SINGLE AND TWO LAYER NEURAL NETWORK

DYNAMICS –

Consider the two-link dynamic system

$$M(\emptyset)\ddot{\emptyset} + C(\emptyset,\dot{\emptyset}) + G(\emptyset) + \tau_d(\emptyset,\dot{\emptyset}) = \tau$$

Where $\emptyset \triangleq \begin{bmatrix}\emptyset_1\\ \emptyset_2\end{bmatrix}, \dot{\emptyset} \triangleq \begin{bmatrix}\dot{\emptyset}_1\\ \dot{\emptyset}_2\end{bmatrix},\ \ddot{\emptyset} \triangleq \begin{bmatrix}\ddot{\emptyset}_1\\ \ddot{\emptyset}_2\end{bmatrix}, \tau \triangleq \begin{bmatrix}\tau_1\\ \tau_2\end{bmatrix} \epsilon\ \mathbb{R}^2$ are the angle, angular velocity, angular acceleration, and input torque of the arm joints.

$$M(\emptyset) \triangleq \begin{bmatrix}\left(m_1 l_1{}^2 + m_2\left(l^2 + 2l_1 l_2 c_2 + l_2{}^2\right)\right) & m_2\left(l_1 l_2 c_2 + l_2{}^2\right)\\ m_2\left(l_1 l_2 c_2 + l_2{}^2\right) & m_2 l_2{}^2\end{bmatrix} \epsilon\ \mathbb{R}^{2\times 2}$$

$$C(\emptyset,\dot{\emptyset}) = \begin{bmatrix}-2m_2 l_1 l_2 s_2 \dot{\emptyset}_1\dot{\emptyset}_2\\ 0\end{bmatrix} + \begin{bmatrix}-m_2 l_1 l_2 s_2 \dot{\emptyset}_2{}^2\\ m_2 l_1 l_2 s_2 \dot{\emptyset}_1{}^2\end{bmatrix} \epsilon\ \mathbb{R}^2,$$

$$G(\emptyset) = \begin{bmatrix}(m_1 + m_2) g l_1 c_1 + m_2 g l_2 c_{12}\\ m_2 g l_2 c_{12}\end{bmatrix} \epsilon\ \mathbb{R}^2,$$

where $m_i, l_i\ \epsilon\ \mathbb{R} > 0$ are the unknow constant mass and length of the link i and gravity is g = 9.8 m/s^2 and,

$$c_i = \cos(\emptyset_i),$$
$$s_i = \sin(\emptyset_i),$$
$$c_{12} = \cos(\emptyset_1 + \emptyset_2),$$
$$s_{12} = \sin(\emptyset_1 + \emptyset_2),$$

And $\tau_d(\emptyset,\dot{\emptyset})\ \epsilon\ \mathbb{R}^2$ is the unknown, unstructured state-dependent disturbance that is a function of the angles and angular velocity.

Assuming that $\underline{m} \le m_i \le \overline{m}, \underline{l} \le l_i \le \bar{l}$ are known bounds on the unknown mass and length of link i, with $m_i = 2kg, l_i = 0.5m, \underline{m} = 1kg, \overline{m} = 3kg, \underline{l} = 0.25m, \bar{l} = 0.25m$. Also, assuming that $\emptyset, \dot{\emptyset}, \ddot{\emptyset}$ are all measurable.

Let the desired trajectory have a magnitude of $\overline{\emptyset_d} = \begin{bmatrix}\emptyset_{d1}\\ \emptyset_{d2}\end{bmatrix} = \begin{bmatrix}\frac{\pi}{8}\\ \frac{\pi}{4}\end{bmatrix}$, at a frequency of $f_{\emptyset d} \triangleq 0.2Hz$, a phase shift $a_{\emptyset d} \triangleq \frac{\pi}{2}$, and bias of $b_{\emptyset d} \triangleq \begin{bmatrix}\frac{\pi}{2}\\ \frac{\pi}{4}\end{bmatrix}$, such that $\begin{bmatrix}\emptyset_{d1}(t)\\ \emptyset_{d2}(t)\end{bmatrix} \triangleq \begin{bmatrix}\overline{\emptyset_{d1}}\sin(2\pi f_{\emptyset d}t - a_{\emptyset d}) - b_{\emptyset d}\\ \overline{\emptyset_{d2}}\sin(2\pi f_{\emptyset d}t - a_{\emptyset d}) + b_{\emptyset d}\end{bmatrix}$ which implies $\begin{bmatrix}\dot{\emptyset}_{d1}(t)\\ \dot{\emptyset}_{d2}(t)\end{bmatrix} \triangleq \begin{bmatrix}\overline{\emptyset}_{d1}(2\pi f_{\emptyset d})\cos(2\pi f_{\emptyset d}t - a_{\emptyset d})\\ \overline{\emptyset}_{d2}(2\pi f_{\emptyset d})\cos(2\pi f_{\emptyset d}t - a_{\emptyset d})\end{bmatrix}$ and $\begin{bmatrix}\ddot{\emptyset}_{d1}(t)\\ \ddot{\emptyset}_{d2}(t)\end{bmatrix} \triangleq \begin{bmatrix}-(2\pi f_{\emptyset d})^2\overline{\emptyset_{d1}}\sin(2\pi f_{\emptyset d}t - a_{\emptyset d})\\ -(2\pi f_{\emptyset d})^2\overline{\emptyset_{d2}}\sin(2\pi f_{\emptyset d}t - a_{\emptyset d})\end{bmatrix}$.

Designing a controller under the assumption that the structure is completely unknown for $\tau_d(\emptyset,\dot{\emptyset})$

From the dynamics,

$$M(\emptyset)\ddot{\emptyset} + C(\emptyset,\dot{\emptyset}) + G(\emptyset) + \tau_d(\emptyset,\dot{\emptyset}) = \tau$$

$M(\emptyset)\ddot{\emptyset} + C(\emptyset,\dot{\emptyset}) + G(\emptyset)$ accounts to the structured uncertainties and $\tau_d(\emptyset,\dot{\emptyset})$ represents the unstructured uncertainties. And as a result, we can design a linear in the parameter design for the structured uncertainties. And the unstructured uncertainties can be estimated using deep neural network.

For deep neural network $\tau_d(\emptyset,\dot{\emptyset})$ can be estimated as

$$\tau_d(\emptyset,\dot{\emptyset}) = W^T\sigma\big(\Phi(\emptyset,\dot{\emptyset}) + \varepsilon(\emptyset,\dot{\emptyset}\big)$$

Where $W\epsilon\ \mathbb{R}^{L\times 2}$ is the output weights, $\sigma(.)\epsilon\ \mathbb{R}^{L}$ is the activation function, $\Phi$is the inner and the reconstruction error is $\varepsilon(\emptyset,\dot{\emptyset})\ \epsilon\ \mathbb{R}^2$.

$$\widehat{\Phi}_{lk} = \sigma_l(\hat{V}_{lk}^T\hat{\phi}_{l-1\,k} + \hat{b}_{lk})$$

Error Dynamics-

$$e = \emptyset_d - \emptyset$$
$$\dot{e} = \dot{\emptyset}_d - \dot{\emptyset}$$
$$\ddot{e} = \ddot{\emptyset}_d - \ddot{\emptyset}$$

Reference tracking error

$$r = \dot{e} + \alpha e$$
$$\dot{r} = \ddot{e} + \alpha\dot{e}$$
$$\dot{r} = \ddot{\emptyset}_d - \ddot{\emptyset} + \alpha\dot{e}$$

Multiplying both sides by $M(\emptyset)$ we get,

$$M(\emptyset)\dot{r} = M(\emptyset)\ddot{\emptyset}_d - M(\emptyset)\ddot{\emptyset} + M(\emptyset)\alpha\dot{e}$$

From the dynamics we know,

$$M(\emptyset)\ddot{\emptyset} = -C(\emptyset,\dot{\emptyset}) - G(\emptyset) - \tau_d(\emptyset,\dot{\emptyset}) + \tau$$

On substitution we get,

$$M(\emptyset)\dot{r} = M(\emptyset)\ddot{\emptyset}_d - M(\emptyset)\ddot{\emptyset} + M(\emptyset)\alpha\dot{e}$$
$$M(\dot{\emptyset})r = M(\emptyset)\ddot{\emptyset}_d - (-C(\emptyset,\dot{\emptyset}) - G(\emptyset) - \tau_d(\emptyset,\dot{\emptyset}) + \tau) + M(\emptyset)\alpha\dot{e}$$

Add and subtract by $\frac{1}{2}\dot{M}(\emptyset,\dot{\emptyset})r$

$$M(\emptyset)\dot{r} = M(\emptyset)\ddot{\emptyset}_d + C(\emptyset,\dot{\emptyset}) + G(\emptyset) + \tau_d(\emptyset,\dot{\emptyset}) - \tau + M(\emptyset)\alpha\dot{e} \pm \frac{1}{2}\dot{M}(\emptyset,\dot{\emptyset})r$$
$$M(\emptyset)\dot{r} = M(\emptyset)(\ddot{\emptyset}_d + \alpha\dot{e}) + C(\emptyset,\dot{\emptyset}) + G(\emptyset) + \tau_d(\emptyset,\dot{\emptyset}) - \tau \pm \frac{1}{2}\dot{M}(\emptyset,\dot{\emptyset})r$$

We can estimate $M(\emptyset)\ (\ddot{\emptyset}_d + \alpha\dot{e}) + C(\emptyset,\dot{\emptyset}) + G(\emptyset) + \frac{1}{2}\dot{M}(\emptyset,\dot{\emptyset})r = Y\theta$ since it is linear in the unknown parameters and we were able to develop the unknown parameters θ as follows,

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \theta_5\end{bmatrix} = \begin{bmatrix} m_1{l_1}^2 + m_2{l_1}^2 + m_2{l_2}^2 \\ m_2 l_1 l_2 \\ m_2{l_1}^2 \\ (m_1+m_2)l_1 \\ m_2 l_2 \end{bmatrix}$$

And $Y = Y_M(\emptyset, \ddot{\emptyset}_d + \alpha\dot{e}) + Y_c(\emptyset, \dot{\emptyset}) + Y_G(\emptyset) + \frac{1}{2}Y_{\dot{M}}(\emptyset, \dot{\emptyset}, r)$

The unstructured state-dependent disturbance $\tau_d(\emptyset, \dot{\emptyset})$ will be estimated using a two-layer neural network.

$$\tau_d(\emptyset, \dot{\emptyset}) = W^T\sigma(\Phi(\zeta) + \varepsilon(\emptyset, \dot{\emptyset})$$

Now,

$$M(\emptyset)\dot{r} = Y\theta + W^T\sigma(\Phi(\zeta)) + \varepsilon(\emptyset, \dot{\emptyset}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r$$

Approximation for

$$\sigma(\Phi) = \sigma(\widehat{\Phi}) + \frac{d\sigma}{d\Phi}\widetilde{\Phi} + \varepsilon_\sigma(\widetilde{\Phi}^2)$$

Where,

$\widetilde{\Phi} = \Phi - \widehat{\Phi} = V^T\zeta - \hat{V}^T\zeta = \tilde{V}\zeta$

$$\sigma = \sigma + \frac{d\sigma}{d\Phi}\widetilde{\Phi} + \varepsilon_\sigma$$

Now on substitution we get,

$$M(\emptyset)\dot{r} = Y\theta + W^T(\sigma(\widehat{\Phi}) + \frac{d\sigma}{d\Phi}\widetilde{\Phi} + \varepsilon_\sigma(\widetilde{\Phi}^2)) + \varepsilon(\emptyset, \dot{\emptyset}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r$$

Add and subtract $\widehat{W}^T\hat{\sigma}'\widetilde{\Phi}$

$$M(\emptyset)\dot{r} = Y\theta + W^T\left(\sigma(\widehat{\Phi}) + \frac{d\sigma}{d\Phi}\widetilde{\Phi} + \varepsilon_\sigma(\widetilde{\Phi}^2)\right) + \varepsilon(\emptyset, \dot{\emptyset}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r \pm \widehat{W}^T\hat{\sigma}'\widetilde{\Phi}$$

$$M(\emptyset)\dot{r} = Y\theta + W^T\sigma(\widehat{\Phi}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r + \widehat{W}^T\hat{\sigma}'\widetilde{\Phi} + \widetilde{W}^T\hat{\sigma}'\widetilde{\Phi} + W^T\varepsilon_\sigma(\widetilde{\Phi}^2) + \varepsilon(\emptyset, \dot{\emptyset})$$

Let

$$\widetilde{W}^T\hat{\sigma}'\widetilde{\Phi} + W^T\varepsilon_\sigma(\widetilde{\Phi}^2) + \varepsilon(\emptyset, \dot{\emptyset}) = \delta$$

$$M(\emptyset)\dot{r} = Y\theta + W^T\sigma(\widehat{\Phi}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r$$

Let the stacked errors

$$\xi = \begin{bmatrix} e \\ r \\ \tilde{\theta} \\ vec(\widetilde{W}) \end{bmatrix} \epsilon\ \mathbb{R}^{2+2+5+2L+5l}$$

Where,

$$vec(\widetilde{W}) = \begin{bmatrix} \widetilde{W_{11}} \\ \vdots \\ \widetilde{W_{L1}} \\ \widetilde{W_{12}} \\ \vdots \\ \widetilde{W_{L2}} \end{bmatrix}$$

Lyapunov Candidate,

$$V = \frac{1}{2}e^Te + \frac{1}{2}r^TM(\emptyset)r + \frac{1}{2}\tilde{\theta}^T\Gamma_\theta^{-1}\tilde{\theta} + \frac{1}{2}tr(\widetilde{W}^T\Gamma_W^{-1}\widetilde{W})$$

$$\dot{V} = e^Te + \frac{1}{2}r^T\dot{M}(\emptyset)r + r^TM\dot{r} + \tilde{\theta}^T\Gamma_\theta^{-1}\tilde{\theta} + tr(\widetilde{W}^T\Gamma_W^{-1}\widetilde{W})$$

We know,

$$r = \dot{e} = r - \alpha e$$
$$\dot{\tilde{\theta}} = -\dot{\hat{\theta}}$$
$$\dot{\widetilde{W}} = -\dot{\widehat{W}}$$

And

$$M(\emptyset)\dot{r} = Y\theta + W^T\sigma(\widehat{\Phi}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r$$

On substitution we get,

$$\dot{V} = e^T e + \frac{1}{2}r^T\dot{M}(\emptyset)r + r^T M\dot{r} + \tilde{\theta}^T\Gamma_\theta^{-1}\tilde{\theta} + tr\left(\widetilde{W}^T\Gamma_W^{-1}\dot{\widetilde{W}}\right)$$

$$\dot{V} = e^T\dot{e} + \frac{1}{2}r^T\dot{M}(\emptyset)r + r^T(Y\theta + W^T\sigma(\widehat{\Phi}) - \tau - \frac{1}{2}\dot{M}(\emptyset, \dot{\emptyset})r) + \tilde{\theta}^T\Gamma_\theta^{-1}\dot{\tilde{\theta}} + tr\left(\widetilde{W}^T\Gamma_W^{-1}\dot{\widetilde{W}}\right)$$

On simplification we get,

$$\dot{V} = e^T(r - \alpha e) + r^T\left(Y\theta + W^T\sigma(\widehat{\Phi}) - \tau\right) - \tilde{\theta}^T\Gamma_\theta^{-1}\dot{\hat{\theta}} - tr\left(\widetilde{W}^T\Gamma_W^{-1}\dot{\widehat{W}}\right)$$

Design the input τ

$$\tau = e + Y\hat{\theta} + \widehat{W}^T\hat{\sigma}_k + \beta_\delta sgn(r) + \beta_r r$$

$$\dot{V} = e^T(r - \alpha e) + r^T\left(Y\theta + W^T\sigma(\widehat{\Phi}) - (e + Y\hat{\theta} + \widehat{W}^T\hat{\sigma}_k + \beta_\delta sgn(r) + \beta_r r)\right) - \tilde{\theta}^T\Gamma_\theta^{-1}\dot{\hat{\theta}} - tr\left(\widetilde{W}^T\Gamma_W^{-1}\dot{\widehat{W}}\right)$$

$$\dot{V} = -(e^T\alpha e) + r^T Y\tilde{\theta} + r^T\widetilde{W}^T\hat{\sigma}_k - r^T\beta_\delta sgn(r) - r^T\beta_r r - \tilde{\theta}^T\Gamma_\theta^{-1}\dot{\hat{\theta}} - tr\left(\widetilde{W}^T\Gamma_W^{-1}\dot{\widehat{W}}\right)$$

Design for $\dot{\hat{\theta}}$, we want to get,

$$r^T Y\tilde{\theta} - \tilde{\theta}^T\Gamma_\theta^{-1}\dot{\hat{\theta}} = 0$$
$$\tilde{\theta}^T\Gamma_\theta^{-1}\dot{\hat{\theta}} = r^T Y\tilde{\theta}$$

So,

$$\dot{\hat{\theta}} = \mathrm{proj}(\Gamma_\theta Y^T r)$$

Design for $\dot{\widehat{W}}$, we want to get,

$$r^T\widetilde{W}^T\hat{\sigma} - tr\left(\widetilde{W}^T\Gamma_W^{-1}\dot{\widehat{W}}\right) = 0$$

We know,

$$tr(ba^T) = a^T b$$
$$tr\left(\widetilde{W}^T\hat{\sigma}r^T\right) = tr(\widetilde{W}^T\Gamma_W^{-1}\dot{\widehat{W}})$$

So,

$$\dot{\widehat{W}} = proj(\Gamma_W\widehat{\sigma_k}r^T)$$

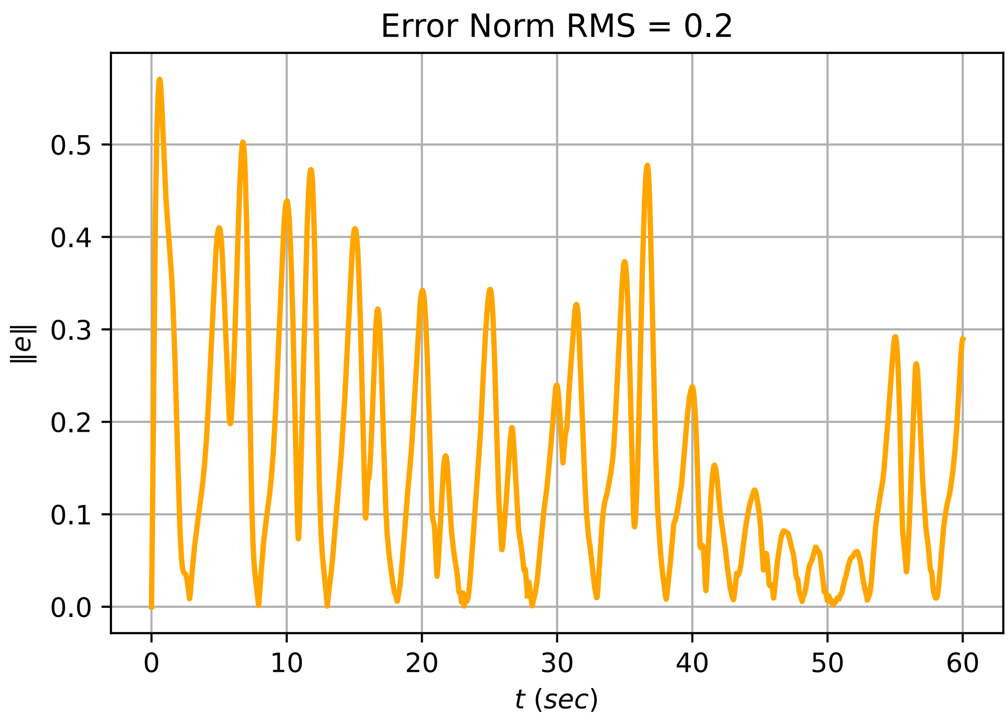
Yielding

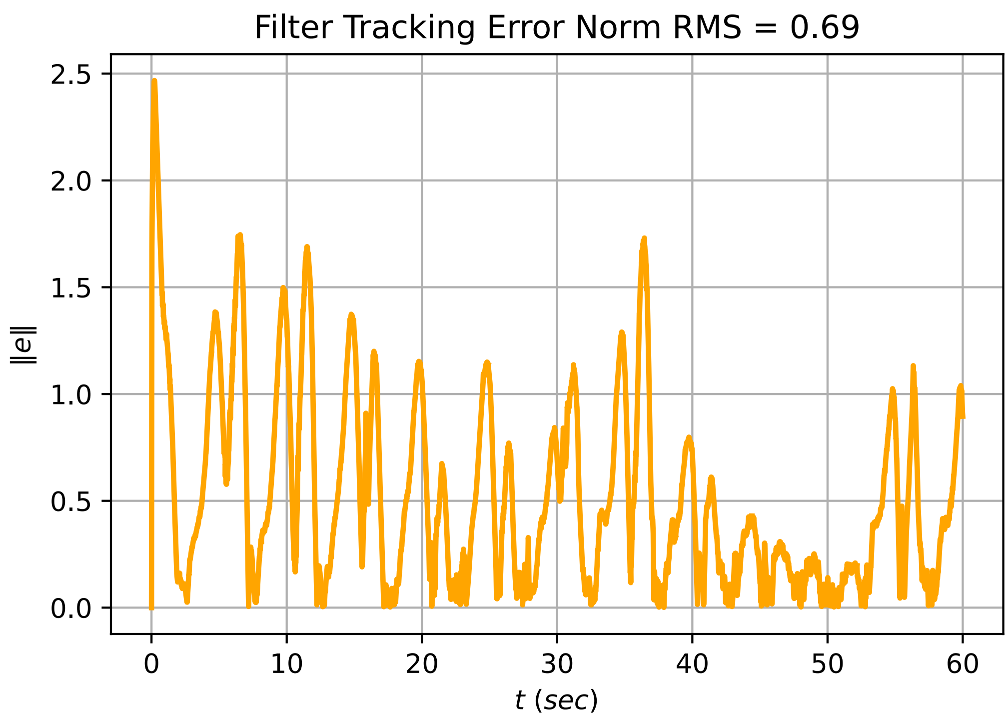
$$\dot{V}_{ly} \leq -(e^T\alpha e) - r^T\beta_r r$$

Using Barbalat's Lemma we can show the $\|e\|, \|r\| \xrightarrow{yields} 0$ which implies Asymptotic tracking

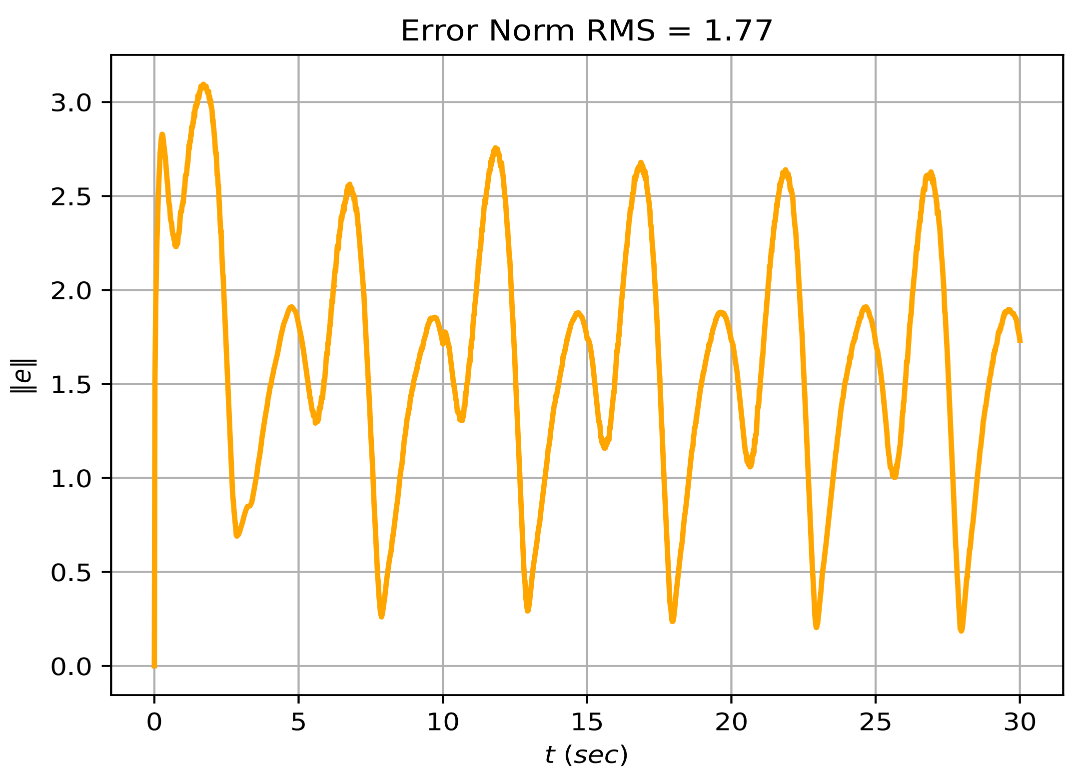
Simulation results

1) The first 4 DNN choices were a good first choice for the architectures because they were easy to implement and these function restricted the bounds of the data to be in the range of -1 to 1, ensuring that the function does not become unstable.
2) After selecting different activation function for each layer, the results showed that linear ReLus with a different output layer give the most accurate approximation of the function, so I used multiple ReLu with a 2 layer of the Sigmoid at the end for the last design.
3) Norm of tracking error and filter tracking error
   From Two-layer Neural network

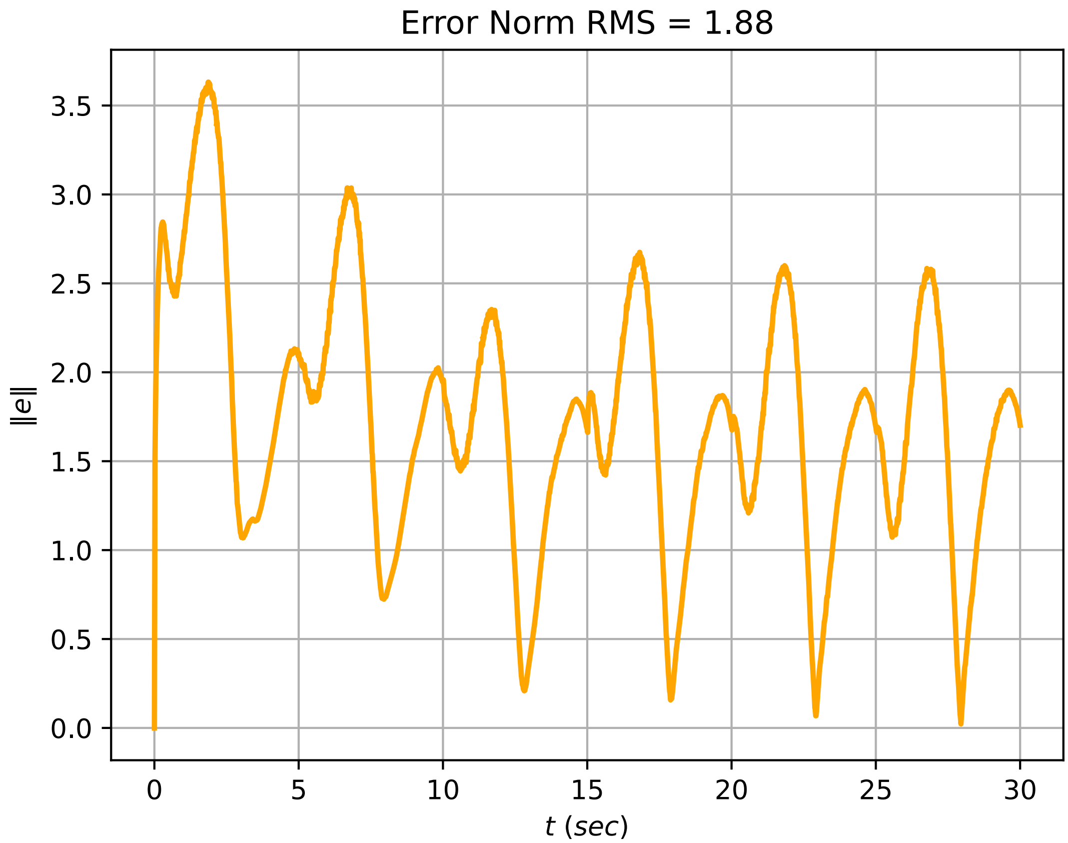




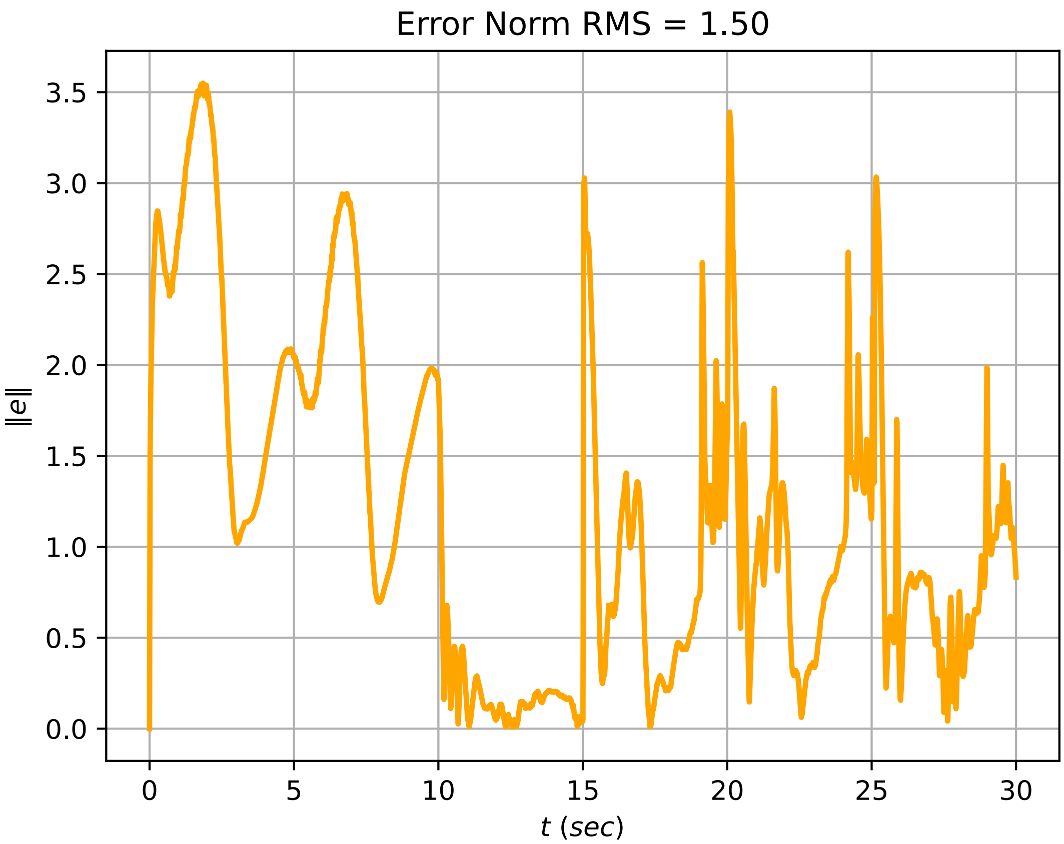
From Gaussian activation functions



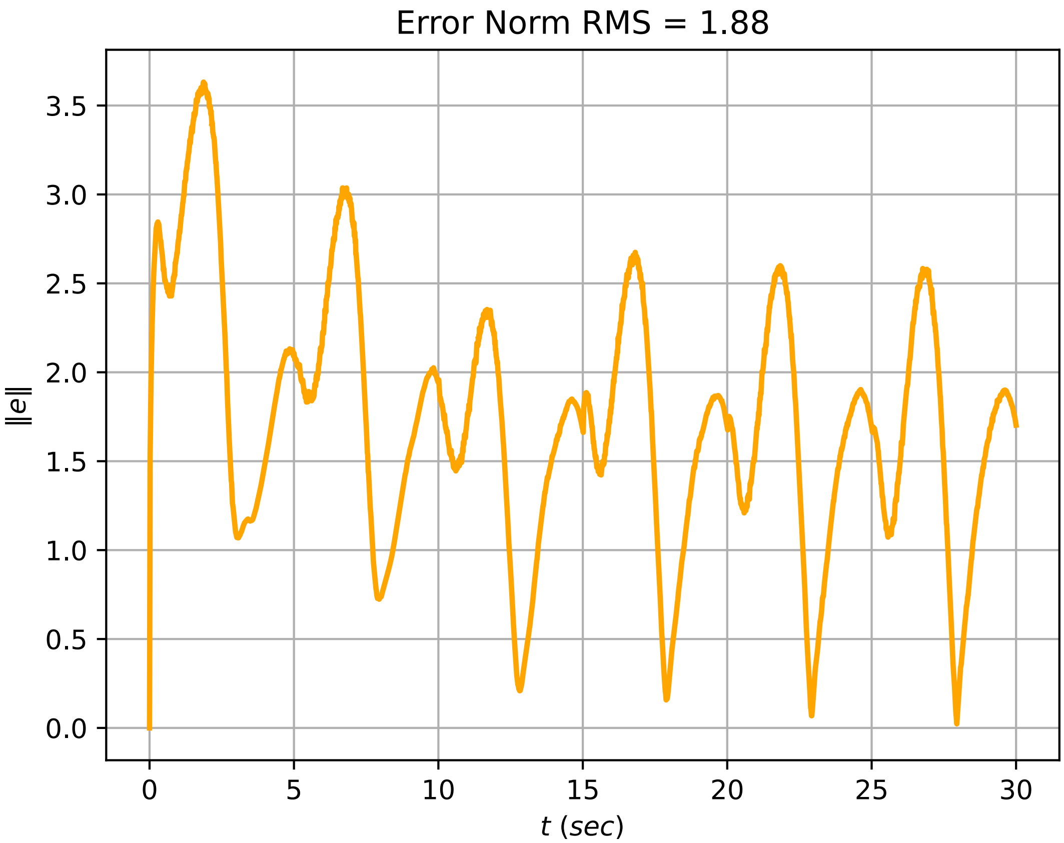
For Tanh activation function



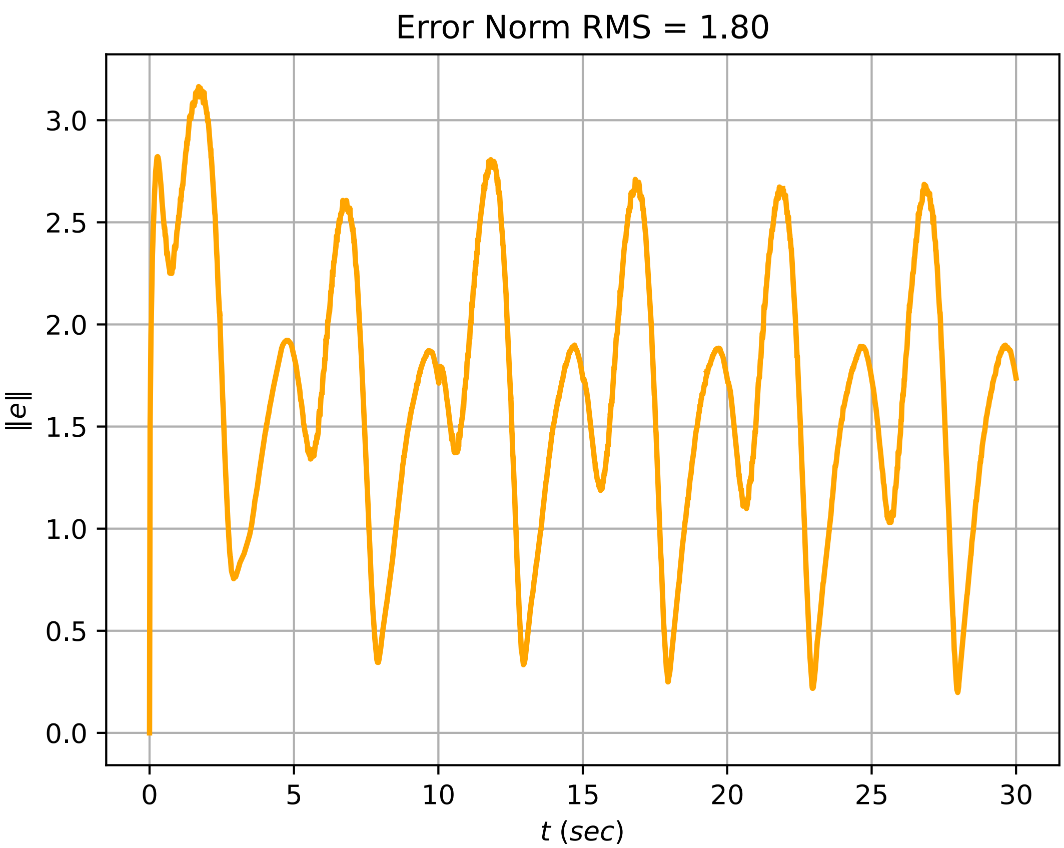
From Relu with Sigmoid output



ReLu with Tanh output



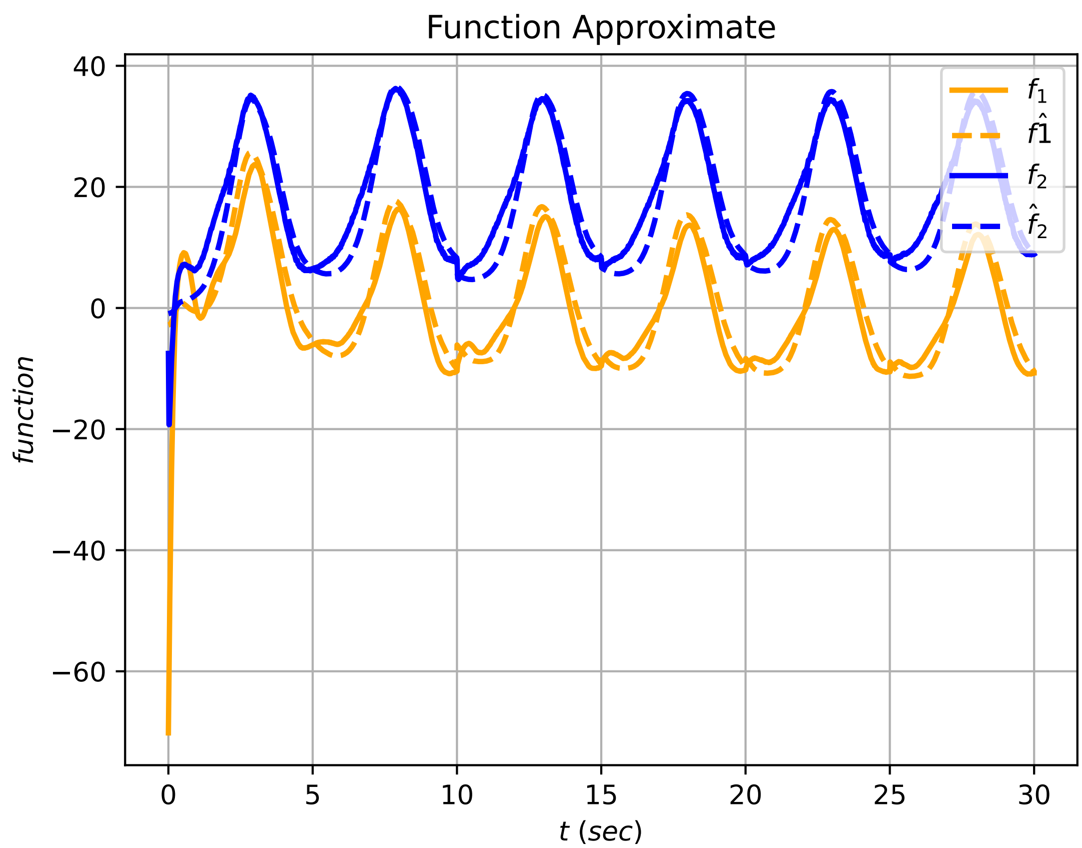
From more ReLu with 2 Sigmoid layers



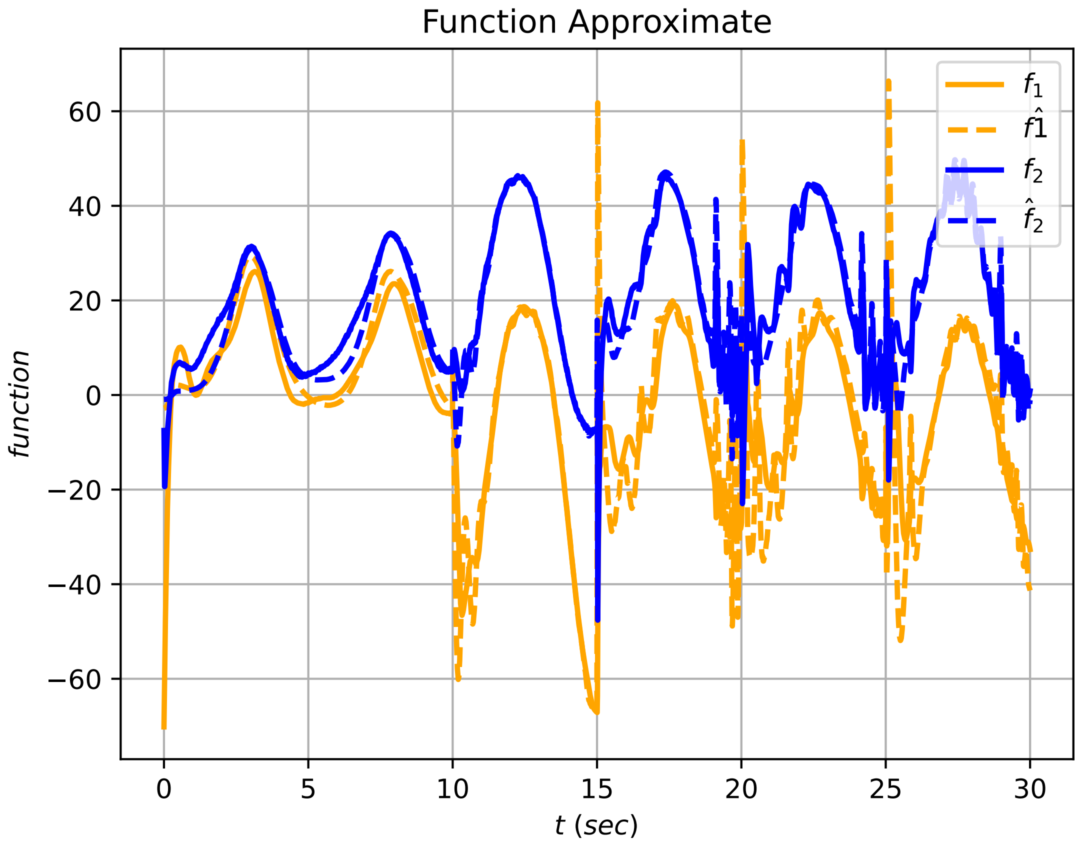
For the graphs it can be seen that the Relu with sigmoid output performed the best with 1.5 RMS error compared to the other architectures. But all the other design performed well in tracking the objective.

(4) Norm of the function approximation

For Gaussian activation

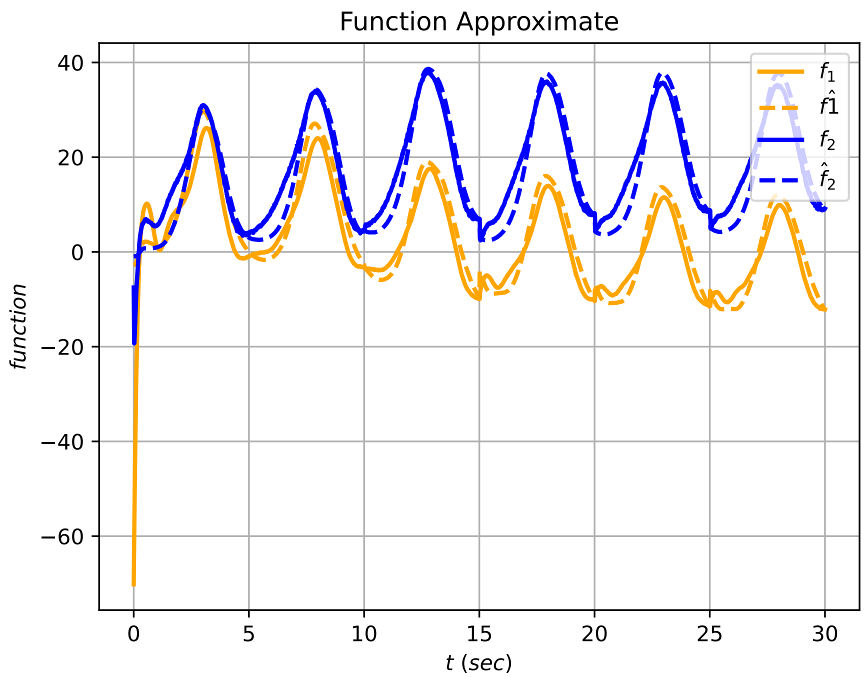


For Tanh activation

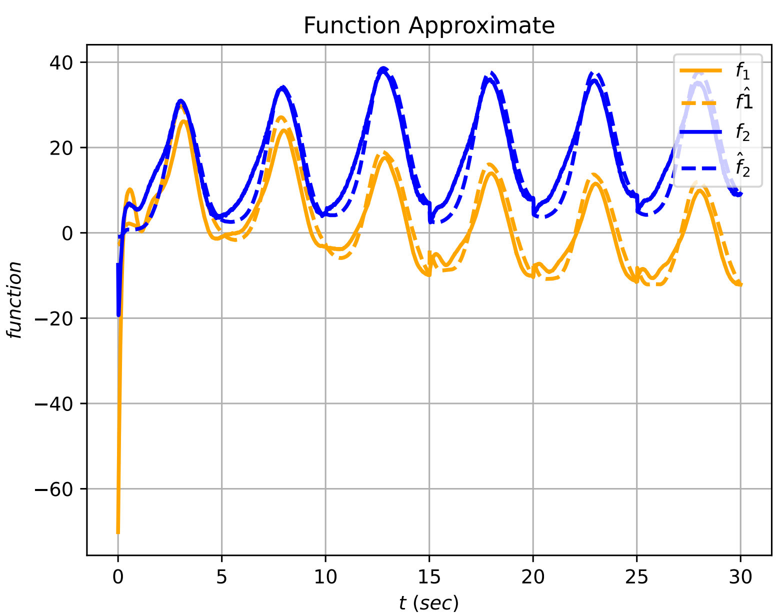


Function Approximate
function
t (sec)
f_1
$f\hat{1}$
f_2
$\hat{f}_2$
60
40
20
0
−20
−40
−60
0
5
10
15
20
25
30

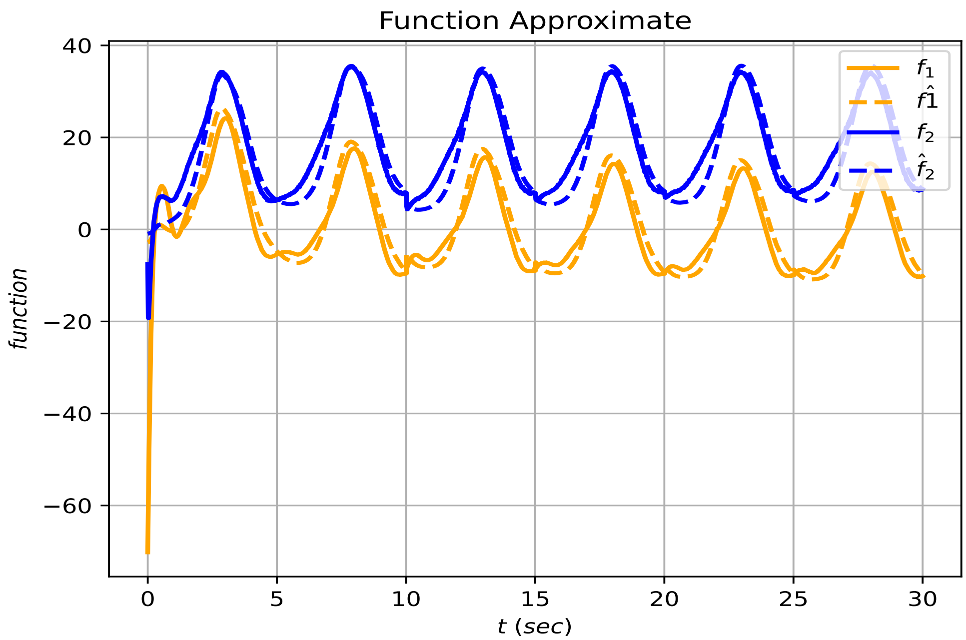
For ReLu with Sigmoid output



For ReLu with Tanh output



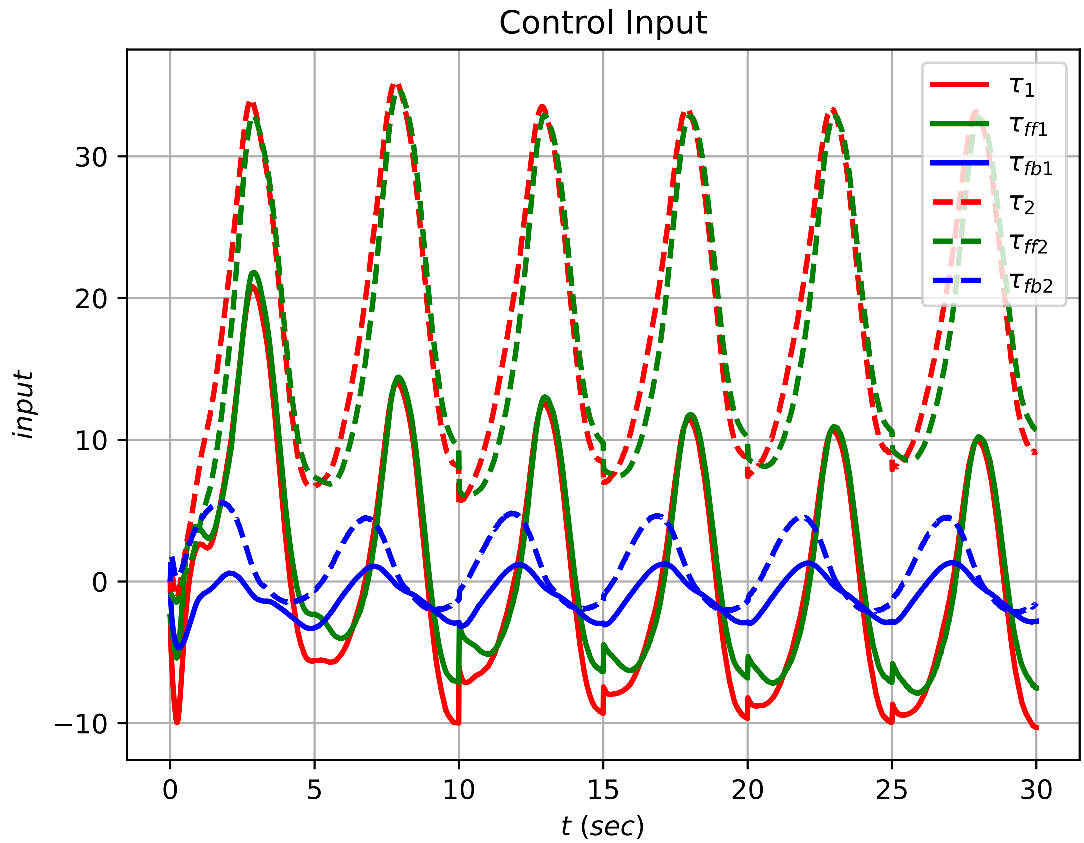
Any design



The last three design seem to perform the best.

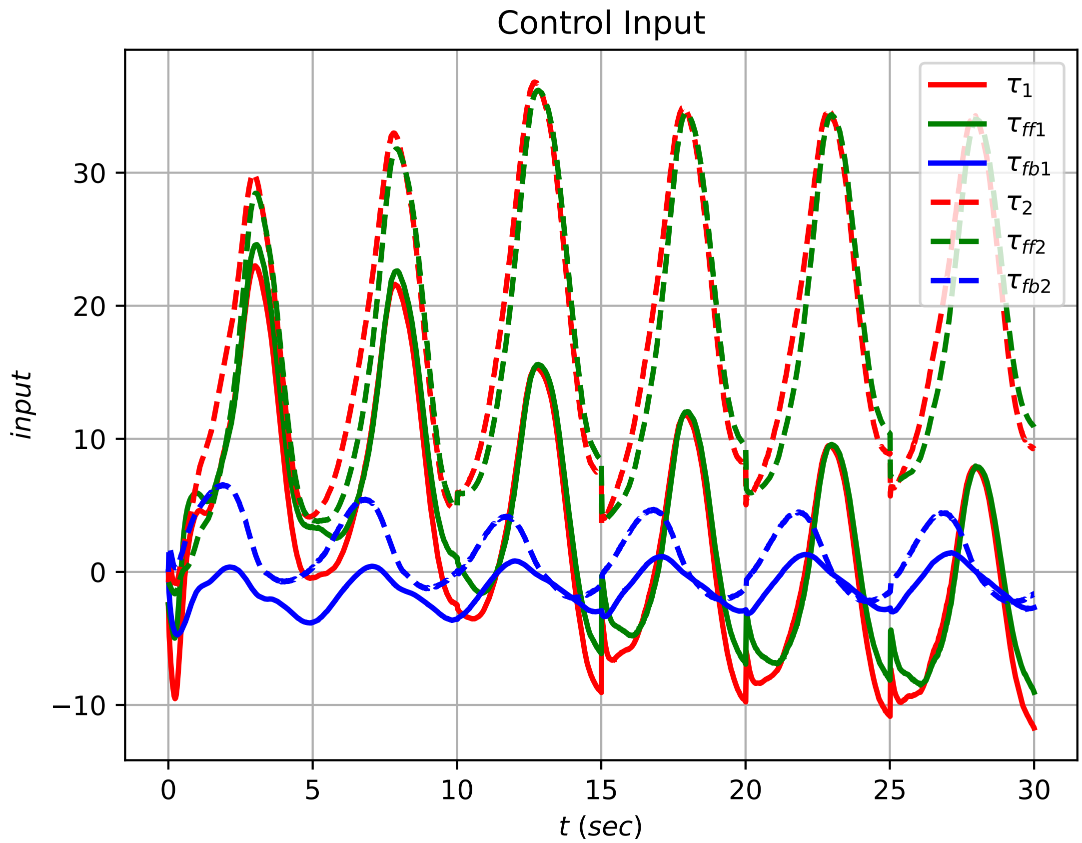
(5) Input plots

(a)



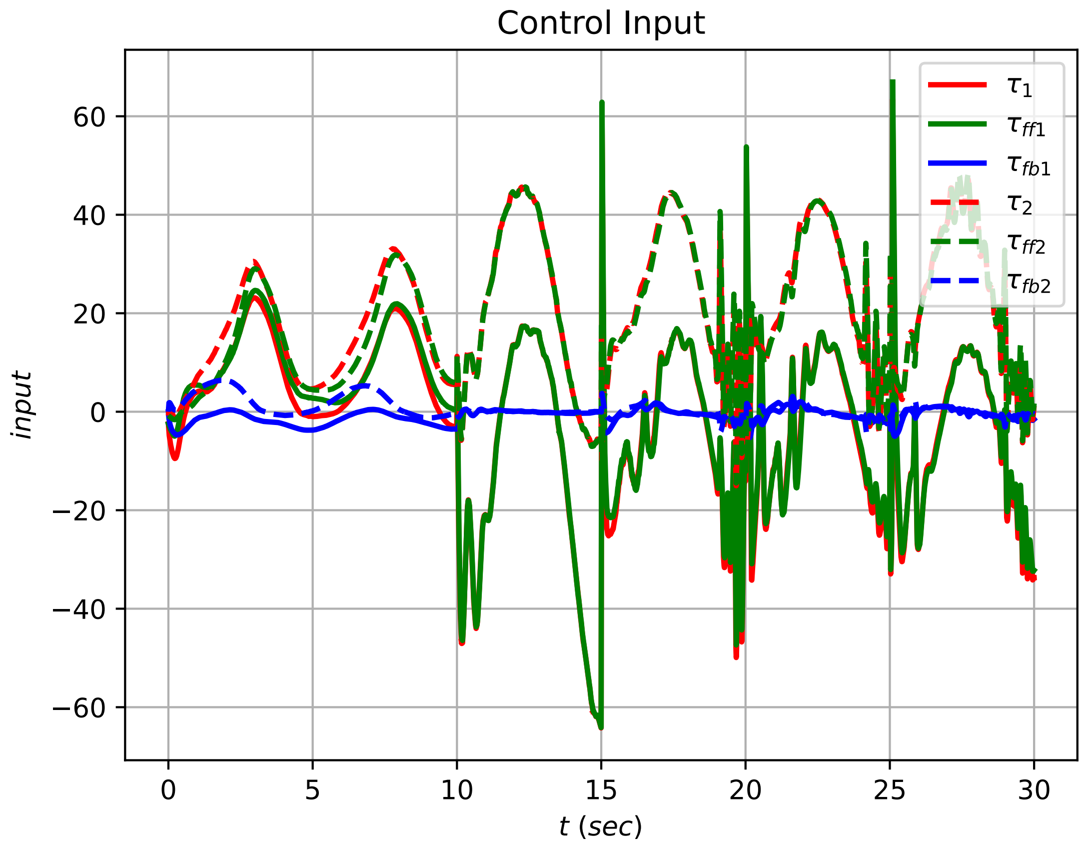
Control Input
τ_1
τ_{ff1}
τ_{fb1}
τ_2
τ_{ff2}
τ_{fb2}
input
30
20
10
0
−10
0
5
10
15
20
25
30
t (sec)

(b)



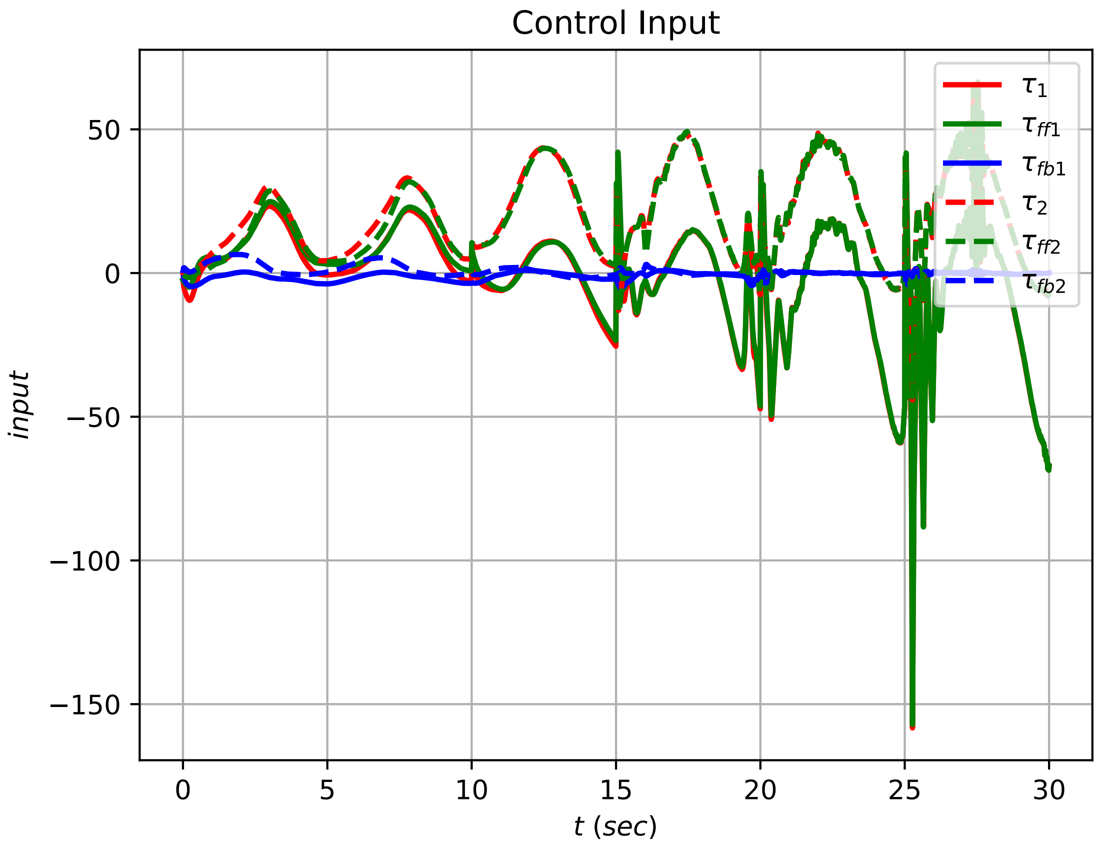
Control Input
τ_1
τ_{ff1}
τ_{fb1}
τ_2
τ_{ff2}
τ_{fb2}
30
20
10
0
−10
input
0
5
10
15
20
25
30
t (sec)

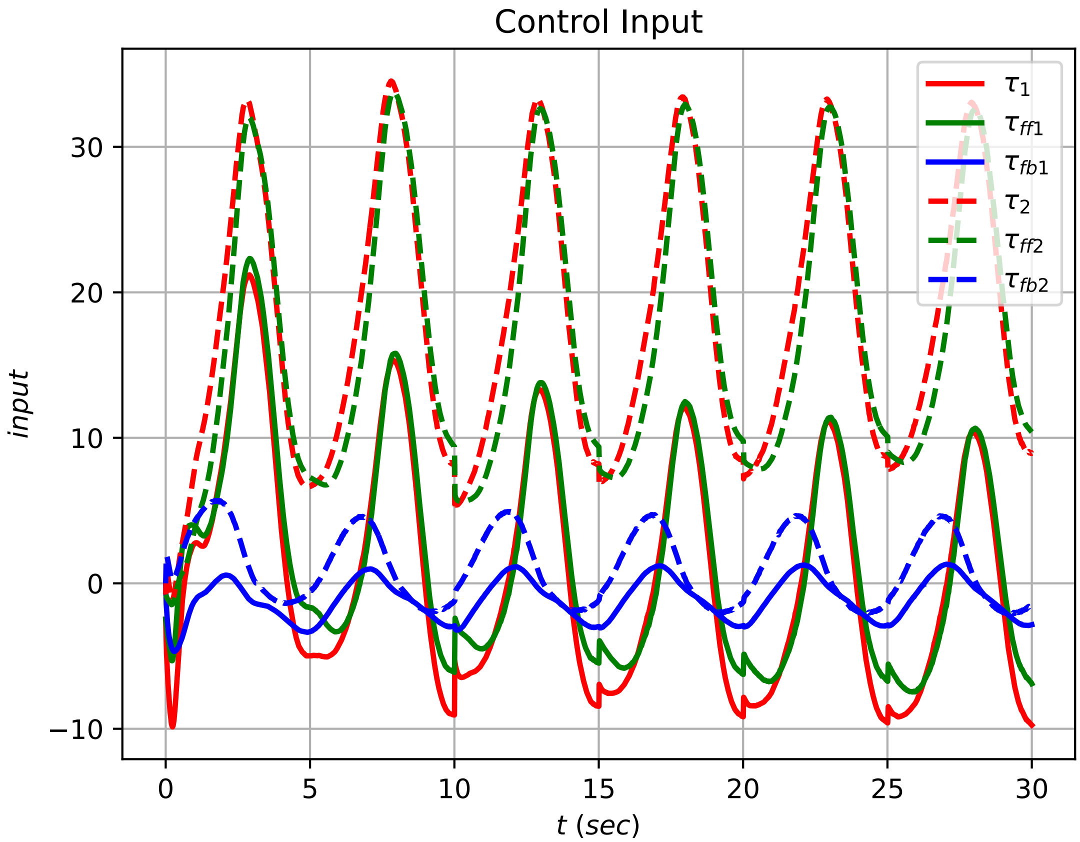
(c)



Control Input
τ_1
τ_{ff1}
τ_{fb1}
τ_2
τ_{ff2}
τ_{fb2}
60
40
20
0
−20
−40
−60
input
0
5
10
15
20
25
30
t (sec)

(d)


Control Input
τ_1
τ_{ff1}
τ_{fb1}
τ_2
τ_{ff2}
τ_{fb2}
50
0
−50
−100
−150
input
0
5
10
15
20
25
30
t (sec)



Control Input
τ_1
τ_{ff1}
τ_{fb1}
τ_2
τ_{ff2}
τ_{fb2}
30
20
10
0
−10
input
0
5
10
15
20
25
30
t (sec)

€