Stevens Institute of Technology

ME 655: Wearable Robotics and Sensors

Course Project - Simulation Lab

<u>Group 7</u>

Tyler Johnson
Jnana Pasagadugula
Mihir Chauhan

Table of Contents

INTRODUCTION

This report outlines the process and results of the control system design for a single Degree of Freedom (DOF) exoskeleton. This exoskeleton is designed to assist the motion of the wearer's left hip in the sagittal plane during treadmill walking. The hip joint is modeled as an ideal revolute joint actuated by a pure torque generator. The exoskeleton is rigidly attached to the wearer's leg affected by the hip extensor and flexor muscles approximated with a torque generator. To model the control system several subsystems are developed to steer and estimate the conditions of the exoskeleton. These subsystems include the inverse dynamics of the system, a pool of Adaptive Frequency Oscillators (AFOs), a model based assistive controller, and a Kernel-based nonlinear filter to develop a model-free assistive controller. The model based and model free controllers effectiveness are compared.

Methods

Part A.1

The torque τ_H required to steer the system is given in the simulink block, human_motor_control, it takes the error between the desired angular positions, and actual position and the desired angular velocity and the actual velocity and the desired angular acceleration, and they would be fit into a PD controller, and it also uses the torque applied by the gravity of leg and robot to calculate the torque required by the humans τ_H.

The θ_L can be plotted using the LegRobotModel, the forward dynamics model of the Leg and Robot is given by,

$$\theta'' = \frac{1}{I_{tot0}} \cdot [\tau_H - m_H \times g \times L_H \times sin(\theta) + \tau_R - m_R \times g \times L_R \times sin(\theta)]$$

$$I_{totO} = I_{HO} + I_{RO} = (I_{HC} + m_H L_H^2) + (I_{RC} + m_R L_R^2)$$

I_{HO}: human leg inertia with respect to the joint O

I_{RO}: robot inertia with respect to the joint O

Part A.2

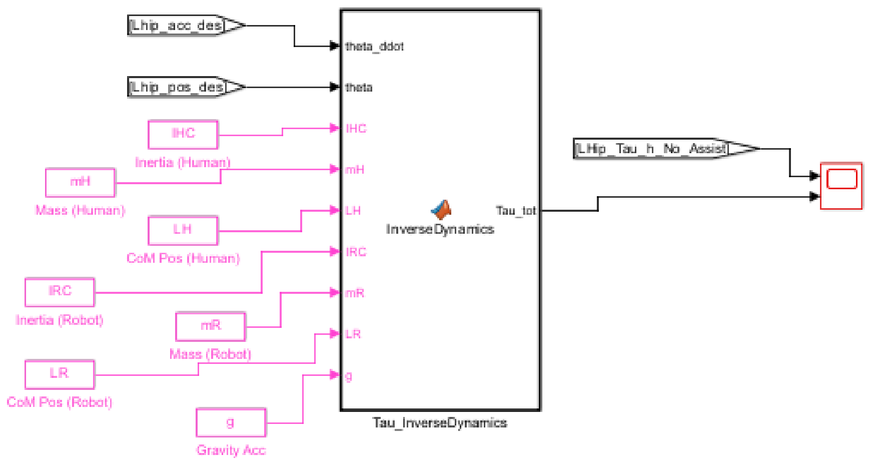
The inverse dynamics of the model can be calculated using the LegRobotModel,

$$\theta'' = \frac{1}{I_{tot}} \cdot [\tau_H - m_H \times g \times L_H \times sin(\theta) + \tau_R - m_R \times g \times L_R \times sin(\theta)]$$

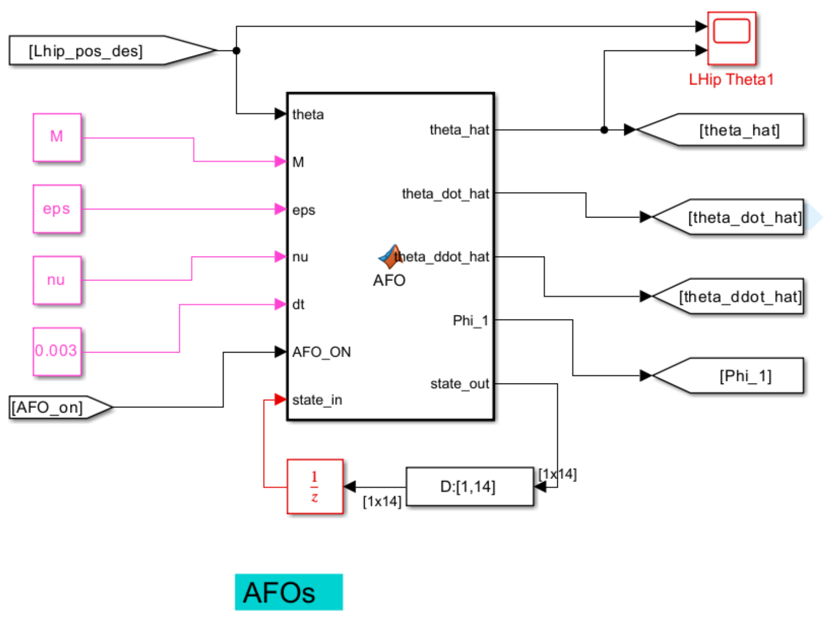
$$\Rightarrow \theta'' = \frac{1}{I_{tot}} \cdot [(\tau_H + \tau_R) - (m_H \times g \times L_H \times sin(\theta) + m_R \times g \times L_R \times sin(\theta))]$$

$$\Rightarrow (\tau_H + \tau_R) = \theta'' \cdot I_{tot} + (m_H \times g \times L_H \times sin(\theta) + m_R \times g \times L_R \times sin(\theta))$$

And this model can be verified using the τ_H calculated by the model, as we assumed the $\tau_R = 0$



Part A.3



Update oscillator parameters using these formulas

```
% Update Oscillator Parameters
Phi = mod(Phi + dt * ((1:M) * w + eps * (theta - theta0 - sum(a .* sin(Phi))) * cos(Phi)), 2 * pi);
a = a + dt * nu * (theta - theta0 - sum(a .* sin(Phi))) * sin(Phi);
w = w + dt * eps * (theta - theta0 - sum(a .* sin(Phi))) * cos(Phi(1));
theta0 = theta0 + dt * nu * (theta - theta0 - sum(a .* sin(Phi)));

state_out = [Phi, a, w, theta0];
```

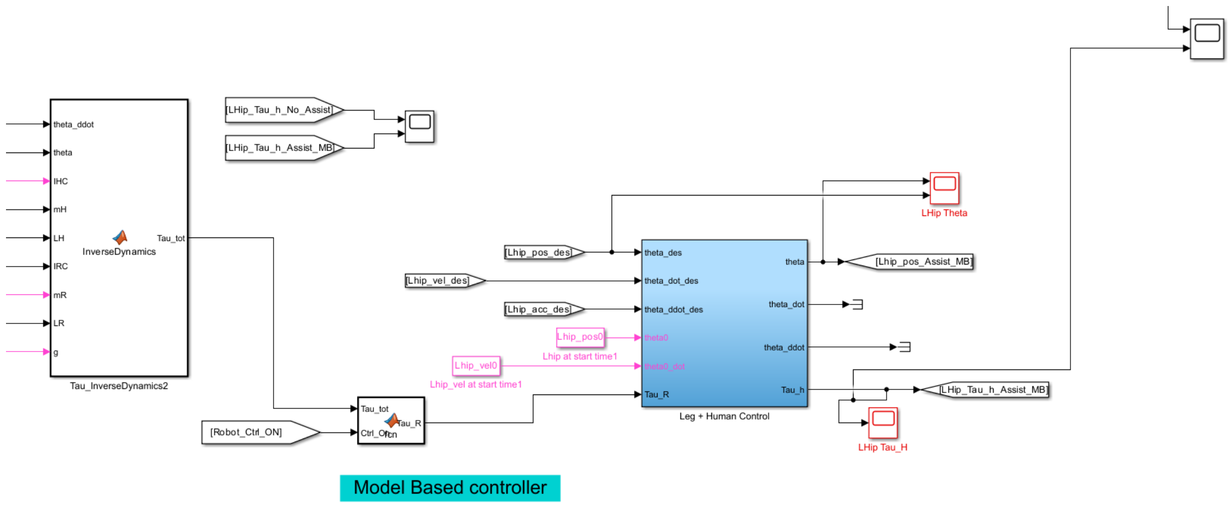
Update estimates of theta_dot and theta_ddot using the following formulas in the MATLAB function

```
% Update estimates of theta_dot and theta_ddot
theta_dot_est = sum((1:M) .* w .* a .* cos(Phi));
theta_ddot_est = -sum((1:M) .* w.^2 .* a .* sin(Phi));
```

AFO will be activated after the defined time below

```
t_start=5.0; %[s] time of AFO activation
```

Part A.4



Defining to use 50% of the torque achieved from the Inverse Dynamics block

```
function Tau_R = fcn(Tau_tot, Ctrl_On)
if Ctrl_On == 0
    Tau_R = 0;
else
    Tau_R = 0.5*Tau_tot;
end
```

Defining that the controller should be activated at 10 seconds after AFO is active

```
dt_active_control=10; %[s] wait 10s after AFO is active before turning
assistance ON
```

Then we compared the plots of the τ_H achieved from Part 1 with the τ_H achieved here.

Part A.5

Estimation of the assisted torque of the hip is done utilizing adaptive frequency oscillators tied to a nonlinear filter. To implement the Kernel-based nonlinear filter, the input error is calculated as a function of $\boldsymbol{\tau}_H$ and $w_i(t)$.

$$e_i(t) = [\tau_H - w_i(t)], \mathrm{i} = 1, \dots, \mathrm{N}$$

The phase is computed using the following formula.

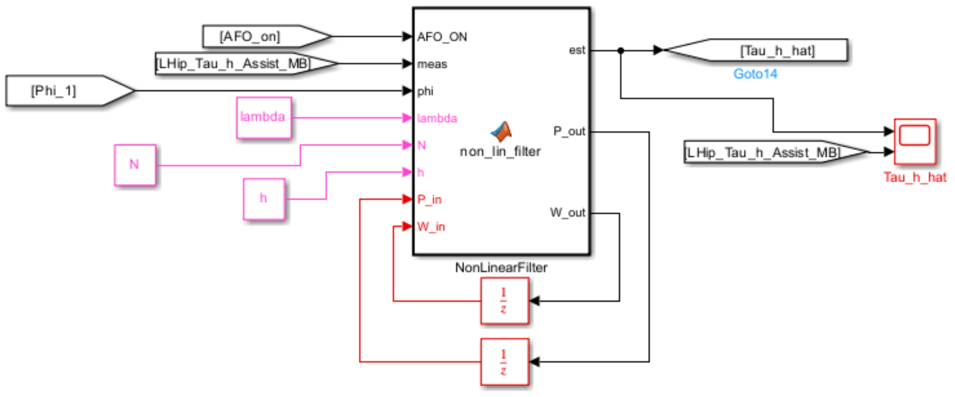
$$P_i(t+1) = \frac{1}{\lambda}\left(P_i(t) - \frac{P_i(t)^2}{\frac{\lambda}{\Psi_i(\varphi_1(t))} + P_i(t)}\right), i = 1, \dots, N$$

From the error and new phase data we can then compute the weighting.

$$w_i(t+1) = w_i(t) + \Psi_i(t)P_i(t+1)e_i(t), i = 1, \dots, N$$

These equations are implemented in MATLAB utilizing the outputs of the AFO in part 3 as the initial gait phase and torque inputs.

The function block below shows the nonlinear filter developed.



Non-linear filter

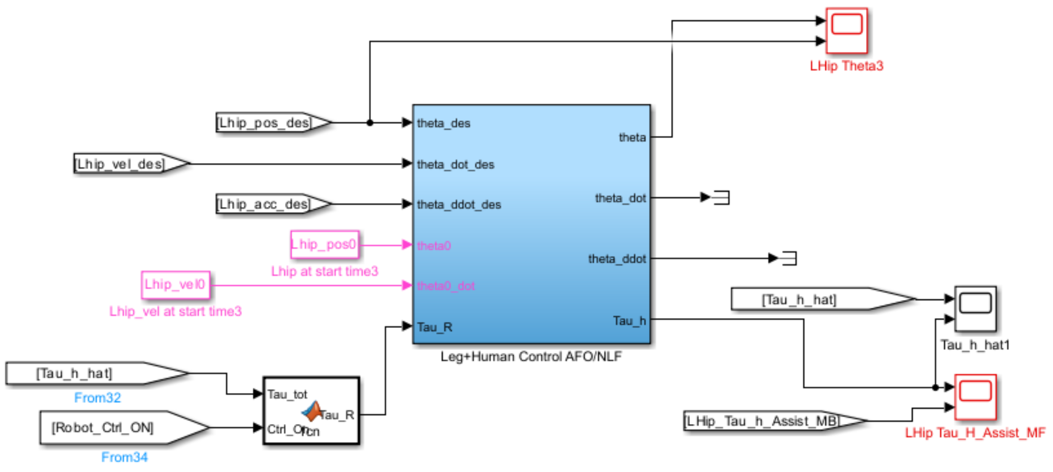
Contained within the function block is the following code which computes the above equations across the range of motion.

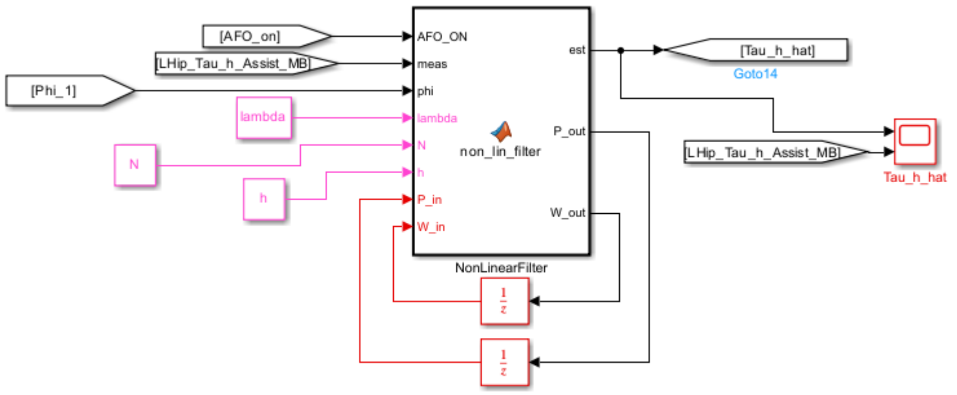
```
function [est,P_out,W_out]= non_lin_filter(AFO_ON, meas, phi, lambda, N,h,
P_in, W_in)
if AFO_ON == 0
   est = meas;
   P_out = P_in;
   W_out = W_in;
else
i=1;
est = meas;
est_num = 0;
est_den = 0;
P_out = P_in;
W_out = W_in;
while i <= N
   psi_i = exp(h*(cos(phi-((2*pi*i)/N))-1));
   %RLS
   e = meas-W_in;
   P_out = (1/lambda)*(P_in-((P_in^2)/((lambda/psi_i)+P_in)));
   W_out = W_in+psi_i*P_out*e;
   est_num = est_num + (psi_i*W_out);
   est_den = est_den + psi_i;
```

```
    est = est_num/est_den;
    i=i+1;
end
end
```

Part A.6

Utilizing the output of the nonlinear filter developed in part 5 as the input torque to the Human-Robot dynamic model, the human torque is plotted with the input robotic assistive torque. This should show that if the AFO+nonlinear filter are assisting, the assisted torque should be less than the unassisted.





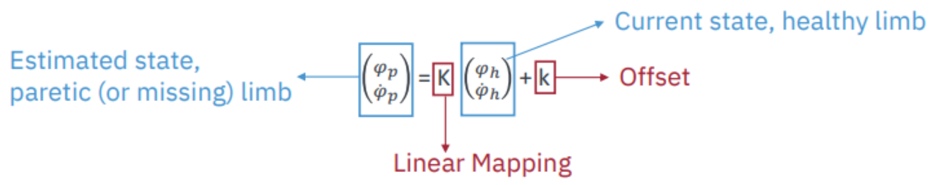
Non-linear filter

Part B.7

The CLME is used to develop an alternative estimator of left hip motion based on the right hip motion (inter-limb coordination). KF is used to compute coherent estimated joint trajectories & from CLME estimated joint trajectories.

$$\overset{(\overline{\theta}_L, \overline{\dot{\theta}}_L)}{\boxed{\text{Left hip trajectory}}} = K \overset{(\theta_{R,des}, \dot{\theta}_{R,des})}{\boxed{\text{Right hip trajectory}}} + k$$

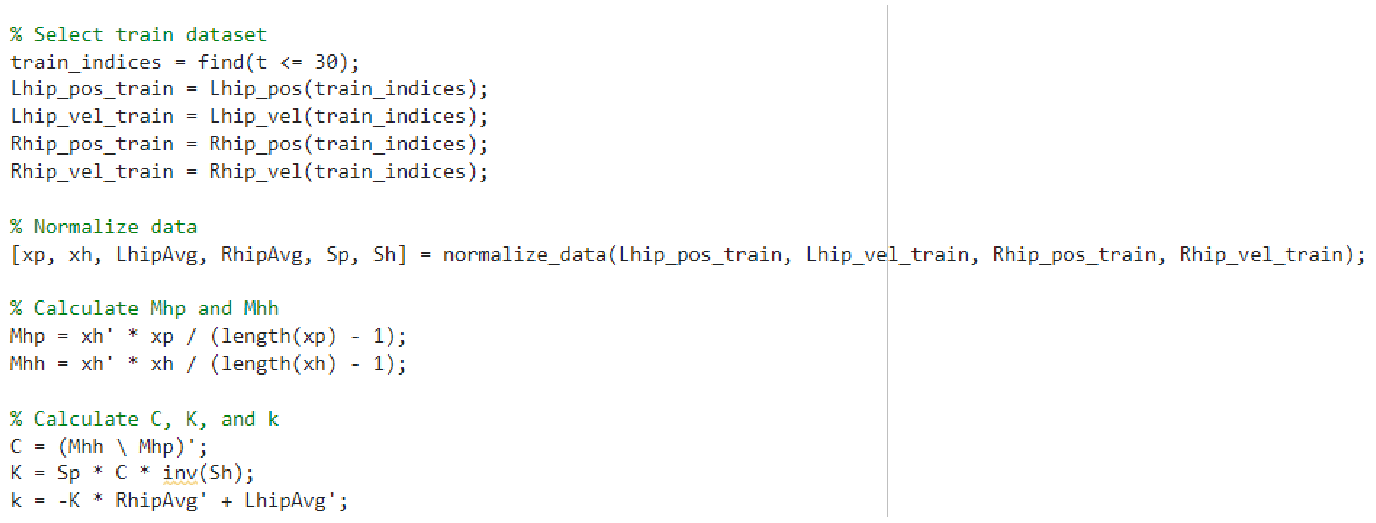
CLME


Current state, healthy limb
Estimated state, paretic (or missing) limb
$\begin{pmatrix}\varphi_p \\ \dot{\varphi}_p\end{pmatrix} = K \begin{pmatrix}\varphi_h \\ \dot{\varphi}_h\end{pmatrix} + k$
Offset
Linear Mapping

Parameters K and k are estimated using the linear regression model, using the first 30 seconds of the gait cycle from the right hip trajectory, the parameters can be calculated as:

$$K = S_p C S_h^{-1} \qquad \mathrm{k} = -\mathrm{K}\begin{pmatrix}\overline{\varphi}_h \\ \overline{\dot{\varphi}}_h\end{pmatrix} + \begin{pmatrix}\overline{\varphi}_p \\ \overline{\dot{\varphi}}_p\end{pmatrix}$$

The code to calculate the CLME parameters is:

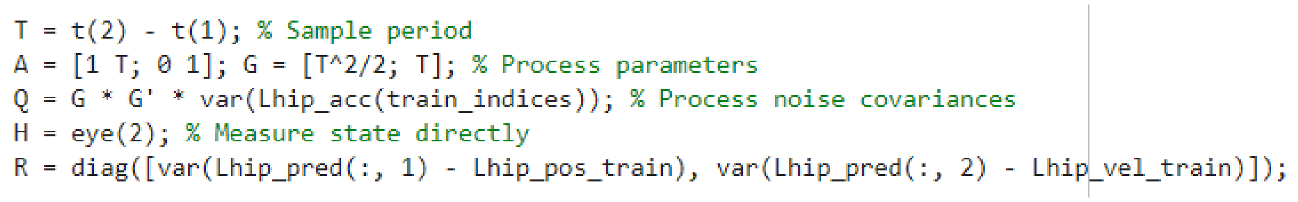


```
% Select train dataset
train_indices = find(t <= 30);
Lhip_pos_train = Lhip_pos(train_indices);
Lhip_vel_train = Lhip_vel(train_indices);
Rhip_pos_train = Rhip_pos(train_indices);
Rhip_vel_train = Rhip_vel(train_indices);

% Normalize data
[xp, xh, LhipAvg, RhipAvg, Sp, Sh] = normalize_data(Lhip_pos_train, Lhip_vel_train, Rhip_pos_train, Rhip_vel_train);

% Calculate Mhp and Mhh
Mhp = xh' * xp / (length(xp) - 1);
Mhh = xh' * xh / (length(xh) - 1);

% Calculate C, K, and k
C = (Mhh \ Mhp)';
K = Sp * C * inv(Sh);
k = -K * RhipAvg' + LhipAvg';
```

And by using these trained parameters, the Left hip trajectory is estimated using the right hip trajectory

Kalman Filter:

The constant matrices A, Q, H, R are calculated by using the code:

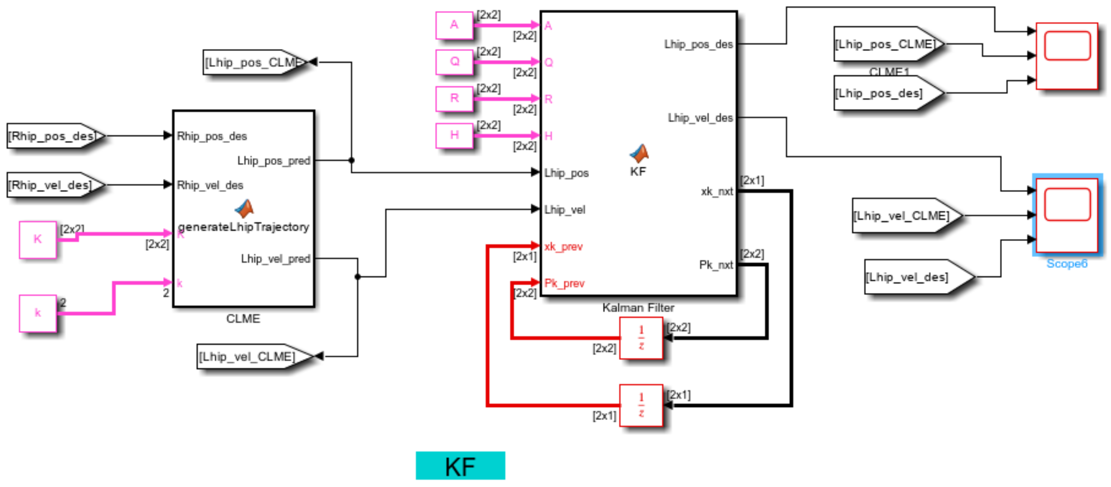


```
T = t(2) - t(1); % Sample period
A = [1 T; 0 1]; G = [T^2/2; T]; % Process parameters
Q = G * G' * var(Lhip_acc(train_indices)); % Process noise covariances
H = eye(2); % Measure state directly
R = diag([var(Lhip_pred(:, 1) - Lhip_pos_train), var(Lhip_pred(:, 2) - Lhip_vel_train)]);
```

And the KF algorithm is implemented in simulink matlab function block as:

```
zk = [Lhip_pos;Lhip_vel];
%prediction stage
xk_pred = A*xk_prev;
Pk_pred = A*Pk_prev*A' + Q;

%correction stage
K = Pk_pred*H'*inv(H*Pk_pred*H' + R);
xk_nxt = xk_pred + K*(zk - H*xk_prev);
Pk_nxt = (eye(2) - K*H)*Pk_pred;

Lhip_pos_des = xk_nxt(1);
Lhip_vel_des = xk_nxt(2);
```

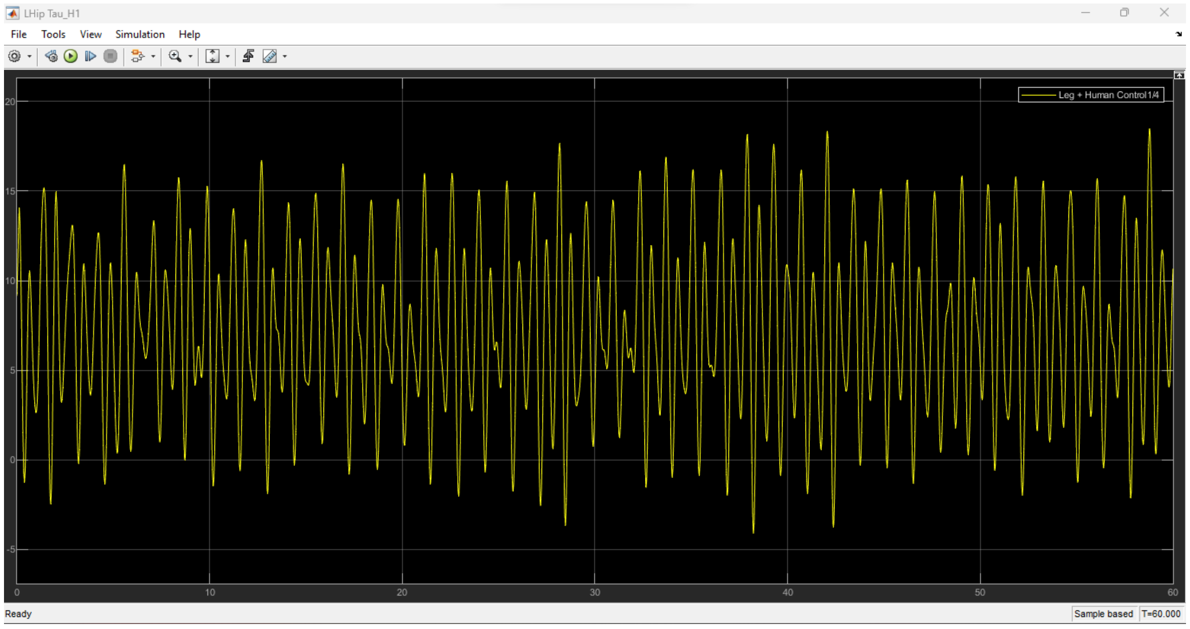
The blocks containing the CLME and Kalman Filter code are built as seen below.



Results

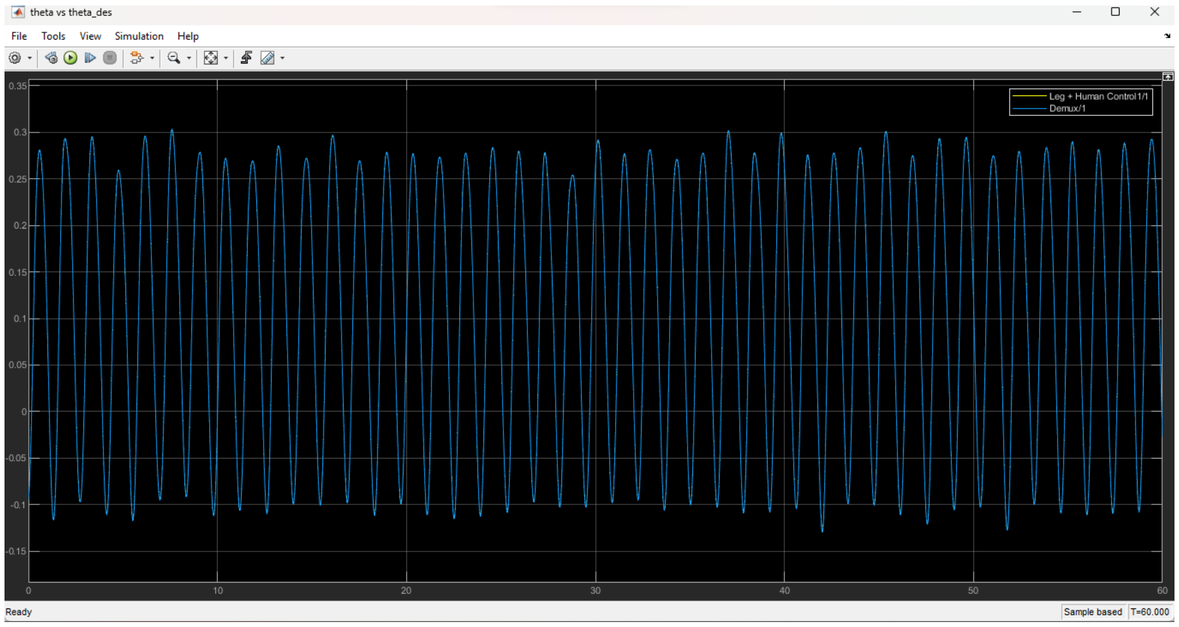
Part A.1

The plot of torque τ_H required to steer the system,

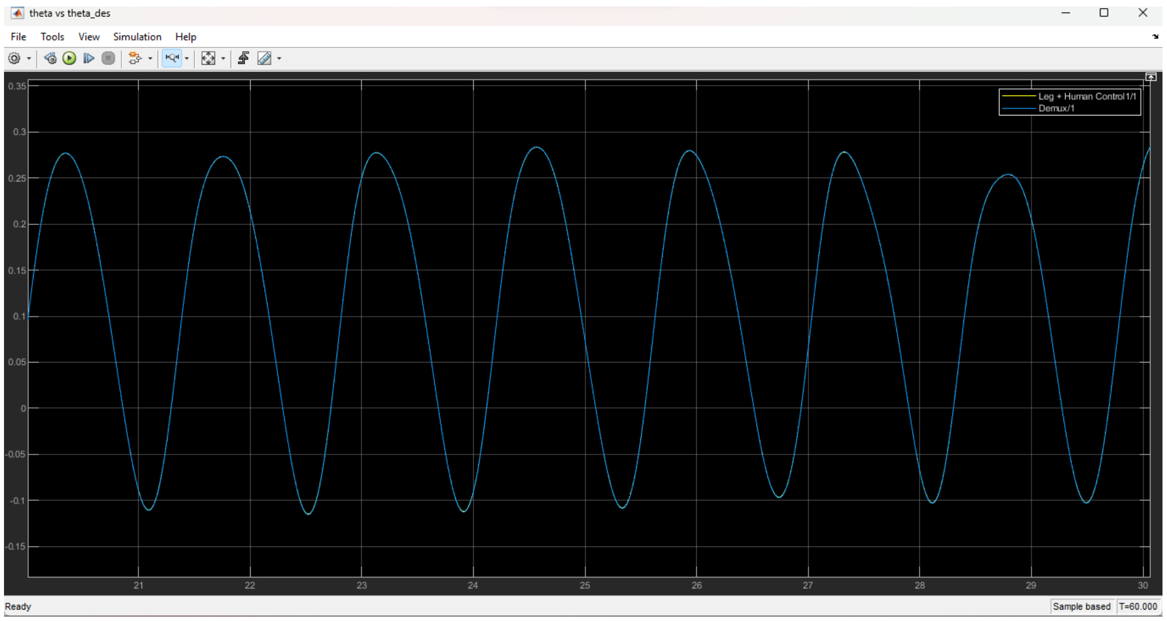


The plot of torque required to steer the system

Plot of θ_L and $\theta_{L,des}$ vs. time,



θ_L in yellow vs $\theta_{L,des}$ in blue over time.

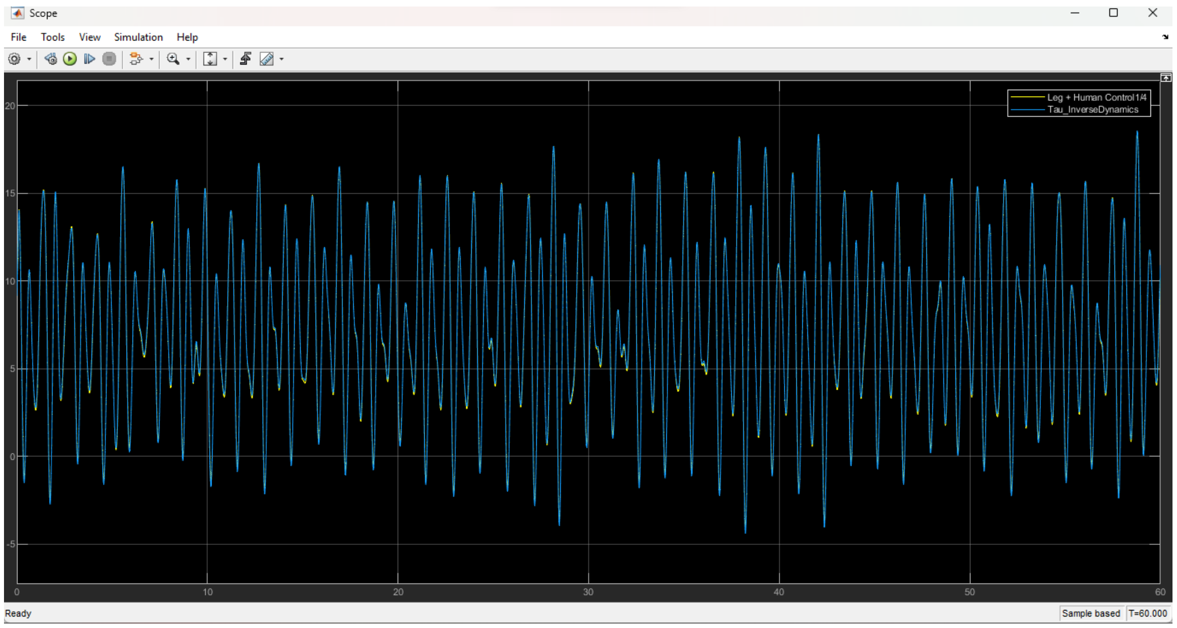


Zoomed in plot of θ_L in yellow vs $\theta_{L,des}$ in blue over time.

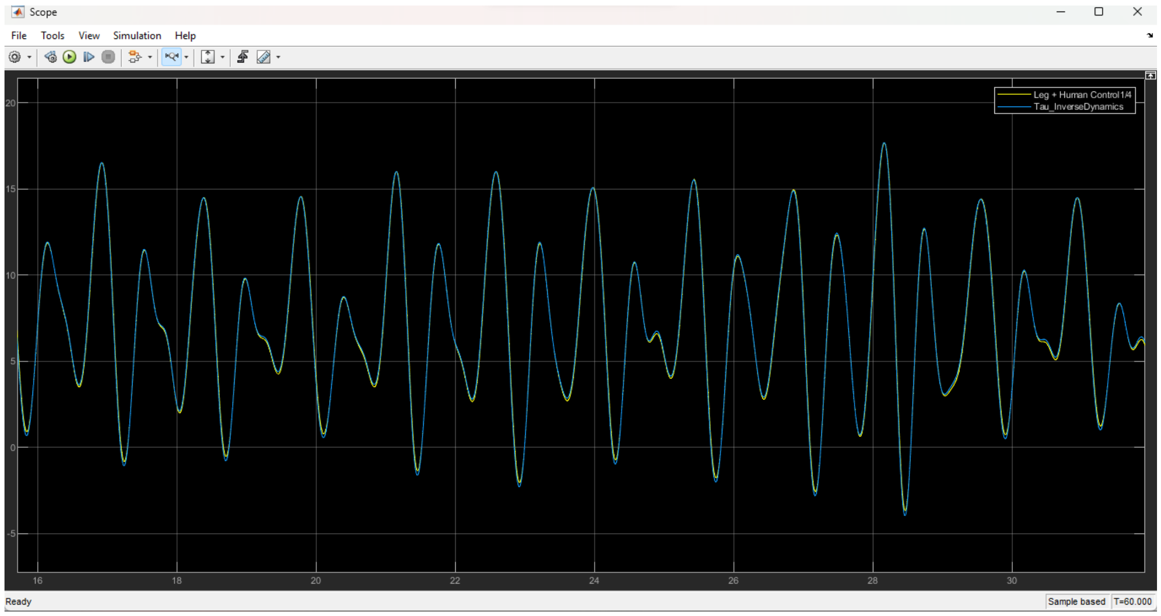
The predicted joint trajectory of the left hip joint matches the desired joint trajectory, and hence proves that the human motor control system can track the desired trajectory.

Part A.2

The inverse dynamics of the overall system (robot + human leg), and implement it as a Simulink block,



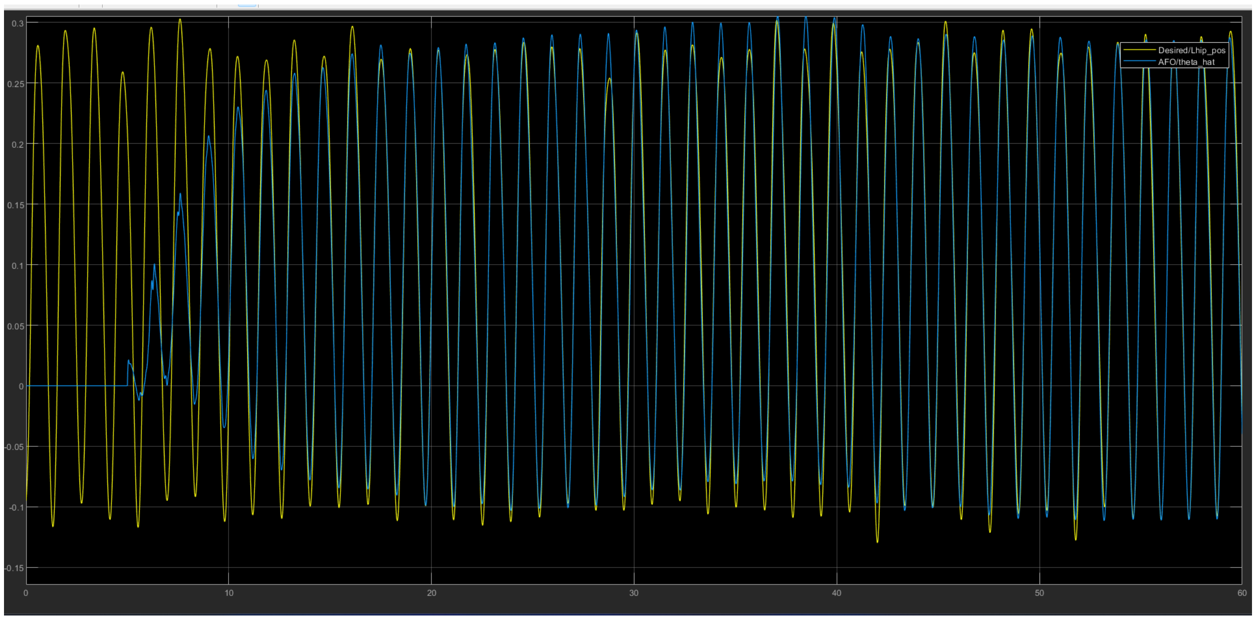
τ_H calculated using the inverse dynamics model in yellow vs the τ_H no assist calculated in (A.1) in blue over time.

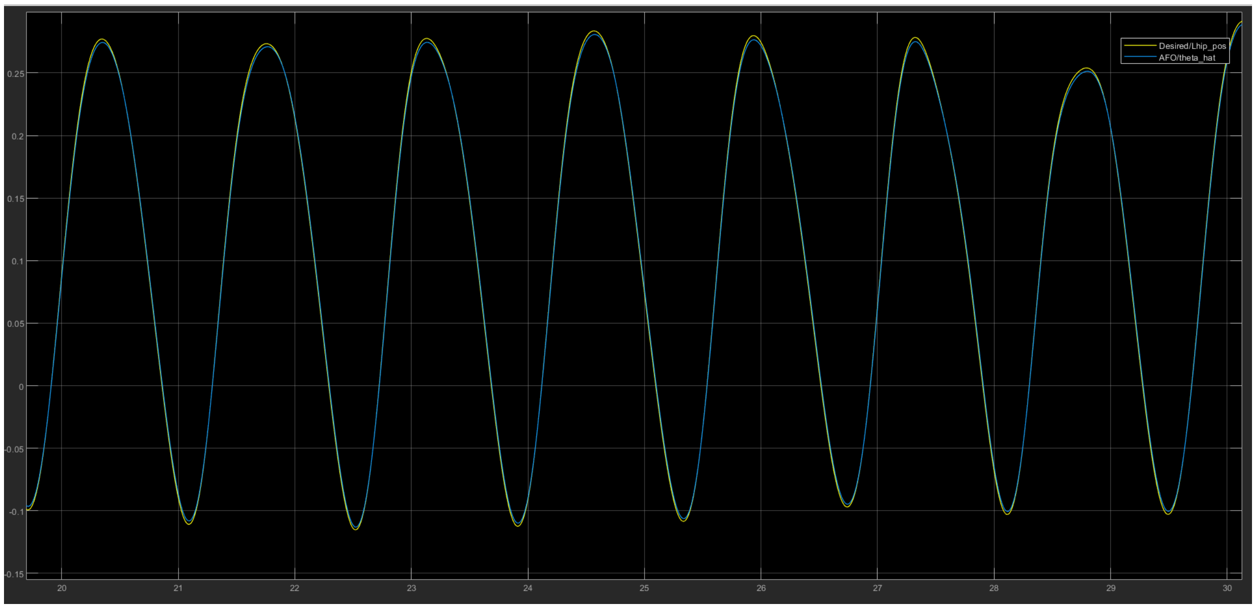


Zoomed in version of τ_H calculated using the inverse dynamics model in yellow vs the τ_H no assist calculated in (A.1) in blue over time.

The τ_H calculated using the inverse dynamics model matches the output of the torque trajectory from the provided human leg model.

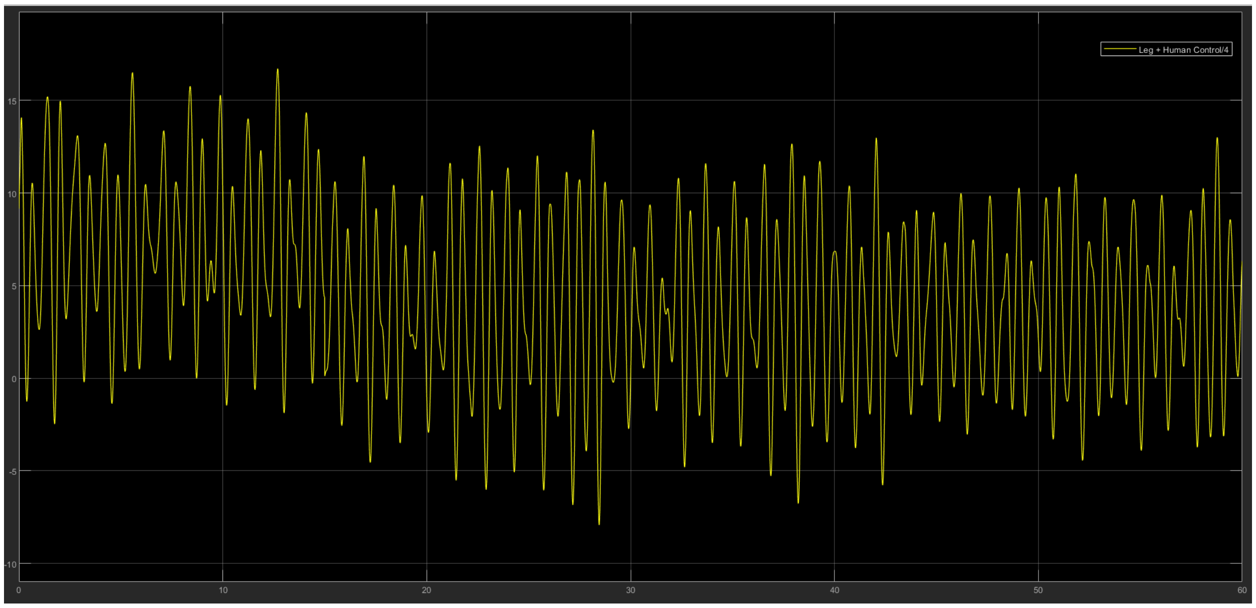
Part A.3



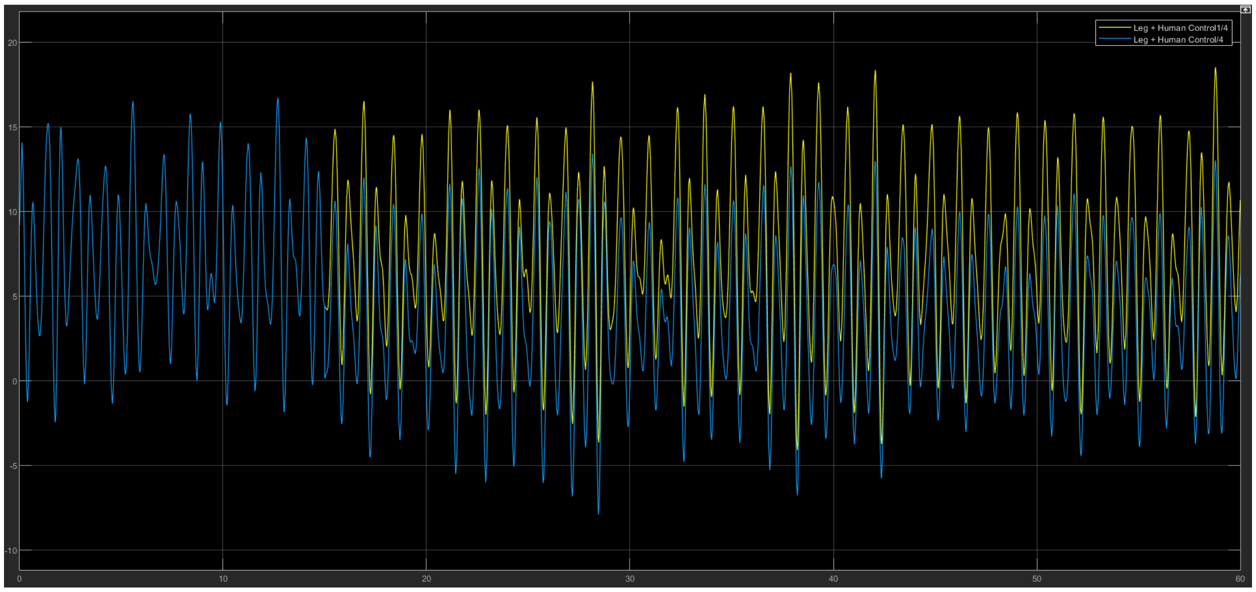


Part A.4

τ_H vs Time

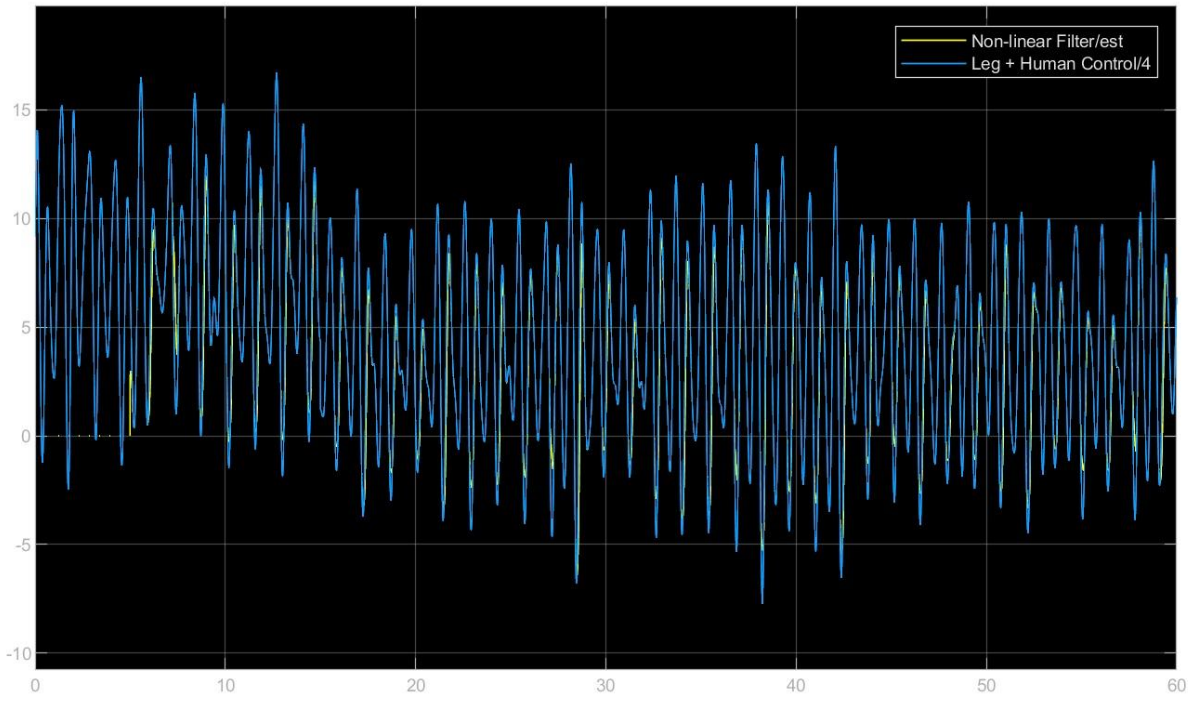


Plot of τ_H in No assist vs Assist



Part A.5

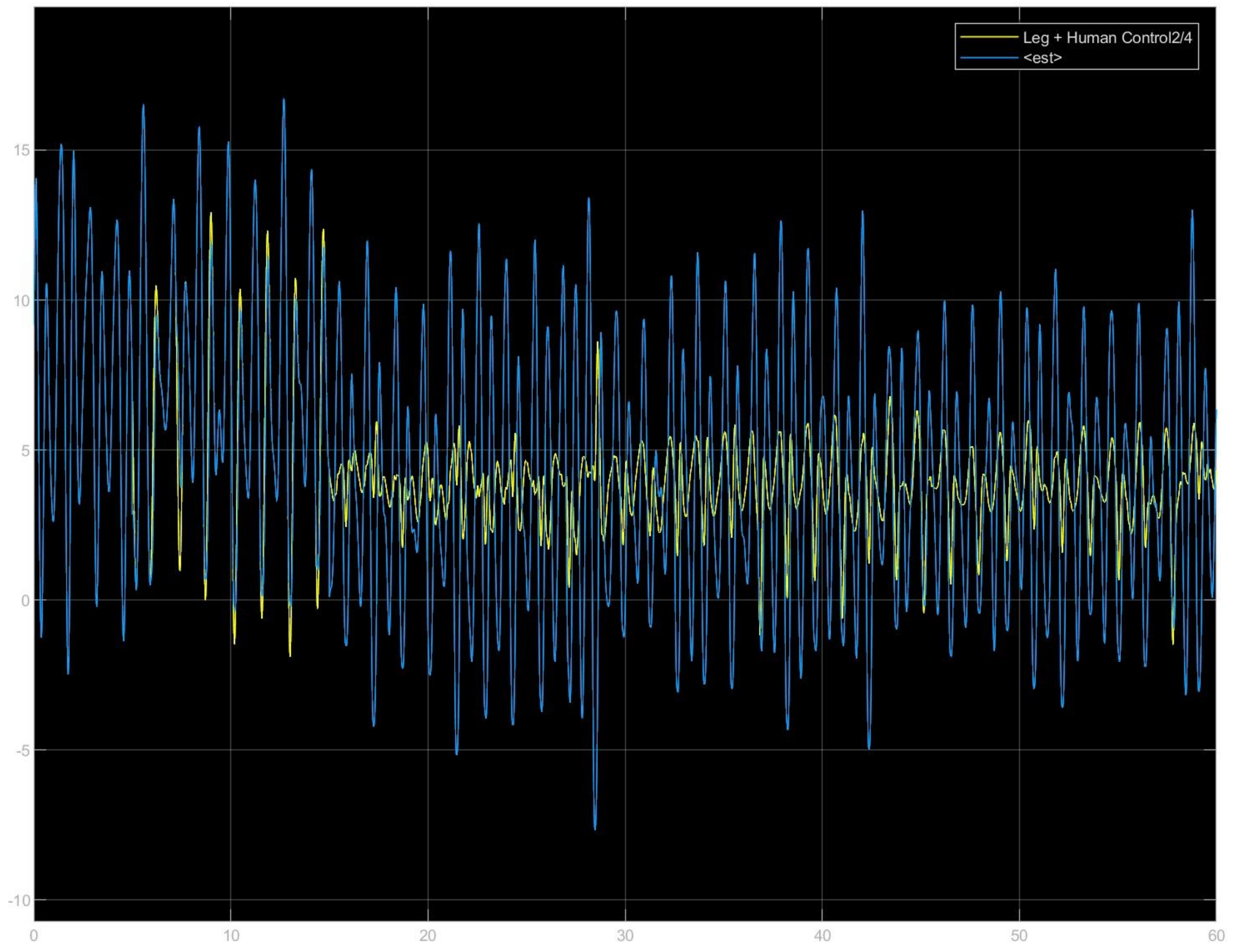
Plotting both the input and output torque values the plots converge within 7 seconds.



Tau_h_hat plotted in yellow and Tau_h from AFO plotted in blue.

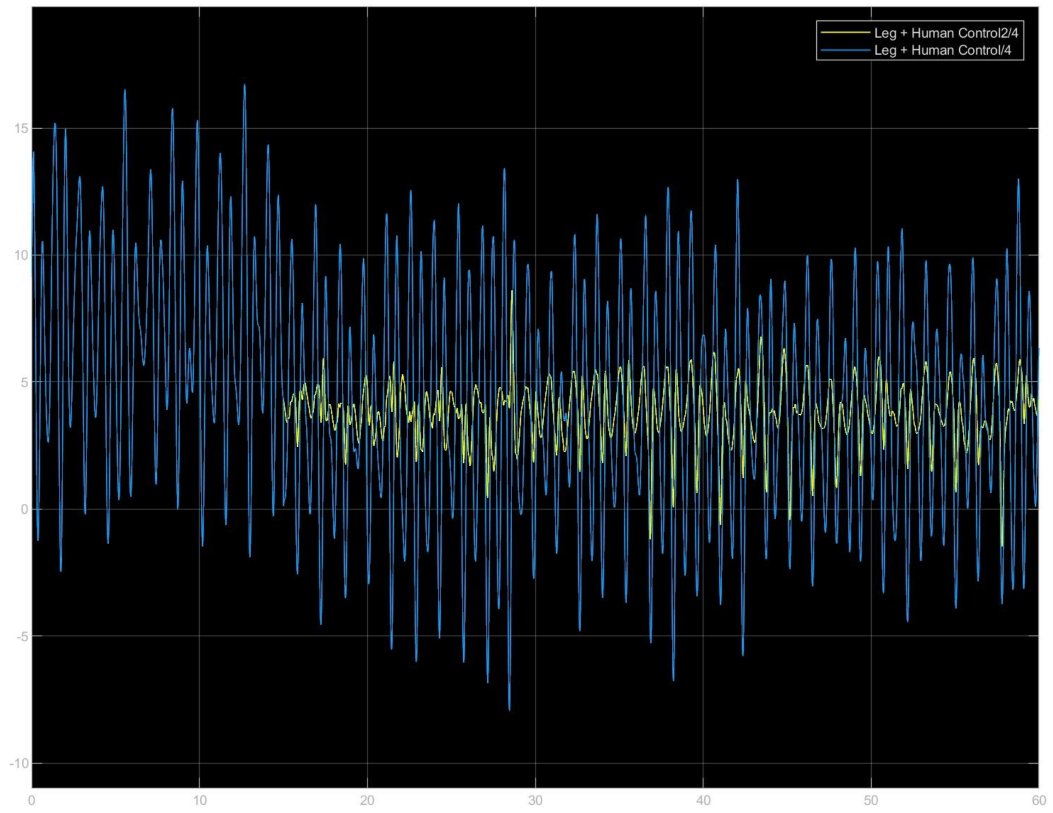
Part A.6

When comparing the input Tau_h hat to the output torque Tau_h, you can see a strong reduction in the required torque once it passes through the controller.



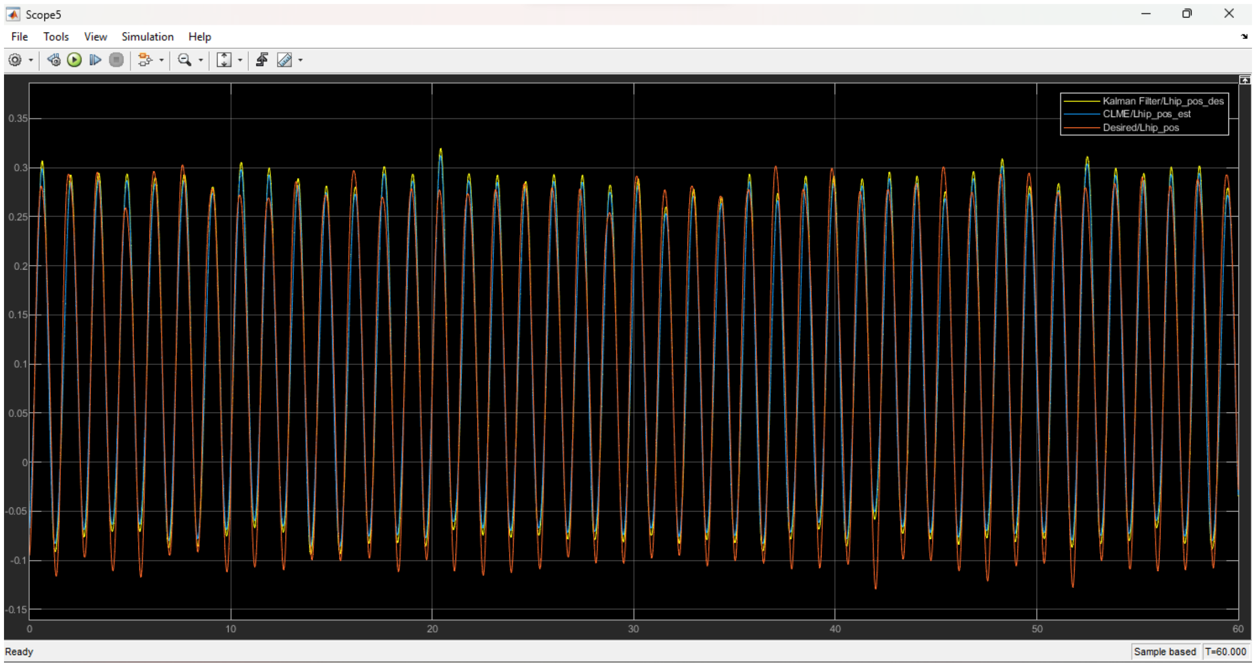
The input torque Tau_h hat in blue and output Tau_h in yellow.

Utilizing the nonlinear filter as a model-free assistive controller, the torque appears to be reduced by more than half when compared to the output of the inverse dynamics based model.

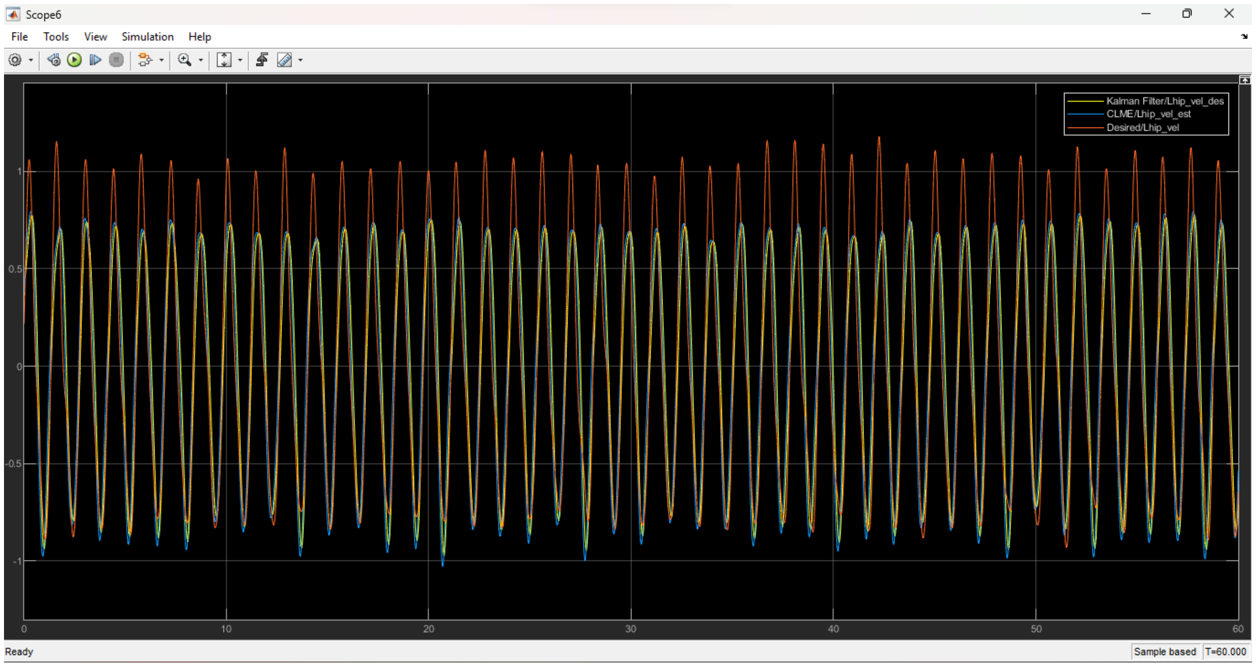


Tau_h from inverse dynamics in Blue and Tau_h from NLF/AFO in yellow.

Part B.7



The plot of the desired joint angle (Red) vs joint angle approximated using CLME (Blue) vs the joint angle position estimated using the KF (Yellow) over time.



The plot of the desired joint angular velocity (Red) vs joint angular velocity approximated using CLME (Blue) vs the joint angle velocity estimated using the KF (Yellow) over time.