Data-Driven Discovery of Differentially Flat Coordinates

Jedidiah Alindogan
Benjamin Rivière

Abstract

The vast majority of dynamical systems found in nature are nonlinear and difficult to control. Yet, the control of such high-dimensional and nonlinear systems is continuously sought. Using differentially flat coordinates is a well-known method to plan trajectories in such systems. Differentially flat coordinates are outputs that explicitly encode information from the state and control of a differentially flat system without integration. Through these flat coordinates, trajectories for a system can be rapidly computed. However, the transforms between state and control space and flat space are difficult to discover. In this research, we develop a machine learning architecture capable of finding differentially flat transforms that led to a successful trajectory generation demonstration for the unicycle system. The modifications that led to this success include enforced differential structure in the latent space; continuous sinusoidal training data; split neural networks for each component of state, control, and flat space; and a closed loop controller for trajectory generation.

1 Related Work

The underlying concept of this approach was developed by Benjamin Rivière. His previous work on the concept leveraged differential flatness, neural networks, and sparse identification of nonlinear dynamics (SINDy) to plan trajectories for a dynamic system. Using an autoencoder, approximations for the differentially flat transforms of the system were rapidly discovered through reconstruction training. The encoder was composed of a SINDy library of polynomial functions with a coefficient vector as a learned parameter. The decoder consisted of two neural networks to reconstruct state and control. With this architecture, reconstruction was mildly successful but trajectory generation failed. Our work identified and addressed the issues of the previous model.

2 Problem Statement

The Control Boundary Value Problem is: given a dynamical system $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$ where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$, initial state $\mathbf{x}_{t_0}$, initial time t_0, terminal state $\mathbf{x}_{t_f}$, and terminal time t_f, find the control inputs $\mathbf{u}(t)$ that drive the initial state to the terminal state, i.e.:

$$\begin{aligned} \mathbf{u}_0, \ldots, \mathbf{u}_K = \arg \min_{\mathbf{u}_0,\ldots,\mathbf{u}_K} 1 \text{ s.t.} \\ \mathbf{x}_0 = \mathbf{x}_{t_0} \\ \mathbf{x}_K = \mathbf{x}_{t_f} \\ \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{1}$$

While there are many ways to solve this problem, a widely known method is to use differential flatness. According to [1], a dynamical system is differentially flat if there exists outputs $\mathbf{y} \in \mathbb{R}^m$ of form

$$\mathbf{y} = \mathbf{y}(\mathbf{x}, \mathbf{u}, \dot{\mathbf{u}}, \ldots, \mathbf{u}^{(p)}) \tag{2}$$

such that

$$\begin{aligned} \mathbf{x} &= \mathbf{x}(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(q)}) \\ \mathbf{u} &= \mathbf{u}(\mathbf{y}, \dot{\mathbf{y}}, \ldots, \mathbf{y}^{(q)}) \end{aligned} \tag{3}$$

These systems are particularly useful for rapid trajectory calculation. A primary example of this is the quadrotor system. Differential flatness has been utilized to dramatically improve trajectory computation for quadrotors [2]. Because flat coordinates explicitly encode the state and control for a flat system, a difficult CBVP can be encoded into flat space, solved more easily, and then decoded back into state and control. To solve this problem with a machine learning architecture, the following tasks must be achieved[1]:

Task 1. Trajectory Reconstruction: given an input state trajectory $\mathbf{X}$ and control trajectory $\mathbf{U}$, the model must output an approximate state trajectory $\widehat{\mathbf{X}}$ and approximate control trajectory $\widehat{\mathbf{U}}$ such that $\mathbf{X} \approx \widehat{\mathbf{X}}$ and $\mathbf{U} \approx \widehat{\mathbf{U}}$.

Task 2. Trajectory Generation: given a control boundary value problem, the model must successfully encode the boundary conditions $\mathbf{x}_{t_0}$ and $\mathbf{x}_{t_f}$ with arbitrary control to approximately differentially flat conditions $\widehat{\mathbf{y}}_{t_0}$ and $\widehat{\mathbf{y}}_{t_f}$. Afterwards, the problem must be solved in this differentially flat space, and decoded into a differentially flat solution to yield a control trajectory that satisfies the control boundary value problem.

3 Methodology

Prior to discussing the issues addressed in the previous model, an overview of the previous model will be considered.

3.1 Previous Model Overview

The previous model considered the following autoencoder structure for reconstruction during training:

$$\left\{ \begin{array}{c} \mathbf{X} \\ \mathbf{U} \end{array} \right\} \rightarrow \widehat{\mathbf{Y}}^{(q)}(\mathbf{X}, \mathbf{U}) \rightarrow \left\{ \begin{array}{c} \widehat{\mathbf{X}}(\mathbf{Y}^{(q)}) \\ \widehat{\mathbf{U}}(\mathbf{Y}^{(q)}) \end{array} \right\}$$

Figure 1: Trajectory Reconstruction of Previous Model

where $\mathbf{U}$ is generated through a uniform random exploration policy where each $\mathbf{U}(t_i)$ is sampled from a uniform distribution. Afterwards, $\mathbf{U}$ is rolled out on a uniform random initial position $\mathbf{x}_0$ to generate $\mathbf{X}$. This rollout is computed via Euler integration. Moreover, the loss of this model was computed as

$$L = \lambda_1 \|\mathbf{X} - \widehat{\mathbf{X}}\|_2^2 + \lambda_2 \|\mathbf{U} - \widehat{\mathbf{U}}\|_2^2 + \lambda_3 \|\tilde{\mathbf{Y}}^{(q)} - \widehat{\mathbf{Y}^{(q)}}\|_2^2 + \lambda_4 \|\boldsymbol{\Xi}\|_1 \tag{4}$$

where λ_i are selected weights.

This loss function is composed of losses in reconstruction of state, reconstruction of control, variance between the a SINDy library representation of the flat derivatives and the neural network encoding of flat derivatives, and a sparsity on the SINDy library coefficient vector, respectively. Because the SINDy library computation was removed in the new model and is not the focus of this research, the details of this computation will be omitted.

[1] See notation table at the end

In the previous model, trajectory generation is accomplished via a polynomial optimization sourced from [3], available at `https://github.com/whoenig/uav_trajectories`. Initially, state and control waypoints are encoded into flat waypoints. Afterwards, polynomial optimization is applied to the flat waypoints to develop a flat trajectory. Finally, this flat trajectory is decoded into a desired state trajectory and control trajectory.

$$\left\{ \begin{array}{c} \mathbf{x}_p \\ \mathbf{u}_p \end{array} \right\} \rightarrow \widehat{\mathbf{Y}}^{(2)}(\mathbf{x}_p, \mathbf{u}_p) \rightarrow \texttt{polynomial optimization} \rightarrow \widehat{\mathbf{Y}^{(2)}} \rightarrow \left\{ \begin{array}{c} \widehat{\mathbf{X}}(\widehat{\mathbf{Y}^{(2)}}) \\ \widehat{\mathbf{U}}(\widehat{\mathbf{Y}^{(2)}}) \end{array} \right\}$$

Figure 2: Trajectory Generation of Previous Model

3.2 Expanding Issue Identification Methods

Previously, issue identification was limited to interpretation of plots for trajectory reconstruction, trajectory generation, and loss. The visualization and analysis of these plots are considered below:

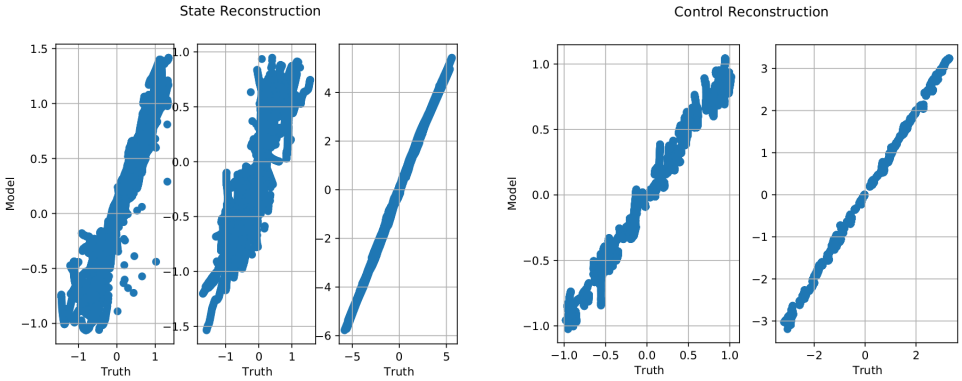


Figure 3: Unicycle Reconstruction of Previous Model by Component [4]

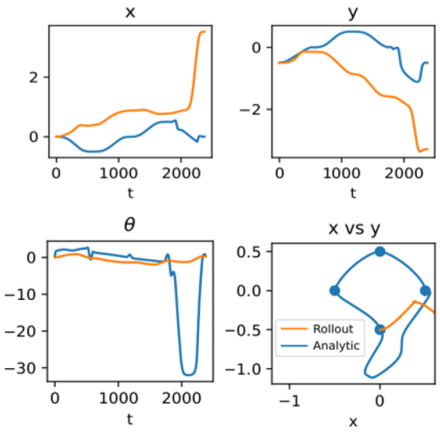


Figure 4: Unicycle Trajectory Generation of Previous Model [4]

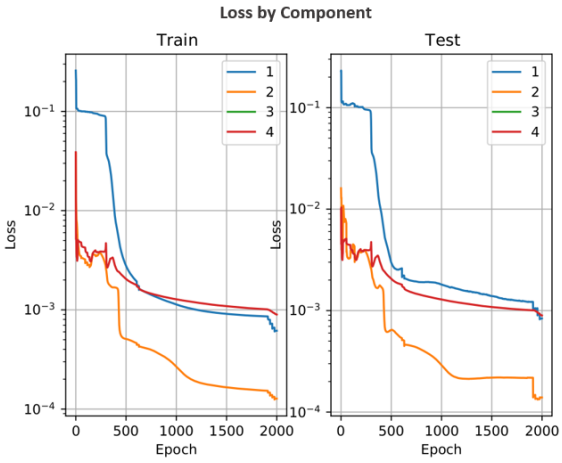


Figure 5: Unicycle Loss of Previous Model by Dataset and Component [4]

As visualized in Figure 3, the trajectory reconstruction of a unicycle seemed to only be successful for the last components. In these reconstruction plots, true values of a component were compared against the model's

reconstruction of those values. Thus, because the reconstruction for the first two components of state and first component of control are noisy and barely resemble a line with a slope of one, the previous model performed poorly. Furthermore, the trajectory generation of the previous model failed, as seen in Figure 4. The rolled out trajectory was unable to approach the specified waypoints unlike an analytic plan involving the same way points. Despite these errors, each component of the loss function was convergent. This implied that the model was finding some local minimum that allowed for moderate reconstruction, but divergent rollouts in the trajectory generation.

To help identify the issues in reconstruction and generation, supplementary plots were developed. A visual of the state and control space was added to confirm that the training data is large enough to represent the space of interest. Moreover, plots for $\mathbf{Y}^{(q)}$ were developed to view the differential relationship being imposed on the flat space. Lastly, histogram plots for the state space and control space were created to check for unnatural values or frequent occurrences of other values.

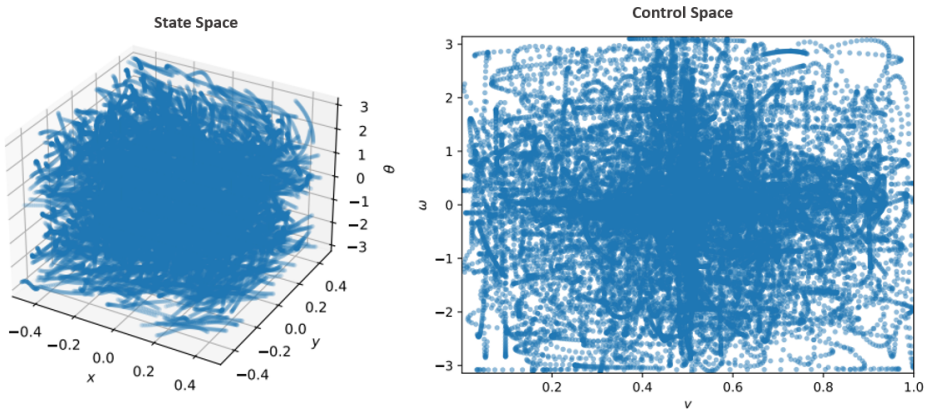


Figure 6: UnicycleState and Control Domain of 1000 Trajectories, 50 Points per Trajectory, and a dt of 0.05 with a Sinusoidal Exploration Policy

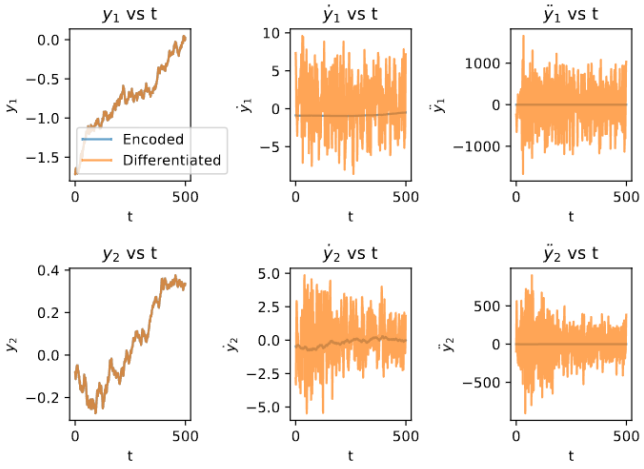


Figure 7: Unicycle Comparison of Encoded and Differentiated Flat Coordinates for the Previous Model

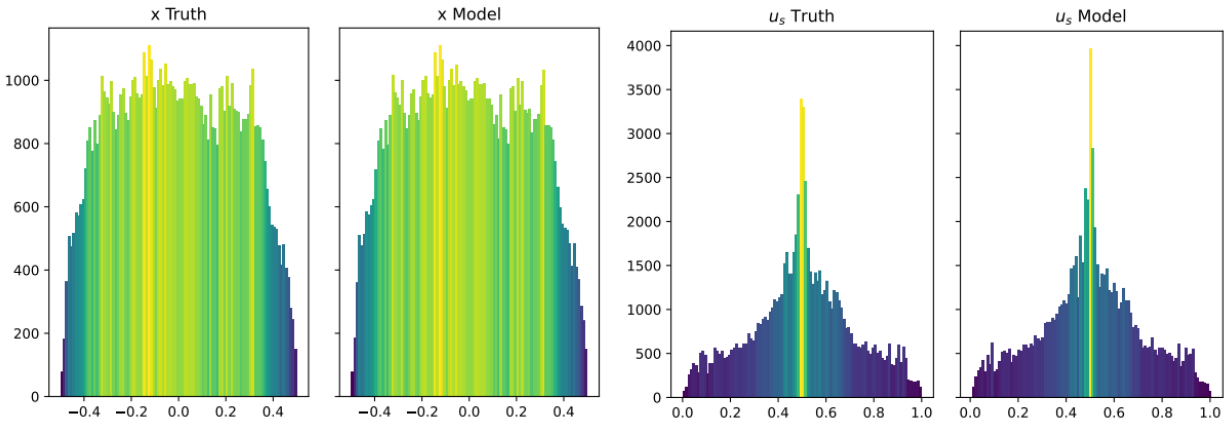


Figure 8: Unicycle Histograms for Horizontal Position x and Speed u_s of a Unicycle for 102400 Datapoints with a Sinusoidal Exploration Policy

3.3 Fixing Differential Structure

After new plots were developed, we determined that the most pressing issue of the model maintaining a differential structure for $\widehat{\mathbf{Y}}^{(q)}$. As seen in Figure 7, the encoded flat coordinate and its encoded first two derivatives were compared against the encoded flat coordinates and the first two numerical derivatives of the encoded flat coordinates. The numerical derivatives were dramatically higher than the encoded derivatives. Thus, the previous model was unable to maintain a differential structure despite being given the SINDy differential structure in the loss function expressed by the λ_3 term.

Because of their unstable values in numerical derivatives, the state and control trajectories originally generated by random rollouts were replaced with smooth state $\mathbf{X}$ trajectories rolled out by smooth control $\mathbf{U}$ trajectories. These smooth control trajectories were generated using a sinusoidal exploration policy given by

$$\mathbf{U}_i(t) = \sum_{j=0}^{N} A_{ij} \sin(B_{ij}t + C_{ij}) \tag{5}$$

where $N \in \mathbb{N}$ and the coefficients A_j, B_j, C_j are random constrained such that $\sum_{i=0}^{N} A_i$ is within the i^{th} control component's boundaries.

In addition to the continuous state trajectories, three more changes were implemented. Firstly, a one dimensional convolution was applied to flat outputs $\mathbf{Y}$ before differentiation. Secondly, the integration method for the rollout was changed from Euler integration to Runge-Kutta 45 integration. This introduced more accurate rollouts of state trajectories $\mathbf{X}$. Lastly, instead of utilizing both state $\mathbf{X}$ and control $\mathbf{U}$ trajectories as input for $\widehat{\mathbf{Y}}$, control $\mathbf{U}$ was removed from the input to simplify the problem. This simplification was inspired by the fact that the unicycle's flat coordinates depend only on state. In other words, for the unicycle model:

$$\widehat{\mathbf{Y}} = \widehat{\mathbf{Y}}(\mathbf{X}) \tag{6}$$

The combination of these methods ensure a differential structure in the flat space with approximately continuous derivatives.

3.4 Modifying Model Architecture

Despite fixing the differential structure, the model was still unable to perform trajectory reconstruction or trajectory generation. Thus, we concluded that changes to the inherent structure of the model had to be made. However, deciding on the proper model complexity and architecture was difficult due to the wide variety of modifications available. To expedite the process, single trajectory tests were utilized. Before training a model on an entire data set, a model was first tested to fit to a single trajectory. Because the single trajectory is much smaller in comparison to an entire data set, it can be used to quickly test if a model is too simple. More precisely, a model is too simple if it is unable to fit to the single trajectory.

In addition to the single trajectory test, the known transformations $\mathbf{X}, \mathbf{U}$, and $\mathbf{Y}$ for a unicycle were utilized to help develop the neural network encoding $\widehat{\mathbf{Y}}$ and decoding $\widehat{\mathbf{X}}, \widehat{\mathbf{U}}$ structures. For example, by replacing the neural network decoding $\widehat{\mathbf{X}}, \widehat{\mathbf{U}}$ with the known decoding transformations $\mathbf{X}, \mathbf{U}$ for a unicycle, the neural network encoding structure can be modified until the model is successful. This is visualized in Figure 9:

$$\left\{ \begin{array}{c} \mathbf{X} \\ \mathbf{U} \end{array} \right\} \to \widehat{\mathbf{Y}}^{(2)}(\mathbf{X}, \mathbf{U}) \to \left\{ \begin{array}{c} \mathbf{X}(\mathbf{Y}^{(2)}) \\ \mathbf{U}(\mathbf{Y}^{(2)}) \end{array} \right\}$$

Figure 9: Model with Known Decoding for a Unicycle

Here, the encoding neural network $\widehat{\mathbf{Y}}^{(2)}$ is adjusted until the model is capable of fitting to a single trajectory. Combining these strategies for both encoding and decoding, the model structure was changed such that each component received its own neural network. In other words, the new model no longer uses a single neural network $\widehat{\mathbf{Y}}^{(q)}$ to encode state trajectories $\mathbf{X}$ to flat trajectories up to the q-th derivative. Instead, it uses n neural networks to encode state trajectories $\mathbf{X}$ to n flat trajectory components $\widehat{\mathbf{Y}}_1, \widehat{\mathbf{Y}}_2, \ldots, \widehat{\mathbf{Y}}_n$. Similarly, the model utilizes neural networks for each state and control component for a total of $n+m$ neural networks for decoding. A summary of the changes made to the previous model can be seen in Figure 10.

$$\text{Rollout } \mathbf{U} \text{ on } \mathbf{x}_0 \to \mathbf{X} \to \left\{ \begin{array}{l} \widehat{\mathrm{Y}}_1(\mathbf{X}) \\ \widehat{\mathrm{Y}}_2(\mathbf{X}) \\ \vdots \\ \widehat{\mathrm{Y}}_n(\mathbf{X}) \end{array} \right\} \to \texttt{numerical differentiation} \to \widehat{\mathbf{Y}}^{(q)} \to \left\{ \begin{array}{l} \widehat{\mathrm{X}}_1(\mathbf{Y}^{(q)}) \\ \widehat{\mathrm{X}}_2(\mathbf{Y}^{(q)}) \\ \vdots \\ \widehat{\mathrm{X}}_m(\mathbf{Y}^{(q)}) \\ \widehat{\mathrm{U}}_1(\mathbf{Y}^{(q)}) \\ \widehat{\mathrm{U}}_2(\mathbf{Y}^{(q)}) \\ \vdots \\ \widehat{\mathrm{U}}_n(\mathbf{Y}^{(q)}) \end{array} \right\}$$

Figure 10: Trajectory Reconstruction of New Model

3.5 Loss Function Modifications

As a consequence of many changes in the architecture, the loss function was modified dramatically. A normalization was added to each term of the loss function. Reconstruction terms were split to match the split neural network structure. Additionally, two rollout terms were added so that the rollout of $\widehat{\mathbf{U}}$ on $\widehat{\mathbf{x}_0}$ would match the desired state reconstruction $\widehat{\mathbf{X}}$ and true state trajectory $\mathbf{X}$. Lastly, the SINDy loss term for the flat derivatives was removed because the differential structure was now enforced by numerical derivatives.

The new loss function takes the following form:

$$\begin{aligned} L = &\sum_{i=1}^{n} \lambda_i \|\underline{\mathbf{X}}_i - \underline{\widehat{\mathbf{X}}_i}\|_2^2 + \sum_{j=1}^{m} \lambda_{n+j} \|\underline{\mathbf{U}}_j - \underline{\widehat{\mathbf{U}}_j}\|_2^2 + \lambda_{n+m+1} \| \underline{\widehat{\mathbf{X}}}_{\text{rollout}} - \underline{\widehat{\mathbf{X}}}\|_2^2 \\ &+\lambda_{n+m+2} \|\underline{\widehat{\mathbf{X}}}_{\text{rollout}} - \underline{\mathbf{X}}\|_2^2 + \lambda_{n+m+3} \sum_{k=1}^{N} \|w_k\|_1 \end{aligned} \tag{7}$$

Here, $\widehat{\mathbf{X}}_{\text{rollout}}$ refers to the RK-45 integration of $\widehat{\mathbf{U}}$ on $\widehat{\mathbf{x}_0}$ or the initial value of $\widehat{\mathbf{X}}$. And, N refers to the total number of weights for all neural networks in the model. Thus, the last term is an L_1 regularization on the weights of the model to encourage sparsity. Moreover, each trajectory is normalized in the loss function according to the formula given by

$$\underline{\mathbf{Z}} = \frac{\mathbf{Z} - \mathbf{Z}_{min}}{\mathbf{Z}_{max} - \mathbf{Z}_{min}} \tag{8}$$

where $\mathbf{Z}_{max}$ and $\mathbf{Z}_{min}$ are the maximum and minimum limits on the space of the trajectory $\mathbf{Z}$.

3.6 Trajectory Generation Modifications

As a result of the modifications to the trajectory reconstruction of the model, numerical differentiation became essential. This dependency required modifications to the trajectory generation method. In particular, the trajectory generation was modified so that waypoints could be converted to two-point trajectories. With a model reliant on numerical derivatives, single points could not be encoded because no derivative information exists for a single point. This problem was addressed by creating two-point trajectories during trajectory generation. Given some state $\mathbf{x}_i$ with an arbitrary control value $\mathbf{u}_i$, the dynamics of the system is utilized to generate the next state $\mathbf{x}_{i+1}$. This pair of state values $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, or $\mathbf{X}_{i,i+1}$, compose a two-point trajectory that can be used by the model to compute the flat derivatives with negligible error. While differential flatness is typically used to plan trajectories with flat coordinates, this method allows trajectories to be planned in state space. Additionally, a polynomial spline was introduced to replace the previous polynomial optimization method to simplify calculations.

The changes made to trajectory generation are summarized in Figure 11.

$$\mathbf{x}_0, \mathbf{x}_f \rightarrow \mathbf{X}_{0,1}\mathbf{X}_{f,f+1} \rightarrow \left\{ \begin{array}{l} \widehat{\mathrm{Y}}_1(\mathbf{X}_{0,1}), \widehat{\mathrm{Y}}_1(\mathbf{X}_{f,f+1}) \\ \widehat{\mathrm{Y}}_2(\mathbf{X}_{0,1}), \widehat{\mathrm{Y}}_2(\mathbf{X}_{f,f+1}) \\ \vdots \\ \widehat{\mathrm{Y}}_n(\mathbf{X}_{0,1}), \widehat{\mathrm{Y}}_n(\mathbf{X}_{f,f+1}) \end{array} \right\} \rightarrow \texttt{polynomial spline} \rightarrow \widehat{\mathbf{Y}}^{(q)} \rightarrow \left\{ \begin{array}{l} \widehat{\mathrm{X}}_1(\mathbf{Y}^{(q)}) \\ \widehat{\mathrm{X}}_2(\mathbf{Y}^{(q)}) \\ \vdots \\ \widehat{\mathrm{X}}_m(\mathbf{Y}^{(q)}) \\ \widehat{\mathrm{U}}_1(\mathbf{Y}^{(q)}) \\ \widehat{\mathrm{U}}_2(\mathbf{Y}^{(q)}) \\ \vdots \\ \widehat{\mathrm{U}}_n(\mathbf{Y}^{(q)}) \end{array} \right\}$$

Figure 11: Trajectory Generation of New Model

4 Results

Testing the model on the unicycle system[2], successful results were obtained for trajectory reconstruction and generation. In particular, the reconstruction of the first two components of the state $\mathbf{X} = (x, y, \theta)$ and the first component of control $\mathbf{U} = (u_s, u_\omega)$ are very accurate. Additionally, there seems to be tolerable error in the last components of both state and control.

The successful model utilizes the following hyperparameters:

Hyperparameter	Value
number of flat derivatives q	2
loss weights λ	[1e+2, 1e+2, 1e-1, 1.0, 1.0, 1.0, 1.0, 1e-2]
number of hidden layers for $\widehat{\mathbf{Y}}$	[0, 0]
number of hidden layers for $\widehat{\mathbf{X}}$	[1, 1, 2]
number of hidden layers for $\widehat{\mathbf{U}}$	[1, 2]
dimension of hidden layers for $\widehat{\mathbf{X}}$	[32, 32, 32]
dimension of hidden layers for $\widehat{\mathbf{U}}$	[32, 32]
activation	ReLU
optimization	Adam
number of trajectories in dataset	2048
number of points in each trajectory	50
number of epochs	600
batch size	32
convolution kernel size	5
convolution padding size	2
convolution padding type	Replication

Table 1: Unicycle Hyperparameters of New Model

To clarify some information, the number of hidden layers for $\widehat{\mathbf{Y}}$ which are $[0, 0]$ means that both $\widehat{\mathbf{Y}}_1$ and $\widehat{\mathbf{Y}}_2$ do not utilize any hidden layers. In other words, $\widehat{\mathbf{Y}}$'s transformation is linear. Furthermore, the number and dimension of hidden layers are given according to each component of the respective trajectory. For example, the number and dimension of hidden layers for $\widehat{\mathbf{X}}$ given as $[1, 1, 2]$ and $[32, 32, 32]$, respectively, refer to neural network parameters for each component of $\widehat{\mathbf{X}}$ or x, y, θ. Lastly, the loss weights λ are matched according to the terms of the loss function applied to a unicycle ($n = 3$, $m = 2$):

$$\begin{aligned} L = &\sum_{i=1}^{3} \lambda_i \|\underline{\mathbf{X}}_i - \underline{\widehat{\mathbf{X}}}_i\|_2^2 + \sum_{j=1}^{2} \lambda_{3+j} \|\underline{\mathbf{U}}_j - \underline{\widehat{\mathbf{U}}}_j\|_2^2 + \lambda_6 \| \underline{\widehat{\mathbf{X}}}_{\text{rollout}} - \underline{\widehat{\mathbf{X}}}\|_2^2 \\ &+\lambda_7 \|\underline{\widehat{\mathbf{X}}}_{\text{rollout}} - \underline{\mathbf{X}}\|_2^2 + \lambda_8 \sum_{k=1}^{N} \|w_k\|_1 \end{aligned} \tag{9}$$

[2]See unicycle dynamics at the end

Although the error in reconstruction for the last components of state and control is tolerable, as demonstrated by the successful trajectory generation, its behavior is peculiar. In particular, their seems to be several lines of different slopes yielding error on the reconstruction of u_ω.

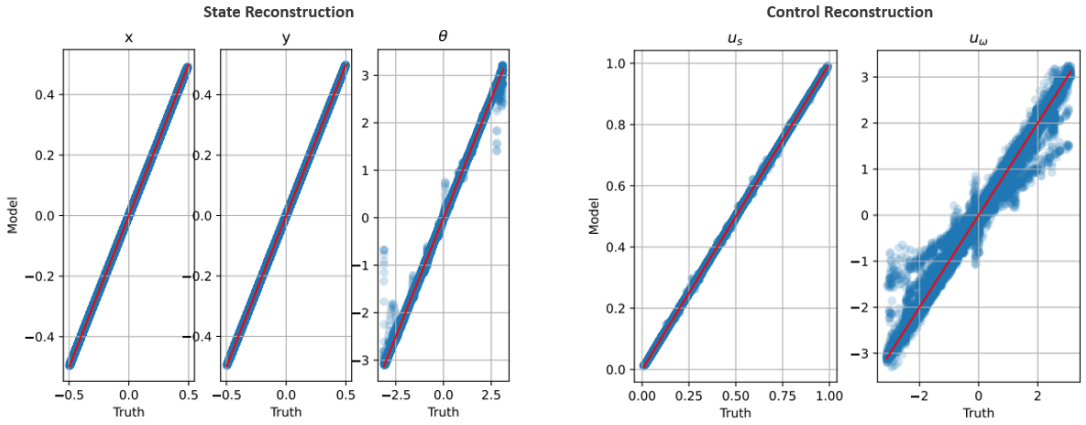


Figure 12: Unicycle Reconstruction of New Model by Component

In comparison, trajectory generation is completely successful with a closed-loop controller. In Figure 13, the model first plans an initial trajectory between two points. Then after rolling out the first ten steps of the control trajectory $\mathbf{U}$, the model plans another trajectory from its current position to the final. This process repeats until the final position is reached. Although this method does not yield the analytic solution generated from using the true differentially flat transformations of the unicycle, a successful trajectory is generated nonetheless.

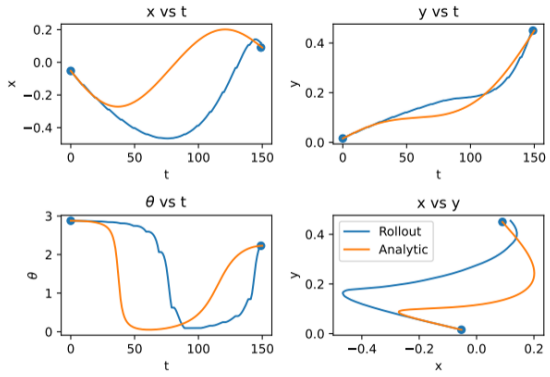


Figure 13: Unicycle Trajectory Generation of New Model

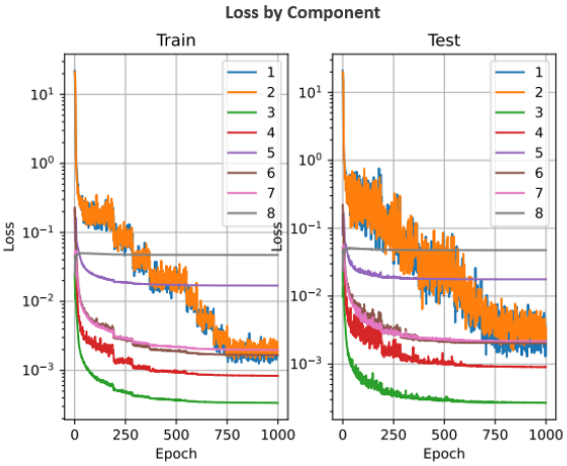
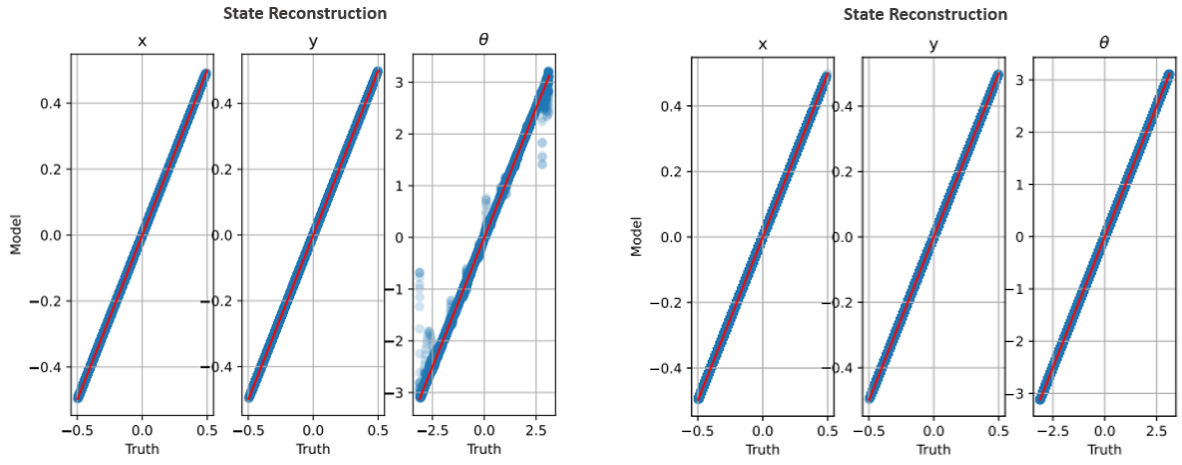


Figure 14: Unicycle Loss of New Model by Dataset and Component

Lastly, each component of the loss function are convergent for both the training and testing datasets as seen in Figure 14.

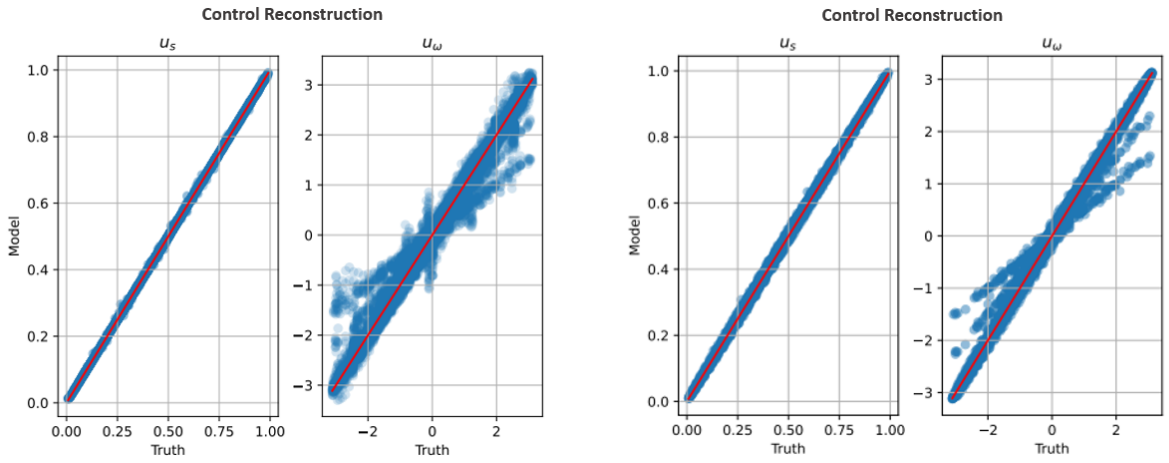
5 Future Work

While the model is largely successful for the unicycle system, there are still other issues of this method that are not yet addressed. As previously mentioned, when comparing the reconstruction between the new model and an analytic model utilizing numeric differentiation and the analytic differentially flat transforms, both the angle θ and angular velocity u_ω err. According to Figure 15, the θ reconstruction possesses error at the endpoints near $\pm\pi$. It is hypothesized that the error may be due to branch cut behavior. In comparison, the cause of error in u_ω reconstruction is more elusive. Both the new model and the analytic model seem to possess other non-one-to-one relationships between the model reconstruction and the true values of u_ω, as seen in Figure 16. It is speculated that this may be due to error with the numerical derivatives at endpoints or singular behavior.



New Model Analytic Model

Figure 15: Comparison of State Reconstruction Between the New Model and Analytic Model



New Model Analytic Model

Figure 16: Comparison of Control Reconstruction Between the New Model and Analytic Model

Despite this issue, we have demonstrated that the data-driven discovery of differentially flat coordinates is feasible and can be used for planning trajectories. In future work, we hope to demonstrate this method's success in more complex systems such as the quadrotor, which is known to be differentially flat. Furthermore, by applying this method to systems known to not be differentially flat, we hope to test whether approximately flat transformations can be discovered.

6 Acknowledgements

We thank Prof. Soon-Jo Chung (California Institute of Technology) for his organization and advisement to make this research possible. We thank NSF Graduate Research Fellow Connor T. Lee (California Institute of Technology) for providing the suggestions and advice on improving the model architecture. Lastly, we thank friends of Dr. Chandler C. Ross for supporting this research.

References

[1] Richard M. Murray, Muruhan Rathinam, and Willem Sluis. "Differential Flatness of Mechanical Control Systems: A Catalog of Prototype Systems". In: *Proceedings of the 1995 ASME Interational Congress and Exposition* (1995).

[2] Charles Richter, Adam Bry, and Nicholas Roy. "Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments". In: *ISRR* 114 (2013), pp. 649–666. DOI: 10.1007/9978-3-319-28872-7_37.

[3] Michael Burri et al. "Real-Time Visual-Inertial Mapping, Re-localization and Planning Onboard MAVs in Unknown Environments". In: (Sept. 2015).

[4] Benjamin Rivière. "Data-Driven Discovery of Differentially-Flat Coordinates". In: *CMS270: Advanced Topics in Computing and Mathematical Sciences: Data-driven modeling of dynamical systems* (2020).

Notation

Symbol	Description
m	state dimension
n	control dimension
x	horizontal position for a unicycle
y	vertical position for a unicycle
θ	angular position for a unicycle
u_s or v	speed for a unicycle
u_ω or ω	angular speed for a unicycle
$\mathbf{x}$	state coordinate
$\mathbf{y}$	flat coordinate
$\mathbf{u}$	control coordinate
$\mathbf{X}$	state trajectory
$\mathbf{Y}$	flat trajectory
$\mathbf{U}$	control trajectory
$\mathbf{Y}^{(q)}$	first q derivatives of the flat trajectory where $q \in 0, 1, 2, ...$
$\mathbf{U}^{(p)}$	first p derivatives of the control trajectory where $p \in 0, 1, 2, ...$
$\widehat{\mathbf{Y}}$	neural network encoding
$\dot{\mathbf{Y}}$	SINDy encoding
$\widehat{\mathbf{X}}$	neural network decoding
$\widehat{\mathbf{U}}$	neural network decoding

Table 2: Notation Guide

Unicycle System

The dynamical system of a unicycle is given by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} u_s \cos\theta \\ u_s \sin\theta \\ u_\omega \end{bmatrix} \tag{10}$$

where the state is

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{11}$$

and the control is

$$\mathbf{u} = \begin{bmatrix} u_s \\ u_\omega \end{bmatrix} \tag{12}$$

The differentially flat transformations of the unicycle are given by

$$\mathbf{y}(\mathbf{x}) = \begin{bmatrix} x \\ y \end{bmatrix} \tag{13}$$

$$\mathbf{x}(\mathbf{y}, \dot{\mathbf{y}}) = \begin{bmatrix} x \\ y \\ \arctan\frac{\dot{y}}{\dot{x}} \end{bmatrix} \tag{14}$$

$$\mathbf{u}(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}) = \begin{bmatrix} \pm\sqrt{\dot{x}^2 + \dot{y}^2} \\ \frac{\dot{x}\ddot{y} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2} \end{bmatrix} \tag{15}$$