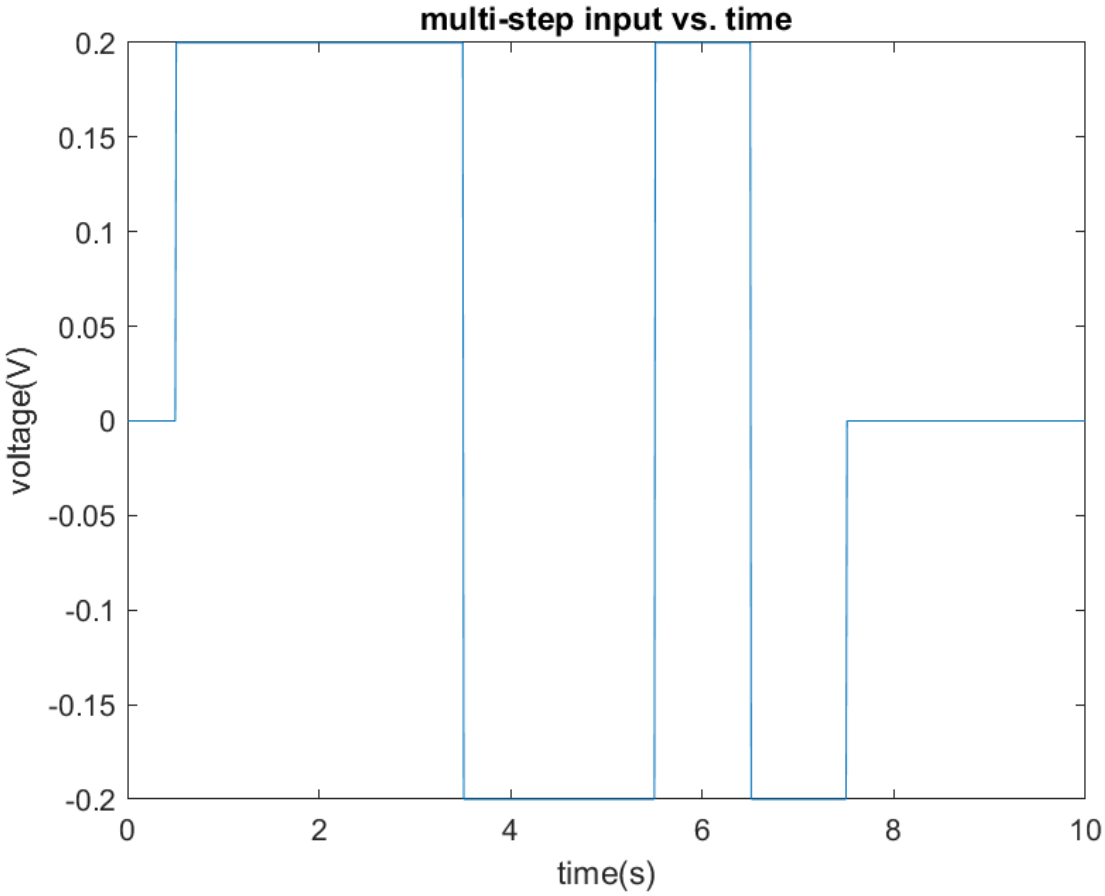
ME 494/5594 – Robotics Systems Identification

HW #5: Due Wednesday November 9th, 2022

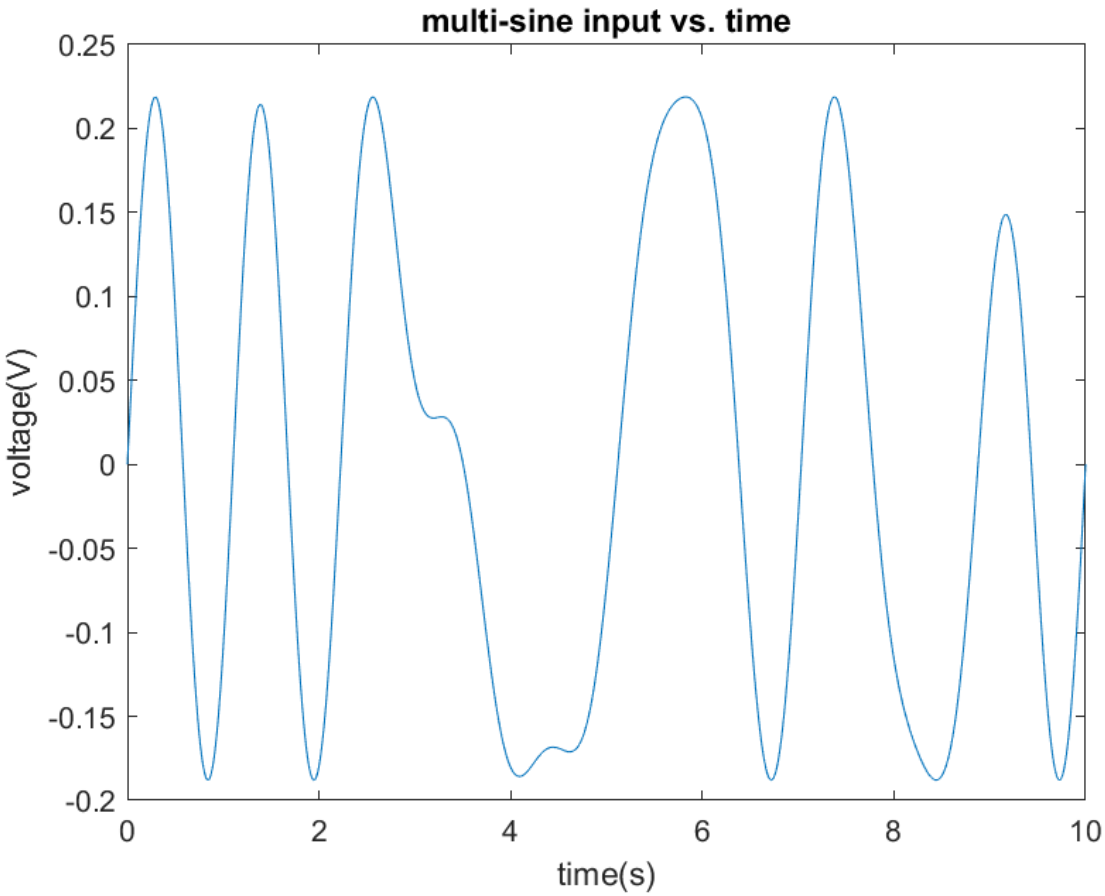
Using your Quanser Qube data collected in the lab, complete the problem(s) below.

1. Create a multi-step (3-2-1-1) input signal. Plot the input design. Comment/discuss your design. Why did you select the amplitude and periods? **Make sure your code saves the input design into the "steve" variables as shown in lab.**



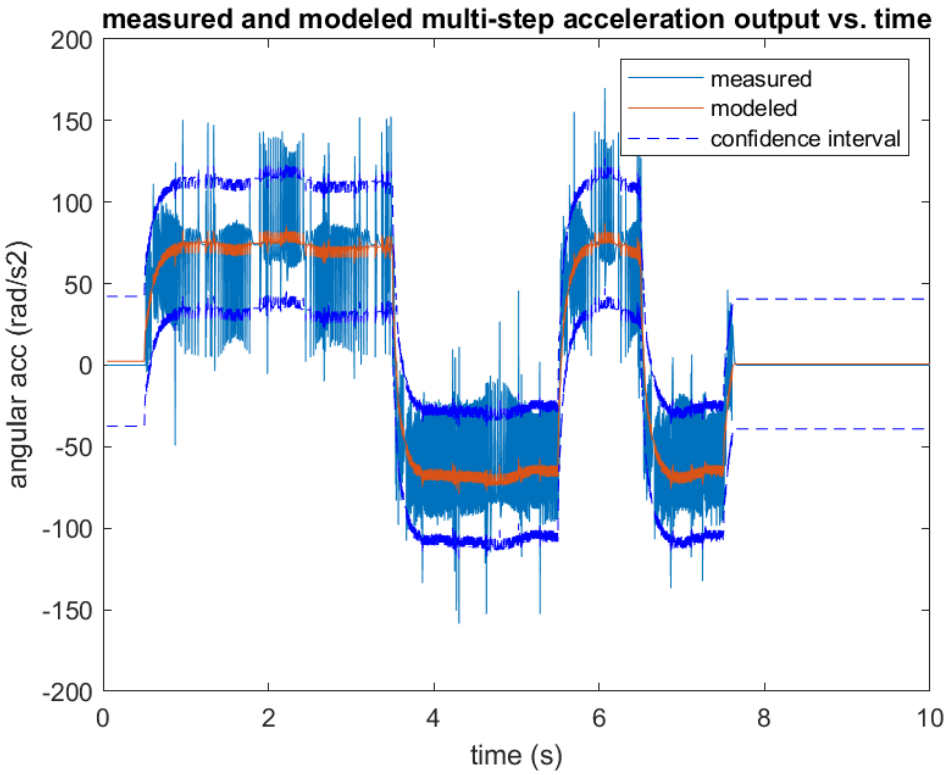
The amplitude of .2 V was selected because a low voltage signal resulted in a slow enough motor speed that the motor would easily follow the input signal. A period of 10 seconds was chosen to incorporate the requested pulse widths.

2. Create a multisine input signal. Plot the input design. Comment/discuss your design. Why did you select the amplitude, minimum and maximum frequencies, and signal length? **Make sure your code saves the input design into the "steve" variables as shown in lab.**



The amplitude of .2 V was selected because a low voltage signal resulted in a slow enough motor speed that the motor would easily follow the input signal. The length of the signal was chosen as it would incorporate at least 1 period of the signal and match the multi-step signal length. A low and small frequency range was chosen as it is sufficiently large to perform a frequency-domain Least Squares regression.

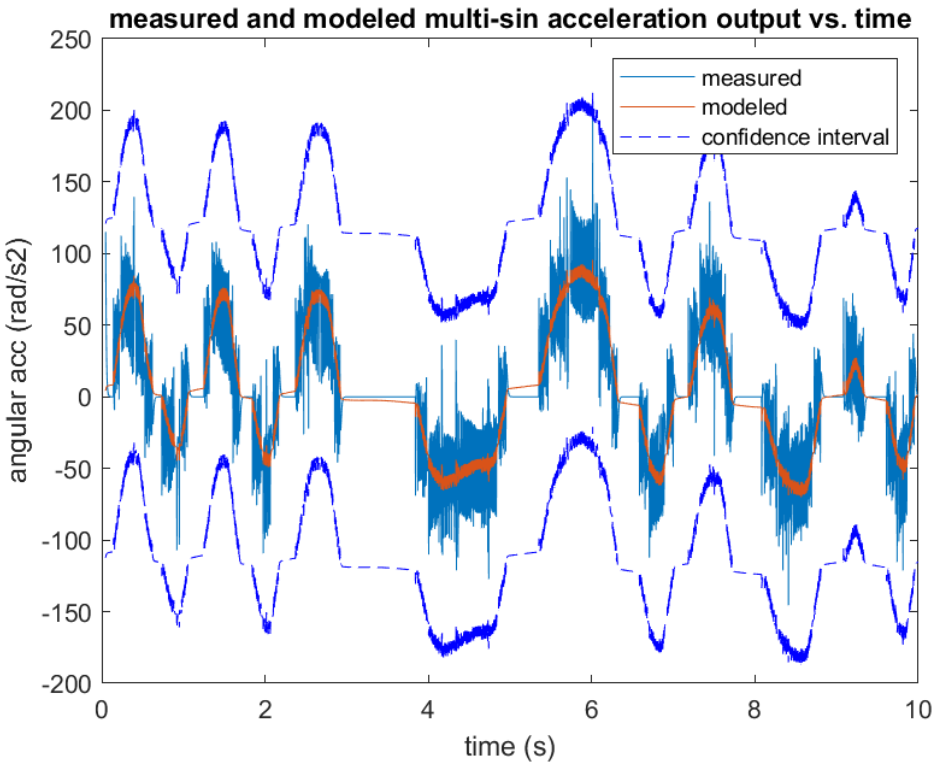
3. Using the multi-step and multisine data collected from the lab (using the input signal provided), estimate a model using OLS of the Qube motion. Use angular acceleration as your dependent variable (z). Show your model structure (with estimated coefficients). Show a plot of the model output and the actual vs. time. Add the confidence bounds onto the model output. What is the Coefficient of Determination of the model fit?



Model Structure with coefficients found with OLS from multi-step data:

$$\ddot{\theta} = 2.4305 + 93.9382 * V - .4279 * \theta + 15.5386 * \dot{\theta}$$

The multi-step R squared value: 0.8887



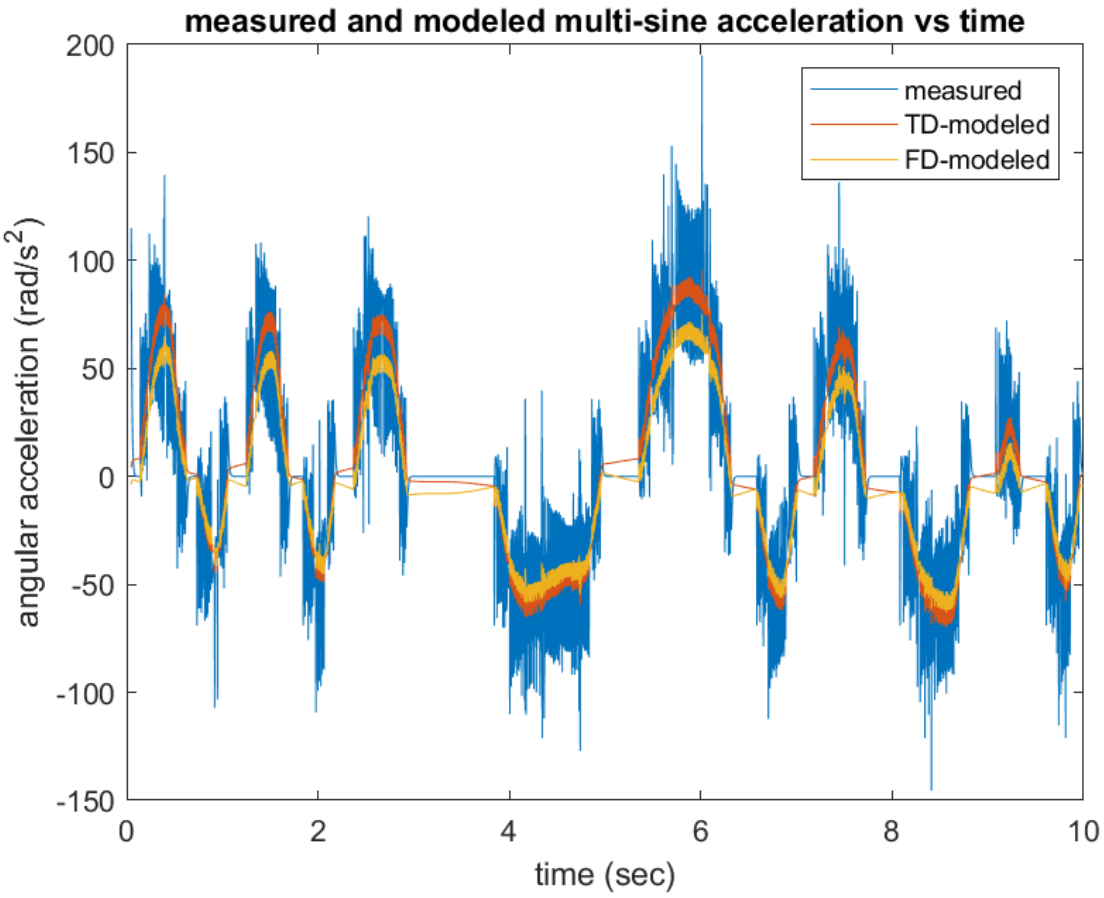
Model Structure with coefficients found with OLS from multi-sine data:

$$\ddot{\theta} = 6.7445 + 11.6253 * V - 4.6440 * \theta + 20.7231 * \dot{\theta}$$

The multi-sine R squared value:

0.7923

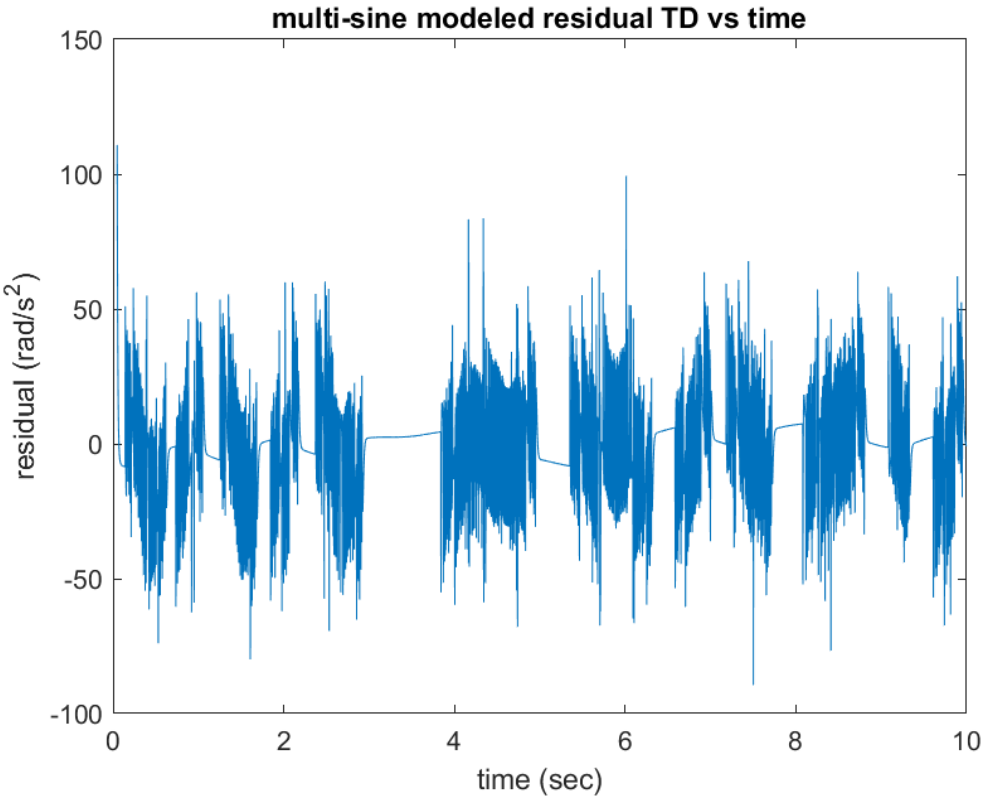
4. Using the multi-step and multisine data, perform a **frequency-domain** Least Squares regression. You need to compare your frequency-domain estimate with a time-domain OLS. Show plots of the model outputs, and residuals. Comment on what your results show (short discussion). You **cannot** use the FFT function to compute the frequency-domain data.

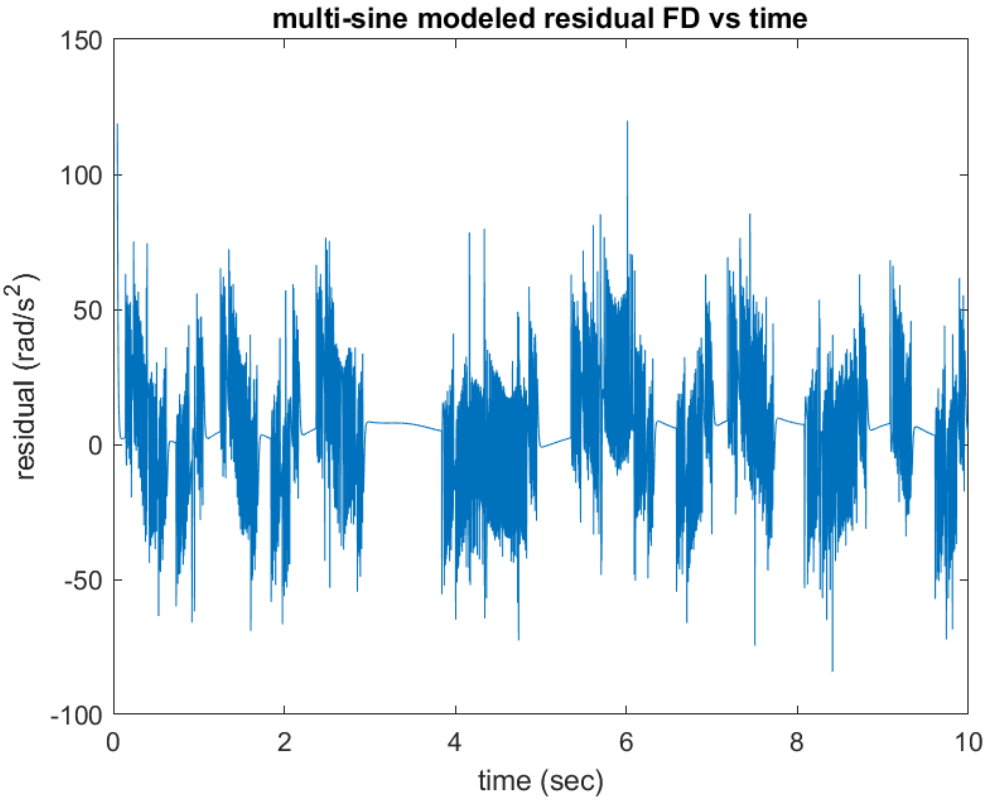


The FD model does not track the data quite as well as the TD model.

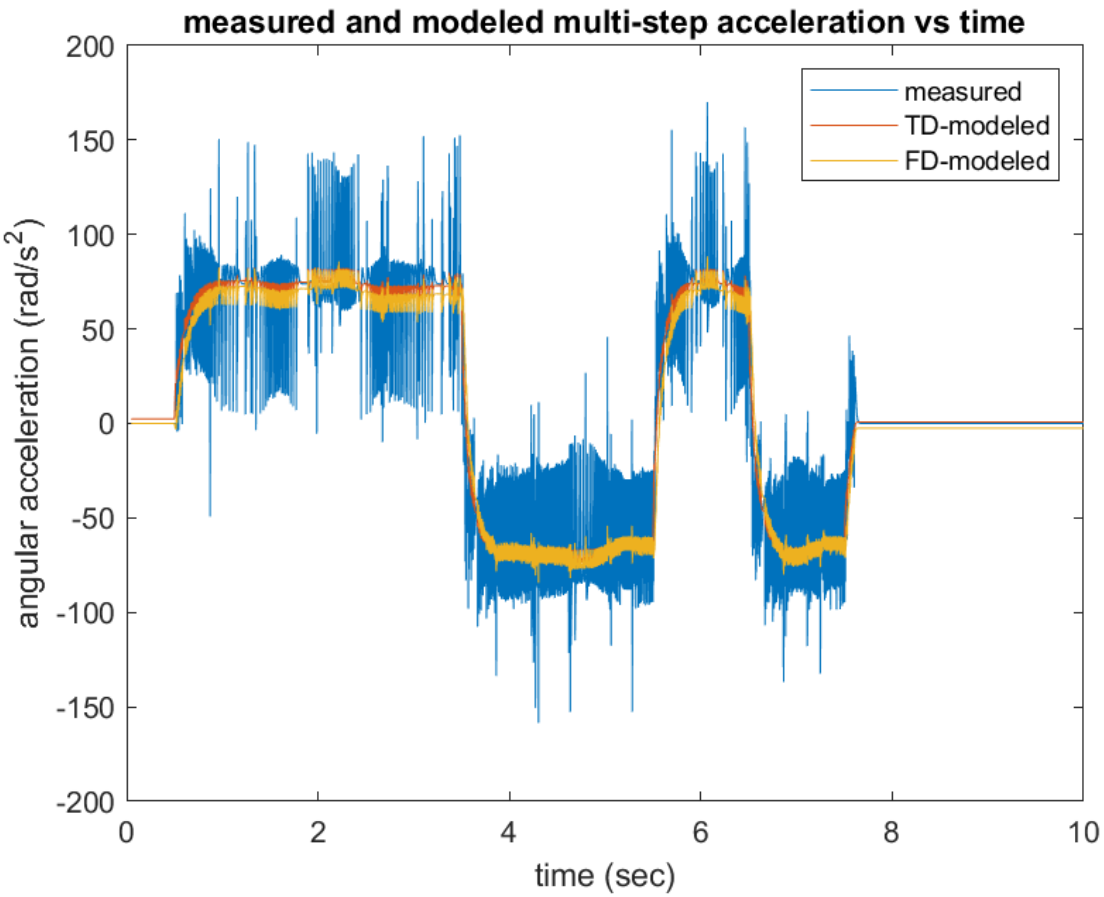
The multi-sine R squared value:

0.6304





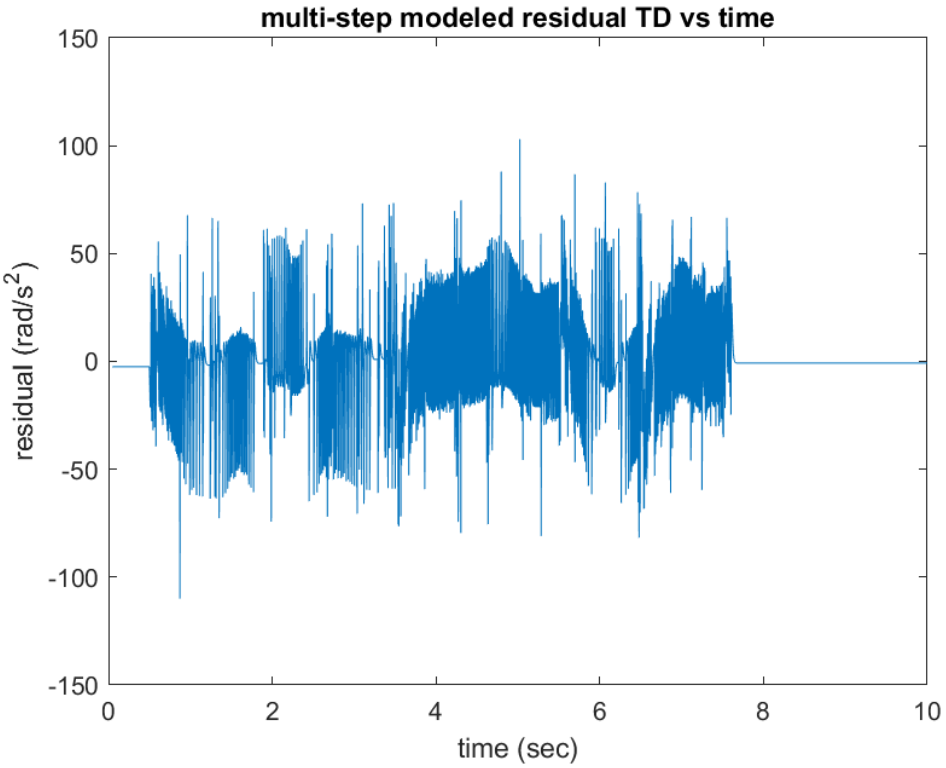
The R2 is quite low at .6304 therefore residual is likely to be substantial. The substantial amount of noise also contributed greatly to the residual.

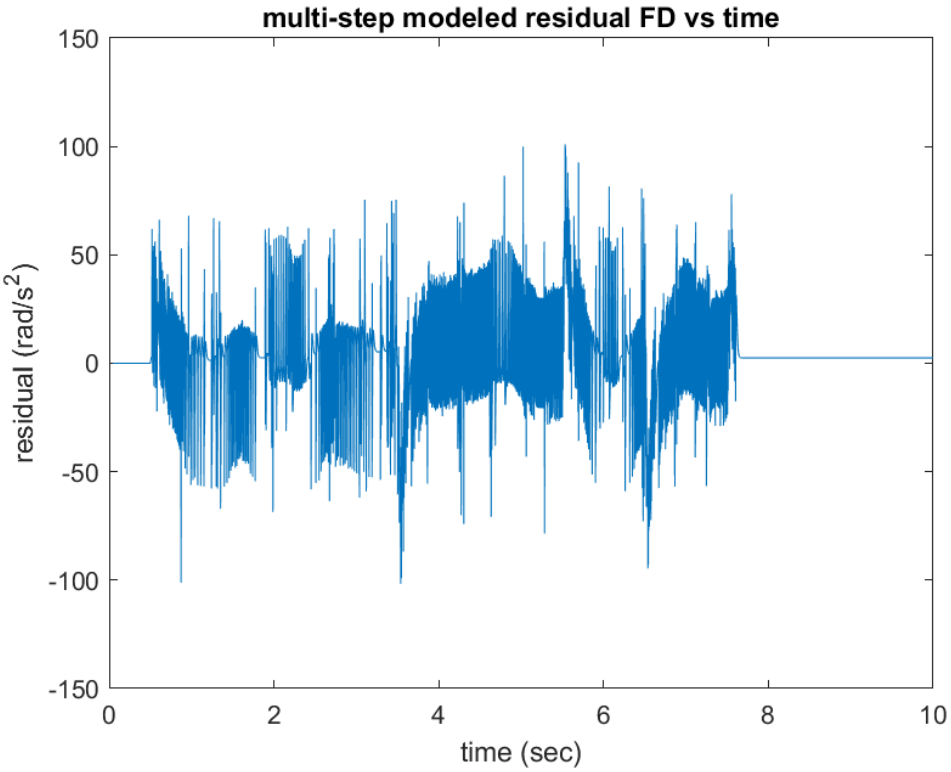


The FD model does not track the data quite as well as the TD model.

The multi-sine R squared value:

0.8563

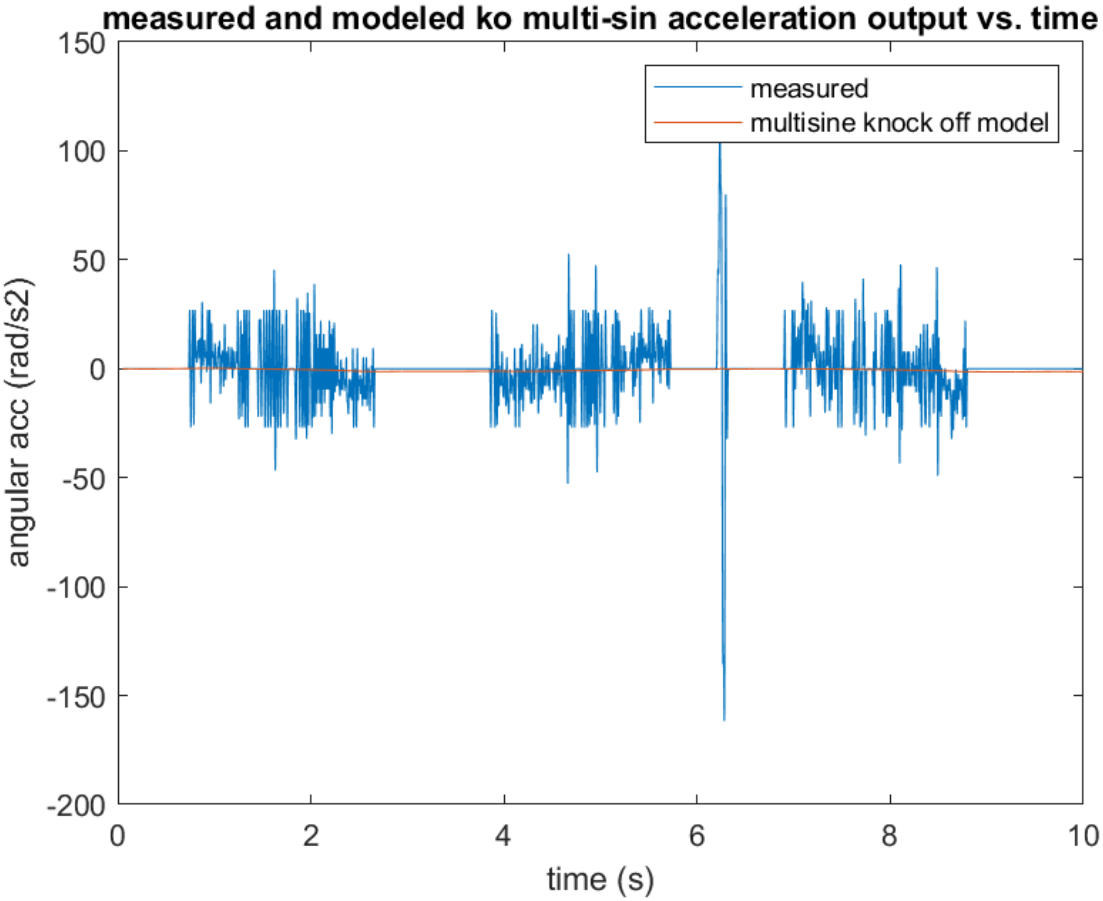




The R2 is decent at 0.8563. The substantial amount of noise contributed greatly to the residual.

5. Using your multisine input (with Voltage = +/- 0.2V), conduct another experiment where you knock the weight off halfway through the test. Perform a **real-time frequency-domain least squares** regression. Show a plot of the model fit using coefficients from the first fit, before the weight knockoff, and ¾ of the way through the test input. Do the models fit the entire datafile (reminder that they shouldn't, as there was a change part way through)?

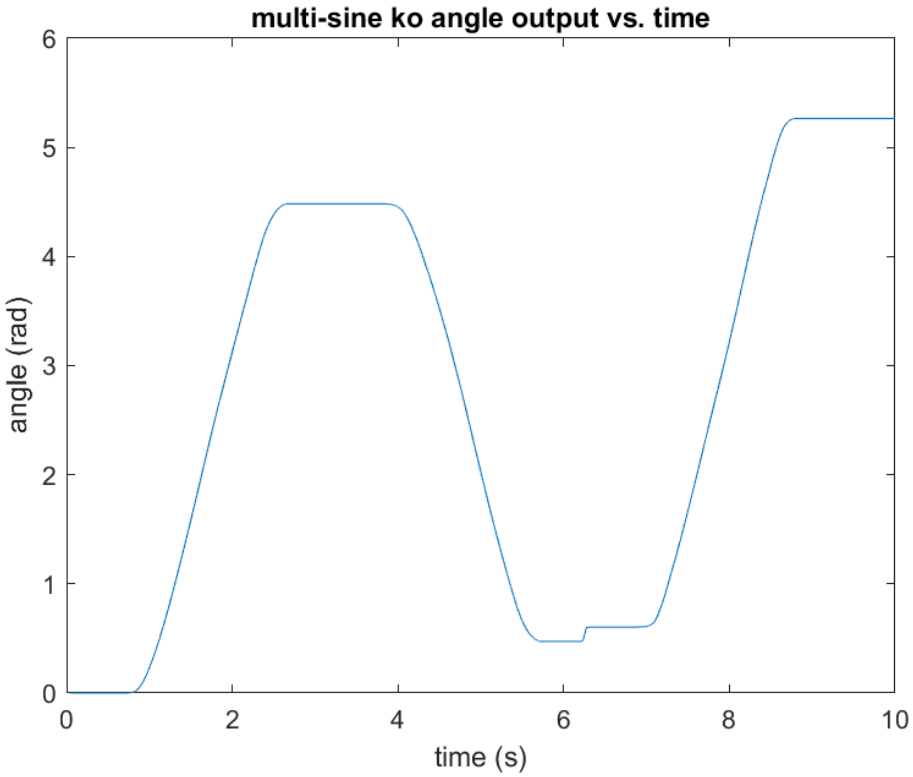
    a. Show a subplot of the coefficients vs. time. Can you identify where the weight was knocked off in the coefficients? Do the value changes make sense?

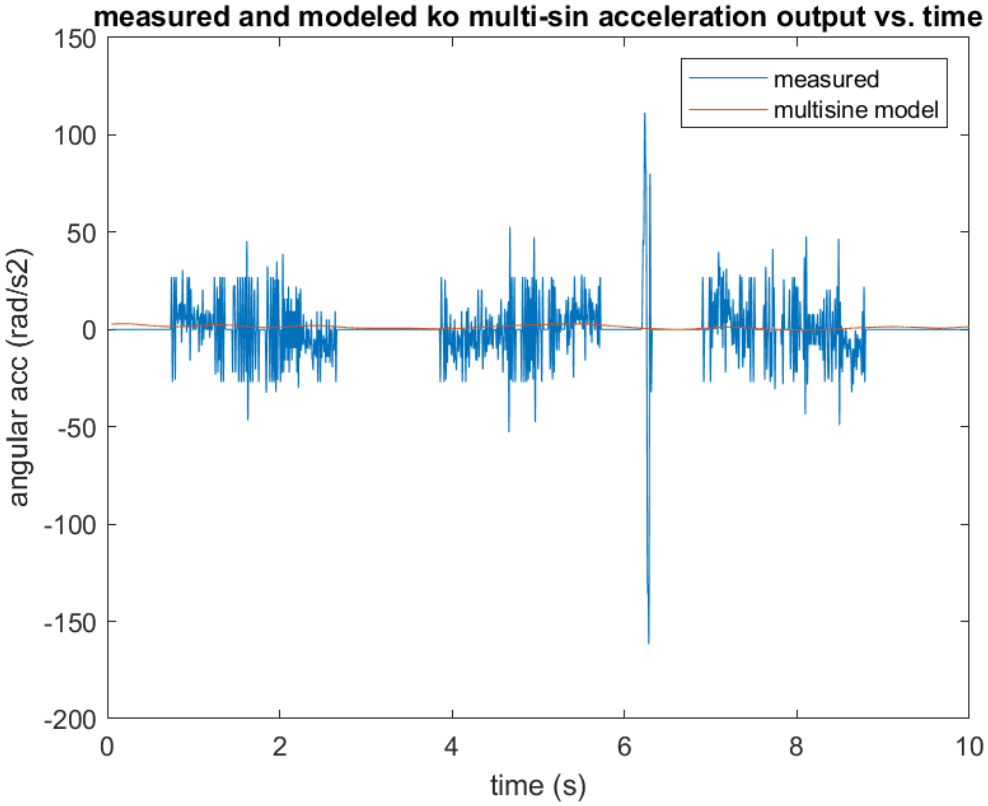


The multi-sine knock-off R squared value:

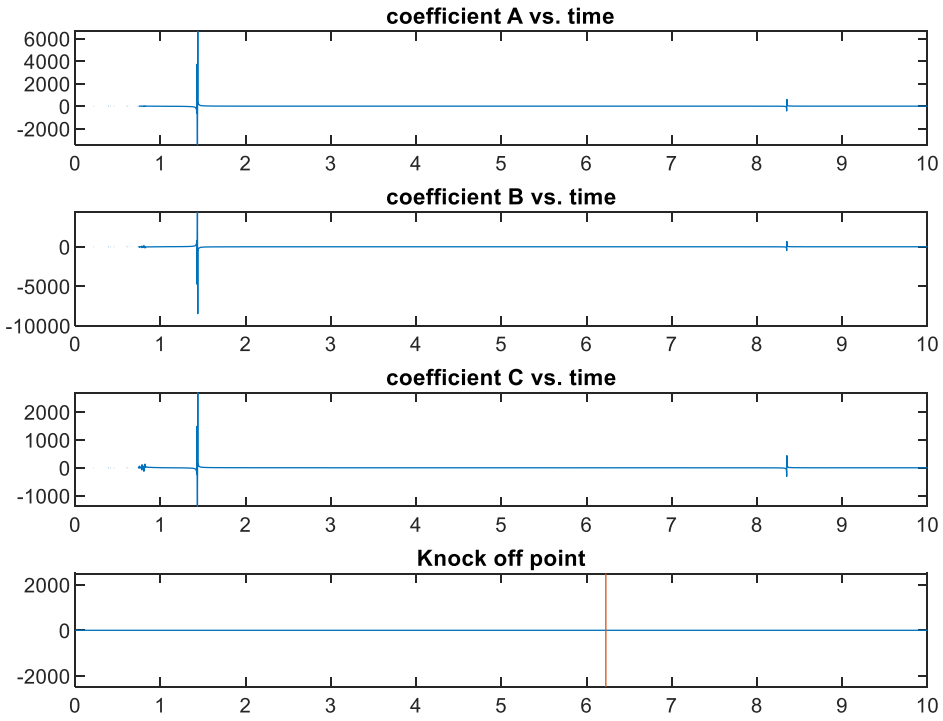
0.0050

With an R2 value so low I wouldn't expect the model to have any utility. I thought the model would follow measured value after the knock off at 6.2 seconds (shown below) but it doesn't.





Using the original multi-sine model with the knock off experiment data does seem to follow the measured data any better. I would have suspected that it would have followed the signal decently well before the weight was knocked off at 6.2 seconds but it does not.



The convergence happens quickly. I would have assumed that the plots would have to reconverge after 6.2 seconds, when the weight was knocked off, but they do not.

CODE:

```
load("ME_494_HW7_sine_input")
load("ME_494_HW7_step_input")
load("ME_494_HW7_sine_data")
load("ME_494_HW7_step_data")
load("ME_494_HW7_sine_data_KO")
load("ME_494_HW7_sine_data_KO_2")

%time%
t = simout_sine.time;
% inputs %
sin_in = simout_sine.signals.values(:,1);
step_in = simout_step.signals.values(:,1);

% angle %
sin_ang = simout_sine.signals.values(:,2);
step_ang = simout_step.signals.values(:,2);
%sin_ang_ko = simout_sine_ko.signals.values;
sin_ang_ko2 = table2array(sin_ang_ko2);
% angular velocity %
sin_vel = simout_sine.signals.values(:,3);
step_vel = simout_step.signals.values(:,3);

% angular accelaeration %
sin_acc = simout_sine.signals.values(:,4);
step_acc = simout_step.signals.values(:,4);

% cutting the first np points %
start = 25;
np = start;
% finish = length(t)/2;
% en = finish;
%time%
t = t(np:end,1);

% inputs %
sin_in = sin_in(np:end,1);
step_in = step_in(np:end,1);

% angle %
sin_ang = sin_ang(np:end,1);
step_ang = step_ang(np:end,1);

% angular velocity %
```

```
sin_vel = sin_vel(np:end,1);
step_vel = step_vel(np:end,1);

% angular accelaeration %
sin_acc = sin_acc(np:end,1);
step_acc = step_acc(np:end,1);

N = length(t);
bias = ones(N,1);
dt =.002;
% % for knockoff sine data %
% sin_ang_ko = sin_ang_ko(np:end,1);
% sin_vel_ko = deriv(sin_ang_ko,.002);
sin_ang_ko = sin_ang_ko2(np:end,1);
sin_vel_ko = deriv(sin_ang_ko,.002);
sin_vel_ko = smooth(sin_vel_ko);
sin_acc_ko = deriv(sin_vel_ko,.002);
sin_acc_ko = smooth(sin_acc_ko);


% OLS for Qube with step data %
x = [bias, step_in, step_ang, step_vel];
T_hat =(x'*x)\x'*step_acc;
Y_hat = x*T_hat;
T_hat_step = T_hat
disp('The step R squared value: ')
v = step_acc - Y_hat;
s_sq = sum(v.^2)/(length(v) -length(T_hat));
R_sq_step = (T_hat'*x'*step_acc - length(step_acc)*mean(step_acc)^2) /
(step_acc'*step_acc - length(step_acc) * mean(step_acc)^2)

% OLS for Qube with sin data %
x2 = [bias, sin_in, sin_ang, sin_vel];
T_hat2 =(x2'*x2)\x2'*sin_acc;
Y_hat2 = x2*T_hat2;
T_hat_sin = T_hat2
disp('The sine R squared value: ')
v2 = sin_acc - Y_hat2;
s_sq2 = sum(v2.^2)/(length(v2) -length(T_hat2));
R_sq_sin = (T_hat2'*x2'*sin_acc - length(sin_acc)*mean(sin_acc)^2) /
(sin_acc'*sin_acc - length(sin_acc) * mean(sin_acc)^2)

% OLS for Qube with sin ko data %
x3 = [bias, sin_in, sin_ang_ko, sin_vel_ko];
```

```
T_hat3 =(x3'*x3)\x3'*sin_acc_ko;
Y_hat3 = x3*T_hat3;
T_hat_sin_ko = T_hat3
disp('The sine ko R squared value: ')
v3 = sin_acc_ko - Y_hat3;
s_sq3 = sum(v3.^2)/(length(v3) -length(T_hat3));
R_sq_sin_ko = (T_hat3'*x3'*sin_acc_ko -
length(sin_acc_ko)*mean(sin_acc_ko)^2) / (sin_acc_ko'*sin_acc_ko -
length(sin_acc_ko) * mean(sin_acc_ko)^2)

% DFT multisine %

z2 = sin_acc;
[ro,col] = size(x2);
xrt = zeros(ro,col);
for j = 1:col
   xrt(:,j) = x2(:,j) - mean(x2(:,1));
end
zrt = z2 - mean(z2);
%Frequency Term
g=sqrt(-1);
%Frequency Range
freq = [0.1:0.1:1];
%Frequency Factor (rad/s)
w = freq*(3.14/180);
C = exp(-g*w*t(1));
dC = exp(-g*w*dt);
%Length of Frequency
fl =length(freq);
zf = zeros(fl,1);
xf = zeros(size(w,2), col);
R=1.0;
Np=4;
for i=2:ro
   xcurr = xrt(i,:);
   zcurr = zrt(i);
   xf = R*xf + C'*xcurr;
   zf = zf + (zcurr'*C)';
   C = C.*dC;
   T_hats(i,:) = real(xf'*xf)\ real(xf'*zf);
end
% # 4 DFT with sine data
reg_xf = [xf(:,2),xf(:,3),xf(:,4)];
fix_x = [x2(:,2),x2(:,3),x2(:,4)];
```

```
T_hat_fd = real(reg_xf'*reg_xf)\ real(reg_xf'*zf)
Y_hat_fd = fix_x*T_hat_fd;
% R2
Nfd = length(t);
R2_FD_sin = (T_hat_fd'*fix_x'*z2 - Nfd*mean(z2)^2) / (z2'*z2 -
Nfd*mean(z2)^2)

% DFT multistep %

z = step_acc;
[ro,col] = size(x);
xrt2 = zeros(ro,col);
for j = 1:col
   xrt2(:,j) = x(:,j) - mean(x(:,1));
end
zrt2 = z - mean(z);
%Frequency Term
g2=sqrt(-1);
%Frequency Range
freq2 = [0.1:0.1:1];
%Frequency Factor (rad/s)
w2 = freq2*(3.14/180);
C2 = exp(-g2*w2*t(1));
dC2 = exp(-g2*w2*dt);
%Length of Frequency
fl2 =length(freq);
zf2 = zeros(fl2,1);
xf2 = zeros(size(w2,2), col);
R2=1.0;
Np2=4;
for i=2:ro
   xcurr2 = xrt2(i,:);
   zcurr2 = zrt2(i);
   xf2 = R2*xf2 + C2'*xcurr2;
   zf2 = zf2 + (zcurr2'*C2)';
   C2 = C2.*dC2;
   T_hats2(i,:) = real(xf2'*xf2)\ real(xf2'*zf2);
end
% # 4 DFT with step data
reg_xf2 = [xf2(:,2),xf2(:,3),xf2(:,4)];
fix_x2 = [x(:,2),x(:,3),x(:,4)];

T_hat_fd2 = real(reg_xf2'*reg_xf2)\ real(reg_xf2'*zf2)
```

```
Y_hat_fd2 = fix_x2*T_hat_fd2;
% R2
Nfd2 = length(t);
R2_FD_step = (T_hat_fd2'*fix_x2'*z - Nfd2*mean(z)^2) / (z'*z -
Nfd2*mean(z)^2)

% DFT multisine ko %

z3 = sin_acc_ko;
[ro,col] = size(x3);
xrt3 = zeros(ro,col);
for j = 1:col
   xrt3(:,j) = x3(:,j) - mean(x3(:,1));
end
zrt3 = z3 - mean(z3);
%Frequency Term
g3=sqrt(-1);
%Frequency Range
freq3 = [0.1:0.1:1];
%Frequency Factor (rad/s)
w3 = freq3*(3.14/180);
C3 = exp(-g3*w3*t(1));
dC3 = exp(-g3*w3*dt);
%Length of Frequency
fl3 =length(freq3);
zf3 = zeros(fl3,1);
xf3 = zeros(size(w3,2), col);
R3=1.0;
Np3=4;
for i=2:ro
   xcurr3 = xrt3(i,:);
   zcurr3 = zrt3(i);
   xf3 = R3*xf3 + C3'*xcurr3;
   zf3 = zf3 + (zcurr3'*C3)';
   C3 = C3.*dC3;
   T_hats3(i,:) = real(xf3'*xf3)\ real(xf3'*zf3);
end
% # 5 DFT with sine ko data
reg_xf3 = [xf3(:,2),xf3(:,3),xf3(:,4)];
fix_x3 = [x3(:,2),x3(:,3),x3(:,4)];
T_hat_fd3 = real(reg_xf3'*reg_xf3)\ real(reg_xf3'*zf3)
Y_hat_fd3 = fix_x3*T_hat_fd3;
% R2
Nfd3 = length(t);
```

```
R2_FD_sin_ko = (T_hat_fd3'*fix_x3'*z3 - Nfd3*mean(z3)^2) / (z3'*z3 -
Nfd3*mean(z3)^2)

% # 5 DFT sine model with sine ko data
Y_hat_fd4 = x2*T_hat_sin_ko;

% knock off point matrix
kop_t = zeros(length(t),1);
for j = 1:N
   kop_t(j) = 6.226;
end
kop = zeros(length(t),1);
for j = 1:N
   kop(j) = j-(length(t)/2);
end
% plot multistep input %
figure(1)
plot(steven_step.time, steven_step.signals.values)
title('multi-step input vs. time')
xlabel('time(s)')
ylabel('voltage(V)')

% plot multisine input %
figure(2)
plot(steven_sin.time, steven_sin.signals.values)
title('multi-sine input vs. time')
xlabel('time(s)')
ylabel('voltage(V)')

% plot multistep angular acceleration%
figure(3)
plot(t, step_acc, t, Y_hat, t, Y_hat + 2 * sqrt(s_sq),'--b', t, Y_hat - 2 *
sqrt(s_sq),'--b')
title('measured and modeled multi-step acceleration output vs. time')
xlabel('time (s)')
ylabel('angular acc (rad/s2)')
legend('measured','modeled', 'confidence interval')

% plot multisine angular acceleration%
figure(4)
plot(t, sin_acc, t, Y_hat2, t, Y_hat2 + 2 * sqrt(s_sq2),'--b', t, Y_hat2 - 2 *
sqrt(s_sq2),'--b')
title('measured and modeled multi-sin acceleration output vs. time')
xlabel('time (s)')
```

```
ylabel('angular acc (rad/s2)')
legend('measured','modeled', 'confidence interval')

% plot multisine angular velocity %
figure(5)
plot(t, sin_vel)
title('multi-sin velocity output vs. time')
xlabel('time (s)')
ylabel('angular vel (rad/s)')

% plot multistep angular velocity %
figure(6)
plot(t, step_vel)
title('multi-step velocity output vs. time')
xlabel('time (s)')
ylabel('angular vel (rad/s)')

% plot multisine angle %
figure(7)
plot(t, sin_ang)
title('multi-sine  angle output vs. time')
xlabel('time (s)')
ylabel('angle (rad)')

% plot multistep angle %
figure(8)
plot(t, step_ang)
title('multi-step angle output vs. time')
xlabel('time (s)')
ylabel('angle (rad)')

% plot knockoff multisine angle %
figure(9)
plot(t, sin_ang_ko)
title('multi-sine ko angle output vs. time')
xlabel('time (s)')
ylabel('angle (rad)')

% plot knockoff multisine velocity %
figure(10)
plot(t, sin_vel_ko)
title('multi-sine ko knockoff velocity output vs. time')
xlabel('time (s)')
ylabel('angular vel (rad/s)')
```

```
% plot knockoff multisine acceleration %
figure(11)
plot(t, sin_acc_ko)
title('multi-sine ko knockoff acceleration output vs. time')
xlabel('time (s)')
ylabel('angular acc (rad/s2)')

% plot multisine angular acceleration DFT%
figure(12)
plot(t,z2,t,Y_hat2,t,Y_hat_fd)
title('measured and modeled multi-sine acceleration vs time')
xlabel('time (sec)')
ylabel('angular acceleration (rad/s^2)')
legend('measured','TD-modeled','FD-modeled')

% plot multisine TD residual %
figure(13)
plot(t,(z2-Y_hat2))
title('multi-sine modeled residual TD vs time')
xlabel('time (sec)')
ylabel('residual (rad/s^2)')

% plot multistep FD residual %
figure(14)
plot(t,(z2-Y_hat_fd))
title('multi-sine modeled residual FD vs time')
xlabel('time (sec)')
ylabel('residual (rad/s^2)')

% plot multisine angular acceleration DFT%
figure(15)
plot(t,z,t,Y_hat,t,Y_hat_fd2)
title('measured and modeled multi-step acceleration vs time')
xlabel('time (sec)')
ylabel('angular acceleration (rad/s^2)')
legend('measured','TD-modeled','FD-modeled')

% plot multistep TD residual %
figure(16)
plot(t,(z-Y_hat))
title('multi-step modeled residual TD vs time')
xlabel('time (sec)')
ylabel('residual (rad/s^2)')
```

```
% plot multistep FD residual %
figure(17)
plot(t,(z-Y_hat_fd2))
title('multi-step modeled residual FD vs time')
xlabel('time (sec)')
ylabel('residual (rad/s^2)')

% plot sin ko model coefficients vs time %
figure(18)
subplot(4,1,1)
plot(t,T_hats3(:,2));
title('coefficient A vs. time')
subplot(4,1,2)
plot(t,T_hats3(:,3));
title('coefficient B vs. time')
subplot(4,1,3)
plot(t,T_hats3(:,4));
title('coefficient C vs. time')
subplot(4,1,4)
plot([0,t(end)],[0,0],kop_t,kop);
title('Knock off point')

% plot sin ko data with  model using coeff found using with sin ko data
figure(19)
plot(t, sin_acc_ko, t, Y_hat_fd3)
title('measured and modeled ko multi-sin acceleration output vs. time')
xlabel('time (s)')
ylabel('angular acc (rad/s2)')
legend('measured','multisine knock off model')

% plot sin ko data with  model using coeff found using with sin data
figure(20)
plot(t, sin_acc_ko, t, Y_hat_fd4)
title('measured and modeled ko multi-sin acceleration output vs. time')
xlabel('time (s)')
ylabel('angular acc (rad/s2)')
legend('measured','multisine model')
```