Data-Driven Discovery of Vehicle Dynamics Using Probabilistic Bayesian Neural Networks

Chinmay Samak
csamak@clemson.edu

Tanmay Samak
tsamak@clemson.edu

Vasanth Seethapathi
vasants@clemson.edu

Responsibility Assignment

- **Data Collection and Preprocessing:**
  - Tanmay
  - Vasanth
- **Problem Formulation:**
  - Chinmay
  - Tanmay
- **Implementation:**
  - Tanmay
  - Chinmay
- **Hyperparameter Tuning and Pipeline Optimization:**
  - Everyone

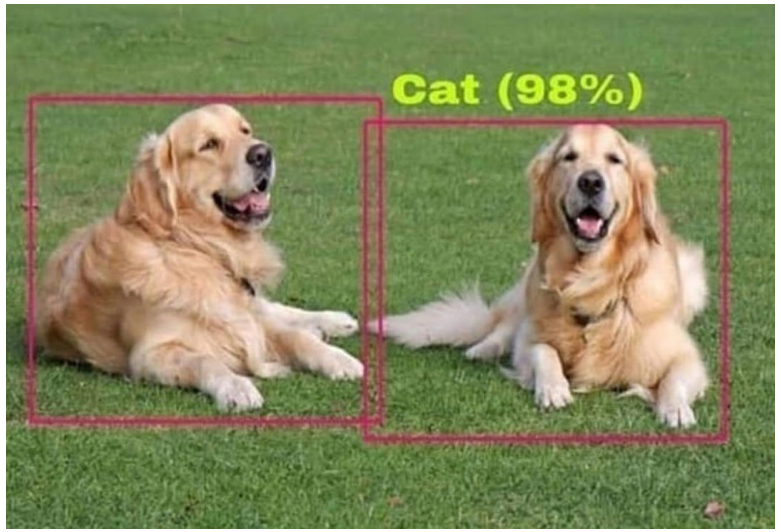
***Note**: Responsibility does not indicate contribution. We have no conflicts of interest to declare.*

Motivation & Objective

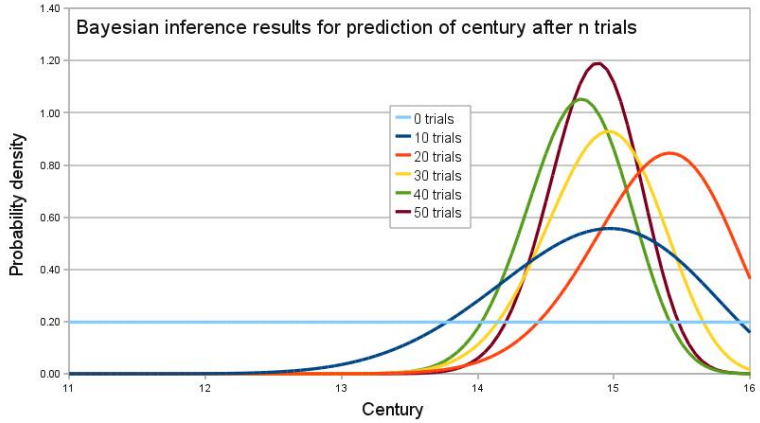
- **Motivation:**
  - Real-time multi-step dynamic simulation of complex multi-body models
  - Deep learning – learn implicit dynamics and generalize across multiple operating conditions
  - Bayesian inference – quantify uncertainty in model predictions for explainability & reliability
- **Objectives:**
  - Forward simulation for state propagation
  - Replace hardware sensor by software model
  - State-estimation in coverage denied areas or in sensor malfunction conditions
  - Redundant variable measurement for more robust state-estimation
  - Predicting anomalies in sensor measurements and fault-tolerant control

Introduction

- **Why Probabilistic Models:**
  - Overconfident point-estimate models
  - Interpretability and explainability
- **Probability and Bayesian Inference:**
  - Probability distributions and sampling
  - Start with a prior belief
  - Provide/sample new data
  - Update (variational) posterior belief



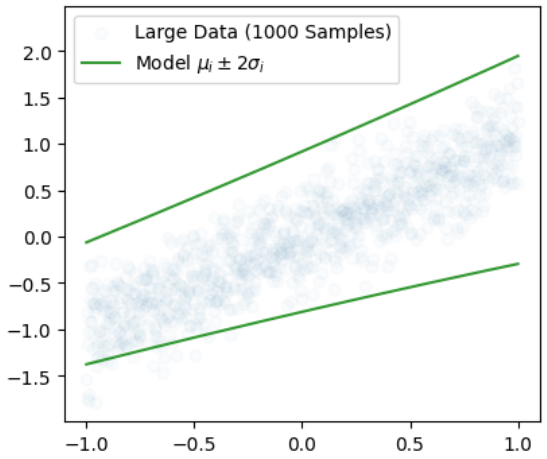
Source: Reddit

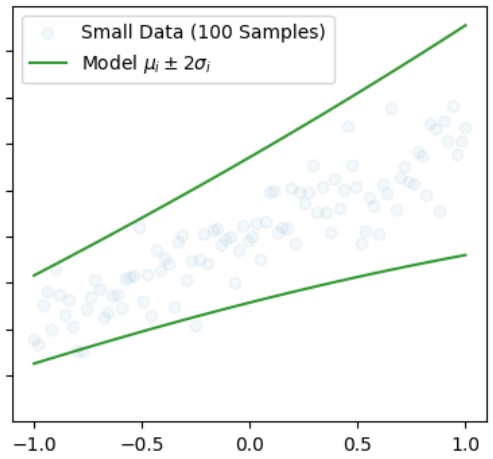


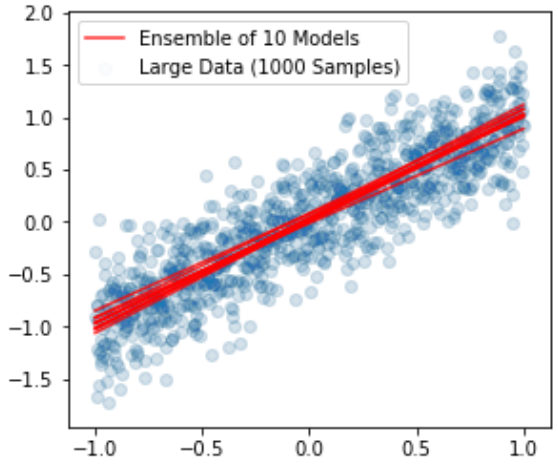
Source: Wikipedia

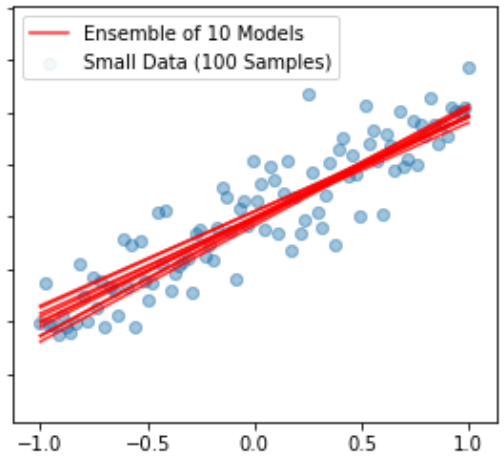
Uncertainty Quantification

- **Aleatoric Uncertainty**
  - Homoscedastic
  - Heteroscedastic
- **Epistemic Uncertainty**





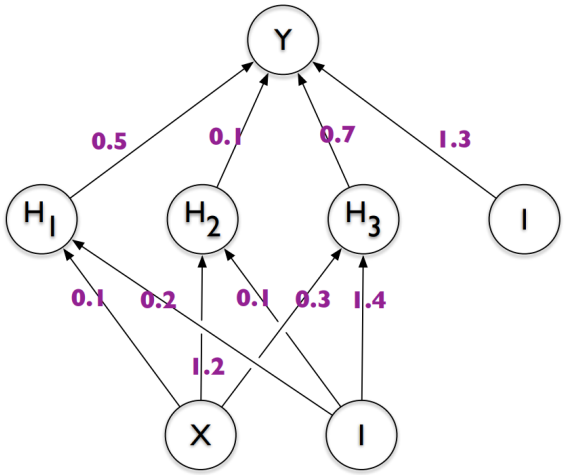




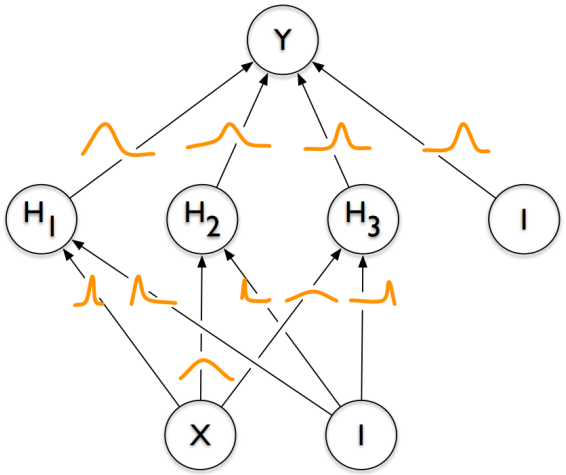
What is Probabilistic Deep Learning?

- **Standard (Point-Estimate) Neural Networks:**
  - Given input, predict output
  - Learn $\boldsymbol{\theta}$, where $\boldsymbol{w}, \boldsymbol{b} \in \boldsymbol{\theta}$
- **Probabilistic Neural Networks:**
  - Given input, predict likelihood of output
  - Learn PDF $\sim \boldsymbol{\theta}$, where $\boldsymbol{w}, \boldsymbol{b} \in \boldsymbol{\theta}$
- **Bayesian Neural Networks:**
  - Given input, predict output with certainty
  - Learn $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ where $w_i, b_i \sim \mathcal{N}(\mu_i, \sigma_i)$
- **Probabilistic Bayesian Neural Networks:**
  - PNN + BNN

Standard DNN

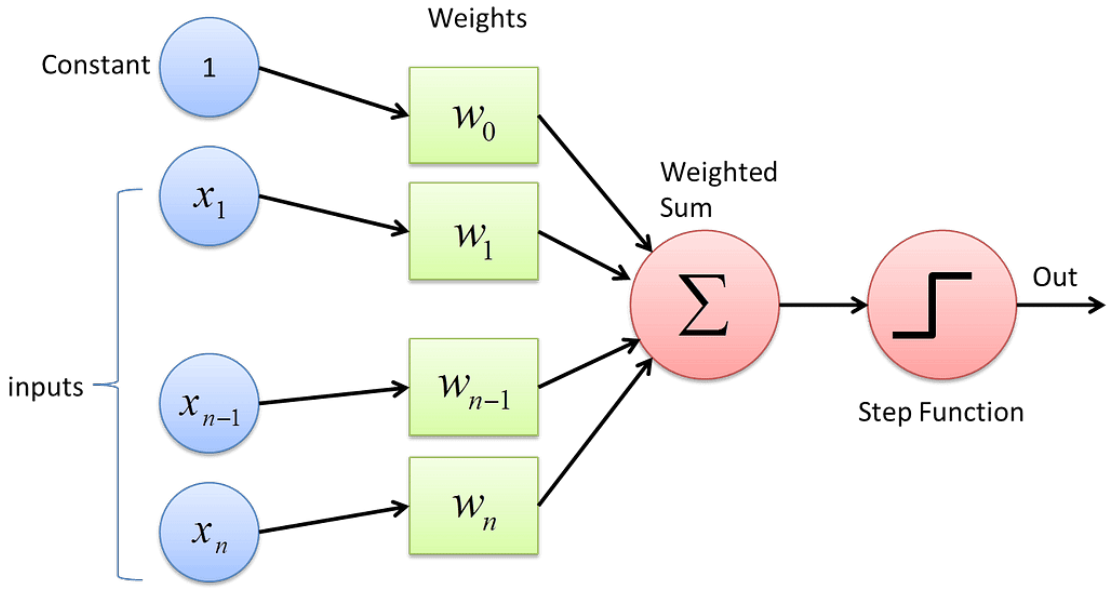


Bayesian DNN



Source: Charles, et al.

Mechanics of Bayesian Deep Learning



Source: Sharma, TDS

- **Notation:**

  Input: $\boldsymbol{x}$ Output: $\hat{\boldsymbol{y}}$ Weights: $\boldsymbol{w}$ Bias: b

  Data: $\boldsymbol{D} = (x_i, y_i)_{i=1}^{n}$ DNN Model: $F_\theta()$

  Activation Function: $g()$ Loss Function: $\mathcal{L}()$

- **Forward-Propagation:**

$$\hat{\boldsymbol{y}} = g(\boldsymbol{w}^T\boldsymbol{x} + b)$$
$$= g(w_1x_1 + w_2x_2 + w_3x_3 + b)$$

Neural Network	Bayesian Neural Network
$w_i, b_i \in \mathbb{R}$	$w_i, b_i \sim \mathcal{N}(\mu_i, \sigma_i)$
To learn $\{w_i, b_i\}$	To learn $\{\mu_i, \sigma_i\}$

Mechanics of Bayesian Deep Learning

Neural Network

- **Training:**

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{(x_i,y_i) \in D} \log[p(y_i|x_i,\theta)]$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{(x_i,y_i) \in D} \mathcal{L}(F_\theta(x_i), y_i)$$

- **Prediction:**

$$p(\hat{y} \mid \hat{x}, \theta^*)$$

$$\hat{y} = F_{\theta^*}(\hat{x})$$

Bayesian Neural Network

$$\mu^*, \Sigma^* = \underset{\mu,\Sigma}{\operatorname{argmax}} \sum_{(x_i,y_i) \in D} \log[p(y_i|x_i,\theta)] - KL[p(\theta), p(\theta_0)]$$

$$\theta \sim \mathcal{N}(\mu, \Sigma) \qquad \theta_0 \sim \mathcal{N}(0, \mathrm{I})$$

$$\mu^*, \Sigma^* = \underset{\mu,\Sigma}{\operatorname{argmin}} \sum_{(x_i,y_i) \in D} \mathcal{L}(F_\theta(x_i), y_i) + KL[p(\theta), p(\theta_0)]$$

$$p(\hat{y} \mid \hat{x}, D) = \int p(\hat{y} \mid \hat{x}, \theta^*)\, p(\theta^*|D)\, d\theta \qquad \theta^* \sim \mathcal{N}(\mu^*, \Sigma^*)$$

$$\hat{y} = \frac{1}{K} \sum_{k=1}^{K} F_{\theta_k^*}(\hat{x}) \qquad \theta_k^* \sim \mathcal{N}(\mu^*, \Sigma^*)$$

Mechanics of Bayesian Deep Learning

- **Back-Propagation:**
  - **Intractability:** must calculate derivatives of the parameters being learnt, i.e., derivatives of distributions

$$p(\hat{y} \mid \hat{x}, D) = \int p(\hat{y} \mid \hat{x}, \theta^*)\, p(\theta^* | D)\, d\theta$$

  - **Solution:** local re-parameterization trick -- "moves" the parameters to be learnt (μ, σ in case of a Gaussian distribution), out of the distribution function for any weight w.

w.k.t. $\theta = (\mu, \sigma)$ let $\epsilon \sim \mathcal{N}(0,1)$ $f(\epsilon) = w = \mu + \sigma \cdot \epsilon$ compute derivatives

- **Parameter Updates:**

$$\Delta\mu = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu} \qquad \Delta\sigma = \frac{\partial f}{\partial w}\frac{\epsilon}{\sigma} + \frac{\partial f}{\partial \sigma}$$

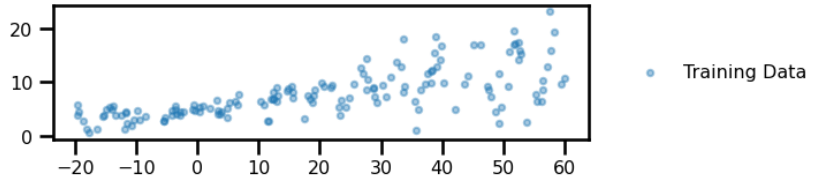
$$\mu := \mu - \alpha\Delta\mu \qquad \sigma := \sigma - \alpha\Delta\sigma$$

$$\theta^* = (\mu^*, \sigma^*)$$

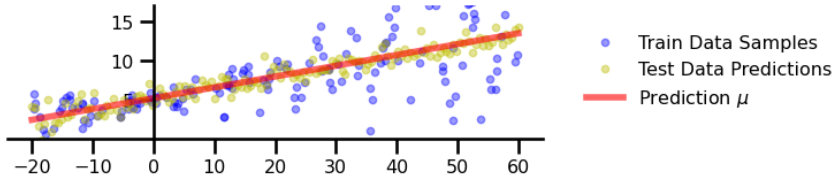
Pros and Cons of Bayesian Deep Learning

- **Pros**
  - They are more robust and generalizable
  - They can quantify the uncertainty in their predictive output
- **Cons**
  - They can be more complicated to train (may require knowledge of probability and statistics)
  - They can be slower to converge and require more data (since the weights of the network are distributions instead of single values)

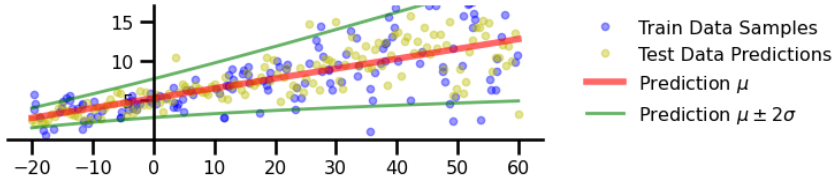
Preliminary Linear Regression Experiments



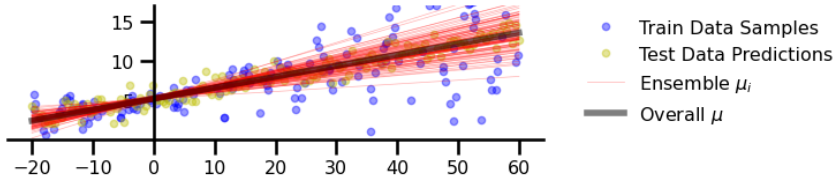
Data



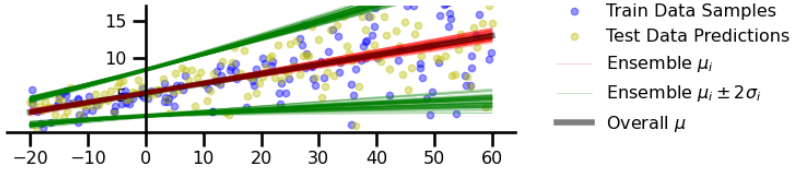
No Uncertainty



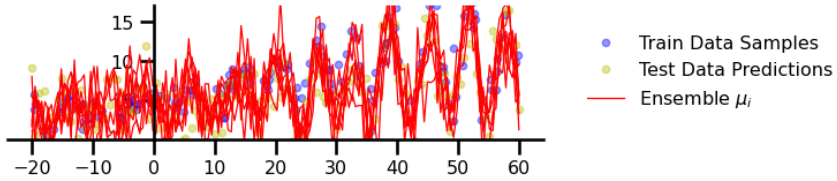
Aleatoric Uncertainty



Epistemic Uncertainty



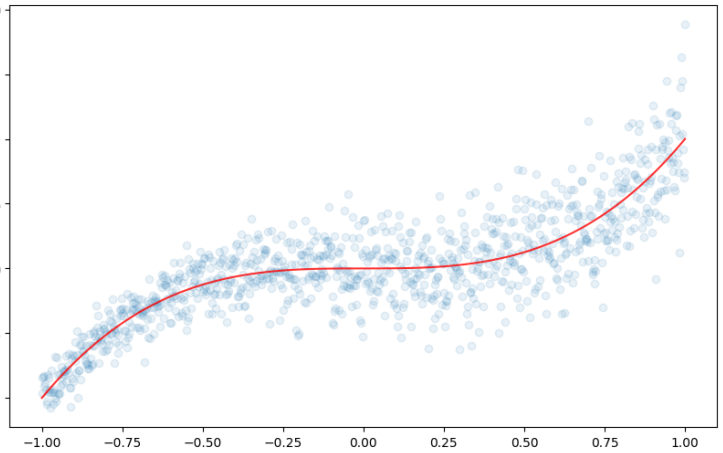
Aleatoric + Epistemic Uncertainties



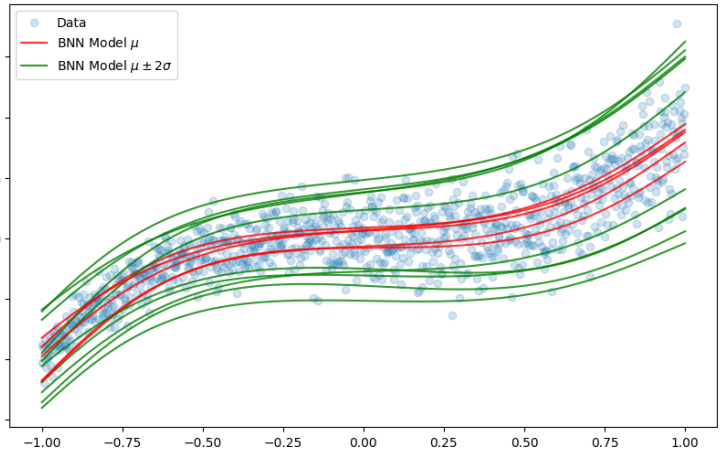
Functional Uncertainty

Preliminary Nonlinear Regression Experiments

Data

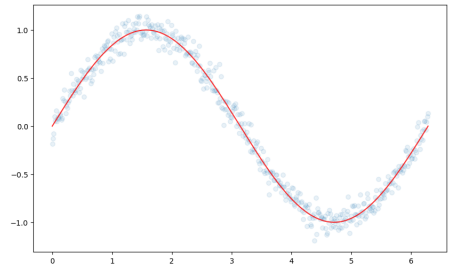


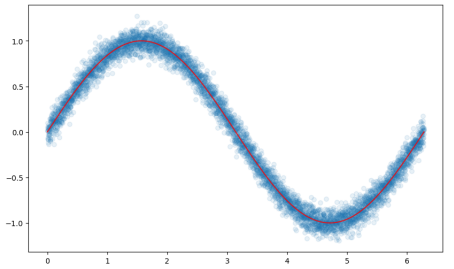
Model Predictions



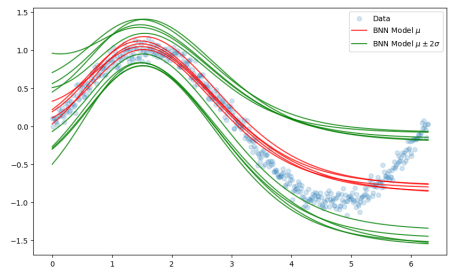
Cubic Data (1000 Samples)

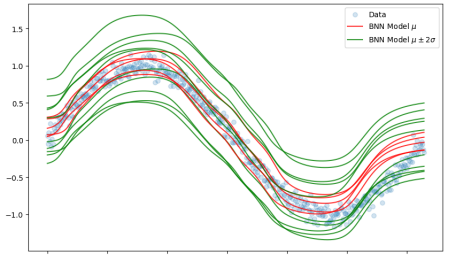
Data



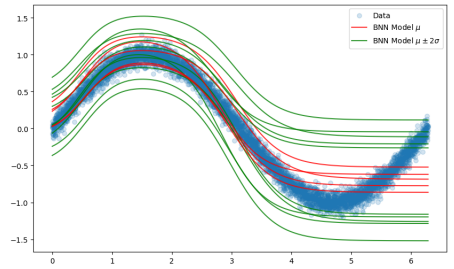


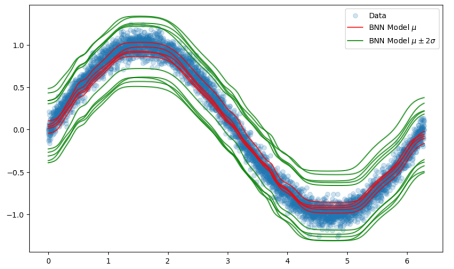
Pure Model





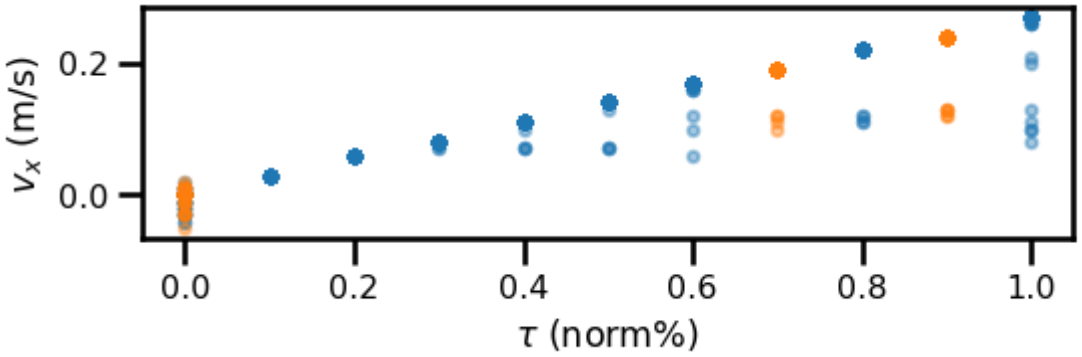
Hybrid Model



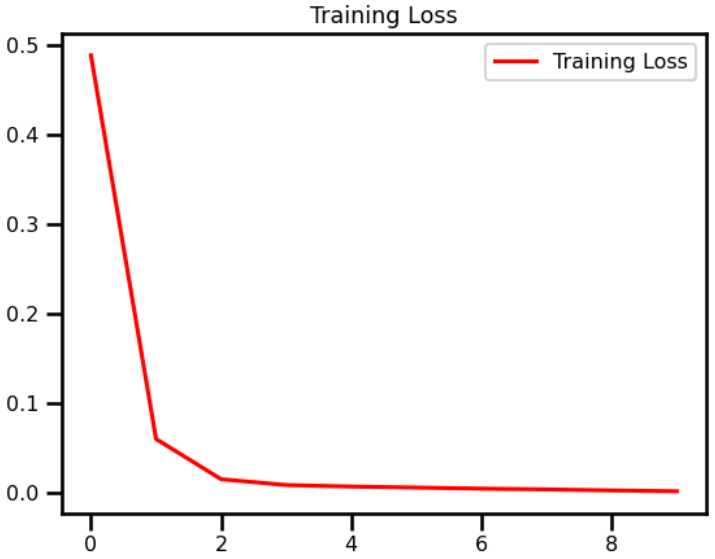


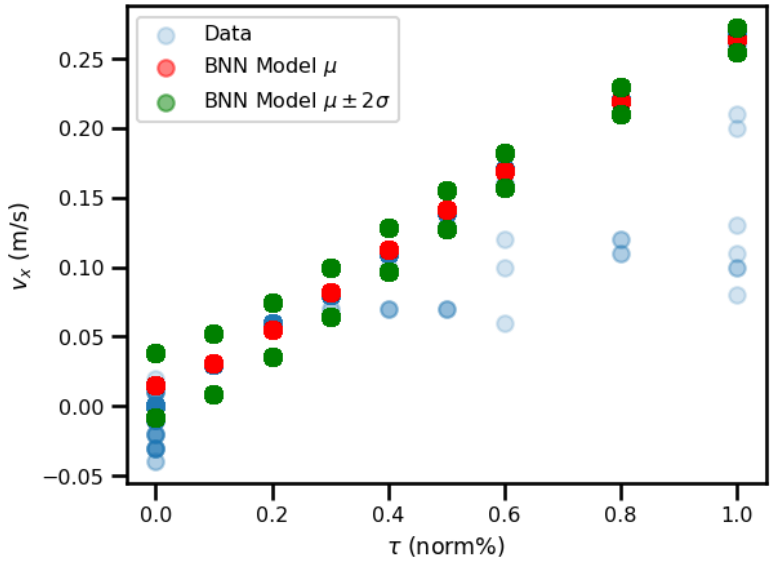
Sinusoidal Data (500 Samples) Sinusoidal Data (5000 Samples)

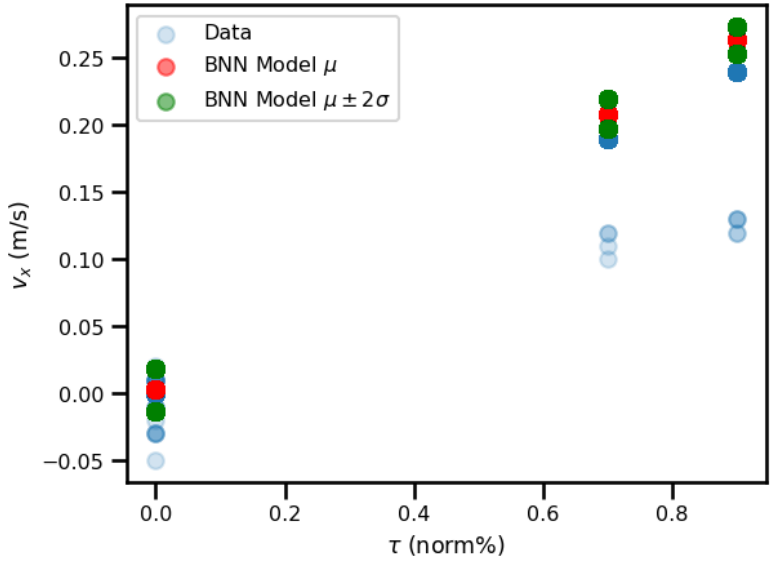
Preliminary 1D VD Experiments



Training Data
Testing Data



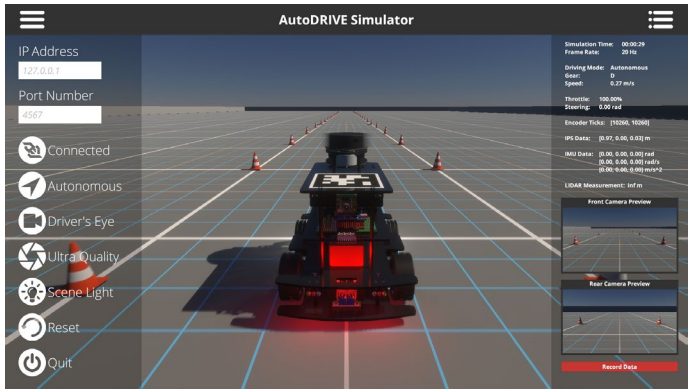


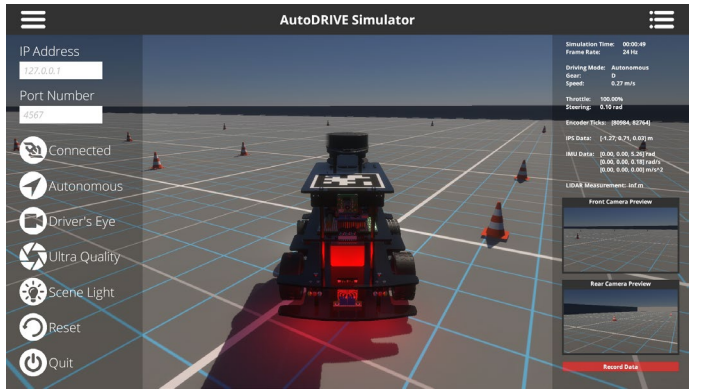


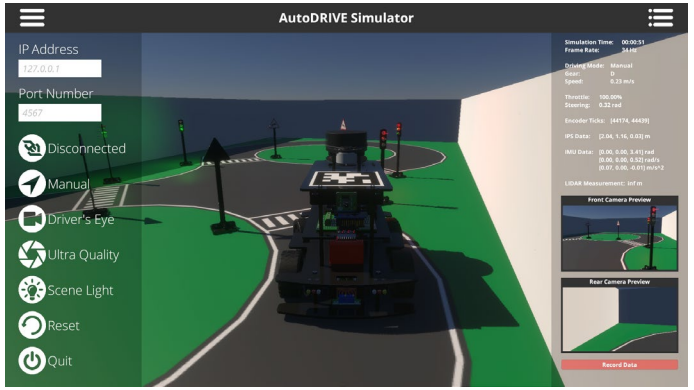
Data Collection

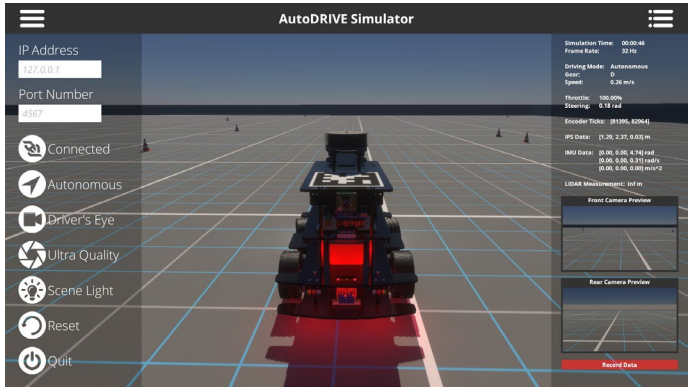
- AutoDRIVE Simulator
- Nigel (1:14) Vehicle
- Dynamics + perception
- Automated script

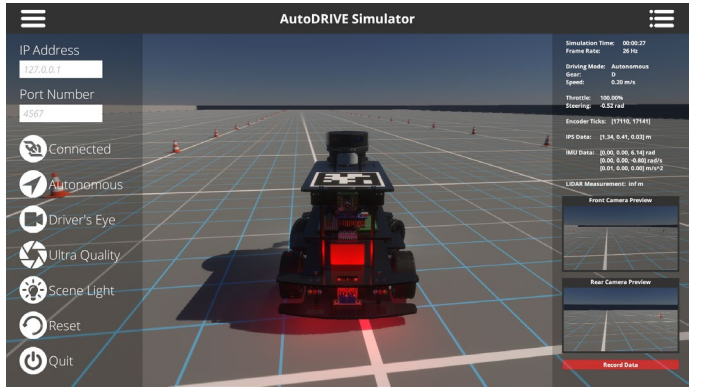
AutoDRIVE











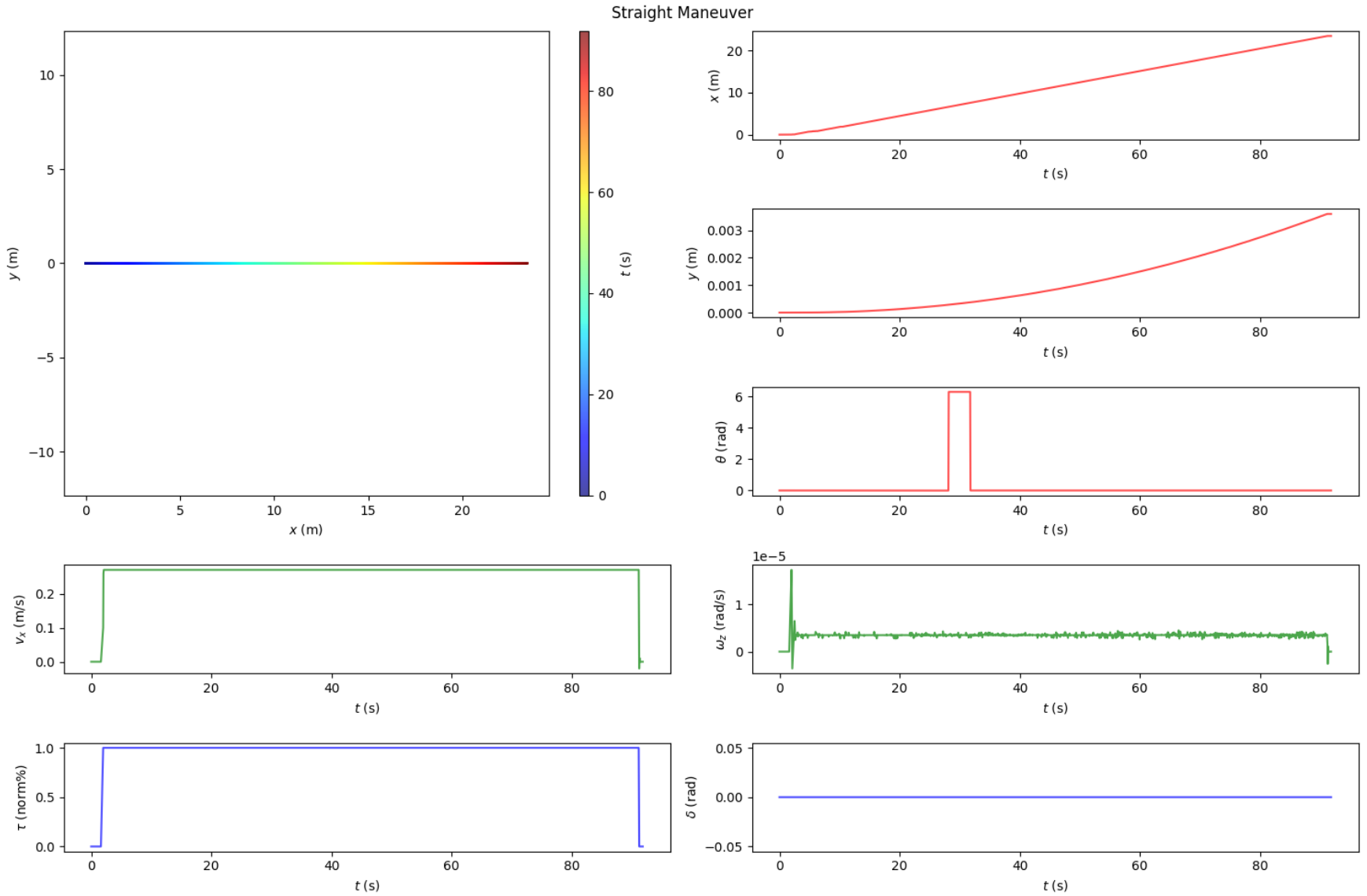
DATA	timestamp	throttle	steering	leftTicks	rightTicks	posX	posY	posZ	roll	pitch	yaw	speed	angX	angY	angZ	accX	accY	accZ	cam0	cam1
UNIT	yyyy_MM_dd_HH_mm_ss_fff	norm%	rad	count	count	m	m	m	rad	rad	rad	m/s	rad/s	rad/s	rad/s	m/s^2	m/s^2	m/s^2	img_path	img_path

DATA	timestamp	state
UNIT	yyyy_MM_dd_HH_mm_ss_fff	Int{0,1,2,3}

Dataset: https://github.com/Tinker-Twins/AutoDRIVE-Nigel-Dataset

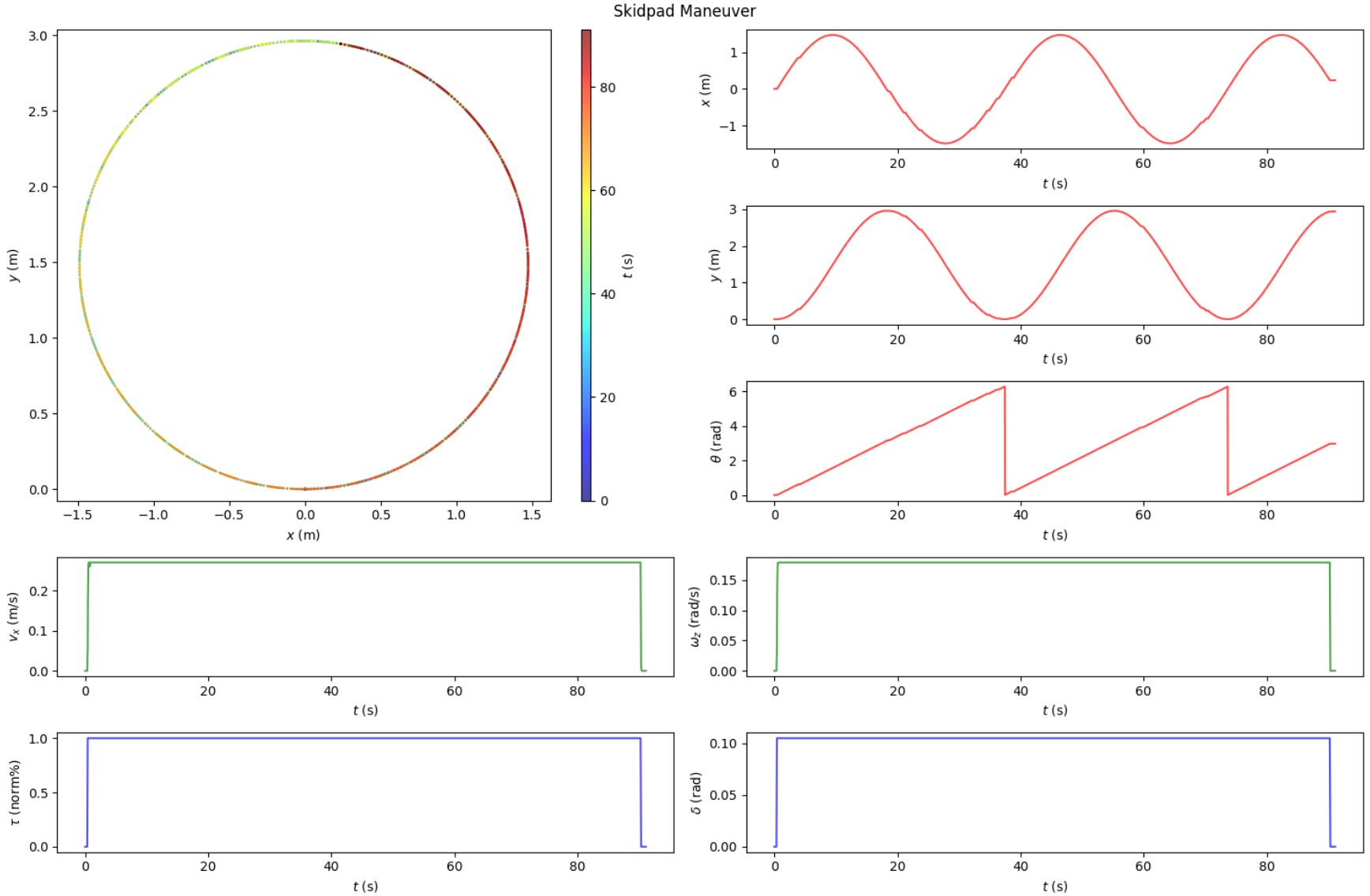
Data Collection

- **Straight**
- Skidpad
- Fishhook
- Slalom
- Eight



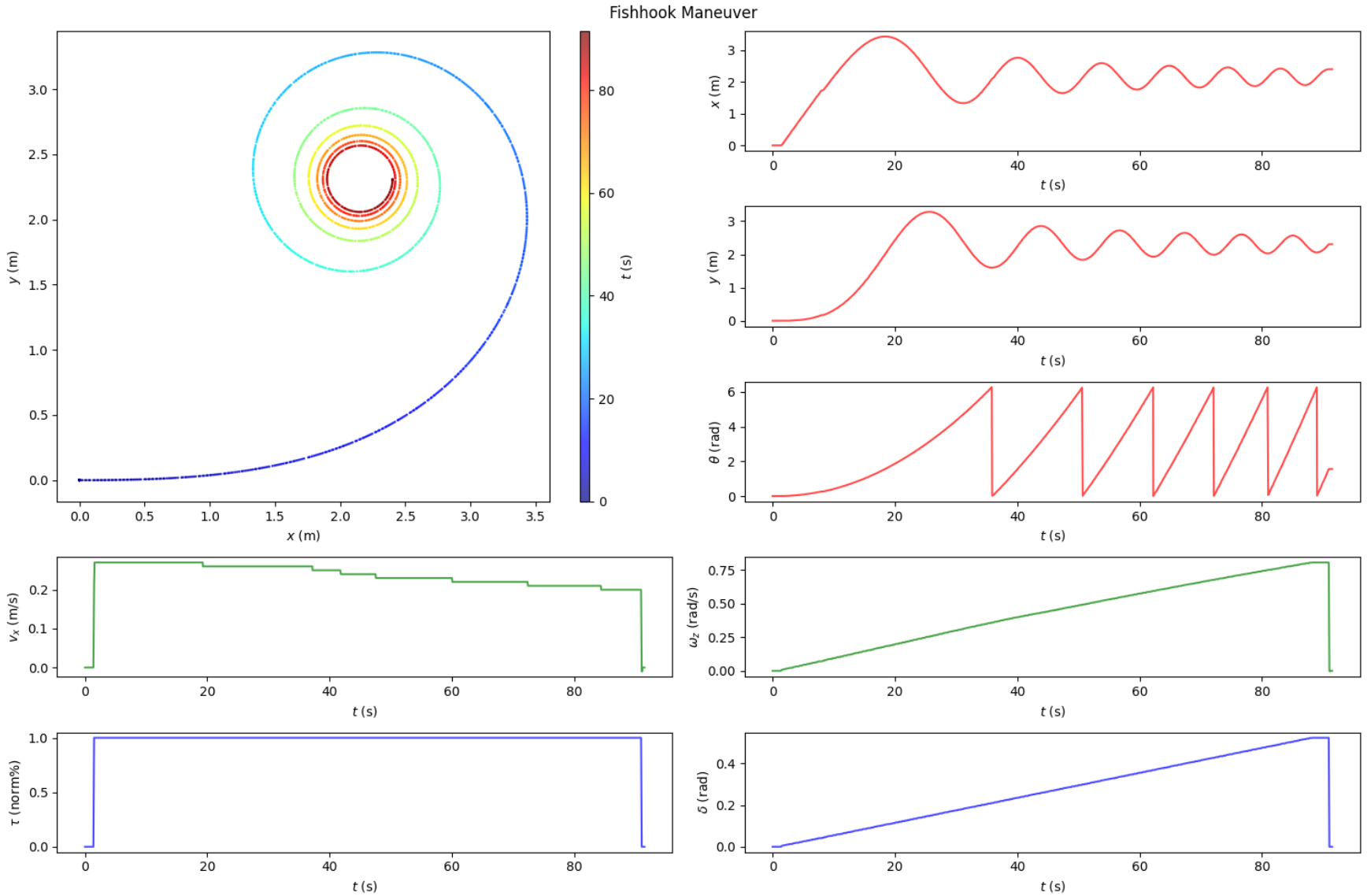
Data Collection

- Straight
- **Skidpad**
- Fishhook
- Slalom
- Eight



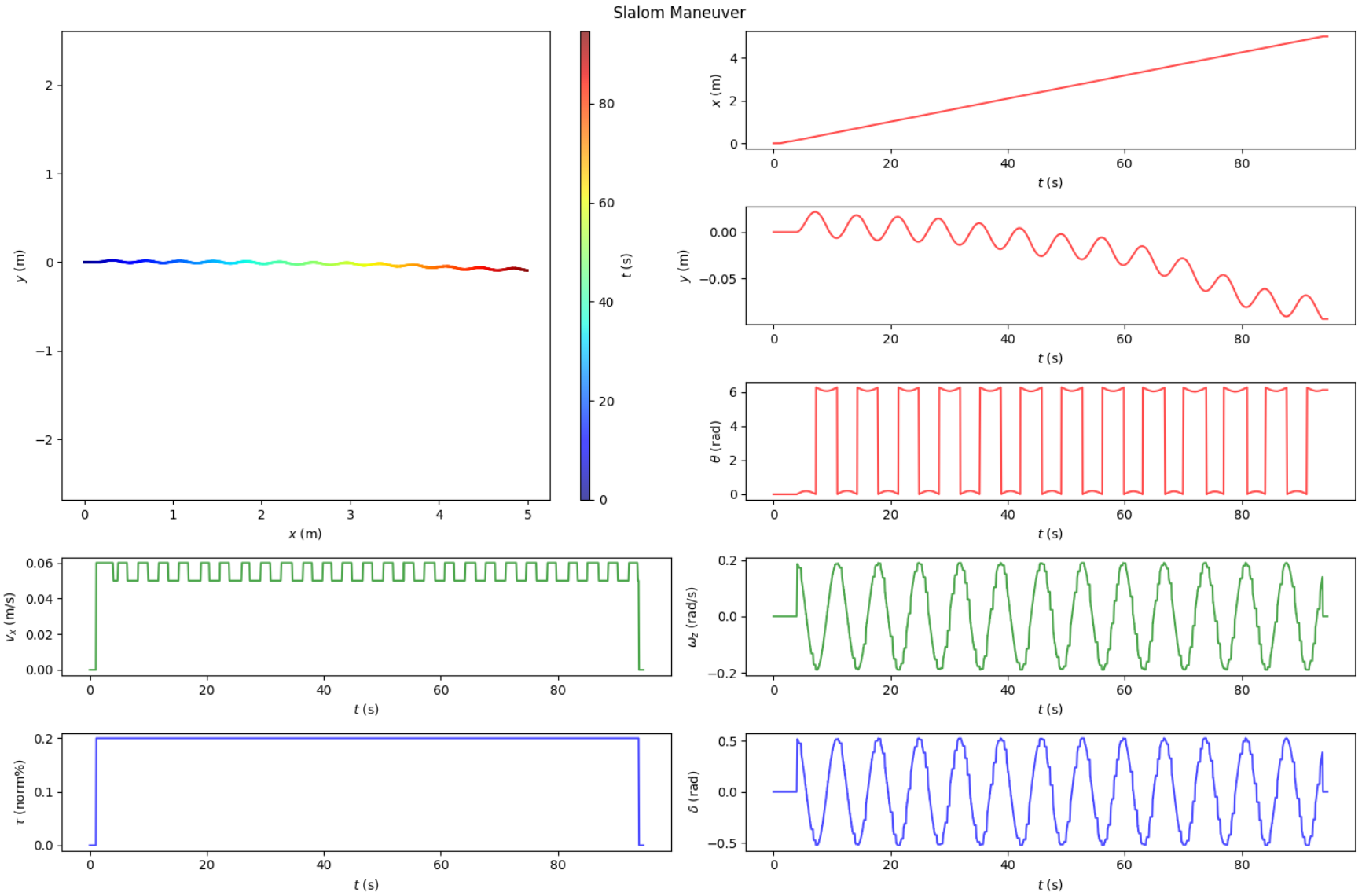
Data Collection

- Straight
- Skidpad
- **Fishhook**
- Slalom
- Eight



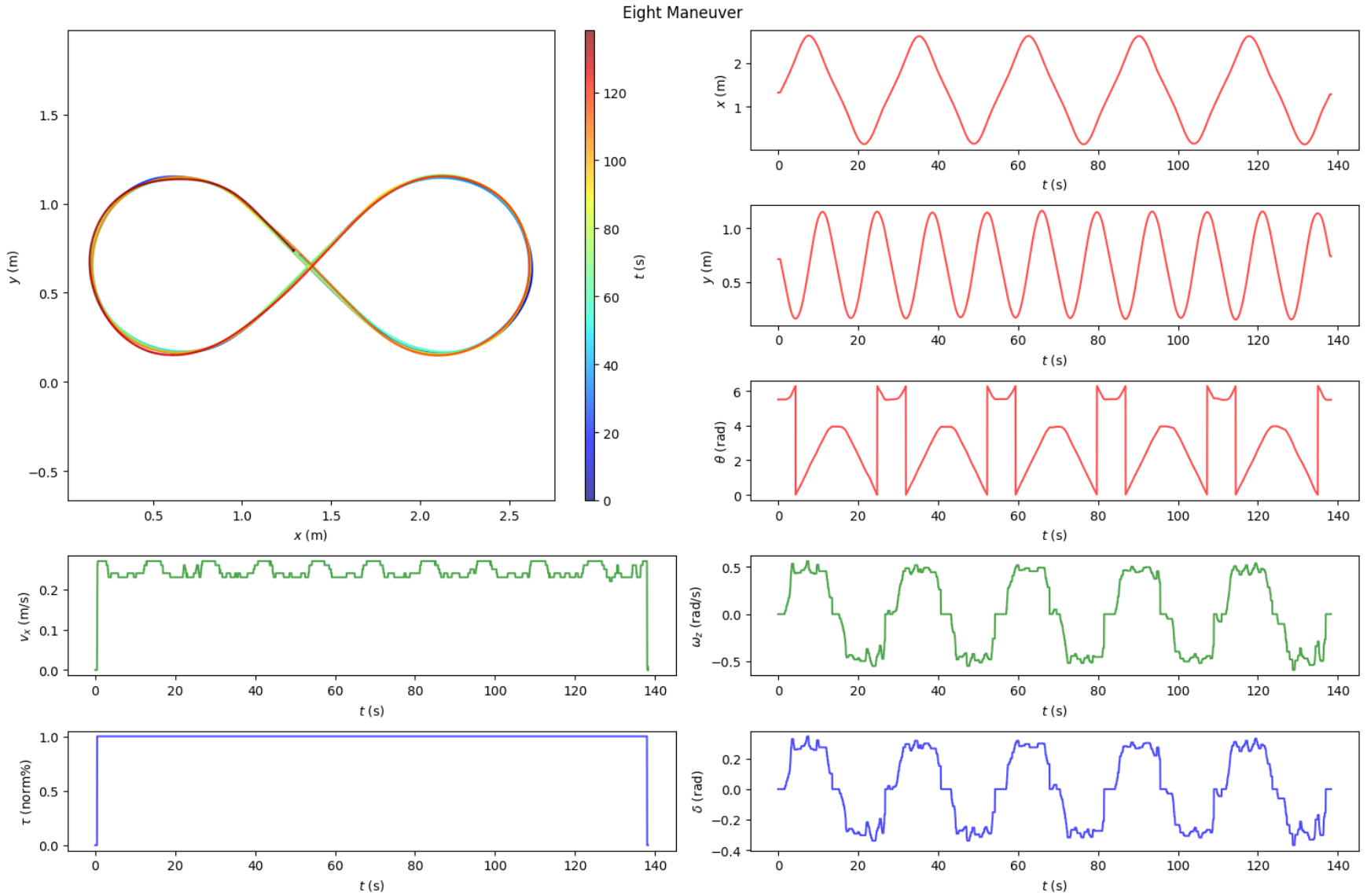
Data Collection

- Straight
- Skidpad
- Fishhook
- **Slalom**
- Eight



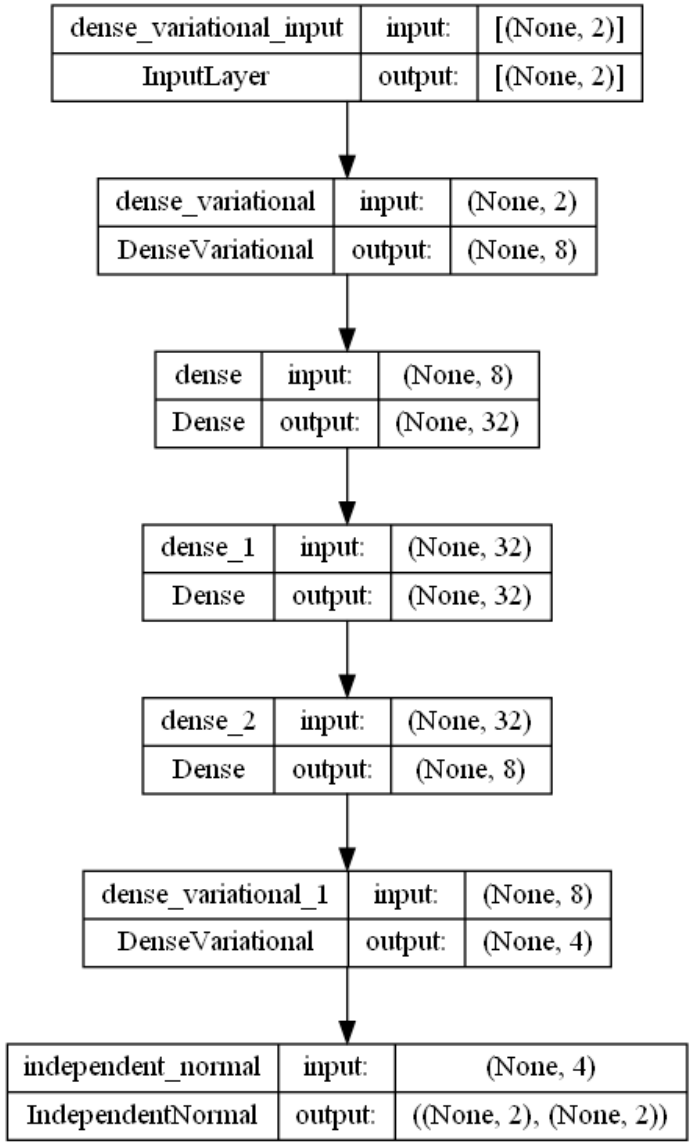
Data Collection

- Straight
- Skidpad
- Fishhook
- Slalom
- **Eight**



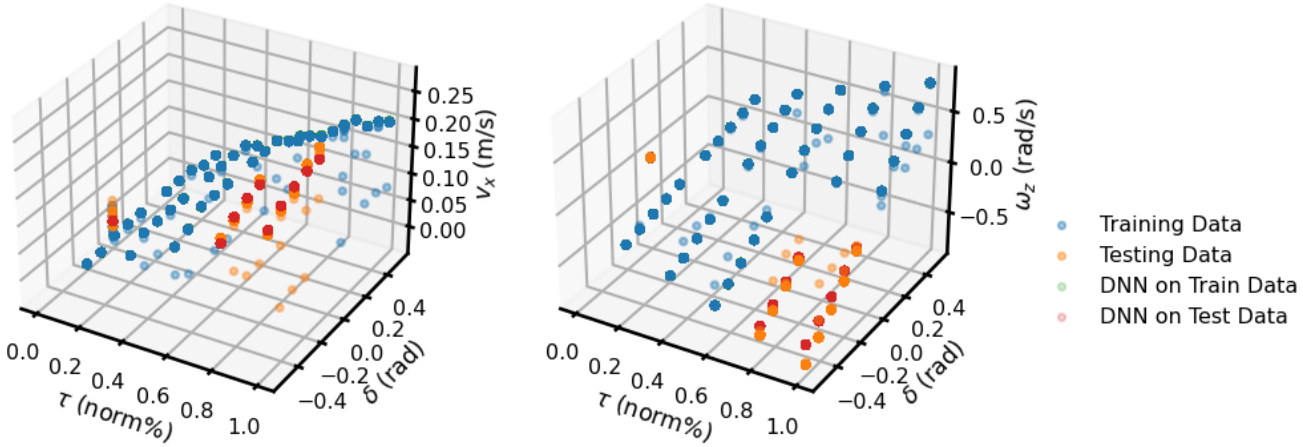
Hyperparameter Tuning

- **# epochs:** {5, 10, 15, 20, 25, 30}
- **# hidden layers:** {3d, 2v, 3v, 5v, 3d2v}
- **Learning rate:** {5e-3, 1e-3, 1e-2, 1e-1}
- **Activation functions:**
  - **Bounded:** {tanh, sigmoid}
  - **Unbounded:** {elu, relu, selu, softplus}
- **Optimizer:** {SGD, RMSprop, Adam}
- **Loss function:** {NLL, MSE, MAE}
- **Prior distribution:** {univariate Gaussian, multivariate Gaussian}
- **Posterior distribution:** {univariate Gaussian, multivariate Gaussian}
- **KL-divergence:** {approx, exact}



Hyperparameter Tuning

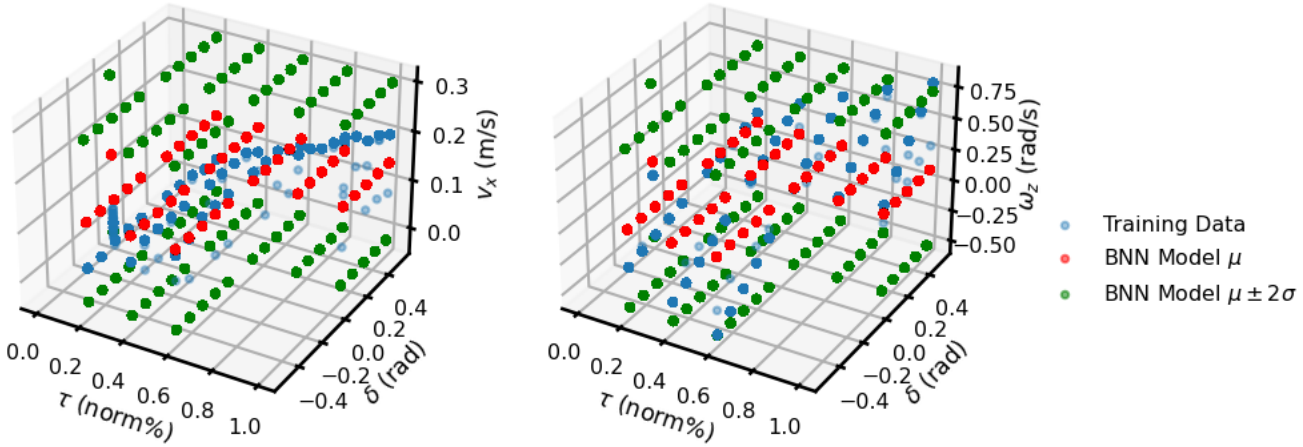
- **# epochs:** {5, 10, 15, 20, 25, 30}
- **# hidden layers:** {3d, 2v, 3v, 5v, 3d2v}
- **Learning rate:** {5e-3, 1e-3, 1e-2, 1e-1}
- **Activation functions:**
  - **Bounded:** {tanh, sigmoid}
  - **Unbounded:** {elu, relu, selu, softplus}
- **Optimizer:** {SGD, RMSprop, Adam}
- **Loss function:** {NLL, MSE, MAE}
- **Prior distribution:** {univariate Gaussian, multivariate Gaussian}
- **Posterior distribution:** {univariate Gaussian, multivariate Gaussian}
- **KL-divergence:** {approx, exact}



Skidpad {5, 1e-3, 3d, relu, Adam, MSE}

Hyperparameter Tuning

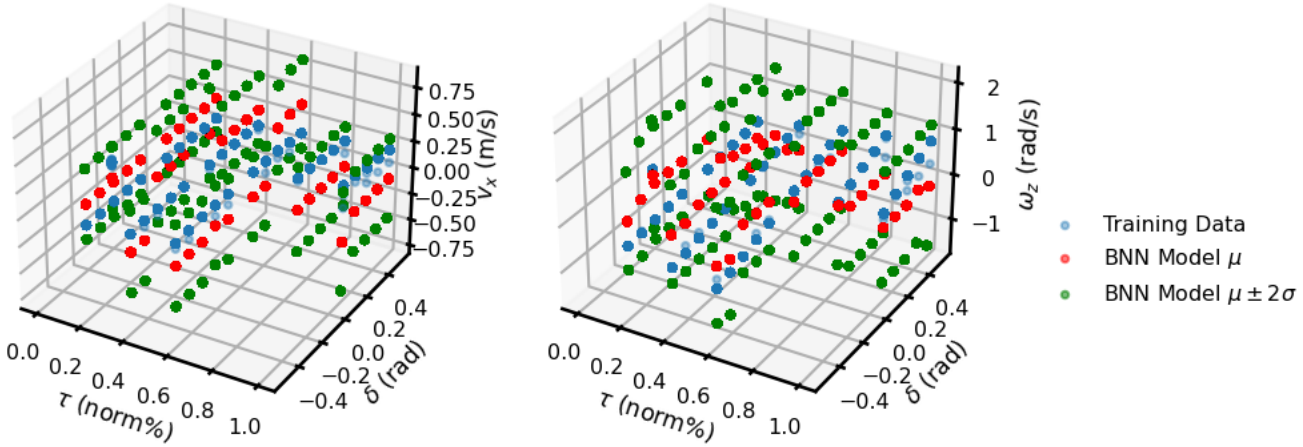
- **# epochs:** {5, 10, 15, 20, 25, 30}
- **# hidden layers:** {3d, 2v, 3v, 5v, 3d2v}
- **Learning rate:** {5e-3, 1e-3, 1e-2, 1e-1}
- **Activation functions:**
  - **Bounded:** {tanh, sigmoid}
  - **Unbounded:** {elu, relu, selu, softplus}
- **Optimizer:** {SGD, RMSprop, Adam}
- **Loss function:** {NLL, MSE, MAE}
- **Prior distribution:** {univariate Gaussian, multivariate Gaussian}
- **Posterior distribution:** {univariate Gaussian, multivariate Gaussian}
- **KL-divergence:** {approx, exact}



Skidpad {10, 1e-3, 3v, sigmoid, Adam, NLL, MVG, approx KLD}

Hyperparameter Tuning

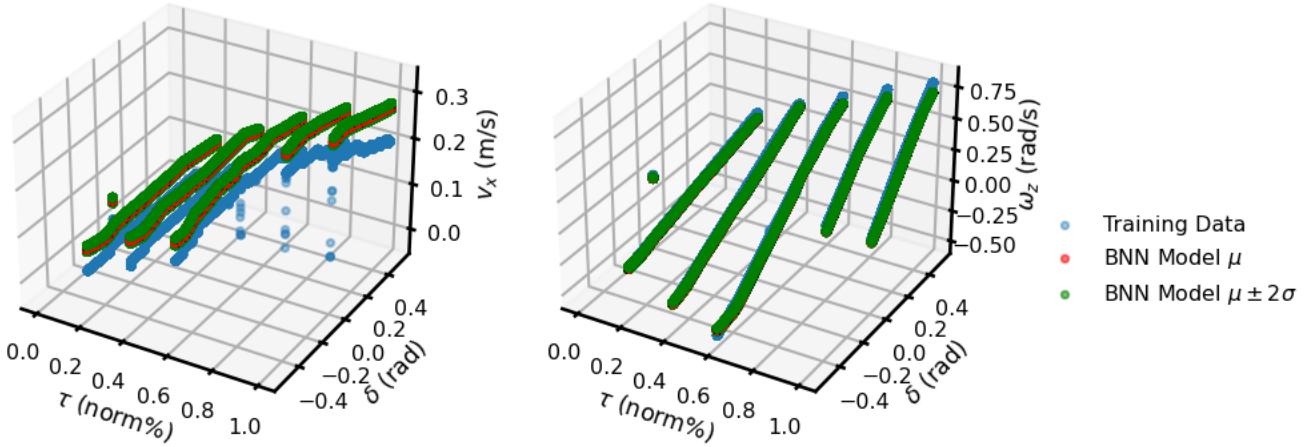
- **# epochs:** {5, 10, 15, 20, 25, 30}
- **# hidden layers:** {3d, 2v, 3v, 5v, 3d2v}
- **Learning rate:** {5e-3, 1e-3, 1e-2, 1e-1}
- **Activation functions:**
  - **Bounded:** {tanh, sigmoid}
  - **Unbounded:** {elu, relu, selu, softplus}
- **Optimizer:** {SGD, RMSprop, Adam}
- **Loss function:** {NLL, MSE, MAE}
- **Prior distribution:** {univariate Gaussian, multivariate Gaussian}
- **Posterior distribution:** {univariate Gaussian, multivariate Gaussian}
- **KL-divergence:** {approx, exact}



Skidpad {25, 5e-3, 2v, tanh, RMSProp, NLL, MVG, approx KLD}

Hyperparameter Tuning

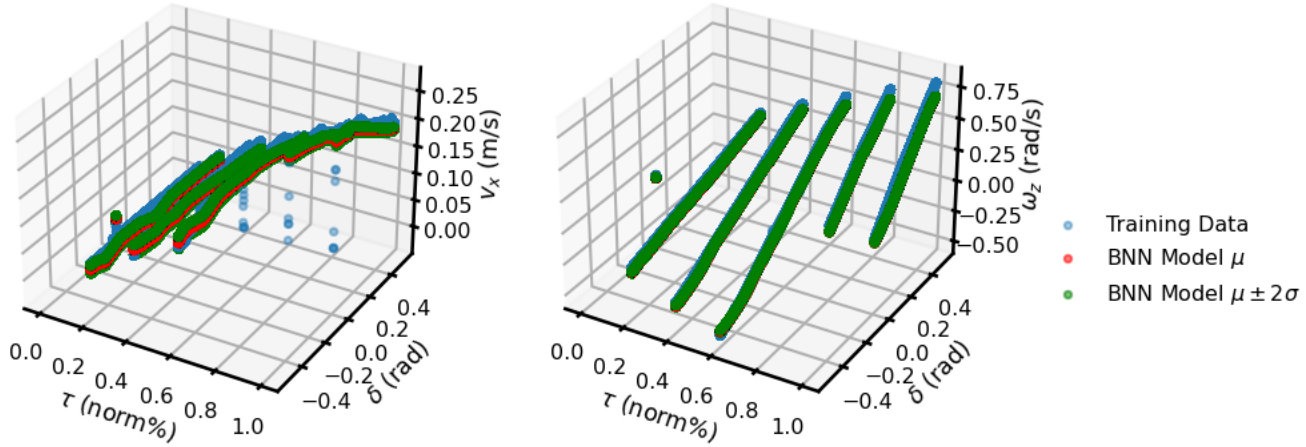
- **# epochs:** {5, 10, 15, 20, 25, 30}
- **# hidden layers:** {3d, 2v, 3v, 5v, 3d2v}
- **Learning rate:** {5e-3, 1e-3, 1e-2, 1e-1}
- **Activation functions:**
  - **Bounded:** {tanh, sigmoid}
  - **Unbounded:** {elu, relu, selu, softplus}
- **Optimizer:** {SGD, RMSprop, Adam}
- **Loss function:** {NLL, MSE, MAE}
- **Prior distribution:** {univariate Gaussian, multivariate Gaussian}
- **Posterior distribution:** {univariate Gaussian, multivariate Gaussian}
- **KL-divergence:** {approx, exact}



Fishhook {25, 1e-3, 3d2v, sigmoid, RMSProp, MAE, MVG, approx KLD}

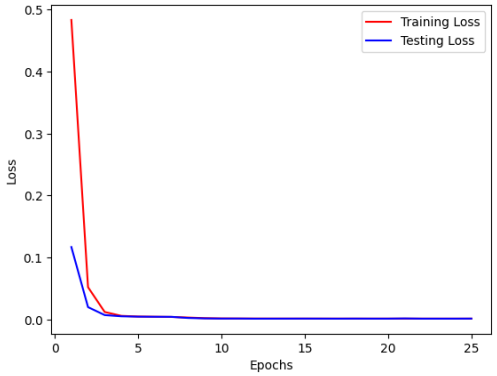
Hyperparameter Tuning

- **# epochs:** {5, 10, 15, 20, 25, 30}
- **# hidden layers:** {3d, 2v, 3v, 5v, 3d2v}
- **Learning rate:** {5e-3, 1e-3, 1e-2, 1e-1}
- **Activation functions:**
  - **Bounded:** {tanh, sigmoid}
  - **Unbounded:** {elu, relu, selu, softplus}
- **Optimizer:** {SGD, RMSprop, Adam}
- **Loss function:** {NLL, MSE, MAE}
- **Prior distribution:** {univariate Gaussian, multivariate Gaussian}
- **Posterior distribution:** {univariate Gaussian, multivariate Gaussian}
- **KL-divergence:** {approx, exact}

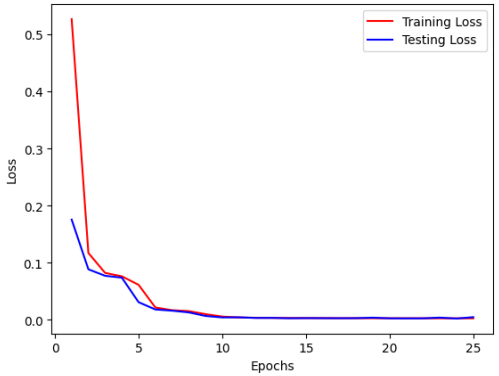


Fishhook {25, 1e-3, 3d2v, sigmoid, Adam, MAE, MVG, exact KLD}

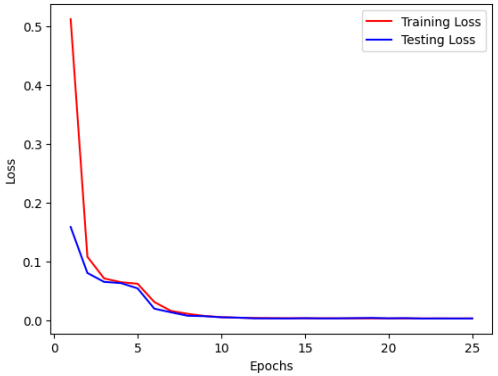
Model Training



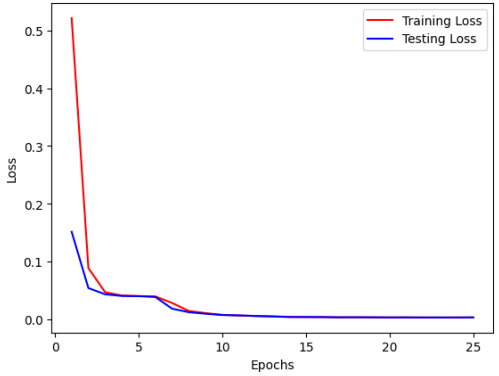
Straight Maneuver



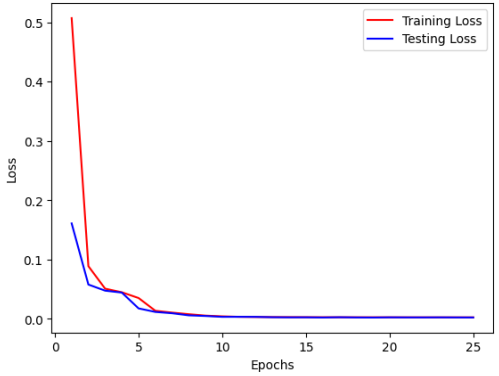
Skidpad Maneuver



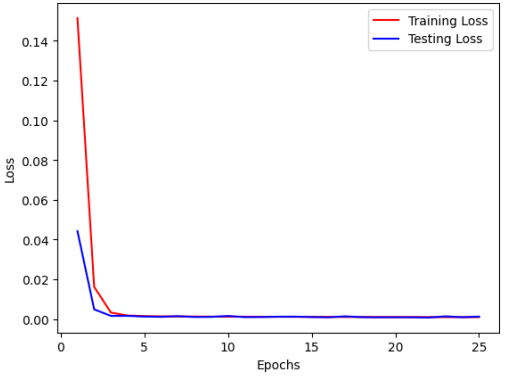
Fishhook Maneuver



Slalom Maneuver

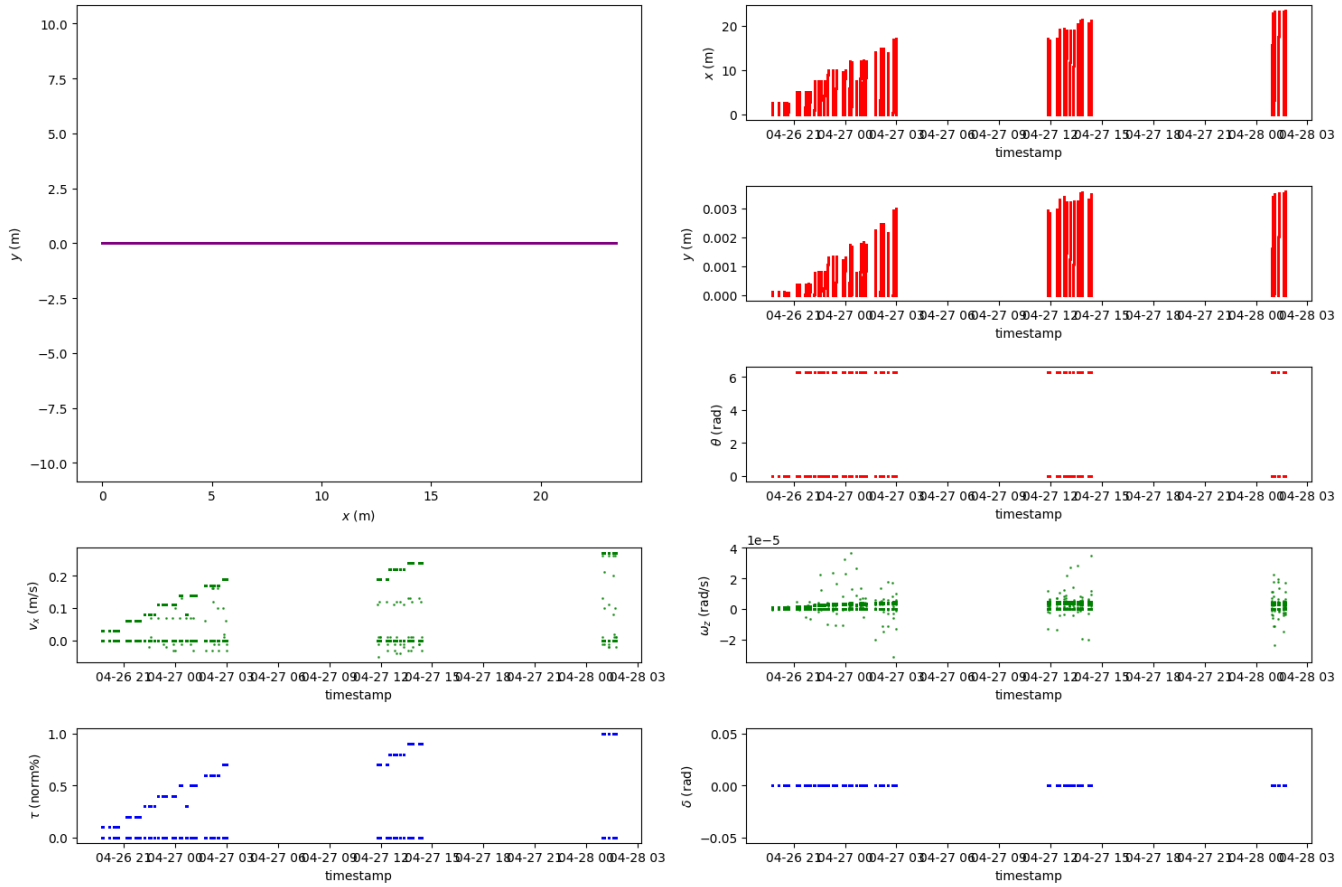


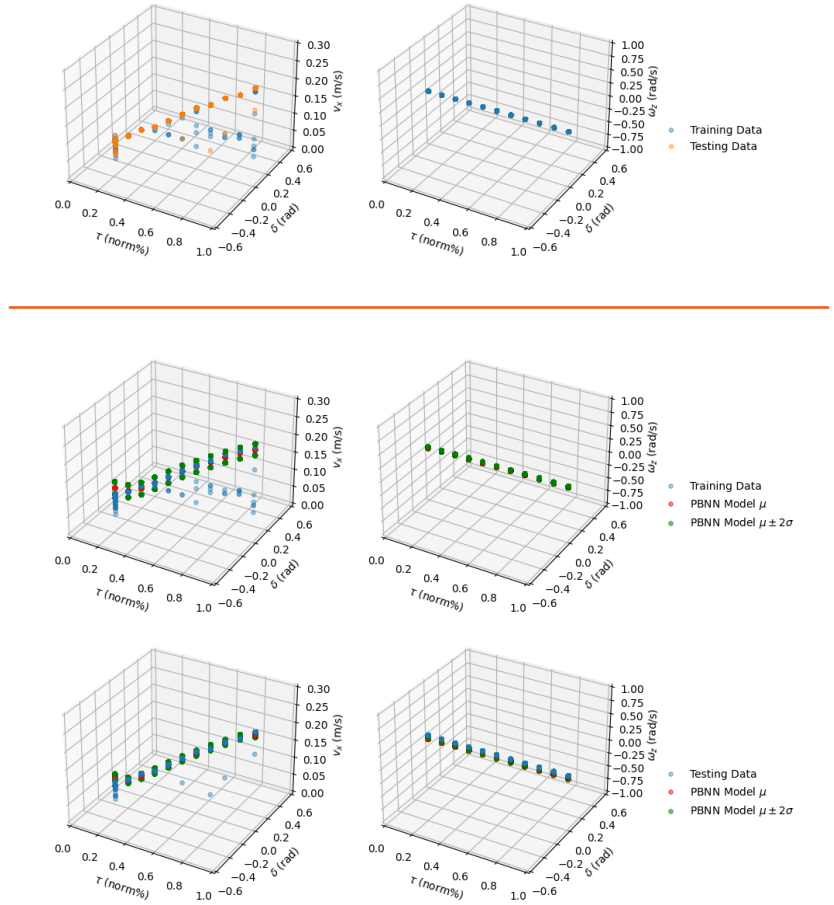
Eight Maneuver



All Maneuvers

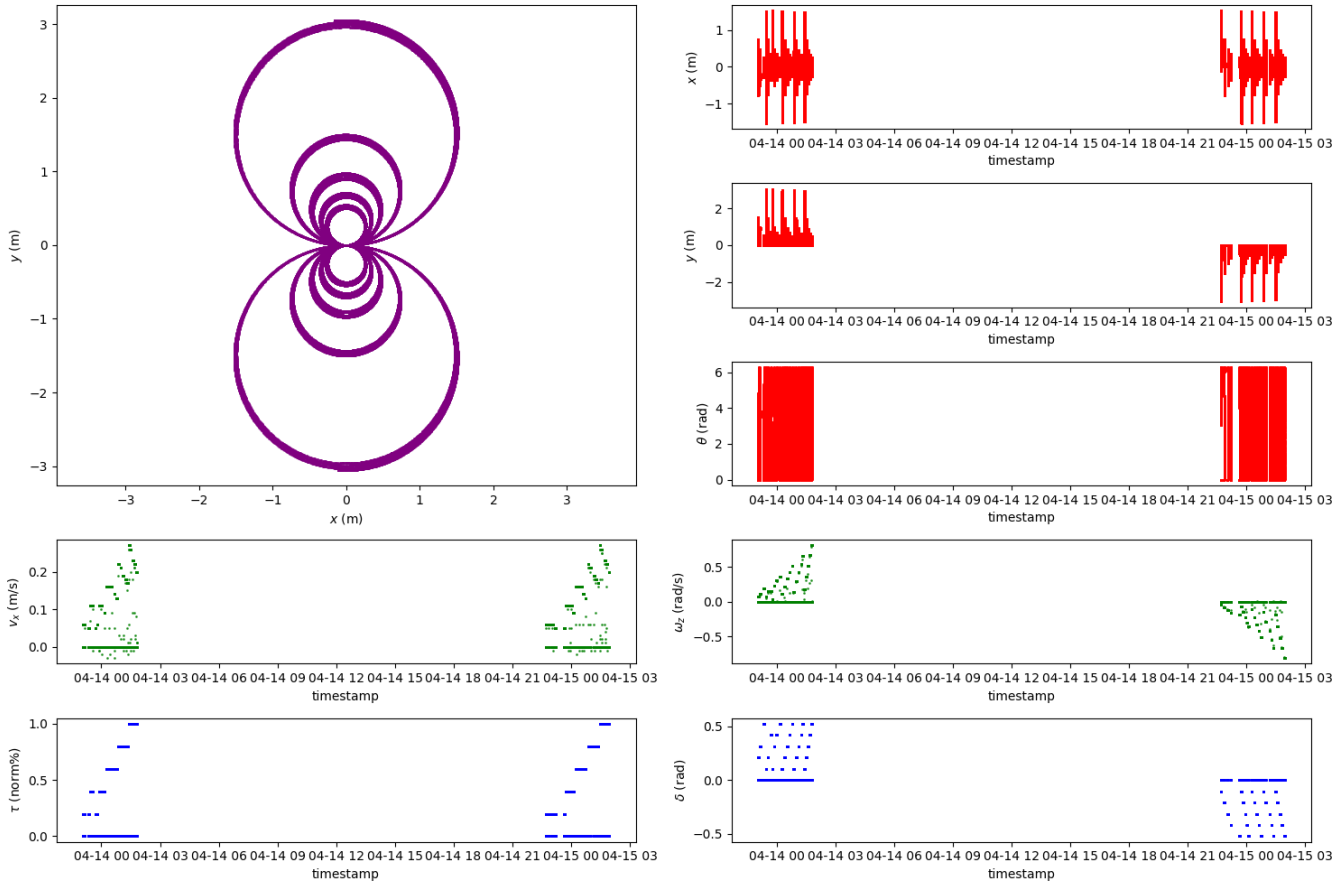
Model Inference – Straight Maneuver

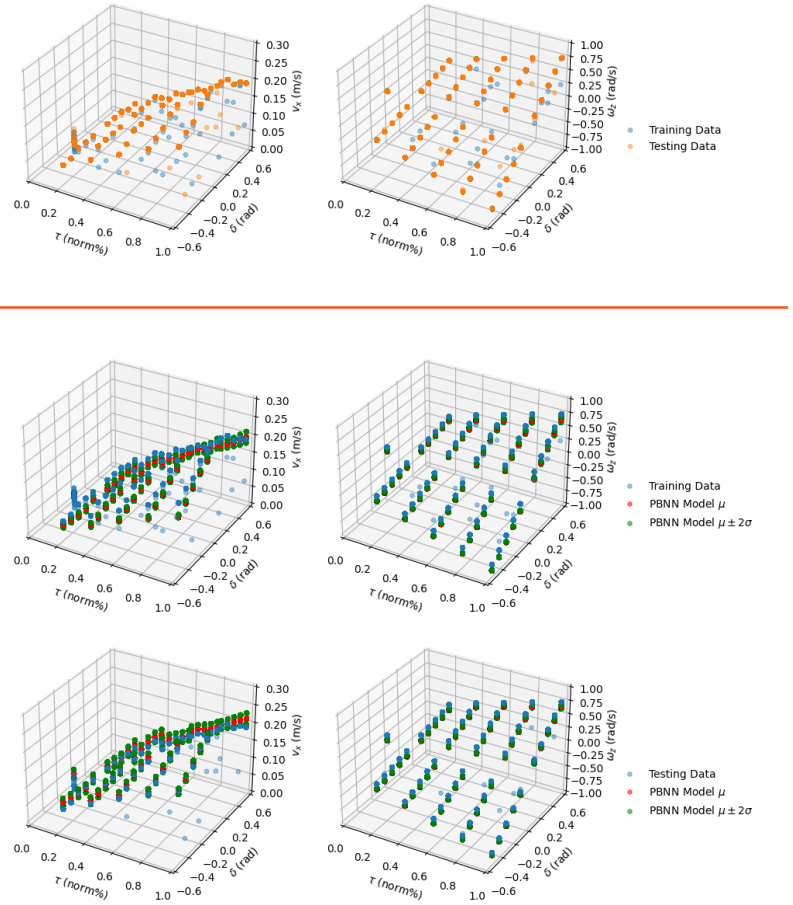




***Note**: This work adheres to Markov assumption.*

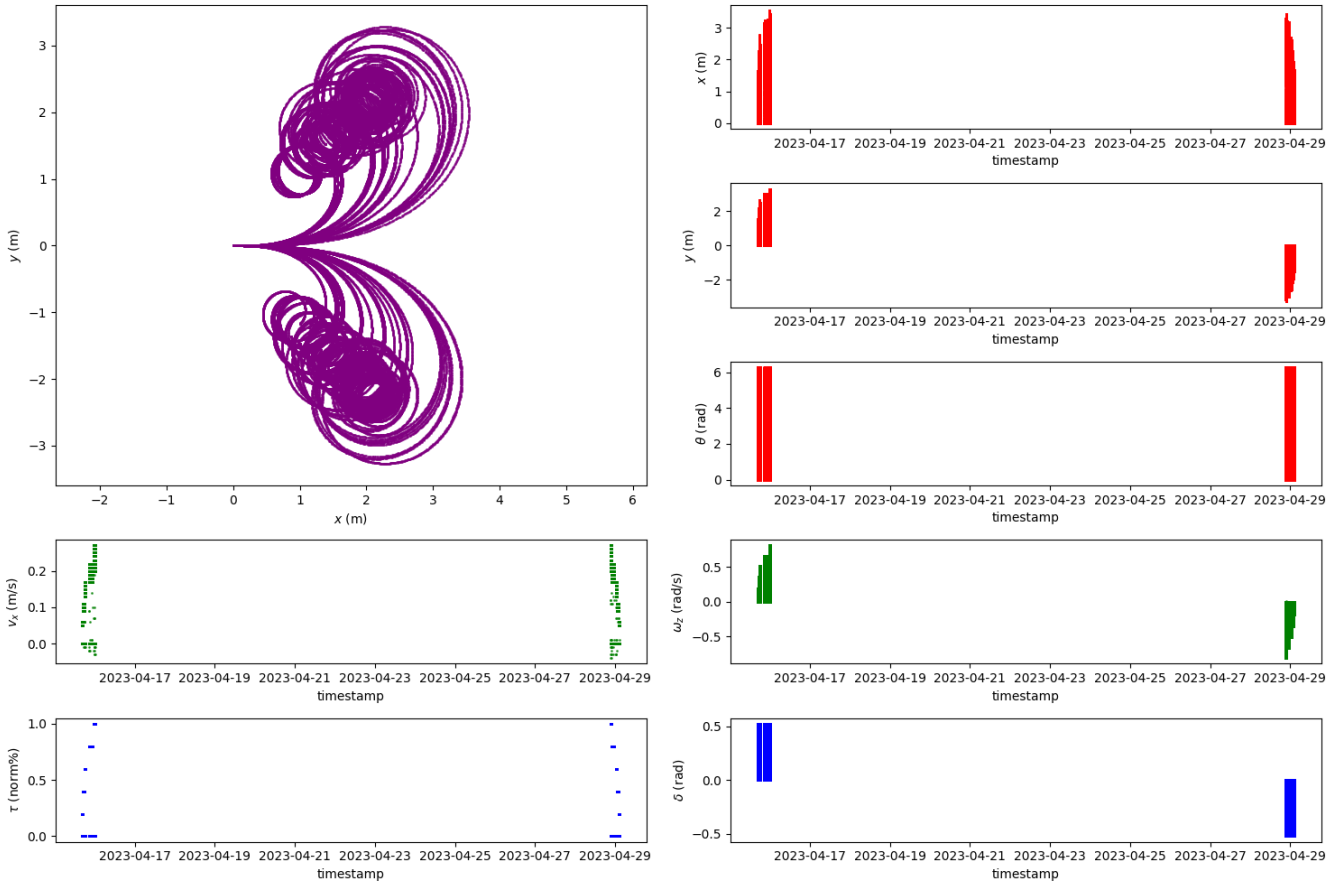
Model Inference – Skidpad Maneuver

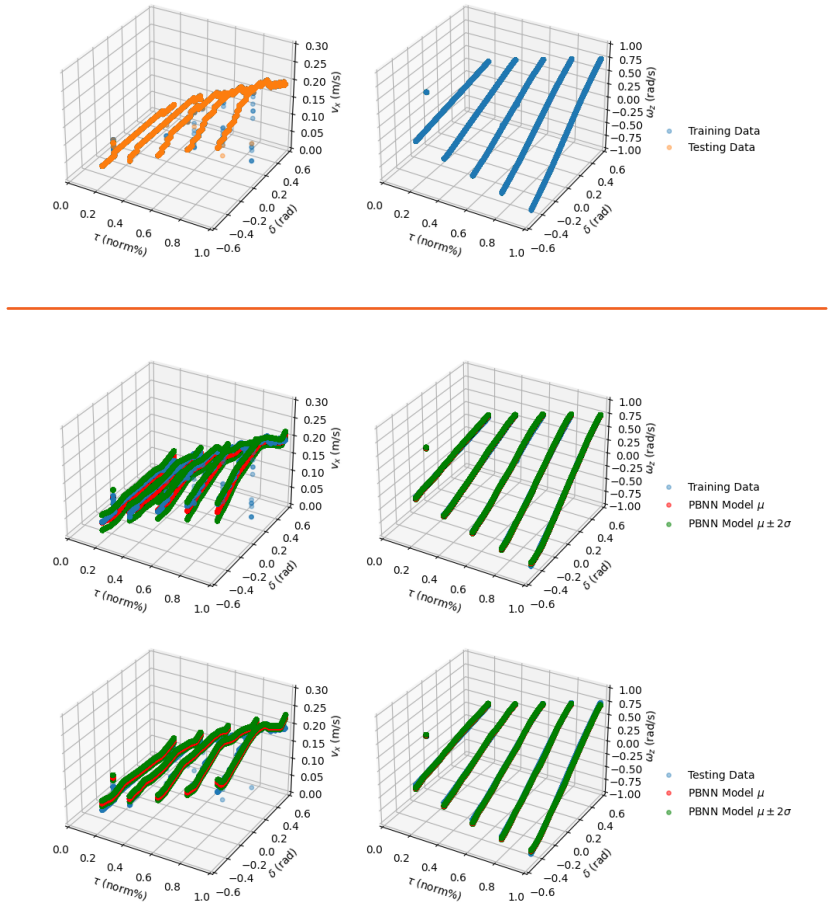




***Note**: This work adheres to Markov assumption.*

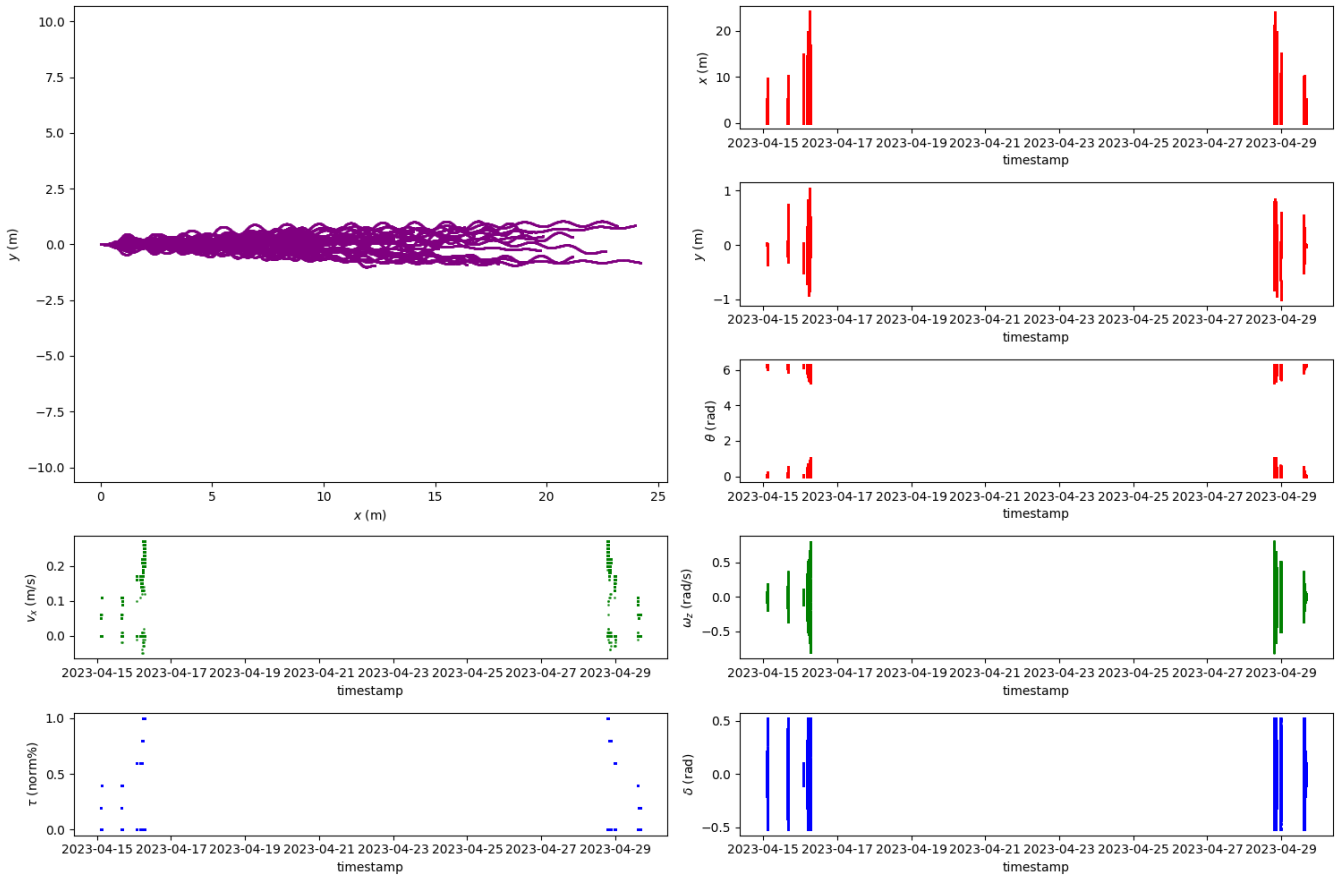
Model Inference – Fishhook Maneuver

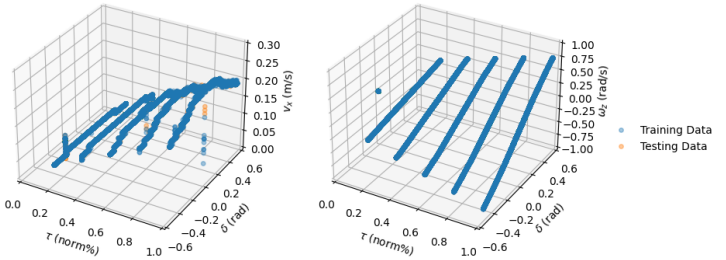


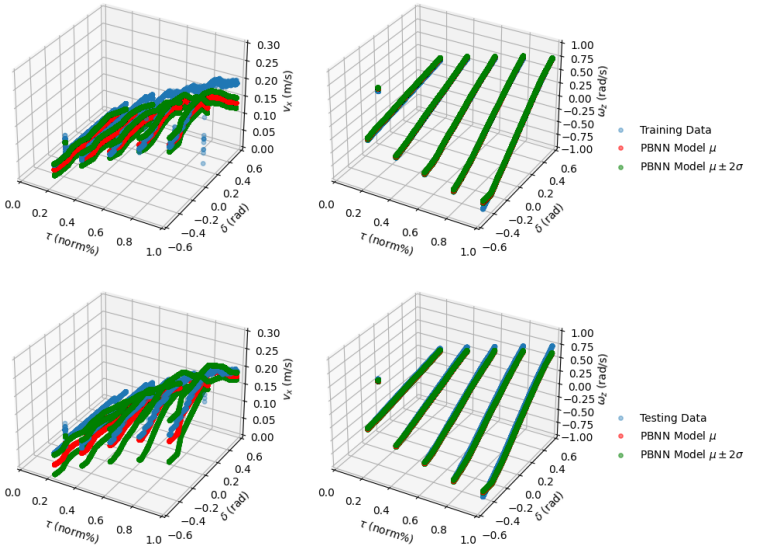


***Note**: This work adheres to Markov assumption.*

Model Inference – Slalom Maneuver

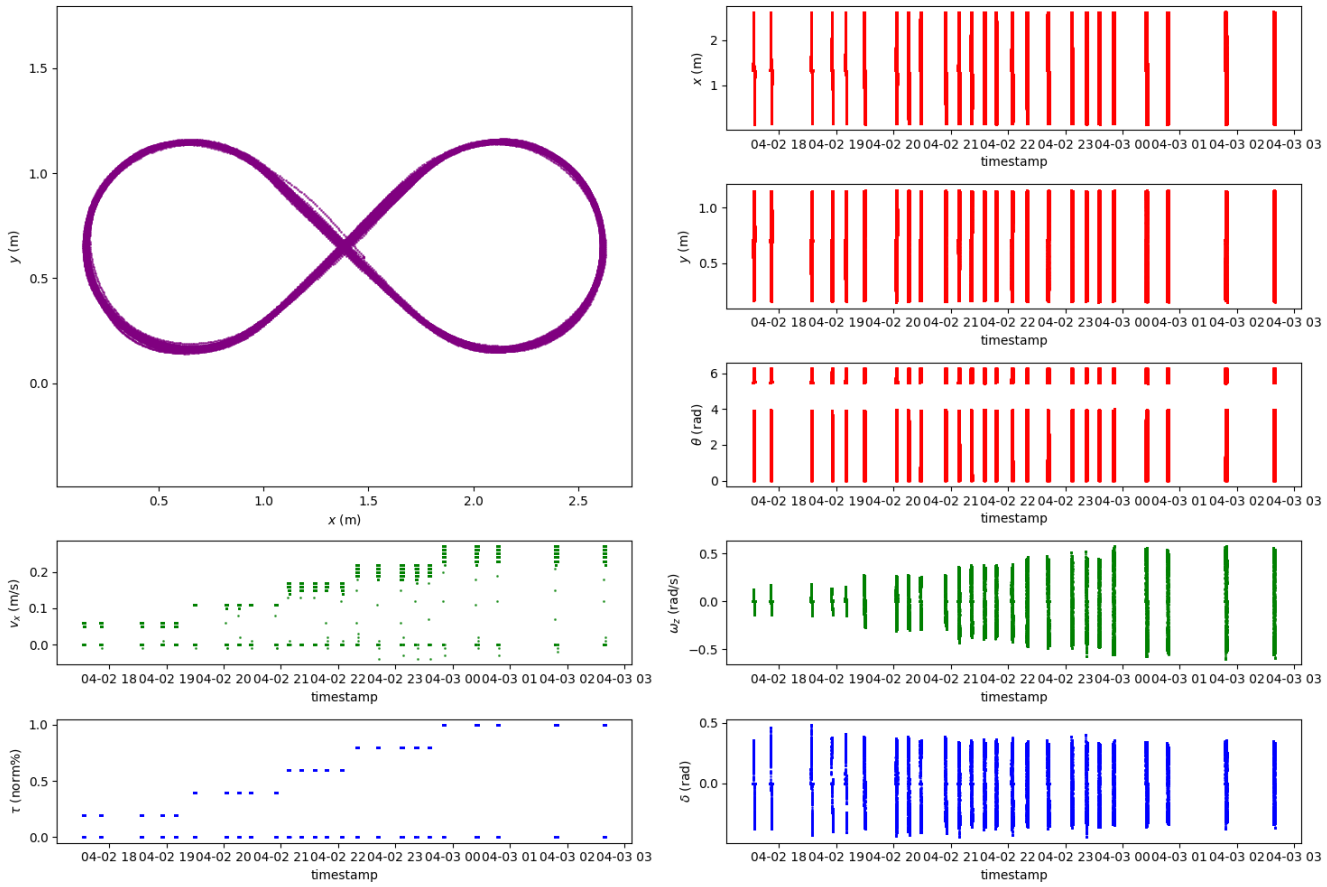


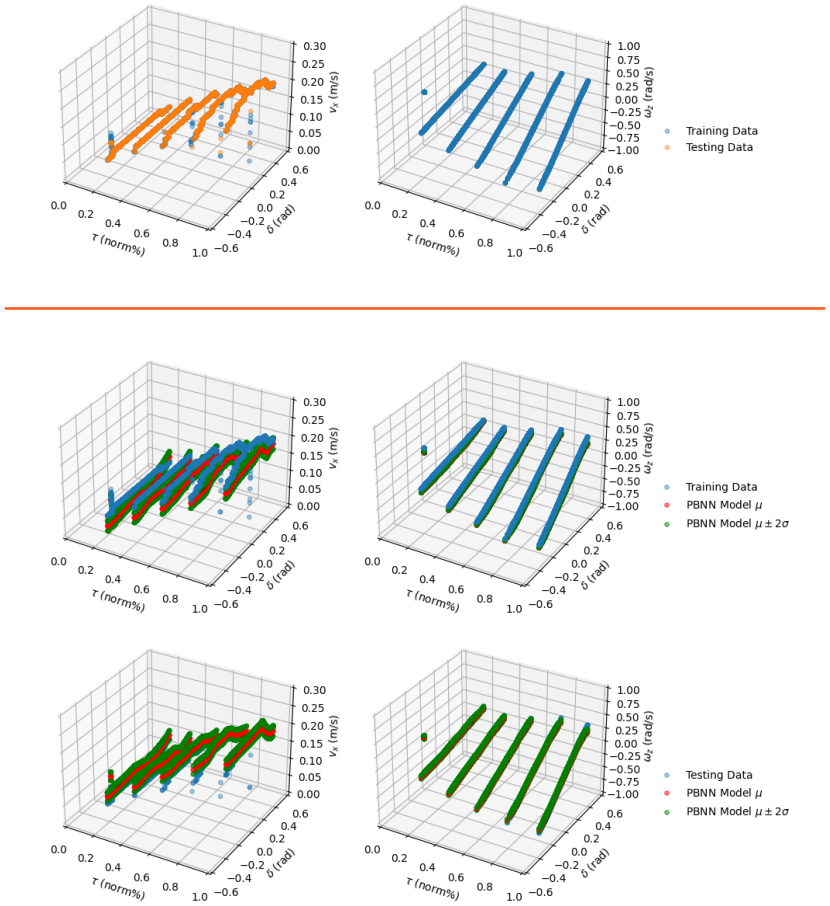




***Note**: This work adheres to Markov assumption.*

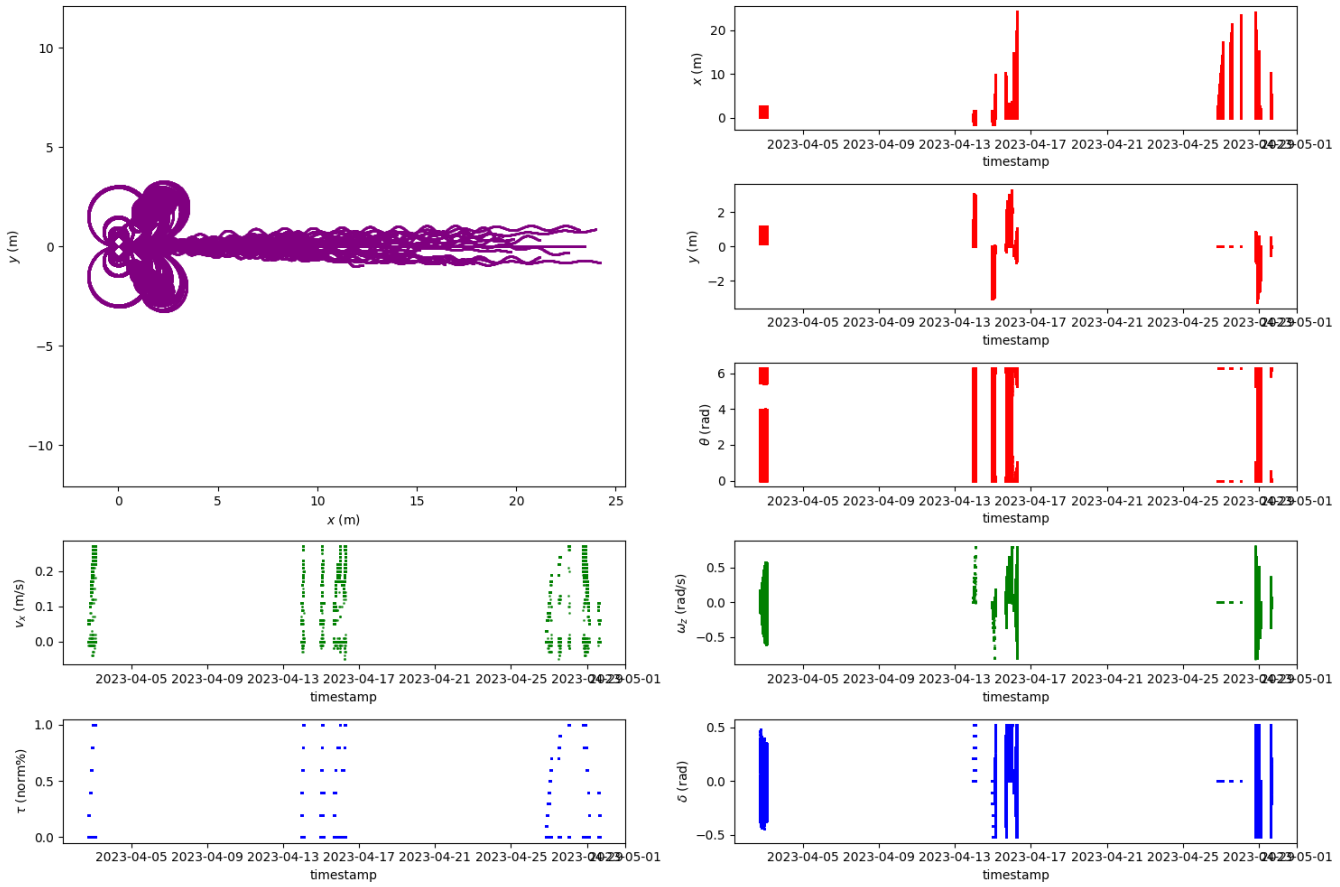
Model Inference – Eight Maneuver

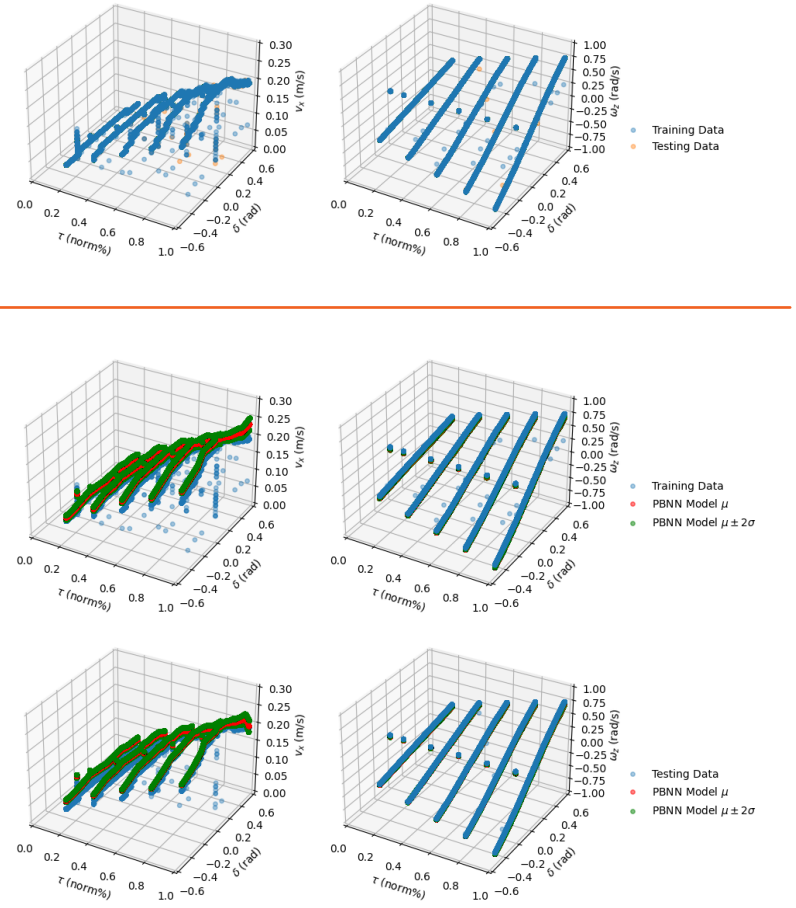




***Note**: This work adheres to Markov assumption.*

Model Inference – Generalization Across All Maneuvers





***Note**: This work adheres to Markov assumption.*

Future Plans

- Formulate and implement probabilistic Bayesian LSTM (delayed embedding)
- Try different input representation technique (e.g. timeseries snapshot of state information)
- Utilize other state (and perception) information from dataset
- Transfer learning for expanding model's ODD to different operating points
- Ensemble models for different operating points
- Simulation + real-world vehicle data (across scales)
- Active sampling to collect (sample) new data points smartly based on epistemic uncertainty

Thank You!

Open to questions and suggestions…