ME 494/5594 – Robotics Systems Identification

HW #6: Due Monday, October 31st, 2022

1. Given the continuous-time models below and the measured variables, create the discrete-time state equations and observer ($x_{k+1} = Ax_k + Bu_k$ & $Z_k = H_k x_k$).

   a. $\dot{y}(t) = my(t) + Au(t)$ & states: $\dot{y}(t), y(t),$ measurements of $y(t)$, and input $u(t)$

$$\begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ m & 0 \end{bmatrix} \cdot \begin{bmatrix} y(t-1) \\ \dot{y}(t-1) \end{bmatrix} + \begin{bmatrix} 0 \\ A \end{bmatrix} \cdot [u(t-1)]$$

$$H = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

   b. $x(t) = x(t_0) + \dot{x}(t)\Delta_t + \frac{1}{2}\ddot{x}(t)\Delta_t^2$ & measurements only of $\ddot{x}(t)$ states: $\ddot{x}(t), \dot{x}(t), x(t),$ no input

$$\begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x(t-1) \\ \dot{x}(t-1) \\ \ddot{x}(t-1) \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

   c. $\ddot{\theta} = -\frac{g}{L}sin\theta - C\dot{\theta} + kU + D$ & states: $\ddot{\theta}, \dot{\theta}, \theta$, measurements of θ and θ, input U. Approximate $sin\theta \approx \theta$.

$$\begin{bmatrix} \theta(t) \\ \dot{\theta}(t) \\ \ddot{\theta}(t) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & 0 \\ -\frac{g}{L} & -C & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta(t-1) \\ \dot{\theta}(t-1) \\ \ddot{\theta}(t-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ k \end{bmatrix} \cdot [u(t-1)] + \begin{bmatrix} 0 \\ 0 \\ D \end{bmatrix}$$

$$H = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

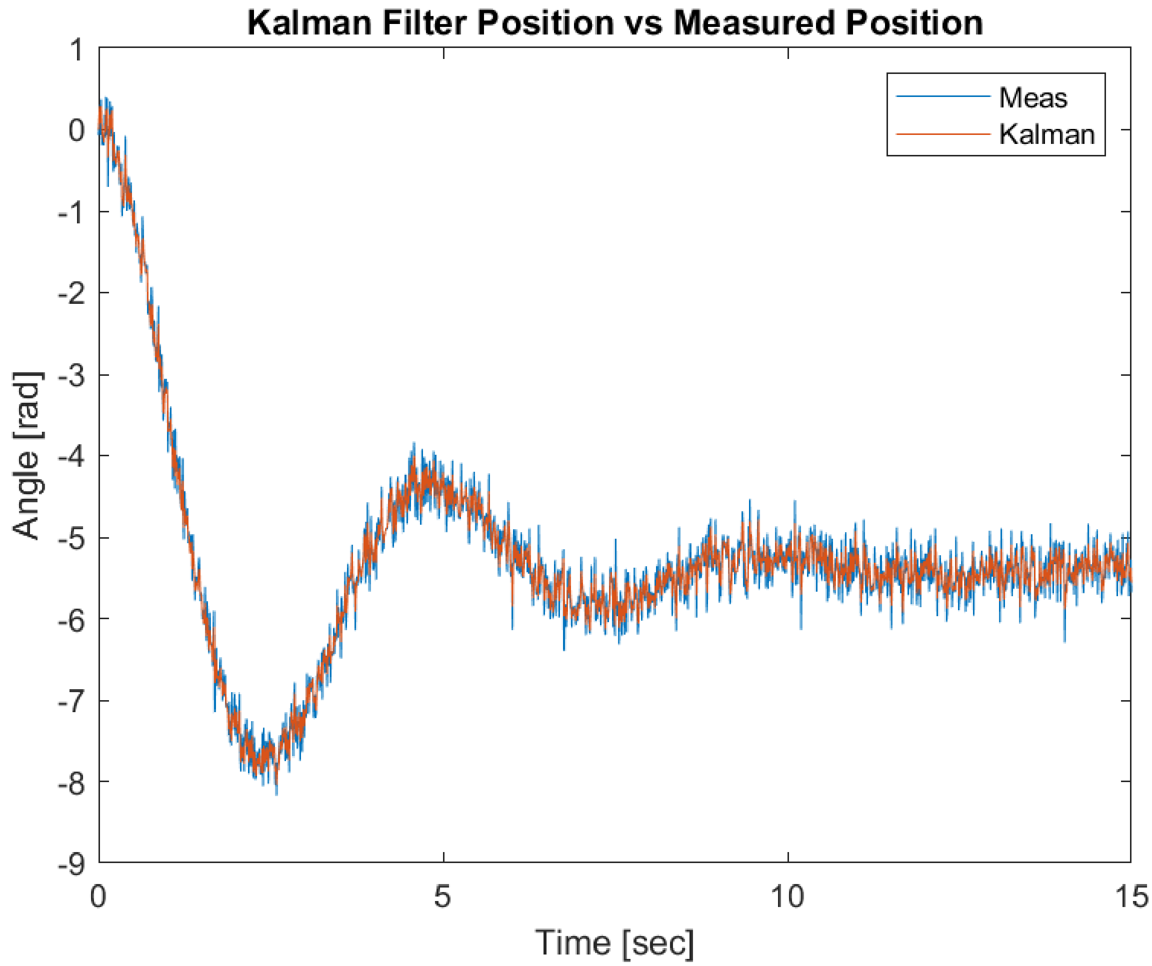
2. Using the Mass-Spring-Damper data posted on Canvas (msd_data_hw6.mat) and a Kalman Filter (KF pendulum code provided on Canvas to assist), estimate the position, velocity, and acceleration. Plot the position and acceleration vs. the measured values.
   Create a second plot that shows the velocity vs. time and a deriv()-based estimate. Comment on what you see. You have measurements of both the position and acceleration.

Use $Q = \begin{matrix} .001 & 0 & 0 \\ 0 & .001 & 0 \\ 0 & 0 & .001 \end{matrix}$ and $R = \begin{matrix} .001 & 0 \\ 0 & 10 \end{matrix}$

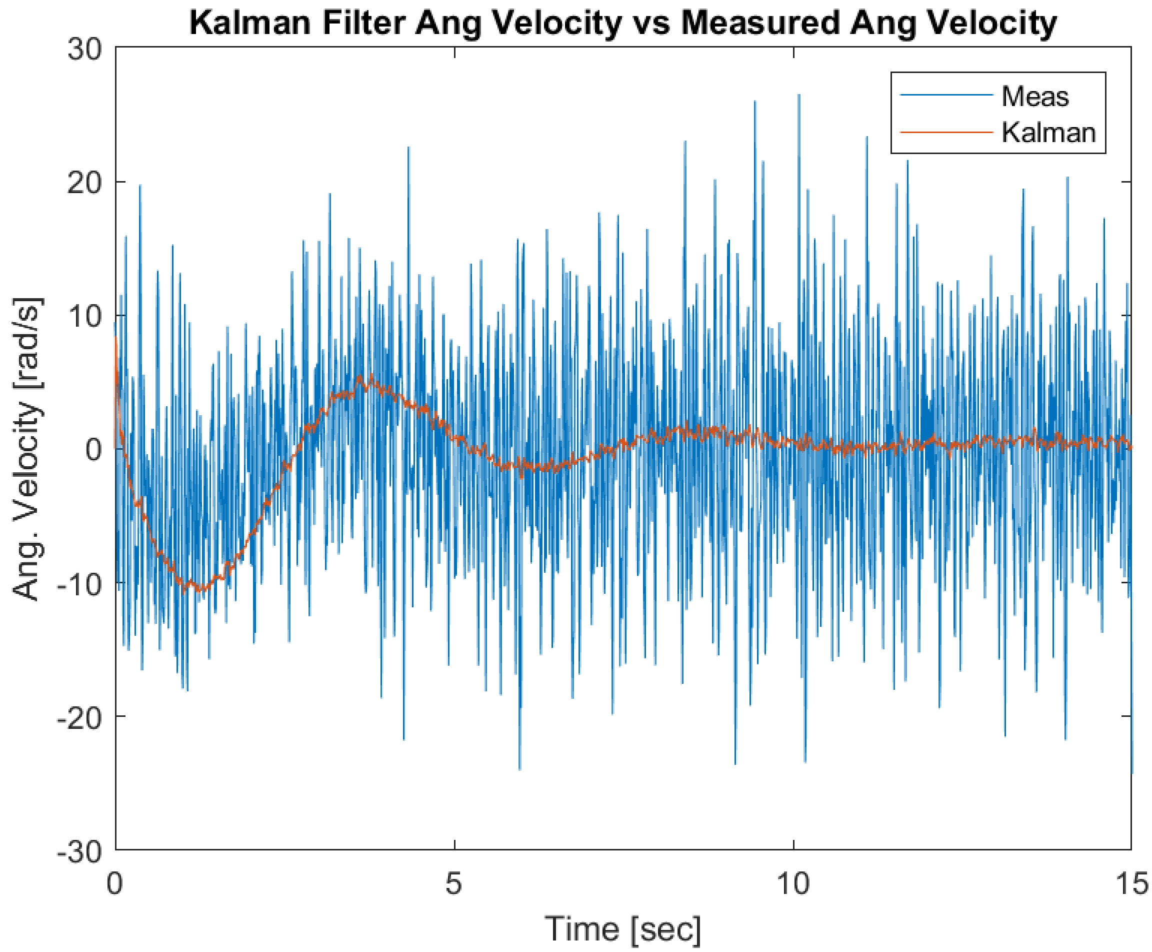
You must provide your $X_{k+1} = A\,X_k + B\,U_{k+1}$ and your H matrix.

$$\begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ -0.0083 & 0.077 & .95 \end{bmatrix} \cdot \begin{bmatrix} x(t-1) \\ \dot{x}(t-1) \\ \ddot{x}(t-1) \end{bmatrix}$$

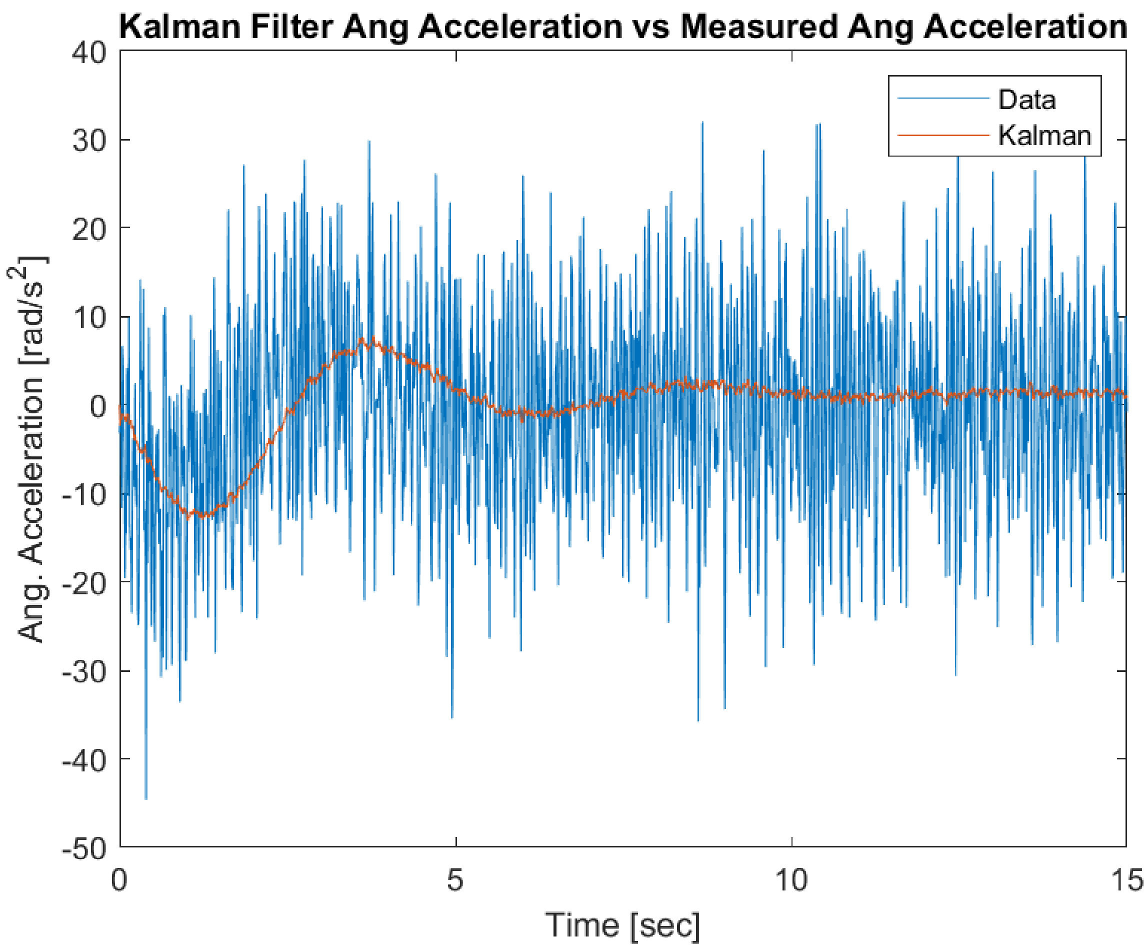
$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Follows the data well but still quite noisy. Might be smoother if the Q and R matrix were modified.



The filtered data is relatively smooth but does not follow the data that well, especially for the first 5 seconds. Might track better if the Q and R matrix were modified.



Similar to the velocity plot, the filtered data is relatively smooth but does not follow the data, especially for the first 5 seconds. Might track better if the Q and R matrix were modified.

3. Using the same dataset, update the acceleration model to utilize a traditional massspring-damper model (gravity pulls the mass down). Repeat the plots from Problem 1. How do the results compare?
    - m = 2.75 kg
    - g = -22.79 m/s² (this includes a bias term)
    - k = -4.2 N/m
    - c = 1.67 N/m-s
    -

$$\ddot{y} = -g + \frac{k}{\ } y + \frac{c}{\ } \dot{y}$$

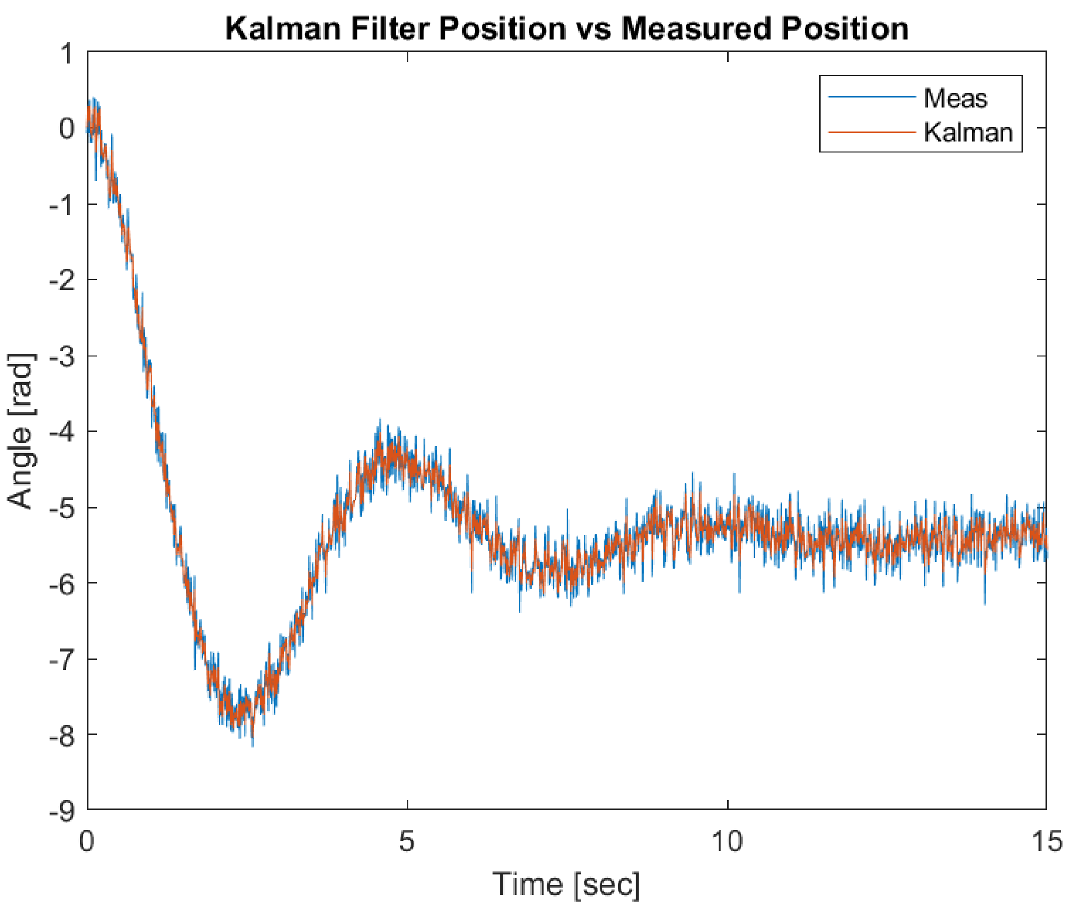
$m \qquad m$

Use the same Q and R matrices from Problem 1.

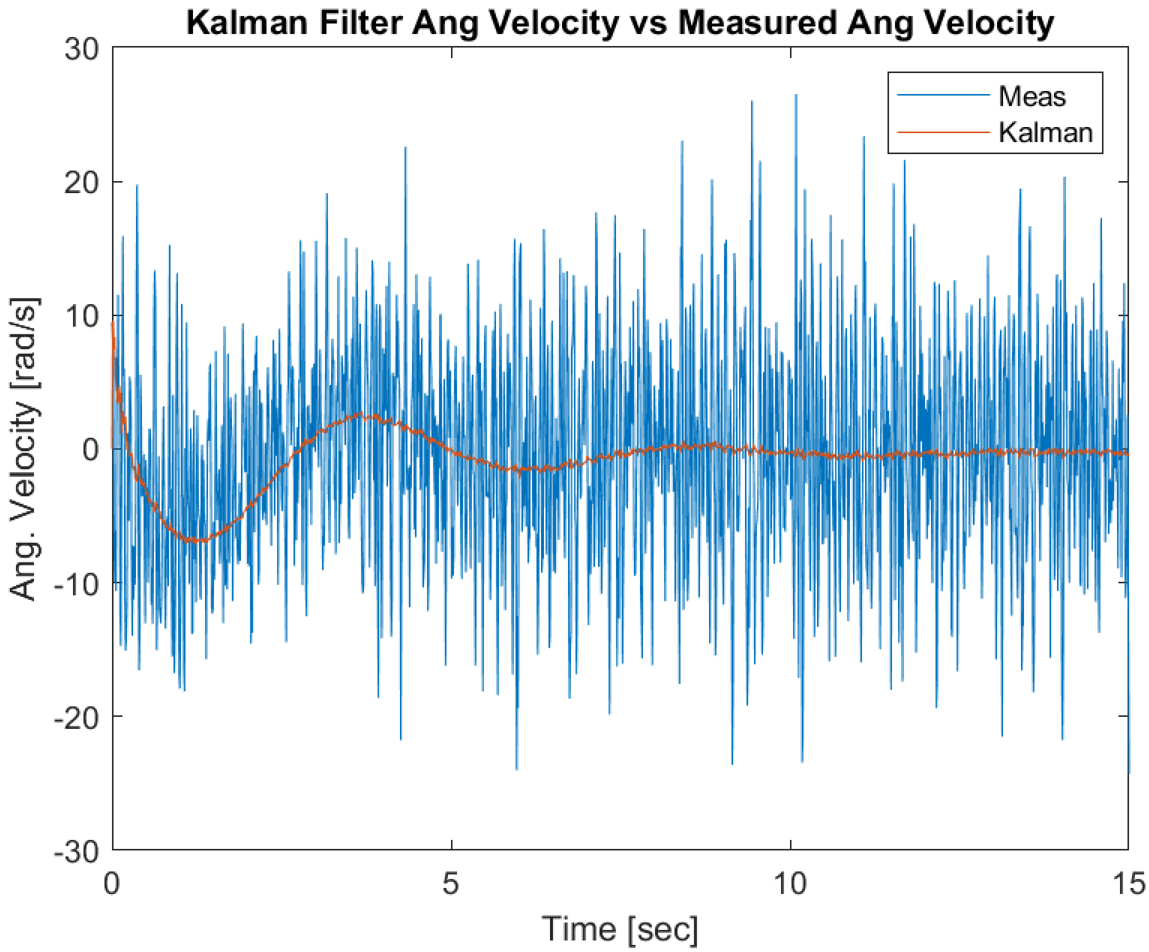
You must provide your $X_{k+1} = A X_k + B U_{k+1}$ and your H matrix.

$$\begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ -1.87 & 0.0483 & 0 \end{bmatrix} \cdot \begin{bmatrix} x(t-1) \\ \dot{x}(t-1) \\ \ddot{x}(t-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -10.42 \end{bmatrix}$$

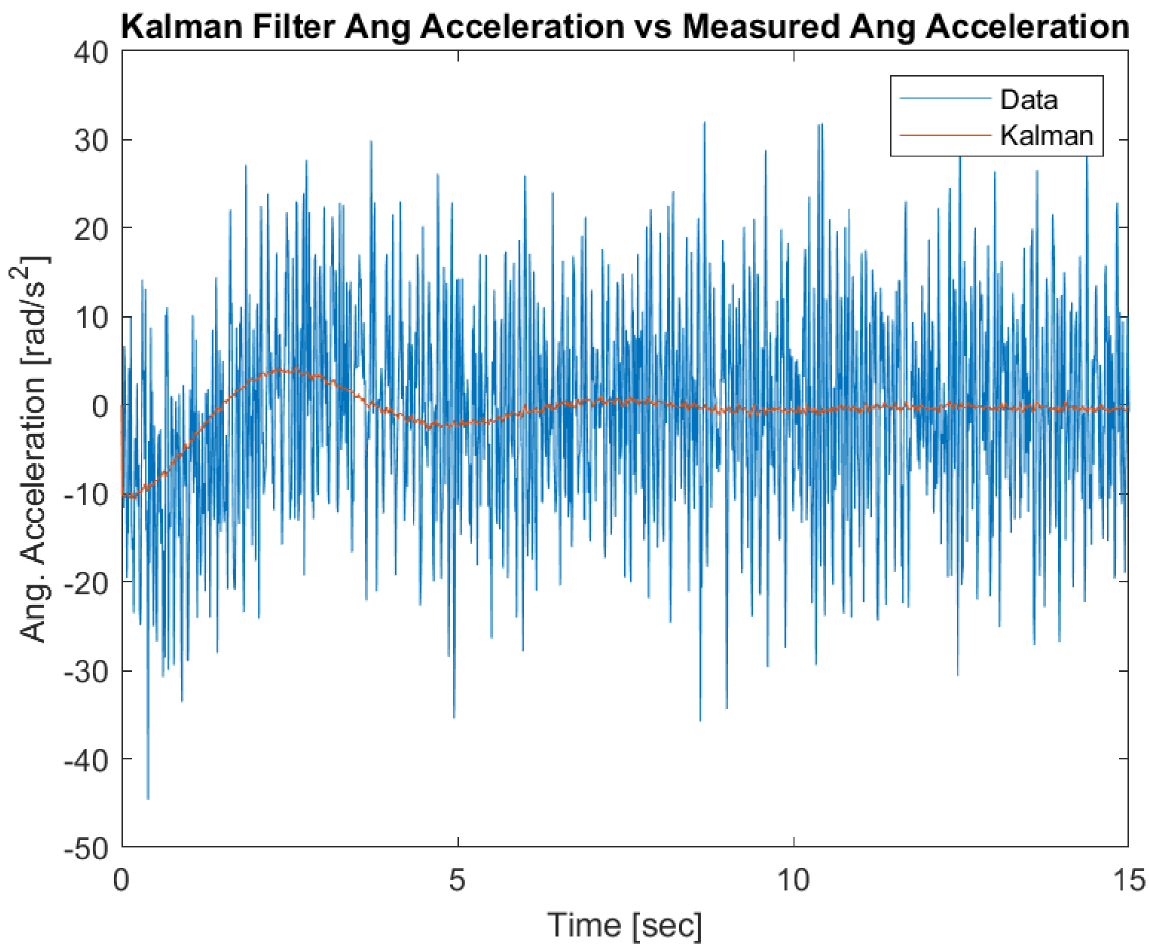
$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Follows the data well but still quite noisy. Might be smoother if the Q and R matrix were modified.



The filtered data is relatively smooth and does follow the data fairly well. The worst tracking is for the first 5 seconds. Might track better if the Q and R matrix were modified.



The filtered data is relatively smooth and does follow the data fairly well. The worst tracking is for the first 5 seconds. Might track better if the Q and R matrix were modified. Does not track quite as well as the filtered velocity signal but the modified model does perform better.

Code:

#2

```
load('msd_data_hw6')

% import data %

time = msd.t;

y = msd.y;

yd = deriv(y,.01);

ydd = msd.ydd;

%% OLS %%

x = [ y, yd];

T_hat = (x'*x)\x'*ydd;

Y_hat = x*T_hat;

%% Setting up the model matrices %%

dt = 0.01;

k = 1;

c1 = T_hat(1,:);

c2 = T_hat(2,:);

c3 = .95;

N = 3;

Xk = [y(k); yd(k); ydd(k)];

A = [1 dt dt^2/2; 0 1 dt; c1 c2 c3];

B = [0 0; 0 0; 0 0];

H = [1 0 0;0 0 1];

P = eye(N)*1.0;

Q = [.001 0 0;

    0  .001 0;
```

```
     0   0  .001];
R = [0.001  0; 0  10];
P_diags(1,:) = diag(P);
%% Kalman %%
for k = 2:length(time)
    X_pred = A*Xk + B*0;
    P_pred = A*P*A' + Q;
    Z = [y(k); ydd(k)];
    yk = Z - H*X_pred;
    Sk = H*P_pred*H' + R;
    Kk = P_pred*H'*Sk^-1;
    Xk = X_pred + Kk*yk;
    P = (eye(N) - Kk*H)*P_pred;
    P_diags(k,:) = diag(P);

    angle_kal(k) = Xk(1);
    rate_kal(k) = Xk(2);
    accel_kal(k) = Xk(3);

end

figure(1)
plot(time, y, time, angle_kal)
title('Kalman Filter Position vs Measured Position')
xlabel('Time [sec]')
ylabel('Angle [rad]')
```

```
legend('Meas','Kalman')



figure(2)
plot(time, yd, time, rate_kal)
title('Kalman Filter Ang Velocity vs Measured Ang Velocity')
xlabel('Time [sec]')
ylabel('Ang. Velocity [rad/s]')
legend('Meas','Kalman')

figure(3) % Comparing derivative of rate data to KF accel output
plot(time, ydd, time, accel_kal)
title('Kalman Filter Ang Acceleration vs Measured Ang Acceleration')
xlabel('Time [sec]')
ylabel('Ang. Acceleration [rad/s^2]')
legend('Data','Kalman')

#3
load('msd_data_hw6')
% import data %
time = msd.t;
N = length(time);
bias = ones(N,1);
y = msd.y;
yd = deriv(y,.01);
ydd = msd.ydd;
```

```
%% OLS %%

x = [ones(N,1), y, yd];

T_hat = (x'*x)\x'*ydd;

Y_hat = x*T_hat;

%% Setting up the model matrices %%

dt = 0.01;

k = 1;

c1 = T_hat(1,:);

c2 = T_hat(2,:);

c3 = T_hat(3,:);

N = 3;

Xk = [y(k) ; yd(k); ydd(k)];

A = [1 dt dt^2/2; 0 1 dt; c2 c3 0];

B = [0 0; 0 0; 0 0];

C = [ 0; 0; c1];

H = [1 0 0;0 0 1];

P = eye(N)*1.0;

Q = [.001 0 0;

     0  .001 0;

     0   0  .001];

R = [0.001  0; 0  10];

P_diags(1,:) = diag(P);

%% Kalman %%

for k = 2:length(time)

    X_pred = A*Xk + B*0 + C;

    P_pred = A*P*A' + Q;
```

```
    Z = [y(k); ydd(k)];

    yk = Z - H*X_pred;

    Sk = H*P_pred*H' + R;

    Kk = P_pred*H'*Sk^-1;

    Xk = X_pred + Kk*yk;

    P = (eye(N) - Kk*H)*P_pred;

    P_diags(k,:) = diag(P);


    angle_kal(k) = Xk(1);

    rate_kal(k) = Xk(2);

    accel_kal(k) = Xk(3);


end


figure(1)

plot(time, y, time, angle_kal)

title('Kalman Filter Position vs Measured Position')

xlabel('Time [sec]')

ylabel('Angle [rad]')

legend('Meas','Kalman')



figure(2)

plot(time, yd, time, rate_kal)

title('Kalman Filter Ang Velocity vs Measured Ang Velocity')

xlabel('Time [sec]')
```

```
ylabel('Ang. Velocity [rad/s]')
legend('Meas','Kalman')


figure(3) % Comparing derivative of rate data to KF accel output
plot(time, ydd, time, accel_kal)
title('Kalman Filter Ang Acceleration vs Measured Ang Acceleration')
xlabel('Time [sec]')
ylabel('Ang. Acceleration [rad/s^2]')
legend('Data','Kalman')
```