Lukas Crockett
ECE 662

Project 1

1. To solve the problem posed in question 1, I used an equinox MLP with 2 hidden layers: [2, 200, 100, 2], where the inputs to the network were θ and h from the s array (first array contained in data.npy) and the outputs were φ_{3z} and φ_{3x}, the two components of force f_3. The first φ array was used for the target training values (The second array contained in data.npy). After the first linear layer a tanh activation function was used, and for the next relu was used. The final layer did not use an activation function.

   Since this was a regression problem, the mse loss function was used, and to evaluate the model's predictions, the root mean squared error (RMSE) was calculated for both outputs compared to the target values to compare the model fit to the training values. Additionally I used an optax adamw optimizer with a learning rate of 1e-3 and a weight decay of 1e-4 which seemed to provide an acceptable training speed along with improved accuracy compared to a standard adam optimizer due to the L2 regularization.

   For training and evaluation of the model, the dataset was shuffled and split into 80% training and 20% test portions. After each epoch, the model is evaluated against the test set. Within each epoch training is performed using minibatches of size 64. After 500 epochs, a loss of ~0.003 and RMSE of ~0.06 on both test and train data sets is achieved.
2. For problem 2, the input layer of the network used in the first problem was expanded to accept 3 inputs (θ, θ', and h). The provided second φ array was used to model the box in pure rotational motion about C_1 and output the same φ_{3z} and φ_{3x} components of f_3. All other model parameters remained unchanged as the model performed acceptably.

   Even with the additional model input, the network trained about as fast as problem 1, reaching a loss of ~0.004 and RMSE of ~0.06 after 500 epochs on both test and train sets.

3. I found that the network from problem 2 also modeled the constant linear velocity of the box in the x direction situation posed in question 3 acceptably. The third provided φ array was used for training this model under the same conditions as problem 2. With the added complexity of modeling this situation, the network did train slower compared to the previous situations. After epoch 500, the model reached a loss of ~0.02 and RMSE of ~0.14 on both train and test, though the model fit does improve somewhat with an expanded number of epochs.

I used python notebooks to develop the models for each problem, and include a standalone py file (run_all_problems.py) that performs training and evaluation of each problem's model sequentially.