Case Study: Learning Control Policies with Safety Filter

Offline training with safety constraints



Online projection to enforce safety properties.



Differentiable Predictive Control

$$u_{DPC} = \pi_{W^*}(x)$$

$$\min_{\boldsymbol{W}} \frac{1}{mN} \sum_{i=1}^{m} \sum_{k=0}^{N-1} \left(\ell_{\text{MPC}}(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i, \boldsymbol{r}_k^i) + p_x(c(\boldsymbol{x}_k^i, \boldsymbol{p}_{c_k}^i)) + p_u(g(\boldsymbol{u}_k^i, \boldsymbol{p}_{g_k}^i)) \right) \quad Loss$$

$$\text{s.t. } \boldsymbol{x}_{k+1}^i = \boldsymbol{f}(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i, \hat{\boldsymbol{\theta}}_k^i),$$

$$\boldsymbol{u}_k^i = \pi_{\mathbf{W}}(\boldsymbol{x}_k^i, \boldsymbol{\xi}_k^i)$$

$$\boldsymbol{x}_0^i \in \mathbb{X} \subset \mathbb{R}^{n_x}$$

$$\boldsymbol{\xi}_k^i = \{\boldsymbol{r}_k^i, \boldsymbol{p}_{c_k}^i, \boldsymbol{p}_{g_k}^i, \hat{\boldsymbol{\theta}}_k^i\} \in \Xi \subset \mathbb{R}^{n_\xi}$$

Barrier-function Safety Filter

$$u^*(x) = \underset{u \in \mathbb{U}}{\text{argmin}} \ \|u - u_{DPC}(x)\|^2$$

subject to:

$$\nabla h^T \big(f(x) + g(x)\big) u \geq -\alpha(h(x)) + \big|\big|\nabla h\big|\big|\delta$$

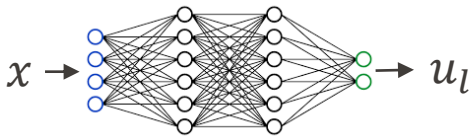
(Safe set: $\mathcal{C} = \{ x \in \mathbb{R}^n : \ h(x) \geq 0 \}$)

[11]W. Shaw Cortez, J. Drgona, A. Tuor, M. Halappanavar, and D. Vrabie, "Differentiable Predictive Control with Safety Guarantees: A Control Barrier Function Approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 932–938.

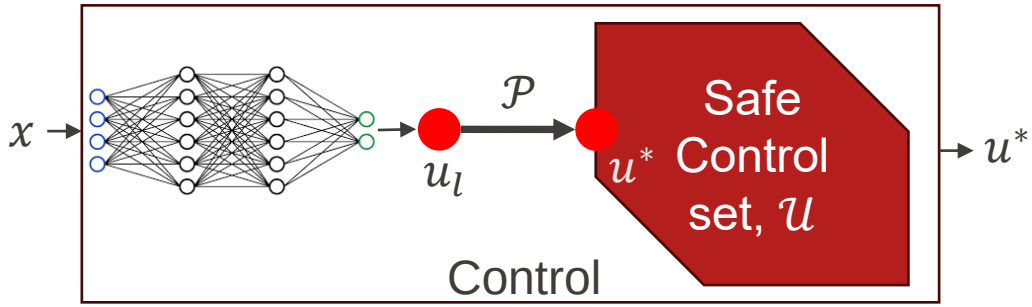
Case Study: Learning Control Policies with Safety Filter

Offline training with safety constraints

$$Loss = f(u \in \mathcal{U})$$



Online projection to enforce safety properties.



Differentiable Predictive Control

$$u_{DPC} = \pi_{W^*}(x)$$

$$\min_{\boldsymbol{W}} \frac{1}{mN} \sum_{i=1}^{m} \sum_{k=0}^{N-1} (\ell_{\text{MPC}}(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i, \boldsymbol{r}_k^i) + p_x(c(\boldsymbol{x}_k^i, \boldsymbol{p}_{c_k}^i)) + p_u(g(\boldsymbol{u}_k^i, \boldsymbol{p}_{g_k}^i)))$$

$$\text{s.t. } \boldsymbol{x}_{k+1}^i = \boldsymbol{f}(\boldsymbol{x}_k^i, \boldsymbol{u}_k^i, \hat{\boldsymbol{\theta}}_k^i),$$

$$\boldsymbol{u}_k^i = \pi_{\mathbf{W}}(\boldsymbol{x}_k^i, \boldsymbol{\xi}_k^i)$$

$$\boldsymbol{x}_0^i \in \mathbb{X} \subset \mathbb{R}^{n_x}$$

$$\boldsymbol{\xi}_k^i = \{\boldsymbol{r}_k^i, \boldsymbol{p}_{c_k}^i, \boldsymbol{p}_{g_k}^i, \hat{\boldsymbol{\theta}}_k^i\} \in \Xi \subset \mathbb{R}^{n_\xi}$$

$Loss$

Incorporate into training

Barrier-function Safety Filter

$$u^*(x) = \text{argmin } \|u - u_{DPC}(x)\|^2$$

$$u \in \mathbb{U}$$

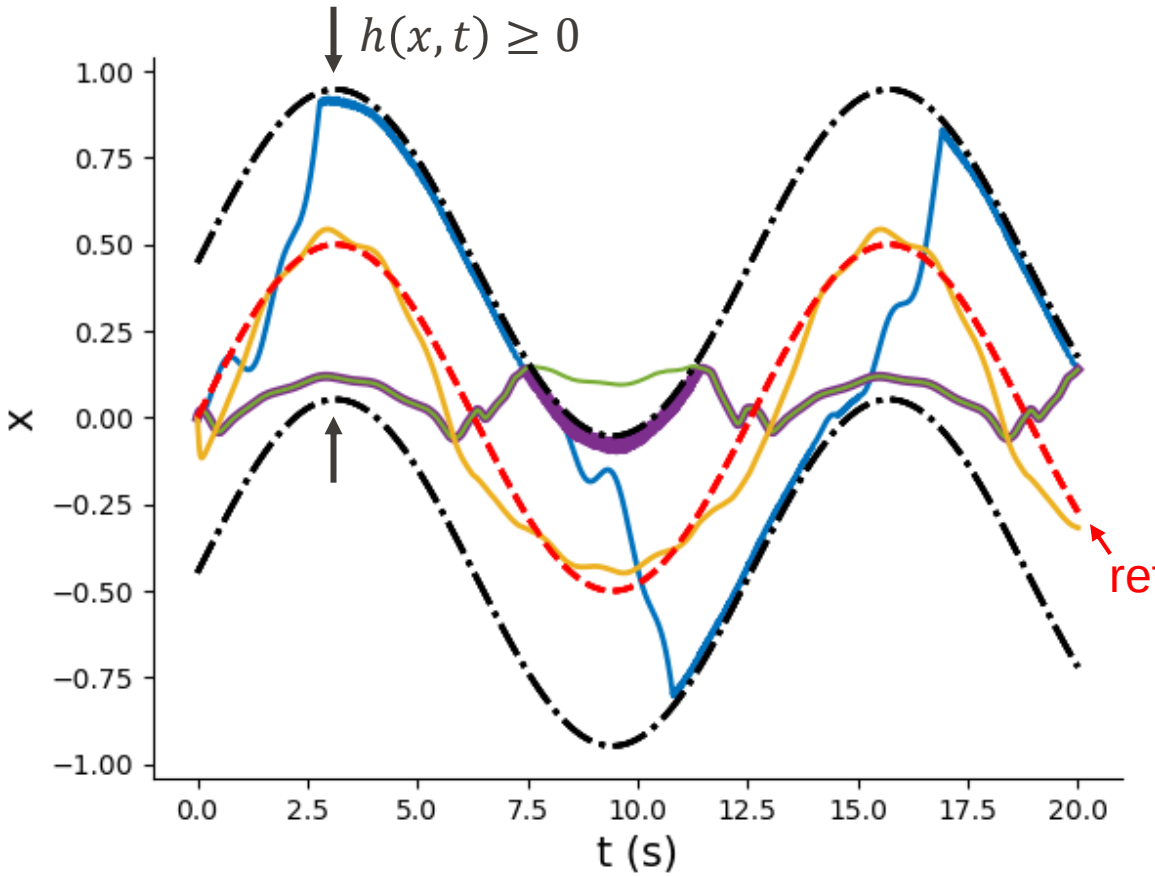
subject to:

$$\nabla h^T \big(f(x) + g(x)\big)u \geq -\alpha(h(x)) + \big|\big|\nabla h\big|\big|\delta$$

(Safe set: $\mathcal{C} = \{ x \in \mathbb{R}^n : \ h(x) \geq 0 \}$)

[11]W. Shaw Cortez, J. Drgona, A. Tuor, M. Halappanavar, and D. Vrabie, "Differentiable Predictive Control with Safety Guarantees: A Control Barrier Function Approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 932–938.

Case Study: Learning Control Policies with Safety Filter



- Barrier only ($u_{DPC} \coloneqq 0$)
- DPC only (trained with incorrect reference)
- ZCBF+DPC (trained with incorrect reference)
- ZCBF+DPC (trained with correct reference)

Dynamics

$$\dot{x} = Ax + u + w(t)$$

Barrier function:

$$h(x,t) := \varepsilon - \| x - x_r(t) \|_2^2$$

Reference trajectory

Safety filter:

$$u^*(x) = \underset{u \in \mathbb{U}}{\operatorname{argmin}} \ \|u - u_{DPC}(x)\|^2$$

subject to:

$$\nabla h^T\big(f(x) + g(x)\big)u \geq -\alpha(h(x)) + \big|\big|\nabla h\big|\big|\delta$$

https://github.com/pnnl/neuromancer

[11]W. Shaw Cortez, J. Drgona, A. Tuor, M. Halappanavar, and D. Vrabie, "Differentiable Predictive Control with Safety Guarantees: A Control Barrier Function Approach," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 932–938.

Analysis and Verification of ML Models

Colin Jones

Physics Informed, not Physics Enforced

Many PIML methods are “encouraging” physics, rather than “enforcing” it

Post-training verification required of key properties Lyapunov, passivity, invariance, etc

Verification & Analysis in Control

System Dynamics and/or Controlled Systems

- Stability / dissipativity / region of attraction
- Lipschitz properties
- Robustness
- Safety properties (constraint satisfaction)

Many people active in the area…

[Zhang et al, 2014], [Engelken et al, 2020], [Vogt et al, 2022], [Batuhan et al, 2019], [Drgona et al, 2022], [Abate et al, 2021], [Grüne et al, 2021], [Chen et al, 2021], etc

[Scaman et al, 2018], [Chakrabarty et al, 2019], [Pauli et al, 2022], [Fazlyab et al, 2022], [Erichson et al, 2021], etc

[Wang et al, 2016], [Gehr et al, 2018], [Jia et al, 2020], [Tjeng et al, 2019], etc

[Zhang et al, 2022], [Robey et al, 2020], [Karg et al, 2020], etc

Verification & Analysis in Control

System Dynamics and/or Controlled Systems

- Stability / dissipativity / region of attraction
- Lipschitz properties
- Robustness
- Safety properties (constraint satisfaction)

Many approaches being studied

- Conservative over-bounding
  - Lipschitz constants
  - GP bounds
- Sampling-based methods
- Function positivity / global optimziation
  - Mixed-integer programming
  - SOS / Semi-definite programming

Many people active in the area…

[Zhang et al, 2014], [Engelken et al, 2020], [Vogt et al, 2022], [Batuhan et al, 2019], [Drgona et al, 2022], [Abate et al, 2021], [Grüne et al, 2021], [Chen et al, 2021], etc

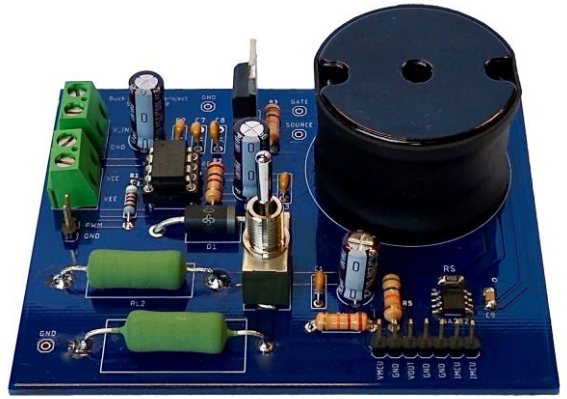
[Scaman et al, 2018], [Chakrabarty et al, 2019], [Pauli et al, 2022], [Fazlyab et al, 2022], [Erichson et al, 2021], etc

[Wang et al, 2016], [Gehr et al, 2018], [Jia et al, 2020], [Tjeng et al, 2019], etc

[Zhang et al, 2022], [Robey et al, 2020], [Karg et al, 2020], etc

Standard MPC Design Process

DC-DC Buck Converter

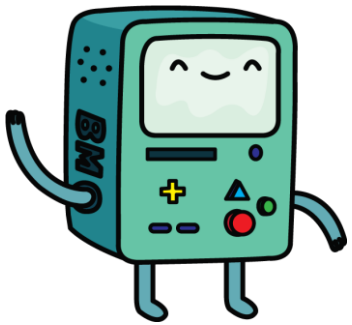


Average dynamics over a switching period, then linearize

$$
\begin{aligned}
\min_{X,U} \quad & \sum_{t=0}^{N-1} \left(\|x_t - x_{\text{eq}}\|_Q^2 + \|u_t - u_{\text{eq}}\|_R^2 \right) + \|x_N - x_{\text{eq}}\|_P^2 \\
\text{s.t.} \quad & \forall t = 0, \ldots, N-1 \\
& x_{t+1} = Ax_t + Bu_t \\
& \begin{bmatrix} I_L^{\min} \\ V_O^{\min} \end{bmatrix} \leq x_t \leq \begin{bmatrix} I_L^{\max} \\ V_O^{\max} \end{bmatrix} \\
& u^{\min} \leq u_t \leq u^{\max} \\
& x_N \in \mathcal{X}_N \\
& x_0 = x(0)
\end{aligned}
$$

State Estimator + Offset-free Tracking MPC

Control frequency of 10 kHz!



STM32L476:

- 80 MHz clock
- 128 kB of RAM
- 1 MB of flash

25-dimensional QP with five parameters

Control Approximation via Deep NN

$$z^\star(x) = \min \|Lz + Fx + f\|^2 + \epsilon\|z\|^2$$
$$\text{s.t. } z \geq 0$$
$$\hat{u}(x) = Gz^\star(x) + g$$

Choose parametric QP structure
→ Approximation of the dual of a standard linear MPC controller

Affine layer $y_1 = Fx + f$ — F, f

QP layer $y_2 = \min_{z \geq 0} \|Lz + y_1\|^2 + \epsilon\|z\|^2$ — L

Affine layer $y_3 = Gy_2 + g$ — G, g

Saturation layer $\hat{u} = \text{sat}(y_3)$

Optimal solution is differentiable
[Amos et al, 2017], [Agrawal et al, 2019]

Differentiable Programming → Approximate Explicit MPC

MPC controller formulation

$$\min_{X,U} \sum_{t=0}^{N-1} (x_t^\top Q x_t + u_t^\top R u_t) + x_N^\top P x_N$$

s.t. $\forall t = 0, \dots, N-1$

$x_{k+1} = Ax_k + Bu_k$

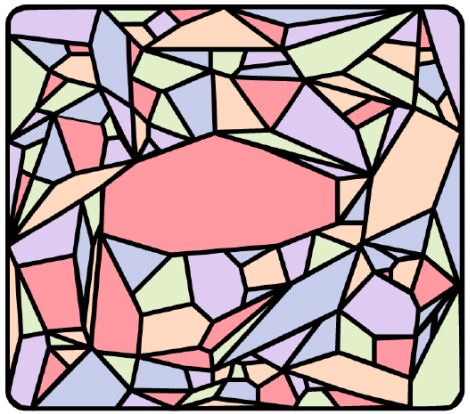
$x^{\min} \leq x_k \leq x^{\max}$

$u^{\min} \leq u_k \leq u^{\max}$

$x_N \in X_N$

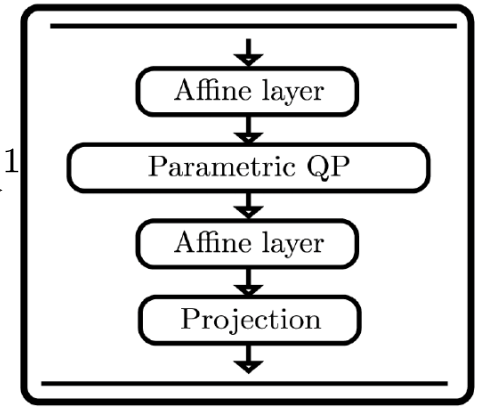
$x_0 = x(0)$

Complex PWA function

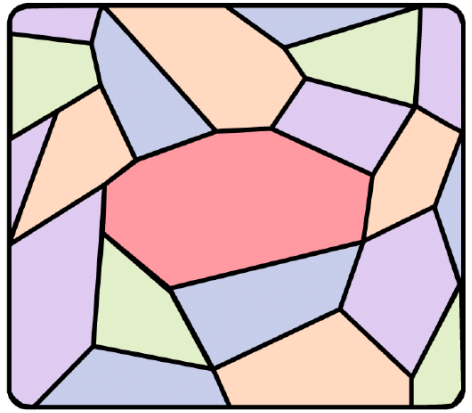


$\{(x_i, u_i)\}_{i=1}^{D}$

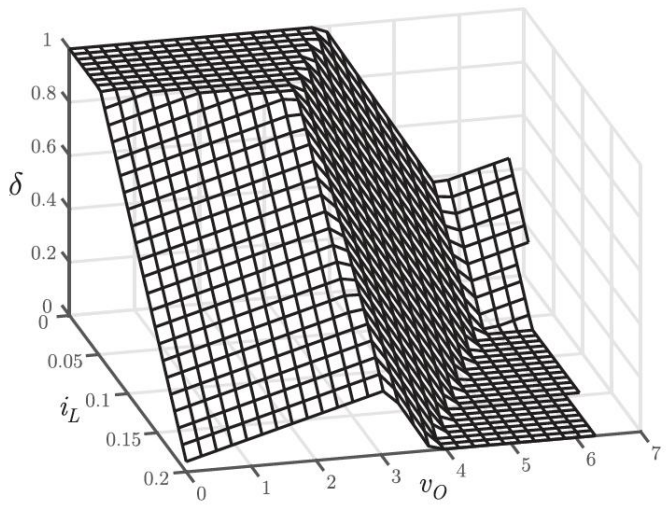
Neural Network



Low complexity PWA function

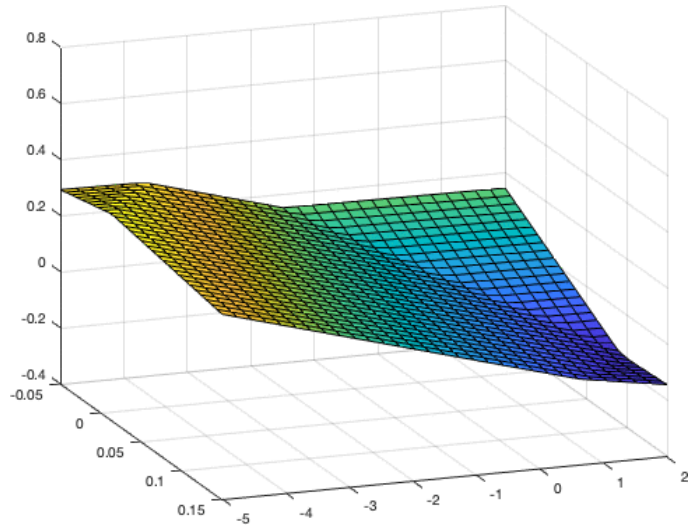


Original controller



91% complexity reduction

Training : 35min



(2021) Embedded PWM Predictive Control of DC-DC Power Converters Via Piecewise-Affine Neural Networks. IEEE OJIES.
E.T. Maddalena, M.W.F. Specq, V.L. Wisniewski, C.N. Jones.

Verification of Approximate Controllers ↔ Function Positivity

Verify that the approximation error is nowhere greater than γ

$$0 \leq \gamma - \max_x \|u^\star(x) - \hat{u}(x)\|_\infty$$

Optimal control law

Approximate control law

Verification → Proving non-negativity of a function

Verification of Approximate Controllers ↔ Function Positivity

Verify that the approximation error is nowhere greater than γ

$$0 \leq \gamma - \max_x \|u^\star(x) - \hat{u}(x)\|_\infty$$

$$\text{s.t. } u^\star(x_0) = \arg\min \sum_{i=0}^{N} l(x_i, u_i)$$
$$\text{s.t. } \forall i = 0, \ldots, N-1$$
$$x_{i+1} = Ax_i + Bu_i$$
$$x_i \in X,\ u_i \in U$$
$$x_N \in X_N,\ x_0 = x$$

Optimal control law

$$z^\star(x) = \min \|Lz + Fx + f\|^2 + \varepsilon\|z\|^2$$
$$\text{s.t. } z \geq 0$$
$$\hat{u}(x) = Gz^\star(x) + g$$

Approximate control law

Verification → Proving non-negativity of a function

MILP-Representable Functions

A function $\psi : X \to V$ with domain $X \subseteq \mathbb{R}^n$ and range $V \subseteq \mathbb{R}^m$ is MILP representable if there exists a polyhedral set $P \subseteq \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^c \times \mathbb{R}^b$ such that $v = \psi(x)$ if and only if there exists $z \in \mathbb{R}^c$ and $\beta \in \{0,1\}^b$ such that $(x, v, z, \beta) \in P$.

MILP representable functions are dense in the family of continuous functions on a compact domain

Verification Problem

$$\begin{aligned} \min\ & \tau \\ \text{s.t. } & x_0 \in X \\ & v_1 = \psi_1(x_0) \\ & v_2 = \psi_2(x_0) \\ & \tau = f(x_0, v_1, v_2) \end{aligned}$$

Mixed-Integer Optimization Problem

$$\begin{aligned} \min\ & \tau \\ \text{s.t. } & x_0 \in X \\ & (x_0, v_1, z_1, \beta_1) \in P_1 \\ & (x_0, v_2, z_2, \beta_2) \in P_2 \\ & (v_1, v_2, \tau, z_3, \beta_3) \in P_f \\ & \beta_i \in \{0,1\}^{b_i} \end{aligned}$$

R.Schwan, C.N. Jones and D.Kuhn, Stability Verification of Neural Network Controllers using Mixed-Integer Programming, IEEE Transactions on Automatic Control. 2023, To appear

Some Examples

Compositions

$$\psi_1 \circ \psi_2$$

Max / min

$$\psi(x) = \max\{\psi_1(x), \psi_2(x)\}$$

Piecewise Affine Functions

$$\psi(x) = A_i x + c_i \quad \forall x \in X_i, \ \forall i \in \mathcal{I}$$

Parametric convex QPs

$$\begin{aligned} \psi(x) = \arg\min_z \ & \frac{1}{2} z^T P z + (Qx + q)^T z \\ \text{s.t. } & Az = Bx + b \\ & Fz \leq Gx + g \end{aligned}$$

MILP-Representable Functions → Mixed-integer programming

Verification Problem

$$
\begin{aligned}
&\min \ \tau \\
&\text{s.t. } x_0 \in X \\
&\quad v_1 = \psi_1(x_0) \\
&\quad v_2 = \psi_2(x_0) \\
&\quad \tau = f(x_0, v_1, v_2)
\end{aligned}
$$

Various approximate control laws can be captured

- Deep neural networks
- Quadratic programs
- Simple saturations
- etc

Verification of various properties can be made

- Lyapnuov function
- Worst-case error
- Region of attraction
- etc

A number of people have developed similar formulations for polynomial control laws and SOS solvers

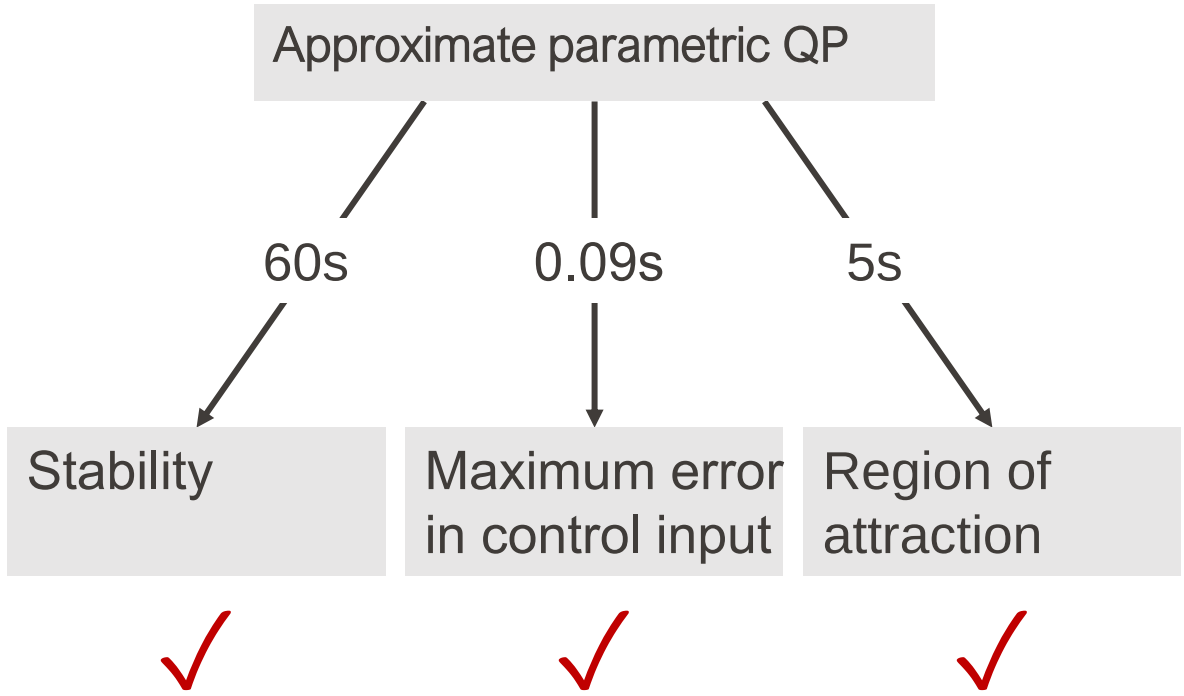
EVANQP

DOI 10.48550/arXiv.2206.13374 Preprint arXiv Grant NCCR Automation (51NF40180545)

EPFL Verifier for Approximate Neural Networks and QPs

https://github.com/PREDICT-EPFL/evanqp

Software tool to train approximate controllers using neural networks and/or parametric quadratic programs, and then to verify key properties.



A similar tool targeted at power systems: *[Venzke, Qu, Low, Chatzivasileiadis, 2020]*

Verification & Analysis in Control

System Dynamics and/or Controlled Systems

- Stability / dissipativity / region of attraction
- Lipschitz properties
- Robustness
- Safety properties (constraint satisfaction)

Many approaches being studied

- Conservative over-bounding
  - Lipschitz constants
  - GP bounds
- Sampling-based methods
- Function positivity / global optimziation
  - Mixed-integer programming
  - SOS / Semi-definite programming

Many people active in the area…

[Zhang et al, 2014], [Engelken et al, 2020], [Vogt et al, 2022], [Batuhan et al, 2019], [Drgona et al, 2022], [Abate et al, 2021], [Grüne et al, 2021], [Chen et al, 2021], etc

[Scaman et al, 2018], [Chakrabarty et al, 2019], [Pauli et al, 2022], [Fazlyab et al, 2022], [Erichson et al, 2021], etc

[Wang et al, 2016], [Gehr et al, 2018], [Jia et al, 2020], [Tjeng et al, 2019], etc

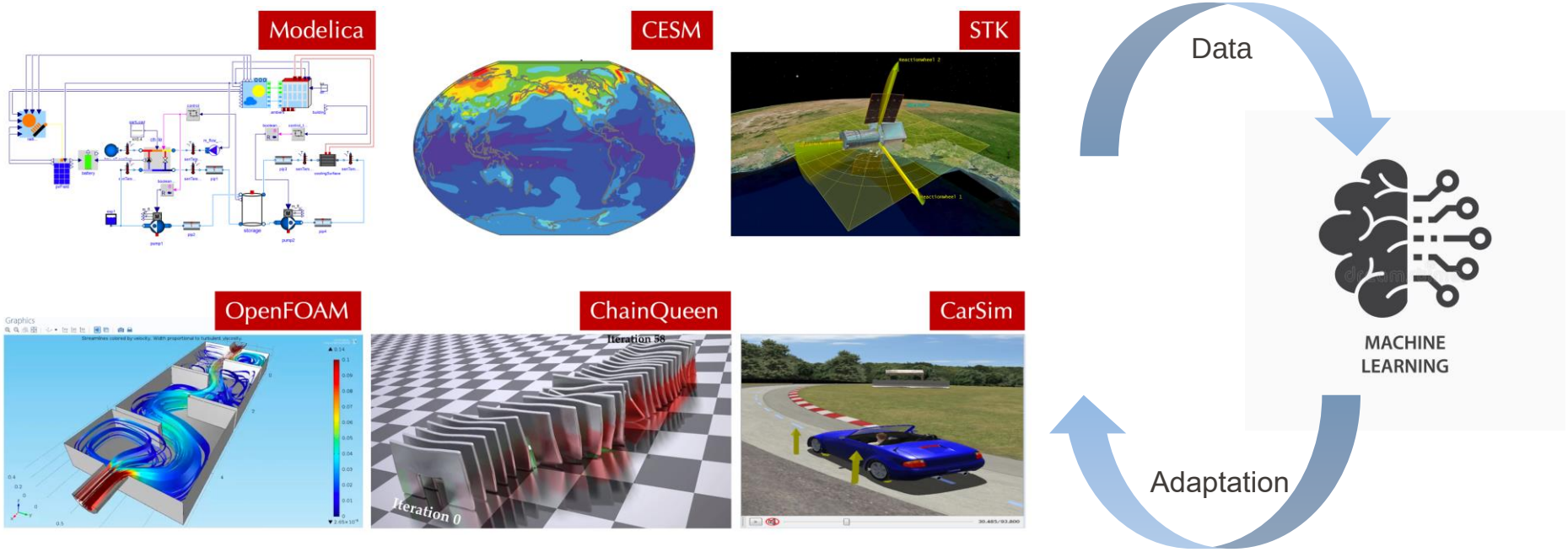
[Zhang et al, 2022], [Robey et al, 2020], [Karg et al, 2020], etc

Physics-Informed Digital Twins

Ankush Chakrabarty

Physics-Informed Digital Twins

- Digital twins becoming widely available across many different domains
  - *Defn.*: Physics-informed modeling toolkits and simulation environments **containing learning modules capable of adapting to operational data** to maintain high prediction accuracy over product lifetime



- Provides a **rapid**, **scalable**, **safe**, and **repeatable** alternative to field experiments
- May not be entirely transparent
  - (*this part of talk*) Critical to design ML methods that can leverage **physics-informed simulation data** for:
    - Controller design and online performance optimization
    - Improving generalization via multi-source data

Calibrating (Tuning) Controllers from Simulations

- Control algorithms (e.g. MPC) have a few tunable parameters, say $x \in X$:
  - Model: unknown parameters, scenario tree
  - Constraints: backoff values, terminal constraints
  - Objective: weights in stage cost, horizon, terminal cost
  - Solvers: tolerance, discretization scheme
  - References: set-points, path trajectories

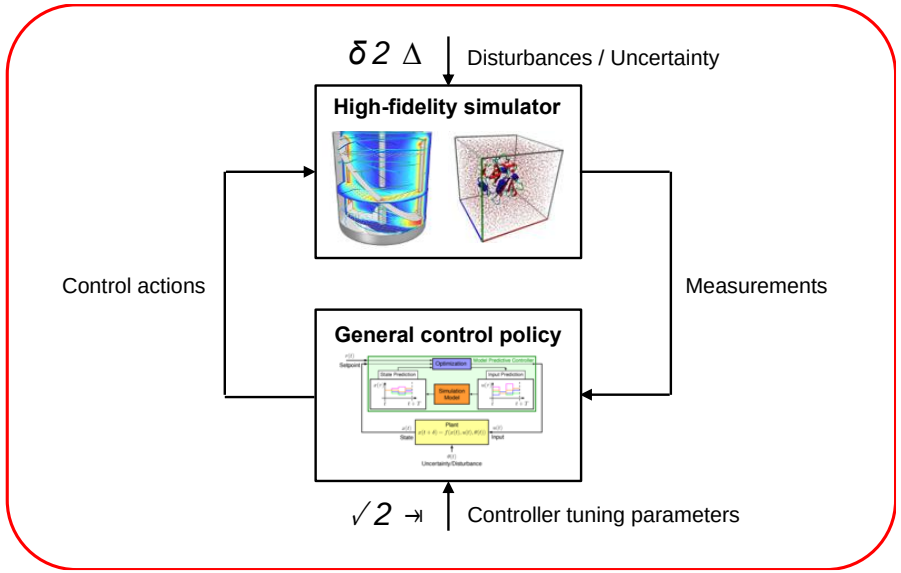
- Must define performance index $f(x)$ over closed-loop simulation: f may be nasty (= expensive, no gradients)

- Auto-tuning = global optimization problem, $\min_{x \in X} f(x)$

- **Bayesian Optimization (BO)** is an effective auto-tuning strategy when working directly with simulation data

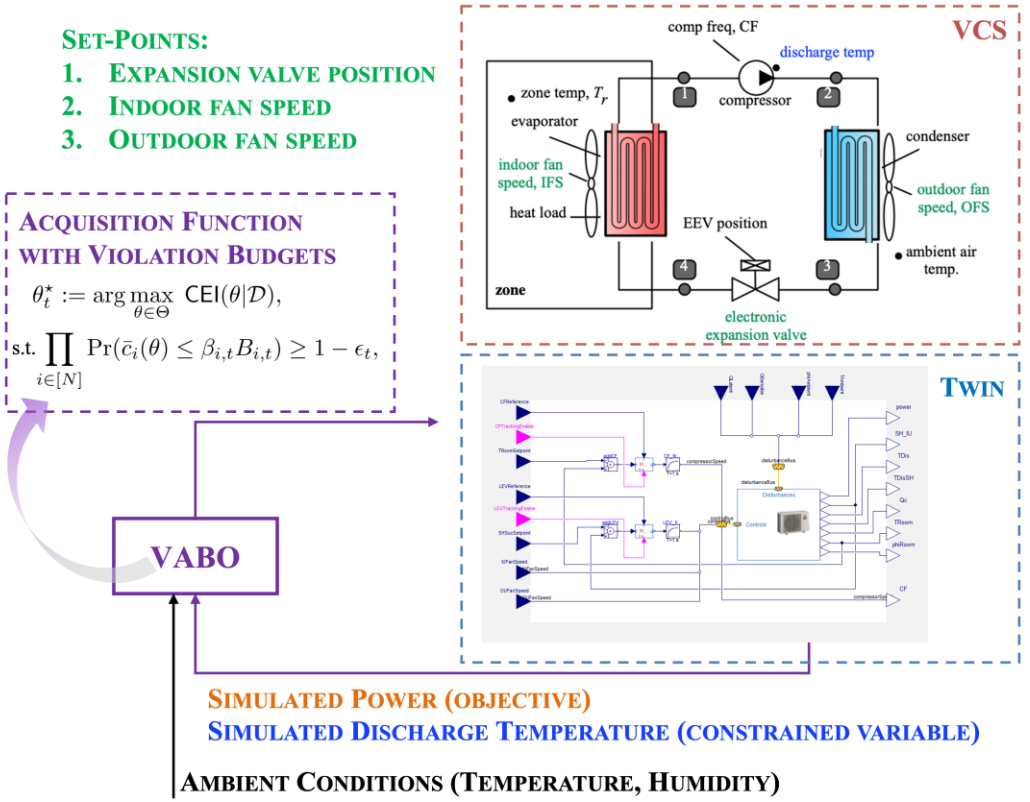
Closed-loop performance measure

$$\min_{\checkmark 2 \twoheadrightarrow} E\{f(\checkmark, \delta)\}$$



Bayesian Optimization (BO)

- BO **does not require any knowledge of structure** and/or gradient of f and can tolerate **noisy** observations
  - Flexible
- BO avoids getting "stuck" in local solutions (**global optimization**)
  - Can "teleport" throughout design space
- BO requires **few evaluations** and concurrently **learns a surrogate model** of the CL performance
- Check out the paper for a real-world use-case of BO-based safe controller tuning in vapor compression systems!



BO is Extremely Flexible & Can Exploit Structure!

- Input-dependent noise $f(x) = g(x) + \varepsilon(x)$

  [Kersting et al., 2007; Lazaro-Gredilla et al., 2011]

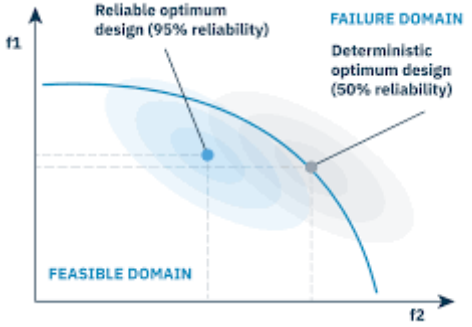
- Robust optimization $f(x) = \max_{w \in \mathcal{W}} g(x, w)$

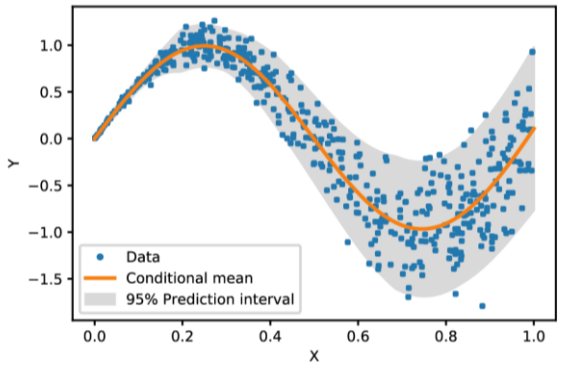
  [Paulson et al., 2021; Kudva et al. 2022]

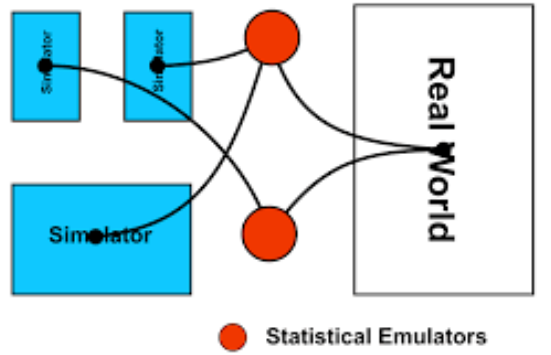
- Multi-objective & constraints

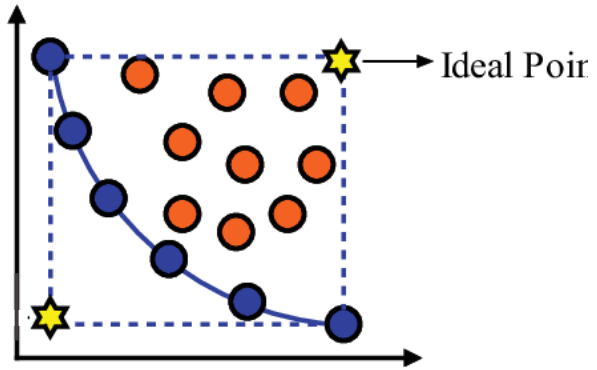
$$f(x) \quad \text{s.t.} \quad c(x) \leq 0$$

[Makrygiorgos et al., 2022; Lu et al., 2022, Xu et al. 2022]





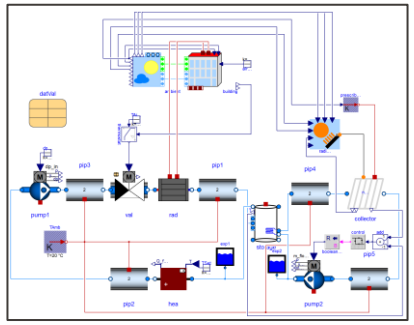




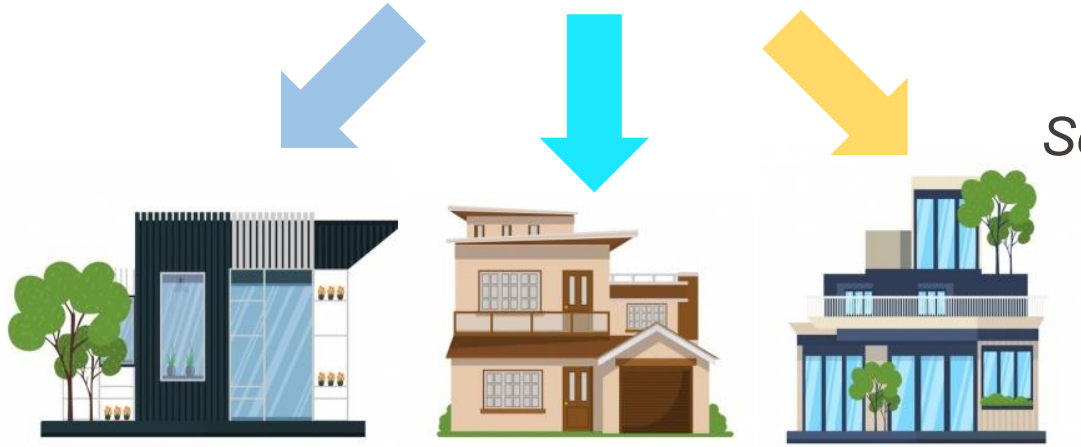
Simulation Environments Enable Generation of Multi-Source Data

- PI digital twins comprise **parameterized components** that can be modified to generate useful data from multiple source systems similar to the target system
  - This **multi-source dataset**:
    - Reduces data requirements from one system by harnessing data from similar systems

- Specific ML frameworks are suited to learning from multi-source data:
  - **Meta-learning and transfer-learning**



Simulator

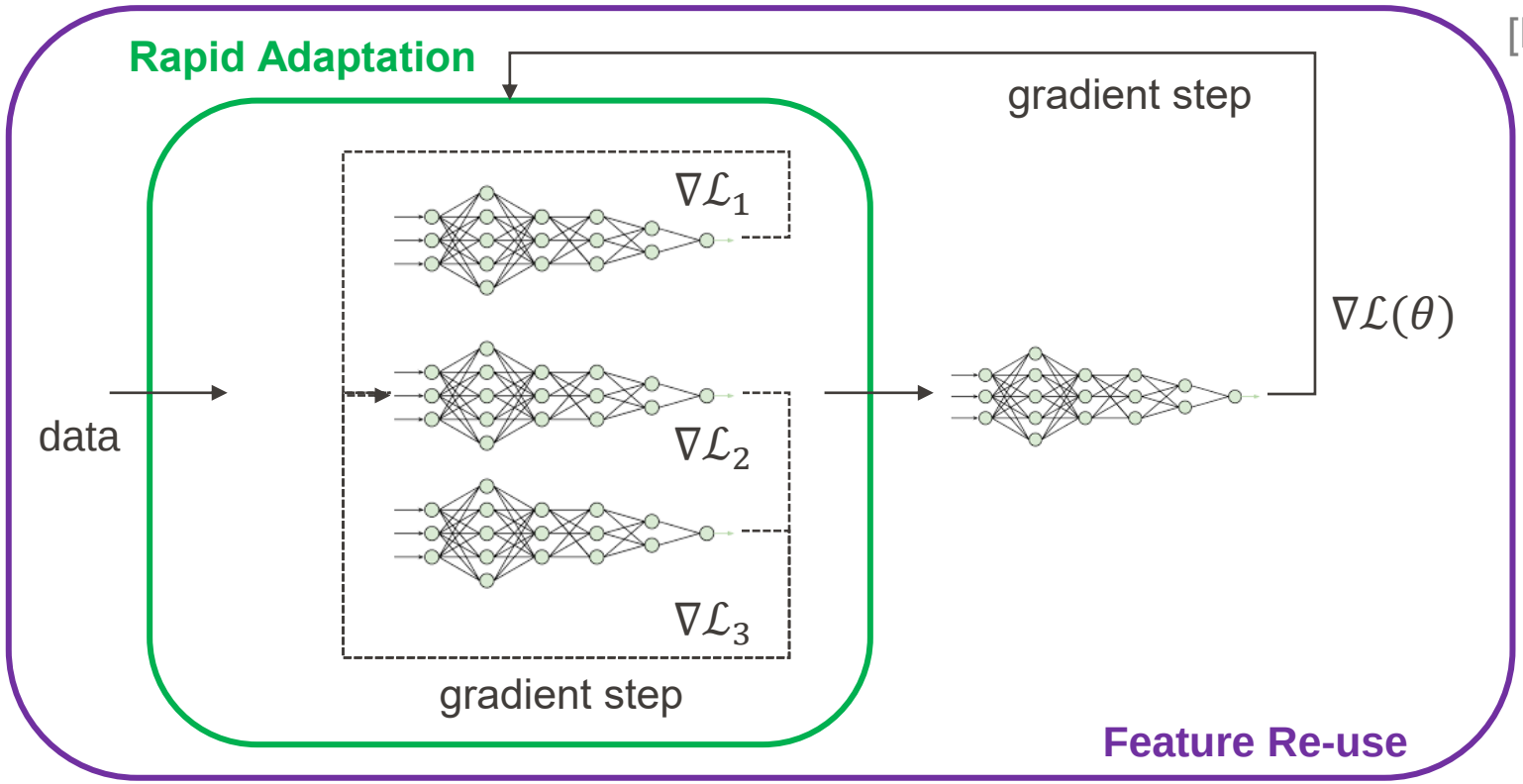


Sources

Targets

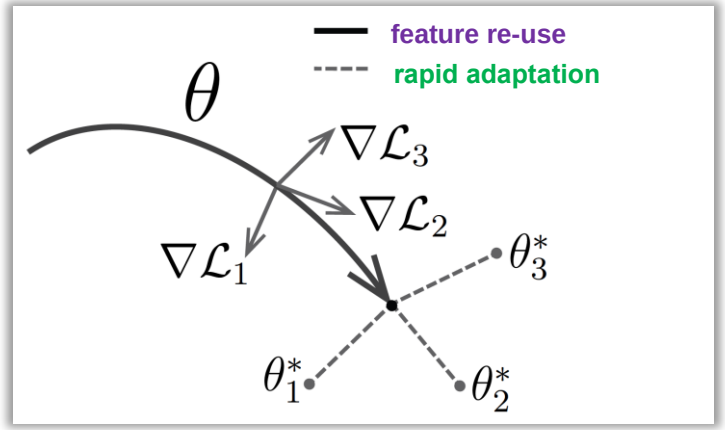
Meta-Learning

- Classical meta-learners rely on 2 training loops:
  - A feature re-use loop for learning features useful over all source data i.e., task-independent
  - A rapid adaptation loop for network to perform well on single task i.e., task-specific
  - The **adaptation is "baked in"** to the training procedure: this will be exploited at inference



[Finn et al., 2017; Raghu et al., 2019]

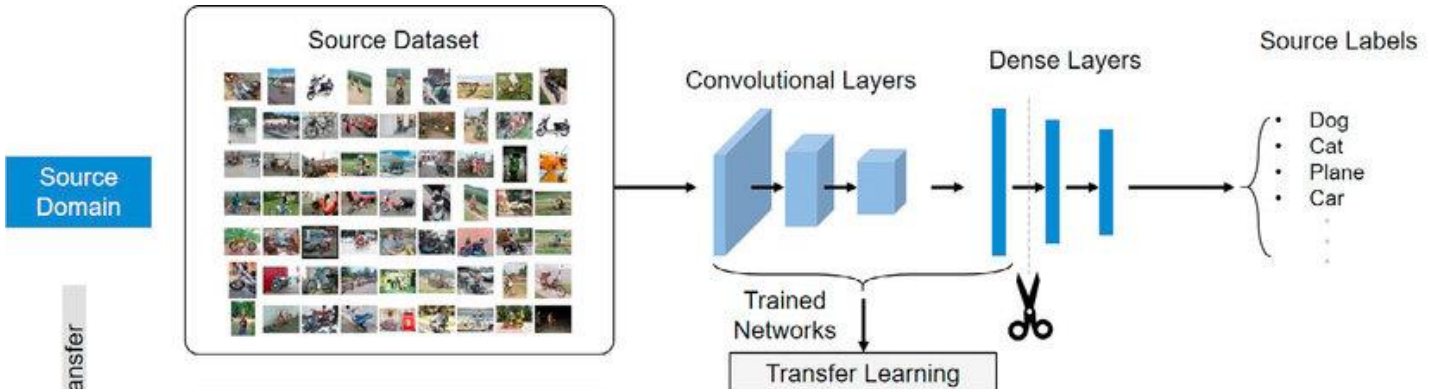
Source: bair.berkeley.edu



Transfer-Learning

- Transfer learning involves two separate single-loop training procedures
  - 1st training phase: learn task-independent representations
  - 2nd training phase: pre-trained network **modified** and **fine-tuned** for the target task

  - Unlike meta-learning:
    - Adaptation is made problem-specific by 2nd phase, not baked in
    - 2nd-phase training not necessarily few-shot (offline)
    - Same network may not be used at training v. inference



Snapshot of Multi-Source Learning for Control

- Meta-learning for model identification:
  - Meta-GP for MPC
  - Meta-learned state-space models

[Arcari et al., 2022]

[Chakrabarty et al., 2023]

- Meta-learning for controller and parameter tuning:
  - Meta-learned BayesOpt
  - Meta-learned adaptive control

[Chakrabarty, 2022; Zhan et al., 2022]

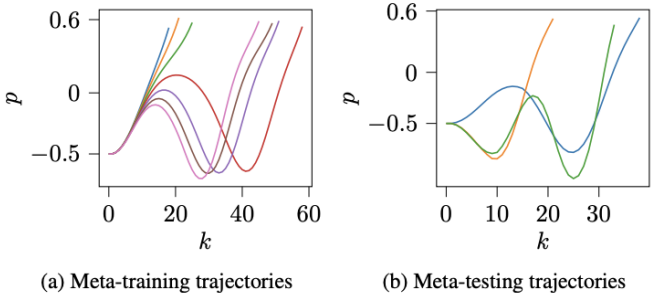
[Richards et al., 2021; Richards et al., 2022]

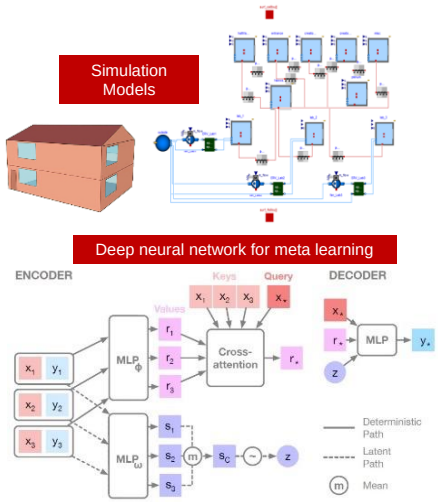
- Transfer-learning for modeling and control

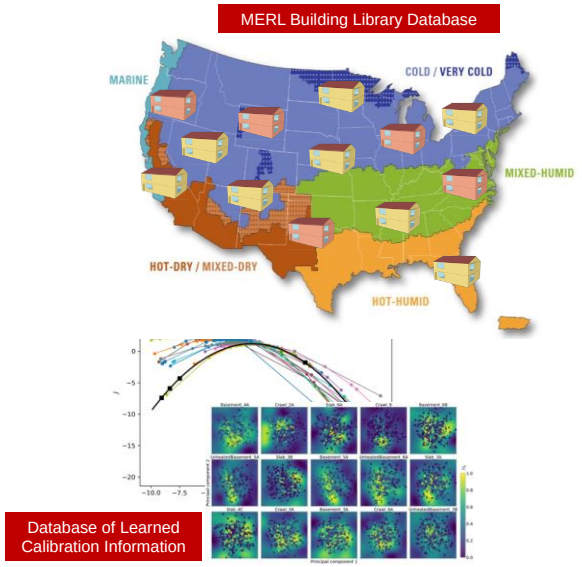
[Chen et al., 2020; Xu et al., 2020]

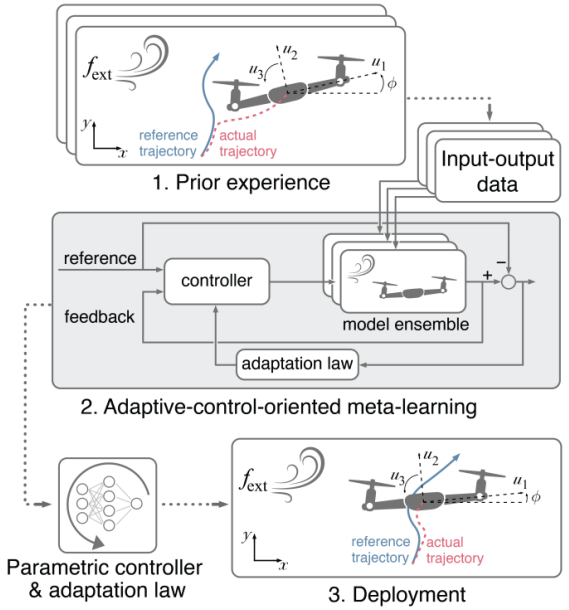
- Multi-fidelity auto-tuning

[Sorourifar et al., 2021]









Opportunities for PIML4C

1. Modeling of human(-in-the-control-loop) behaviors
2. PIML for scalable, interaction-aware controllers for multi-component systems
3. Interpretable and verifiable explicit MPC policies via physics injection during learning
4. Providing safety and performance guarantees – new theoretical tools
5. Automated adaptation via simulation (closing sim2real gap)
6. Integration of multimodal signals (speech, image/video) into control loops

Challenges for PIML4C

1. Quantification of ‘minimal’ = ‘useful’ and ‘useful’ data requirements
2. Quantification of uncertainty and modeling errors
3. Quantification of similarity in multi-source learning
4. Formal guarantees of safety/stability for closed loop PIML based controllers
5. Scaling up of verification methods

Summary & Conclusions

- Classically, ML methods do not integrate known physical constraints
- PIML methods have emerged to systematically combine data-driven ML with physics-based modeling
- PIML methods offer new and exciting opportunities for human-in-the-loop, multi-scale and multi-physics systems
- Open challenges remain: **an exciting time to solve problems in PIML!**