Final Project Report

Optimal Control with Nonlinear System Identification

Alvin Sun

Fall 2020
ECE 551

Abstract

Optimal control, especially for nonlinear systems, has become one of the most powerful modern control techniques that produces intelligent behaviors. Most of the optimal control methods require a-priori knowledge of the system, namely, its governing dynamical equations. However, in reality, many of the physical systems has unknown or even time-varying dynamics. As a result, identifying those dynamics becomes a crucial part in controlling them. This project aims at controlling nonlinear plants with unknown dynamics, bringing together one of the best performing nonlinear system identification methods, SINDy, with the powerful model predictive controller.

1 Problem Statement

The problem of controlling arbitrary systems with unknown dynamics can be divided into two parts

1. Estimation of the dynamics
2. Controlling the plant

1.1 Estimation

Most of the real-world physical systems can be expressed in the following first order vector differential equation

$$\dot{x} = f(x) \tag{1}$$

where x is the internal states of the system. Despite its simple form, the vector function f can be arbitrarily complex and nonlinear. To generalize a little more to fit the settings of control, the dynamical equations becomes

$$\dot{x} = f(x, u) \tag{2}$$

where u is a vector for the controlled actuation. Consider the case where the states are directly observable from the plants, namely, we can obtain both $\dot{x}$ and x from sensor readings, the problem of identifying the system becomes estimating this nonlinear vector function f. For the case where the states and derivatives are partially observable, we can build seperate observers and filters dedicated to estimating the missing variables. For the scope of this project, we assume that everything is directly obtainable from the plants.

1.2 Control

The problem for controlling a dynamical system is fairly simple. The overall objective is just regulating the states of the system to some desirable location with the help of actuation. Formally, for all control functions $u(\cdot)$ and initial conditions x_0, there exists a solution to Equation 2 that can be interpreted as the resulting state trajectory $x(\cdot)$. We will then have some reference state trajectory $x_{\text{ref}}(\cdot)$ that we would like the system to achieve. Therefore, the problem of control is then to find some control function, $u(\cdot)$, such that the resulting state trajectory is close or even converges to the reference trajectory. However, due to the limitation of digital computing, we would need to dicretize Equation 2 into the following form.

$$x_{k+1} = F(x_k, u_k) \tag{3}$$

Then, instead of finding a function defined in an infinite number of time steps, we only need to find a finite number of control sequence, $\{u_k\}_{k\in\mathcal{K}}$, which achieves state tracking on a finite number of state sequences, $\{x_k\}_{k\in\mathcal{K}}$.

2 Background

Nonlinear system identification has become a rapidly growing field in the recent years due to high quality sensors and high performance digital computing devices becoming more accessible. There are several popular approaches, some of which are starting to be applied in a variety of applications. This project has explored two of the popular methods, Koopman Analysis and Sparse Identification of Nonlinear Dynamics.

2.1 Koopman Theory

Koopman Theory is originally proposed in the early 20th century. It stated that any nonlinear system can be transformed into an infinite dimensional space where the dynamics become linear. Formally,

$$x_{k+1} = F(x_k) \equiv g(x_{k+1}) = \mathcal{K}g(x_k) \tag{4}$$

where g is an infinite dimensional nonlinear function and $\mathcal{K}$ is an infinite dimensional linear operator. Koopman Theory has been brought back to people's attention in recent years. Dynamic Mode Decomposition [1] and its nonlinear varient eDMD [2] are two of the recent system identification approaches that draw a strong connection to the Koopman Theory. Due to the rapid growth in the development of neural networks, Deep Koopman [3] proposed to use an autoencoder network to learn a finite approximation of the Koopman Operator space of any arbitrary nonlinear system. However, those work are built around systems without external actuation. There are, however, some efforts [4] for generalizing Koopman Theory to systems with control inputs with

$$g(x_{k+1}, u_{k+1}) = \mathcal{K}g(x_k, u_k) \tag{5}$$

which also drew a strong connection to DMDc [5], which incorporates control inputs with the DMD method. The

major attraction on Koopman related approaches centers around the ability to linearize a nonlinear system in a global sense. A large portion of those work focuses on the estimation aspects of identifying an unknown nonlinear plant, but the connection between classical linear control theories and Koopman analysis has yet to be explored.

2.2 Sparse Identification

Another type of methodology is built around explicitly recovering the structure of the dynamical equations. A recent framework, SINDy [6] (Sparse Identification of Nonlinear Dynamics), is developed to estimate for both the structure and coefficients of the nonlinearity of a dynamical system. They put the Equation 1 into the form

$$\dot{x} = \theta(x)A \tag{6}$$

where θ is a nonlinear vector function that transforms the states with some predefined nonlinearity such as sinusoids and polynomials, and A is a sparse coefficient matrix corresponding to each of those nonlinear terms. The popularity of SINDy based approaches grew rapidly due to its simplicity. SINDYc [7] further generalizes SINDy to incorporate external control inputs, modeling Equation 2 in similar fashion.

$$\dot{x} = \theta(x, u)A \tag{7}$$

Autoencoder-like neural network is also applied in [8] to learn a latent space coordinate capable of compressing high dimensional measurement data. Because of the simplicity, the SINDy based methods can also be potentially applied in an online adaptive setting.

2.3 Model Predictive Control

Linear systems are very well studied and have closed-form solutions and properties for many control scenarios. For example, Linear Quadratic Regulator, which is a form of optimal controller on linear systems, has closed form solutions for both finite horizon and infinite horizon cases. Those solutions are known as the Riccati Differential Equation and the Algebraic Riccati Equation. For nonlinear systems, however, they are much harder to control in general. There are many advanced theories established around stablizing those systems, but for this project, we will focus on applying model predictive controller directly to obtain actuated nonlinear behaviors. Model Predictive Control (MPC) is considered the de facto control technique that is widely applied to a variety of industries nowadays. It can be combined with system identification techniques to control systems with unknown dynamics. MPC formulates control objectives as a constrained optimization problem. As a result, the optimal solution inheritly explores the nonlinear state space which results in intelligent control behaviors.

3 Methodologies

There are three main components to this project:

1. Synthetic data generation from simulated dynamical systems
2. Nonlinear system identification
3. Nonlinear optimal control

3.1 Simulation

I chose three systems to verify the effectiveness of my design in combining system identification with optimal control. Each of them has a nonlinear governing equation

1. Simple Rod Pendulum.

$$I\ddot{\theta} = -mgl\sin\theta - b\dot{\theta} + \tau \tag{8}$$

2. Lorenz Attractor

$$\begin{cases} \dot{x} = \sigma(y - x) + u \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z \end{cases} \tag{9}$$

3. Planar Rotor

$$\begin{cases} m\ddot{x} = -(u_l + u_r)\sin\theta \\ m\ddot{y} = (u_l + u_r)\cos\theta - mg \\ I\ddot{\theta} = r(u_r - u_l) \end{cases} \tag{10}$$

Using the ground truth model, I can obtain a time series of state evolutions for each of the chosen systems by applying some white noise control intputs. Since this project focuses on the effectiveness of the overall system for controlling nonlinear plants with unknown dynamics, we assume all the states as well as derivatives are directly observable. To make the simulation more realistic, various levels of white noises are added to the observations. The generated synthetic data is then passed down to the system identification algorithms.

3.2 System Identification

For this project, I originally proposed to build on top of the Deep Koopman work and to explore its relationship with linear control theories. However, the qualities of system identification was not good enough for control scenarios. As a result, I steered my project towards the SINDy based system identification methods, and combined that with nonlinear MPC to achieve the control objectives. In this section, I will discuss both SINDy and Koopman based methods regardless of their success or failure.

3.2.1 Deep Koopman with Control

The original Deep Koopman method [3] learns the g function in Equation 4, for dynamical systems without external control inputs. I extended their ideas to incorporate control variables and also to adapt to continuous cases. Generally, we would like to learn some nonlinear transformations

$$z = \varphi_z(x, u) \tag{11}$$
$$v = \varphi_v(x, u) \tag{12}$$

such that we have a linear dynamics on those latent space

$$\dot{z} = Az + Bv \tag{13}$$

for some matrix A and B. A proof-of-concept experiment is conducted considering the special case where u enters the dynamics additively. With this assumption, we could decouple the nonlinear transformations on the states and control inputs. Namely,

$$z = \varphi_z(x) \tag{14}$$
$$v = \varphi_v(u) \tag{15}$$

An overview of the network architecture is shown in Figure 1. In addition to training the autoencoder that should reconstruct the original states and inputs, we also want the network to learn a Koopman invariant subspace that is capable of computing the dynamics with linear equations. Therefore, it is necessary to obtain ground truth latent space derivatives, $\dot{z}$, from the training data. Applying the chain rule from calculus, we obtain

$$\begin{aligned} \dot{z} &= \frac{d}{dt}\varphi_x(x) \\ &= \nabla\varphi_x(x) \cdot \frac{dx}{dt} \\ &= \nabla\varphi_x(x) \cdot \dot{x} \end{aligned} \tag{16}$$

With the help of modern deep learning framework such as TensorFlow, $\nabla\varphi(x)$ can be easily obtained from automatic differentiation. Now we can construct the different losses for training the network.

1. Autoencoder reconstruction loss for states

$$\mathcal{L}_x = \|x - \tilde{\varphi}_x^{-1}(\varphi_x(x))\|^2 \tag{17}$$

2. Autoencoder reconstruction loss for control inputs

$$\mathcal{L}_u = \|u - \tilde{\varphi}_u^{-1}(\varphi_u(u))\|^2 \tag{18}$$

3. Koopman space linear dynamics loss

$$\mathcal{L}_\mathcal{K} = \|\nabla\varphi_x(x)\dot{x} - (A\varphi_x(x) + B\varphi_u(u))\|^2 \tag{19}$$

The final loss is a weighted summation of the three terms. We can then jointly train the weights of the neural network and the linear dynamics matrices A and B.

3.2.2 SINDy with Control [7]

Unlike Koopman baesd methods, SINDy is an algorithm that explicitly reconstructs the structure of the dyanmics. In order to obtain the structure of Equation 2 for arbitrary sytsem, SINDYc method assumes the governing equation to be the form of a summation of multiple nonlinear terms. For example, the Lorenz Attractor as modeled by Equation 9 can be rewritten in the form of Equation 7.

$$\underbrace{\begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}x & y & z & u & xy & xz\end{bmatrix}}_{\theta(x,u)} \underbrace{\begin{bmatrix}-\sigma & \rho & 0 \\ \sigma & -1 & 0 \\ 0 & 0 & -\beta \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0\end{bmatrix}}_{A} \tag{20}$$

With the synthetic dynamical data, we are able to obtain a time series of measurements on both the states and the derivatives. We can construct the nonlinear dictionary $\theta(\cdot, \cdot)$ with some commonly seen nonlinearity such as polynomials and sinusoids. The problems for estimating the dynamical equation then reduces to a linear regression problem.

$$\begin{aligned} A &= \underset{A}{\operatorname{argmin}} \|\boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{u})A - \dot{\boldsymbol{x}}\|^2 \\ &= (\boldsymbol{\theta}^T\boldsymbol{\theta})^{-1}\boldsymbol{\theta}^T\dot{\boldsymbol{x}} \end{aligned} \tag{21}$$

However, in reality, the measured data is noisy, and doing the direct linear regression will result in a dense coefficient matrix A. To solve this problem, we follow the Sequential Thresholded Least Square algorithm as mentioned in SINDy [6]. Instead of directly performing the linear regression, we solve for a ridge linear regression, which penalizes the magnitude of the coefficients with an $\mathcal{L}_2$ norm.

$$\begin{aligned} A &= \underset{A}{\operatorname{argmin}} \|\boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{u})A - \dot{\boldsymbol{x}}\|^2 + \lambda\|A\|^2 \\ &= (\boldsymbol{\theta}^T\boldsymbol{\theta} + \lambda I)^{-1}\boldsymbol{\theta}^T\dot{\boldsymbol{x}} \end{aligned} \tag{22}$$

We can drop out terms in A that are below some threshold, and iteratively perform the ridge linear regression on a subsets of dimensions, until the number of nonzero coefficients converges. This will help obtain a parsimonious model that not only accurately estimates the dynamics, but also correctly recovers the underlying structure of the governing equation.

3.3 Finite Horizon MPC

For controlling an identified plant, this project uses the direct transcription method, which explicitly formulate the

finite horizon MPC as a constrained optimization problem. Consider the following discrete dynamics,

$$x_{k+1} = F(x_k, u_k) \tag{23}$$

we can define a finite window h such that the MPC can optimize for the future h-step control inputs. The cost at each step can be generally formulated as

$$J = m(x_h) + \sum_{i=1}^{h} l(x_i, u_{i-1}) \tag{24}$$

where $h(\cdot, \cdot)$ imposes some per-step penalties on both the states and the control inputs and $m(\cdot)$ imposes some penalty on the final state of the future window. For the purpose of this project, I simplified the cost by removing the m function and by applying quadratic costs for both control inputs and states

$$J = \sum_{i=1}^{h} x_i^T Q x_i + u_{i-1}^T R u_{i-1} \tag{25}$$

where Q and R are some positive semi-definite matrices. To decouple the effects between different states and inputs, the matrices can be further simplified to be diagonal. The diagonal entires then can be interpreted as a tradeoff on how much we want to regulate each state and each inputs. The MPC controller can then be formulated as joint optimization of the states and control inputs subject to the dynamics constraints. Namely,

$$\begin{aligned} \underset{\substack{u_0,\dots,u_{h-1} \\ x_1,\dots,x_h}}{\text{minimize}} \quad & J \\ \text{subject to} \quad & x_{k+1} = F(x_k, u_k),\ k = 0, \dots, h-1 \\ & \text{additional constraints} \end{aligned} \tag{26}$$

This problem can be solved using `Scipy Optimization Toolbox`. The system identification, however, gives continuous time dynamics. Therefore, we need to discretize it in order to make use of the finite dimensional optimization tools described above. We first notice the integral form as following.

$$x(t + \Delta t) = x(t) + \int_t^{t+\Delta t} f(x(\tau), u(\tau)) d\tau \tag{27}$$

If we use small enough time step, and approximate the integral with forward Euler integration, we can write the dynamics in a discrete fashion.

$$x_{k+1} \approx x_k + f(x_k, u_k)\Delta t = F(x_k, u_k) \tag{28}$$

However, in order to make Euler integration accurate, the time step often needs to be on the order of 10^{-3} second. On the other hand, the finite prediction horizon is usually around 5 seconds depending on the speed of the dynamics. This introduces way too many variables for the optimization problem. This issue can be mitigated by reusing the same control inputs to perform multiple integration steps. For example we can keep a integration step of 10^{-3} second, but performing a dynamical prediction every 100 integrations. Effectively, we will achieve a prediction step of 0.1 second. This method still maintain the discrete dynamics structure, $x_{k+1} = f(x_k, u_k)$, while making the optimization tractable by reducing the number of free variables.

4 Results

In this section, I will first analyze the failure of my originally proposed Deep Koopman with Control method, and then I will analyze the performance for SINDYc method on all three nonlinear dynamical systems described in Section 3.1. I will also discuss the control performance on those systems with the identified model.

4.1 System ID with Koopman

An experiment is conducted to learn a single pendulum dynamics using the method proposed in Section 3.2.1. The neural network is constructed with 2 hidden layers for both encoders and decoders, where there are 80 hidden neurons in each layer. The latent space for the states and control inputs has 16 and 2 dimensions respectively, i.e. $z \in \mathbb{R}^{16}$ and $v \in \mathbb{R}^2$. The validation settings for the pendulum is using a random intial condition with a sinusoidal input. Figure 2 shows a qualitative result of using the learned Koopman space to perform linear dynamics update with a given initial condition. When comparing with the ground truth in this one-second window, it is clear that the Koopman space prediction does not capture the nonlinear behaviors well. In fact, the prediction diverges quickly from the ground truth. However, we can still see the training process having some effects on forcing the linear prediction to have locally accurate prediction. Figure 3 shows another comparison where the prediction is manually brought back to the ground truth every 50 ms. Still, the model prediction is not accurate enough to perform any nonlinear control alogrithm.

4.2 System ID with SINDYc

Using the SINDYc framework with Sequential Thresholded Least Square solver, I am able to obtain very accurate model despite the presence of noise in the training data. Table 1 shows the system identification performance on all of three systems:

Noise Level	Pendulum		Lorenz Attractor		Planar Rotor	
	Validation MSE	Structure Recovered	Validation MSE	Structure Recovered	Validation MSE	Structure Recovered
1e-3	2.42e-7	✓	8.34e-7	✓	2.32e-6	✓
5e-3	4.80e-6	✓	3.44e-6	✓	3.09e-6	✓
1e-2	2.26e-4	✓	7.03e-5	✓	4.03e-6	✓
5e-2	5.33e-3	✗	3.28e-4	✓	1.44e-5	✓
1e-1	-	-	2.23e-3	✓	1.41e-3	✗

Table 1: SINDYc Performance

- The noise standard deviation is represented by the number of multiples of the original dataset variation.
- Validation MSE calculates a metric on how good the model estimation is on the validation dataset.
- The algorithm successfully recovers the structure of the dynamics if it identifies the correct nonzero terms in the sparse matrix.

Hyperparameter settings for each of the systems are attached in the Appendix Section B.

4.3 MPC with SINDYc

With the identified systems, now we can control those plants with the nonlinear MPC as described in Section 3.3. For each of the systems, I used the regression model estimated from the dataset with noise level 1e-2. This way, we get a descent amount of noise while ensuring correct model structure recovery. Animated results are contained in the notebooks from the source code.

4.3.1 Pendulum

The ground truth model parameters in Equation 8 are specified in Table 2. An absolute torque saturation of 1.2

m [kg]	l [m]	b [N·m / (rad/s)]	I [kg·m^2]
1	2/5	1/10	4/25

Table 2: Pendulum Model Parameters

N·m is also applied to the MPC constraints. Figure 4 shows that under such extreme saturation, the optimization is sitll capable of coming up with swing up / down controllers to perform accurate reference tracking.

4.3.2 Lorenz Attractor

The ground truth model parameters in Equation 9 are specified in Table 3. This system is particularly more difficult for both identification and control. A slightest bit of error will cause the prediction to diverge exponentially due to its chaotic nature. More interestingly, this system is underactuated. It has 3 degrees of freedom while having only one actuator. Nevertheless, Figure 5 demonstrated that the system is still capable of performing good reference trakcing in such challenging chaotic scenario.

σ	ρ	β
10	28	8/3

Table 3: Lorenz Attractor Parameters

4.3.3 Planar Rotor

The planar rotor model is a 2D simplification of a quadcopter. It is closer to a real-world physical system compared to the above two systems. The ground truth model parameters in Equation 10 are specified in Table 4. This experiment specified the rotor to fly between two waypoints while performing a flip for each waypoint transition. With an accurately identified model, the nonlinear MPC algorithm is able to explore the highly nonlinear state space to perform such maneuver. Figure 6 shows both the rotation tracking and position tracking of the rotor, and we can see the flip behavior is done by accurately tracking the rotation reference.

m [kg]	l [m]	I [kg·m^2]
2	1/2	1/24

Table 4: Planar Rotor Parameters

5 Discussion

For the failure of learning a descent linear embedding for a nonlinear pendulum system, there are several explanations that could back up this experimental results. By

Koopman Theory, it is known that not all systems can be trasformed into a finite dimensional Koopman invariant subspace. On the Control Theory side, we also know that linear systems can only have one fixed point at the origin, so it cannot really represent any systems with multiple fixed points. Even though the single pendulum is one of the simplest nonlinear systems, its phase plane exhibits an orbital behavior with infinite numbers of fixed points. Both of these theories explain why it is very hard to obtain a finite dimensional approximation of that infinite dimensional space where the Koopman Operator lives in. There is, I believe, another direction to pursue. We can relate more closely to the general forms of Koopman Analysis with Control given by [4]. Instead of seperately learning the embeddings for states and control inputs, we can jointly perform the estimation. However, this will also require additional thoughts to make a connection with linear control theories as it changes the formulations.

On the other side, this project successfully demonstrated the capabilities of combining SINDYc system identification method with nonlinear Model Predictive Control. It is really fascinating to see how we could create intelligent control behaviors on highly nonlinear systems with unknown dynamics. A future direction is to make the system identification run online in an adaptive fashion. This could also potentially deal with time varying dynamics in the system. Another interesting direction is to come up with control strategies that provide more information to the estimation process when we don't have a reliable model. This can potentially reduce the amount of data needed to identify an accurate and parsimonious model. One challenge of the SINDYc method is that it does not perform well with implicit dynamics such as double pendulums or acrobots. Equivalently, those implicit dynamics introduce many fractional nonlinearities that are really hard to model as nonlinear libraries. Some work, such as [9], has been done to tackle this difficulty, but it remains a big challenge for SINDy based nonlinear identification methods.

References

[1] P. J. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of fluid mechanics*, vol. 656, pp. 5–28, 2010.

[2] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[3] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, no. 1, p. 4950, 2018.

[4] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[5] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[6] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the national academy of sciences*, vol. 113, no. 15, pp. 3932–3937, 2016.

[7] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Sparse identification of nonlinear dynamics with control (sindyc)," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 710–715, 2016.

[8] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22445–22451, 2019.

[9] K. Kaheman, J. N. Kutz, and S. L. Brunton, "Sindy-pi: A robust algorithm for parallel implicit sparse identification of nonlinear dynamics," *arXiv preprint arXiv:2004.02322*, 2020.

Appendices

A Source Code

All source codes are git tracked in this repository

B SINDYc Settings

All hyperparameters for applying the SINDYc algorithm are included in the `<model>_params.py` files in the source code. Table 5 shows a summary of those parameters. λ is the $\mathcal{L}_2$ regularization defined in Equation 22. The thresholds are used in the Sequential Thresholded Least Square algorithm to drop out small coefficients at the end of each iteration.

Model	Nonlinear Library	λ	Threshold
Pendulum	Identity Sinusoids	8	0.1
Lorenz Attractor	Polynomial (deg. 2)	1	0.1
Planar Rotor	Identity Sinusoids Multiplication Constant	20	0.1

Table 5: SINDYc Hyperparameters

C Figures

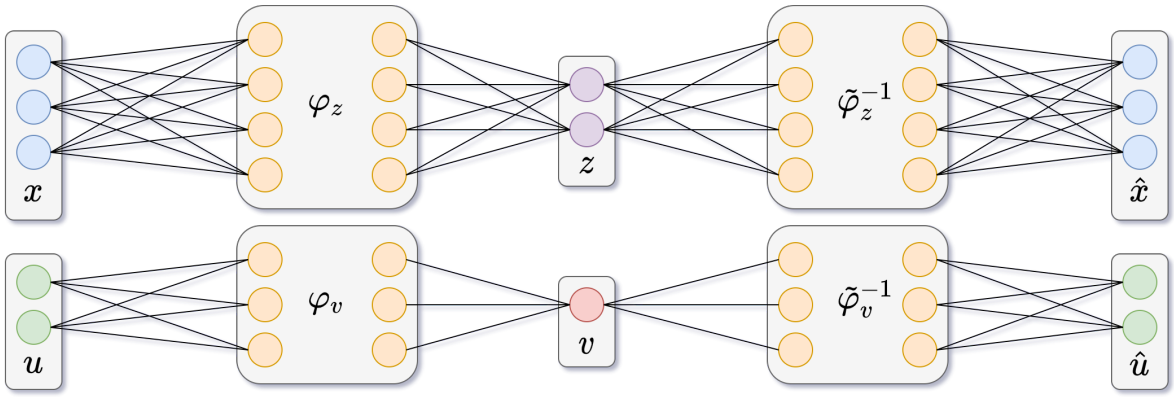

Figure 1: Deep Koopman with Control

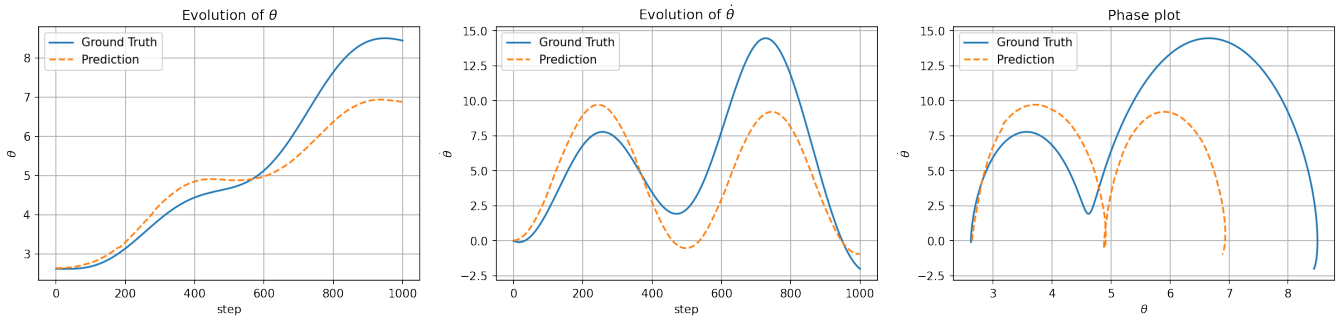

Figure 2: Identified Pendulum with Koopman Space Prediction

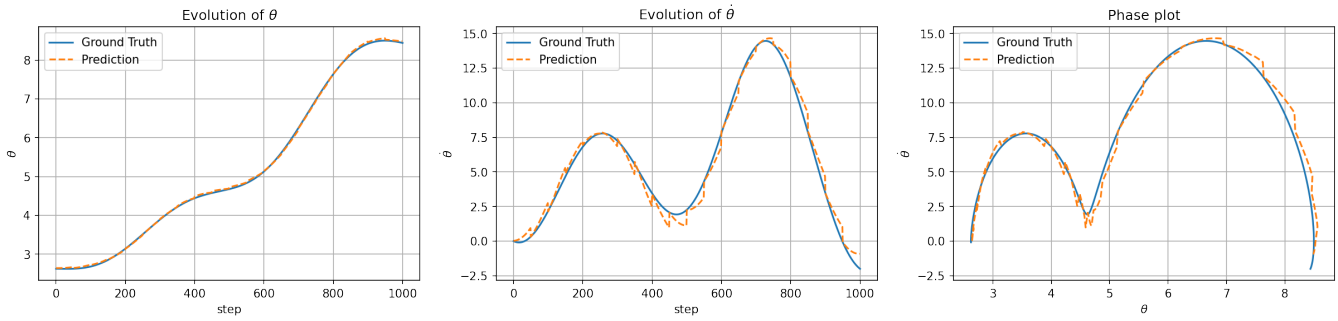

Figure 3: Identified Pendulum with Koopman Space Prediction (with periodic correction)

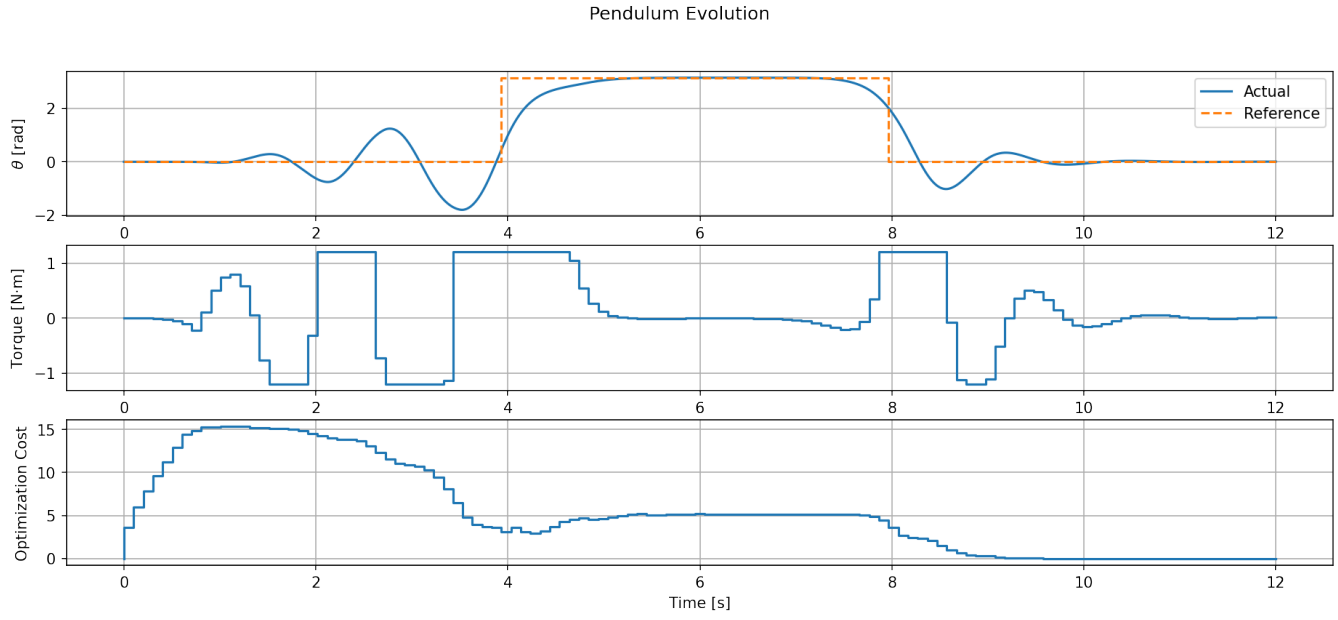

Figure 4: SINDYc with MPC – Pendulum Control

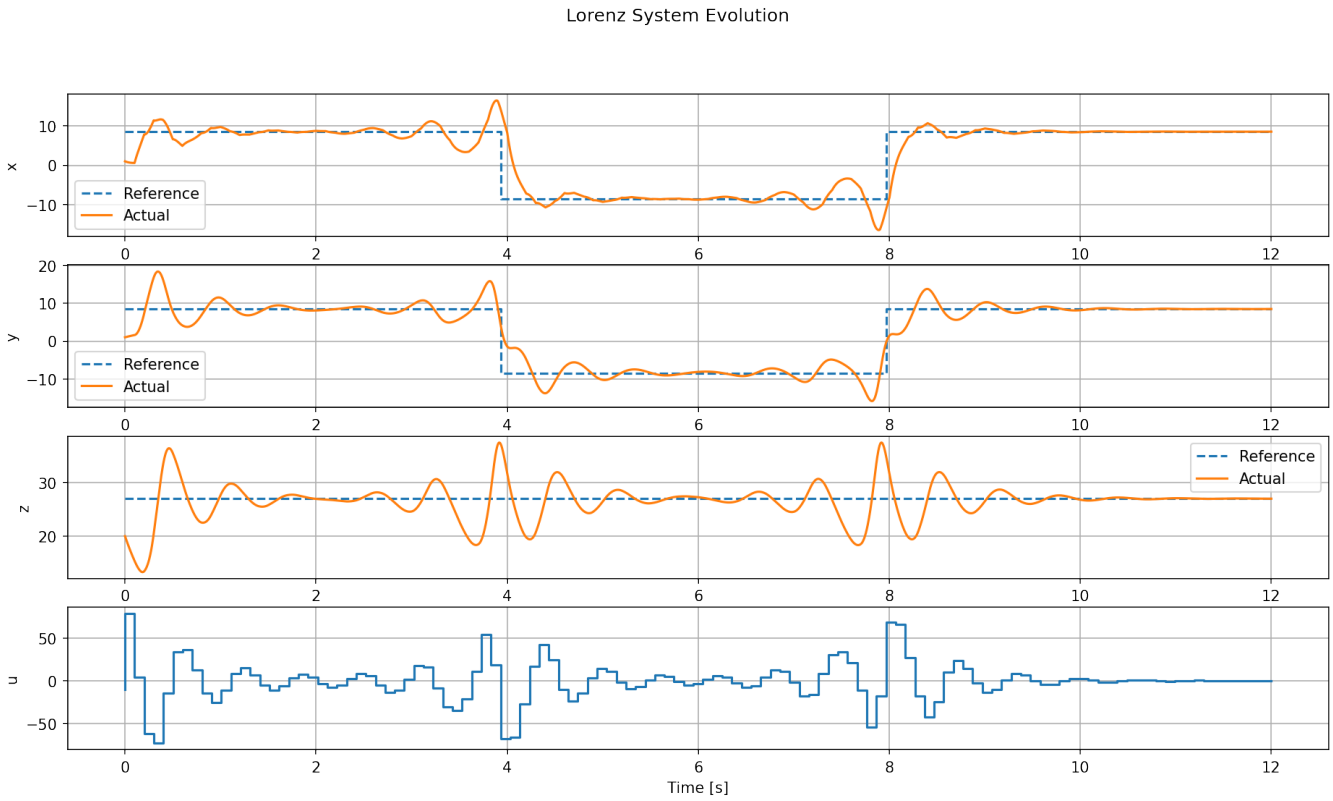

Figure 5: SINDYc with MPC – Lorenz Attractor Control

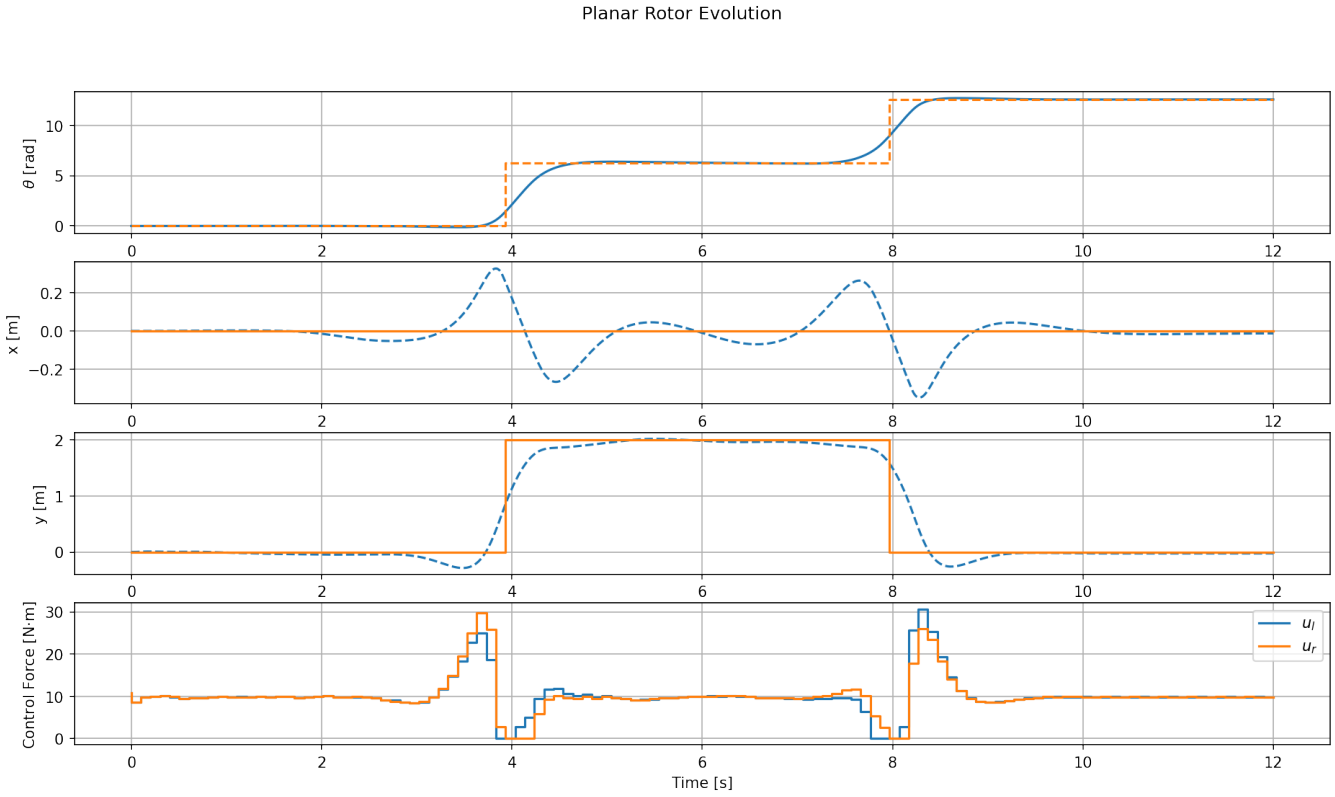

Figure 6: SINDYc with MPC – Planar Rotor Control