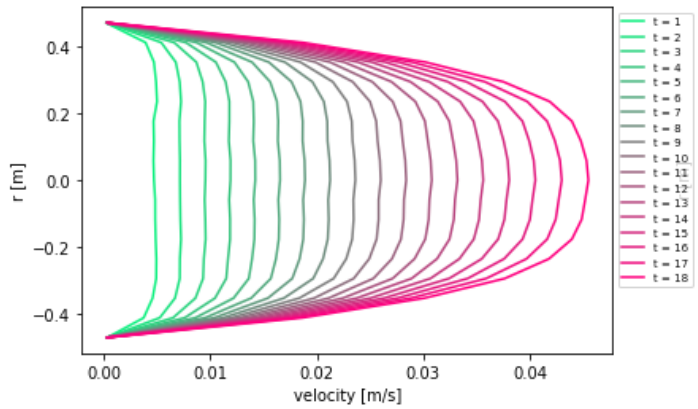
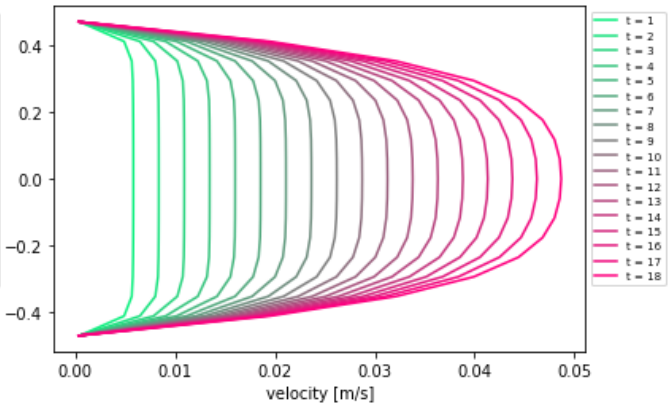
P1

Monday, March 22, 2021 2:14 PM



Predicted Velocity Graph



Ground Truth Velocity Graph

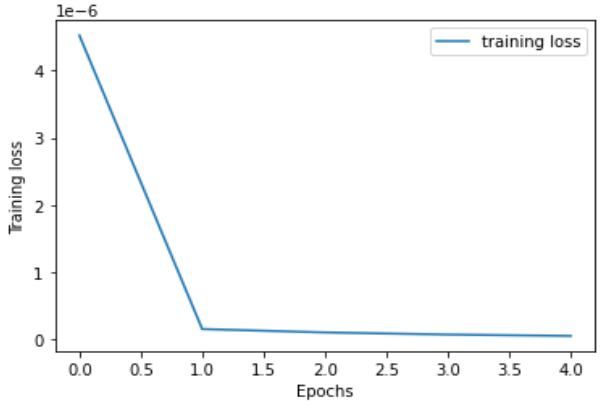
Model:
I used Pytorch's LSTMCell module for both train and test in order to control the model parameters more easily. I also defined a nn.Linear layer to transform the output so that its last dimension is the same as the inputs', not the hidden layers'.

For training, since the hidden states hx and cx should be of dimensions (batch_size, self.hidden_size), I defined a helper function init_hidden_state to initialize them to zero. Using a for loop through each timestep, I called self.lstmCell and appended the output at each timestep to construct a final output (the predictions).

For testing, there is only one initial 'x' predicting multiple outputs; this is a One-to-Many structure. In my for loop through the timesteps, I propagate the output of each lstmCell into the next lstmCell's input.

Loss Function: Since this is essentially a regression problem, I used Mean-Squared Error loss to evaluate the loss between the predicted velocities and the ground truth.

Test L1 error: 1.12
Test L2 error: 0.0038



Hyperparameters:

num_epochs = 5
lr = 0.0025 (no learning rate decay/scheduler)
num_layers = 2
dropout = 0.1

When I initially experimented with 20 epochs with a learning rate of 0.005, I observed a training loss graph that drastically diverged after 3 epochs. This meant the learning rate was too high. In addition, the L1 error was quite high (1.98).

Using a smaller learning rate of 0.0025 helped to prevent overshooting, but still ensured quick convergence. Increasing the num_layers to four did not significantly boost the accuracy, and also took longer to run because the recurrent depth is greater.