Safe Physics-Informed Learning in Dynamics and Control

Ján Drgoňa
Associate Professor
Civil and Systems Engineering Department
The Ralph O'Connor Sustainable Energy Institute

Electrical Engineering and Systems Science > Systems and Control

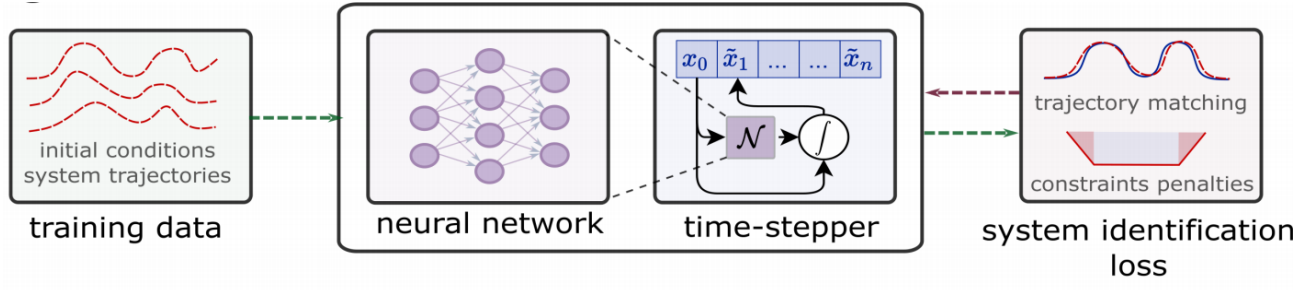
[Submitted on 17 Apr 2025 (v1), last revised 12 Jun 2025 (this version, v2)]

Safe Physics-Informed Machine Learning for Dynamics and Control

Jan Drgona, Truong X. Nghiem, Thomas Beckers, Mahyar Fazlyab, Enrique Mallada, Colin Jones, Draguna Vrabie, Steven L. Brunton, Rolf Findeisen

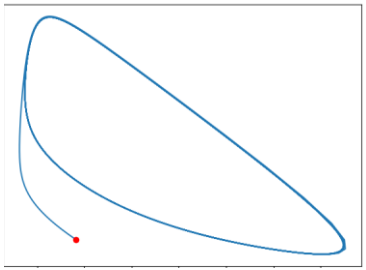
arXiv:2504.12952

Learning to Model (L2M) Dynamical Systems



Dataset: batched time-series of states and inputs.

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



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

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

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

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

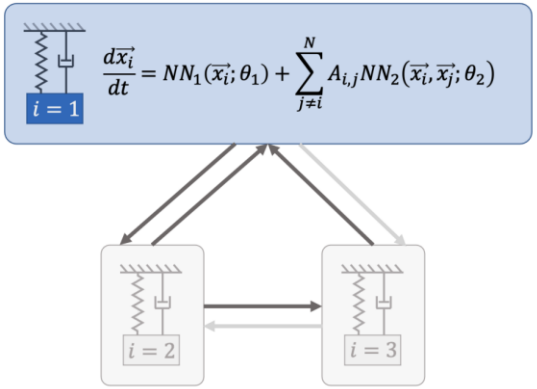
Loss function: trajectory matching, regularizations, and constraints penalties.

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

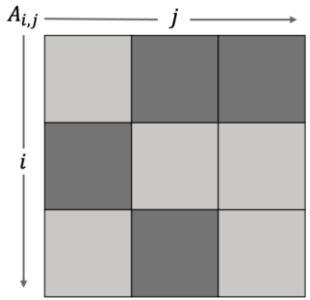
https://github.com/pnnl/neuromancer/blob/master/examples/ODEs/Part_1_NODE.ipynb
https://github.com/pnnl/neuromancer/blob/master/examples/ODEs/Part_7_DeepKoopman.ipynb

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

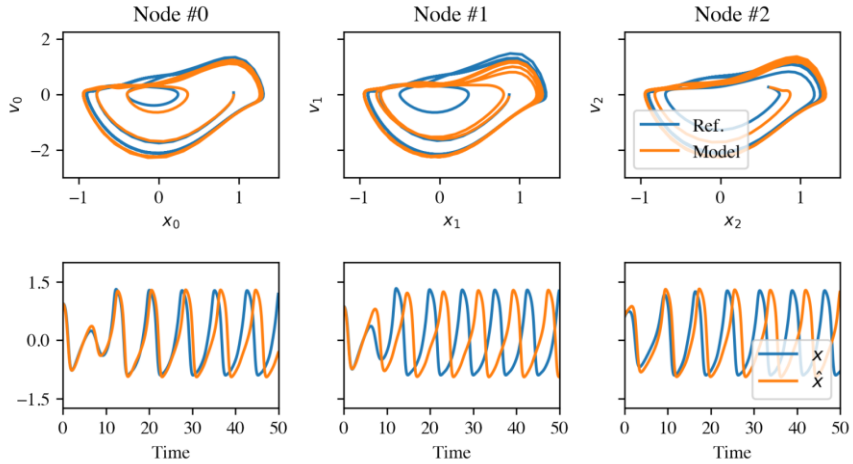
Learning Networked Dynamical Systems via Universal Differential Equations



Network of unknown oscillators



Coupling adjacency



Ground truth system: $$\frac{d^2x_i}{dt^2} = -x_i - a_1\frac{dx_i}{dt}\left(a_2x_i^4 - a_3x_i + a_4\right) + \sum_{j=1}^{N}\mathbf{A}_{i,j}\left(\frac{d^2x_i}{dt^2} - \frac{d^2x_j}{dt^2}\right),$$

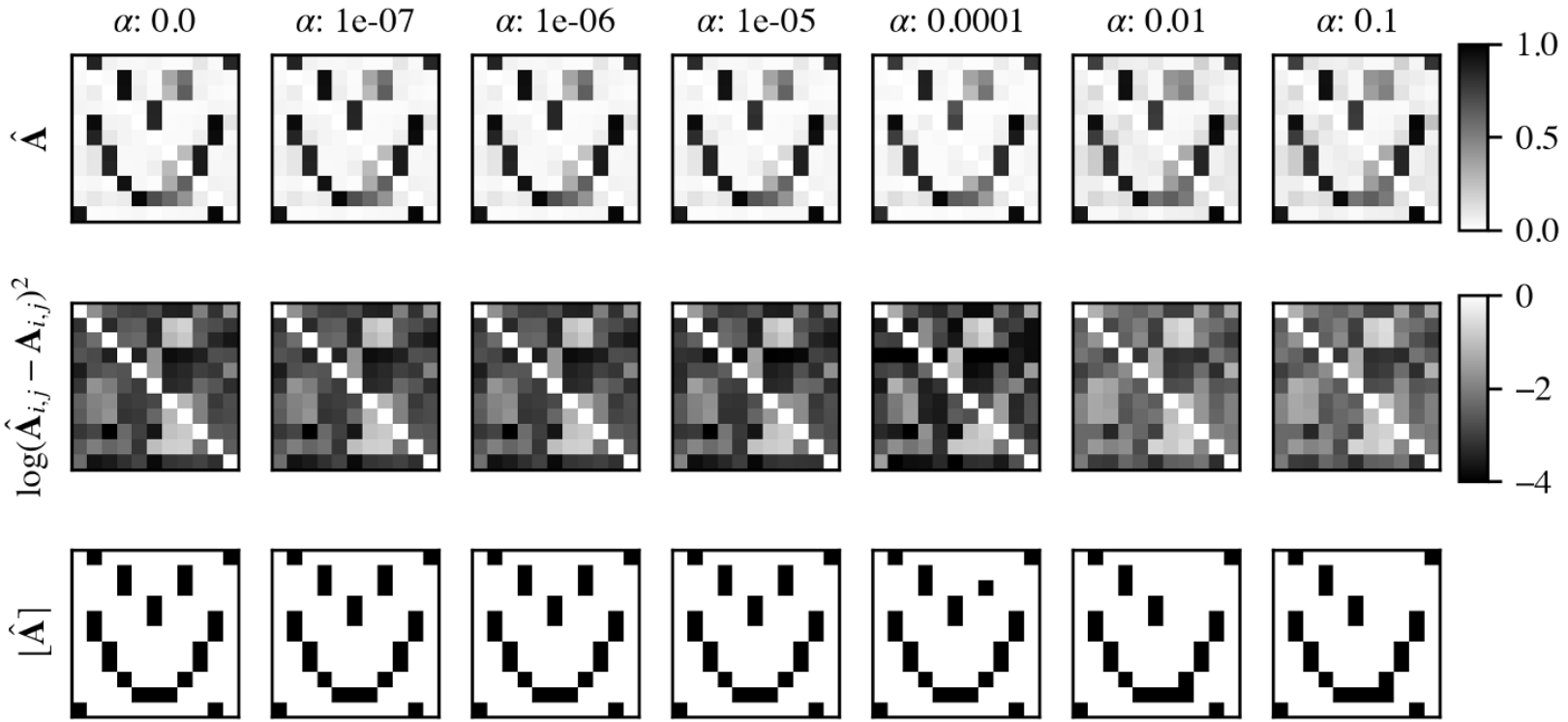
Networked NODE: $$\frac{d\mathbf{x}_i}{dt} = \mathbf{f}\left(\mathbf{x}_i;\theta_f\right) + \sum_{j\neq i}^{N}\mathbf{A}_{i,j}\mathbf{g}\left(\mathbf{x}_i,\mathbf{x}_j;\theta_g\right), \quad A_{i,j}(\theta_3) = \sigma\left(\mathrm{vec}^{-1}(\theta_3) - I/\varepsilon\right)$$

Sparsity-promoting loss: $$\mathscr{L}(\Theta,\mathbf{A}) = \frac{1}{N_b\cdot N_f\cdot N\cdot d}||\hat{\mathbf{X}}^{(k)}(t;\Theta) - \mathbf{X}^{(k)}(t)||_2^2 + \alpha||\mathbf{A}||_1,$$

Generalization of learned dynamics on never-seen network topology.
Learned dynamics qualitatively retains the attractor shape even **on out of distribution data.**

James Koch, Zhao Chen, Aaron Tuor, Jan Drgona, Draguna Vrabie, Structural Inference of Networked Dynamical Systems with Universal Differential Equations, Chaos: An interdisciplinary Journal of Nonlinear Science, doi.org/10.1063/5.0109093, 2023

Learning Networked Dynamical Systems via Universal Differential Equations



Learning adjacency matrix of the unknown network ODE system.

James Koch, Zhao Chen, Aaron Tuor, Jan Drgona, Draguna Vrabie, Structural Inference of Networked Dynamical Systems with Universal Differential Equations, Chaos: An interdisciplinary Journal of Nonlinear Science, doi.org/10.1063/5.0109093, 2023

Learning Neural Differential Algebraic Equations via Operator Splitting

DAE parameter estimation problem:

$$\min_{\Theta} \quad \int_{t_0}^{t_N} \left(||x - \hat{x}||_2^2 + ||y - \hat{y}||_2^2 \right)$$

$$\text{s.t} \quad \frac{dx}{dt} = f\left(x, y, u; \Theta\right),$$

$$0 = g\left(x, y, u; \Theta\right),$$

$$x(t_0) = x_0, \quad y(t_0) = y_0,$$

The **Picard–Lindelöf theorem** (also called the Cauchy–Lipschitz theorem) gives **sufficient conditions** under which an **initial value problem** (IVP) for an ordinary differential equation (ODE) has a **unique solution**.

https://github.com/pnnl/NeuralDAEs

Assumptions:

1. f is continuous in both t and x
2. f is Lipschitz continuous in x on some neighborhood of initial conditions

Theorem 3.1 (Picard-Lindelöf for index-1 DAEs [30]): Under the above assumptions, there exists a unique, continuously differentiable solution $(x^{(t)}, y^{(t)})$ defined on an interval $[t_0, t_0 + \Delta t)$, for some $\Delta t > 0$, such that the DAE system is satisfied and $(x^{(t_0)}, y^{(t_0)}) = (x_0, y_0)$. Furthermore, the algebraic constraint can be locally solved for y as a function of x and u, thereby reducing DAEs into a system of ODEs $\frac{\partial x}{\partial t} = f(x, h(x, u), u)$ to which the classical Picard-Lindelöf Theorem 3.2 applies.

[30] K. E. Brenan, S. L. Campbell, and L. R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. SIAM, 1996.

J. Koch, M. Shapiro, H. Sharma, D. Vrabie, J. Drgona, Learning Neural Differential Algebraic Equations via Operator Splitting, arXiv:2403.12938, 2024.

Learning Neural Differential Algebraic Equations via Operator Splitting

Our neural DAE time stepper architecture is inspired by operator splitting methods for ODEs:

$$\frac{dx}{dt} = A(x) + B(x),$$

Lie-Trotter splitting scheme for ODEs:

$$\hat{x}^{(t+\Delta t)} = \text{ODESolve}\left(A(x), x^{(t)}\right),$$
$$x^{(t+\Delta t)} = \text{ODESolve}\left(B(x), \hat{x}^{(t+\Delta t)}\right).$$

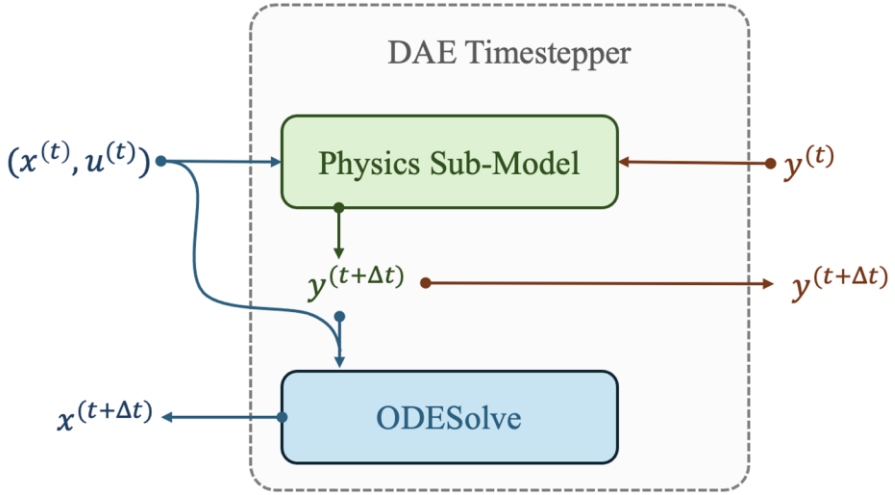
Operator splitting for DAEs:

$$y^{(t+\Delta t)} = h\left(x^{(t)}, y^{(t)}, u^{(t)}; \theta_h\right)$$
$$x^{(t+\Delta t)} = \text{ODESolve}\left(f, \{x^{(t)}, y^{(t+\Delta t)}, u^{(t)}\}; \theta_f\right)$$

https://github.com/pnnl/NeuralDAEs

Neural DAE learning problem:

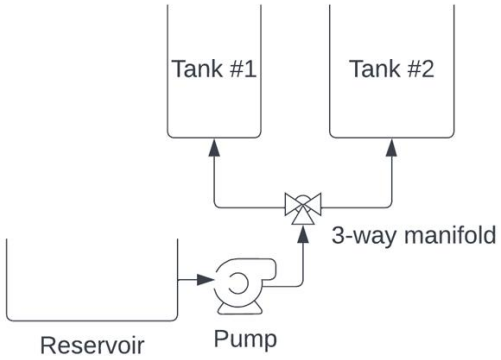
$$\min_{\theta_1,\theta_2} \lambda_1 \sum_{k=1}^{N} ||x^{(k)} - \hat{x}^{(k)}||_2^2 + \lambda_2 \sum_{k=1}^{N} ||y^{(k)} - \hat{y}^{(k)}||_2^2$$
$$\text{s.t.} \quad \{x^{(k+1)}, y^{(k+1)}\}$$
$$= \text{DAESolve}\left(f, h, \{x^{(k)}, y^{(k)}, u^{(k)}\}; \{\theta_1, \theta_2\}\right),$$

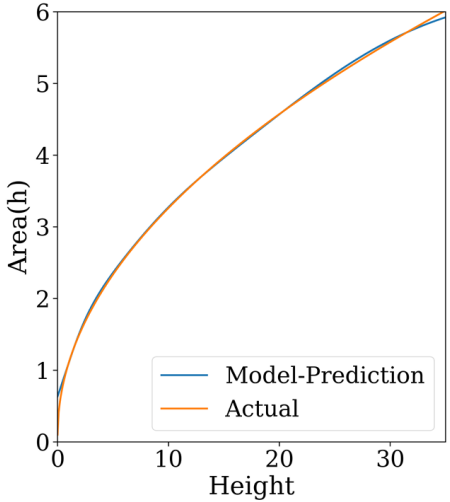


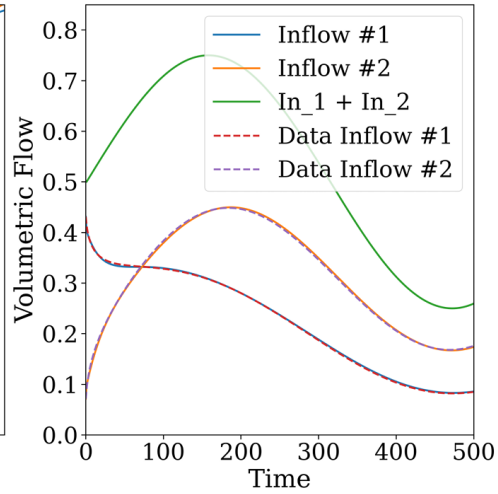
J. Koch, M. Shapiro, H. Sharma, D. Vrabie, J. Drgona, Learning Neural Differential Algebraic Equations via Operator Splitting, arXiv:2403.12938, 2024.

Learning Neural Differential Algebraic Equations via Operator Splitting

Case study: coupled tank modeling







DAE with unknown components:

$$\frac{dx_1}{dt} = \frac{y_1}{\phi_1(x_1)}$$
$$\frac{dx_2}{dt} = \frac{y_2}{\phi_2(x_2)}$$
$$0 = y_{\text{in}} - y_1 - y_2 \quad 0 = x_1 - x_2,$$

Neural DAE:

$$f : \begin{cases} \frac{y_1}{3} \\ \frac{y_2}{\text{NN}_1(x_2;\theta_1)}, \end{cases}$$

$$h : \begin{cases} u\,\text{NN}_2(x_1, x_2, y_1, y_2; \theta_2) \\ u\left(1 - \text{NN}_2(x_1, x_2, y_1, y_2; \theta_2)\right), \end{cases}$$

Inverse modeling, robustness to noise and, out of distribution generalization.

https://github.com/pnnl/NeuralDAEs

J. Koch, M. Shapiro, H. Sharma, D. Vrabie, J. Drgona, Learning Neural Differential Algebraic Equations via Operator Splitting, arXiv:2403.12938, 2024.

Learning Neural Lyapunov Functions from Time-series Data

We aim to find neural Lyapunov function candidate satisfying the following conditions:

$$V_\theta(x) : R^n \to R;\;\; V_\theta(0) = 0;\;\; V_\theta(x) > 0, \forall x \neq 0$$

Neural Lyapunov function candidate via input-convex neural network (ICNN) architecture

$$\begin{aligned}
&\mathbf{z}_1 = \sigma_0\left(\mathbf{W}_0\mathbf{x} + \mathbf{b}_0\right),\\
&\mathbf{z}_{i+1} = \sigma_i\left(\mathbf{U}_i\mathbf{z}_i + \mathbf{W}_i\mathbf{x} + \mathbf{b}_i\right), i = 1, \ldots, k-1,\\
&\mathbf{g}(\mathbf{x}) \equiv \mathbf{z}_\mathbf{k},\\
&V(\mathbf{x}) = \sigma_{k+1}(\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{0})) + \epsilon\|\mathbf{x}\|_2^2,
\end{aligned}$$

Physics-informed loss function with discrete-time Lyapunov penalty:

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

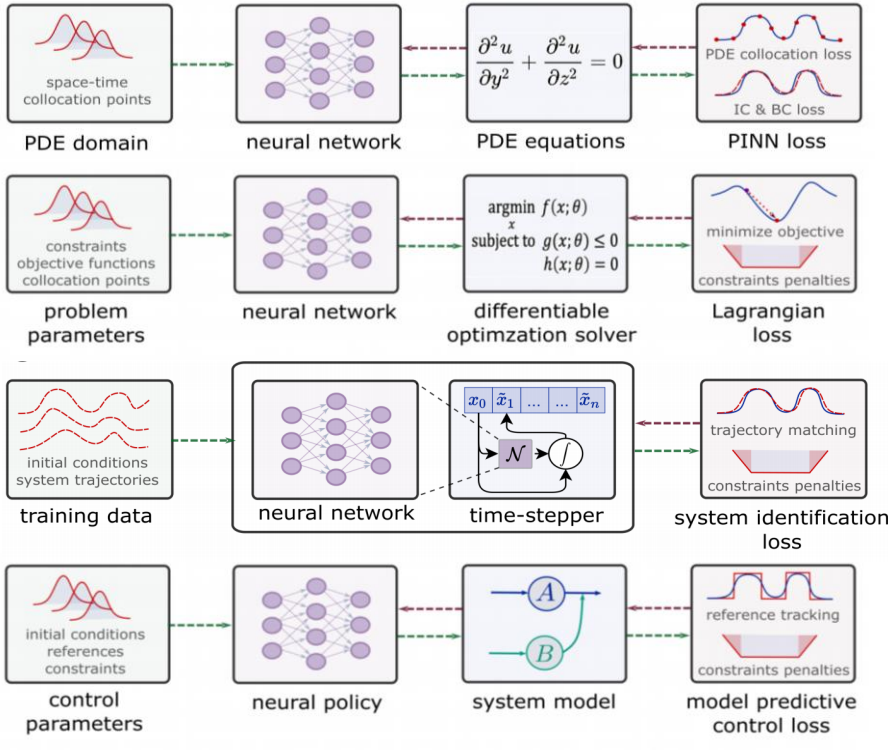
Learning problem formulation for joint learning of neural Lyapunov functions and control policies via Differentiable Predictive Control (DPC)

$$\begin{aligned}
\min_{\theta,\phi} \frac{1}{mN} \sum_{i=1}^{m} \sum_{k=0}^{N-1} &\Big(\ell(\mathbf{x}_k^i, \mathbf{u}_k^i) + Q_V p_V(\mathbf{x}_{k+1}^i, \mathbf{x}_k^i) + \\
&Q_x p_x(\mathbf{x}_k^i) + Q_u p_u(\mathbf{u}_k^i)\Big),\\
\text{s.t. } &\mathbf{x}_{k+1}^i = f(\mathbf{x}_k^i, \mathbf{u}_k^i),\;\; k \in \mathbb{N}_0^{N-1},\\
&[\mathbf{u}_0, \ldots, \mathbf{u}_{N-1}]^i = \boldsymbol{\pi}_\theta(\mathbf{x}_k^i),\\
&\mathbf{x}_0^i \in \mathbb{X} \subseteq \mathbb{R}^{n_x} \sim \mathcal{D}.
\end{aligned}$$

https://github.com/pnnl/neuromancer/blob/master/examples/control/Part_5_neural_Lyapunov.ipynb
https://github.com/pnnl/neuromancer/tree/Lyapunov_DPC

Gaurav Manek, J. Zico Kolter, Learning Stable Deep Dynamics Models, NeurIPS 2019
S. Mukherjee, J. Drgoňa, A. Tuor, M. Halappanavar, D. Vrabie, Neural Lyapunov Differentiable Predictive Control, CDC, 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

Summary

- **Challenges**
  - Uncertainty Quantification (UQ)
  - Conservatism vs. performance tradeoff
  - Safe exploration in learning-based control
  - Scalability of guarantees and verification methods
- **Opportunities**
  - Integrated UQ and safety filters
  - Differentiable control theory-inspired learning architectures
  - Sample efficient safe learning-based control
  - Benchmarking and software tools
- **Follow up**
  - TuC04 Tutorial Session, Governor's Sq. 15, 15:30-17:00,

Energy at Hopkins

drgona.github.io jdrgona1@jh.edu github.com/pnnl/neuromancer

NeuroMANCER Team

Aaron Tuor

James Koch

Madelyn Shapiro

Rahul Birmiwal

Bruno Jacob

Draguna Vrabie

U.S. DEPARTMENT OF ENERGY

Pacific Northwest NATIONAL LABORATORY