LABORATORY EXPERIMENTS

Introduction to the Motor Experiments

General description

Experiments are performed with electric motors using PC's for data acquisition and control. Computer-control is achieved through algorithms coded in C. Data is analyzed using $Matlab$. The basic set-up for the experiments is shown in Fig. 8.21. A brush DC motor is shown, with an encoder mounted on the shaft of the motor to determine the angular position. A voltage is applied to the motor through a power amplifier and a digital to analog (D/A) converter located on a board inside the PC. The encoder signals are decoded by a board, also located inside the PC.

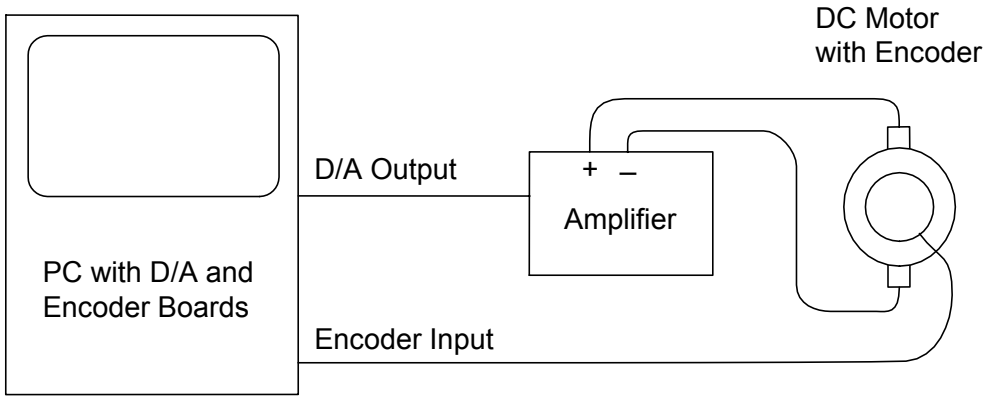

Figure 8.21: Set-up for motor experiments

Hardware configuration

- *Personal Computer:* Computing is provided by computers with a *Pentium* motherboard. The computers operate under *Windows* 95, but the C-programs run in a *DOS* window. *Matlab* is used for data analysis. For digital control, a fixed sampling rate is set using the real-time clocks of the computers. A sampling rate of 500 Hz is selected by default.

- *Digital to Analog Board:* A D/A board from *Omega* provides two channels of analog output, with a range of $\pm 5V$ and a resolution of 12 bits.

- *Encoder Board:* A board from *Dynamics Research Corporation* is used to decode the encoder signals. The encoders provide two signals, labelled cos and sin, which generate 500 pulses per revolution of the shaft. The two signals are decoded to produce a direction signal and a position signal with 2000 pulses per revolution. The resolution is therefore 0.18 degrees. The pulses are counted in a 24-bit register accessible to the PC. It takes $4,194$ turns in either direction to overflow the counter.

- *Power Amplifiers:* Power amplifiers were built in the Department of Electrical & Computer Engineering at the University of Utah. The amplifiers are linear amplifiers based on the power op-amp LM-12 from *National Semiconductor Corporation* and are voltage to voltage converters with a gain of 5. The maximum output voltage is $25V$, and the maximum continuous current is $2A$. The peak current is limited to $13A$ by the power op-amps. Two power amplifiers are integrated in a single box, enabling the control of two-phase motors in ECE 5570.

- *Motors:* Brush DC motors are used in ECE 3510. In ECE 5570, stepper motors and induction motors are also used. Encoders are mounted on the back of the brush DC motors to measure angular position. Stepper motors and induction motors are either connected to DC motors or to encoders mounted on a separate frame.

Software configuration

The software needed to operate the testbeds for data acquisition and control is provided, in order to enable you to concentrate on the experiments. Nevertheless, it may be useful to gain some limited understanding of the programs. All the routines are written in C (there is no assembly code).

D/A board

The D/A board is accessed using the routine

$$void\ put\_da\ (int\ channel, float\ voltage)$$

where *channel* defines the output channel to be accessed, and *voltage* is the voltage to be applied. There are two channels, labelled 0 and 1. The procedure automatically clips the

output voltage to the allowable range of $-5V$ to $5V$, and returns the variable *voltage* within that range. The routine *put_da*() is part of the file $INOUT.H$.

Two additional programs are provided that use *put_da()*. The program DAZ sets both D/A outputs to zero. **This function is useful because the D/A outputs are reset to maximum voltage (as opposed to zero) every time the computer is turned on**. The program $DATEST$ allows the user to enter a channel number and a voltage on the keyboard, for conversion by the D/A. The program continues to read voltage entries until interrupted by Ctrl-C (or Ctrl-Break).

Encoder board

The encoder board is initialized by using the function

$$void\ encoder\_init\ ()$$

The function specifies some encoder parameters and resets the counter on the board to zero. The value of the counter can then be read by using the function

$$float\ get\_encoder\ ()$$

The position of the shaft is returned as a float variable with the units of radians. The routines *encoder_init* () and *get_encoder* () are part of the file $INOUT.H$.

A program $ENCTEST$ is provided to test the proper operation of the encoder and encoder board. The program reads the value of the angular position given by the encoder and prints it on the screen. The program continues indefinitely until interrupted by Ctrl-C (or Ctrl-Break).

Timer

A program is available that provides the functions required to run programs at specified sampling rates. The program for timer handling is available in the file $TIMER.H$. The program requires the definition of 5 user routines:

- *void user_init ()* is a routine that initializes the variables accessed by the user routines, and that sets the sampling frequency (through the float variable $freq$). Typically, the D/A and encoder boards are also initialized by the routine.

- *void user_task ()* is the user routine that is executed at the sampling rate. Typically, the procedure reads the encoder position, performs a series of calculations, and sends voltages to the D/A's. Data may also be stored temporarily in RAM.

- *void user_interface ()* is a routine that is executed continuously in the foreground. It is typically used to adjust some parameters of *user_task ()* in real-time, through keyboard entry. For example, reference values of the controller may be set in this manner.

- *void user_abort ()* is a routine that is executed when the program is aborted using the Ctrl-C or Ctrl-Break keys. Typically, the procedure resets the D/A's to zero and saves data previously stored in RAM to a disk file.

- *void user_terminate ()* is a routine that performs similar functions as *user_abort()* and is not used in this course.

Additional information

The teaching assistant will give information on how to run the programs. It will be helpful to bring a floppy disk to store the data collected. The compiler program is called *tc.exe*. To run the program, first open a *DOS* window: from the *Start* button, click on *Programs*, and then on $MS-DOS\ Prompt$. Type $Alt-Enter$ to return to the Windows format, if needed. Then, simply type *tc* to run the compiler. Some basic commands are as follows: $Alt-F$ loads a file, $Alt-C$ brings a set of compilation options (use the "Build all" option to create an executable program), $Alt-E$ calls an editor, and $Alt-X$ quits.

To run *Matlab*, simply click on the *Matlab* icon. The programs store data in a file named *data.m*. The data is then loaded in *Matlab* by typing *data*. Be sure to rename the file *data.m* appropriately after each experiment is succesfully completed. You may then store the file on a floppy disk.

Some labs require that the motor velocity be computed and plotted. Because the velocity is reconstructed from the position measurements, which are quantized, the reconstructed velocity will appear noisy. Filtering may be applied to reduce the noise. In some cases, however, large glitches may be observed, which are not due to quantization but rather to incorrect time measurements. This problem can be avoided (or at least reduced) by restarting the computer in *DOS* mode. From the *Start* button, click on *Shut Down*, and then on $Restart\ in\ MS-DOS\ mode$. After completing the experiment, return to *Windows* by typing *exit*.

If you believe that a component (amplifier, motor,...) is defective, please talk to your teaching assistant. If he or she concurs, please write a note explaining the problem and get a replacement from the stockroom attendant.

First-Order Systems

Objectives

The objective of this lab is to study the characteristics of step responses of first-order systems. A brush DC motor is used for the experiments. Concepts of time constant and DC gain are introduced. Discrepancies between the idealized first-order model and the actual responses are observed. An objective of the first lab is also to become familiar with the hardware and the software in the lab.

Introduction

The response of a brush DC motor with the voltage v (in V or volts) considered as an input and the angular velocity ω (in rad/s or s^{-1}) considered as an output, is approximately described by a first-order differential equation

$$\frac{d\omega}{dt} = -a\omega + kv. \tag{8.1}$$

In the Laplace domain, the transfer function is given by

$$P(s) = \frac{\omega(s)}{v(s)} = \frac{k}{s+a}. \tag{8.2}$$

The parameter a is the pole of the system and has the dimension of rad/s or s^{-1}. The parameter k has the dimension of $V^{-1}s^{-2}$. The *time constant* of the system is the parameter T given by

$$T = 1/a. \tag{8.3}$$

T has the units of seconds. The *DC gain* of the system is given by

$$P(0) = \frac{k}{a}, \tag{8.4}$$

and has the units of $V^{-1}.s^{-1}$ or $(rad/s)/V$. In practice, a common unit is rpm/V, where rpm refers to revolutions per minute.

The step response of the system is the response for a step of input voltage

$$v(t) = v_0, \qquad t \geq 0. \tag{8.5}$$

For a step input, the output of the system is given by

$$\omega(t) = \frac{k}{a} v_0 \left(1 - e^{-t/T}\right). \tag{8.6}$$

The steady-state value of the output is given by $(k/a)v_0$, *i.e.,* the DC gain multiplied by the steady-state value of the input. Therefore, the DC gain indicates how much voltage is needed to reach a certain speed.

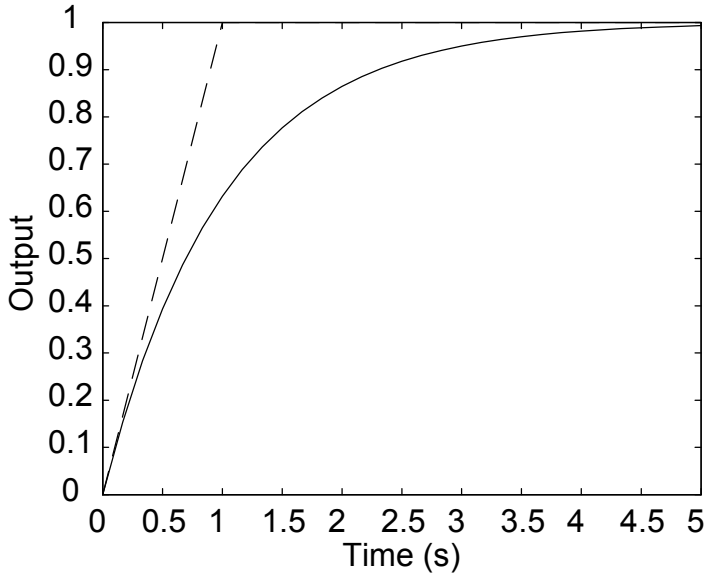

Figure 8.22: Step response of a first-order system

The step response of a system with transfer function $P(s) = 1/(s+1)$ is sketched in Fig. 8.22. On the graph, one finds that the tangent to the output at $t = 0$ reaches the steady-state value of the step response for $t = T = 1s$. It also turns out that, for $t = T$, the output reaches 63% of its final value ($1 - e^{-1} \simeq 0.63$).

Pre-lab

Using partial fraction expansions, verify that the step response of the first-order system (8.2) is indeed given by equation (8.6) and also prove the property that the tangent at $t = 0$ reaches the steady-state value for $t = T$.

Experiments

Equipment needed

You will need a brush DC motor, a dual power amplifier, a voltmeter, and a cable rack. Also check out a small wheel and a large wheel to be mounted on the shaft of the motor later.

Preliminary testing

Be sure to read the hand-out introducing the motor experiments for background information. Compile and create executable programs for the files DAZ.C, DATEST.C, ENCTEST.C, and LAB1.C. Then, perform the following tests in the order indicated.

- Check the operation of channel 0 of the D/A converter using DATEST. Measure the voltage with a voltmeter and vary the voltage using the program. Terminate the program with a nonzero voltage and check that DAZ resets the voltage of the D/A to zero.

- Connect channel 0 of the D/A converter to one of the channels of the amplifier and check the correct operation of the amplifier using DATEST. Also verify that the value of the amplifier gain is 5 using the voltmeter.

- Connect the amplifier to the motor, and check the operation of the DC motor using DATEST. The full voltage may be applied to the motor if it is not locked.

- Connect the encoder input cable from the PC to the encoder plug on the motor support bracket. Check the operation of the encoder by manually spinning the motor and reading the position using the program ENCTEST.

Step response

The program LAB1 will allow you to apply voltages to the motor through keyboard entries. Note that the keyboard entry in this program specifies the voltage to be applied to the motor and accounts for the value of the amplifier gain. The program stores 4000 samples at 500 Hz, which translates into 8 seconds of data. Operation starts after the first keyboard entry, and continues until Ctrl-C or Ctrl-Break is hit. Beyond 8 seconds however, no data is stored. The file DATA.M contains the results, and the data is available in Matlab simply by typing “data”. The time, voltage, and position variables are stored in variables ”t”, ”vol” and ”pos”, with units of seconds, volts, and radians respectively.

Using the program LAB1, apply a step input of 25V. Interrupt the program after about a second using Ctrl-C. In Matlab, observe the data. Reconstruct numerically the velocity as a function of time, using the formula

$$\omega(t) \simeq \frac{pos(t) - pos(t - T_S)}{T_S} \tag{8.7}$$

where T_S is the sampling period (2 ms for a 500 Hz sampling frequency).

Determine the steady-state velocity by taking the mean of the velocity over a period of time where it is (approximately) constant. Plot the response over 0.05s, and compare it to the response shown in Fig. 1 (the scales will be different, but the shape should be comparable). Obtain an estimate of the time constant based on the time it takes for the output to reach 63% of its steady-state value. From the results, calculate the values of the DC gain and of the parameters k and a of the system. Also give the DC gain in rpm/V. Repeat the experiments after mounting the small wheel on the shaft of the motor, and then again with the large wheel (get the appropriate screwdriver from the TA or lab attendant). Discuss the effect of added inertia on the motor parameters.

These simple experiments should demonstrate the validity of the first-order model for the DC motor, as well as its limitations. Taking a close look at the step response around $t = 0$ should reveal that the response is not linear with respect to time, contrary to Fig. 8.22. A more detailed model of the DC motor would account for the inductance of the motor, and for the time that it takes for the current to change. A second-order model, rather than first-order, would result and would more accurately reflect the behavior of the motor. In practice however, any model, no matter how high order or how detailed it is, is only an approximation of the real system.

Report at a glance

Be sure to include:

- Pre-lab calculations.
- Plots of the step responses.
- Values for the steady-state velocity, the time constant, the DC gain, a and k. Give the units for all quantities, and report the DC gain both rad/s and in rpm/V.
- Comments.

Second-Order Systems

Objectives

The objective of this lab is to study the characteristics of step responses and of sinusoidal responses for second-order systems. Critically damped and underdamped systems are considered. Concepts of rise time, settling time, percent overshoot, and frequency of oscillations are introduced for step responses. For sinusoidal inputs, the steady-state responses are studied first, and peaking in the frequency domain is observed for underdamped systems. Transient responses are also investigated.

Introduction

In the lab on first-order systems, the response of a brush DC motor with the voltage v (V or volts) considered as an input, and the angular velocity ω (rad/s or s^{-1}) considered as an output, was found to be approximately described by a first-order model

$$P(s) = \frac{\omega(s)}{v(s)} = \frac{k}{s+a}. \tag{8.8}$$

If θ, *the angular position*, is considered to be the output, a slightly different model results. Since the angular position is the integral of the velocity, a second-order transfer function is obtained

$$P(s) = \frac{\theta(s)}{v(s)} = \frac{k}{s(s+a)}. \tag{8.9}$$

Note that the second-order system has a pole at $s = -a$ and a pole at $s = 0$.

We consider the motor under a feedback control law of the form

$$v(t) = k_P(r(t) - \theta(t)), \tag{8.10}$$

where k_P is a parameter to be selected, called the *proportional gain*. The objective is for $\theta(t)$ to track $r(t)$, the reference input, which is now considered to be the input to the system. The control law is called a *proportional* control law because the control input is proportional to the error between the reference input and the output. In the Laplace domain, equation (8.10) becomes

$$v(s) = k_P(r(s) - \theta(s)). \tag{8.11}$$

The transfer function from the input r to the output θ can be verified to be

$$P_{CL}(s) \triangleq \frac{\theta(s)}{r(s)} = \frac{kk_P}{s^2 + as + kk_P}. \tag{8.12}$$

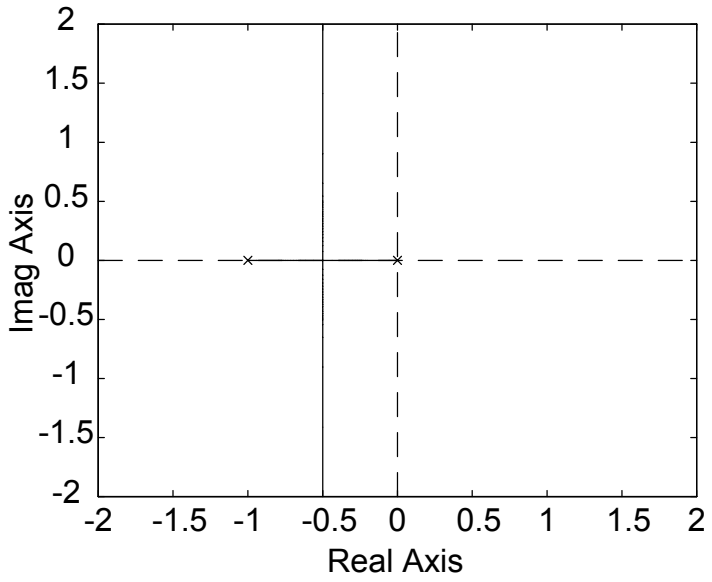

Figure 8.23: Root-locus for second-order system

For different values of the parameter k_P, various second-order systems are obtained. Fig. 8.23 shows the locus of the poles of the transfer function (8.12) as k_P varies from 0 to ∞, assuming $k = 1$ and $a = 1$. For small k_P, the two poles are close to the original values at $s = 0$ and $s = -1$. As the gain increases, the two poles move closer together, eventually merging and splitting from the real axis as a complex pair. For larger values of k_P, the real parts of the poles remain constant, and the imaginary parts continue to grow. A system for which the two poles are real is called *overdamped*. If the two poles are complex, it is called *underdamped*. The limit case where the poles are real and equal is called *critically damped*. This situation occurs for $a^2 = 4kk_P$.

The step response of a system is the response to an input $r(t) = r_0$. Fig. 8.24 shows the step responses for two values of k_P. The magnitude of the input is $r_0 = 1$. The dashed line is the response that corresponds to $k_P = 0.25$, the value such that the two poles are real and equal (critical damping). The solid line corresponds to $k_P = 1.25$, the value such that

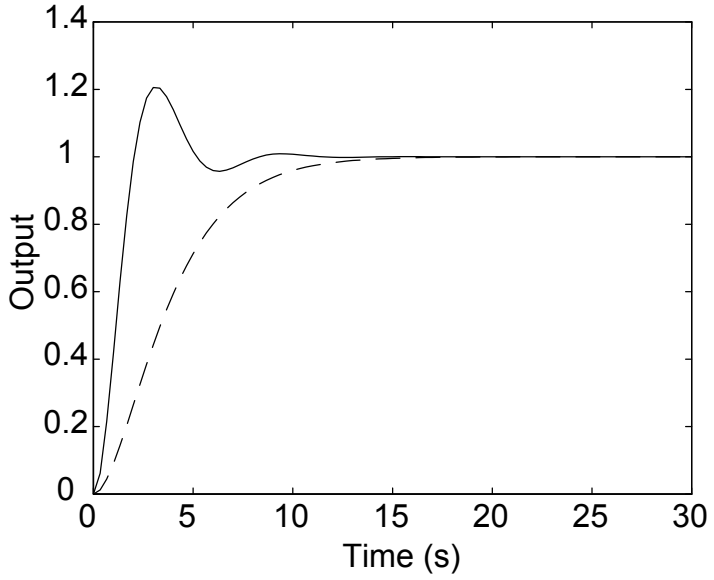

Figure 8.24: Step responses of second-order systems

the poles are complex and with imaginary parts equal to twice the real parts (underdamped). When the poles are complex, with imaginary parts greater than or equal to the real parts, the response exhibits overshoot and oscillations. The angular frequency of the oscillations is equal to the imaginary part of the poles (in rad/s). In Fig. 8.24, the period of the oscillations is approximately 6 seconds, and the imaginary part of the poles is 1 rad/s.

One defines the *rise time* as the time it takes for the output to reach the steady-state value. Because the value is only reached asymptotically, it is common to define the *10-90% rise time*, as the time it takes for the output to move from 10% to 90% of the steady-state value. The *settling time* is the time it takes for the output to reach and stay within a certain percentage of the final value. Several values of the percentage are used, with 2% being common. We will refer to this settling time as the *98% settling time*. One also defines the *percent overshoot* as the percentage of overshoot over the steady-state value.

In this lab, we also consider the responses to sinusoidal inputs

$$r(t) = r_0 \sin(\omega_0 t + \phi_0) \tag{8.13}$$

In the steady-state, the response of the system is given by

$$\theta_{ss}(t) = M(\omega_0)\ r_0 \sin(\omega_0 t + \phi_0 + \alpha(\omega_0)), \tag{8.14}$$

where the magnitude $M(\omega_0)$ and the phase $\alpha(\omega_0)$ satisfy

$$M(\omega_0) = |P_{CL}(j\omega_0)|\,, \qquad \alpha(\omega_0) = \arg(P_{CL}(j\omega_0)). \tag{8.15}$$

and P_{CL} is the transfer function from r to θ given in (8.12).

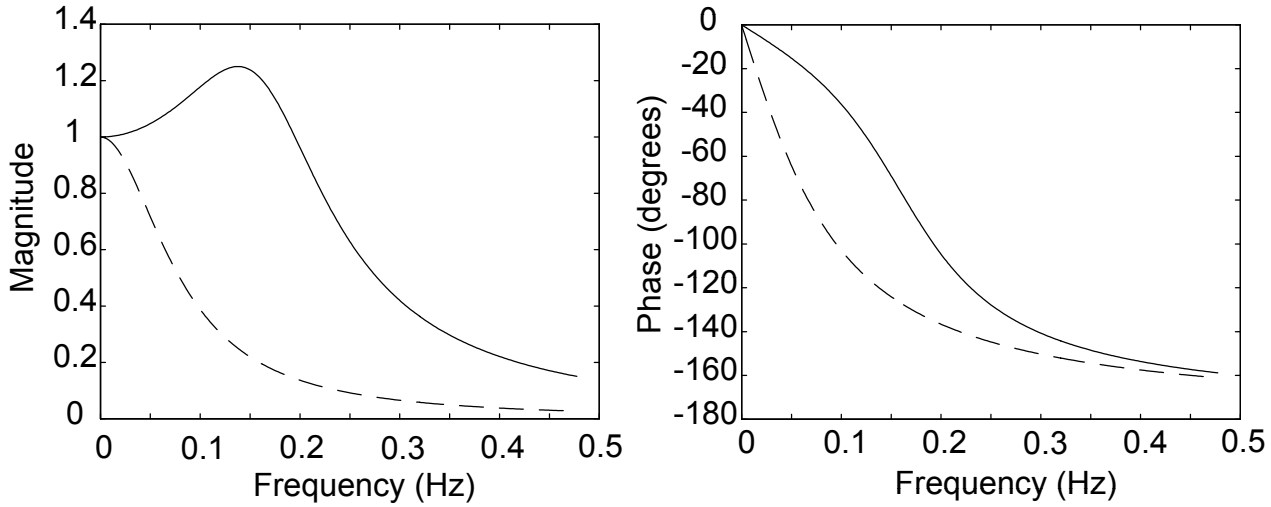

Figure 8.25: Frequency responses of second-order systems

Fig. 8.25 shows the frequency responses of the two second-order systems for $k_P = 0.25$ and $k_P = 1.25$, assuming $a = k = 1$. The dashed lines are for $k_P = 0.25$ (critical damping) and the solid lines are for $k_P = 1.25$ (underdamped). On the left are the magnitudes of the frequency responses (that is, M in equation (8.15)), and on the right are the phases in degrees (that is, α in equation (8.15)). The x-axis has been labelled in Hz, rather than rad/s, with 0.16 Hz being approximately 1 rad/s. Note that, for the underdamped case, the magnitude of the frequency response peaks to a value about 25% higher than the DC value. The steady-state response for that frequency will, therefore, be higher than for any other frequency. This phenomenon is usually called *resonance*. For a lightly damped system, the frequency of peaking is close to the value of the imaginary part of the pole, which is 1 rad/s in this example.

Pre-lab

Verify analytically equation (8.12). Then, for the values of k and a corresponding to the DC motor (for simplicity, you may let $k = 1000$ and $a = 100$), calculate the values of k_P such that

the closed-loop poles are real and equal. Also compute k_P such that the imaginary parts are equal to twice the real parts. In Matlab, calculate and plot the step responses of the system for the two values of k_P. The Matlab function *step* may be used. You should check the help files for the functions *step* and *tf* to get the correct syntax.

Determine from the plots the values of the 10-90% rise time and of the 98% settling time. For the underdamped case, determine the percent overshoot and the frequency of oscillation. To obtain the frequency of oscillation, you may estimate the half period as the time between the first peak (overshoot) and the second peak (undershoot). Compare the frequency of oscillation to the imaginary part of the pole.

In Matlab, calculate and plot the frequency response of the DC motor under proportional control, for the critical and for the underdamped cases. In other words, plot the equivalent of Fig. 8.25 for the specific values of the motor. The Matlab function *freqs* may be used for that purpose. With *freqs*, you will need to enter the frequencies in rad/s, but be sure to label your plots in Hz, as in Fig. 8.25. From the plots, determine the value of the frequency where peaking occurs, and give the amount of peaking as a percentage of excess over the value of the DC gain. Also determine the phase lag of the response at the frequency of peaking.

Experiments

Equipment needed

You will need a brush DC motor, a dual power amplifier, and a cable rack.

Preliminary testing

Carry out the same testing procedure as in the lab on first-order systems. Transfer the files LAB2.C and LAB3.C to your account. Compile and create executable programs for these files.

Step responses

The program LAB2 lets you first enter a value of the proportional gain. Then, you apply the value of the reference input r through the keyboard, and may change it in real-time. The reference input is for the angular position of the rotor, and is given in degrees. Using the program, obtain the step responses for the critical and underdamped cases and for a reference input equal to 90°. From the plots, determine the rise times and settling times, and for the underdamped case, the percent overshoot and the frequency of the oscillations. Follow the same procedure as in the pre-lab. You should find that the overshoot and the frequency

of oscillation are somewhat higher than expected. The origin of the discrepancy is in the additional dynamics of the motor related to the inductance of the motor, an effect that was observed in the step responses of the lab on first-order systems, but was neglected in the model.

Sinusoidal responses

The program LAB3 lets you enter values for the proportional gain, for the magnitude of the reference input r_0, and for the phase ϕ_0. Then, the value of the frequency ω_0 is entered and may be changed in real-time. Note that the magnitude of the reference input is expressed in degrees (giving the reference angular position of the motor in degrees), and that the frequency of the reference input is given in Hz. All the experiments concern the underdamped case, so that the value of k_P that should be used is the value calculated for that case.

Test the program by applying a sinusoidal input with $r_0 = 10$ degrees, $\phi_0 = 0$, and a frequency of 1 Hz. Having checked that all works according to expectations, run an experiment where you step the frequency every second according to the sequence 5, 10, 15, 20, 25, and 30 (Hz). Plot the response and estimate the frequency of peaking, as well as the percentage of peaking (you may run separate experiments if you want to estimate the frequency of peaking more precisely). Compare the results to those predicted by the analysis. You should find that the percentage and the frequency of peaking are somewhat higher than expected. As for the step responses, this effect is due to the neglected dynamics related to the inductance of the motor. From the plot of the response with steps in frequency, measure the magnitude of the response for each frequency, and draw a crude plot of the magnitude of the frequency response as a function of frequency (in Hz) using the results.

Run a separate experiment at the frequency of peaking, and plot both the input and output signals on the same graphs. Estimate the phase lag of the frequency response α and compare to the value of the pre-lab. Next, run an experiment with a frequency of 50 Hz and a phase $\phi_0 = 0$ and observe that the transient response reaches values that are considerably higher than the steady-state value. However, the transient response varies significantly with the phase of the input signal. To illustrate this point, repeat the experiment with a different phase, adjusted to minimize or at least significantly reduce the transient response. Calculate the amount of overshoot, or peaking in the time domain, as a percentage of excess over the steady-state response (for $\phi_0 = 0$ and for the phase that minimizes the transient response).

Report at a glance

Be sure to include:

- Pre-lab calculations.
- Plots of the step responses, with the values of the rise time, settling time, percent overshoot, and frequency of oscillation.
- Plot of the response with steps in frequency. Value of the percentage of peaking in the frequency domain and of the frequency of peaking. Crude magnitude plot.
- Plot of the response at the frequency of peaking, with the value of α.
- Plot of the transient responses at 50Hz for two values of the phase. Percentage value of the peaking in the time domain.
- Comparison of the results of the pre-lab and experiments.
- Comments.

PID Control

Objectives

The objective of this lab is to study basic design issues for proportional-integral-derivative control laws. Emphasis is placed on transient responses and steady-state errors. The first control problem consists in the regulation of velocity for brush DC motors and is solved using proportional-integral control. The second problem consists in the regulation of position and requires derivative compensation in the form of velocity feedback.

Introduction

In the lab on first-order systems, the response of a brush DC motor with the voltage v (V or volts) considered as an input and the angular velocity ω (rad/s or s^{-1}) considered as an output was found to be approximately described by a model

$$P(s) = \frac{\omega(s)}{v(s)} = \frac{k}{(s+a)}. \tag{8.16}$$

A *proportional control law* (P) consists in having

$$v = k_P(r - \omega), \tag{8.17}$$

where r is the reference input for the velocity, in rad/s. k_P is called the *proportional gain*. The resulting closed-loop transfer function is given by

$$P_{CL}(s) = \frac{\omega(s)}{r(s)} = \frac{kk_P}{s + a + kk_P}. \tag{8.18}$$

Note that the closed-loop pole is given by $-a - kk_P$. In theory, it would appear that the closed-loop pole could be moved arbitrarily far in the left-half plane through the use of a sufficiently large proportional gain. The response of the system could be made arbitrarily fast in that manner. As this lab will show, there are limits on the gains that can be applied, however. These limits are due to effects that are neglected in the model (such as the inductance of the motor and the limit on the voltage), but are nevertheless present in the physical system.

Proportional-integral control for velocity tracking

The DC gain in (8.18) is equal to $kk_P/(a + kk_P)$. For large k_P, this gain approaches 1, but large gains are impractical. Therefore, it is useful to modify the control law in order to adjust the DC gain. Specifically, replacing (8.17) by

$$v = k_P(k_F r - \omega), \tag{8.19}$$

yields a closed-loop transfer function

$$P_{CL}(s) = \frac{\omega(s)}{r(s)} = \frac{k_F k k_P}{s + a + kk_P}. \tag{8.20}$$

The closed-loop pole is equal to the original one, but the DC gain can now be adjusted to 1 by setting

$$k_F = \frac{a + kk_P}{kk_P}. \tag{8.21}$$

We will call k_F the *feedforward gain.*

Despite the capability of adjusting the feedforward gain k_F in order to obtain a DC gain of 1, perfect tracking of reference inputs is usually not achieved because the parameters of the system are not exactly known or may vary, and because disturbances may affect the response of the system. These problems can be resolved through the use of a *proportional-integral* (PI) control law of the form

$$v = k_P(r - \omega) + k_I \int (r - \omega)dt, \tag{8.22}$$

where k_P and k_I are called the *proportional gain* and the *integral gain*, respectively. Then, the closed-loop transfer function becomes

$$P_{CL}(s) = \frac{\omega(s)}{r(s)} = \frac{kk_P(s + \frac{k_I}{k_P})}{s^2 + (a + kk_P)s + kk_I}. \tag{8.23}$$

The DC gain is equal to 1, regardless of what the parameters of the system or of the control law are. Of course, it should be remembered that the DC gain reflects the steady-state conditions only if the closed-loop system is stable, *i.e.,* if the poles of (8.23) are all in the open left-half plane. Generally, the responses cannot be made as fast for a PI control law, so that the benefit of a zero steady-state error has to be weighted against that of the speed of response.

Proportional-integral-derivative control for position tracking

To control *position*, instead of velocity, it is common to use of *proportional-integral-derivative* (PID) control law

$$v = k_P(r - \theta) + k_I \int (r - \theta)dt + k_D \frac{d}{dt}(r - \theta). \tag{8.24}$$

Note that the derivative term can be viewed as a proportional feedback acting on the velocity error. In general, derivative feedback improves the stability and the damping of the closed-loop system.

In practice, the control law (8.24) is often modified in two ways. First, the derivative action is applied only to the output θ, and not to the reference input. This is done because reference inputs often change in steps, and the derivative is then either zero or not defined (infinite). Second, a feedforward gain is often applied to the reference input. This is not done to adjust the DC gain (as for the control law without integral term), but rather to place the zero of the closed-loop transfer function. This will be explained shortly. The modified control law is given by

$$v = k_P(k_F r - \theta) + k_I \int (r - \theta)dt - k_D \frac{d}{dt}\theta. \tag{8.25}$$

The closed-loop transfer function for the system with transfer function

$$P(s) = \frac{\theta(s)}{v(s)} = \frac{k}{s(s + a)}, \tag{8.26}$$

and the PID control law (8.25), is given by

$$P_{CL}(s) = \frac{\theta(s)}{r(s)} = \frac{kk_F k_P(s + \dfrac{k_I}{k_F k_P})}{s^3 + (a + kk_D)s^2 + kk_P s + kk_I}. \tag{8.27}$$

Note that the closed-loop transfer function (8.27) has three poles. There is also a zero at $-k_I/(k_F k_P)$. For the original control law with $k_F = 1$, the zero may have a small magnitude compared to the closed-loop poles, yielding overshoot in the step response even if the closed-loop poles were well-damped. Reducing the value of k_F allows one to push the zero farther in the left-half plane and to improve the step response.

Pre-lab

Derive equation (8.20) and calculate values of k_P and k_F such that the closed-loop pole is at an arbitrary location $-b$ and such that the DC gain is 1. Specialize the results to the cases

$b = 2a$, $b = 6a$, and $b = 11a$. Calculate the specific values of the gains for the DC motor ($a = 100$, $k = 1000$) for all three cases.

Derive equation (8.23) and calculate values of k_P and k_I such that the closed-loop poles are both at an arbitrary location $-b$. Specialize the results to the case where $b = a$, and to the specific values of the DC motor.

Derive the transfer function given in (8.27) and calculate the values of the PID parameters such that all three poles are placed at some $-b$. Calculate the parameters that correspond to $b = a$, and also for the specific motor parameters ($a = 100$, $k = 1000$).

Experiments

Equipment needed

You will need a brush DC motor, a dual power amplifier, and a cable rack.

Preliminary testing

Carry out the usual testing procedure. Transfer the file LAB4.C to your account, and create an executable program for the file.

Proportional control

The program LAB4 lets you enter values for the proportional, integral, and feedforward gains. Then, you enter a value for the reference input, and may change it in real-time. The reference input is a reference speed, given in rpm. First, experiment with proportional control by letting $k_I = 0$. Set k_P and k_F according to the pre-lab calculations, and apply a reference input that steps from 0 to 1000 rpm and then to 2000 rpm and then back to zero. Repeat the experiment for all three cases and plot the results. Discuss what happens when the gain k_P becomes large.

Proportional-integral control

Apply the values calculated for the PI control law, setting $k_F = 1$ in the program. You may also experiment with other values of k_P and k_I, in particular those resulting in faster responses. Plot the results for your best experiment.

Proportional-integral-derivative (PID) control

Adjust the program LAB4.C in order to:

- Read the derivative parameter k_D from the keyboard (change a *printf* statement and add a *scanf* statement in *user_init*).
- Read the reference position in degrees (change a *printf* statement in *user_init* and compute r in rad in *user_task*).
- Implement the PID control law for position (change the computation of u in *user_task*).
- Declare the new variables at the beginning of the program.

Once this is done, you may apply the calculated values of the PID parameters, with $k_F = 1$, and a step of reference input of 90°. The settling time should be approximately 100ms, with an overshoot of the response. Adjusting the parameter k_F should yield a better response. Plot the results for a few values of k_F on a single graph. Indicate what value of k_F gives the best response (minimum settling time with negligible overshoot).

DEMONSTRATE YOUR FINAL EXPERIMENT TO THE TA

Report at a glance

Be sure to include:

- Pre-lab calculations.
- Plots of the responses with the proportional control law, for the three cases.
- Plot of the response with the proportional-integral control law, with the values of the gains that were used.
- Plot of responses with PID control law, and a few values of k_F.
- Written note from the TA that the program worked.
- Comments.

Basic Phase-Locked Loop

Objectives

The objective of this lab is to learn the basic concepts of operation of phase-locked loops (PLL). Experiments cover the measurement of the gain of a voltage-controlled oscillator and the construction of a PLL with a first-order filter. PLL properties such as capture range, hold range, transient response, and steady-state ripple are measured, and correlated with analysis results.

Introduction

A *phase-locked loop* (PLL) is a device in which a periodic signal is generated and its phase is locked to the phase of an incoming signal. Phase-locked loops are used for the demodulation of frequency-modulated signals, for frequency synthesis, and for other applications. The principles of operation of phase-locked loops are discussed in the course notes. Familiarity with the relevant equations is assumed.

Pre-lab

The pre-lab consists in computing the gain of phase detector I of the phase-locked loop chip used in the lab. In the notes, a phase detector based on a multiplier was discussed. Here, the phase detector is a logical XOR operator. The two signals entering the phase detector are assumed to be square waves with logic levels at 0 and V_s (in Volts). The output of the XOR phase detector also varies between 0 and V_s.

Assuming that the two input signals have the same frequency, sketch the output signal when the inputs signals have a phase difference of 0°, 90°, and 180°. Then, plot the *average value* of the output signal for a phase difference ranging from $-180°$ to 180°. Show that, within the range $0 \rightarrow 180°$, the average value of the output of the phase detector satisfies

$$\phi = k_{pd}\,(\theta - \theta_{vco}) \tag{8.28}$$

Give the value of k_{pd} in V/rad when $V_s = 12V$. The voltage-controlled oscillator (VCO) of the PLL chip is biased so that the center frequency is produced when the applied voltage is $V_s/2$. What phase difference produces this output of the phase detector? The phase difference is the one that will be observed when the PLL is locked and there is no frequency error.

Laboratory

The laboratory covers two main tasks: measuring the gain of the VCO and designing a PLL with first-order filter. Parts of this lab and of the next lab use the same circuits. Therefore, review the schematic of the next circuit to be built before disassembling the current one. There is a low temperature coefficient capacitor (3900pF) that is needed in the lab. It is necessary to insure repeatability in the experiments. The small epoxy coated, shiny, smooth, light brown capacitors with a 392 printed on them should work fine. Do not use the bigger, dull, brown rectangular or disc-shaped capacitors. The 39 stands for 39 and the 2 means to add two zeros to get 3900pF.

Measuring k_{vco}

EQUIPMENT NEEDED (use a bench having the equipment, if available): oscilloscope, frequency counter, function generator, power supply, and a DMM.
EQUIPMENT TO BE CHECKED OUT: wire kit, two 10x probes, and a Wavetek function generator.
PARTS NEEDED: CD4046 CMOS PLL IC, 3900pF low temp. coef. capacitor, 2 resistors that will be determined in the lab, $0.1\mu F$ cap., $10k\Omega$ multi-turn POT (with "tweaker"), $10k\Omega$ and $1k\Omega$ resistors, proto-board and wires.

Begin by calibrating the 10x probes to your oscilloscope. If you are unsure of how to do this, check out the booklet titled "The XYZ's of Using An Oscilloscope," and read Chapter 8. The information is still useful even though the book is written for a different oscilloscope. If you are unsure where the probe adjustment is, check with your classmates, the TA, or the stockroom attendant. Then, set the power supply to 12 volts DC.

A WORD ABOUT STATIC
The PLL IC that you will be using is a CMOS part and is static sensitive. Following a few precautions will avoid zapping your IC. First, ground yourself to your circuit before inserting the IC or working on the circuit. Second, never assemble or change your circuit with the power applied. Third, connect signal and power sources ground lead first. Fourth, apply circuit power first, then signal sources. To change your circuit, remove signal sources first, then power. Last, disconnect signal and power sources ground lead.

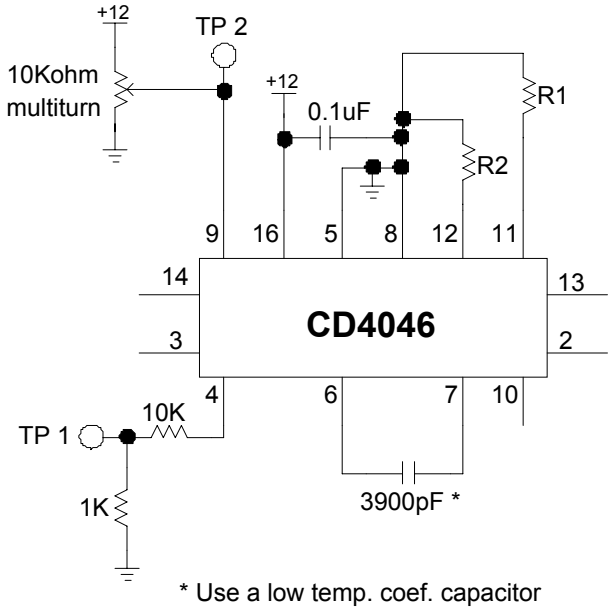

Figure 8.26: Schematic used to measure the gain (k_{vco}) of the VCO

Assemble the circuit shown in Fig. 8.26. The VCO frequency range will be from 30kHz to 50kHz. The datasheet for the CD4046 indicates that this range is achieved for R1=18$k\Omega$, R2=18$k\Omega$, and C=3900pF. Be sure to keep the 0.1μF capacitor as close to the PLL IC as possible without trimming the leads.

The significant pins of the PLL chip are:

- pin 14: input #1 of the phase detector.
- pin 3: input #2 of the phase detector.
- pin 2: output of phase detector I.
- pin 4: VCO output.
- pin 9: VCO input.

AC COUPLING OR DC COUPLING?
Incorrect channel coupling can wreak havoc on measurements and cost you a lot of time. AC coupling places a high-pass filter in series with the probe to remove the DC component from the measurement. Use AC coupling if you are measuring high frequencies that have large DC offsets. However, AC coupling at low frequencies can lead to phase and amplitude errors. For accurate measurements below 300Hz, use DC coupling. It is best to use DC coupling whenever you can because it is so easy to forget that you are on AC coupling.

Having built the circuit, connect the DMM to TP 2 and the frequency counter to TP 1. Adjust the VCO control voltage with the $10k\Omega$ POT, and make a plot of frequency *vs.* voltage. Determine the range of control voltage that results in a linear VCO response and determine the gain of the VCO, or k_{vco} (in Hz/V), in that range. Deduce the value of the loop gain, $k_{pll} = 2\pi k_{vco} k_{pd}$, using the value of k_{pd} determined in the pre-lab. Note that the transfer function from the VCO input to the output of the phase detector is then

$$P(s) = \frac{k_{pll}}{s} \tag{8.29}$$

and constitutes the "plant" to be controlled.

Basic PLL

In this section, you will build a PLL circuit with first-order loop filter and measure its characteristics. Considering the circuit shown in Fig. 8.27, show that the transfer function of the loop filter (which takes the role of the compensator) is

$$C(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{k_f}{s + a_f} \tag{8.30}$$

and give k_f and a_f as functions of Rf and Cf. Using the value of the gain k_{pll} determined in the previous section, determine the condition that Rf and Cf must satisfy so that the closed-loop poles have damping $\varsigma = 0.707$ (partial answer: $a_f^2 = 2k_{pll}k_f$). From the list below, find the correct combination of Rf and Cf:

a) $10k\Omega$ and 3300pF
b) $18k\Omega$ and 2200pF
c) $15k\Omega$ and 4700pF
d) $18k\Omega$ and 820pF
e) $51k\Omega$ and 1000pF
f) $15k\Omega$ and 390pF
g) $22k\Omega$ and 470pF
h) $120k\Omega$ and 4700pF

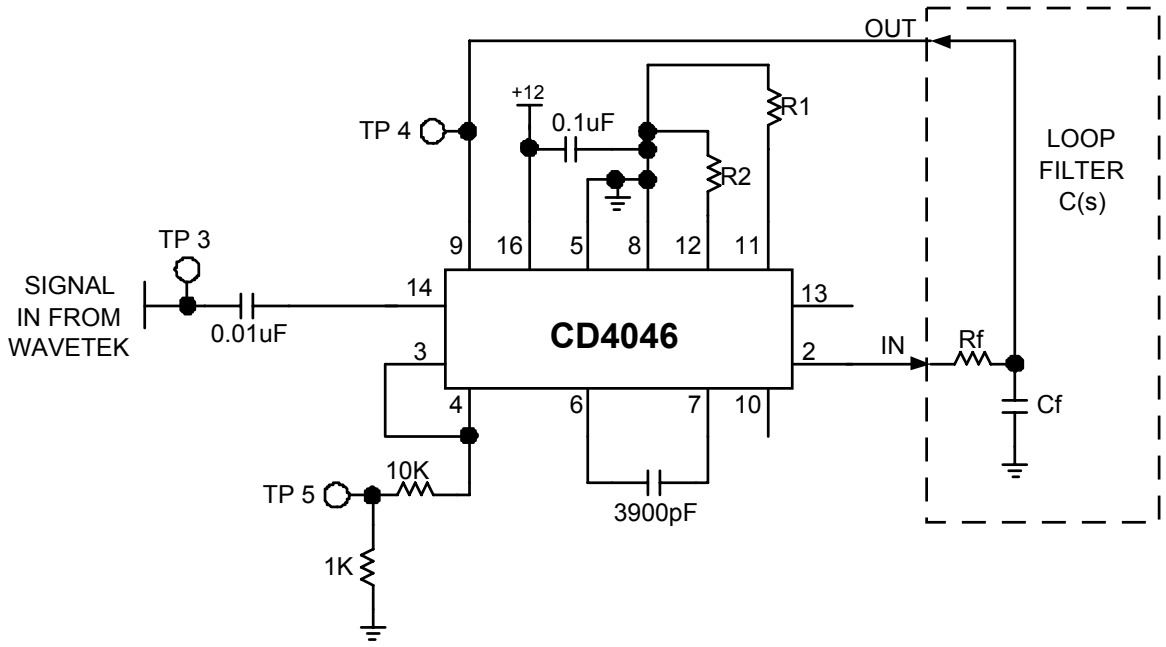

Figure 8.27: Schematic of a PLL with first-order filter

Verify your choice with the TA and build the circuit of Fig. 8.27. The main added parts from the previous circuit are Rf and Cf, but note the small changes. Apply circuit power, set the Wavetek to 40kHz, 8 volts p-p sinusoid, and connect it to the circuit. Observe the signal input on TP 3 and the VCO signal on TP 5. Trigger the scope on TP 3. Also, for your convenience, connect the frequency counter to the input signal TP 3. With this set-up, the two waveforms on the scope should appear "in sync." In other words, the VCO waveform should be stationary and at the same frequency as the input waveform (the PLL is locked). If not, double check your PLL circuit and the loop filter values.

Now, slowly decrease the input frequency. As you drop below 30kHz or so, you should notice that the VCO signal looses sync with the input signal. To explain what happened, set the input back to 40kHz and measure the voltage at TP 4 with the DMM (set to DC). TP 4 is the VCO control voltage input. It is the voltage that you measured when making the plot of VCO frequency *vs.* control voltage. Again, decrease the input frequency and observe the voltage on the DMM. Now, can you explain what happens when the input frequency drops below 30kHz? Compare to what happens when you adjust the input frequency above 50kHz. Between 30kHz and 50kHz, does the VCO frequency track the input frequency?

A QUICK REVIEW OF OSCILLOSCOPE PHASE MEASUREMENT

Measuring phase on the oscilloscope is quite easy. It is simply the measurement of the time between two points on two waveforms. The time measured is divided by the signal period and multiplied by 360 degree to give the phase difference. Specifically, set the scope to view two channels simultaneously. Set the trigger source to the channel you wish to be the reference channel. For this lab, the trigger should be set to the rising edge. Choose a point on the reference signal. Designate this point as the $t = 0$ point and position the point so that it crosses the center horizontal graticule. Find the same point on the second channel signal and position it vertically until it crosses the center horizontal graticule. Measure the time delay between the two points using the marks on the horizontal graticule and the horizontal time base. If the signal frequency is known, multiply the time measured by the frequency (equivalent to dividing by the cycle period), and then multiply by 360 to obtain the phase difference in degrees.

Measure the PLL's hold range and capture range. The hold range is the range of input frequencies for which the PLL maintains phase lock. The capture range is the range for which the PLL acquires phase lock. To measure the hold range, start the input frequency at a point where the PLL is phase-locked, then reduce the input frequency until the PLL looses lock. The frequency is the lower edge of the hold range. The upper edge is obtained similarly by raising the input frequency. To measure the capture range, start the input frequency at a point where the PLL is not phase-locked, and raise the frequency until the PLL acquires phase lock. The frequency is the lower edge of the capture range. The upper edge is obtained by lowering the incoming frequency from a frequency above the capture range. The frequency counter should help in quickly determining the input frequency. If the frequency adjust knob is too coarse of an adjustment, try using the frequency vernier knob for fine adjustments. Next, make a plot of the input phase *vs.* input frequency over the range in which the PLL is locked (4 or 5 frequency points). Does the phase remain constant over the input frequency range? Can you explain the answer based upon the properties of phase-locked loops with first-order filters?

Next, set the input frequency to 40kHz and measure the VCO control voltage input (TP 4) with the scope. Is this control voltage a nice clean DC voltage? If not, how much ripple is present? What is the source of the ripple? Hint: remember the time constant of the loop filter, and use the scope to observe pin 2 on the PLL. In theory, the ripple could be decreased by lowering the bandwidth of the loop filter. However, the constants k_f and a_f cannot be set independently. To find out what would happen if the bandwidth of the RC filter was decreased, sketch the root-locus of the closed-loop poles as functions of the product Rf.Cf, and explain why increasing the time constant of the loop filter is not desirable.

Next, the Wavetek frequency will be modulated by a square wave to view the PLL step response. Set-up the extra generator (mounted in the workbench) for a frequency of 200Hz and a square wave output. With the output at minimum, connect the output to the VCG input of the Wavetek (lower left connector). Also, using the BNC "T", observe the bench generator output on one channel of the scope while observing TP 4 on the other channel. Trigger from the bench generator. Slowly increase the bench generator output. The square wave output is frequency-modulating the Wavetek output. If you adjust the bench generator output too high, the Wavetek output frequency will shift beyond the hold range of the PLL. With the Wavetek frequency knob still set to 40kHz, adjust the bench generator output level until TP 4 has a 5 volt p-p square wave. Describe the response of the VCO control voltage (TP 4) in terms of speed of response and overshoot.

This is the end of the basic PLL lab. Do not take apart your PLL circuitry. Most of it will be used in the advanced PLL lab.

Advanced Phase-Locked Loop

Objectives

The objective of this lab is to refine the design of the loop filter used in the basic PLL lab, and to evaluate the improvements in performance. A second-order filter replaces the first-order filter of the basic PLL lab, and a better phase detector is used. The selection of the filter is an example of control system design with integral action and lead compensation.

Introduction

The basic PLL lab ended with the testing of a basic PLL with first-order filter. While the experiments validated the design, the output signal was found to contain a large amount of ripple. Generally, such a high ripple is intolerable. In most communication systems, the ripple (or harmonic frequency content) of the loop filter output must be attenuated by 50 to 90dB. Unfortunately, the simple RC filter did not provide enough flexibility to improve the design. This lab demonstrates how a more sophisticated loop filter can reduce the output ripple while maintaining good tracking of the incoming signal.

Pre-lab

Show that the loop filter of Fig. 8.28 has transfer function

$$C(s) = \frac{V_{out}(s)}{I(s)} = \frac{k_f(s+b_f)}{s(s+a_f)} \tag{8.31}$$

Give the values of k_f, a_f, and b_f as functions of C_a, C_b, and R_b. Note that the input of the loop filter is a current, rather than a voltage. Accordingly, the output of the phase detector will be a current (the phase detector acts as a current source controlled by the phase error). Such a phase detector is called a *charge pump phase detector*. Assume that the "plant" transfer function is

$$P(s) = \frac{k_{pll}}{s} \tag{8.32}$$

with $k_{pll} = 2.435$ (A/V.s).

The three parameters of the loop filter are k_f, a_f, and b_f . In general, a_f needs to be greater than b_f (this type of control is called *lead compensation*). The locations of a_f and

b_f on the real axis determine the location of the closed-loop poles. We will let $a_f = 8.696\ 10^4$ rad/s, $b_f = 6.25\ 10^3$ rad/s. Plot the root-locus of the PLL system for varying k_f (use the Matlab function *rlocus* to check your results). Note that, for the values of a_f and b_f chosen, the root-locus has two break-away points. Compute the two break-away points, as well as the feedback gain and closed-loop poles associated with the break-away point that is reached first when the feedback gain increases. Deduce what the filter gain k_f should be. Using Matlab, plot the step response of the closed-loop system from the scaled modulating signal x_s to the VCO input signal x_{vco}. Interpret the results. Finally, from the values of k_f, a_f, and b_f , deduce the values of the filter parameters C_a, C_b, and R_b.

Laboratory

Construction and calibration of the charge pump phase detector

EQUIPMENT NEEDED (use a bench having the equipment, if available): oscilloscope, frequency counter, function generator, power supply, and a DMM.
EQUIPMENT TO BE CHECKED OUT: wire kit, two 10x probes, and a Wavetek function generator.
ADDITIONAL PARTS NEEDED: $1k\Omega$, $1k\Omega$ POT (with "tweaker"), 2 - 330Ω, 2 - 470Ω, $0.1\mu F$, 2N3904, 2N3906, 2 – 1N4148.

In this first section of the lab, you will build and calibrate a charge pump circuit to be used with the CD4046 phase detector II (pin 13). The phase detector that was used in the basic PLL lab was an XOR operator. Therefore, if the two signals entering the phase detector were of frequencies separated by $\Delta\omega$, the output was a periodic signal with frequency $\Delta\omega$. If this difference was too large, the PLL's frequency would oscillate and the PLL would not reach

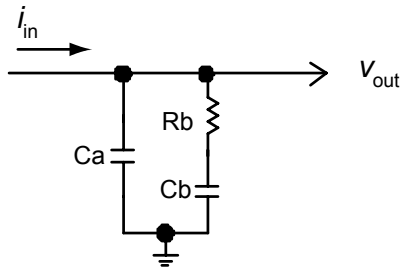

Figure 8.28: A current-to-voltage second-order loop filter

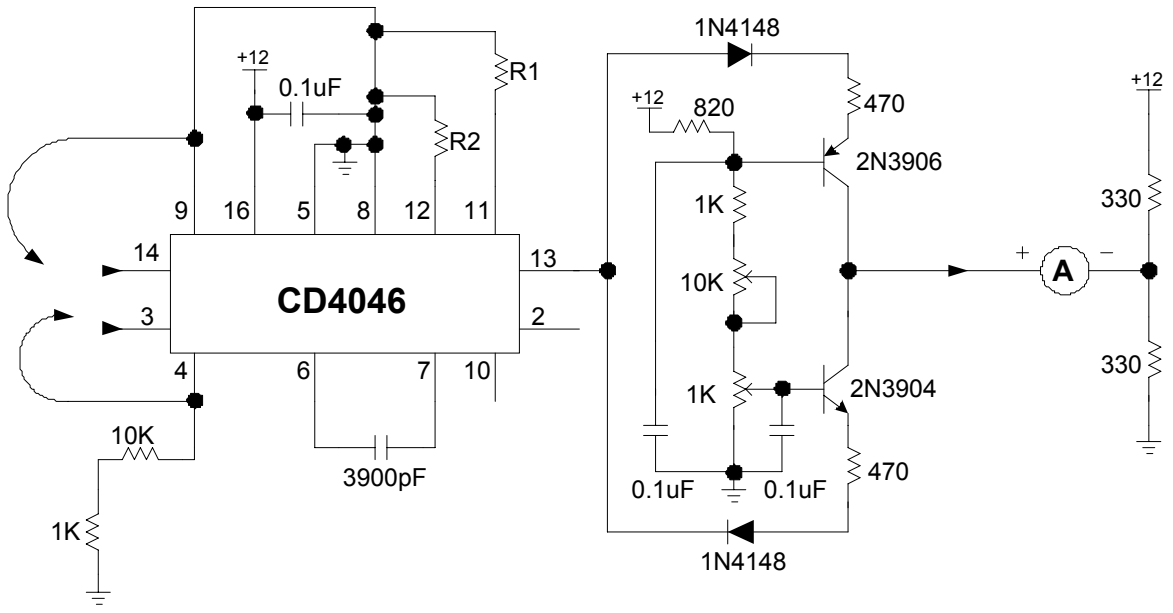

Figure 8.29: PLL with charge pump calibration schematic

lock. Phase detector II is called a phase/frequency detector because it produces an output with positive average value when there is a positive difference between the frequency of the modulated signal and of the PLL. This property improves the ability of the PLL to acquire lock. Another characteristic of phase detector II is that, when the PLL is locked and the center frequency error is zero, the signals are in-phase, rather than in quadrature.

In the CD4046, phase detector II produces pulses whose width is proportional to the phase error. Between the pulses, the output impedance of phase detector II is very high (open). If phase detector II was connected directly to a loop filter input, the resulting PLL system would be severely nonlinear. To ensure linearity, a charge pump conversion circuit must be built and added to the existing PLL circuit. The purpose of the first part of this lab is to build and calibrate the circuit.

The last circuit built in the basic PLL lab should still be assembled. Expand the circuit to build the circuit shown in Fig. 8.29. However, leave pin 13 of the CD4046 disconnected for now. That connection will be made after the transistor circuit is verified. Note that the connections from pin 9 to pin 14 and from pin 4 to pin 3 will change later on.

Double-check your circuit wiring, especially the bipolar transistor circuit. Pay careful attention to the fact that the 820Ω, $1k\Omega$, $10k\Omega$ pot, and $1k\Omega$ pot are in series. Also, double-

check the correct orientation of your transistors and diodes. If you are confident that the circuit is wired correctly, apply power (keeping pin 13 disconnected). The ammeter should read zero. If smoke came from your circuit, replace the bad parts and re-check your circuit. Next, connect the common node between the diodes to +12 volts. Adjust the $10k\Omega$ pot for a reading of $500\mu A$. If you cannot obtain the correct reading, check the top diode and the 2N3906 for proper installation. If $500\mu A$ was obtained, connect the common node between the diodes to ground. Adjust the $1k\Omega$ pot for a –$500\mu A$ reading. If you cannot obtain the correct reading, check the bottom diode and the 2N3904 for proper installation. Once you obtain +/- $500\mu A$ readings, disconnect power.

Connect pin 13 as shown on the diagram. Also connect pin 3 to ground, pin 14 to pin 4 (VCO output), and then re-apply power. Now, adjust the $10k\Omega$ pot for a $+600\mu A$ reading +/- 2%. Then, remove power, connect pin 14 to ground and pin 3 to pin 4, and re-apply power. Adjust the 1kΩ pot for a -$600\mu A$ reading. If successful, the charge pump circuit is now calibrated. The maximum output of the phase detector will be $+600\mu A$ for a $+\pi$ phase shift and a minimum output of -$600\mu A$ for a -π phase shift. Does this correlate with the k_{pll} value used in the pre-lab?

PLL with second-order filter

EQUIPMENT NEEDED (use a bench having the equipment, if available): oscilloscope, frequency counter, function generator, power supply, and a DMM.
EQUIPMENT TO BE CHECKED OUT: wire kit, two 10x probes, and a Wavetek function generator.
ADDITIONAL PARTS NEEDED: two capacitors and a resistor to be determined in the lab.

In this section of the lab, you will use the charge pump circuit (previously calibrated) in conjunction with the current to voltage loop filter shown in Fig. 8.28. Modify your circuit to look like Fig. 8.30. Notice that there are minor wiring changes in addition to the loop filter. In the pre-lab, the values of C_a, C_b, and R_b were computed. Using those values, pick the closest match from the list of choices below.

a) $C_a = 1000$pF, $C_b = 22000$pF, and $R_b = 39k\Omega$
b) $C_a = 390$pF, $C_b = 12000$pF, and $R_b = 27k\Omega$
c) $C_a = 470$pF, $C_b = 15000$pF, and $R_b = 18$k$k\Omega$
d) $C_a = 4700$pF, $C_b = 47000$pF, and $R_b = 5.1k\Omega$
e) $C_a = 1800$pF, $C_b = 6800$pF, and $R_b = 6.8k\Omega$
f) $C_a = 1200$pF, $C_b = 15000$pF, and $R_b = 10k\Omega$
g) $C_a = 2700$pF, $C_b = 18000$pF, and $R_b = 15k\Omega$

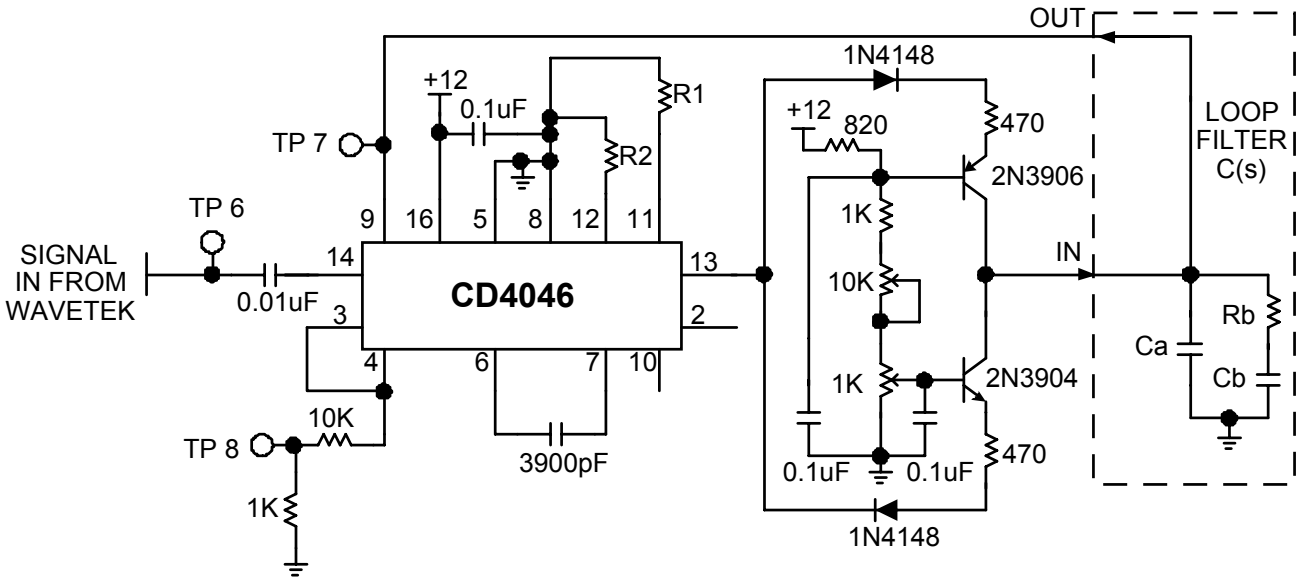

Figure 8.30: PLL with charge pump second-order filter

h) $C_a = 3300$pF, $C_b = 33000$pF, and $R_b = 12k\Omega$.

Verify your choice with the TA and insert the correct C_a, C_b, and R_b in your circuit. Apply power to the circuit and, using the Wavetek, input a 40kHz 8Vp-p sinusoid. Observe the signal input on TP 6 and the VCO signal on TP 8 with the scope triggered from TP 6 (connecting the frequency counter to TP 6 will help to speed-up frequency measurements). The input frequency may need slight adjustment in order to obtain phase lock. Measure and record the PLL hold range and capture range as described in the basic PLL lab. Verify your answers with the TA. Measure the ripple of the VCO control voltage (TP 7). How does this compare with the measurement of ripple in the basic PLL lab?

To observe the PLL step response, the frequency of the Wavetek will be modulated. Set the Wavetek center frequency (frequency knob and vernier) to 40kHz. Next, set up the bench-mounted function generator to modulate the Wavetek frequency. Set the frequency to 200Hz and a square wave output. With the output at minimum, connect it to the VGC input of the Wavetek (lower left connector). Also, using the BNC "T", observe the bench generator output on one channel of the scope while observing TP 7 on the other channel. Trigger from the bench generator (it may be possible to use external triggering on the oscilloscope and trigger from the "sync out" or "TTL out" of the bench generator). Slowly increase the bench generator output. You should notice that the control voltage of the VCO is also a square wave of same

frequency as the bench generator. Characterize the step response. What is the time taken to settle within ±10%? As you increase the output amplitude, how well does the PLL track (observe TP 7)?

Ball and Beam

Objectives

The objective of this lab is to gain experience in the design of control algorithms, taking the ball and beam system as an example. Using an animation program in Matlab, the challenges of controlling the system manually will be observed. Next, a proportional-derivative controller will be designed and evaluated.

Introduction

The ball and beam system is an educational experiment that is fun to watch and play with. A diagram is shown in Fig. 8.31. A beam is attached to a motor so that its angle θ with respect to the horizontal can be controlled at will. A ball is placed on the beam and is free to roll under the action of gravity (a small channel in the beam may keep the ball from rolling sideways). The distance of the ball from the center of the beam is denoted x. The ball can be placed at any location on the beam, and will stay there if its velocity v is zero and the beam angle is zero.

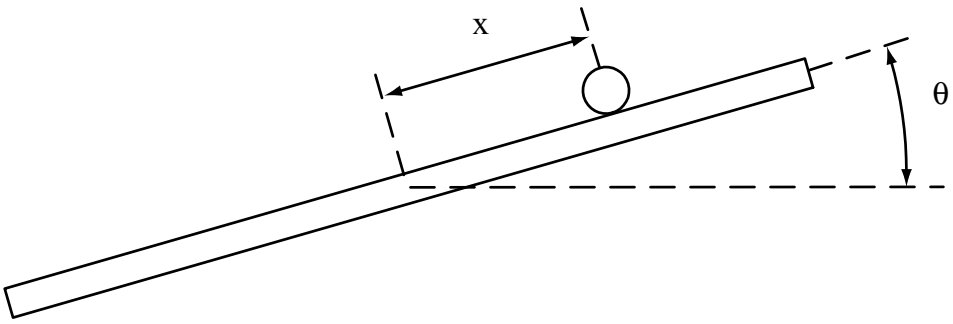

Figure 8.31: Ball and Beam System

Assuming that the only force acting on the ball is gravity, the movement of the ball is determined by Newton's law

$$m\frac{d^2x}{dt^2} = -mg\sin(\theta) \tag{8.33}$$

where g is the acceleration of gravity and m is the mass of the ball. A more careful analysis shows that the rotational inertia of the ball adds itself to the translational inertia, resulting

in a factor of 5/7 in the right-hand side of the equation of motion. The resulting state-space model of the ball and beam system is then

$$\begin{aligned} \frac{dx}{dt} &= v \\ \frac{dv}{dt} &= -\frac{5}{7} g \sin(\theta). \end{aligned} \tag{8.34}$$

where x is the position of the ball and v is its velocity. In order to apply linear control theory, one assumes that the angle θ is small and one replaces $\sin(\theta)$ by θ. The transfer function of the system is then

$$P(s) = \frac{X(s)}{\Theta(s)} = \frac{k}{s^2}, \qquad \text{where } k = -\frac{5}{7} g \tag{8.35}$$

The transfer function is the so-called *double integrator*, which is often encountered in control applications. Newton's law, $F = m.a$, generally yields this transfer function if force is the control variable and position is the output variable. Moving a spacecraft with thrusters is a practical example of such a system. Note that the model of the ball and beam system is only approximate. Also, a laboratory testbed may only permit to specify the torque of the motor, rather than its position. The model for such a system will have four states, instead of two. Nevertheless, the simplified model (8.35) exhibits the most interesting part of the dynamics of the ball and beam system, and is tractable for manual control.

Real-time simulation and visualization

The real-time simulation comes in the form of a Matlab macro (m-file). Visualization is provided through a Matlab figure, and a joystick is used for manual control. The file *bbeam.m* contains the m-file with the simulation of the system. A file called *jstick.dll* must be placed in the working directory of Matlab to provide readings of the joystick commands. It may be necessary to first calibrate the joystick by clicking on *Start/Settings/Control Panel/Game Controllers*. After calibration, typing *ref=jstick* in Matlab returns a Matlab variable *ref* with value between -1 (full left) and 1 (full right). *Please contact the instructor if you want to work on the labs at home, and have trouble using the joystick function.*

The model of the ball and beam system is implemented in *bbeam.m* using an Euler approximation

$$\begin{aligned} x(i) &= x(i-1) + dt \cdot v(i-1) \\ v(i) &= v(i-1) + dt \cdot (-\frac{5}{7} g \sin(\theta(i-1)) \end{aligned} \tag{8.36}$$

where i is the time instant and dt is the sampling period. The sampling period is set to 0.05s, or a frequency of 20Hz. Timing is obtained in Matlab using the *tic* and *toc* commands. The sampling period is close to the resolution of these commands, so that the joystick input and the visualization do not occur at exactly 20Hz. However, deviations are not normally noticeable.

Visualization is provided in a window using Matlab graphic functions. The window is scaled so that the unit lengths of the x and y axes are identical on the computer screen used to develop the labs. It may be necessary to adjust the window size to your own computer, and information is given in the code about the parameters that may need to be modified.

The simulation program gives the option of manual control or automatic control. Manual control is immediately available. For automatic control, two m-files must be written: a file called *bbeamc.m* containing the control algorithm, and an initialization file called *bbeamcinit.m.* The initialization file is called once before the simulation starts, while the control algorithm is called at the same rate as the ball and beam simulation. As provided, the program does not store the time histories of the signals. You will need to define arrays in the initialization macro, and store relevant variables in the control macro in order to plot the results of your experiments.

The control signal is the angle of the beam (called *theta* in the code), and is limited to ± 5 degrees. Measured variables are the position of the ball (called *xball* in the code) and the velocity of the ball (called *vball* in the code). The variable t in the code gives the time. In the simulation program, the position of the ball is limited to the length of the beam ($\pm 0.4m$) by setting the velocity to zero when the end of the beam is reached. Friction of the ball rolling on the beam is simulated by setting the velocity to zero if the ball velocity and the beam angle are small enough.

Manual control

The animation program is a good opportunity to have fun and get a sense for the challenges of controlling the double integrator. As an objective, try to move the ball from the left line to the right line as fast as possible. You will notice the importance of accounting for the ball velocity in your strategy. An automatic controller also needs to use this information. For the report, explain the challenges of the manual control problem and describe the strategies that you have developed for control.

Automatic control

The objective of this part of the lab is to develop an automatic controller, and to appreciate its performance compared to manual control. Choose a proportional-derivative controller, so that

$$\theta = k_p e + k_v \frac{de}{dt}, \qquad \text{with } e = x_{REF} - x \tag{8.37}$$

Assuming that both the ball position and the ball velocity can be measured, and neglecting dx_{REF}/dt, the control law may be implemented simply with

$$\theta = k_p e - k_v v \tag{8.38}$$

Note that the closed-loop system is a second-order system whose poles can be placed arbitrarily. Choose the parameters k_p and k_v so that the two closed-loop poles are real and equal, with an associated time constant of 0.3 seconds. Explain what would happen if the derivative gain k_v was set to 0.

Implement the control algorithm in the real-time simulation, and generate steps of position reference alternating between $0.3m$ and $-0.3m$ every 6 seconds. Plot the responses of x, v, and θ for 20 seconds. Repeat the experiment with a time constant of 0.2 seconds and explain the overshoot observed in the responses. Show your code and demonstrate the real-time operation to the TA.

Report at a glance

Be sure to include:

- Description of challenges and strategies for manual control.
- Description of PD controller and evaluation. Plots of responses with two controller settings.
- Observations and comments.
- Listing of *bbeamc.m* and *bbeamcinit.m* files.

Inverted Pendulum

Objectives

The objective of this lab is to experiment with the stabilization of an unstable system. The inverted pendulum problem is taken as an example and the animation program gives a feel for the challenges of manual control. A stabilizing linear controller is designed using root-locus techniques, and the controller is refined to enable stabilization of the inverted pendulum at any position on the track.

Introduction

The inverted pendulum system is a favorite experiment in control system labs. The highly unstable nature of the plant enables an impressive demonstration of the capabilities of feedback systems. The inverted pendulum is also considered a simplified representation of rockets flying into space.

Fig. 8.32 shows a diagram of the experiment. A cart rolls along a track, with its position x being controlled by a motor. A beam is attached to the cart so that it rotates freely at the point of contact with the cart. The angle of the beam with the vertical is denoted θ, and an objective is to keep the angle close to zero. Since the pendulum may be stabilized at any position on the track, a second objective is to specify the track position. However, recovery from non-zero beam angles may require significant movements of the cart along the track, and stabilization may be impossible if an insufficient range of motion remains.

A model of the inverted pendulum may be derived using standard techniques. A careful derivation (H. Khalil, *Nonlinear Systems,* 2nd edition, Prentice Hall, Upper Saddle River, NJ, 1996) shows that

$$(I + mL^2)\frac{d^2\theta}{dt^2} + mL\cos(\theta)\frac{d^2x}{dt^2} = mgL\sin(\theta) \tag{8.39}$$

where m is the mass of the beam, $2L$ is the length of the beam, $I = mL^2/3$ is the moment of inertia of the beam around its center of gravity, and g is the acceleration of gravity. For small angles θ, it follows that

$$\frac{4}{3}L\frac{d^2\theta}{dt^2} - g\theta = -\frac{d^2x}{dt^2} \tag{8.40}$$

and, the transfer function of the system is

$$\frac{\Theta(s)}{X(s)} = \frac{-s^2}{(4/3)Ls^2 - g} \tag{8.41}$$

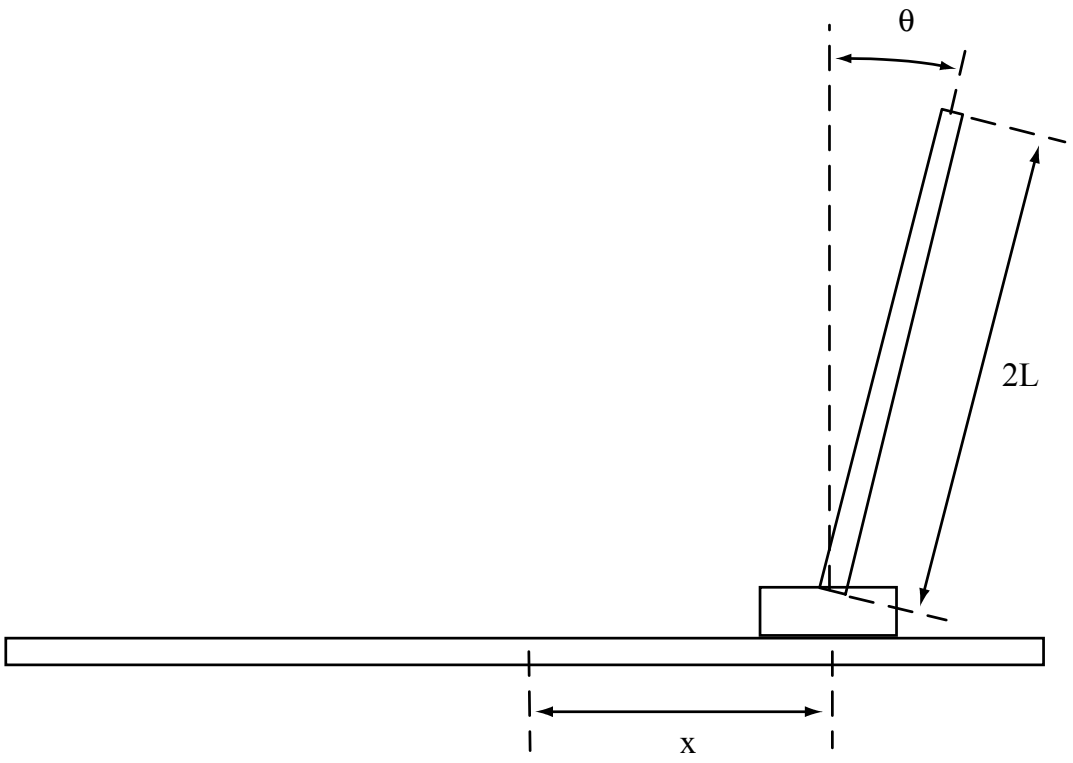

Figure 8.32: Inverted Pendulum System

The system has poles at $s = \pm\sqrt{3g/4L}$. The positive root is unstable, and instability worsens when the beam is short (as one might expect). Another tricky problem is that the system has two zeros at $s = 0$. This is due to the fact that an *acceleration* of the cart is required to impact the beam angle. On the other hand, this property makes stabilization possible for arbitrary cart positions.

More complex models assume that the control variable is the force applied to the cart, rather than its position. This assumption is more realistic in many cases, but makes manual control very difficult. Instead, the simulation program of this lab represents the delay in the motion of the cart by the response of a first-order system

$$X(s) = \frac{f}{s+f}\ X_{com}(s) \tag{8.42}$$

where x_{com} is the commanded cart position, x is the cart position, and $f > 0$. The overall transfer function of the system is then

$$\frac{\Theta(s)}{X_{com}(s)} = \frac{-fs^2}{((4/3)Ls^2 - g)(s+f)} \tag{8.43}$$

or

$$\frac{\Theta(s)}{X_{com}(s)} = \frac{bs^2}{(s+a)(s-a)(s+f)} \tag{8.44}$$

where $a = \sqrt{3g/4L}$, $b = -3f/4L$.

Manual control

The simulation program resides in the file *invpend.m*. The program assumes that $L = 1$ m, and that x is limited to $\pm 2L$. The beam angle is prevented from exceeding $\pm 30°$. The value of f is 5, so that the cart responds relatively fast to commands. Even with rapid position control of the cart, however, the system is difficult to control with a joystick. The program has a "*cheat*" variable that allows you to make the problem easier. The variable reduces the magnitude of the gravity constant, and the choice $cheat = 6$ is preset in the program (it 's as if you controlled the pendulum on the moon!).

Have fun trying to keep the beam balanced. If you feel comfortable with it, try to move the cart from the left line to the right line, while keeping the beam balanced. Describe in words the challenges of the control problem and your strategies for control. How close to 1 can you let the *cheat* variable be? For how long? You may be asked to demonstrate!

Automatic control

Stabilizing controller

First, you will design a controller to bring the beam angle to zero at the zero track location. Try to find a stabilizing controller using root-locus techniques, and noting that the plant transfer function is

$$P(s) = \frac{\Theta(s)}{X_{com}(s)} = \frac{bs^2}{(s+a)(s-a)(s+f)} \tag{8.45}$$

Explain why it is necessary to use an *unstable* controller, in order to have a chance at stabilizing the system.

Then, consider the second-order controller (with zero reference input)

$$C(s) = \frac{X_{com}(s)}{-\Theta(s)} = k\frac{(s+a)(s+f)}{(s-a)(s+c)} \tag{8.46}$$

Note that, with the cancellations, the closed-loop system has three poles. The controller has two parameters k and c to be adjusted. Show that it is possible to choose the parameters so that all three closed-loop poles are placed at $s = -a/2$. Compute the parameters required for

the system and draw the root-locus for varying k. Verify that the closed-loop system is stable by use of the Nyquist criterion (function *nyquist* in Matlab). Plot the Bode plots and compute the gain and phase margins of the system (function *margin* in Matlab). Indicate how much the gain of the system can be increased or decreased while preserving stability.

The controller must be implemented in discrete-time, assuming a sampling rate of 20Hz. Check the help files for the Matlab functions *tf*, *c2d*, and *ssdata* to get information on how to do it (you may also look in the simulation code how the transfer function (8.41) was implemented). Use the option 'zoh' for the discretization. Implement the controller transfer function in a file *invpendc.m*, with its initialization in a file *invpendcinit.m*. Test the response of the system by setting the initial value of the variable *theta* at 5 degrees (the initial value can be set at the beginning of the simulation program). Plot the responses of the beam angle and of the cart position resulting from this initial error. Check that the beam angle returns to zero within the expected time period and with the expected dynamics, given the pole locations.

Tracking controller

Modify the controller so that an arbitrary set-point of the cart position can be imposed. Replace $X_{com}(s) = -C(s)\Theta(s)$ by

$$X_{com}(s) = X_{ref}(s) - C(s)\Theta(s) \tag{8.47}$$

Note that this is an unusual configuration for a feedback system, because the reference command is applied to the plant input rather than the controller input. Interestingly, the plant output *cannot* be set to an arbitrary position in this problem, but the plant input can. Generate steps of position reference alternating between $1m$ and $-1m$ every 20 seconds, and plot the responses of x, x_{com}, and θ for 60 seconds. Let the initial beam angle be zero.

You should observe an overshoot in the response of the cart. Compute the transfer function from $X_{ref}(s)$ to $X(s)$ and observe that there is a zero at a frequency lower than the closed-loop poles. This problem may be addressed by using a pre-filter $C_F(s)$ so that

$$X_{com}(s) = C_F(s)X_{ref}(s) - C(s)\Theta(s) \tag{8.48}$$

where $C_F(s)$ is a first-order filter with unity DC gain and a pole located at the same value as the low-frequency zero. Plot the responses of x, x_{com}, and θ for 60 seconds, with steps of position reference alternating between $1m$ and $-1m$ every 20 seconds. Explain the fact that the variable x_{com} initially moves in the direction *opposite* to the step applied. Can this effect be eliminated by prefiltering of the reference command (as was done for the overshoot)? Test the controller for the *cheat* variable equal to 6, and again for the variable equal to 1. Show your code and demonstrate the real-time operation to the TA.

Report at a glance

Be sure to include:

- Description of challenges and strategies for manual control.
- Design and evaluation of the stabilizing controller with the variable *cheat=6*. Include controller parameters, root-locus, Nyquist criterion, gain and phase margins, and responses to initial beam angle.
- Design and evaluation of the controller for set-point tracking with the variable *cheat=6*. Plots of responses of x, x_{com}, and θ to step changes in set-point of the cart position.
- Design and evaluation of the controller for set-point tracking with the variable *cheat=1*.
- Observations and comments.
- Listing of *invpendc.m* and *invpendcinit.m*.

Flexible Beam

Objectives

The objective of this lab is to learn about the challenges posed by resonances in feedback systems. An intuitive understanding will be gained through the manual control of a flexible beam resembling a large space robotic arm. Control design will be performed in the frequency domain using a lead controller. A notch filter will be incorporated in the feedback loop in order to reduce the excitation of resonances.

Introduction

Systems with lightly-damped, complex poles (resonances), are encountered in many applications. An example is a large robotic arm in space, whose transversal dimensions are made small to reduce weight. The arm will bend and oscillate if moved rapidly. In a computer disk drive, a read/write head is attached to the end of a small, rigid structure which is rotated rapidly to access various tracks. When the head is positioned within fractions of microns, even such a rigid structure behaves like a flexible structure.

The diagram of the flexible beam is shown on Fig. 8.33. The angle of the beam at the shaft is denoted θ, while ϕ is the angle at the tip. If there was no flexibility, the two angles would be equal. Experimental data was collected on a flexible beam of length $0.4m$. Fig. 8.34 shows the frequency response that was measured from the motor current to the angular acceleration of the shaft, while Fig. 8.35 shows the response measured from the motor current to the angular acceleration of the tip. The acceleration at the shaft was obtained by measuring the position with an encoder (and multiplying the frequency response by $-\omega^2$ to obtain the acceleration), while the acceleration at the tip was obtained with an accelerometer. The plots show the experimental data, as well as approximate fits obtained with fourth-order models (as dashed lines).

The approximate models shown on the plots are given by

$$\begin{aligned} \frac{s^2\Theta(s)}{I(s)} &= \frac{k_\theta(s-z_1)(s-z_1^*)(s-z_2)(s-z_2^*)}{(s-p_1)(s-p_1^*)(s-p_2)(s-p_2^*)} \\ \frac{s^2\Phi(s)}{I(s)} &= \frac{k_\phi(s-z_3)(s-z_4)(s-z_5)(s-z_6)}{(s-p_1)(s-p_1^*)(s-p_2)(s-p_2^*)} \end{aligned} \tag{8.49}$$

where $p_1 = -3 + 74j$, $p_2 = -3 + 215j$, $z_1 = -0.07 + 18j$, $z_2 = -0.07 + 180j$, $z_3 = 100$, $z_4 = -120$, $z_5 = 200$, and $z_6 = -300$ (note that the poles are very lightly damped). The input

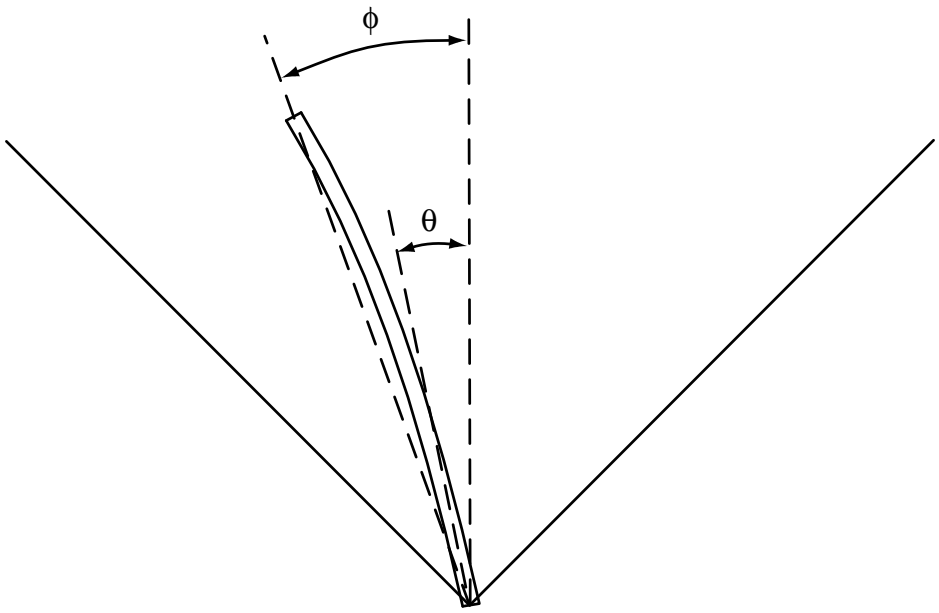

Figure 8.33: Flexible Beam

variable is the current in the motor, measured in A, and the angles θ and ϕ are measured in radians. The constants k_θ and k_ϕ are such that the DC gains of the transfer functions are equal, with

$$k_p = \left(\frac{s^2\Theta(s)}{I(s)}\right)_{s=0} = \left(\frac{s^2\Phi(s)}{I(s)}\right)_{s=0} = 5.5 \tag{8.50}$$

The equality for θ and ϕ follows from the fact that there is no bending of the beam near zero frequency. For low frequencies, the transfer functions are therefore approximately given by

$$\frac{\Theta(s)}{I(s)} \simeq \frac{\Phi(s)}{I(s)} \simeq \frac{k_p}{s^2} \tag{8.51}$$

This approximation of the system is the double integrator encountered with the ball and beam. The feedback design is more difficult than for the ball & beam, because of additional poles close to the $j\omega$-axis, and because of zeros close to the $j\omega$-axis and in the right half-plane.

Manual control

The simulation file is called *flex.m*. You should play with the simulation and move the beam from the 45° line to the −45° line. You will encounter two difficulties: the $1/s^2$ behavior and the flexibility of the beam. Resonances can be avoided by moving the beam slowly, but

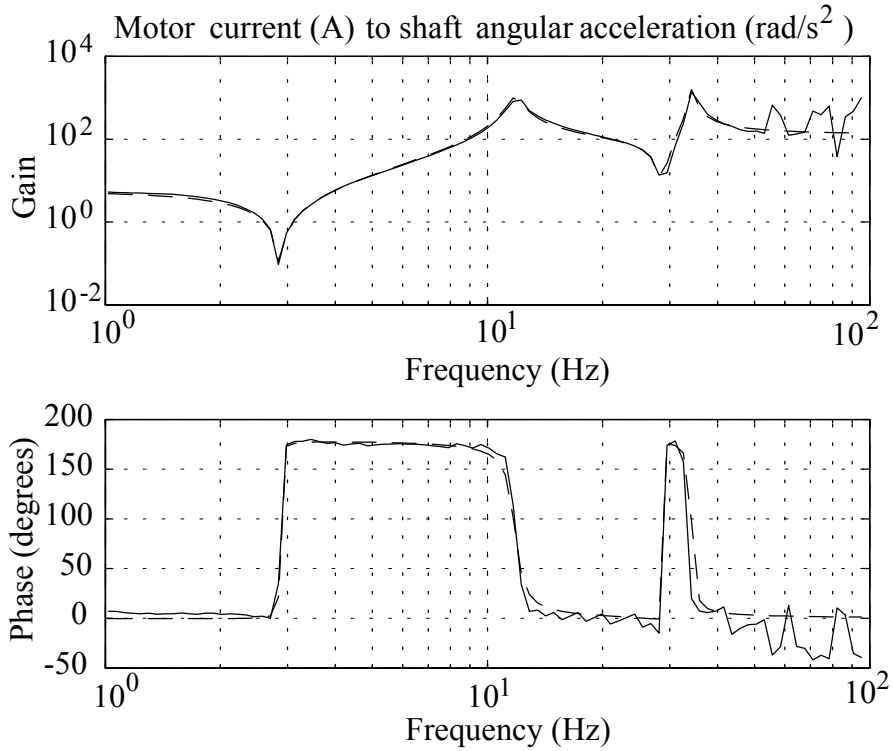

Figure 8.34: Frequency response of the flexible beam from motor current to shaft angular acceleration

performance will be unimpressive. It can be enlightening (and fun) to "excite" the resonances by applying commands in the same frequency range as the flexible modes. The beam will bend to large angles. Once this mechanism is understood, you may return to the task of rapidly moving the beam from side to side without exciting such resonances.

Note that the simulation was implemented differently from the previous ones. The continuous-time model was discretized assuming a sampling period of $200Hz$. Since the program runs at a rate of approximately $20Hz$, the visualization slows the dynamics by a factor of 10. This result is helpful, because the dynamics of the actual system are too fast to be controlled manually.

Lead controller design

The objective is to design a *lead controller*

$$C(s) = \frac{I(s)}{\Phi_{ref}(s) - \Phi(s)} = k_c \frac{(s+b)}{(s+a)} \tag{8.52}$$

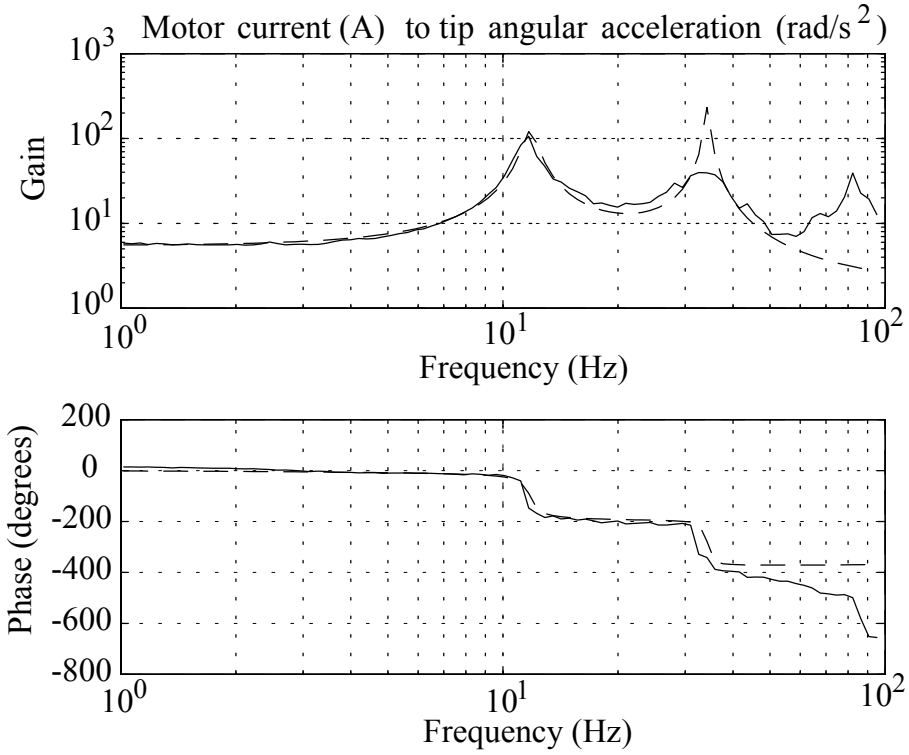

Figure 8.35: Frequency response of the flexible beam from motor current to tip angular acceleration

Such a controller was designed for a phase-locked loop in the advanced PLL lab. Here, the design will be performed in the frequency domain. The motor current i is the control signal, and the tip angle ϕ is the output to be regulated, with a reference value ϕ_{ref}.

The Bode plots of the lead controller are shown in Fig. 8.36. In the notes for the course, the variables are shown tosatisfy the following constraints

$$\begin{aligned} \omega_p &= \sqrt{ab} \\ m_p &= k_c\sqrt{\frac{b}{a}} \\ \frac{a}{b} &= \frac{1+\sin(\phi_p)}{1-\sin(\phi_p)} \end{aligned} \tag{8.53}$$

To determine the controller parameters, we consider an approximate model of the plant with the two poles at the origin and the first two resonant modes. The tranfer function from

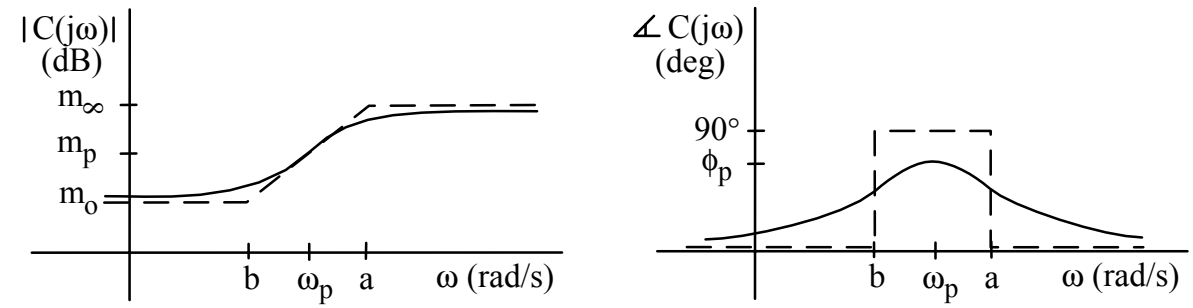

Figure 8.36: Bode plots of lead controller

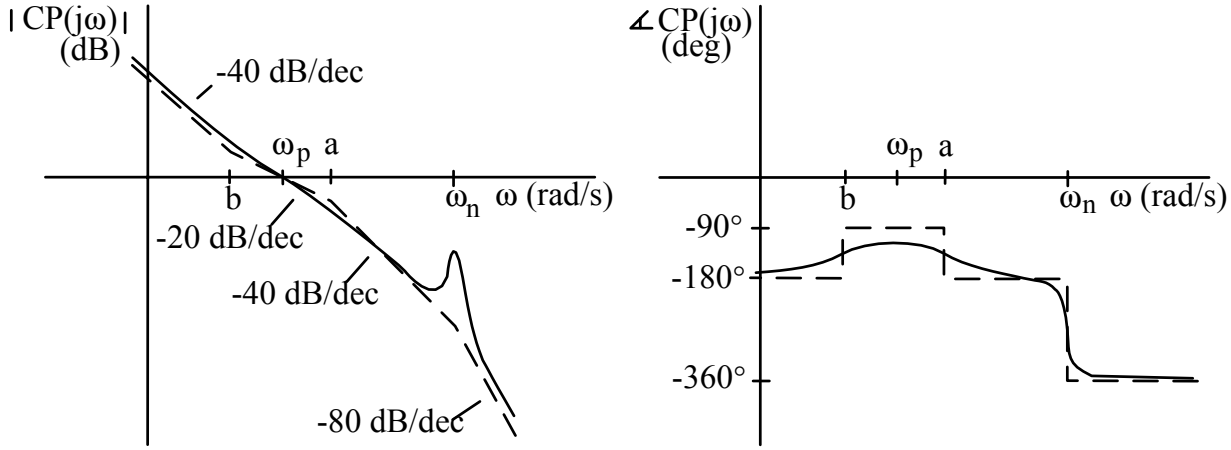

Figure 8.37: Bode plots of plant and lead controller (only one resonant mode shown)

i to ϕ is then

$$P(s) = \frac{\Phi(s)}{I(s)} = \frac{k_p}{s^2} \cdot \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{8.54}$$

where $k_p = 5.5$, $\omega_n = |p_1|$, $\zeta = -\operatorname{Re}(p_1)/|p_1|$, and $p_1 = -3 \pm j73$. Fig. 8.37 shows the Bode plots of the loop transfer function for the approximate plant and the lead compensator. The magnitude plot shows ω_p as the crossover frequency, a condition that is to be enforced by proper choice of the parameters. Assuming that $\omega_n \gg \omega_p$, show using (8.53) that

$$\begin{aligned} |C(j\omega_p)P(j\omega_p)| &\simeq \frac{k_c k_p}{\omega_p^2}\sqrt{\frac{b}{a}} \\ |C(j\omega_n)P(j\omega_n)| &= \frac{k_c k_p}{2\zeta\omega_n^2} \end{aligned} \tag{8.55}$$

Then, find the values of the controller parameters k_c, a, and b such that

$$\phi_p = 60^\circ, \quad |C(j\omega_p)P(j\omega_p)| = 1, \quad |C(j\omega_n)P(j\omega_n)| = 0.1 \tag{8.56}$$

The idea is that the first two conditions will ensure a crossover frequency at ω_p and a phase margin of 60°. The third condition will yield a gain margin of 10, given that the phase crossover frequency will be close to ω_n.

Using the parameters of the lead controller, plot the step response of the closed-loop system (function *step* in Matlab) and the Bode plots with the gain and phase margins (function *margin* in Matlab). *Make sure to use the complete plant transfer function for this step, not the approximate one.* The feedback system may be assembled using the functions *series* and *feedback* in Matlab, if desired. Also compute the locations of the closed-loop poles (function *pole* or *roots* in Matlab).

Automatic control

Design #1

Implement the controller in the simulation with files *flexc.m* and *flexcinit.m*. A discrete-time equivalent of the control system should be computed, as was done in the previous lab. Discretization should be based on the 200Hz sampling frequency, so that visualization will show the system at a rate slowed down by a factor of 10. Plot the responses of i, θ, ϕ, and $\phi - \theta$ to step inputs. Let ϕ_{ref} switch between 45° and −45° every 2 seconds, and record the data for 6 seconds (which will take 60 seconds in the real-time simulation). The visualization should show a slow response, with a large overshoot, due to the low frequency zero at $s = -b$. If you reduce the gain margin by increasing $|C(j\omega_n)P(j\omega_n)|$, you will find that the crossover frequency increases and that the response speeds up, but oscillations are observed due to excitation of the beam's flexible modes.

Design #2

Improve the design by cascading the lead/lag controller with a notch filter and by prefiltering the reference input, so that

$$I(s) = C(s)C_{notch}(s)\left(C_F(s)\Phi_{ref}(s) - \Phi(s)\right) \tag{8.57}$$

where $C(s)$ is the lead controller considered earlier. Let the prefilter be

$$C_F(s) = \frac{1.2b}{s + 1.2b} \tag{8.58}$$

where b is the zero of the compensator. Let the notch filter be

$$C_{notch}(s) = \frac{s^2 + \omega_n^2}{s^2 + 2\omega_n s + \omega_n^2} \tag{8.59}$$

where ω_n is the natural frequency of the first resonant mode. The prefilter will eliminate the overshoot, and the notch filter will allow you to increase the crossover frequency and, as a result, the speed of response.

Re-design the lead controller with $\omega_p = 12$ rad/s, keeping the condition that $\phi_p = 60°$ and the crossover frequency condition

$$\frac{k_c k_p}{\omega_p^2}\sqrt{\frac{b}{a}} = 1 \tag{8.60}$$

Check the properties of this re-designed controller in combination with the prefilter, notch filter, and plant transfer function: plot the step response, the Bode plots with gain and phase margins, and compute the values of the closed-loop poles. Implement the controller in the simulation, and plot the responses of i, θ, ϕ, and $\phi - \theta$ to step inputs in ϕ_{ref}. Compare the responses to those of the previous controller. Show your code and demonstrate the real-time operation to the TA.

Report at a glance

Be sure to include:

- Description of challenges and strategies for manual control.
- Design and evaluation of the lead controller, with plots of the step response, Bode plots with gain and phase margins, and values of the closed-loop poles. Responses of i, θ, ϕ, and $\phi - \theta$ to step inputs of ϕ_{ref} in the simulation.
- Design and evaluation of the lead controller with notch filter and prefilter, with plots of the step response, Bode plots with gain and phase margins, and values of closed-loop poles. Responses of i, θ, ϕ, and $\phi - \theta$ to step inputs of ϕ_{ref} in the simulation.
- Observations and comments.
- Listing of *flexc.m* and *flexcinit.m*.