

# EML 6351 Simulation Project 6

---

## Implement Neural Network-based controller

---

### README

1. All the codes are inside **src** file:

- The **Neural\_Networks\_Continuous.m** file is for implementing with a neural network-based controller with a typical continuous feedback control law for the dynamics, along with the **Model\_Composite\_2Link\_NN\_Continuous.slx** file.
- The **Neural\_Networks.m** file is for implementing with a neural network-based controller with a discontinuous sliding-mode feedback control law for the dynamics, along with the **Model\_Composite\_2Link\_NN.slx** file.
- The **Neural\_Networks\_RISE.m** file is for implementing with a neural network-based controller with a continuous RISE feed-back control law for the dynamics, along with the **Model\_Composite\_2Link\_NN\_RISE.slx** file.

### Dicussion

- **(A) Simulation Section**

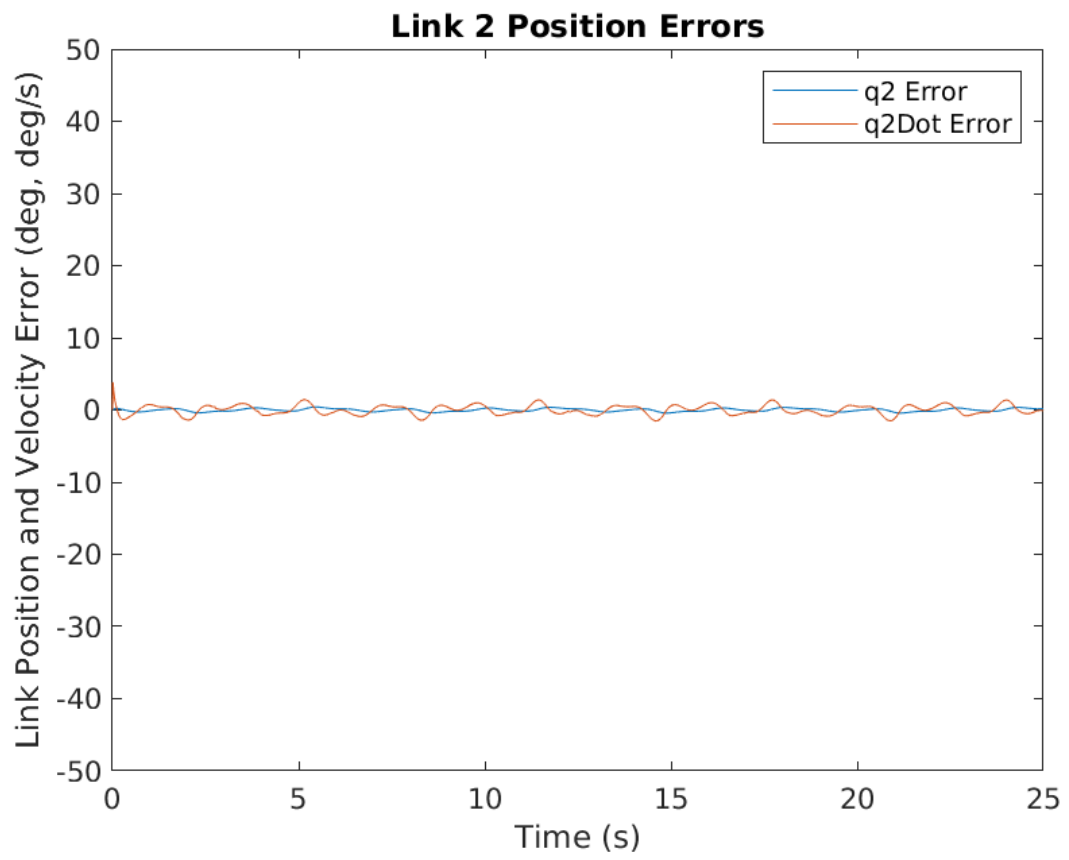
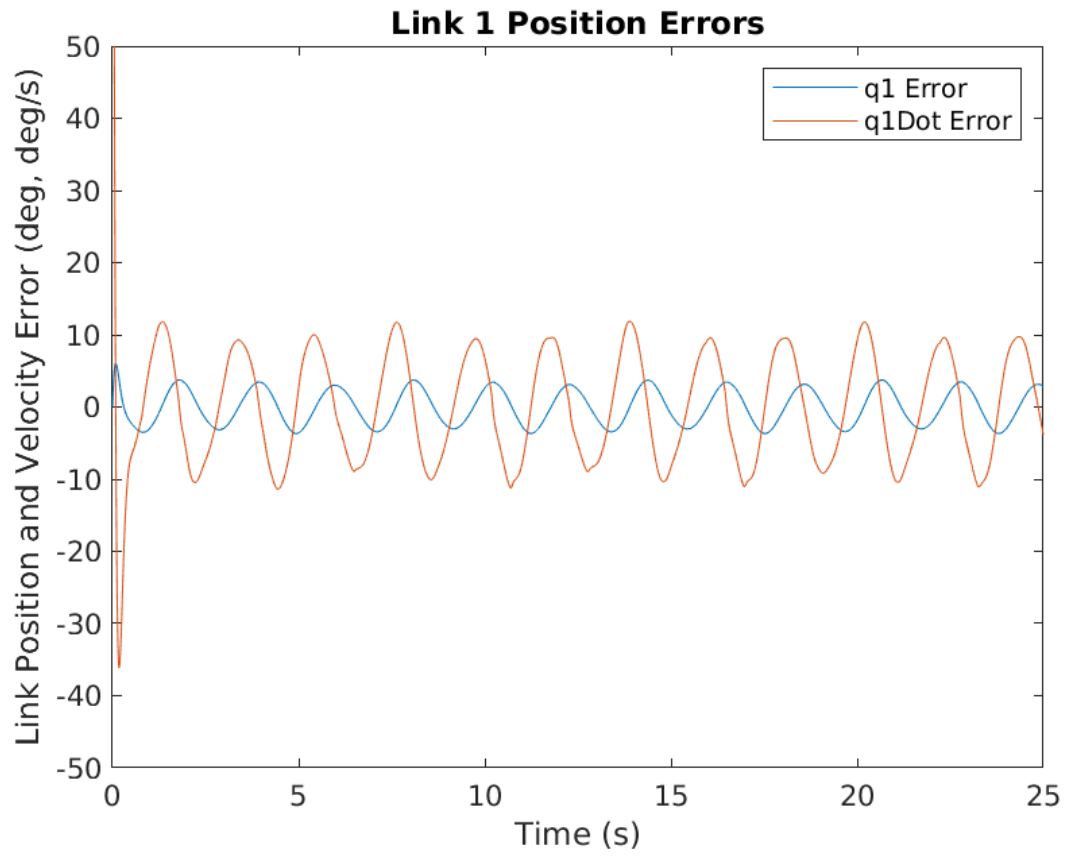
- **neural network-based controller with a typical continuous feedback control law for the dynamics**

**1. Control gains.**

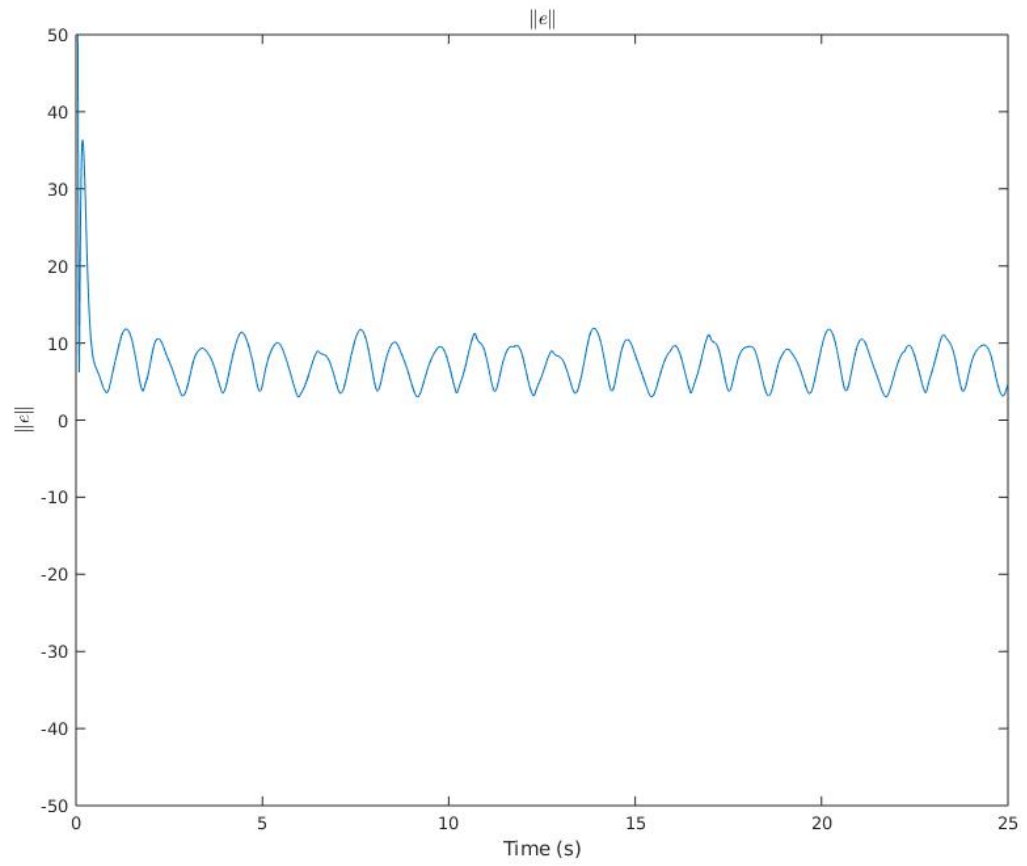
Control gains:  $k = 50, k_n = 10$ ; and  $\alpha = 10$  .

Neural Network learning rate:  $\gamma_1 = 5, \gamma_2 = 10$  .

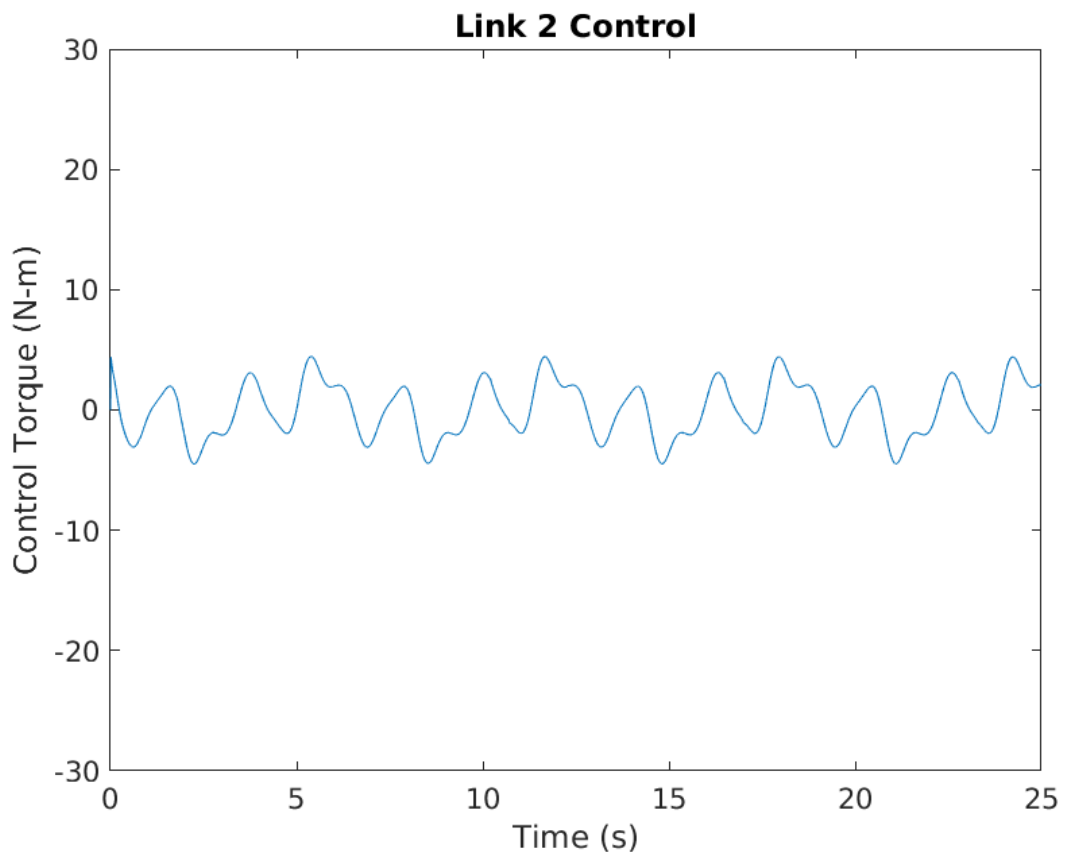
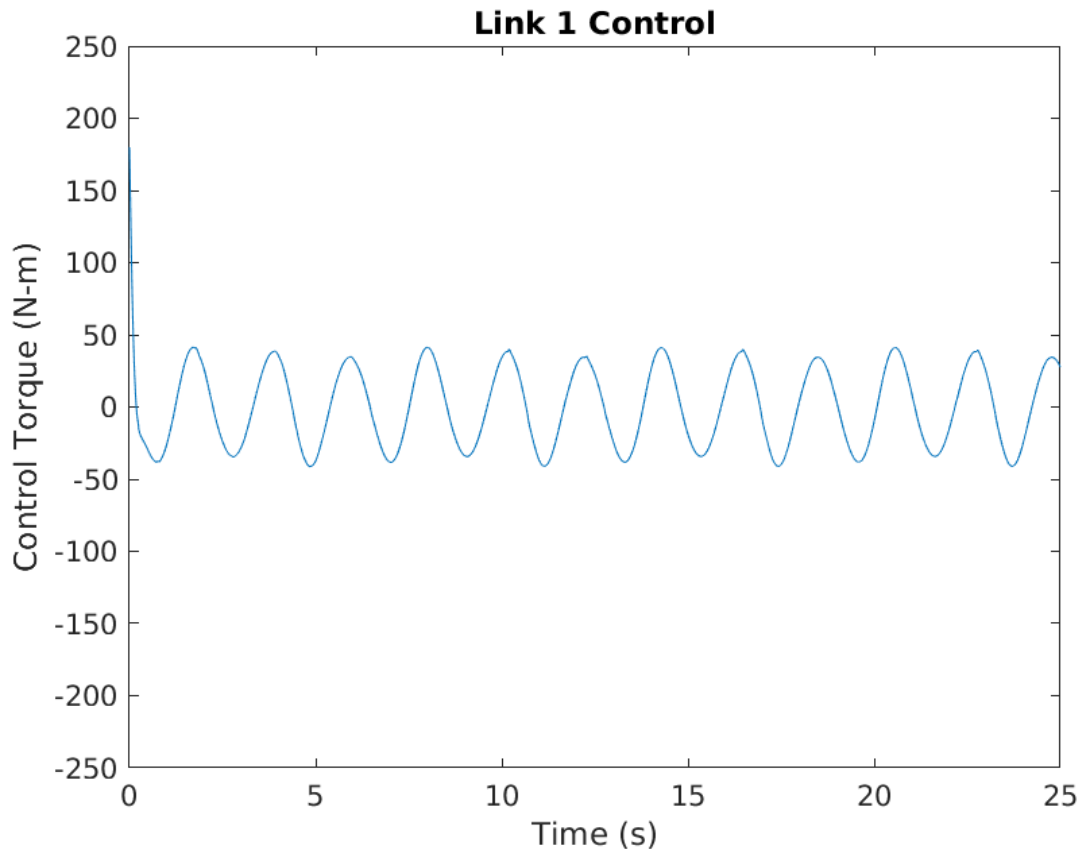
**2. Tracking error plot for each link.**



Root Sum Squared of error is as below:

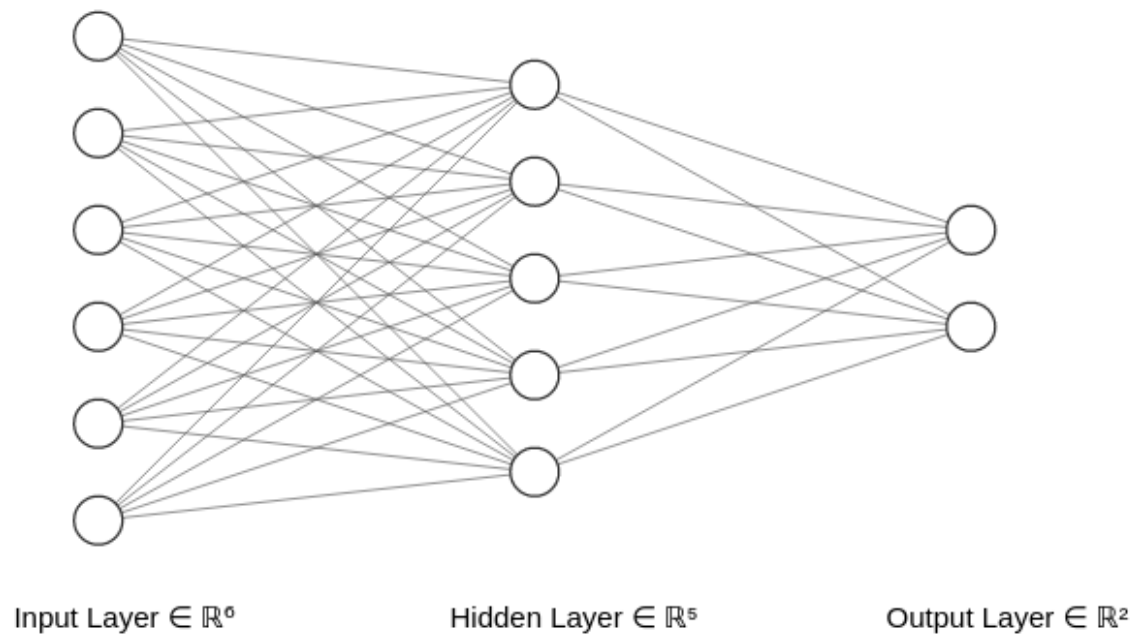


**3. Control input plot for each link.**



#### 4. Plot of the adaptive estimates.

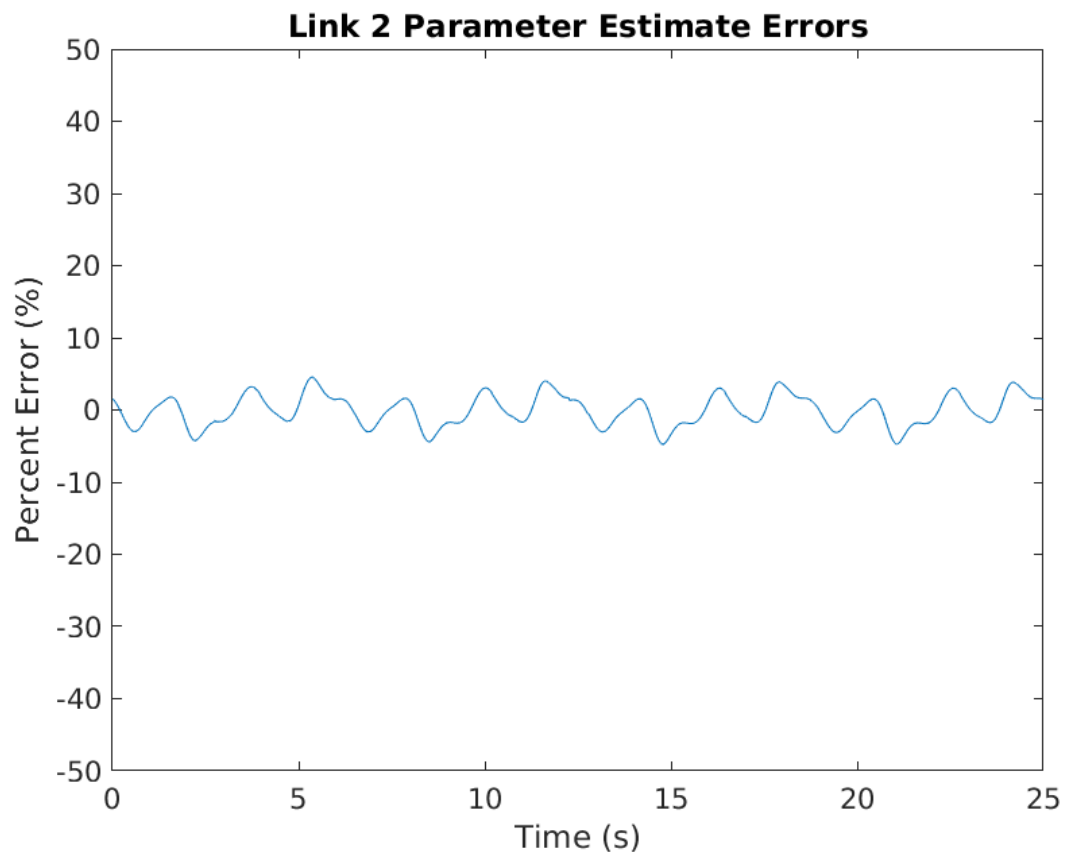
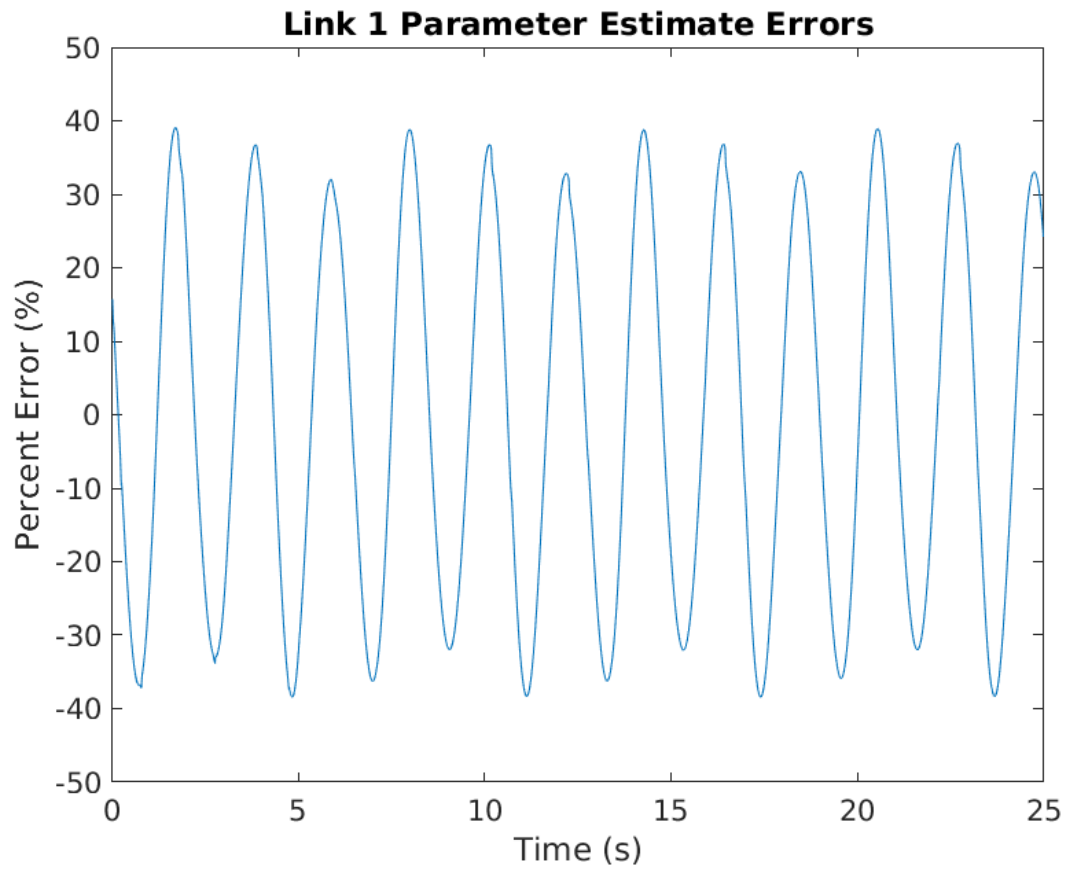
In this project, we don't explicitly estimate the parameters in dynamics. We use a FCNN architecture to estimate the dynamics in the system as shown below:



So we don't have a meaningful parameter estimates plot.

### **5. Plot of the parameter estimate errors.**

In this project, we plot the errors between the FCNN output and the ground truth.



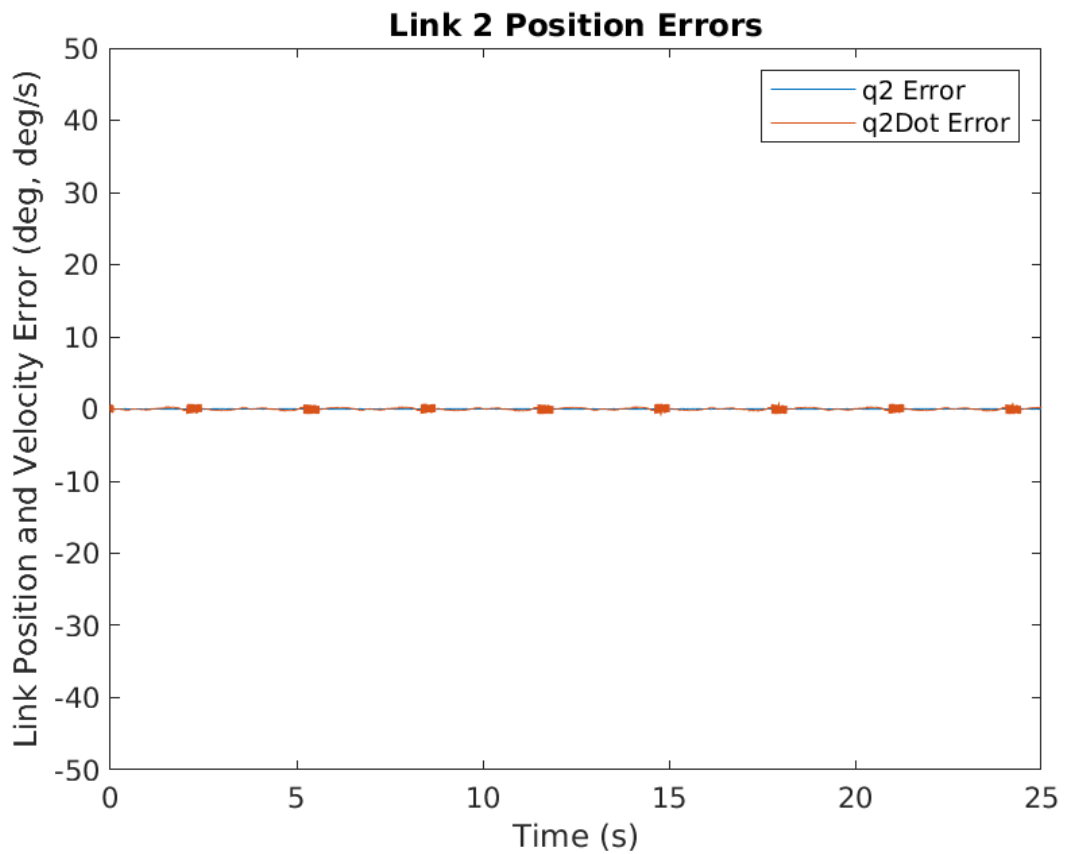
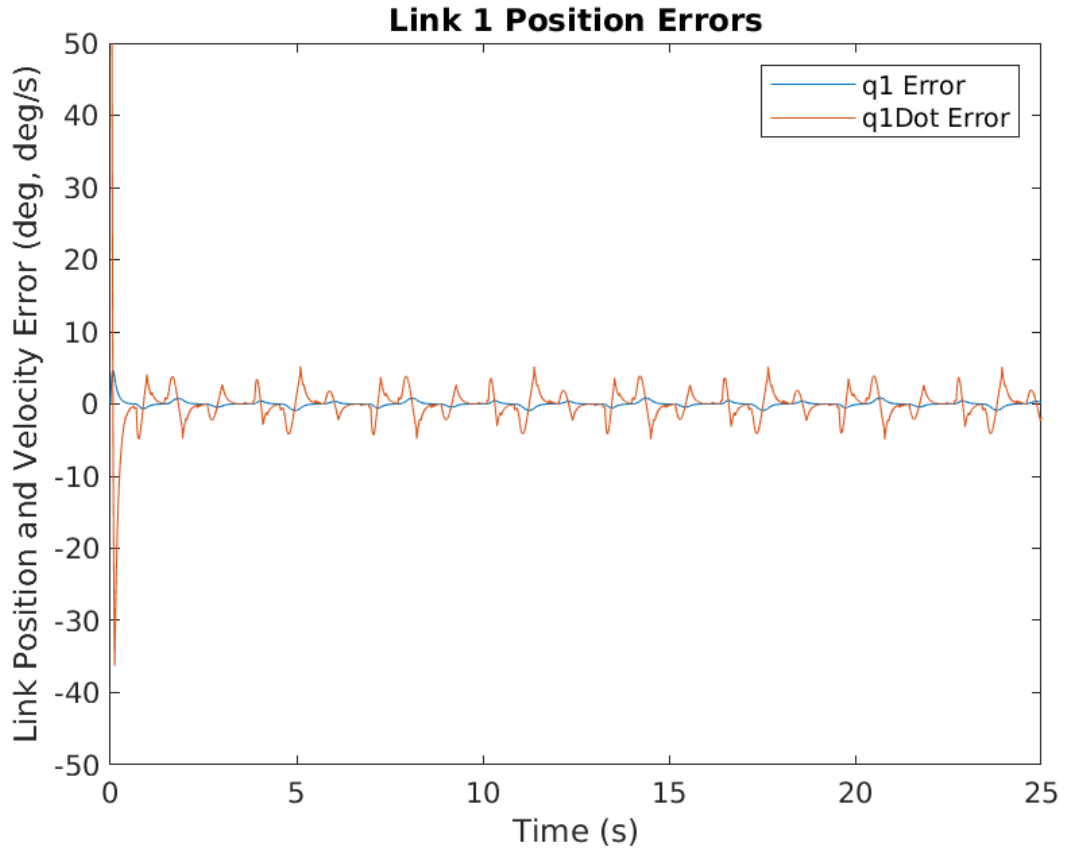
- **neural network-based controller with a discontinuous sliding-mode feedback control law for the dynamics**

## 1. Control gains.

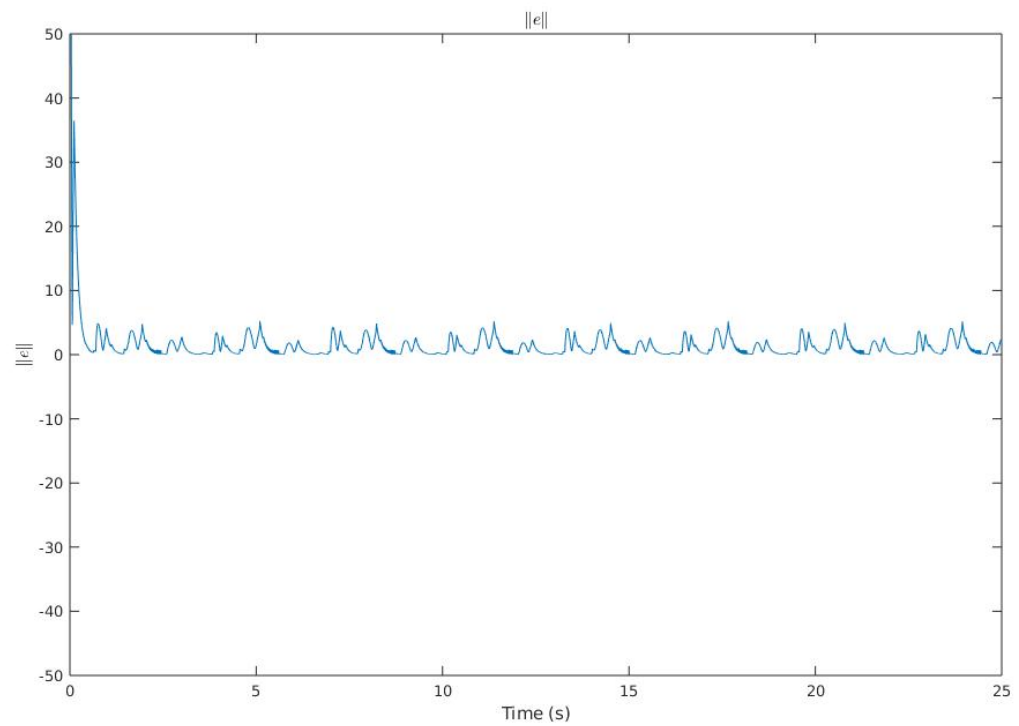
Control gains:  $k = 50, k_n = 20, k_s = 28$  and  $\alpha = 10$  .

Neural Network learning rate:  $\gamma_1 = 5, \gamma_2 = 10$  .

## 2. Tracking error plot for each link.

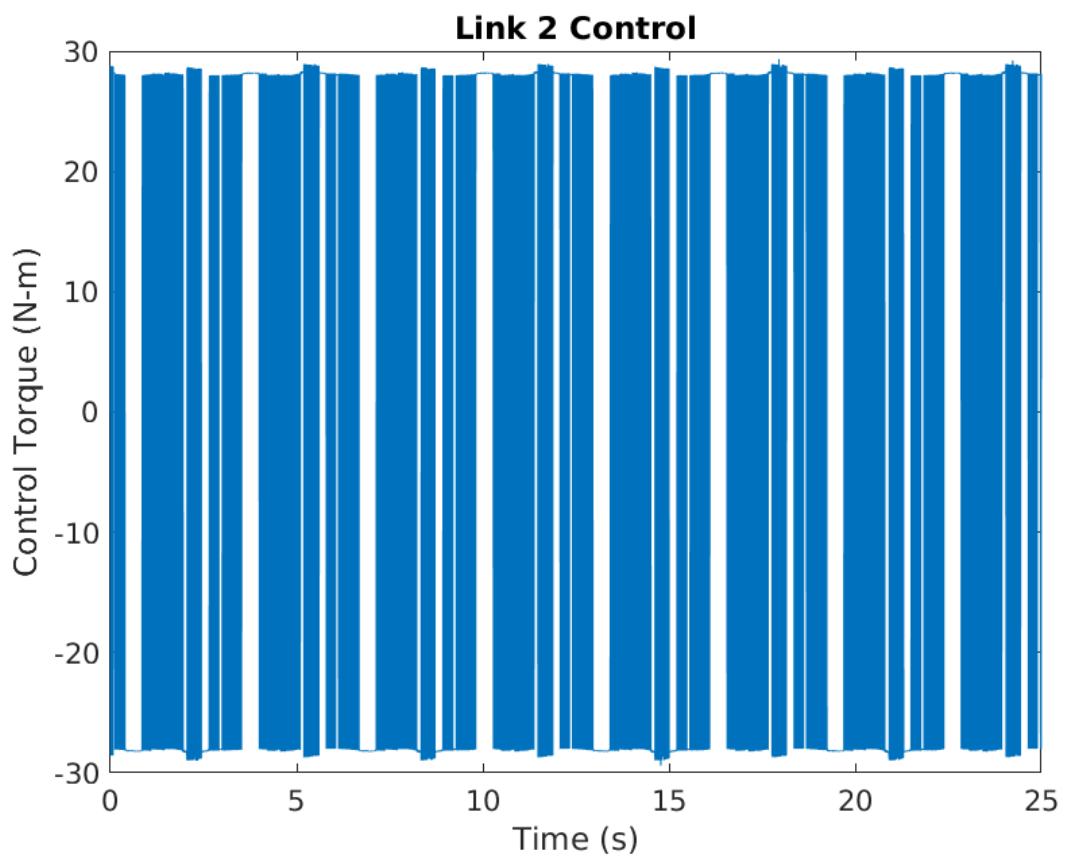
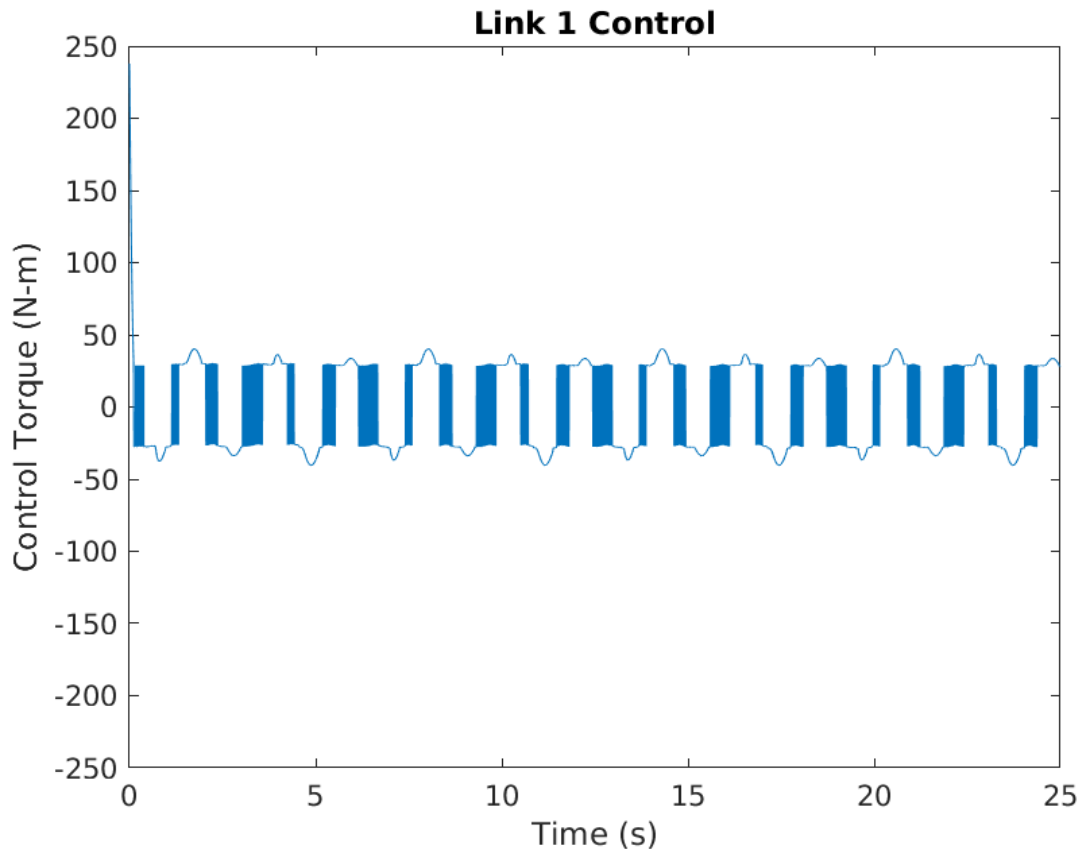


Root Sum Squared of error is as below:



**3. Control input plot for each link.**



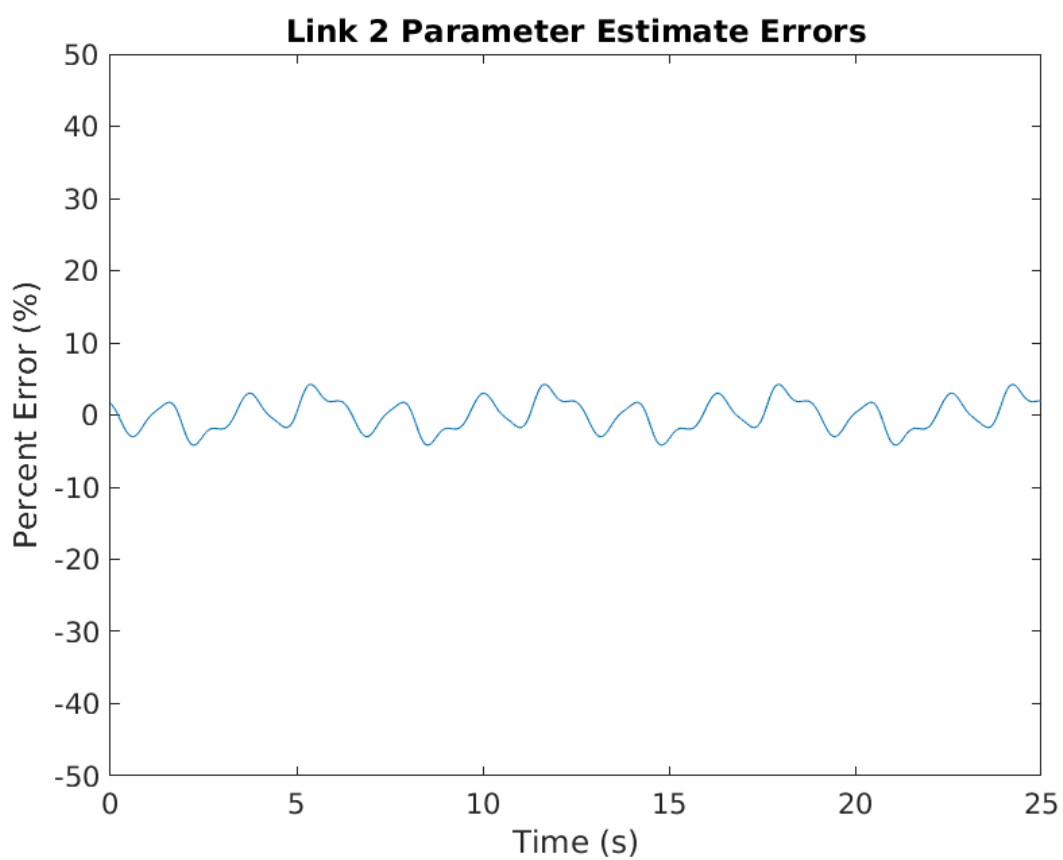
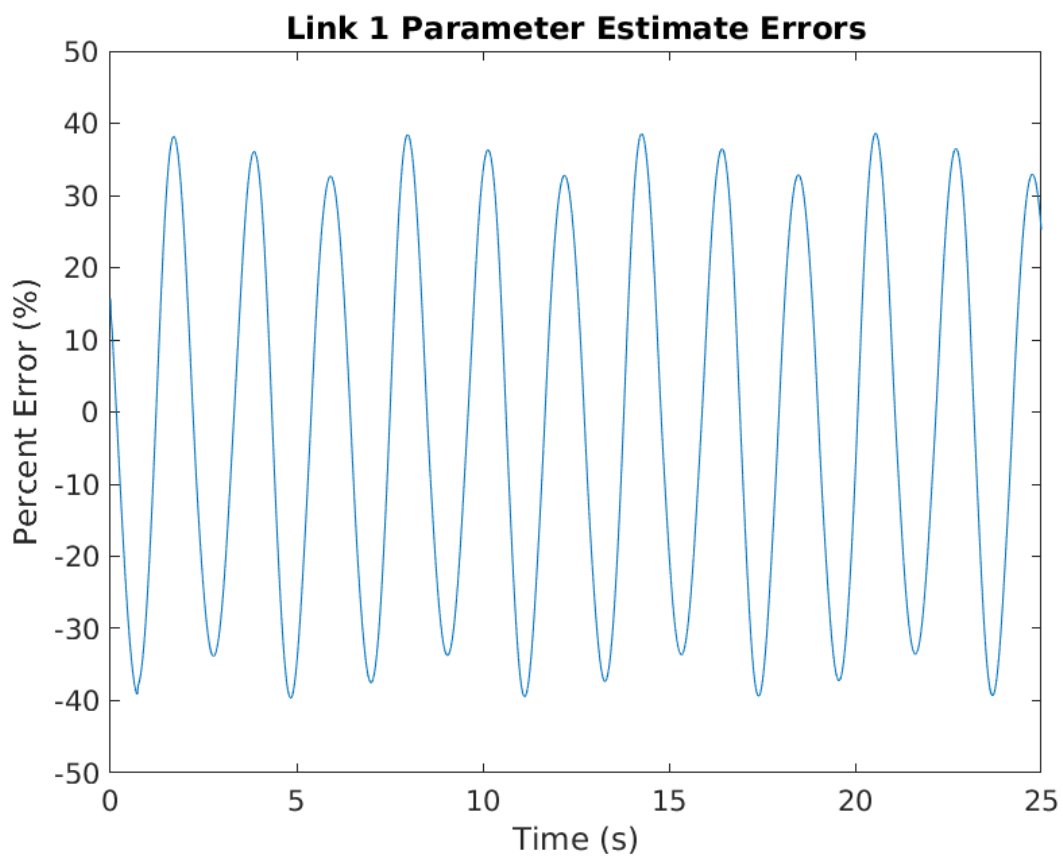


#### 4. Plot of the adaptive estimates.

We don't have a meaningful parameter estimates plot as shown above.

#### 5. Plot of the parameter estimate errors.

In this project, we plot the errors between the FCNN output and the ground truth.



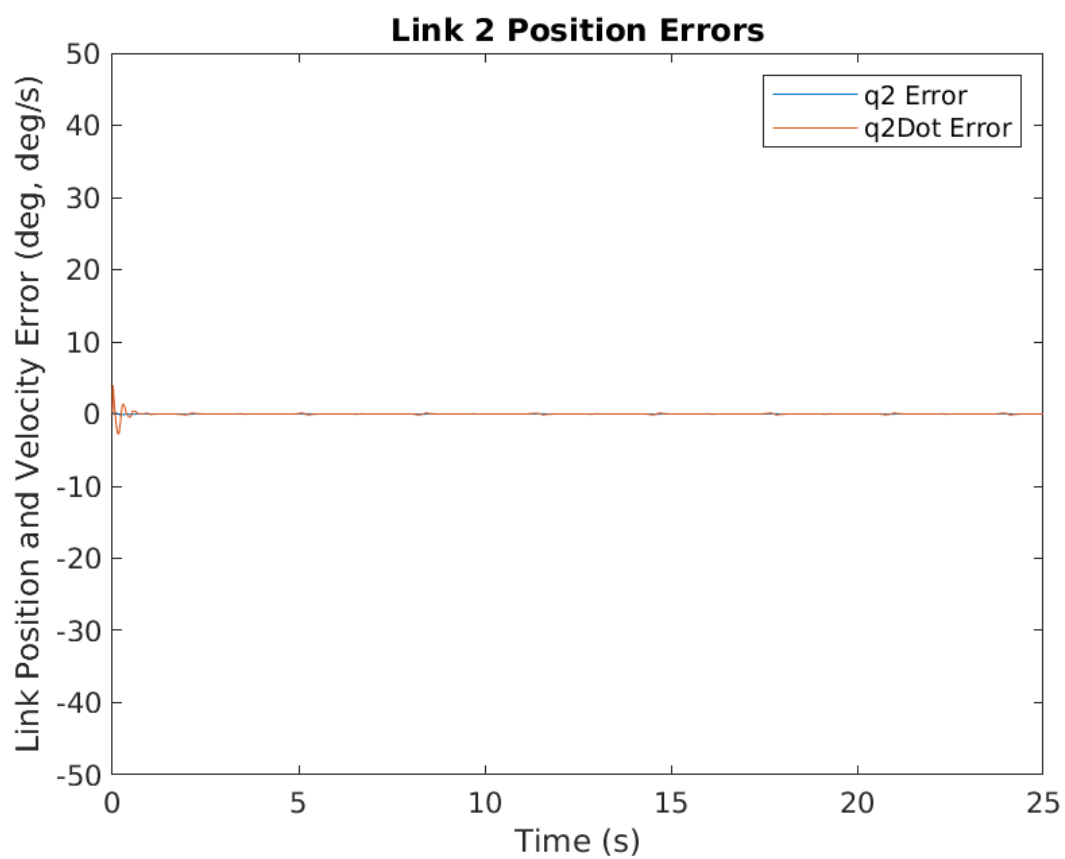
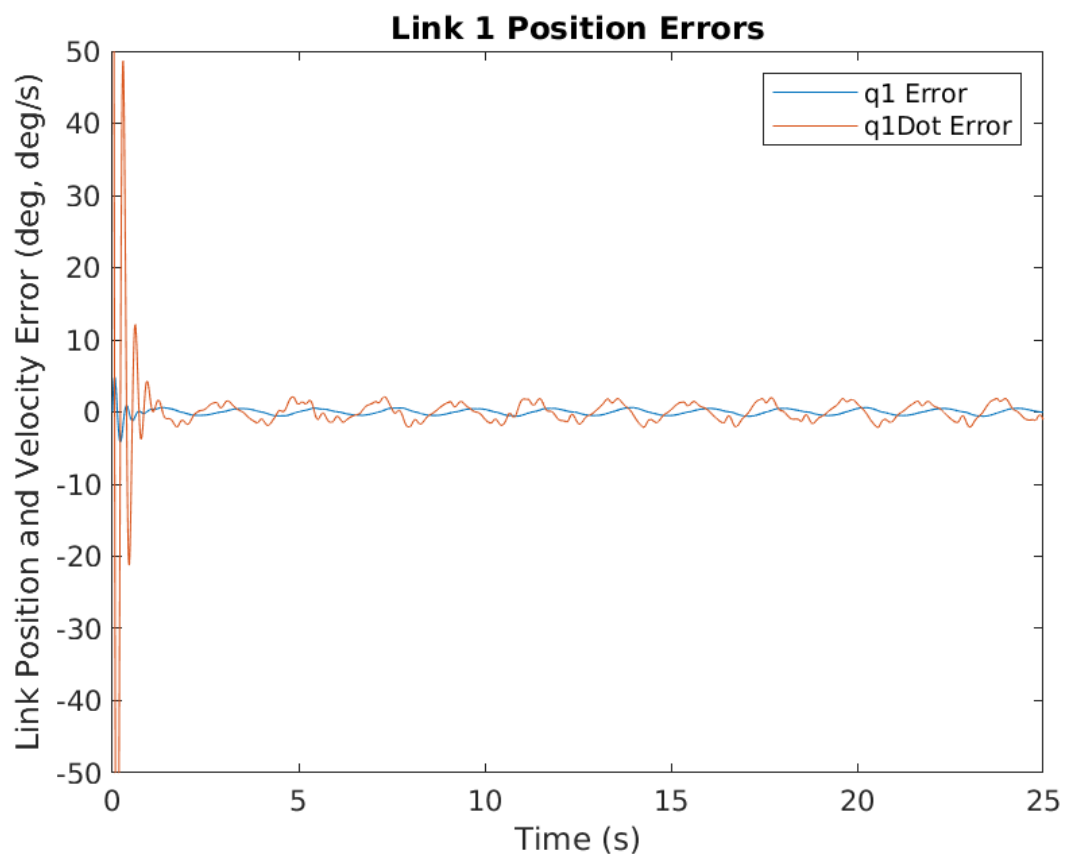
- **neural network-based controller with a continuous RISE feed-back control law for the dynamics**

**1. Control gains.**

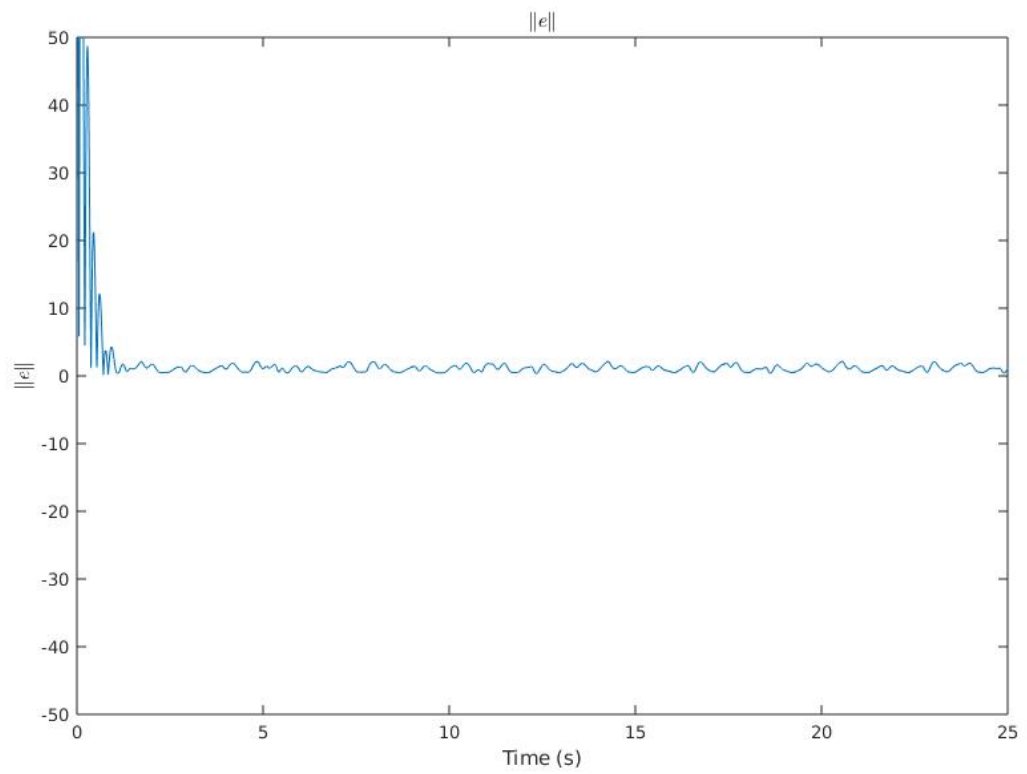
Control gains:  $k_s = 50$  and  $\alpha_1 = 10, \alpha_2 = 20, \beta_1 = 10$  .

Neural Network learning rate:  $\gamma_1 = 5, \gamma_2 = 10$  .

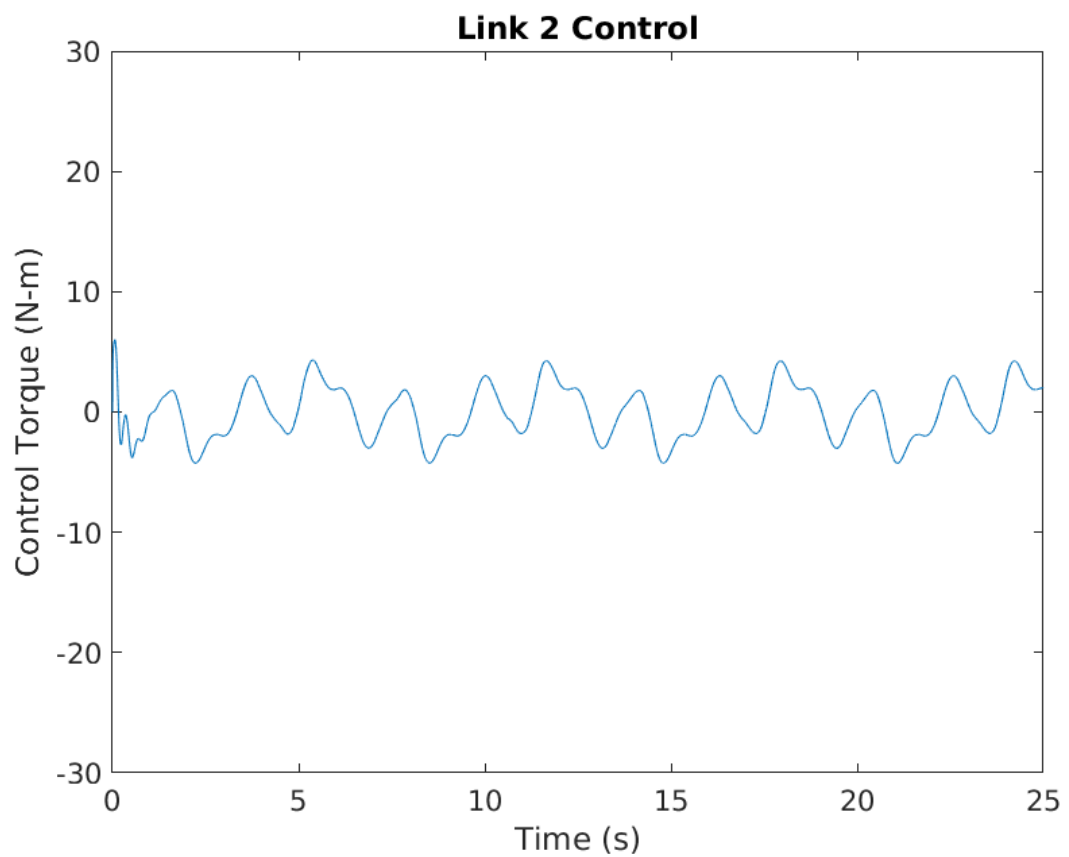
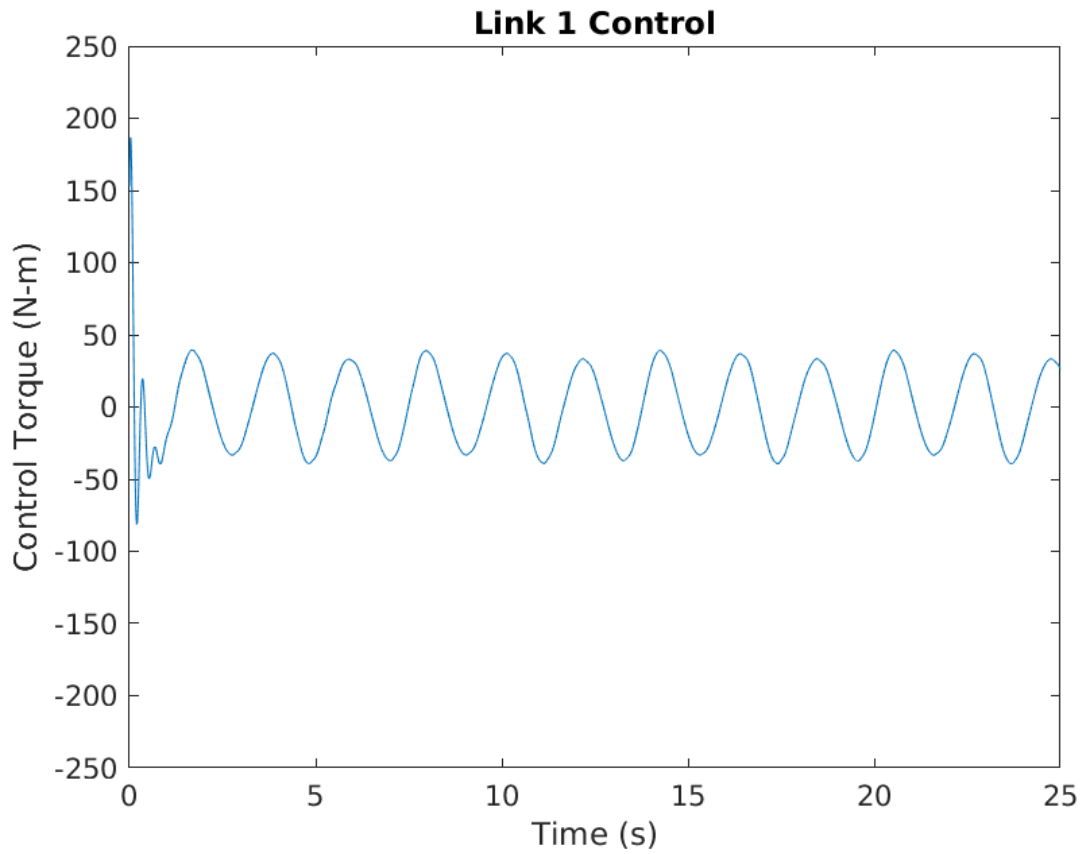
**2. Tracking error plot for each link.**



Root Sum Squared of error is as below:



**3. Control input plot for each link.**

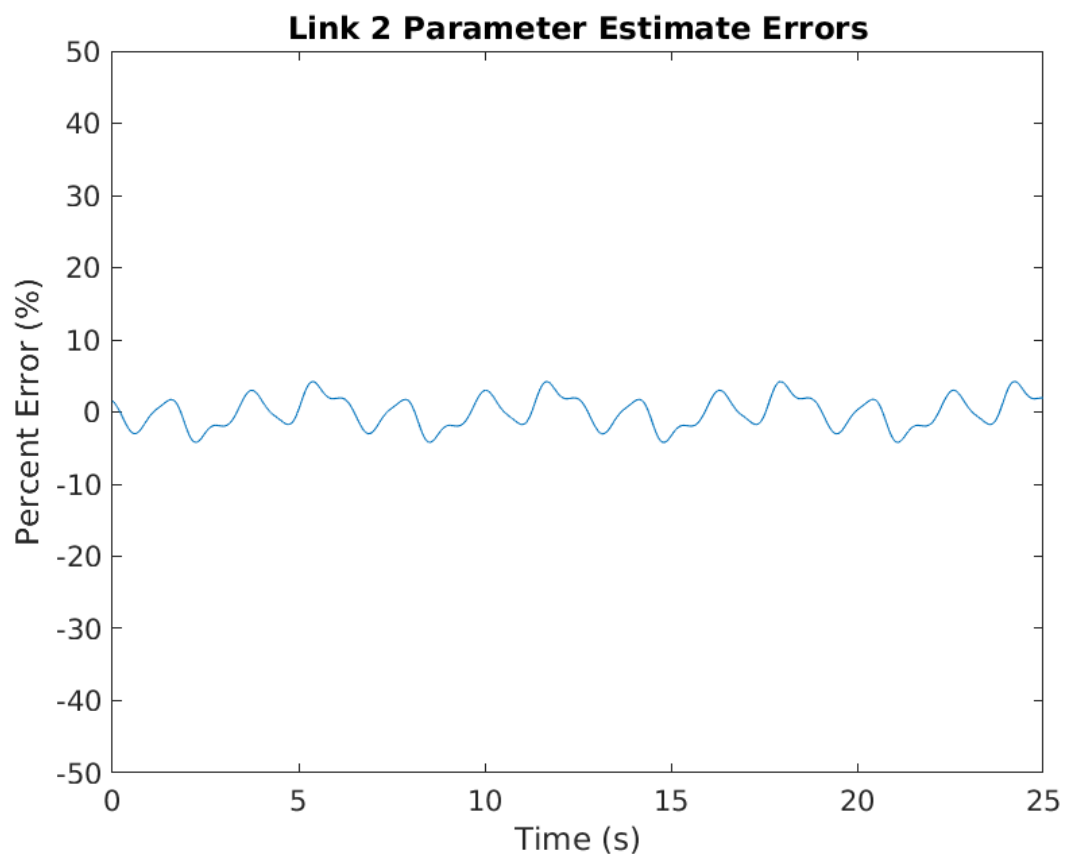
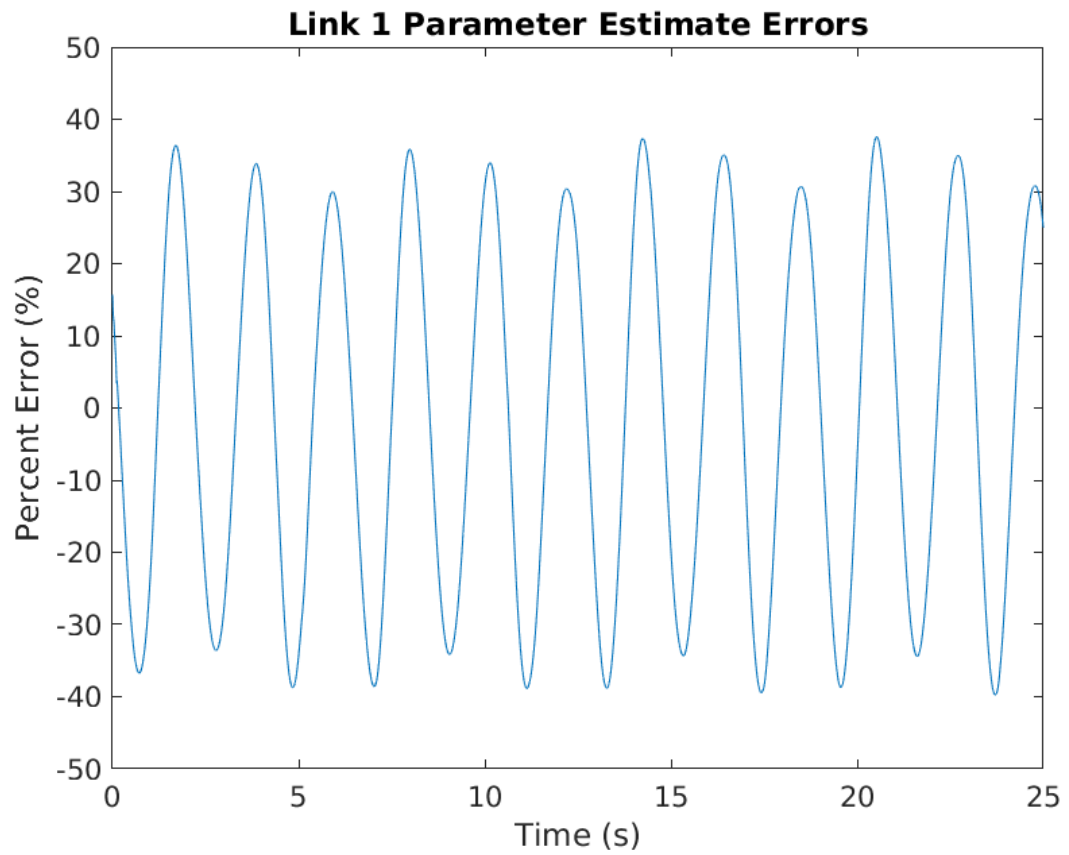


#### 4. Plot of the adaptive estimates.

We don't have a meaningful parameter estimates plot as shown above.

#### 5. Plot of the parameter estimate errors.

In this project, we plot the errors between the FCNN output and the ground truth.

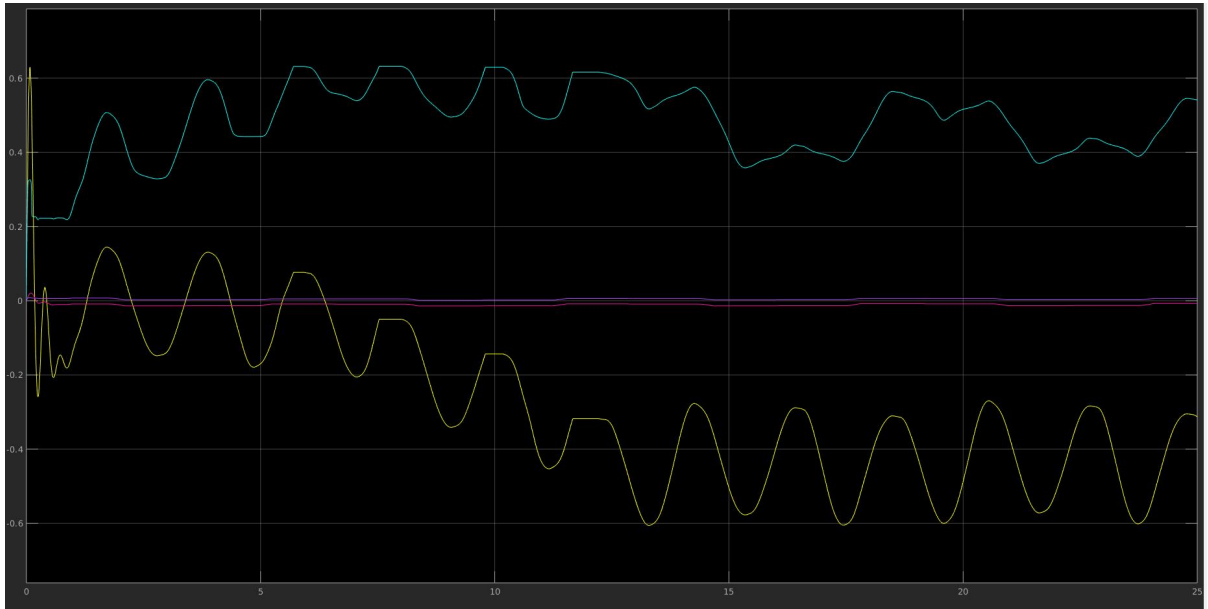


- **(B) Discussion section**

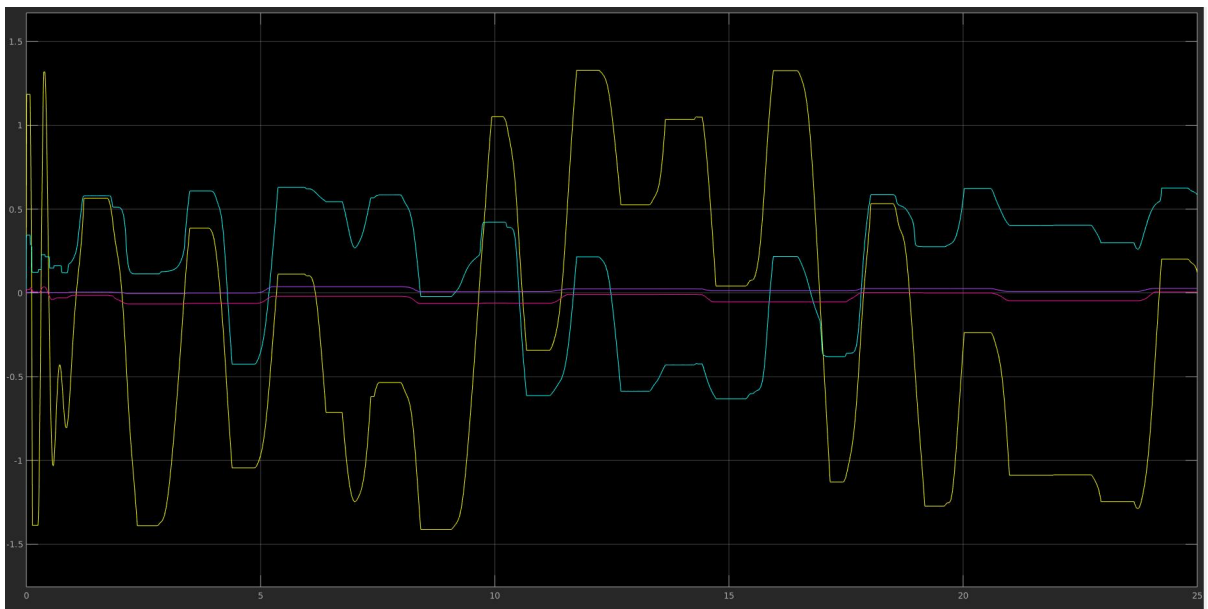
### ◦ Differences in tuning the control gains/adaptations

For  $\gamma$ , it control the derivate of  $\hat{W}$  and  $\hat{V}$ . However, after twiddling  $\gamma$ , I found that they don't effect the performance of my controller. I think the reason may be that the feedback part in the controller contribute the most effort, and it suppress the feedforward Neural Network part.  $\hat{W}$  in 2 different sets of  $\gamma$  are shown below:

The first one is when  $\gamma_1 = 5, \gamma_2 = 10$  :



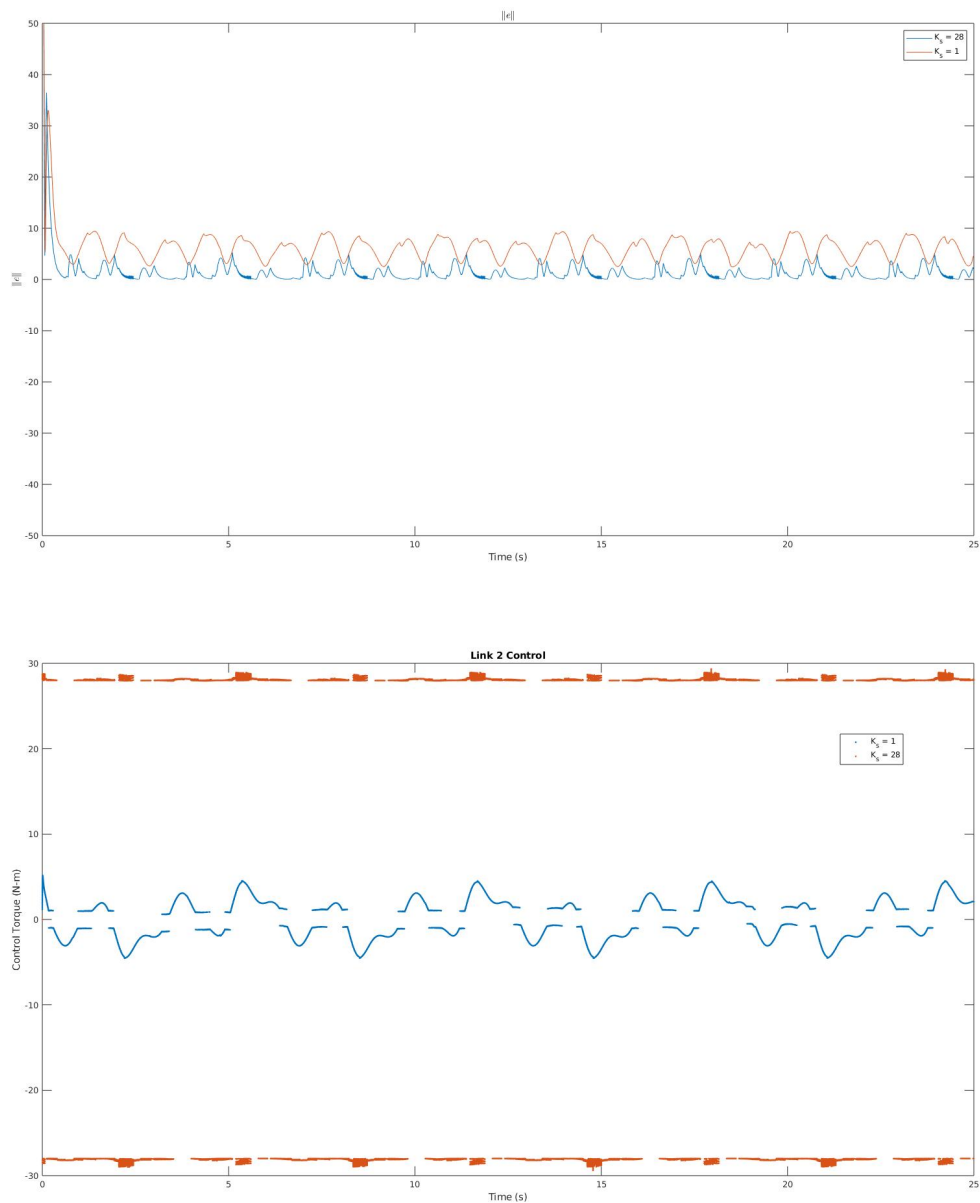
The second one is when  $\gamma_1 = 50, \gamma_2 = 100$  :



The saturation above is because we are using projection method, to prevent them being too large.

For  $k_s$ , it is contained both in the Sliding mode term and RISE term, and it is essential in these two implementation. In both cases, only when  $k_s$  is big enough, we can conclude the asymptomatic tracking in the lyapunov analysis. But, when it become large, the control input would be oscilated and increased. I will show Sliding mode controller as examples.



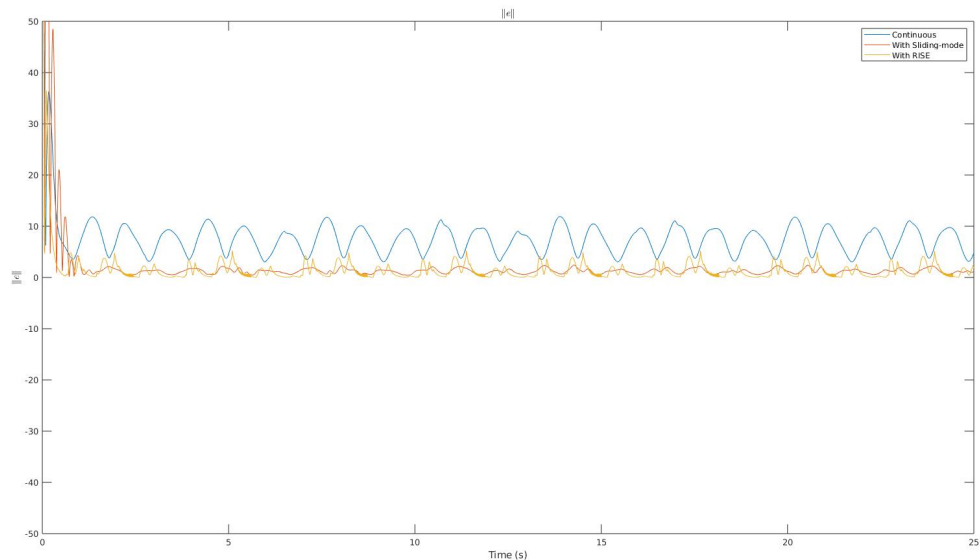


For  $k, k_n$ , it is contained both in our first continuous controller and the discontinuous controller (with Sliding mode control). We can sum them up and see them as a constant, they contribute a state squared part in the lyapunov analysis, and it's even more essential than the  $k_s$ . It determine if our tracking error would blow up. When we increase it, the tracking error would decrease faster, but we would increase the input effort as trade-off.

For  $\alpha$ , it is demanded to be large enough in the Neural Netrok-based controller with RISE term. In the other two controllers, it can improve the convergence of the tracking error, but it also comes with a price as the input effort would be increased.

For  $\beta_1$  in the Neural Netrok-based controller with RISE term, we knew that it should be big enough to conclude the Semi-GAT result, however, after I twiddling it, it seems it doesn't make a explicit effect on the performamnce of the controller.

- **Performance of the tracking error for each controller**



In the comparison above, we can know that in term of performance of the tracking error, the first controller with a typical continuous feedback control law behave the worst, and the other two has the similar performance. The neural network-based controller with a discontinuous sliding-mode feedback control law for the dynamics is slightly better than the neural network-based controller with a continuous RISE. But in my implementation, I use a fixed-step solver to deal with the discontinuous sliding-mode feedback, so it tends be inaccurate

- **Performance of the adaptation for each case**

The parameter estimate errors are almost the same in these three cases, but I don't think it's meaningful, because we have no feedback during the Neural-Network update, We use unsupervised learning instead. In such case, we don't have to do symmetry breaking and back propagation. I don't think we are doing meaningful estimation, because the feedback term has contributed the main effort during the error tracking. It seems to me that the FCNN is meaningless especially we are enforcing projection method on the neural network.

- **Control effort for each case**

In terms of control effort, the neural network-based controller with a discontinuous sliding-mode feedback control law for the dynamics behave the worst. So, I think the neural network-based controller with a continuous RISE feed-back control law for the dynamics is the best controller among the three.