ME 155C Control System Lab

Project

Alex Nguyen

June 12, 2020

Contents

Abstract

This project was tailored towards concepts learned in system identification and controller design. The first part of the project involved system identification section. This required performing nonparametric identification (Correlation Method) and parametric identification (Least-Squares Method) on the given experimental data. After performing these identification techniques, we have arrived at two different process for our system. The second part of the project was controller design of the identified process. The controller was developed using a lead compensator design then tested for its closed-loop performance. The performance of the identified and ideal process was tested with a step response then analyzed through frequency response methods.

1 Introduction

The problem proposed is the identification and controller design of a two-cart mechanical system. The identification required is both parametric and non-parametric then designing a feedback controller based on a process identified in either identification method. The controller design has a few requirements which need to be met:

1. Step response control input does not exceed the maximum required by hardware where the track is 1m long
2. Step response outputs an overshoot < 15 %
3. Step response outputs the smallest settling time achieveable
4. Closed-loop is robust with respect to measurement noise

After designing a controller, we are required to verify the closed loop performance of the system. First, we are asked to state the properties of the closed loop system (eg rise-time, overshoot, settling time, and maximum control magnitude). As well as, identify the closed loop frequency response for the step response.

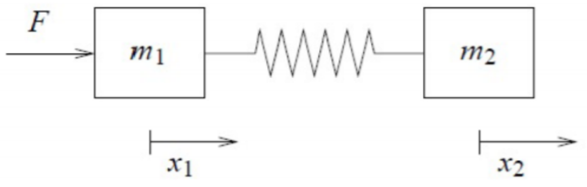


Figure 1: Two-Cart Masses Attached by Spring

The process to be controlled is two-carts with a spring apparatus shown above in Figure 1. By Newton's second law, we can derive the following relations:

$$\begin{aligned} m_1\ddot{x}_1 &= k(x_2 - x_1) + F \\ m_2\ddot{x}_2 &= k(x_1 - x_2) \end{aligned}$$

where x_1 and x_2 are the positions of each cart, m_1 and m_2 are the masses of each cart, and k is the spring constant. The force F is controlled due to an electric motor by an applied voltage V given by the following equation:

$$F = \frac{K_m K_g}{R_m r}\left(V - \frac{K_m K_g}{r}\dot{x}_1\right)$$

where K_m, K_g, R_m, and r are motor parameters. The control input is the applied voltage ($u := V$) with the goal of controlling the second cart's position x_2. The two-cart system has an ideal continuous-time transfer function which was provided in previous ECE courses (which Mechanical Engineers have not taken):

$$G(s) = \frac{Y(s)}{U(s)} = \frac{181.8}{s^4 + 13.24s^3 + 127.2s^2 + 810.4s} \tag{1}$$

where the input units for V(s) is volts and the output units for Y(s) is in meters. Although, Equation 1 is not the true transfer function but an ideal version of it.

A great reference for understanding concepts in identification and controller design project is *Advanced Undergraduate Topics in Control System Design.* Particularly, Chapters two through seven. Chapters two through five covered system identification for nonparametric and parametric methods. Chapters six through seven covered robust stability and control design by loop shaping.

The rest of the document will outline process identification and the controller design process. First, will be the experimental identification of the two cart system using nonparametric and parametric methods. Next, will be the controller design of the identified process from the experimental identification section. Then, the closed loop performance of the system will be tested after implementing the designed controller. Lastly, there will be final remarks and thoughts on the project with suggestions for improvement.

2 Experimental Identification

2.1 Nonparametric Identification

Due to Covid-19, the data was given to students rather than collected ourselves in lab. This data has already taken into account input magnitude and sampling frequency which is in the linear region of operation. Although the time frame of the data given is between 1 and 10 seconds so the experiment time is nine seconds all together. This data will therefore need to be adjusted to account for the phase loss and be trimmed to adjust for the transient response of the system.

We are given 40 different frequency values to be used in our nonparametric identification. These frequency values range from 2 rad/s up to 90 rad/s. The input data is one column vector for the sine-wave input used while the output data has three column vectors. The first column for the output data is the time vector which ranges from one to ten seconds. The second and third columns correspond to the position in ticks (or pulses) of the two carts in the system. The second column is the first cart's position (x_1) and the third column is the second cart's position (x_2).

The input used has the units volts [V] while the output used has the units meters [m]. Initially, the output data given is in pulses of an encoder but using a conversion knowing the wheel circumference and the pulses per revolution we can obtain an output in meters. The following is a conversion used to get meters rather than pulses :

$$\text{conv} = \frac{0.1 \text{ m}}{4096 \text{ pulses}}$$

This encoder conversion will yield a transfer function with the correct units of $\frac{m}{V}$, just like equation 1 shown in the introduction section. Our goal for this identification is to get a transfer function which looks similar. Although, the obtained transfer function will not be the same due to measurement noise and possible disturbances while collecting data.

The nonparametric method uses frequency response identification with sinusoidal inputs and outputs which look like the following:

$$\begin{aligned} u(k) &= \alpha \cos(\Omega k), \quad \forall k \in 1, 2, 3, ..., N \\ y(k) &= \alpha A_\Omega \cos(\Omega k + \phi_\Omega) + \epsilon(k) + n(k) \end{aligned}$$

To minimize the errors caused by noise, the amplitude α should be large. Although, small enough to not leave the linear regieme. If n(k) is much smaller than αA_Ω and for k sufficiently large so that $\epsilon(k)$ is negligible we get the following equation:

$$y(k) \approx \alpha A_\Omega \text{cos}(\Omega k + \phi_\Omega)$$

Here we assume that H(z) is BIBO stable where the magnitude of y(k) is $A_\Omega := |H(e^{j\Omega})|$ and phase of y(k) is $\phi_\Omega := \angle H(e^{j\Omega})$. Repeating this identification for different distinct frequencies Ω, you can obtain bode plot points for H(z). Eventually, you can estimate the frequency response $H(e^{j\Omega})$ over a range of frequencies.

Now we are ready to being finding the frequency response for the range of 40 frequencies. For this, we use the correlation method using the output y(k) we got from the applied input u(k).

$$K_\Omega := \frac{1}{T}\sum_{k=1}^{T} e^{-j\Omega k} y(k) = \frac{1}{T}\sum_{k=1}^{T} (\text{cos}(\Omega k) - j\text{sin}(\Omega k))y(k) \tag{2}$$

As T $\rightarrow \infty$, terms converge to zero then we can use a z-transform to get the following estimate for frequency response $H(e^{j\Omega})$ which is valid for a large T.

$$\hat{H}(e^{j\Omega}) = \frac{2}{\alpha} K_\Omega \tag{3}$$

It is important for the data to have a small sample time and the magnitude to be as large as possible while not leaving the linear regime. This is important because with the proper input magnitude and sampling frequency the transfer function estimate will be very robust to noise.

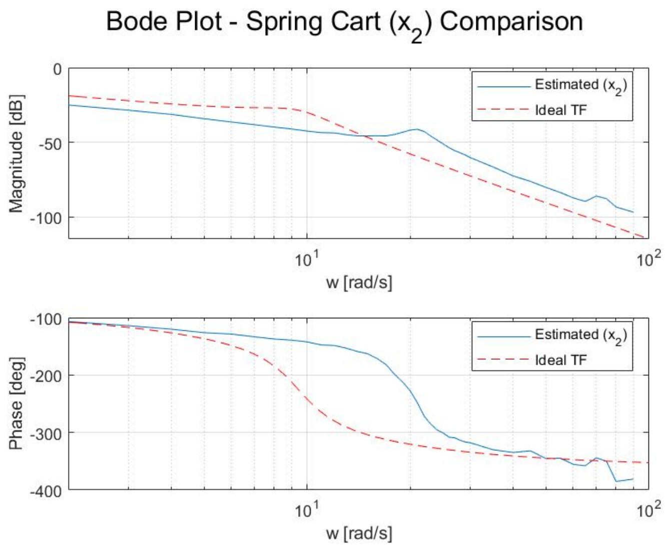


Figure 2: Bode Plot Using Correlation Method

My analysis began with loading all the data into MATLAB to begin the identification. The first task was converting the output units from pulses (or ticks) to meters. After, I adjusted the data to get the correct phase data and get rid of the transient response. Once the data was cropped and adjusted, the correlation method technique was performed to get an estimate of the frequency response using equations 2 and 3. The magnitude was obtained by applying the log to the absolute value of the estimate then multiplying by 20. The phase was obtained by using the phase() function in MATLAB then converting from radians to degrees. The ideal transfer function was then plotted against the estimated transfer function to get the following bode plot in Figure 2.

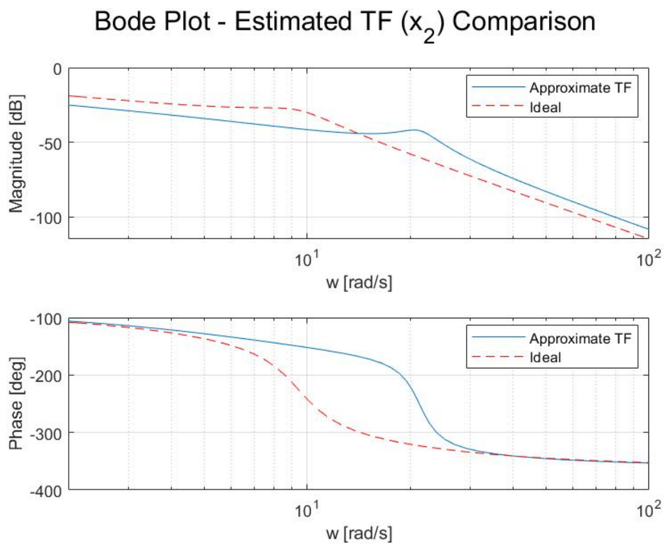


Figure 3: Bode Plot Transfer Function Estimate

The next task was to estimate a transfer function for the nonparametric frequency response estimate. I used the MATLAB function tfest() to estimate the locations of the possible poles, zeros, and gain. This yielded the transfer function shown below and the bode plot shown in Figure 3.

$$P_0(s) = \frac{366.5}{(s + 7.025)(s + 0.03806)(s^2 + 4.587s + 455.1)} \tag{4}$$

The identified transfer function (Equation 4) is similar to the ideal transfer function (Equation 1) since it has volts as the input and meters as the output. The estimated transfer function won't exactly overlap the ideal due to measurement noise and possible disturbances but, as you can see from Figure 3, they look very similar.

2.2 Parametric Identification

For parametric identification, we are required to find an ARX model for the two-cart system using least-squares. Again due to covid-19, the data for the parametric identification section was given to students. The input types given are square waves of low frequency and chirp signals. Since the data was given, the selection of input magnitude was addressed already for students.

The input chosen should have taken the following into account:

1. The input, u(k), should be sufficiently rich such that the R matrix is not singular (and well conditioned)
2. The amplitude of the resulting output, y(k), should be much larger than the measurement noise. Check the quality of the fit by computing $\frac{SSE}{||Y||^2}$
3. The input should be representative of the class of inputs that are expected to appear in the feedback loop

There is an optimal choice of input magnitude which is between noise dominating and non-linear behavior. So, after collecting data the you should validate the input-output behavior with the list above since the input choice is the most critical aspect in good system identification. Also, one should check to see if the input chosen is in the linear regime by multiplying the input and output vector by a gain then comparing with the non-scaled vector.

The first step to analyzing the parametric identification data is to account for the system having an integrator (discrete-time pole at z = 1). If this not taken into account, some experiments will lead to an unstable system and others to a stable one. This will make the identification of the data very difficult.

Dealing with known transfer functions, we can get the following transfer function for the known pole:

$$\frac{Y(z)}{U(z)} = H(z) = \frac{1}{z - \lambda}\tilde{H}(z) \tag{5}$$

where $\tilde{H}(z)$ corresponds to the unknown portion of the transfer function. The transfer function with the known parameter can be manipulated to the following form:

$$\frac{\tilde{Y}(z)}{U(z)} = \tilde{H}(z)$$

Then, we can get the following equation for $\tilde{Y}(z)$ since we know there is a discrete-time integrator (z = 1):

$$\tilde{Y}(z) = (z-1)Y(z) \quad \rightarrow \quad \tilde{y}(k) = y(k+1) - y(k)$$

Here, $\lambda = 1$ since the known parameter is the discrete-time integrator. Now we can directly estimate $\tilde{H}(z)$ by computing $\tilde{y}(k)$ prior to identification then using $\tilde{y}(k)$ rather than y(k).

To get H(z) you must multiply $\tilde{H}(z)$ by $\frac{1}{z-1}$ (or generally, $\frac{1}{z-\lambda}$) to get the original transfer function. Although, now the vector length will be difference for the adjusted input so that must be accounted for as well.

The next step will be to use signal scaling on the input data to make sure the least-square estimate is well conditioned by having the input and output have roughly the same order of magnitude magnitude. The known parameter data is scaled by a constant value by the following relation:

$$\begin{aligned}\tilde{u}(k) &= \alpha_u u(k)\\ \tilde{y}(k) &= \alpha_k y(k)\end{aligned}$$

Then the original transfer function is obtained by reversing the scaling on the unknown portion of the transfer function from u to y:

$$H(z) = \frac{\alpha_y}{\alpha_u}\tilde{H}(z) \tag{6}$$

By using Equation 6, I was able to obtain a better estimate for the transfer function. The α_y was set to 7500 and the α_u was set to 1000. Eventually, I will have to rescale the obtained transfer function to get the original transfer function, H(z), rather than the unknown portion of the transfer function, $\tilde{H}(z)$.

Lastly, the data needs to be adjusted to mitigate the quantization noise due to signal scaling and accounting for the known parameter. Sometimes the hardware used to collect data allows us to sample frequencies much larger than what is needed for identification - oversampling. Due to this, one should down-sample the signals to remove measurement noise from the signals.

If the input-output signals have been sampled with a period T_{low} but one wants to obtain new input-output signals $\tilde{u}(k)$ and $\tilde{y}(k)$ sampled with period $T_{high} := LT_{low}$ where L is an integer greater than one. Instead of disregarding the remaining samples, one can receive some noise reduction by averaging the sample data with the new sample period.

$$\tilde{u}(k) = \frac{u(Lk-1)+u(Lk)+u(Lk+1)}{3} \tag{7}$$

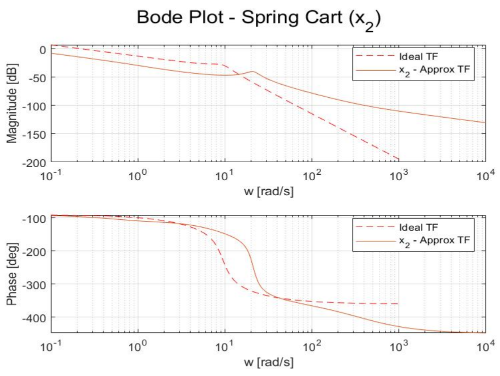
$$\tilde{y}(k) = \frac{y(Lk-1)+y(Lk)+y(Lk+1)}{3} \tag{8}$$

The down-sampled signals exhibit lower noise than the original ones because the noise is being "averaged-out." If one has the signal processing toolbox, you can use the filtering function called resample(). This is a down-sampling function with a more sophisticated averaging algorithm to reduce noise more than the averaging technique shown above in Equations 7 and 8. This is the method I used to analyze the parametric data for the project.

The data is now ready to be used in parametric identification of an ARX model for the two cart spring system. In order to get the best fit bode plot, one needs to decide on the model order for the ARX model. For my data, the input-output data was sampled at 1 kHz. The transfer functions were obtained using the arx() MATLAB function with na = 3, nb = 4, and nk = 0 which reflects the expectation of 3 poles with one at z = 1 and no delay from u to $\tilde{y}(k)$ with the data downsampled to 25 Hz.

To choose the model order, I developed a script which tested the arx() function in MATLAB for various numbers of poles and zeros. In each case, the number of poles was set equal to the number of zeros then a least-squares estimation was performed to see the optimal values of poles and zeros. The SSE decreased as na was increased but the standard deviation increased as well for the coefficients.

The next step is to merge multiple experiments together to make a sufficiently rich R matrix. My parametric transfer function included many input/output data that was combined to create one ARX model. Although, I created two different bode plots to check which data sets gave a bode plot most similar to the ideal transfer function shown in Figure 4. The first bode plot, shown in Figure is from merging all the data sets together. The second bode plot is from merging the individual input signals together (ie chirp signal data only and square wave data only.



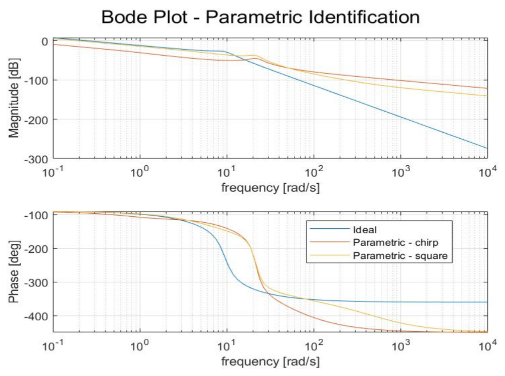


Figure 4: Different Merged Data Sets for Parametric Identification

The transfer function for each of the ARX models was as follows:

$$P0_{\text{all data}}(s) = \frac{0.002982(s-374.8)(s-10.6)(s+1.397)}{s(s+0.9098)(s^2+6.034s+446)} \tag{9}$$

$$P0_{\text{chirp}}(s) = \frac{0.007977s^3-0.6052s^2+7.597s+17.62}{s^4+6.937s^3+475.2s^2+561.3s-1.255e-10} \tag{10}$$

$$P0_{\text{square}}(s) = \frac{0.0008799s^3-0.4526s^2+5.433s+573.6}{s^4+13.15s^3+501s^2+3294s+1.317e-09} \tag{11}$$

Similar to nonparametric, the transfer functions above have an input of volts and an output of meters like equation 1. These transfer functions can be used to compare different step response inputs for the next section.

3 Controller Design

The controller design section requires students to design a feedback controller for the process identified in the experimental identification section. The controller was designed for the ideal transfer function which needs to satisfy the following requirements:

1. The control input does not exceed the maximum one allowed by hardware for a step-response (assuming the track is 1m long)
2. The step response has an overshoot < 15 %
3. The step response exhibits the smallest settling time you are able to achieve
4. The closed-loop step response is somewhat robust to noise

To begin designing the controller, open-loop gain shaping was used to develop a lead compensator for the identified process. In classical lead compensation one starts with the basic unit-gain basic controller C(s) = 1 then one shapes the desired open-loop gain $L(s) := C(s)P_0(s)$ to satisfy the open loop constraints.

The loop shaping beings by multiplying the controller by a proportional gain, k, which moves the bode plot up or down without changing the phase. Then, we can develop a lead compensator based off of the open-loop controller with gain. The lead controller will have the following form:

$$C_{\text{lead}}(s) = \frac{Ts+1}{\alpha Ts+1} \tag{12}$$

Multiplying the proportional gain to the lead compensator, the final developed controller will be of the form C(s) = K$\frac{Ts+1}{\alpha Ts+1}$.

The initial controller design was developed for the ideal transfer function in Equation 1. The first step to developing the controller is to define the overshoot which is used to find the damping ratio. The damping ratio then dictates the required phase margin needed for the system.

$$\zeta = \frac{-ln(\frac{OS}{100})}{\sqrt{\pi^2 + ln(\frac{OS}{100})^2}} \tag{13}$$

$$\phi_{\text{req}} = \tan^{-1}\frac{2\zeta}{\sqrt{-2\zeta^2 + \sqrt{1+4\zeta^4}}} \tag{14}$$

The next step is to choose the proportional gain which will scale the open loop gain L(s). The phase margin for this gain is used to determine the maximum phase necessary to get the correct overshoot. My model required a phase margin of at least 53^o to have no greater than 15% overshoot using the ϕ_{req} equation above. The maximum phase necessary is determined by subtracting the phase margin for the scaled open loop gain system by the required phase margin plus 10^o since additional phase lead is required. We can then solve for the maximum phase lead angle, α, which will be used in the lead controller Equation 15. The equation for alpha is the following:

$$\alpha = \frac{1 - \sin^{-1}(\phi)}{1 + \sin^{-1}(\phi)}$$

The new gain crossover frequency can be found by the following equation then inspection on a bode plot:

$$K_m = -20\log_{10}(\frac{1}{\sqrt{\alpha}})$$

This yielded me with a new crossover frequency of $w_m = 9.87$ rad/s. Now the period of the lead controller can be calculated by the maximum phase lead angle and the new crossover frequency. The value T can be found from the following equation:

$$T = \frac{1}{w_m\sqrt{\alpha}}$$

Using this equation, I found the period to be T = 0.1548 seconds. After these calculations, the controller which I ended up designing was the following:

$$C(s) = \frac{0.7952s + 12}{0.1548s + 1} \tag{15}$$

The design choice for overshoot was given but I chose the gain to be 20 for steady state error reduction of the system. Then, I created a controller which satisfied all the specified requirements in the handout.

The identified model I chose to control was the parametric identification model, but I analyzed the nonparamtric identification too. This model seemed like a better identification of the true transfer function because there was a variety of different inputs used to develop the ARX model. The nonparametric identification used a sine-wave input to estimate the frequency response for the two cart system. Also, the correlation method seemed more approximate than the transfer function identification method outlined in ch 5 of the textbook.

Implementing the controller to the ideal transfer function, we get the following step response with an output less than the maximum length of the hardware (1 m).

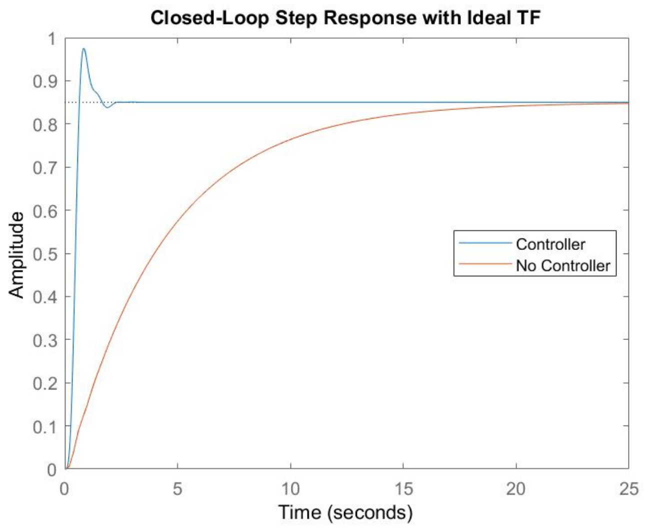


Figure 5: Comparision Between with and without a Controller

Here, the output is around 0.85 m which is below the max allowed length but there is some overshoot. Also, notice the quicker response time in Figure 5. The applied controller allows for a quicker response and settling time from the system due to a step input. Although, there is an overshoot but it meets the requirements by being < 15 % of the input value. The parameters of the lead compensator were tuned to allow for the quickest settling time but the fastest I could manage, while the controller stabilized all processes, was equation 15.

The previous figure (Fig. 5) was for the ideal case alone with a controller and without a controller. The next figure shows the closed loop step response of the identified (both) compared to the ideal with the controller.

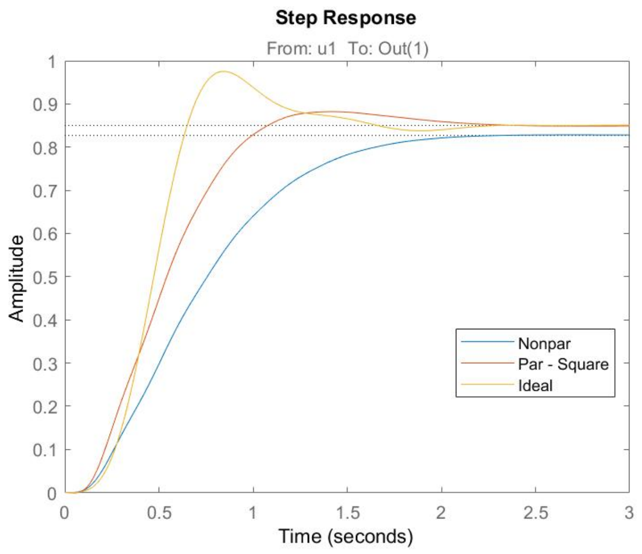


Figure 6: Closed-Loop Robustness

Looking at Figure 6, we can see the closed loop responses are fairly similar for the two identified processes and the ideal process. This shows the closed-loop system is somewhat robust to measurement noise by comparing the outputted values with the ideal closed-loop system. Although, the ideal process transfer function yielded the most oscillations and overshoot when compared to the experimentally identified processes. There was also some trouble developing a lead compensator controller for the ideal process. I was able to develop a controller which would fix the steady state error of the two identified processes but caused the ideal to go unstable. Therefore, I had to settle on a controller which had some steady state error but would work for all processes.

4 Closed-Loop Performance

The closed-loop performance section required the implementation and redesign of your process controller. The developed controller yielded parameters and plots for the identified parametric system.

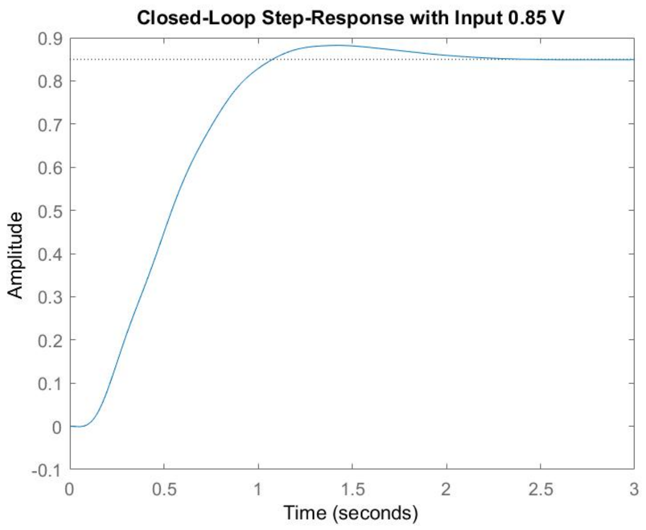


Figure 7: Closed-Loop Step Response - Parametric

The closed-loop step response shows a small overshoot but then an eventual decay to the steady state value of 0.85 m. The input was set to be 0.85 V, so the step response tracks well other than the minor overshoot.

Closed-Loop Step Response Properties

Rise Time (s)	0.6533
Settling Time (s)	1.8152
Overshoot (%)	3.7738
Peak (m)	0.8821
Peak Time (s)	1.4145

The closed-loop step response's properties are shown above in the table. It is important to note that the settling time was the best I could achieve while being robust to other noisy processes or the ideal process. Multiple controllers were developed using lead compensation on all the process systems (ideal, nonparametric, parametric) but issues would always arise when trying to obtain a robust controller. Either the steady state error would be large or one of the step response outputs would go unstable. As stated in the last section, this controller had a trade off of a fast response time to be somewhat robust to measurement noise. The overshoot ended up being vary small for the identified process in comparison to the ideal transfer function which can be seen in Figure 6. Also, the peak value during overshoot was 0.8821 m.

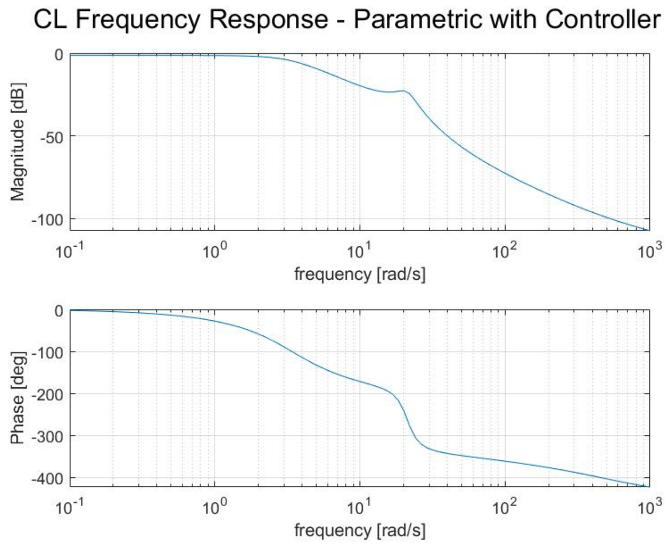


Figure 8: Closed-Loop Frequency Response

The closed loop frequency response for the parametric identified process can be seen above in Figure 8. The method used to identify the frequency response is the bode() command in MATLAB which gives the magnitude and phase plot of the closed loop transfer function.

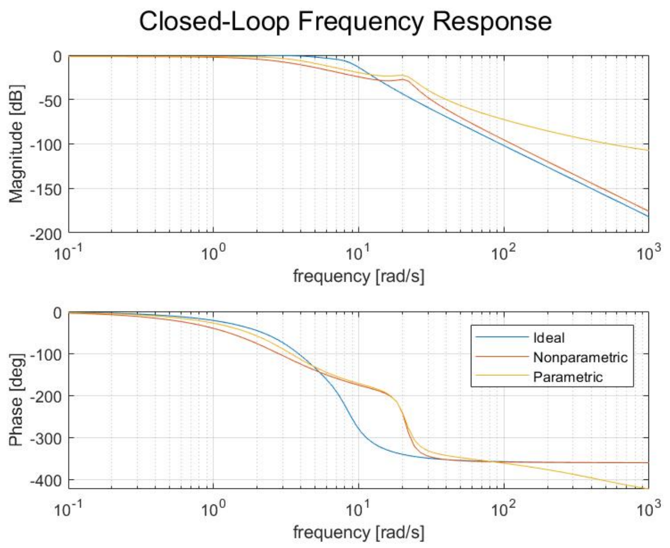


Figure 9: Closed-Loop Frequency - All Processes

The plot above, in Figure 9, represents the closed-loop frequency response for the two identified processes and the ideal process. Notice how the parametric and nonparametric frequency responses look fairly similar when compared with each other. This is to be expected since the identified processes contain measurement noise which cannot be completely mitigated with a designed controller. Therefore, the ideal and identified responses will look different from each other. The ideal process's bode plot looks smoothed out with the controller whereas the identified processes still have a bump at their cutoff frequency. Also, after around 100 rad/s the magnitude and phase of the closed loop parametric response starts to diverge from the ideal process. There is also a phase "bump" between the ideal and identified closed loop response but I assume that would be due to measurement noise and possible perturbations to the system. Lastly, it would seem a good region for the identified transfer functions lie between [1,100] rad/s.

5 Conclusion

This report outlined the theory and analysis performed on the given data for system identification and controller design. After identification, the obtained process in the experimental identification was similar to the ideal process. In the controller design section, the lead compensator developed was robust to measurement noise while optimizing the rise/settling time for the closed-loop step response. In the future, it would be nice to collect data for the input choice of the two-cart system. Due to covid-19, the data was given to us so much of the work was relieved by obtaining input choice data for nonparametric and parametric identification. Also, if more time was available, I wished to develop a better controller which had a settling time of less than one second while being robust to measurement noise and having minimal steady-state error.

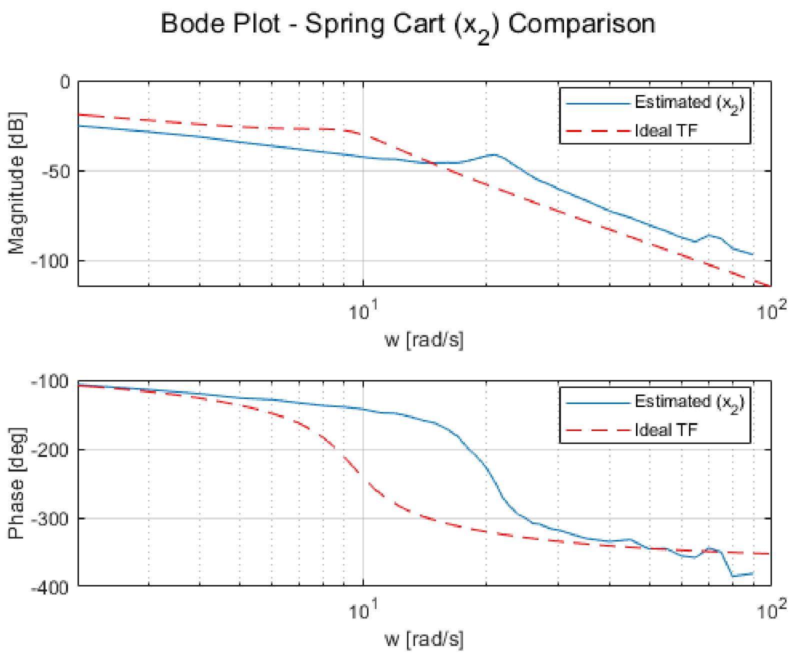
6 MATLAB Script

6.1 Nonparametric Identification

```
% ME 155C Control System Lab Project: Non-Parametric Identification
 Correlation Method
% By: Alex Nguyen

clc; clear; close all;

load('Part1.mat')

%FREQUENCY DATA
item = sort([10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 29 2 30 35 ...
    3 40 45 4 50 55 5 60 65 6 70 75 7 80 8 90 9]);
```

Correlation Method

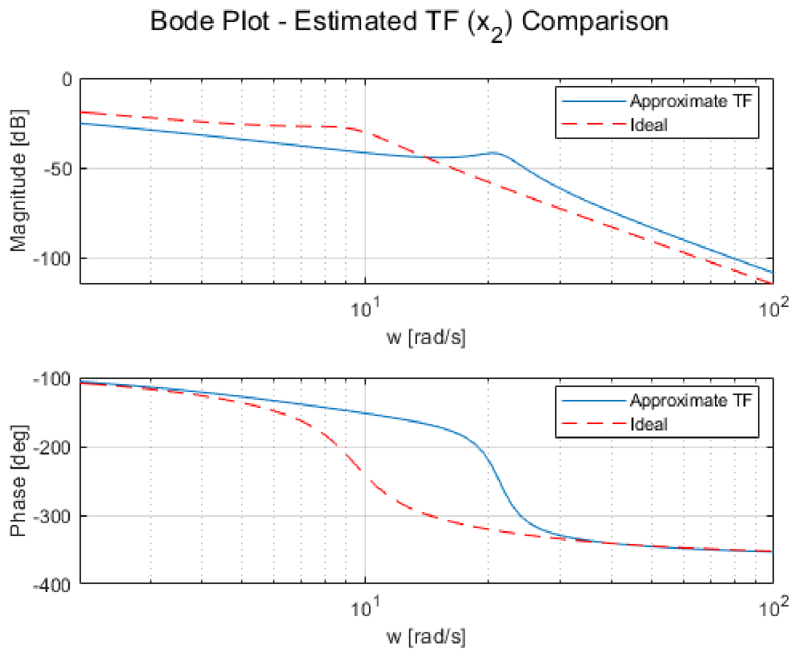
```
alpha = 2; %input amplitude
H = zeros(length(item),2); %preallocation

for i = 1:length(item)
    %DEFINE INPUT & OUTPUT DATA
    u = eval(sprintf('Part1_%drad_s_input_1',item(i))); %input [V]
    y = encoder(eval(sprintf('Part1_
%drad_s_output_1',item(i)))); %output [s,m,m]
    t = y(:,1); %time vector [s]
    w = item(i); %continuous time frequency [rad/s]

    %     t = y(1:end-1,1); %time vector [s]
    %     y = y(2:end,2:3) - y(1:end-1,2:3); %assuming integrator at z
 = 1

    %ADJUST DATA
    [~,ia] = findpeaks(u); %local max
    n1 = ia(1); %new index starting point
    n2 = ia(end); %index ending point

    %CORRECTING PHASE IN DATA
    u = u(n1:n2); y = y(n1:n2,2:3); t = t(n1:n2);

%     u = u(n1:n2); y = y(n1:n2,:); t = t(n1:n2); %assuming integrator
 at z = 1

    %CROP THE DATA
    T = t(end)-t(1); %input sine-wave duration [s]
    Ts = t(2)-t(1); %sample time [s]
    Omega = Ts*w; %discrete time angular velocity [rad]
    f = w/2/pi; %frequency [Hz] - specifically 10 rad/s
    tpeak = floor((T-3)*f)/f; %keep last 3 s of experiment
    kpeak = round(tpeak/Ts); %convert to samples

    %CORRELATION METHOD
    ycrop = y(kpeak:end,1:2); %cropped output data [m,m]
    k = 1:length(ycrop); %sample number
    z = exp(-1i*Omega*k);
    K = 1/length(ycrop)*(z*ycrop);
    H(i,:) = 2*K/alpha; %H estimate
end

%CONSTRUCTING BODE PLOT
Hdb = 20*log10(abs(H)); %Gain [dB]
Hdeg1 = phase(H(:,1))*180/pi; %Cart x1 Phase [deg]
Hdeg2 = phase(H(:,2))*180/pi; %Cart x2 Phase [deg]
w = sort(item); %frequency values [rad/s]

%IDEAL TRANSFER FUNCTION
G = tf(2.97*61.2,[1 13.24 127.15 810.37 0]);
[mag,ang,wout] = bode(G); %bode mag & phase values
mag = squeeze(mag); phase = squeeze(ang);
mag = 20*log10(abs(mag));

figure;
subplot(2,1,1)
semilogx(w,Hdb(:,2),wout,mag,'--r'); grid on;
xlabel('w [rad/s]')
ylabel('Magnitude [dB]')
legend('Estimated (x_2)','Ideal TF','location','best')
xlim([2 100])
subplot(2,1,2)
semilogx(w,Hdeg2,wout,phase,'--r'); grid on;
xlabel('w [rad/s]')
ylabel('Phase [deg]')
legend('Estimated (x_2)','Ideal TF', 'location','best')
xlim([2 100])
sgtitle('Bode Plot - Spring Cart (x_2) Comparison')
```



ESTIMATED TRANSFER FUNCTION

```
gain = 10.^(Hdb(:,2)/20);
response = gain.*exp(1i*Hdeg2*pi/180);
gfr = idfrd(response,w,Ts);
x2_est = tfest(gfr,4,0); %estimated x2 c-t transfer function

%ESTIMATE TRANSFER FUNCTION ZPK
b = [366.5]; %numerator coefficients
a = [1 11.65 487.8 3216 121.7]; %denominator coefficients
[z1,p1,k1] = tf2zp(b,a);
sys_est1 = zpk(z1,p1,k1);
save('Process1.mat','sys_est1','z1','p1','k1','Ts') %saving .mat data

%ESTIMATED BODE MAGNITUDE AND PHASE DATA
[m,a,wnew] = bode(x2_est); %estimated x2 bode values
m = squeeze(m); a = squeeze(a);
m = 20*log10(abs(m));

%BODE PLOT COMPARISON
figure;
subplot(2,1,1)
semilogx(wnew,m,wout,mag,'--r'); grid on;
xlabel('w [rad/s]')
ylabel('Magnitude [dB]')
```

```
legend('Approximate TF','Ideal','location','best')
xlim([2 100])
subplot(2,1,2)
semilogx(wnew,a,wout,phase,'--r'); grid on;
xlabel('w [rad/s]')
ylabel('Phase [deg]')
legend('Approximate TF','Ideal', 'location','best')
xlim([2 100])
sgtitle('Bode Plot - Estimated TF (x_2) Comparison')
```



Published with MATLAB® R2020a

6.2 Parametric Identification

Table of Contents

```
% ME 155C Control System Lab Project: Parametric Identification Least-
Squares Method
% By: Alex Nguyen

clc; clear; close all;

load('Part2.mat')
```

Ideal Transfer Function

```
G = tf(2.97*61.2,[1 13.24 127.15 810.37 0]);
[m,ang,w] = bode(G); %bode mag & phase values
m = 20*log10(abs(squeeze(m))); phase = squeeze(ang);
```

SQUARE WAVE - INPUT/OUTPUT DATA

```
u1 = Part2_square_dif_10_input_1; %input
y1 = encoder(Part2_square_dif_10_output_1); %output
u2 = Part2_square_10_input_1; %input
y2 = encoder(Part2_square_10_output_1); %output

%TIME
t = y2(:,1); %time vector [s]
Ts = t(2)-t(1); %sample time [s]

% ADJUST DATA - DEALING WITH KNOWN PARAMETERS
%assume the process has an integrator (d-t pole at z = 1) in its TF
y1 = y1(:,3); y2 = y2(:,3); %redefining data
ybar1 = y1(2:end) - y1(1:end-1); ybar2 = y2(2:end) -
 y2(1:end-1); %output [m]
ubar1 = u1(1:end-1); ubar2 = u2(1:end-1); %input [V]

% ADJUST DATA - SCALING OUTPUT SIGNAL
alpy = 7500;
alpu = 1000;
ybar1 = ybar1*alpy; ybar2 = ybar2*alpy; %scaled output
ubar1 = ubar1*alpu; ubar2 = ubar2*alpu; %scaled input

% ADJUST DATA - DOWN SAMPLING
L = round(0.04/Ts); %down sample from 1kHz to 40 Hz
```

```
ybar1 = resample(ybar1,1,L,1); ybar2 =
 resample(ybar2,1,L,1); %downsampling output [V]
ubar1 = resample(ubar1,1,L,1); ubar2 =
 resample(ubar2,1,L,1); %downsampling input [V]
```

CHIRP SIGNAL -INPUT/OUTPUT DATA

INPUT

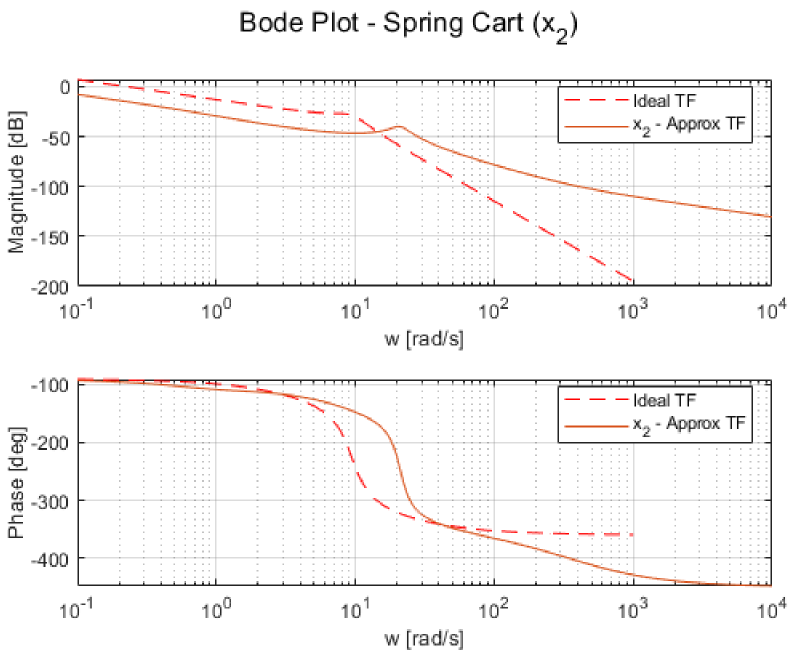
```
u = zeros(9001,6); %preallocation

%STORING INPUT DATA
u(:,1) = Part2_chirp_0p001_1_rad_s_input_1; %chirp signal [0.001 Hz,1
 Hz]
u(:,2) = Part2_chirp_0p001_25_rad_s_input_1; %chirp signal [0.001
 Hz,25 Hz]
u(:,3) = Part2_chirp_0p001_90_input_1; %chirp signal [0.001 Hz,90 Hz]
u(:,4) = Part2_chirp_10_50_input_1; %chirp signal [10 Hz,50 Hz]
u(:,5) = Part2_chirp_1_10_rad_s_input_1; %chirp signal [1 Hz,10 Hz]
u(:,6) =  Part2_chirp_20_90_input_1; %chirp signal [20 Hz,90 Hz]

%OUTPUT
y = zeros(9001,12); %preallocation

%STORING OUTPUT DATA - UNITS [s,m,m]
y(:,1) = Part2_chirp_0p001_1_rad_s_output_1(:,3); %chirp signal [0.001
 Hz,1 Hz]
y(:,2) = Part2_chirp_0p001_25_rad_s_output_1(:,3); %chirp signal
 [0.001 Hz,25 Hz]
y(:,3) = Part2_chirp_0p001_90_output_1(:,3); %chirp signal [0.001
 Hz,90 Hz]
y(:,4) = Part2_chirp_10_50_output_1(:,3); %chirp signal [10 Hz,50 Hz]
y(:,5) = Part2_chirp_1_10_rad_s_output_1(:,3); %chirp signal [1 Hz,10
 Hz]
y(:,6) =  Part2_chirp_20_90_output_1(:,3); %chirp signal [20 Hz,90 Hz]

%CONVERT TICKS TO METERS
encoder = 0.1/4096; %[rev/ticks]
y = encoder*y;

% ADJUST DATA - DEALING WITH KNOWN PARAMETERS
%assume the process has an integrator (d-t pole at z = 1) in its TF
ybar = zeros(length(y)-1,size(y,2)); ubar = ybar;
for i = 1:6
    ybar(:,i) = y(2:end,i) - y(1:end-1,i); %output
    ubar(:,i) = u(1:end-1,i); %input
end

% ADJUST DATA - SCALING OUTPUT SIGNAL
alpy = 7500;
alpu = 1000;
ybar = alpy.*ybar(:,1:6); %scaled output
ubar = alpu.*ubar(:,1:6); %scaled input
```

```
% ADJUST DATA - DOWN SAMPLING
L = round(0.04/Ts); %down sample from 1kHz to 40 Hz
Tnew = L*Ts; %new sample time [s]
y = zeros(length(resample(ybar(:,1),1,L,1)),6); u = y; %preallocation
for i = 1:size(ybar,2)
    y(:,i) = resample(ybar(:,i),1,L,1); %output resampled
    u(:,i) = resample(ubar(:,1),1,L,1); %input resampled
end
```

DEVELOP ARX MODEL FOR DIFFERENT SIGNALS OF X2 (CART ATTATCHED TO SPRING)

```
%CREATE THE BEST ARX MODEL
na = 3; nb = 4; nk = 0;

dat = {0}; %preallocation
dat{1} = iddata(ybar1,ubar1,Tnew);
dat{2} = iddata(ybar2,ubar2,Tnew);
for i = 1:6
    dat{i+2} = iddata(y(:,i),u(:,i),Tnew);
end
data = merge(dat{1},dat{2},dat{3},dat{4},dat{5},dat{6},dat{7},dat{8});

model = arx(data,[na nb nk]); %estimated model
z = tf('z',Tnew); %used to create a d-t TF model
sysd = alpy/alpu/(z-1)*tf(model,'Measured'); %d-t transfer function
sys = d2c(sysd,'zoh'); %c-t transfer function

%ESTIMATE TRANSFER FUNCTION ZPK
b = [0.002982 -1.145 10.24 16.55]; %nustepmerator coefficients
a = [1 6.944 451.5 405.8 8.516e-10]; %denominator coefficients
[z2,p2,k2] = tf2zp(b,a);
sys_est2 = zpk(z2,p2,k2);
save('Process2.mat','sys_est2','z2','p2','k2','Ts') %saving .mat data

%MAGNITUDE AND PHASE DATA
[mag,ang,wout] = bode(sys);
mag = squeeze(mag);
mag = 20*log10(abs(mag)); %magnitude [dB]
ang = squeeze(ang)-360; %phase shifted by 2pi [ang]

figure;
subplot(2,1,1)
semilogx(w,m,'--r',wout,mag); grid on;
xlabel('w [rad/s]')
ylabel('Magnitude [dB]')
legend('Ideal TF','x_2 - Approx TF','location','best')
subplot(2,1,2)
semilogx(w,phase,'--r',wout,ang); grid on;
xlabel('w [rad/s]')
ylabel('Phase [deg]')
```

```
legend('Ideal TF','x_2 - Approx TF', 'location','best')
sgtitle('Bode Plot - Spring Cart (x_2)')
```



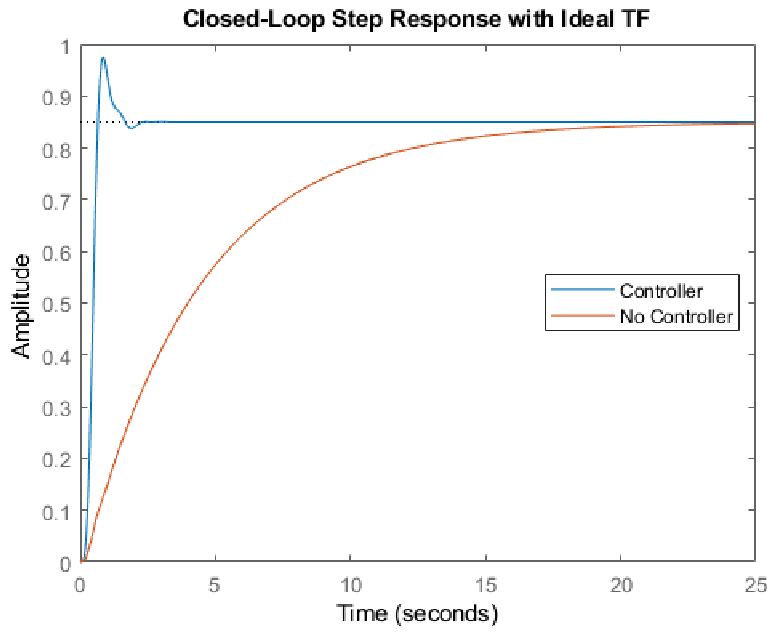
Published with MATLAB® R2020a

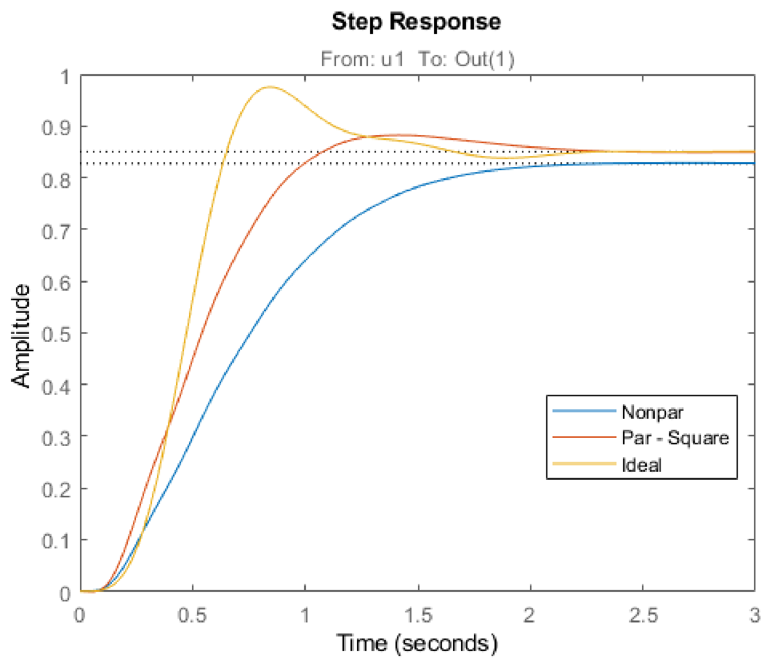
6.3 Controller Design

```
% ME 155C Control System Lab Project: Controller Design
% By: Alex Nguyen

clc; clear; close all;

%NONPARAMETRIC AND PARAMETRIC PROCESS TRANSFER FUNCTION
load('Process1.mat'); P0a = zpk(z1,p1,k1); %nonparametric process
load('Process2.mat'); P0b = zpk(z2,p2,k2); %parametric process - all
 data
load('ParametricTF.mat'); %parametric process - chirp and square data
 TF

%IDEAL TRANSFER FUNCTION
s = tf('s'); %ct variable 's'
G = tf(2.97*61.2,[1 13.24 127.15 810.37 0]);

%REQUIRED PHASE MARGIN
OS = 15; %percent overshoot
zeta = -log(OS/100)/sqrt(pi^2+log(OS/100)^2); %damping ratio
pm_req = atand(2*zeta/sqrt(-2*zeta^2+sqrt(1+4*zeta^4))); %phase margin
 required

%DEVLOPING CONTROLLER - LEAD COMPENSATOR
K = 20; %proportional gain
[~,Pm,~,~] = margin(K*G); %phase margin
phi = pm_req - Pm + 10; %maximum phase margin
alpha = (1-sind(phi))/(1+sind(phi));
[~,~,~,wm] = margin(K*G/sqrt(alpha)); %new crossover frequency [rad/s]
T = 1/wm/sqrt(alpha);
C = 12*(alpha*T*s+1)/(s*T+1); %lead compensator

%STEP RESPONSE INFO
L = C*G; %open loop gain
v = .85; %voltage step input [V]
Y1 = v*feedback(L,1); %closed loop - controller
Y2 = v*feedback(G,1); %closed loop feedback - no controller
stepinfo(Y1) %step-response characterisitics

figure;
step(Y1,Y2); %step-response output - Estimated & Ideal
legend('Controller','No Controller','location','best')
title('Closed-Loop Step Response with Ideal TF')

% %GAIN & PHASE MARGIN
% [Gm,Pm,Wcg,Wcp] = margin(L);
% fprintf('Gain Margin: %4.4f, Phase Margin: %4.4f\n',Gm,Pm)
% fprintf('Gain Crossover Freq: %4.4f, Phase Crossover Freq: %4.4f
\n',Wcg,Wcp)

%ROBUSTNESS WITH RESPECT TO MEASUREMENT NOISE
Y2 = v*feedback(C*P0_square,1); %parametric - square signal
Y3 = v*feedback(C*P0a,1); %nonparametric
```

```
figure;
step(Y3,Y2,Y1)
legend('Nonpar','Par - Square','Ideal','location','best')

%SAVE .MAT FILE
save('controller.mat','C')
```

```
Warning: The closed-loop system is unstable.

ans = 

  struct with fields:

        RiseTime: 0.3436
    SettlingTime: 1.4839
     SettlingMin: 0.7978
     SettlingMax: 0.9754
       Overshoot: 14.7566
      Undershoot: 0
            Peak: 0.9754
        PeakTime: 0.8493
```





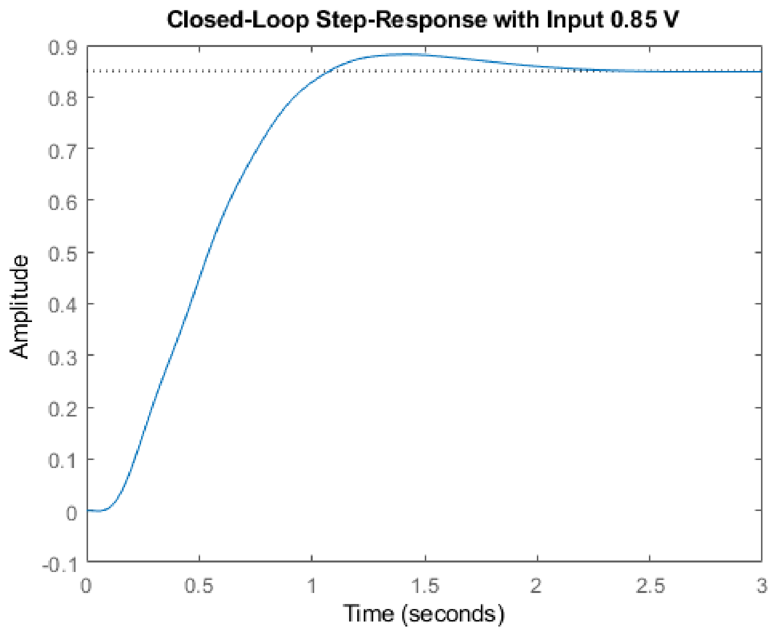
Published with MATLAB® R2020a

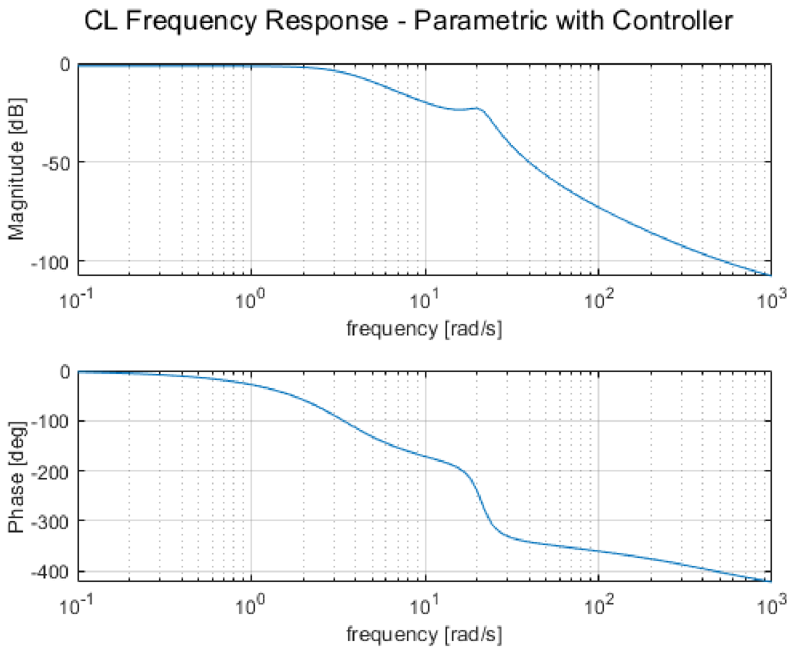
6.4 Close-Loop Performance

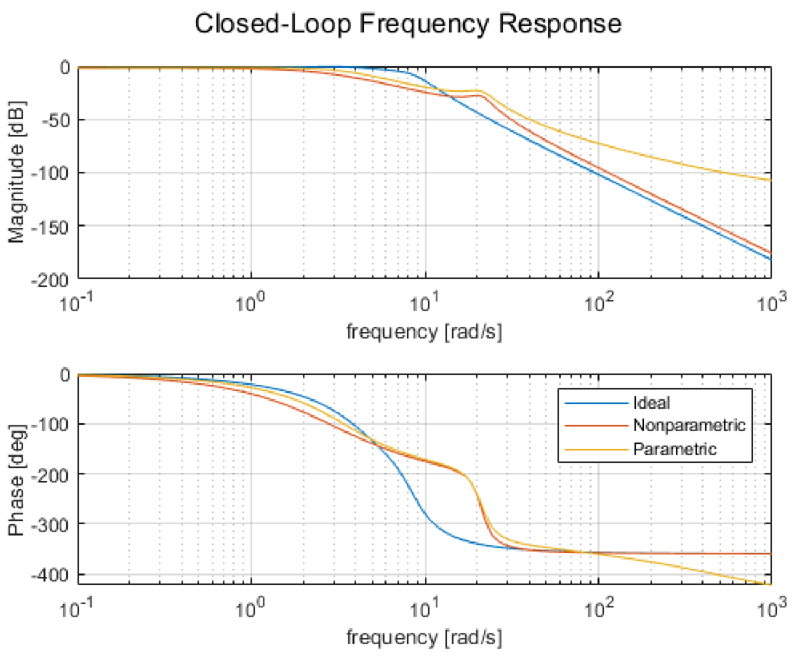
```
% ME 155C Control System Lab Project: Closed-Loop Performace
% By: Alex Nguyen

clc; clear; close all;

%LOADING DATA
load('controller.mat') %lead compensator controller
load('Process1.mat') %nonparametric identification
load('Process2.mat') %parametric identification - all data
load('ParametricTF.mat') %parametric identification - chirp and square
 data

%EVALUATE CLOSED LOOP STEP RESPONSE - ADJUSTED (v = 0.85)
v = 0.85; %input voltage
Y1 = v*feedback(G*C,1); %ideal closed loop
Y2 = v*feedback(sys_est1*C,1); %nonparametric closed loop
Y3 = v*feedback(P0_square*C,1); %parametric closed loop - square input

ideal = stepinfo(Y1); %ideal
nonparametric_step_info = stepinfo(Y2); %nonparametric
parametric_step_info = stepinfo(Y3); %parametric - square input

figure;
step(Y3);
title(sprintf('Closed-Loop Step-Response with Input %4.2f V',v))

%CLOSED-LOOP FREQUENCY RESPONSE
w = logspace(-1,3,100); %frequency [rad/s]
[m1,a1] = bode(Y1,w); m1 = 20*log10(abs(squeeze(m1))); a1 =
 squeeze(a1); %ideal
[m2,a2] = bode(Y2,w); m2 = 20*log10(abs(squeeze(m2))); a2 =
 squeeze(a2); %nonparametric
[m3,a3] = bode(Y3,w); m3 = 20*log10(abs(squeeze(m3))); a3 =
 squeeze(a3) - 360; %parametric

figure;
subplot(2,1,1)
semilogx(w,m3); grid on;
ylabel('Magnitude [dB]')
xlabel('frequency [rad/s]')
subplot(2,1,2)
semilogx(w,a3); grid on;
ylabel('Phase [deg]')
xlabel('frequency [rad/s]')
sgtitle('CL Frequency Response - Parametric with Controller')

figure;
subplot(2,1,1)
semilogx(w,m1,w,m2,w,m3); grid on;
ylabel('Magnitude [dB]')
xlabel('frequency [rad/s]')
subplot(2,1,2)
```

```
semilogx(w,a1,w,a2,w,a3); grid on;
ylabel('Phase [deg]')
xlabel('frequency [rad/s]')
legend('Ideal','Nonparametric','Parametric','Location','best')
sgtitle('Closed-Loop Frequency Response')

%PRINT RESULTS
nonparametric_step_info
parametric_step_info
```

```
nonparametric_step_info =

  struct with fields:

        RiseTime: 1.0647
    SettlingTime: 1.7757
     SettlingMin: 0.7469
     SettlingMax: 0.8283
       Overshoot: 0.1455
      Undershoot: 0
            Peak: 0.8283
        PeakTime: 2.7044


parametric_step_info =

  struct with fields:

        RiseTime: 0.6533
    SettlingTime: 1.8152
     SettlingMin: 0.7684
     SettlingMax: 0.8821
       Overshoot: 3.7738
      Undershoot: 0.0937
            Peak: 0.8821
        PeakTime: 1.4145
```


Closed-Loop Step-Response with Input 0.85 V
Amplitude
Time (seconds)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
0.5
1
1.5
2
2.5
3


CL Frequency Response - Parametric with Controller
Magnitude [dB]
0
-50
-100
frequency [rad/s]
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
Phase [deg]
0
-100
-200
-300
-400
frequency [rad/s]



Published with MATLAB® R2020a