MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

MECHENG 590 Presentation

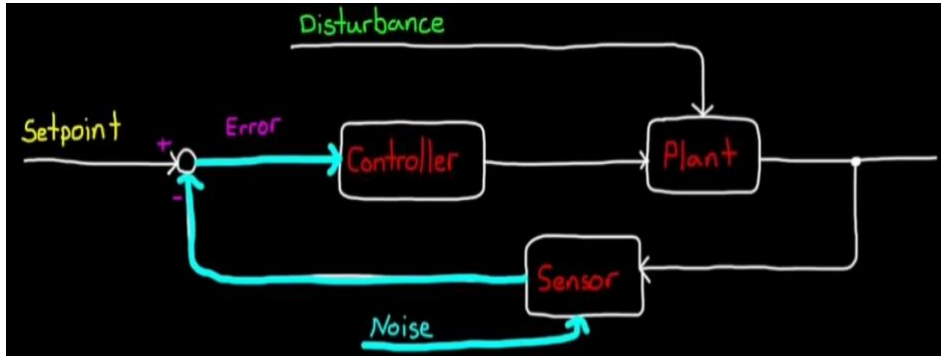
Co-state Neural Network for Realtime Nonlinear Optimal Control with Input Constraints

Presenter: Lihan Lian, Supervisor: Uduak Inyang-Udoh

Nov 21st, 2024

Preliminary

- Optimal Feedback Control Approaches:
  - LQR can be applied by solving ARE, but it does not respect to control input constraints.
  - MPC suffers from high computational burden for high dimensional and nonlinear systems or constraints.
- Pontrygain's Maximum Principle (PMP)
  - PMP provides necessary conditions by solving two-point boundary value problems (TPBVPs).
  - LQR can be viewed as a special case of PMP.
  - MPC solves finite horizon optimization problem to approximate the solution of the original optimal control problem (OCP) from PMP.



Source: Control Systems in Practice (YouTube) (MATLAB)

Consider the following final horizon and free final state LQR problem desribed as follows. We will show that **Algebraic Riccati Equation (ARE)** can be derived using the previously mentioned procedures as $T \to \infty$.

$$J = \frac{1}{2}\int_0^T (x(t)^T Q x(t) + u(t)^T R u(t))\, dt + \frac{1}{2} x(t)^T H x(t)$$

$$s.t. \quad \dot{x} = Ax(t) + Bu(t)$$

where $Q > 0$, $R \geq 0$ and H is a constant positive definite matrix.

- Formulate Control Hamiltonian:

$$\mathcal{H}(x, u, \lambda, t) = \frac{1}{2}(x^T Q x + u^T R u) + \lambda^T (Ax + Bu)$$

- Construsct state and co-state equations:

$$\dot{x}^* = \nabla_\lambda \mathcal{H}(x, u, \lambda, t) = Ax + Bu$$

$$\dot{\lambda}^* = -\nabla_x \mathcal{H}(x, u, \lambda, t) = -Qx - A^T \lambda$$

- Obtaining Optimal u:

Source: From Control Hamiltonian to ARE and PMP (Blog Post)

Introduction

- Motivation:
  - MPC does not utilize the fact that patterns from previous solutions may be used to alleviate the computational burden of subsequent solutions.
  - Indirect method typically cannot be used in real-time.
- Related Work
  - Only for specific initial conditions.
  - No disturbance in the validation and control policy is applied in a feedforward way.
  - Difficult for NN training to converge due to the design of satisfying input constraints.

2. PROBLEM STATEMENT

We consider continuous time optimal control problems with fixed initial condition and fixed final time. The cost functional is in the Bolza form, including both a running (ℓ) and a terminal cost (ϕ) in the functional objective (J) (Bryson and Ho, 1975):

$$\begin{aligned} \min_{\theta} \quad & J = \int_{t_0}^{t_f} \ell\left(x(t), \pi_\theta(x)\right) dt + \phi(x(t_f)), \\ \text{s.t.} \quad & \dot{x}(t) = f(x(t), \pi_\theta(x), t), \\ & x(t_0) = x_0, \\ & g(x(t), \pi_\theta(x)) \leq 0, \end{aligned} \tag{1}$$

where the time window is fixed $t \in [t_0, t_f]$, $x(t) \in \mathbb{R}^{n_x}$ is the state, $\pi_\theta(x) : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ is the state feedback controller and $\theta \in \mathbb{R}^{n_\theta}$ are its parameters.

3.3 Control constraints

The controller outputs are box-constrained to U directly from the nonlinearity of the last activation function of the policy. We use a custom scaled sigmoid function $\sigma(\cdot) : \mathbb{R} \to [0, 1]$ for each control component u_i. Then, the last layer of the policy has the functional form

$$u_i = u_i^{(lb)} + (u_i^{(ub)} - u_i^{(lb)})\sigma_i(\cdot),$$

where $u_i^{(lb)}$ and $u_i^{(ub)}$ are the lower and upper bounds of the control component, respectively, and $i \in [1, \ldots, n_u]$. By construction, any parameter set in the policy yields a feasible control sequence satisfying the control constraints.

Source: Neural ODEs as Feedback Policies for Nonlinear Optimal Control.

Introduction

- Contribution
  - We present a novel perspective of using neural networks that learns the mapping from an initial state to its corresponding optimal co-state trajectory.
  - Instead of handling control input constraints directly during the process of training neural network, we propose a framework for handling input constraints by extracting a NN-predicted co-state trajectory and then simply solving a QP.
  - Realtime feedback validation.

III. Problem Statement

For a control affine system, consider the infinite final time and fixed final state optimal control problem with quadratic running cost in continuous time as follows:

$$\min \quad J = \frac{1}{2}\int_{t_0}^{t_f=\infty} \left(x^\top(t)Qx(t) + u^\top(t)Ru(t)\right) dt, \tag{8a}$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t)) + g(x(t))u(t), \tag{8b}$$

$$x(t_0) \in X \tag{8c}$$

$$x(t_f = \infty) = x_f, \tag{8d}$$

$$u(t) \in \mathcal{U}, \tag{8e}$$

Since we are **NOT** concerned with the control input constraints here, the optimal control input u can be directly obtained by setting the following equation to zero.

$$\nabla_u \mathcal{H} = Ru + B^T\lambda = 0, \quad u^* = -R^{-1}B^T\lambda$$

Substitute u^* into state equation and put them all together into matrix form:

$$\begin{bmatrix} \dot{\mathbf{x}}^*(t) \\ \dot{\lambda}^*(t) \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{bmatrix} \begin{bmatrix} \mathbf{x}^*(t) \\ \lambda^*(t) \end{bmatrix}$$

Now suppose the costate λ is linearly dependent on state x (See Appendix for more detail about the validation of this assumption), and its time derivative can be expressed as follows:

$$\lambda^*(t) = \mathbf{P}(t)\mathbf{x}^*(t)$$

$$\dot{\lambda}^*(t) = \dot{\mathbf{P}}(t)\mathbf{x}^*(t) + \mathbf{P}(t)\dot{\mathbf{x}}^*(t)$$

Source: From Control Hamiltonian to ARE and PMP (Blog Post)

Background

- Model Predictive Control
  - Transcribes OCP into an optimization problem with finite number of decision variables, falls into the paradigm of *discretize then optimize*.
  - Solves the similar optimization problem repetitively in a receding horizon fashion.
  - Normally no guarantee for timely convergence, especially for nonlinear and high-dimensional systems.
  - In certain cases, explicit-mpc can be used to alleviate the computational burden.

A. Model Predictive Control (MPC)

In practice, when realtime optimal control is needed, (1) is solved using MPC, or receding horizon control. MPC uses the system model to predict the future states of the system and solve the online optimization problem in a moving horizon fashion [5]. First, the continuous system dynamics in (1b) is discretized based on suitable sampling interval to yield a discrete-time system:

$$x_{t+1} = f(x_t, u_t), \tag{2}$$

$$J_t^*(x_t) = \min_{u_{t:t+P-1|t}} \ell_f(x_{t+P|t}) + \sum_{k=0}^{P-1} \ell(x_{t+k|t}, u_{t+k|t}) \tag{3a}$$

$$\text{s.t.} \quad x_{t+k+1|t} = f(x_{t+k|t}, u_{t+k|t}), \quad k = 0, \ldots, P-1 \tag{3b}$$

$$u_{t+k|t} \in \mathcal{U}, \quad k = 0, \ldots, P-1 \tag{3c}$$

$$x_{t|t} = x_t, \tag{3d}$$

$$x_{t+P|t} \in \mathcal{X}_f. \tag{3e}$$

Background

- Pontryagin's Minimum (Maximum) Principle
  - Can handle input constraints with the use of co-state and control Hamiltonian.
  - PMP transcribes the OCP to a point-wise optimization to avoid minimizing over a function space.
  - Belongs to the paradigm of *optimize then discretize*.

B. Pontryagin's Minimum Principle

Pontryagin's Minimum (Maximum) Principle (PMP) is a fundamental result in optimal control theory, providing necessary conditions for optimality in systems governed by differential equations. For the problem in (1), PMP introduces the control Hamiltonian H, defined as:

$$H(x(t), u(t), \lambda(t), t) = L(x(t), u(t), t) + \lambda^{\top}(t) f(x(t), u(t), t), \quad (4)$$

where $\lambda(t) \in \mathbb{R}^n$ is the co-state (or adjoint) vector. Constraints on state variables and co-state variables are then derived by taking the partial derivatives of H as follows:

$$\dot{x}(t) = \frac{\partial H}{\partial \lambda}, \quad (5)$$

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x}. \quad (6)$$

$$u^*(t) = \arg \min_{u(t) \in \mathcal{U}} H(x^*(t), u(t), \lambda^*(t), t). \quad (7)$$

Problem Statement

- OCP formulation
  - Multiple initial conditions, uses numerical solver to compute the optimal solutions off-line, for each initial conditions.
  - Uses quadratic running cost function and formulates the control Hamiltonian.
  - The optimal solution obtained from numerical solver is in the unconstrained scenario.
  - Control input constraint is then imposed by solving an easier QP.

$$\min \quad J = \frac{1}{2}\int_{t_0}^{t_f=\infty} \left(x^\top(t)Qx(t) + u^\top(t)Ru(t)\right) dt, \tag{8a}$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t)) + g(x(t))u(t), \tag{8b}$$

$$x(t_0) \in X \tag{8c}$$

$$x(t_f = \infty) = x_f, \tag{8d}$$

$$u(t) \in \mathcal{U}, \tag{8e}$$

$$H(x(t), u(t), \lambda(t), t) = \frac{1}{2}x^\top(t)Qx(t) + \frac{1}{2}u^\top(t)Ru(t) + \lambda^\top(t)\left(f(x(t)) + g(x(t))u(t)\right), \tag{9}$$

$$\dot{x}(t) = \frac{\partial H}{\partial \lambda} = f(x(t)) + g(x(t))u(t), \tag{10}$$

$$\dot{\lambda}(t) = -\frac{\partial H}{\partial x} = -Qx(t) - \left(\frac{\partial f(x(t))}{\partial x}\right)^\top \lambda(t) - \left(\frac{\partial g(x(t))}{\partial x}u(t)\right)^\top \lambda(t). \tag{11}$$

Problem Statement

- OCP formulation
  - Multiple initial conditions, uses numerical solver to compute the optimal solutions off-line, for each initial conditions.
  - Uses quadratic running cost function and formulates the control Hamiltonian.
  - The optimal solution obtained from numerical solver is in the unconstrained scenario.
  - Control input constraint is then imposed by solving an easier QP.

there are no constraints on the control input, the optimal control law can be obtained by setting:

$$\frac{\partial H}{\partial u^*} = 0, \tag{12}$$

which yields

$$u^*(t) = -R^{-1} g^\top(x(t)) \lambda^*(t). \tag{13}$$

When there are constraints, the optimal control input can be obtained from:

$$u^*(t) = \arg \min_{u(t) \in \mathcal{U}} \left(\frac{1}{2} u^\top(t) R u(t) + \lambda^{*\top}(t) g(x(t)) u(t) \right). \tag{14}$$

$$\mathcal{L}^* = \frac{1}{2}(x^{*T} Q x^* + u^{*T} R u^*) + \lambda^T (A x^* + B u^* - \dot{x}^*)$$

$$\Phi^*|_{t_f} = \frac{1}{2}(x^{*T}(t_f) H x^*(t_f))$$

$$-\lambda^*(t_f) + H x^*(t_f) = 0, \quad \lambda^*(t_f) = H x^*(t_f)$$

Thus, the other n boundary conditions come from the costate state equation at the final time and we need to solve the ARE **backward** in time to get costate trajectory $\lambda^*(t)$. Then the optimal control input trajectory $u^*(t)$ can obtained by the preivously derived optmality condition ($u^*(t) = -R^{-1} B^T \lambda^*(t)$).

Source: From Control Hamiltonian to ARE and PMP (Blog Post)

Methodology

- Neural Network Architecture
  - Fully connected linear layers.
  - Input is state (x), output is a vector with the size of predefined prediction horizon.
  - Essentially learn a mapping from state to its corresponding optimal co-state trajectory.

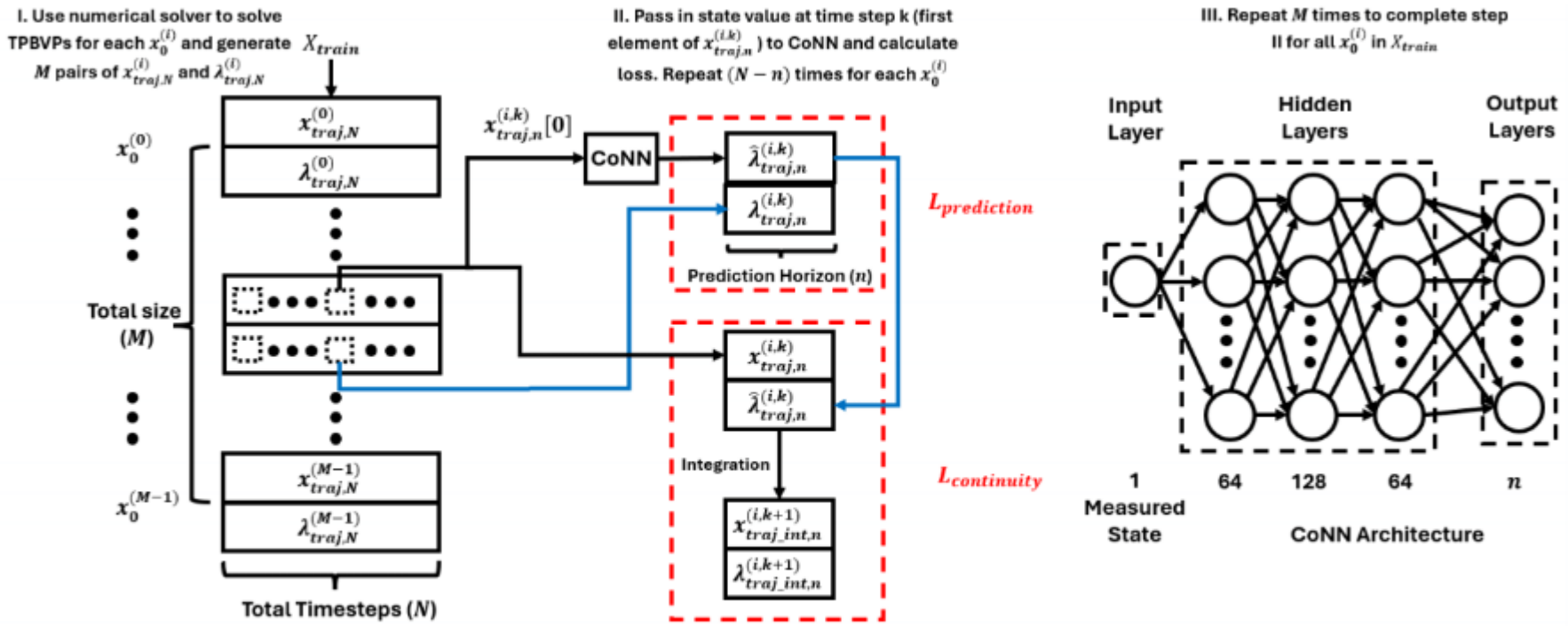


Fig. 1: CoNN architecture and training procedures illustrated for $x_0^{(i)} \in \mathbb{R}, \forall i$, same as the one-dimensional system shown in the Sec. V. Note that the index $[j]$ denote the j^{th} entry of the trajectory starting from 0.

Methodology

- Training Procedures
  - Prediction loss

$$\mathcal{L}_{\text{prediction}}^{(i,k)} = \frac{1}{n}\sum_{j=0}^{n-1}\left(\hat{\boldsymbol{\lambda}}_{\text{traj},n}^{(i,k)}[j] - \boldsymbol{\lambda}_{\text{traj},n}^{(i,k)}[j]\right)^2. \quad (16)$$

  - Continuity loss

$$\mathcal{L}_{\text{continuity}}^{(i,k)} = \frac{1}{n-1}\left(\sum_{j=0}^{n-2}\left(\boldsymbol{x}_{\text{traj},N}^{(i,k)}[j+1] - \boldsymbol{x}_{\text{traj\_int},n}^{(i,k+1)}[j]\right)^2 + \sum_{j=0}^{n-2}\left(\hat{\boldsymbol{\lambda}}_{\text{traj},n}^{(i,k)}[j+1] - \boldsymbol{\lambda}_{\text{traj\_int},n}^{(i,k+1)}[j]\right)^2\right). \quad (17)$$

Algorithm 1 CoNN Training Procedures

procedure TRAINING SET GENERATION
  Define set of initial conditions X_{train}, prediction horizon n, total time steps $N \geq n$
  For $x_0^{(i)} \in X_{\text{train}}$ **do**
    Solve the TPBVP results from Eq. (8) and PMP by numerical solvers
    Store optimal state solution trajectory $\boldsymbol{x}_{\text{traj, N}}^{(i)}$
    Store optimal co-state solution trajectory $\boldsymbol{\lambda}_{\text{traj, N}}^{(i)}$
  End for
procedure CONN TRAINING
  Define total training epochs N_{epoch}, learning rate α and initialize CoNN parameters θ
  For e in $range(N_{\text{epoch}})$ **do**
    For $x_0^{(i)} \in X_{\text{train}}$ **do**
      For k in $range(N-n)$ **do**
        Get $\boldsymbol{\lambda}_{\text{traj, n}}^{(i,k)}$ and $\boldsymbol{x}_{\text{traj, n}}^{(i,k)}$
        $\hat{\boldsymbol{\lambda}}_{\text{traj},n}^{(i,k)} = \text{CoNN}_\theta(\boldsymbol{x}_{\text{traj, n}}^{(i,k)}[0])$
        Calculate $\mathcal{L}_{\text{prediction}}^{(i,k)}$ and $\mathcal{L}_{\text{continuity}}^{(i,k)}$ based on Eqn. (16) and (17)
        Update CoNN parameters θ
      End for
    End for
  End for

Methodology

- Control Input Constraints Handling
  - Only extract the first element from the predicted optimal co-state trajectory.
  - Solve a much easier QP to obtain optimal control input that satisfy both constraints and optimality.
- CoNN-based Controller Validation
  - In a real-time feedback control loop manner.
  - Including disturbance, unseen initial conditions (not in the training data set).

$$u_k^* = \arg\min \left(\frac{1}{2} u_k^\top R u_k + (\hat{\lambda}_{\text{traj},n}^k[0])^\top g(x(t)) u_k \right), \tag{18a}$$

$$\text{s.t.} \quad \hat{\lambda}_{\text{traj},n}^k = \text{CoNN}_\theta(x_k), \tag{18b}$$

$$u_k \in \mathcal{U}. \tag{18c}$$

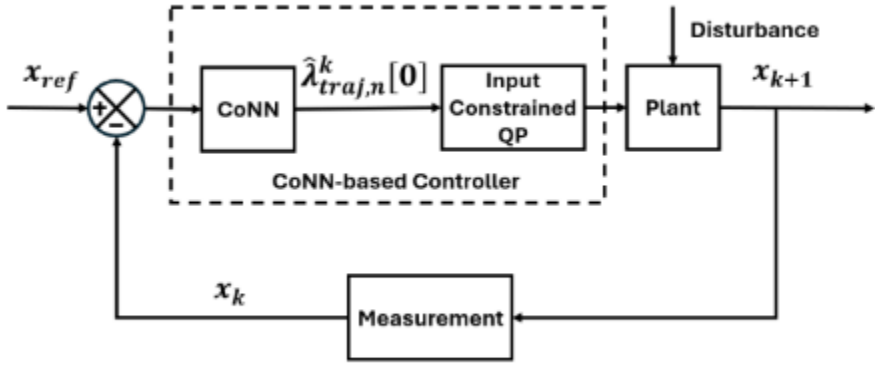


Fig. 2: CoNN-based controller block diagram for validation.

Example

- One-Dimensional OCP
  - Nonlinear, box constraints on control input.
  - Multiple initial conditions, ranging from [-5, 5] with 0.1 resolution.
  - Code in Python. Use *scipy.solve_bvp* function as the numerical solver for TPBVPs.

Based on PMP, the control Hamiltonian H can be expressed as follows:

$$H = \frac{1}{2}x^2 + \frac{1}{2}u^2 + \lambda\left(-x^2 + x + u\right). \tag{20}$$

We can then derive the differential equation that optimal state and costate should follow:

$$\dot{x}^* = \frac{\partial H}{\partial \lambda} = -x^{*2} + x^* + u^*, \tag{21}$$

$$\dot{\lambda}^* = -\frac{\partial H}{\partial x} = -\left(x^* - 2\lambda^* x^* + \lambda^*\right). \tag{22}$$

V. Example

Consider the following one-dimensional nonlinear optimal control problem in continuous time that has quadratic running cost:

$$\min_u \quad J = \frac{1}{2}\int_0^\infty \left(x^2 + u^2\right) dt, \tag{19a}$$

$$\text{s.t.} \quad \dot{x} = -x^2 + x + u, \tag{19b}$$

$$u_{\min} \le u \le u_{\max}, \tag{19c}$$

$$x_0 \in \mathbb{R}, \tag{19d}$$

$$x_\infty = 0, \tag{19e}$$

where X_{train}, is chosen to be $[-5.0, 5.0]$ for the range of initial conditions that produce TPBVPs. A total of 101 data points were sampled uniformly within the range of X_{train}

$$\frac{\partial H}{\partial u^*} = u^* + \lambda^* = 0, \tag{23}$$

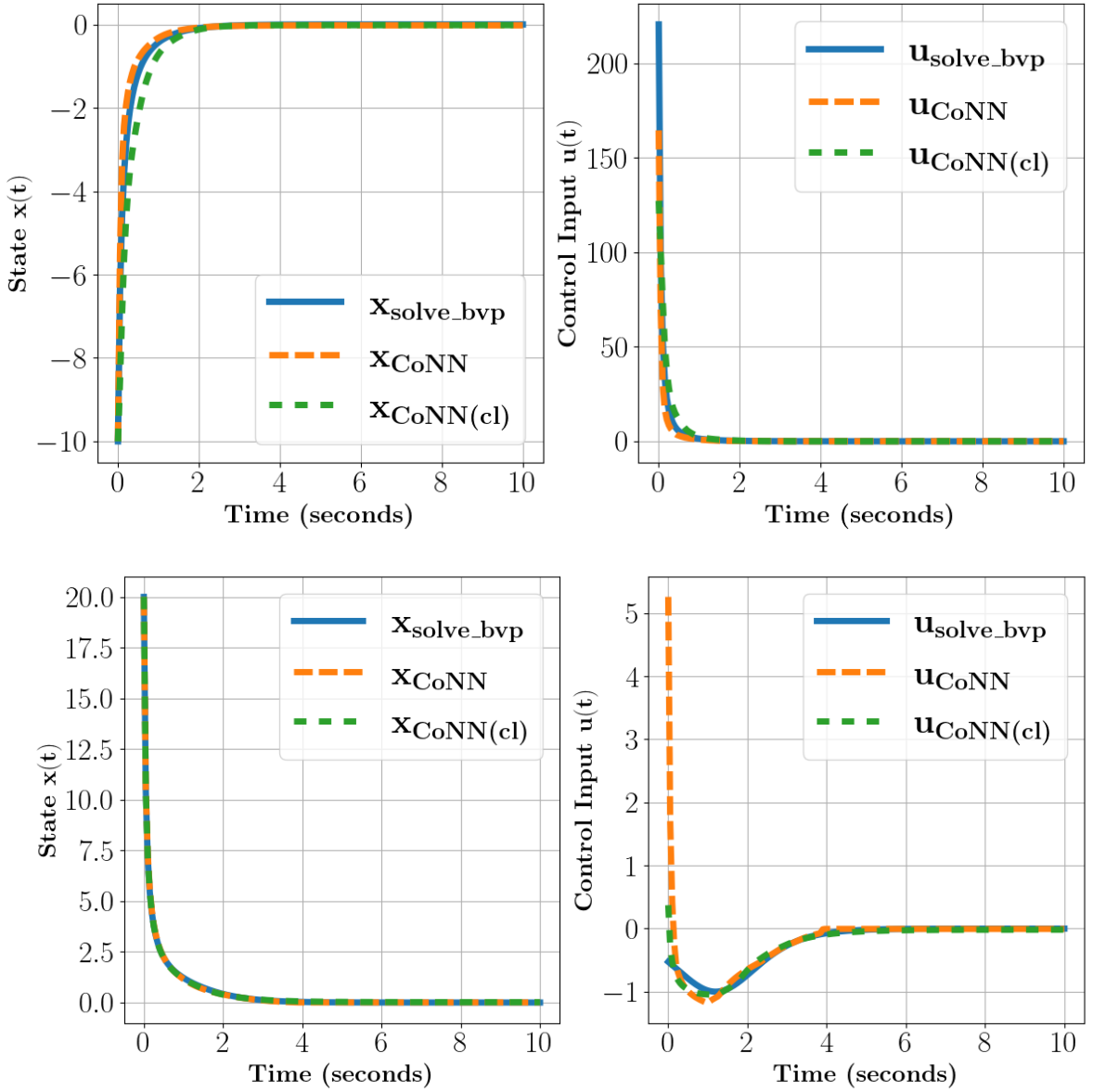
$$u^* = -\lambda^*. \tag{24}$$

When there are control input constraints, the optimal control input should minimize the control Hamiltonian H. For this problem formulation, H is a quadratic function of u, thus, the optimal control policy can be obtained by simply saturate the unconstrained optimal control input:

$$u^* = \text{Sat}_{u_{\min}}^{u_{\max}}(-\lambda^*). \tag{25}$$

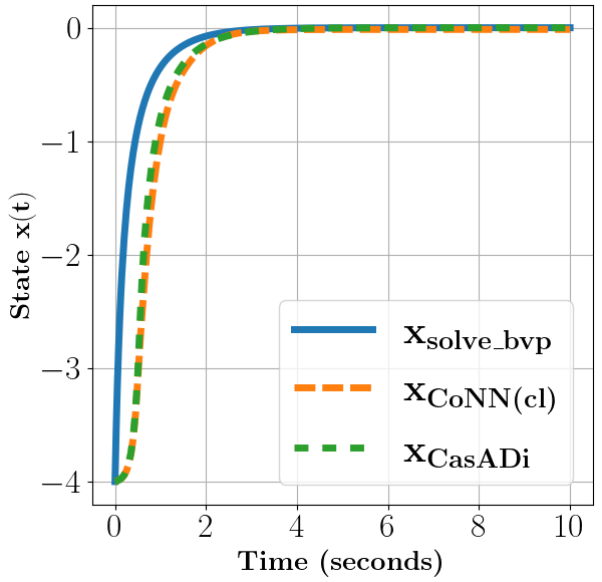
Example

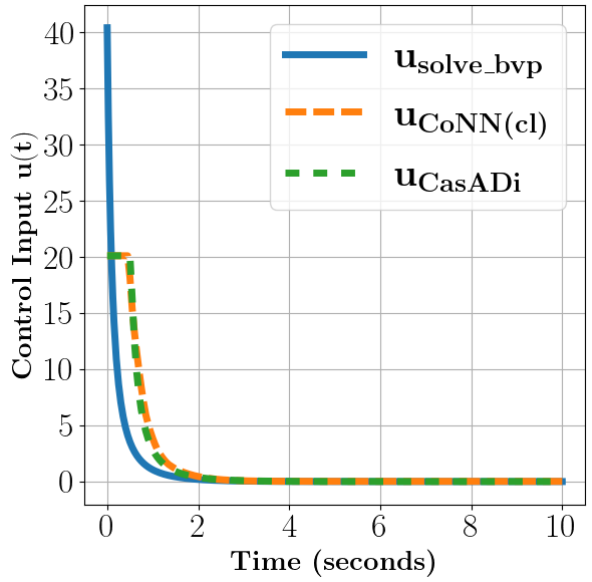
- Unconstrained Control Input with Unseen x_0
  - x_0 = -10 and x_0 = 20.
  - Both model trained with (CoNN(cl)) and without (CoNN) continuity loss make system converge to zero.
  - Model trained with continuity loss is more conservative.

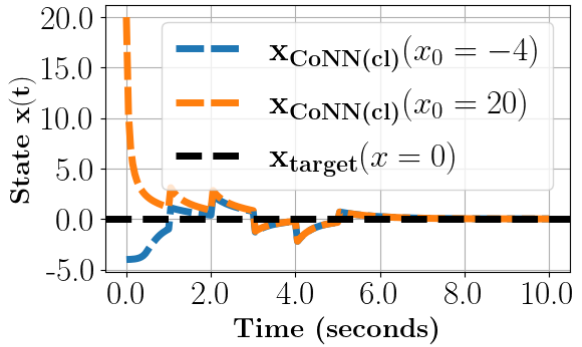


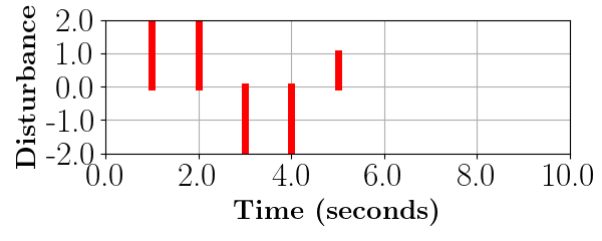
Example

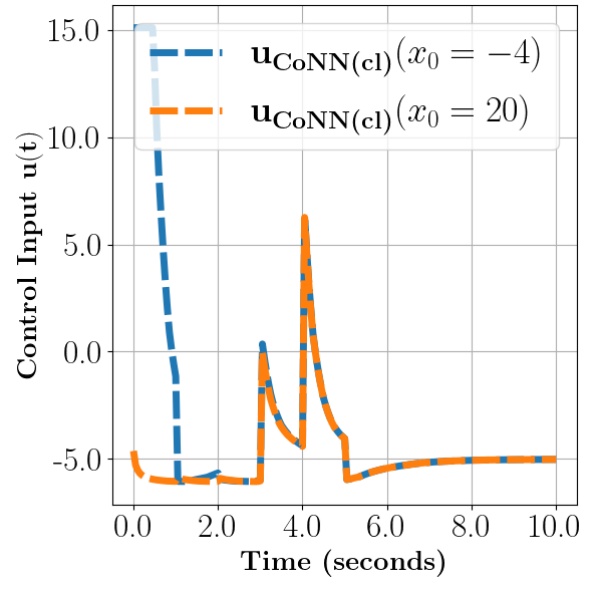
- Constrained Control Input
  - $x_0 = -4$
  - $-20.1 \leq u \leq 20.1$
  - Used direct collocation with trapezoidal approximal as the trajectory optimization algorithm. Used CasADi as optimization solver for direct method solution.
- Validation with Disturbance
  - $x_0 = -4$ and $x_0 = 20$
  - Disturbance are applied at five timesteps .











Conclusion

- Future Work
  - Improve the neural network architecture to be better at handling high-dimensional systems.
  - Integrate state constraints into the design of the loss function.
  - Provide a comprehensive algorithmic benchmark for MPC and reinforcement learning algorithms, addressing aspects such as sampling efficiency, real-time computational complexity, and performance metrics (e.g., objective cost function).

3.2 THE CART-POLE SYSTEM

The other model system that we will investigate here is the cart-pole system, in which the task is to balance a simple pendulum around its unstable equilibrium, using only horizontal forces on the cart. Balancing the cart-pole system is used in many introductory courses in control, including 6.003 at MIT, because it can be accomplished with simple linear control (e.g. pole placement) techniques. In this chapter we will consider the full swing-up and balance control problem, which requires a full nonlinear control treatment.

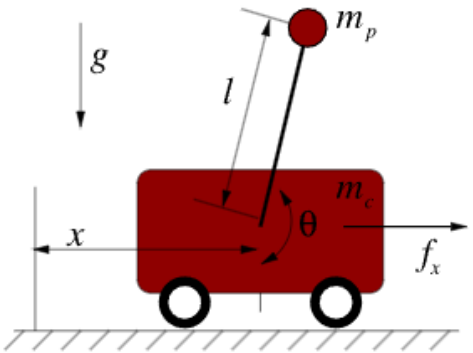


Figure 3.2 - The Cart-Pole system. Click here to see a real robot.

The figure shows our parameterization of the system. x is the horizontal position of the cart, θ is the counter-clockwise angle of the pendulum (zero is hanging straight down). We will use $\mathbf{q} = [x, \theta]^T$, and $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$. The task is to stabilize the unstable fixed point at $\mathbf{x} = [0, \pi, 0, 0]^T$.

Source: Underactuated Robotics (Prof. Russ Tedrake)

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

Thank you!

Any questions ?

AI SYS Lab