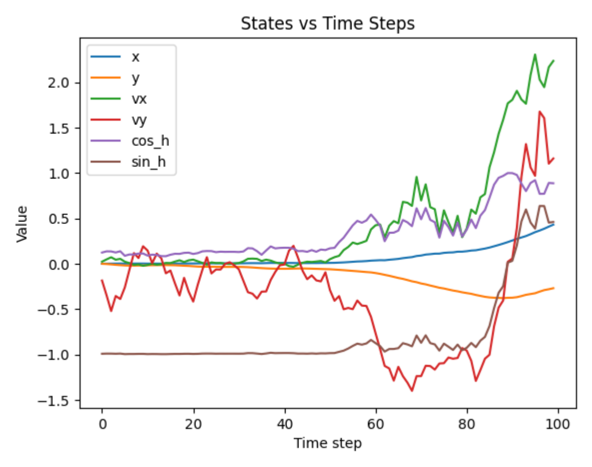
Generated at Thu Feb 15 03:07:43 2024.

Source at https://jupyterhub-dev.cheme.cmu.edu/user/tungyuh@andrew.cmu.edu/lab/tree/s24-06642/tungyuh_P2.ipynb.

Ans 1.1

Run one episode with random actions. Check the observation dictionary returned by the environment. Store the observations (states) and rewards over time! What is your target state (desired goal) s∗? What is your final state sn at the 100-th time step? Plot the values of all the states over time!

Ans



Final state at 100th time step : [0.43126793 -0.26964677 2.23678071 1.16072831 0.88760593 0.46060364]

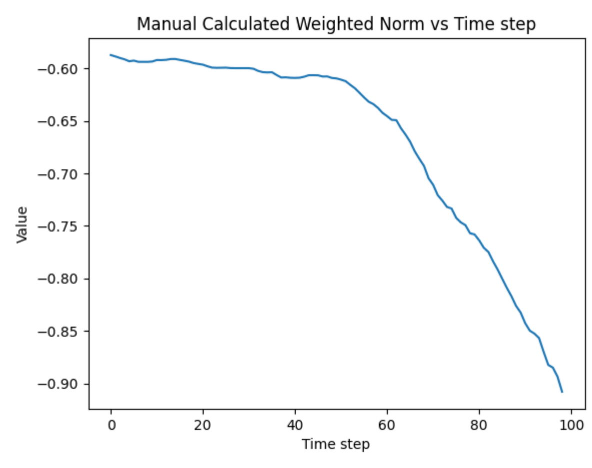
Desired state : [-2.600000e-01 1.400000e-01 0.000000e+00 0.000000e+00 6.123234e-17 1.000000e+00]

Ans 1.2

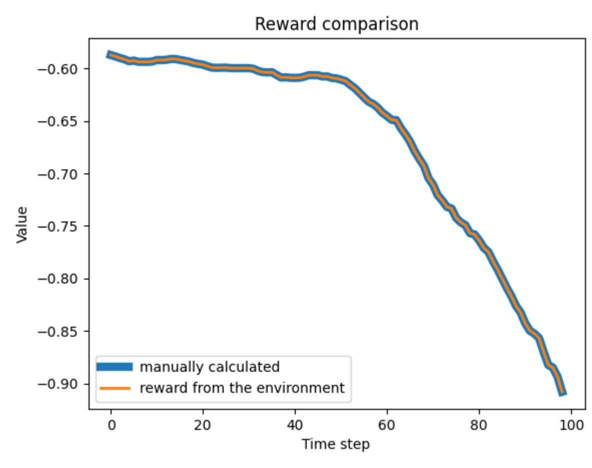
Calculate the reward of the environment manually. The reward rt is defined as the negative of the weighted Euclidean norm of |st −s∗|. That is, for some weight β = [β(1),··· ,β(d)],

where d is the dimension of the state variables and β = [1,0.3,0,0,0.02,0.02]. (Hint: Compare your manual calculation with the environment reward to check if you are correct!)

Manual Calculated Weigthed norm



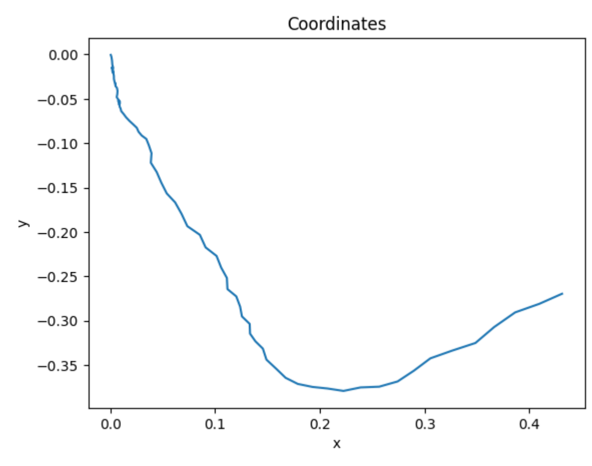
Comparison



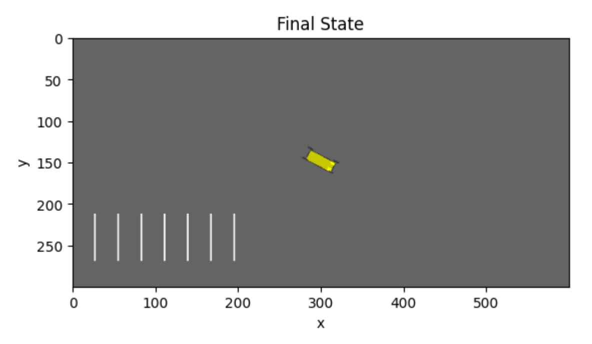
Ans 1.3

Plot the positions (x and y coordinate) of the vehicle! What is the reward at the final time step in the episode? Show the rendered image of the environment at the final time step in the episode!

Coordinates

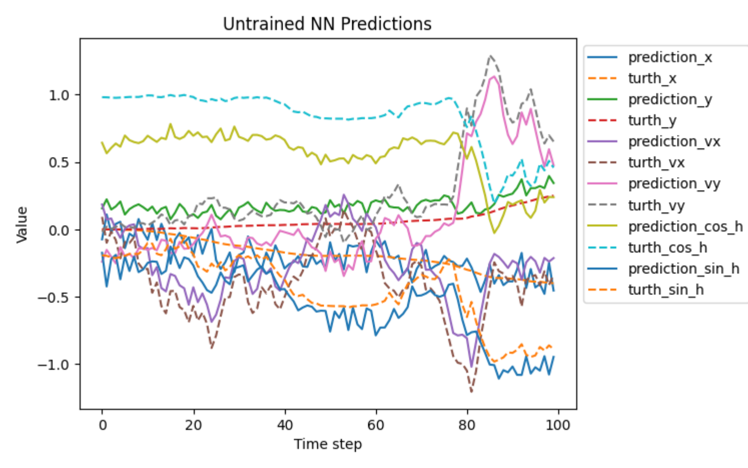


Final State



Ans 2.1

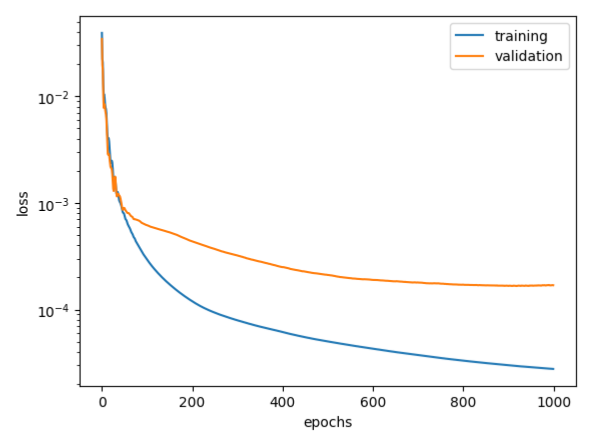
Construct a NN model with one hidden layer with random initialization (the structure is given in the Colab notebook). Implement the forward pass. Perform one run of episode and plot the untrained NN predictions and the true values of all the states over time!



Ans 2.2

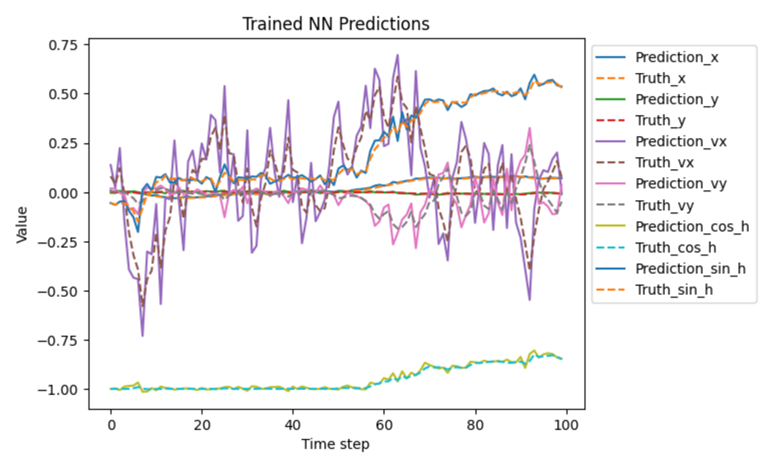
Create a batch of experiences dataset D with n = 2000 using the provided function. Train a NN model that consists of one hidden layer with 128 nodes and ReLU activation (the structure is already provided in the Colab notebook). Use this dataset with training and validation ratio 70%-30% using Adam optimizer with learning rate 0.01 and training epoch 1,000. Plot the MSE loss for training and validation set!

NLL loss



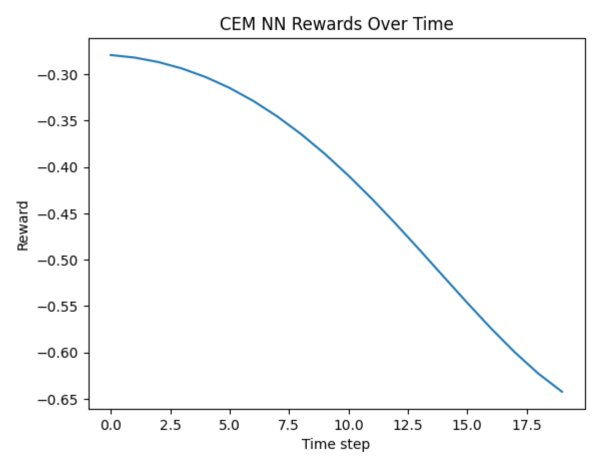
Ans 2.3

Perform one run of episode again and plot the trained NN predictions and the true values of all the states over time!



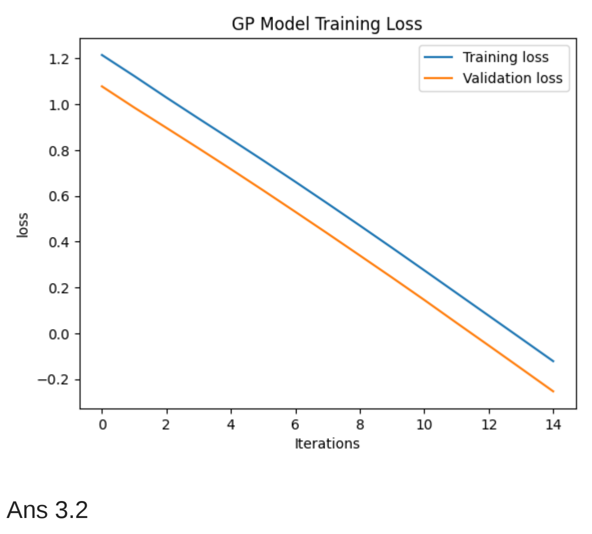
Ans 2.4

Implement the CEM planner. Combine the NN model with CEM planner. Evaluate the performance of your model when planning using a time horizon H = 5 and K = 10 possible action sequences only for n = 20 timesteps. Plot the rewards over time!



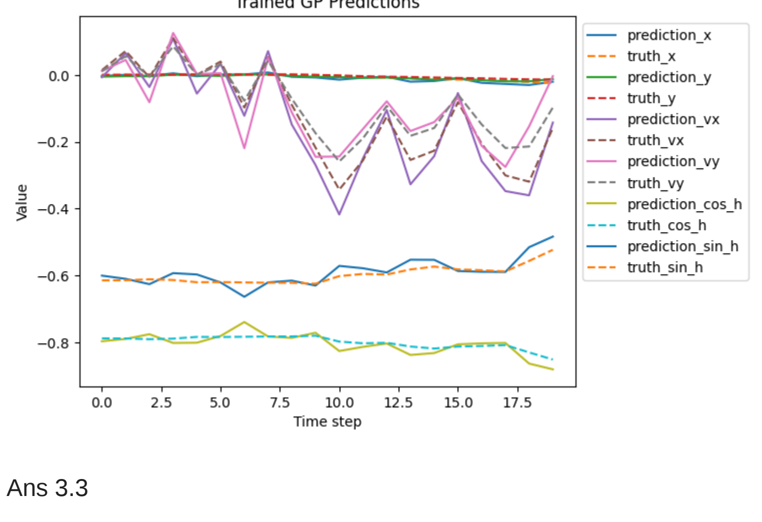
Ans 3.1

Construct and train the provided GP model. Train the model using dataset D from 2.2 with training and validation ratio 20%-80% using Adam optimizer with learning rate 0.2 and training epoch 15. Note that GP is more sample-efficient than NN, so we need fewer training samples for GP. More training samples will also slow down the speed. Plot the NLL loss for training and validation set!



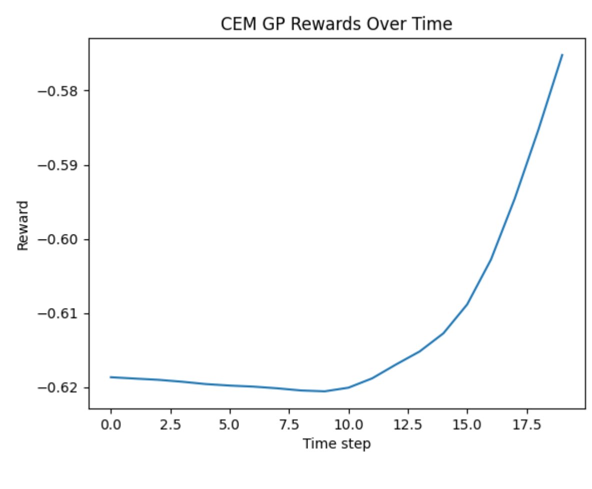
Ans 3.2

Perform one run of episode again for only n = 20 timesteps and plot the trained GP predictions and the true values of all the states over time!



Ans 3.3

Combine the GP model with planning. Evaluate the performance of your model when planning using a time horizon H = 5 and K = 10 possible action sequences only for n = 20 timesteps. Plot the rewards over time!



Ans 3.4

Compare NN and GP with the following metrics: final reward, prediction error, and computation time!

Ans

Final reward of NN with CEM: -0.6422626130203086

Final reward of GP with CEM: -0.5752008359081332

Neural network prediction error: 0.0018271555891260505

Neural network computation time: 0.20585942268371582

Gaussian process prediction error: 0.0008697591838426888

Gaussian process computation time: 98.4386670589447

%run ~/s24-06642/s24.py %pdf

In [ ]:

```
%run ~/s24-06642/s24.py
%pdf
```