Friedrich-Alexander-Universität
Technische Fakultät

Recurrent Neural Networks (RNN) for Modeling of Nonlinear Systems

Sebastian Hirt | 20.10.2022
Lehrstuhl für Regelungstechnik

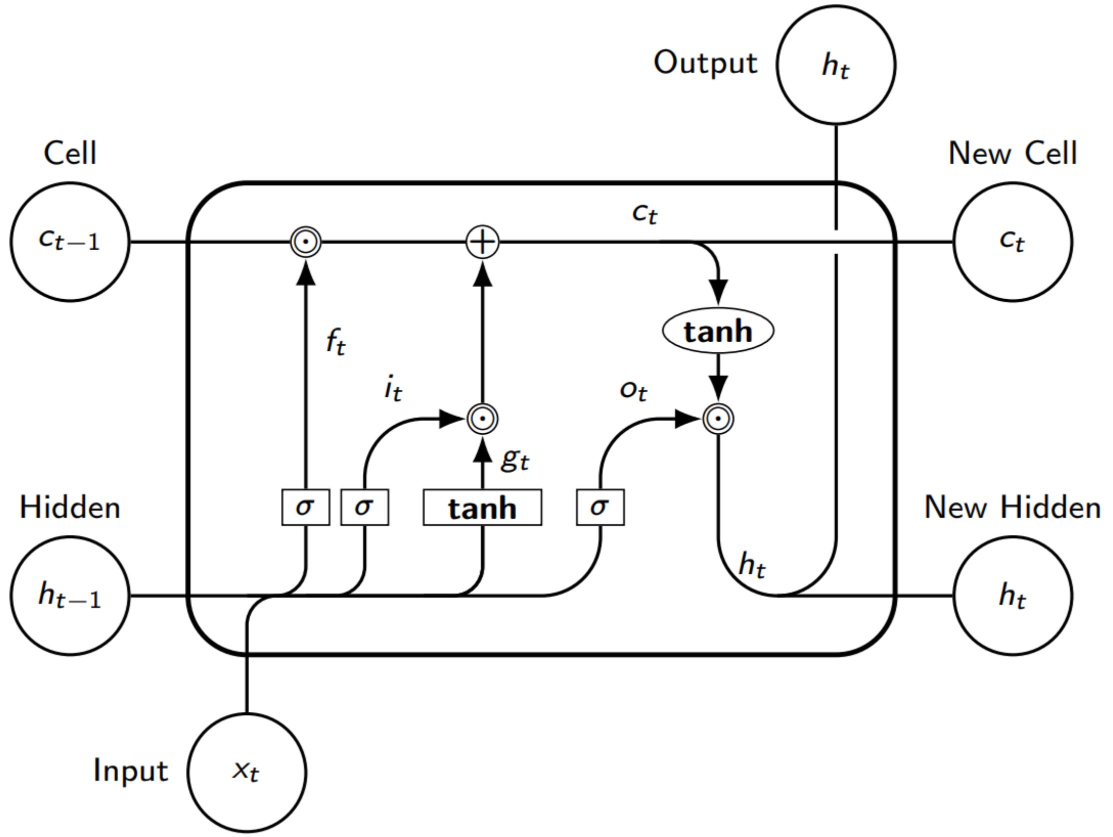
Introduction

Introduction

- ODEs describe nonlinear systems over time
- Recurrent neural networks capable of nonlinearities and time series
- Generation of training data
- Hyperparameter search

Methods

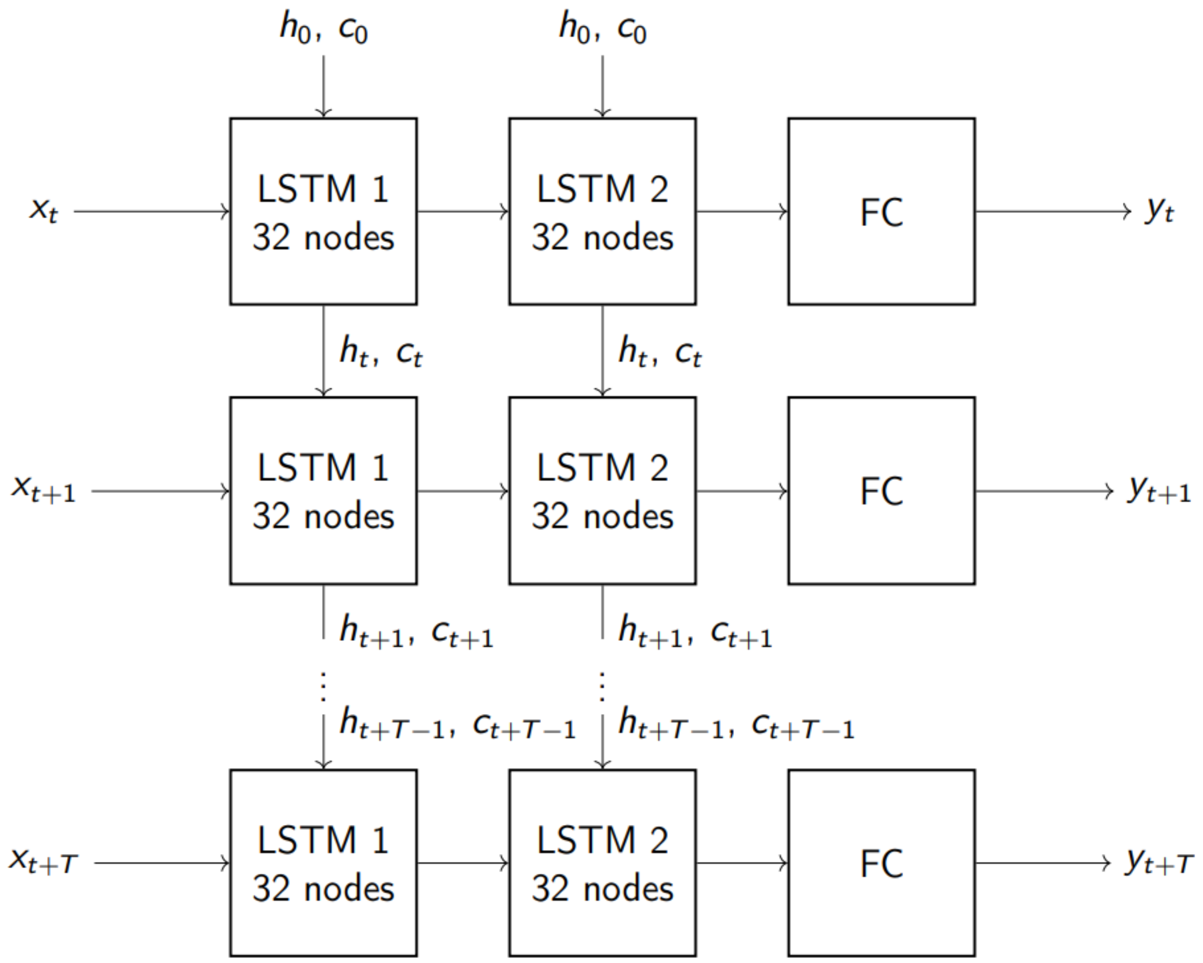
LSTM



$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{1}$$
$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{3}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

Adapted from (Yu, Y., Si, X., Hu, C., & Zhang, J., 2019)

Architecture



- Example of model with 32 nodes and 2 hidden layers
- States are carried to next time steps
- Weights are the same at all time steps
- Ability to predict indefinite future states
- Fixed number of timesteps for training

Optimization

- Forward pass
- MSE loss function
- Backpropagation through time (Werbos, P., 1990)
- Adam optimizer

Dataset creation

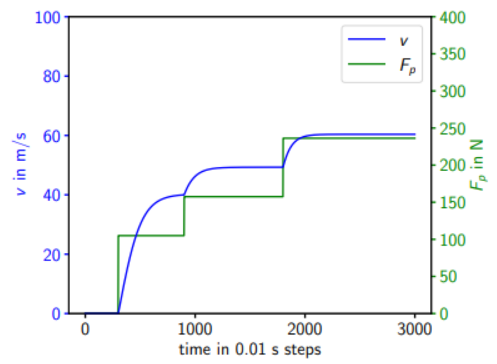
Problem description

- Pushing force on object
- Drag on object

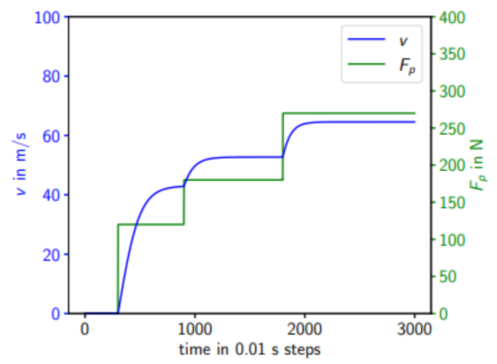
$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = F_p(t) - \frac{b}{m} v^2$$

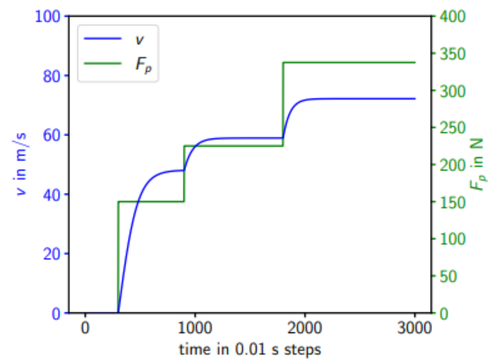
Drag simple



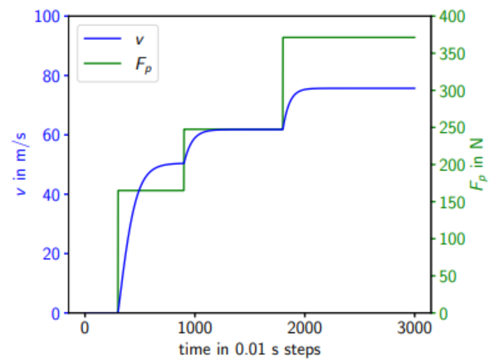
(a) Sample of the Drag-Simple dataset with steps of different height.



(b) Sample of the Drag-Simple dataset with steps of different height.

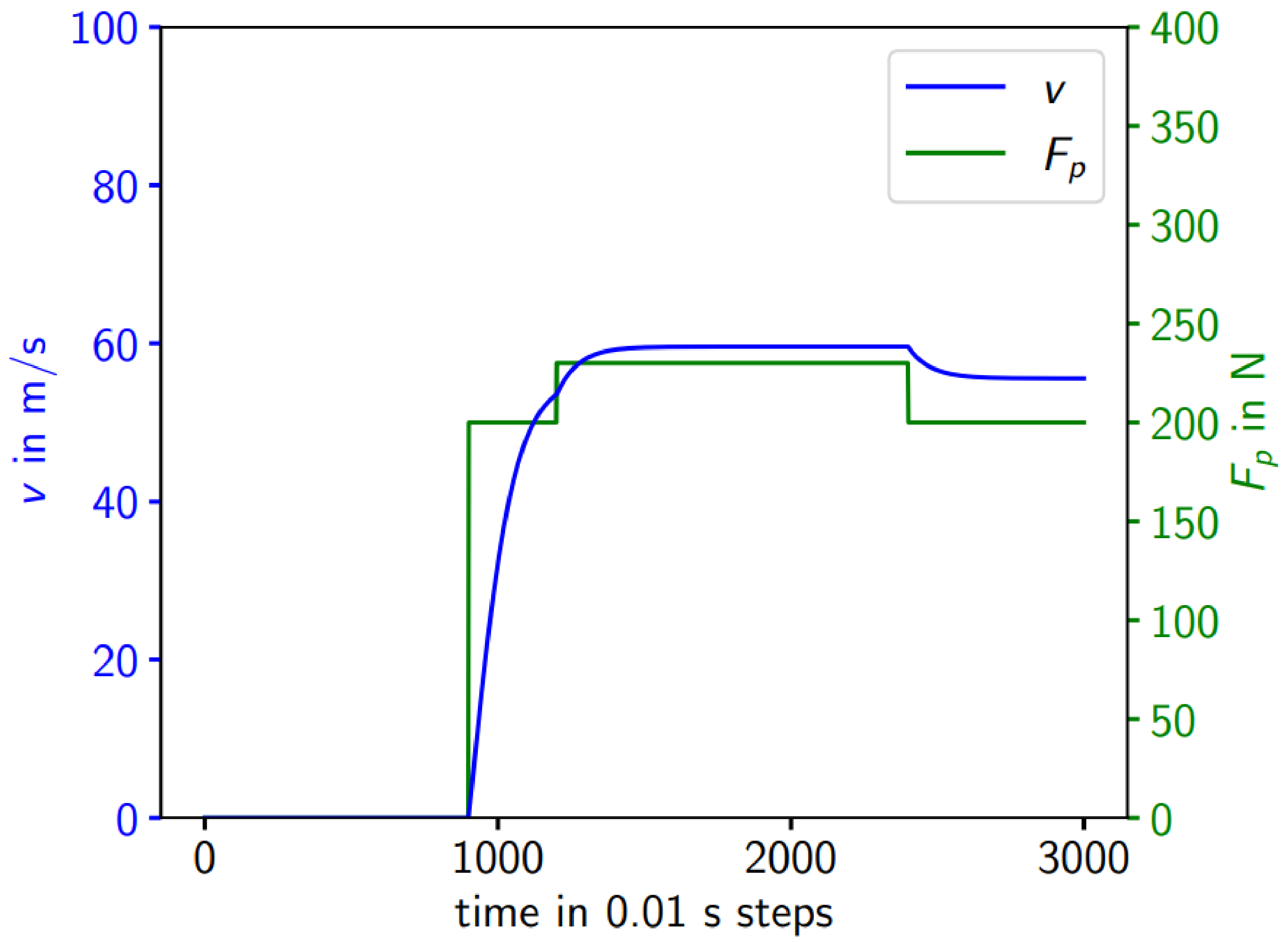


(c) Sample of the Drag-Simple dataset with steps of different height.



(d) Sample of the Drag-Simple dataset with steps of different height.

Drag complex



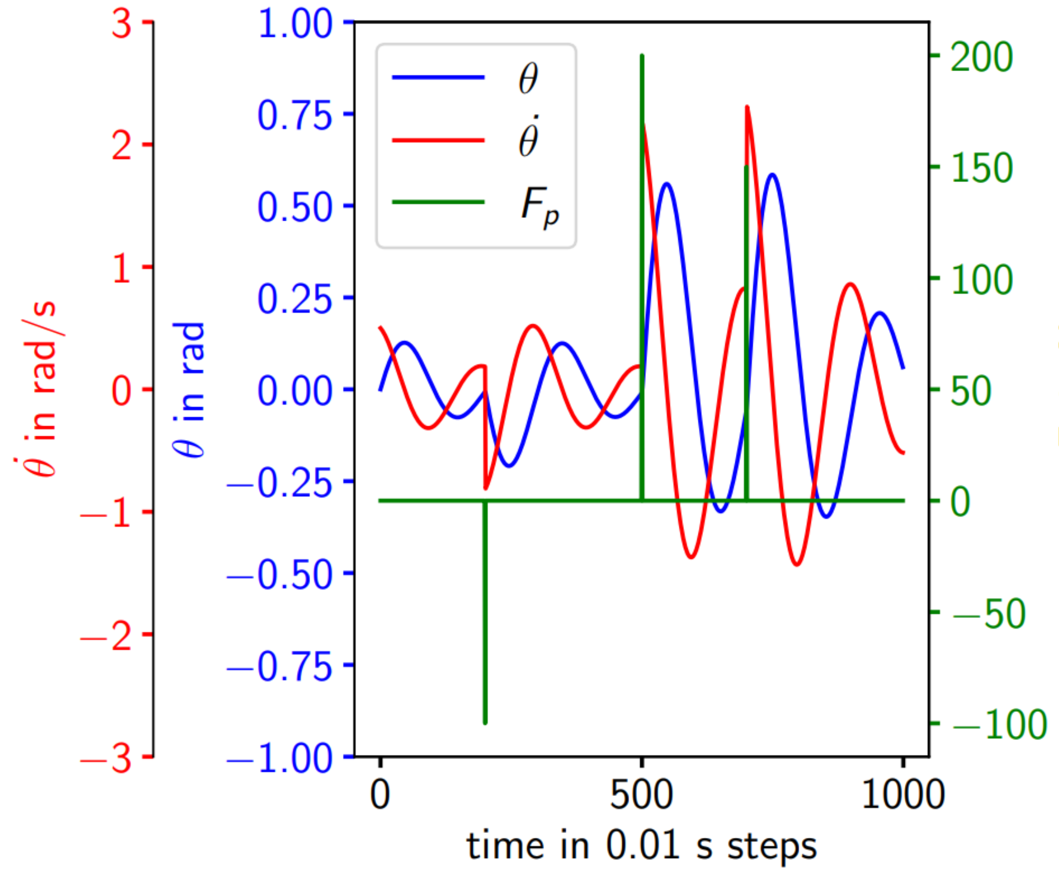
Problem description

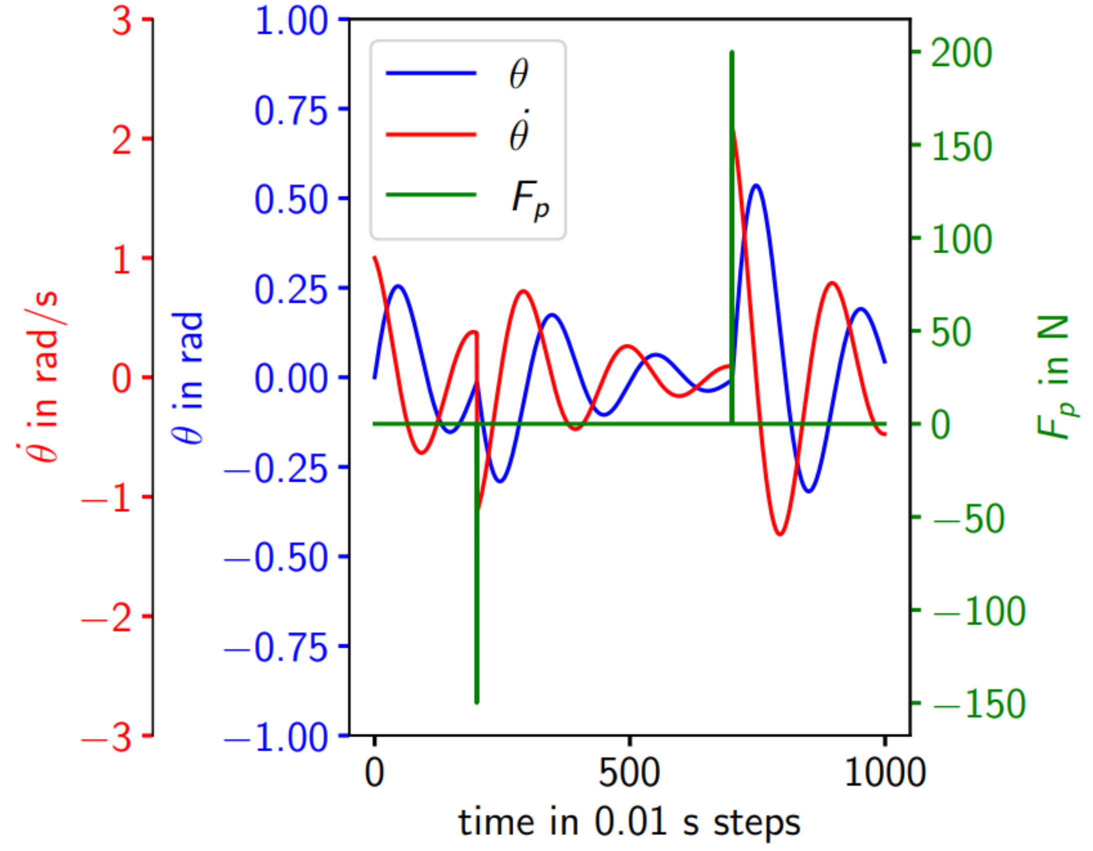
- Pushing a pendulum
- Friction due to air

$$\frac{d\theta}{dt} = \dot{\theta}$$

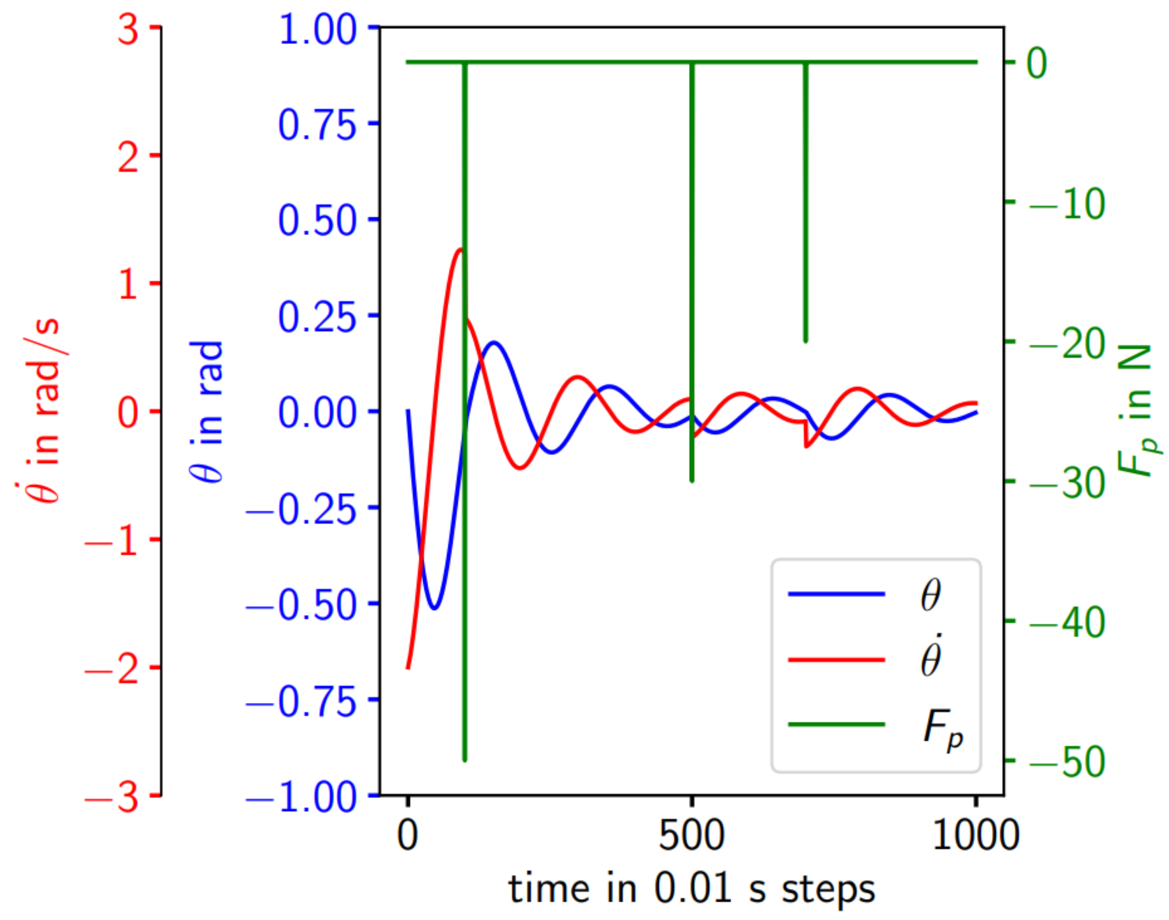
$$\frac{d\dot{\theta}}{dt} = F(t)\cos(f\,t) - q\dot{\theta} - \frac{g}{l}\sin(\theta)$$

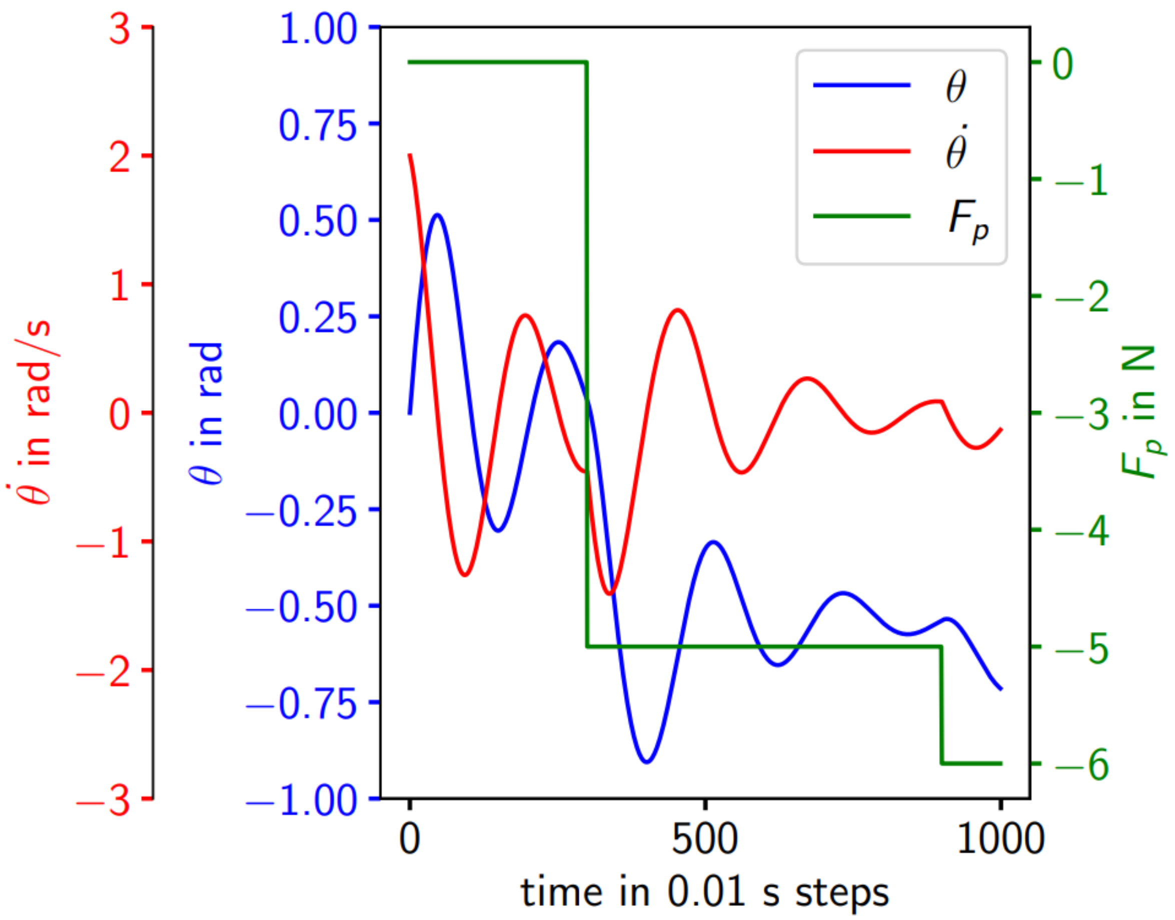
Pendulum simple





Pendulum complex





Preparation for Training

- Split into training, validation and test set
- Scaling with MinMaxScaler to [-1,1] to match LSTM output
- Initial condition given at first timestep, after zero
- Force input given at every time step

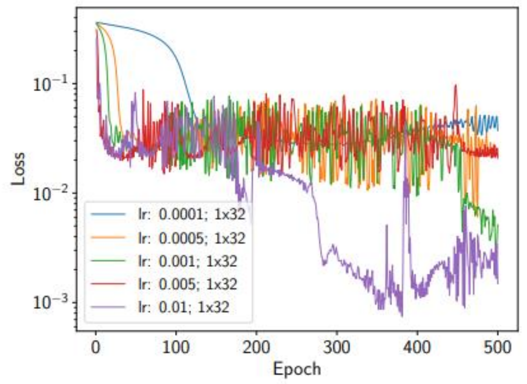
$$X = \begin{bmatrix} \theta_t & \dot{\theta}_t & F_t \\ 0 & 0 & F_{t+1} \\ 0 & 0 & F_{t+2} \\ \vdots & \vdots & \vdots \\ 0 & 0 & F_{L-1} \end{bmatrix} \quad y = \begin{bmatrix} \theta_{t+1} & \dot{\theta}_{t+1} \\ \theta_{t+2} & \dot{\theta}_{t+2} \\ \theta_{t+3} & \dot{\theta}_{t+3} \\ \vdots & \vdots \\ \theta_L & \dot{\theta}_L \end{bmatrix}$$

Hyperparameter search

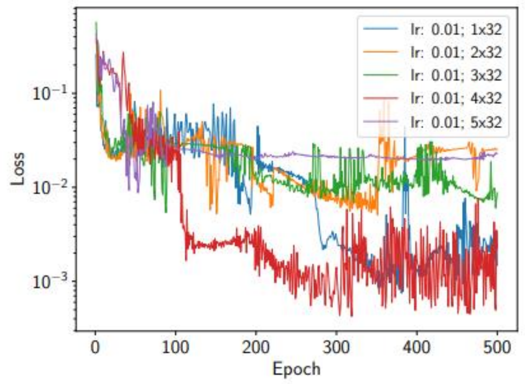
Manual search

- Monitoring validation loss for early stopping
- Check hyperparameters one by one
- Take best parameter for next step
- Order of searching:
  1. Learning rate
  2. Number of layers
  3. Number of nodes in each layer
- Training for 500 epochs, then 5000 with best model

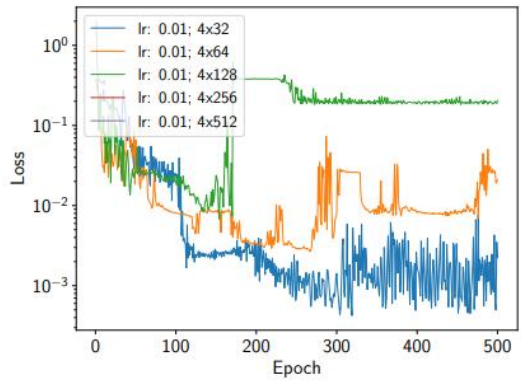
Drag simple



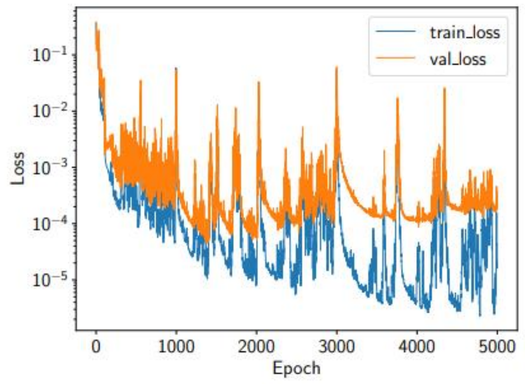
(a) Comparison of validation loss for different learning rates.



(b) Comparison of validation loss for amounts of layers.

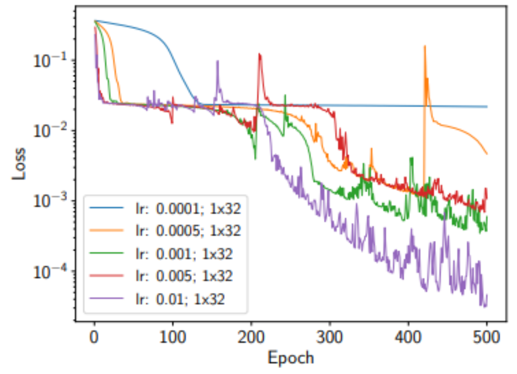


(c) Comparison of validation loss for amounts of nodes in each layer.

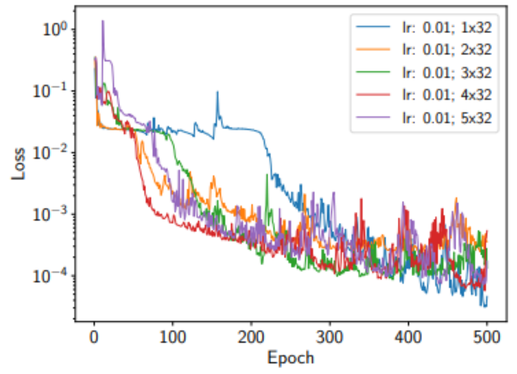


(d) Comparison of training and validation loss of the best model.

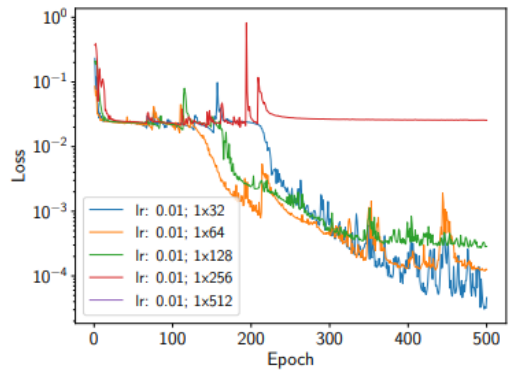
Drag complex



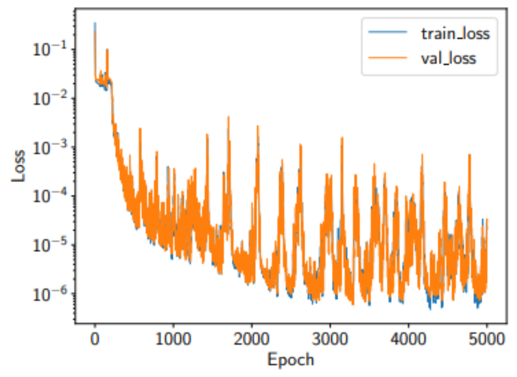
(a) Comparison of validation loss for different learning rates.



(b) Comparison of validation loss for amounts of layers.

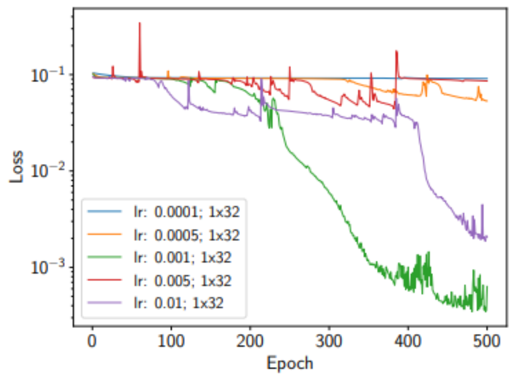


(c) Comparison of validation loss for amounts of nodes in each layer.

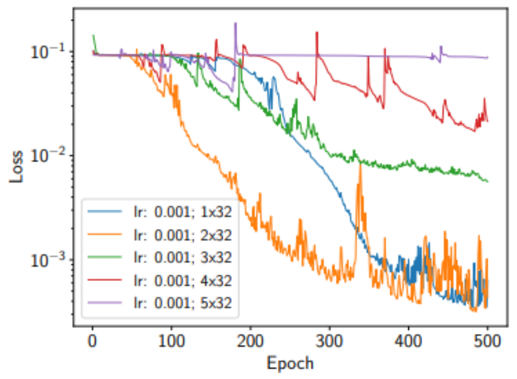


(d) Comparison of training and validation loss of the best model.

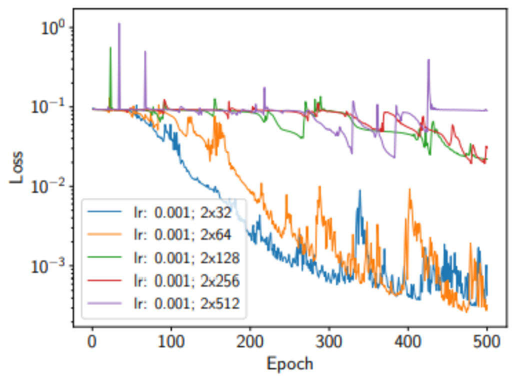
Pendulum simple



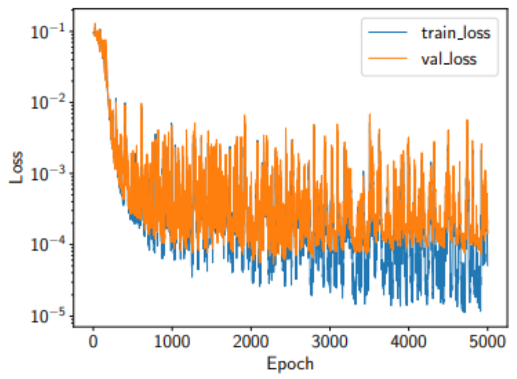
(a) Comparison of validation loss for different learning rates.



(b) Comparison of validation loss for amounts of layers.

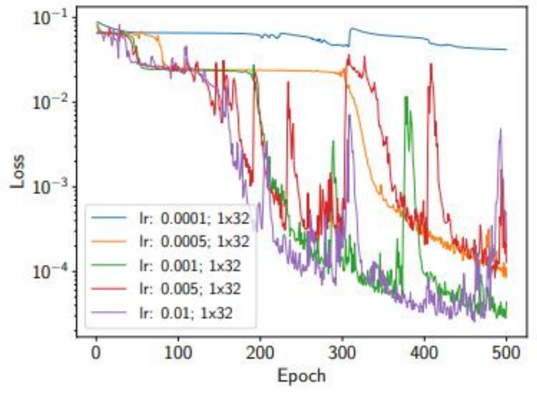


(c) Comparison of validation loss for amounts of nodes in each layer.

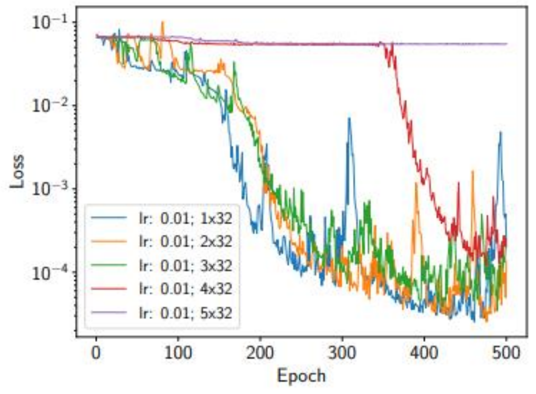


(d) Comparison of training and validation loss of the best model.

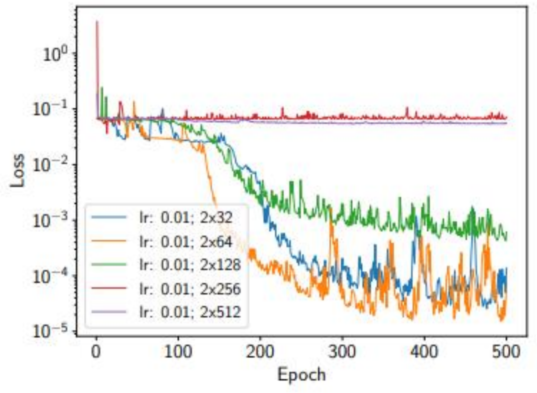
Pendulum complex



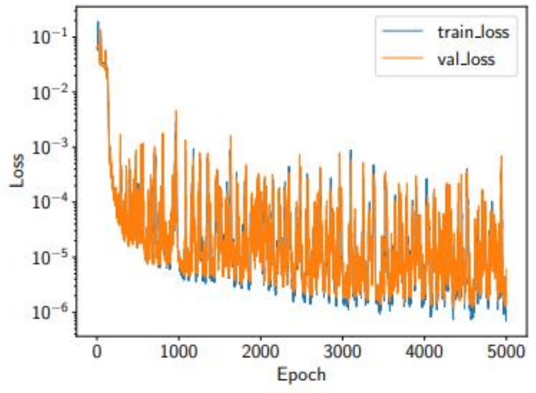
(a) Comparison of validation loss for different learning rates.



(b) Comparison of validation loss for amounts of layers.



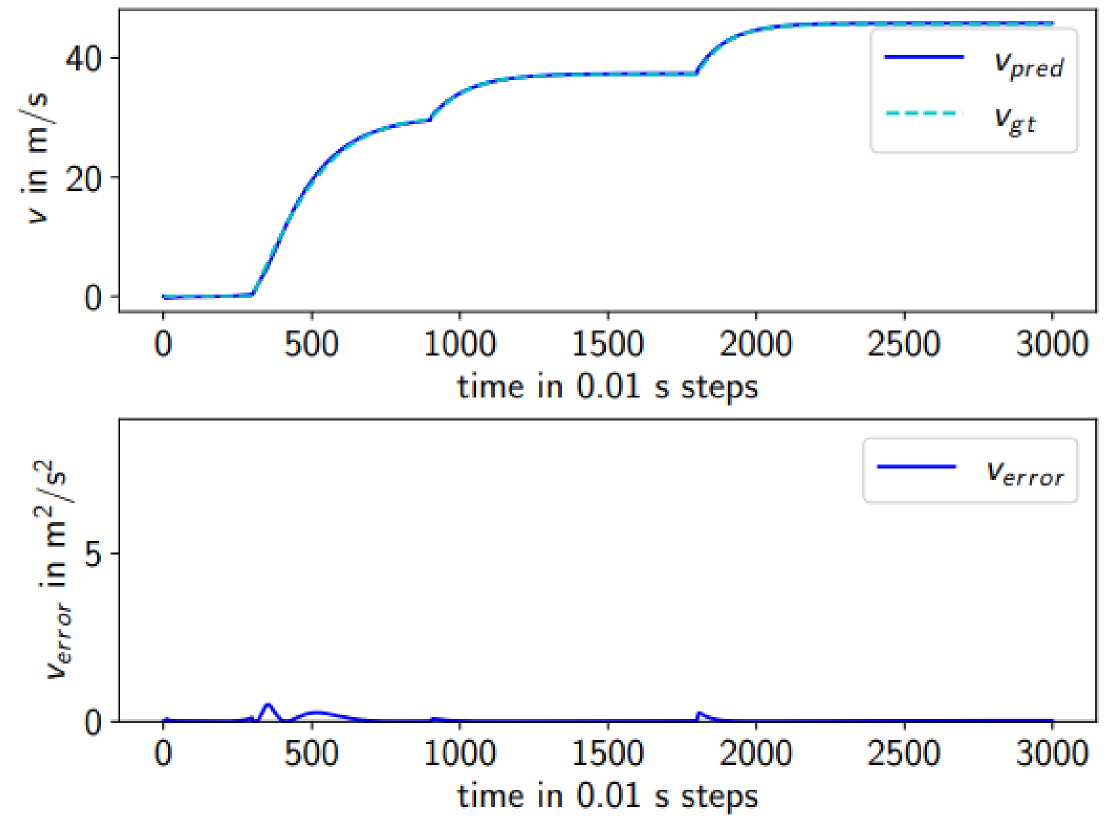
(c) Comparison of validation loss for amounts of nodes in each layer.

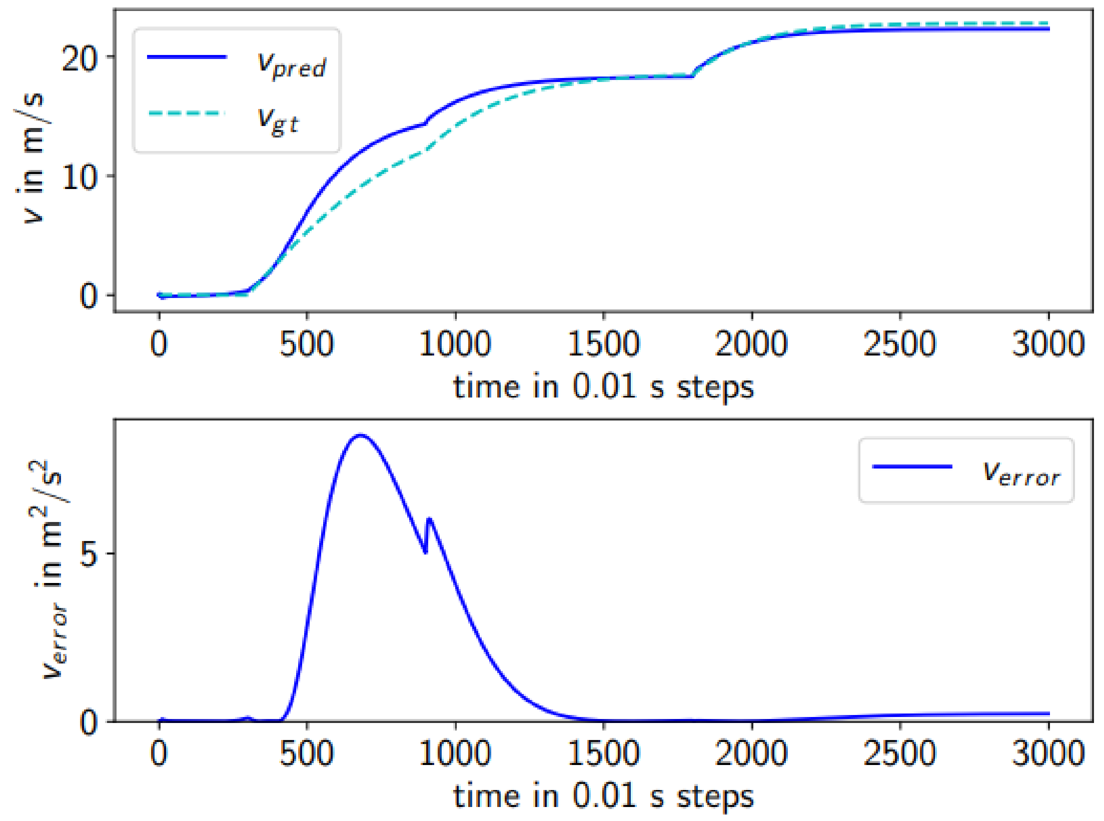


(d) Comparison of training and validation loss of the best model.

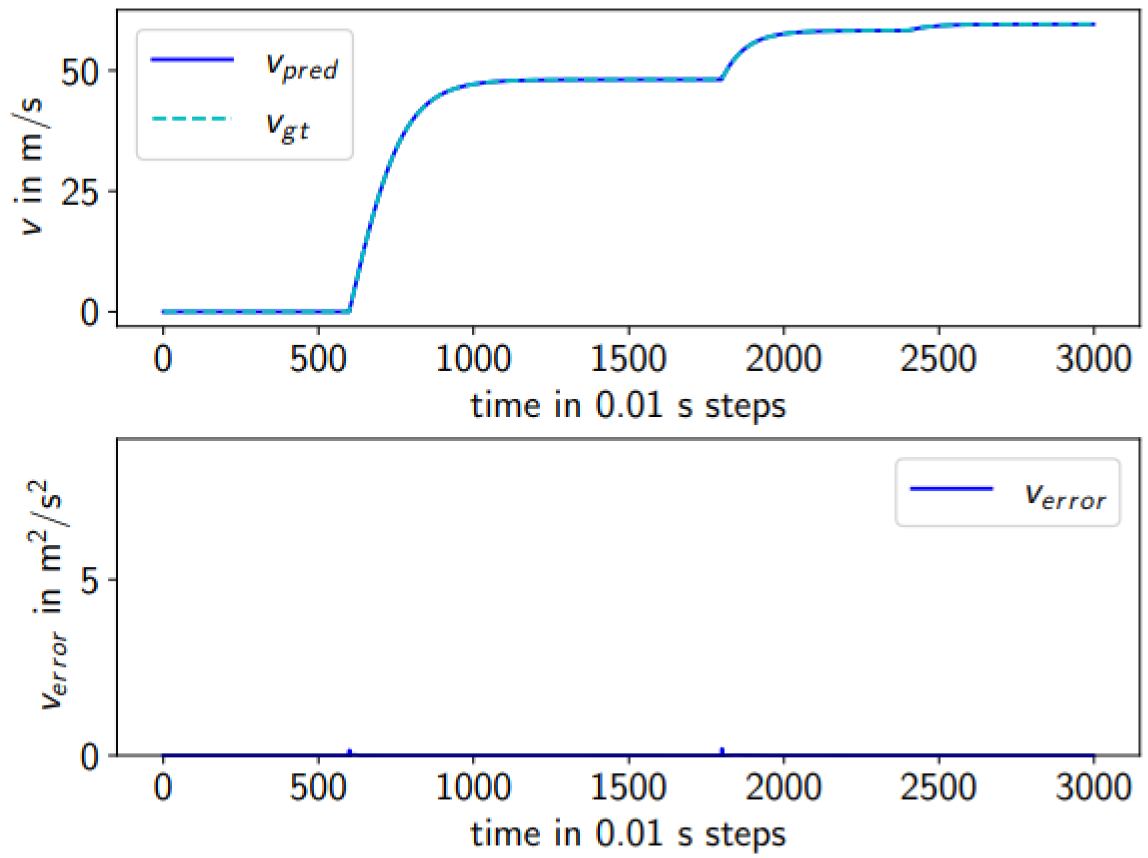
Evaluation

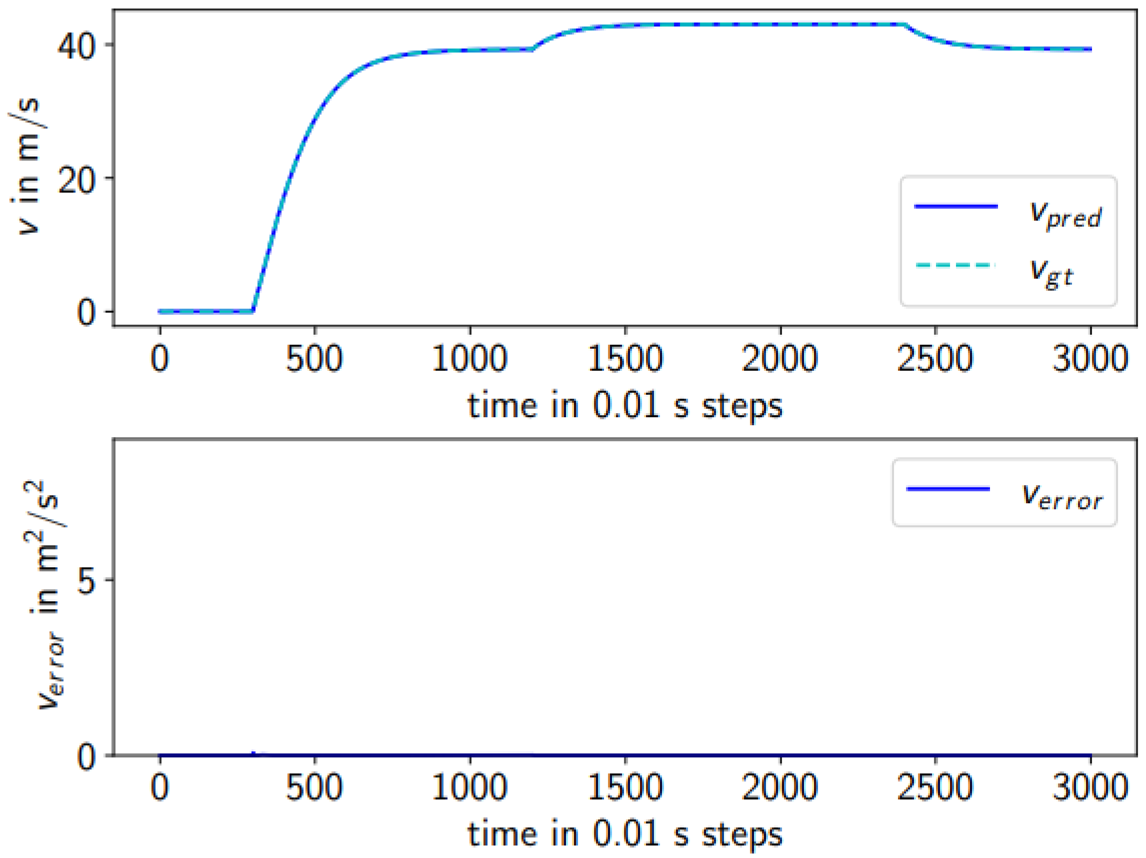
Evaluation – Drag simple



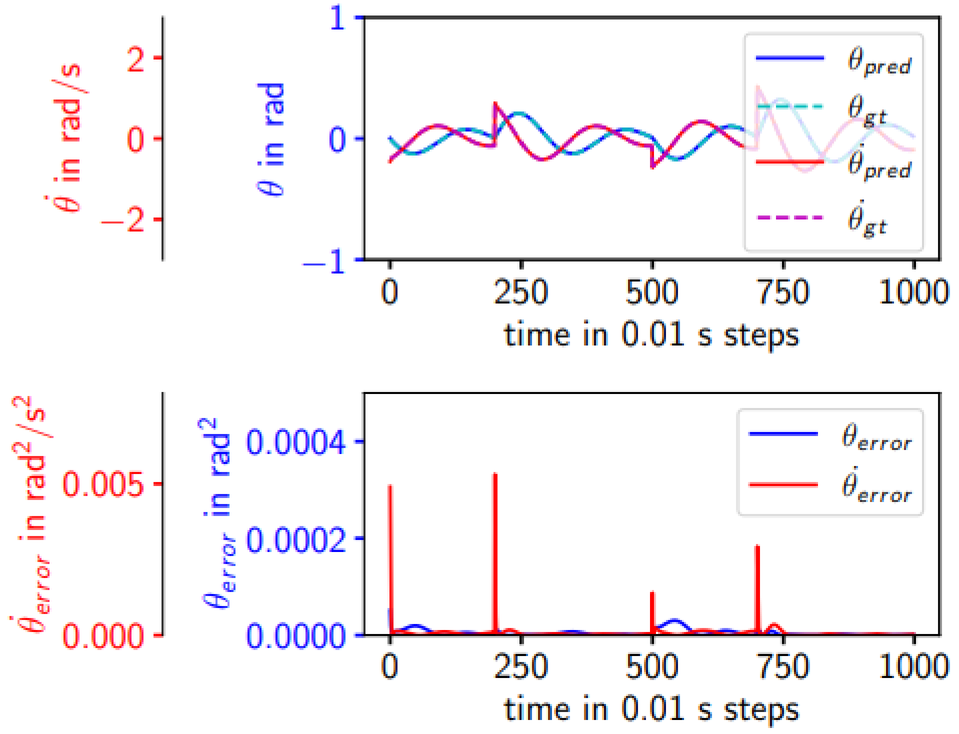


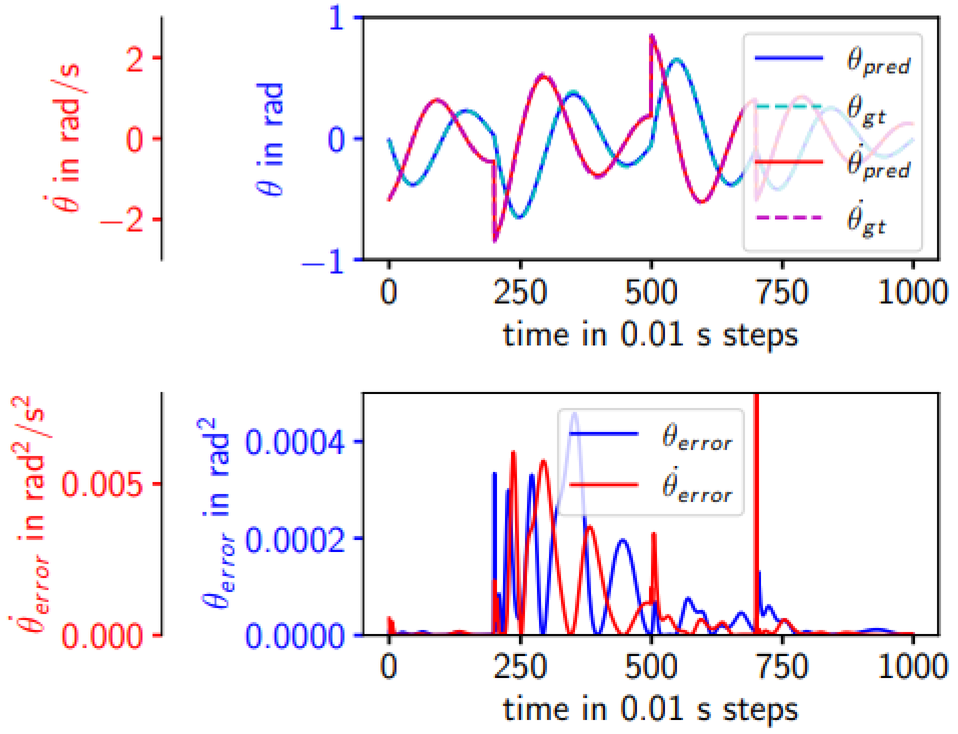
Evaluation – Drag complex



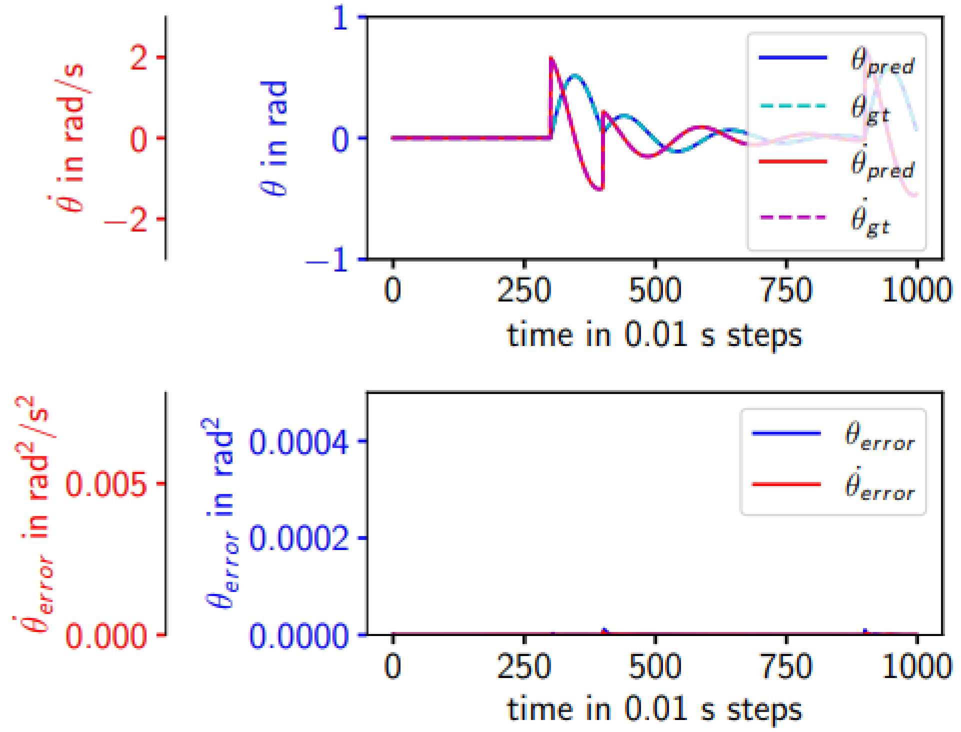


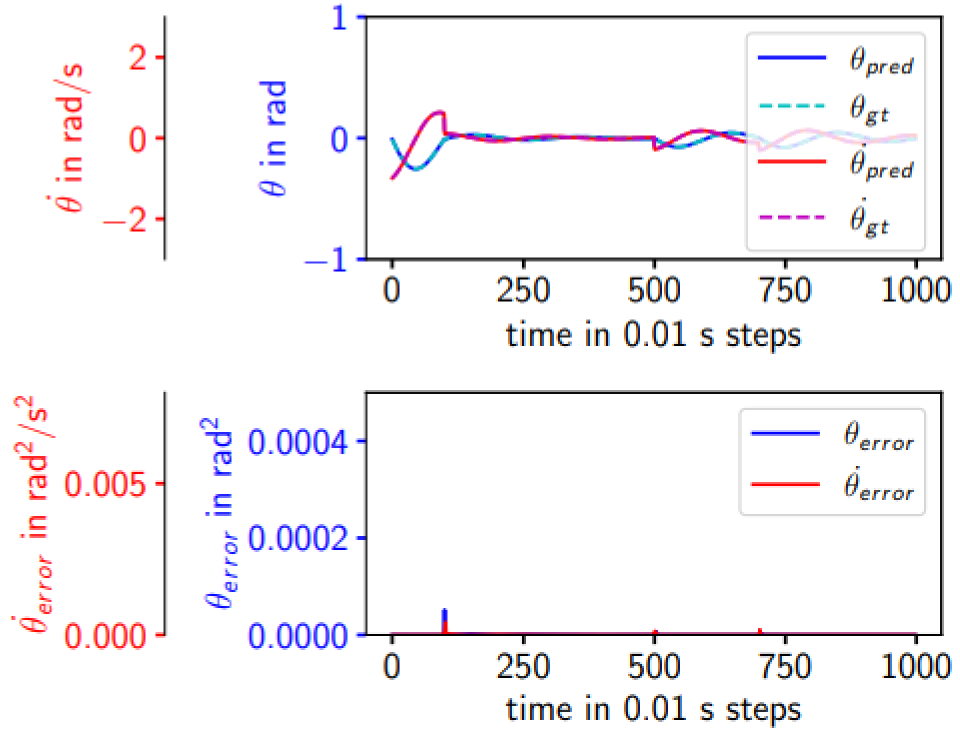
Evaluation – Pendulum simple





Evaluation – Pendulum complex





Conclusion

- Difficulties in training for larger model
- Clear ability to predict nonlinear system
- Error spikes on input change
- Recommendation:
  - Use 2 LSTM layers with 32/64 nodes and learning rate of 0.01 for similar systems
  - New hyperparameter search for different systems

Future work

- Create better dataset to test specific situations
- Analysis on what and how much training data is necessary to achieve good results
- Testing for more complex ODEs
- Deeper look into training instabilities necessary for larger models

Vielen Dank!

References

- Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, *31*(7), 1235–1270.
- Werbos, P. (1990). Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10), 1550–1560.

Problem description

- Pushing force:
  - $F_p = ma$
- Air resistance (Drag):
  - $F_d = \frac{1}{2}\rho v^2 C_D A$
- Simplification:
  - $b = \frac{1}{2}\rho C_D A$
- Resulting force:
  - $F = F_p - F_d$

Problem description

- Resulting force:
  - $F = F_p - F_d$
- Express as second order differential equation:
  - $m\frac{d^2x}{dt^2} = F_p - b\frac{dx}{dt}$
- Transform into system of two first order equations:
  - $\frac{dx}{dt} = v$
  - $\frac{dv}{dt} = \frac{F_p - bv^2}{m}$
  - $\rightarrow$ use in odeint python function to get values of velocity and/or position