Massachusetts Institute of Technology

Department of Mechanical Engineering

2.004 *Dynamics and Control II*

Laboratory Session 7: Frequency Domain Control

Laboratory Objectives:

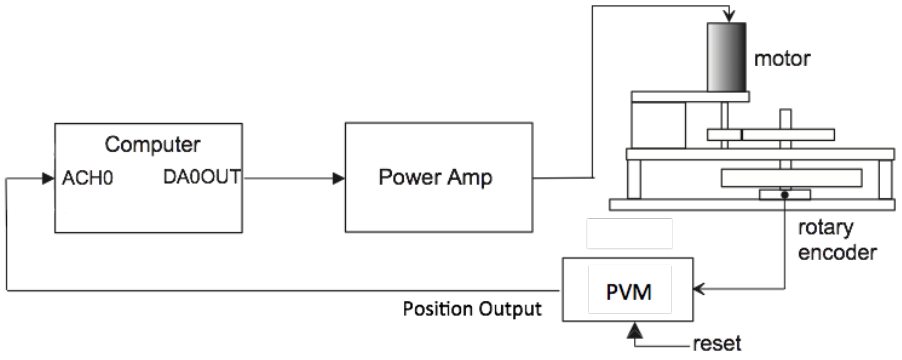
(i) To investigate open and closed loop Bode plots of the 2.004 flywheel system.

(ii) To design a lead compensator to achieve given closed-loop specifications.

(iii) To compare root locus and frequency methods.

Introduction: In previous labs we approached controller design through root locus techniques based on pole-zero models. In this session we switch to a controller design method based on frequency analysis. We will determine the accuracy of our flywheel transfer function across a range of frequencies by performing a frequency sweep. From the frequency response plots we can then design a compensator to lift the phase margin to a desired value, which will in turn produce a satisfactory closed loop response.

The Experimental Setup: The set-up is very similar to that of the Flywheel Position Control lab by using the "Position" output of the PVM as our feedback signal (see below). In this implementation we will use several Simulink blocks to generate setpoints internally, instead of using the function generator. Make cable connections according to the diagram below.



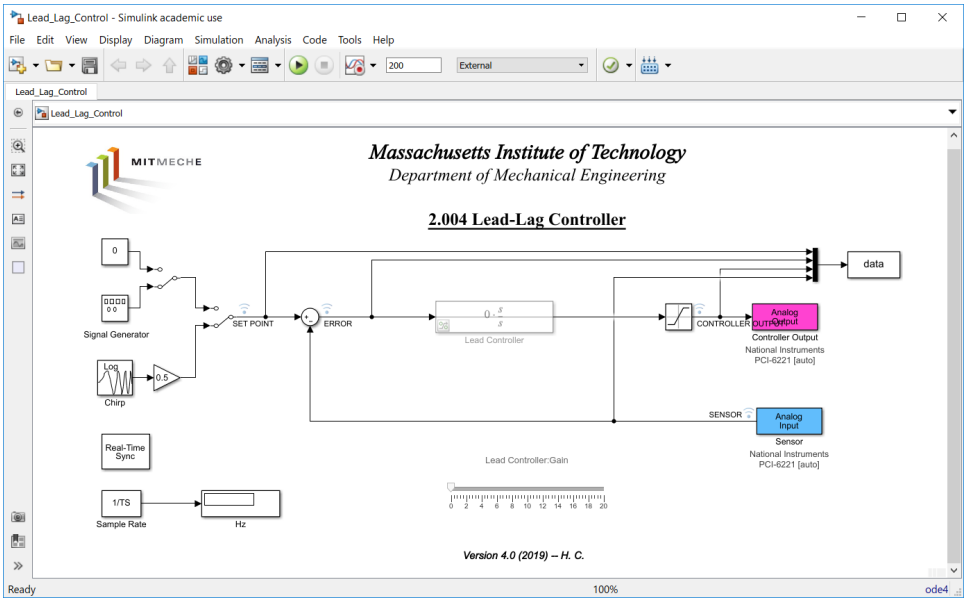
Recall the "Position" BNC connector on the PVM provides a voltage signal that is proportional to the angular position of the flywheel. As a reminder: optical encoders, such as those used in this system, are *incremental* – that is, they do not have an inherent absolute zero position. The **RESET** button on the side of the PVM is used to set the current position of the flywheel as the "home" (or zero) position.

Before proceeding, copy "Lab 7" folder from 2.004 course locker to Windows Desktop or a folder on your local system.

Experiment #1: Open and Closed Loop Bode Plots

In order to build a Bode plot from experimental data, we will perform a frequency sweep using Simulink's "Chirp" function (see Appendix D) as input, which varies the frequency of a sinusoidal signal over time. To improve low frequency responses, we pre-configured the "Chirp" block to sweep the frequency from low (0.01 Hz) to high frequency (2 Hz) logarithmically. This frequency range was chosen in order to capture major frequency features.

(a) Open the Simulink model `Lead_Lag_Control.slx` and ensure the value of the gain block placed after the chirp block is set to 0.5.



Note that this Simulink model performs "unity error feedback" that is equivalent to pure proportional control with a gain of 1, when the "Lead Controller" block is "commented through." This is to constraint the motion of the flywheel and to get both open and closed loop Bode plots at the some time.

(b) Run the Simulink model with the chirp signal as input (make sure the manual switch points to the correct signal source). Monitor the output waveforms by enabling the SDI function on Simulink. The program is setup to run for 200 seconds. Observe the motion of the flywheel while it is running.

(c) Read the MATLAB script `estimate_freq_resp.m`. Edit the code so that it builds the open and closed loop model Bode plots. Determine how it differs from your Prelab implementation, you will need to enter your transfer function from the Prelab on line 50.

(d) Run `estimate_freq_resp.m` to build the experimental Bode plots for open and closed loop transfer functions. The script takes the "data" variable in MATLAB workspace and builds Bode plots for both open and closed loops plotted against the theoretical model.

(e) Correct for any mismatch in magnitude gain at the crossover frequency of the open loop Bode plots by inserting an adjustment factor to the DC gain term of the model transfer function in the script. This will ensure the model matches the experimental data. *Question: At lower frequencies, the mismatch in open loop gain is significant, why is that?*

Experiment #2: Controller Design

Here we wish to design a controller that will produce a closed-loop step response to meet the following time domain specifications: an overshoot of 20% and settling time $T_s = 1.5$ seconds or less. A quick inspection of your plots should reveal that the uncompensated system does not reach this performance level. In order to fix this, we will use the script `Lead_compensator_design.m` to design the required lead compensator of the form (see Appendix A):

$$G_c(s) = K_c G(s) = K_c \left(\frac{\tau s + 1}{\alpha \tau s + 1} \right).$$

(a) Determine the required performance parameters Phase Margin ϕ_m and Crossover Frequency ω_c from the time domain specifications. Hint: think about how to relate overshoot to ζ and then to ϕ_m, and ω_c to settling time through their respective equations (see Appendix B).

(b) Copy the adjusted theoretical transfer function from the previous experiment to line 21, and enter the required performance parameters ϕ_m and ω_c, the phase to be added and the location of the target frequency to lines 42 and 43, respectively.

(c) Enter the required formulae to calculate `alpha` and `tau` on lines 50 and 53. Check in Appendix A if you need help.

(d) Run the code to determine the form of the compensator, and to verify from the plots that it has the expected margins.

(e) Using the `stepinfo` function, determine if your newly compensated system meets the time domain performance requirement. Tweak the values if required.

(f) Using `sisotool` as in previous labs, plot the root locus, step response and Bode plots of the controller you've designed to check their relationship.

Experiment #3: Controller Testing and Verification

Now we want to implement the controller we designed to validate its theoretical behavior. This will be done by modifying the existing Simulink realtime model.

(a) Modify your Simulink model by uncommenting the "Lead Controller" block, and entering your newly designed controller. Note you can do this by entering the pole and zero locations, and the gain directly in the block.

(b) Lower the gain factor of the chirp signal **to between 0.2 and 0.3 to avoid saturation of power amp** and run the Simulink model. Monitor the waveforms and observe the motion of the flywheel.

(c) Modify G_c in the `estimate_freq_resp.m` script so that it includes the newly designed controller, and build the experimental and theoretical Bode plots for open and closed loop behavior by running the script.

(d) Set up a square wave by using the "Signal Generator" block in Simulink (make sure the manual switch points to the correct signal source). We recommend an amplitude of 0.5, but you need to pick a frequency that allows the response to reach steady-state. You can determine the square wave frequency from the step response plot in SISOTOOL, and you may want to add a little bit more time for the response to stay at stead-state. Run this Simulink realtime model and save the step response. Verify the overshoot and settling time requirements.

(e) Now configure the signal generator to output a sine wave in Simulink, with a frequency equal to the crossover frequency (ω_c) in the open-loop Bode plot. Obtain a pair of pure sinusoidal time domain signals (error and sensor) and manually measure the magnitude and phase of the two signals and compare them with your Bode plots. Refer to Appendix C for determining gain and phase of two sine waves.

★ Additional Extra Credit Task ★:

In today's extra credit task, we will explore an alternate method to build the frequency response of a plant. Instead of a chirp signal, we will inject a band limited white noise signal (see Appendix E). This technique is commonly used in industry, and takes advantage of some interesting mathematics - please ask a lab staff if you have questions!

1. Modify the existing Simulink model by adding the "Band Limited White Noise" block in lieu of the "Constant" block and set the manual switches to use this signal as input. You should set the power to 0.1 and the sample time to 0.01 inside the block - to match the rest of the system.

2. Unfortunately, this creates white noise between 0 and 100Hz (the frequency of the system), whereas we only want to inject noise in the frequency range under test. Therefore, we need to also insert a low-pass filter (LPF) to remove high frequency noise. Insert a transfer function block, set the numerator to 1, and the denominator to $[\tau, 1]$, where τ is the time constant of the low-pass filter. What value should τ have? Hint: think about the maximum expected signal frequency, it should be in the range $\tau \in [\frac{1}{10}, \frac{1}{15}]$.

3. Attach the white noise block to the LPF, then attach this to a gain block with a gain value set to around 0.1, to ensure the power amplifier does not trip (if your system trips, try reducing this value).

4. Set the maximum simulation time to 120s, as you do not need as much time as in the chirp case.

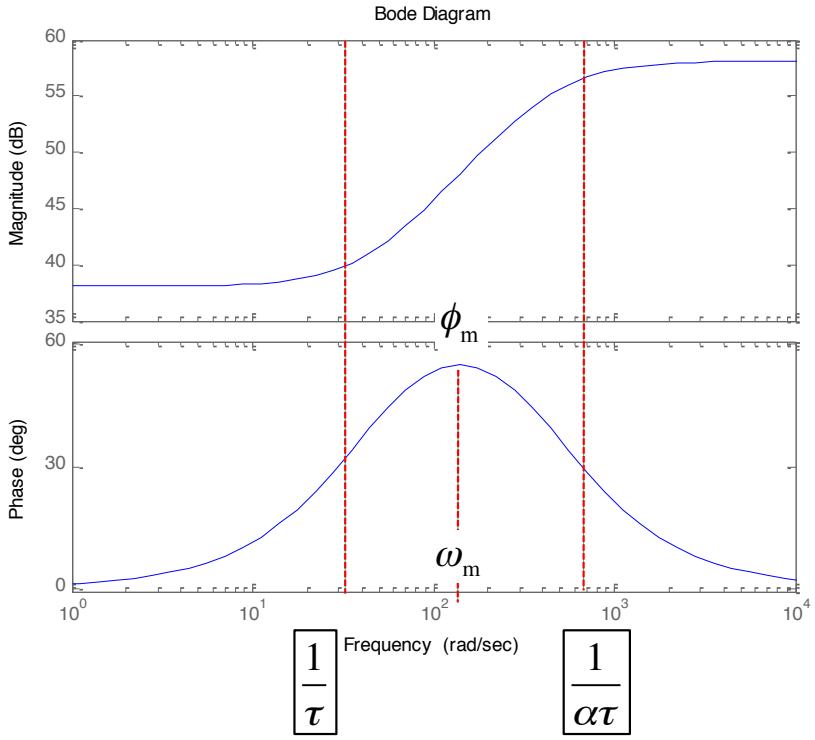
5. You're done with the setup, have a lab staff check your Simulink model, and then you're ready to test!

Once checked, run the simulation, and observe the results. Run the MATLAB script `estimate_freq_resp.m` to plot the Bode plots against the theoretical behaviour. Ensure the model you are plotting against matches the system you're using - if you're using the compensated system, use the compensated model.

Experiment with running the code with different `tmin` and `tmax` values (lines 5 and 6). How does this affect the result?

Appendix A: Lead Controller Frequency Design

You may find the following formulas relating the phase margin and frequency of a lead compensator to its transfer function useful. The following plot shows the contribution of the compensator in the region of interest.



$$
\begin{aligned}
G_c(s) &= K_c G(s) \\
G(s) &= \frac{\tau s + 1}{\alpha\tau s + 1} \qquad 0 \le \alpha < 1 \\
\phi_m &= \sin^{-1} \frac{1-\alpha}{1+\alpha} \\
\omega_m &= \frac{1}{\tau\sqrt{\alpha}} \\
1 &= |K_c G(j\omega_c) G_p(j\omega_c)|
\end{aligned}
$$

Typically we use $\alpha \ge 0.1$ to avoid large gains at high frequency, and to place the maximum phase increase ϕ_m at the desired crossover frequency ω_c.

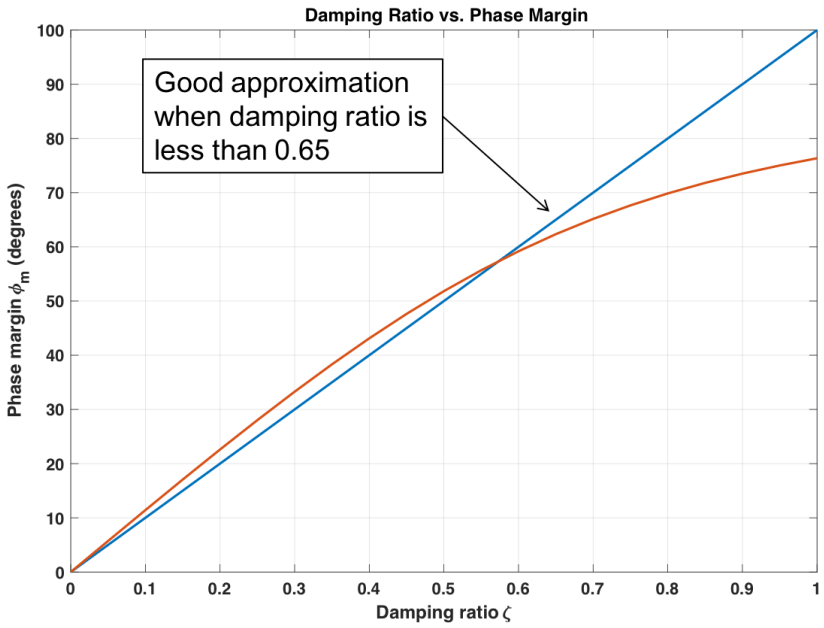
Appendix B: Mathematical Relationships

These equations allow you to map a percentage overshoot requirement to ζ to a phase margin (PM) requirement.

$$\zeta = \frac{-\ln\left(\%OS/100\right)}{\sqrt{\pi^2 + \ln^2\left(\%OS/100\right)}}$$

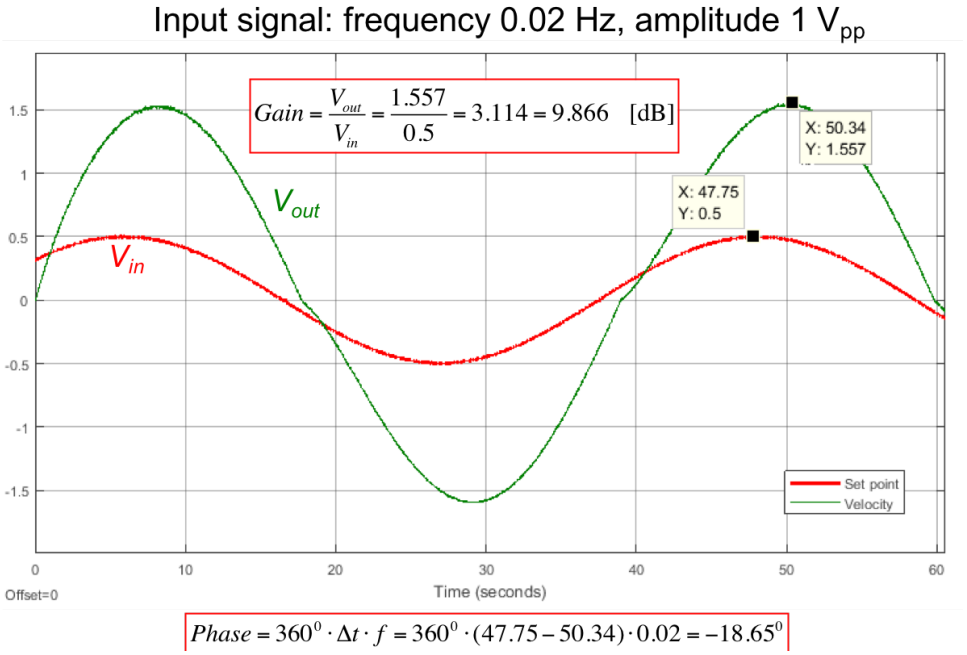
$$PM = \arctan\left(\frac{2\zeta}{\sqrt{\sqrt{4\zeta^4 + 1} - 2\zeta^2}}\right)$$

From the plots below, observe where these functions are valid, and factor this uncertainty in if required.



Appendix C: Measurement of Sinusoids

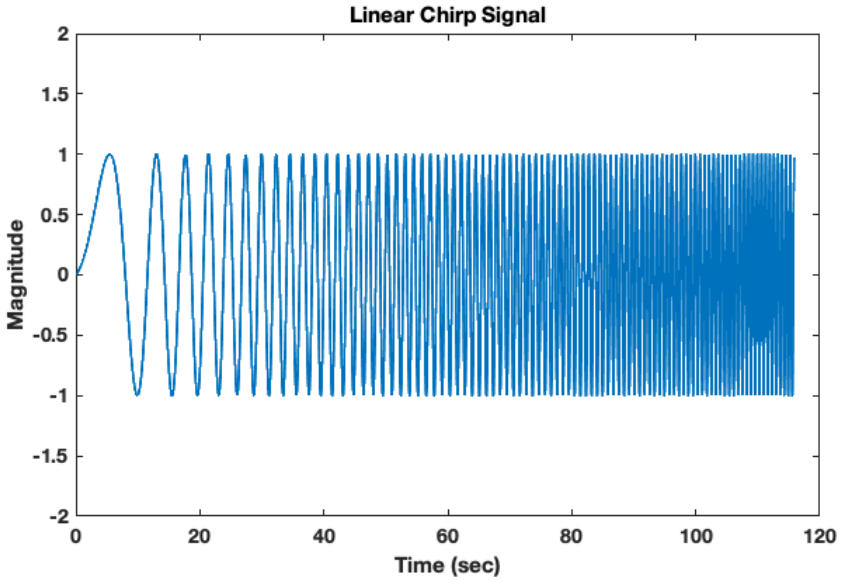
The graph below demonstrates how one can measure the magnitude and phase relationship between input and output sinusoid signals. This technique can be used to build a Bode plot, or to calculate the complex value of the transfer function ($H(i\omega) = Ae^{i\phi}$)for a given frequency ω.

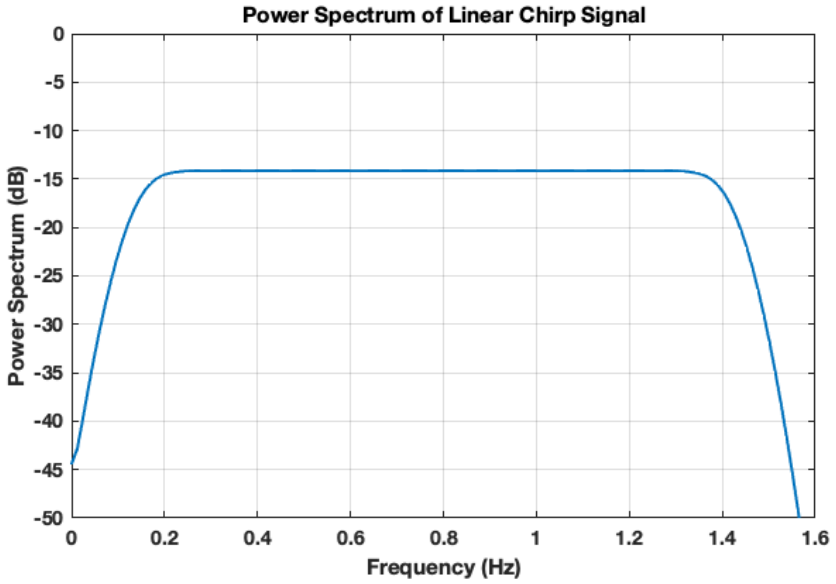


Appendix D: Chirp (Swept-Sine) Function

A (linear) chirp function is a sinusoidal function that has its frequency vary in a linear manner between an initial and final frequency. It is usually represented in the following time domain form, where T is the sweep time, ω_0 is the initial frequency and ω_1 is the final frequency.

$$\sin\left(\omega t + \frac{\omega_1 - \omega_0}{2T} t^2 + \phi_0\right)$$





Appendix E: Extra Credit Notes: White Noise

White noise refers to a process for sampling data that has zero mean and a fixed variance. It's called white because its power spectrum density has no unique features and is perfectly flat. White noise is formed (in a discrete system) by a chain of points from a normal distribution, which is independent and identically distributed (IID). That's a lot of maths, but the part that is useful for us is that white noise has a constant power spectral density - this means that the frequency domain is flat (for infinite sample length - I've taken a reasonably long sample length and you can see this effect in the plot below).

So, if we have something that is flat in the frequency domain, why is this useful? Well, if a sine wave is a delta function in the frequency domain, building a Bode plot from sine waves involves exciting the system with a train of different delta functions (this is what we did in this lab with the chirp function), and then combining the outputs.

However, wouldn't it be nicer and faster to just test all the frequencies at once? This is what the constant PSD lets us do - we can then reconstruct the behaviour at every frequency from the sample points that occur at each frequency over the sample time, to produce an equivalent plot in theoretically less time.