For the following sections this is a two-week lab.
Tue 7:30 am
Wed 11:50 am
Thur 7:30am
Thur 3:05 pm

For the Monday section this is a one-week lab, Monday 2/26/07. Make sure you have all the prelab finished and are ready to perform the lab. You may wish to visit one of the other labs in the prior week to make sure you are ready.

ECE3510 Lab #~~3~~ 5

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

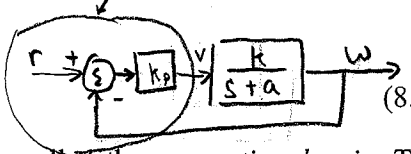
In the lab on first-order systems, the response of a brush DC motor with the voltage v (V or volts) considered as an input and the angular velocity ω (rad/s or s^{-1}) considered as an output was found to be approximately described by a model

$$P(s) = \frac{\omega(s)}{v(s)} = \frac{k}{s+a}. \qquad (8.1)$$

"control law"

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

$$v = k_P(r - \omega), \qquad (8.2)$$



proportional gain

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

$$H(s) = P_{CL}(s) = \frac{\omega(s)}{r(s)} = \frac{kk_P}{s+(a+kk_P)} \qquad (8.3)$$

$\frac{\frac{k_p k}{s+a}}{1+\frac{k_p k}{s+a}} = \frac{k_p k}{(s+a)+k_p k}$

$k \approx 1000$
$a \approx 100$

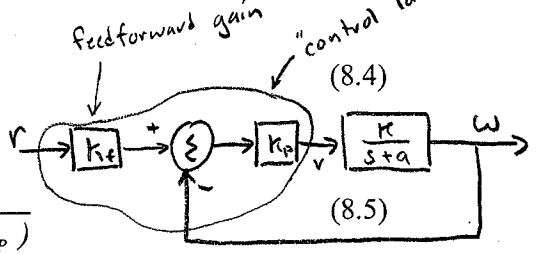
Note that the closed-loop pole is given by $-a - kk_P$. In theory, it would appear that the closed-loop pole could be moved arbitrarily far in the left-half plane through the use of a sufficiently large proportional gain. The response of the system could be made arbitrarily fast in that manner. As this 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.

↑ see the results of our lab 4

Proportional-integral control for velocity tracking

The DC gain in (8.3) 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 (8.2) by

$$v = k_p(k_F r - \omega), \tag{8.4}$$



yields a closed-loop transfer function

$$P_{CL}(s) = \frac{\omega(s)}{r(s)} = \frac{k_F k k_P}{s + (a + kk_P)} \tag{8.5}$$

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

$$k_F = \frac{(a + kk_P)}{kk_P} \tag{8.6}$$

We will call k_F the *feedforward* gain.

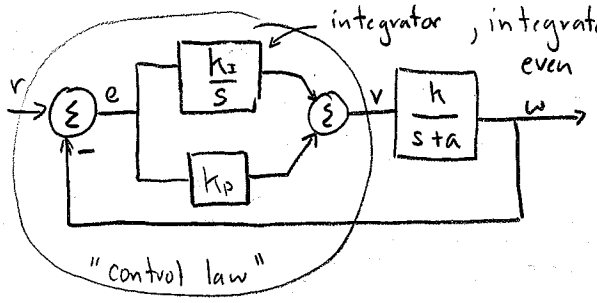
Despite the capability of adjusting the feedforward gain k_F in order to obtain a DC gain of 1, perfect tracking of reference inputs is usually not achieved because the parameters of the system are not exactly known or may vary, and because disturbances may affect the response of the system. These problems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

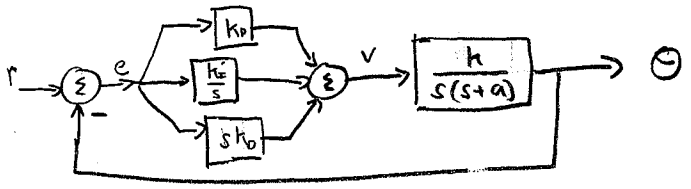
$$v = k_p(r - \omega) + k_I \int (r - \omega) dt, \tag{8.7}$$

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

$$H(s) = P_{CL}(s) = \frac{\omega(s)}{r(s)} = \frac{kk_P(s + k_I / k_P)}{s^2 + (a + kk_P)s + kk_I} \tag{8.8}$$

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

$$V(s) = k_p\left(r(s) - \theta_{(s)}\right) + \frac{k_i}{s}\left(r(s) - \theta(s)\right) + sk_D\left(r(s) + \theta(s)\right)$$

$$v = k_P(r-\theta) + k_I\int(r-\theta)dt + k_D\frac{d}{dt}(r-\theta). \tag{8.9}$$

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. & speed of initial response.

In practice, the control law (8.9) 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

$$V(s) = k_p k_f\, r(s) - k_p\theta_{(s)} + \frac{k_i}{s}r(s) - \frac{k_i}{s}\theta_{(s)} - sk_D\theta(s)$$

$$v = k_p(k_F r - \theta) + k_I\int(r-\theta)\,\mathrm{d}t - k_D\frac{d}{dt}\theta. \tag{8.10}$$

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

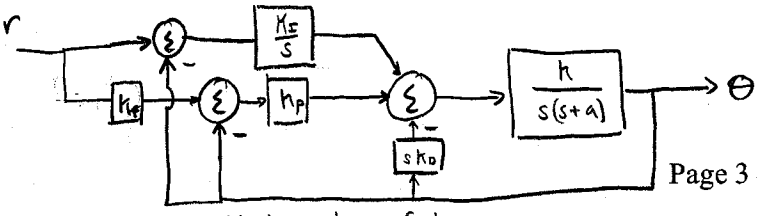
$$P(s) = \frac{\theta(s)}{v(s)} = \frac{k}{s(s+a)}, \tag{8.11}$$

and the PID control law (8.10), is given by

$$P_{CL}(s) = \frac{\theta(s)}{r(s)} = \frac{kk_Fk_P(s + \frac{k_I}{k_Fk_P})}{s^3 + (a + kk_D)s^2 + kk_Ps + kk_I} \tag{8.12}$$

Note that the closed-loop transfer function (8.12) has three poles. There is also a zero at $-k_I/(k_Fk_P)$. 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.

see p. 35 in text.



Not helpful

Pre-lab

1 • Derive equation (8.5) 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.

2 • Derive equation (~~8.23~~ 8.8) 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.

3 • Derive the transfer function given in (~~8.27~~ 8.12) 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:

1. Brush DC motor
2. Dual power amplifier
3. ~~Voltmeter~~
4. ~~Cable rack~~ (wire kit)
5. DSpace kit ~~which includes an encoder cable and I/O breakout box.~~

Preliminary testing

Carry out the same testing procedure as in the lab on first-order systems.

Lab overview

For this experiment control of the velocity of the motor will be explored using Proportional and Proportional-Integral (PI) control. Control of the motor position will be explored using Proportional-Integral-Differential (PID) control.

The dSpace ~~experiment~~ control screen designed to ~~perform~~ work with this lab is setup to allow a choice between the different control types. Values for each of the ~~proportional~~ 3 gains can ~~also~~ be entered into the Layout. A reference value of either speed or position is available to be used with the control law type.

In the case of the PID controller, you will implement the equation for the control law by modifying the underlying C/C++ code and rebuilding the software that will be provided to the DSpace board.

www.ece.utah.edu/courses/ece3510/PID_Control_DC_Motor.zip

Proportional Control

Load the experiment *PID_Control_DC_Motor.cdx*, It should open ~~with~~ in the *proportional* control ~~type selected~~ mode. The reference input for the proportional control is a reference speed, given in RPM. *It should be noted that any gains not used for the selected control law are ignored.*

First, experiment with proportional control by setting 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, capture the data and plot the results. Discuss what happens when the gain k_P becomes large to the output. specifically discuss steady-state error and overshoot or ringing. b = 2a, b = 6a, b = 11a

Proportional-Integral Control

Change the control type to *PI* and apply the values calculated for the PI control law, (prelab 2). Set ~~Setting~~ $k_F = 1$ in the experiment. ~~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, & record the k_P and k_I used

Working with Dspace software Design

The programs used by the dSpace board to implement experiments are created using C/C++ code and a compiler/linker provided with the system. For each of the labs the code for an experiment is available in *main.c* and *driver.c*. All of the control algorithms and the main loop are found in the *main.c* file, which will be referred to as the *lab code*. Any code that supports the hardware interface to the system is found in *driver.c* file, which will be called the *driver code*.

Looking at the lab code it can be broken down in to three sections. The first section contains global variables that are used for connection to the buttons, plot windows and displays in the layout window. The next section is the *main()* function that is processed when the board is loaded. Last is a function called *user_isr()*, this function contains the logic and equations that implement the control laws for the experiments.

For an embedded system, like dSpace, most of the code that does the work is not located in the *main()* function. The main includes initializations for the system that sets up the framework

of hardware interrupts or ISRs. An ISR is a signal that tells the system that data is ready to be sent/received or that a certain function should be performed.

In the case of the lab experiments, a built-in timer provides an ISR every 2 msec to perform the operations of getting the current position for the incremental encoder (INC1) and setting the output value of ADCH1. The function *isr_srt()* in the driver code performs the commands to read and write the encoder and ADC respectively. *isr_srt()* also runs the function from the lab code *user_isr()* which performs the control processing. The system then sits and waits for the next ISR.

The lab code and driver code must be compiled/linked to be used by the system. This requires an MSDOS prompt window. A shortcut to MSDOS is provided in the directory with the lab experiments. From the DOS prompt the program *down1104.exe* can be used to compile/link and load the program for the system. The syntax for this command is:

> down1104 filename1.c fileaname2.c ...

All the .c files created to implement the experiment software must be listed as arguments to *down1104.exe*. In turn, they are used to generate a .ppc file that contains the machine code used on the board. Because of how the experiments are set up, the first filename argument should be main.c followed by driver.c. With a valid execute of *down1104* the board should be loaded and ready to perform experiments.

Proportional-Integral-Derivative (PID) Control

Adjust the file main.c in order to implement the PID control law for position. All of the required variables and the location of the control law are outlines inside the *user_isr()* function of the .c file.

As a hint, you should remember the relations between position and velocity. Both position and velocity are provided with radian measure by the encoder. For the case of integration in a discrete system, think of it as a Riemann Sum. Look at the control law for the PI controller to get an idea of how integration is performed in that case.

Once the experiment is compiled, linked and downloaded, you may apply the calculated values of the PID parameters, with $k_F = 1$, and a reference input step of 90 degrees. The settling time should be approximately 100ms, with an underdamped 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 that gives the best response (minimum settling time with negligible overshoot).

DEMONSTRATE YOUR FINAL EXPERIMENT TO THE TA

~~**Report at a glance**~~ Your notebook should include:

~~*For this lab a written formal report will be required.*~~

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. (check-off)
- Comments. & conclusions