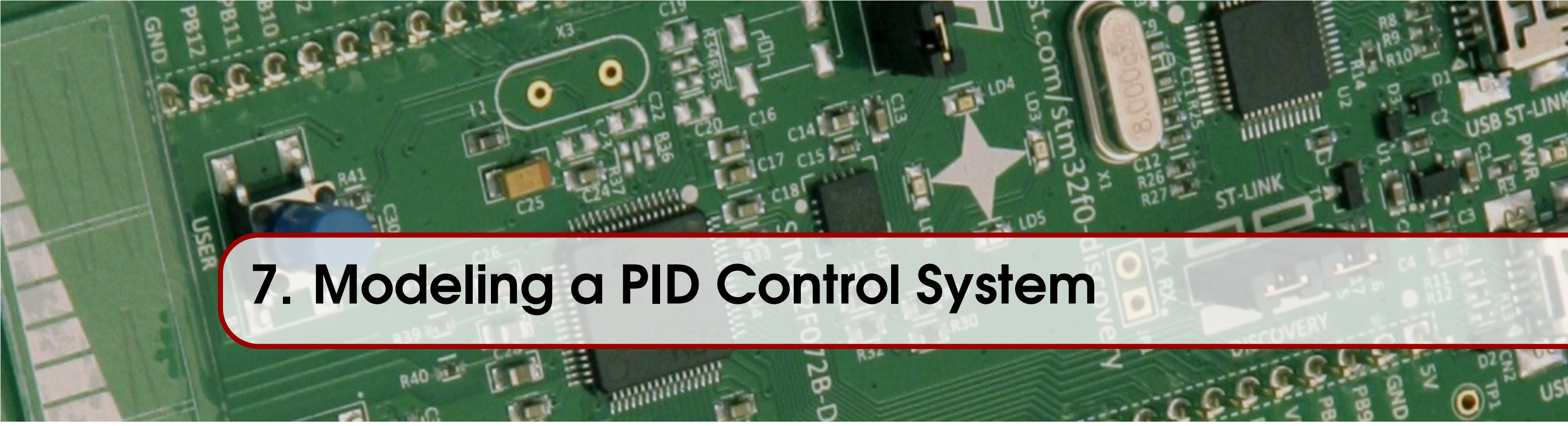

7. Modeling a PID Control System

7.1 Control System Basics

The purpose of a control system is to direct the behavior of other devices to produce an output state that matches a requested condition. Control systems vary widely in their design and are used in many different applications such as a thermostat for household heating and cruise-control for vehicles.

Control systems can be either *open-loop* or *closed-loop* in design. Open-loop systems apply a process or algorithm to directly generate their output state from their inputs; they have no method of measuring the actual effect of their actions. Closed-loop control systems use their own output as a secondary input, and calculate a course of action depending on the error between the desired and current state. This process is called *feedback*.

Figure 7.1 demonstrates the four basic parts of a closed-loop control system.

Input/Setpoint: The input to a control system represents some parameter of the desired output. This *setpoint* is compared against the current output state.

Plant/Process: The *plant* of a system represents the external device under the command of the control system. The plant generates the observable output.

Feedback: In a closed-loop system the output is compared against the input and the system's error is calculated. This error is often either the combination of the output and input (positive feedback) or the difference between them (negative feedback).

Control: The control system adjusts its output (input to the plant) in order to minimize the system error.

7.1.1 PID Control Systems

The acronym PID represents the three mathematical relations used within the control system; and stand for proportional, integral, and derivative. PID controllers are commonly used in industrial systems because they offer rapid error correction, good stability and can be tuned to react properly to the unique characteristics of a specific system.

The overall equation of a PID control system can be modeled by the following equation, where $c(t)$ represents the output control, and $e(t)$ the input error.

$$c(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

The outputs of each portion (proportional, integral, derivative) are combined to form the control signal fed to the system plant. The PID controller is tuned by adjusting scaling coefficients for each of these. Each of the three portions can be used in isolation to make simpler control systems. The following sections describe the effect each of the mathematical relationships has on the control signal depending on the input error.

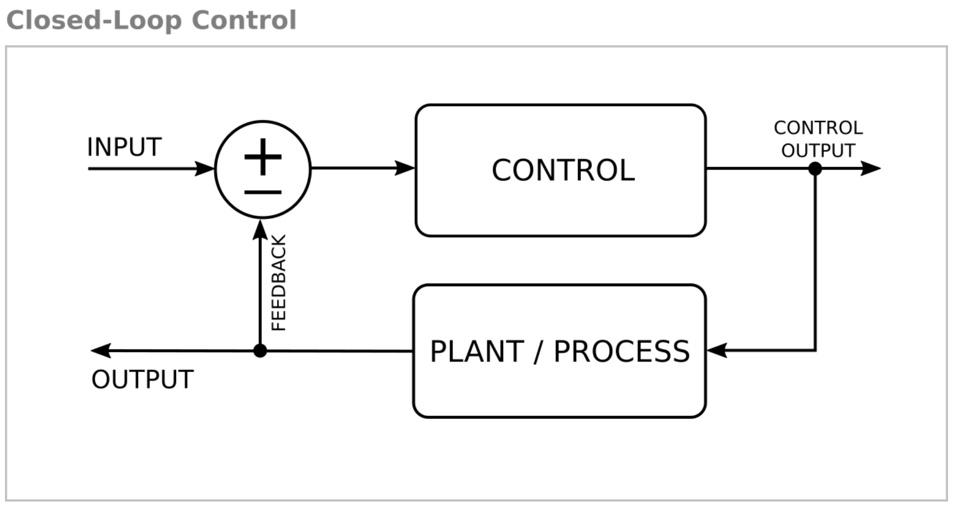

Figure 7.1: Basic closed-loop control system.

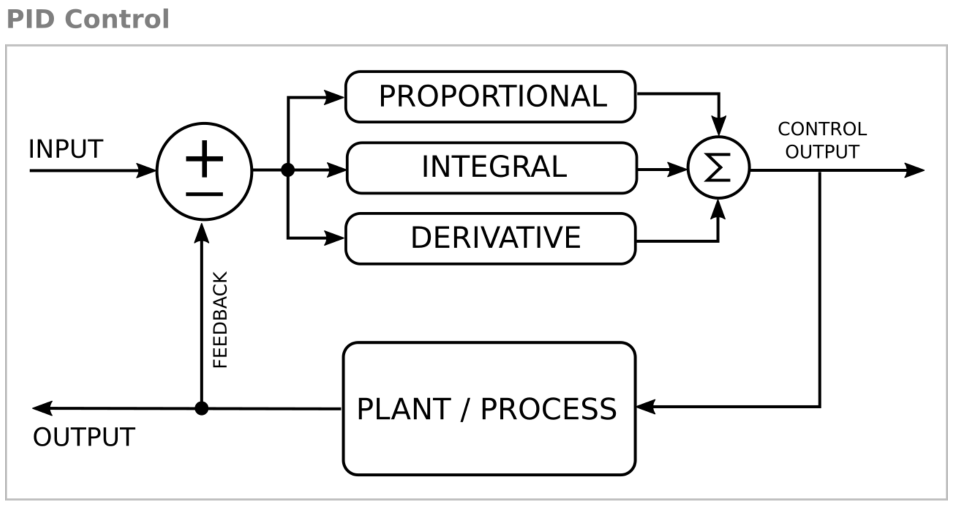

Figure 7.2: PID Control System Block Diagram.

Proportional

The proportional control factor represents the following relationship between the output control signal and the input error:

$$c(t) = K_p e(t)$$

The constant K_p is the proportional scaling coefficient and determines the strength of the action taken to correct the error signal. Proportional feedback applies an output signal proportional to the error, essentially scaling the error by K_p. Proportional control provides rapid correction when the error signal is large, but loses effectiveness as the plant output nears the setpoint. Additionally, proportional control has the limitation that can not not adjust if the error persists through the initial action.

Integral

Integral control grows proportionally to the integral of the error signal, and represents the following portion of the PID equation:

$$c(t) = K_i \int_0^t e(\tau)d\tau$$

Integral control begins with a small value regardless of the error's magnitude, but increases with the duration of the error. This means that even small errors will eventually build into large correction factors. This offers an advantage over proportional control as an integral based system will continually adjust until the error is corrected. The growth of the accumulated error is scaled by the integral scaling coefficient K_i.

Because these systems are based on the integral of the error signal, they have the disadvantages of beginning slowly, and overshooting the target setpoint. This overshoot is caused by the need for an equal amount of negative error to be accumulated to return the integral's value back to zero. This phenomenon is called "wind-up" and is managed in most systems by setting a maximum value that the integral is allowed to accumulate.

Derivative

The final portion of the PID controller is derivative control, representing the last portion of the system equation:

$$c(t) = K_d \frac{de(t)}{dt}$$

A derivative controller reacts to the rate of change in the output of the system. Typically derivative effects are used to minimize overshoot oscillations caused by integral control. However, they also respond rapidly to unexpected changes in the output of the system. The reactive strength of a derivative controller depends on it's scaling factor K_d.

The major drawback for derivative controlled systems is that they can amplify minor errors or oscillate in systems with natural fluctuations. Derivative systems are more difficult to tune properly, and depending on the process many systems use only proportional and integral control.

7.2 Modeling DC Motors

DC motors when operating steady-state appear mostly as a series resistive-inductive loop with a voltage drop dependent on the current angular velocity. This voltage drop is due to the back-electromotive force or EMF which is generated from the motion of the motor armature relative to the windings. Figure 7.3 shows a model of a simple DC motor system.

7.2.1 Equations Modeling a Motor System

There are a number of basic equations that model aspects of a motor system. In this lab we will be primarily concerned with only first-order effects, many of the complex parameters are either simplified or removed.

The following equations use the following terminology:

- V – DC drive voltage

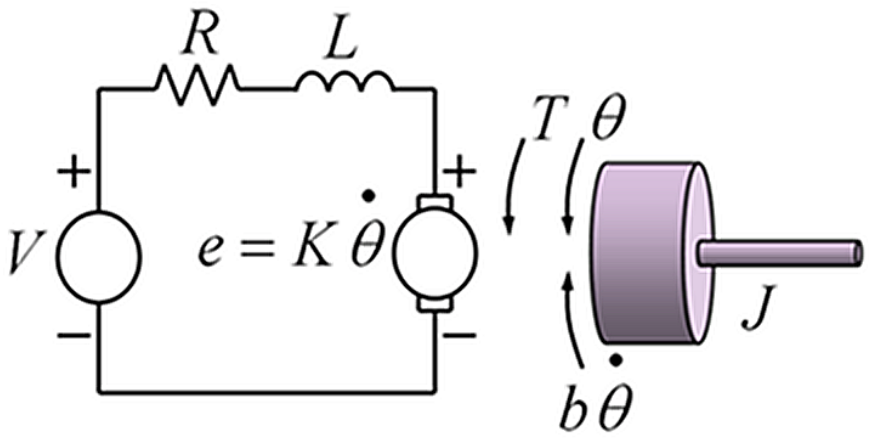

Figure 7.3: Variables and constants in a motor system.

- R – Motor winding resistance
- L – Motor winding inductance
- i – Motor current
- θ – Angular position
- $\dot{\theta}$ – Angular velocity (rotational speed)
- $\ddot{\theta}$ – Angular acceleration
- T – Motor Torque
- f – System friction
- K – Torque constant
- e – Electromotive force
- J – Moment of inertia
- t – Time, (s is used once in Laplace domain)

Torque Equation

$$T(t) = Ki(t) = J\frac{d\dot{\theta}}{dt} = J\ddot{\theta}(t) \tag{7.1}$$

Electromotive Force

$$e(t) = K\dot{\theta}(t) \tag{7.2}$$

Kirchhoff's Law

$$v(t) = Ri(t) + e = Ri(t) + K\dot{\theta}(t) = \frac{RJ}{K}\ddot{\theta}(t) + K\dot{\theta}(t) \tag{7.3}$$

1st-Order Transfer Function

$$\frac{\dot{\theta}(s)}{v(s)} = \frac{1}{\frac{RJ}{K}s + K} \tag{7.4}$$

7.2.2 Deriving the Transfer Functions

When modeling any system the first step is to derive the characteristic transfer function. This equation represents the modeled system and supposedly responds in a mathematically accurate way when set with parameter values. Within this lab you are going to derive the transfer function of a theoretical DC motor and use simple integer values for many of its parameters.

The typical form of a transfer function of a DC motor is the angular velocity (output) divided by the input voltage $\frac{\dot{\theta}(s)}{v(s)}$. Using this function gives a reasonable estimate of motor velocity for a specific input voltage. However, as shown in the lab exercises, other coefficients such as friction will play a large role in the output of the system.

To derive the 1st-order transfer function, follow these steps. Your goal is to combine the torque and Kirchoff's equations together and then simplify using Laplace transformations.

1. Start with the mechanical torque equation.
2. Solve for the motor current i by rearranging the torque equation.
3. Replace the torque T term with its acceleration form in the rearranged current equation. (acceleration form listed above)
4. Replace the current i term in Kirchoff's law with the rearranged current equation.
5. Replace the electromotive force e term in Kirchoff's law with its velocity and torque constant form.
6. Perform a Laplace transform of the modified equation, using an identity to convert the acceleration term into velocity.
7. Isolate the velocity term on the left-hand side with the voltage as the denominator

7.3 Lab Assignment: Derivation of a DC Motor Transfer Function

The following portions of the exercise create the "plant" or motor model used in the simulated control system. You will use your derived transfer functions within the Simulink model created in the second exercise.

1. Derive the first-order transfer function (velocity/voltage) of a DC motor without any friction and inductive effects.
   - You may wish to review class lecture 16 slides, the transfer function is shown there as well as this manual. You will need to show derivation work for credit!
   - You will need to use Laplace transformations to solve your first-order transfer function.
   - Your final equation should have the form: $\frac{\dot{\theta}(s)}{v(s)} = ?$
2. Assuming the simple case where the torque constant K, the moment of inertia J, and the electrical resistance R, all have a numerical value of 1, simulate the response of the function to a step of amplitude 1 using Matlab.
   - The step function represents suddenly turning the motor on (providing voltage) when the motor was previously idle.
   - You will need to create a vector containing values representing the step function and pass it through the derived transfer function.
   - The resulting output vector is the motor's velocity response.
3. Study and report the effects that changing the torque constant K, moment of inertia J, or electrical resistance R, have on the motor's response to the step function.

4. Derive the transfer function of a DC motor with friction. Note that the effect of friction can be modeled as an opposed torque of value $f\dot{\theta}$ (with f the friction constant).

7.4 Lab Assignment: Simulating the Motor in Simulink

Within this exercise you will convert your transfer function into a form that can be used within a Simulink model. You will use your model to experiment with changing the basic motor parameters under the influence of friction.

1. In order to use your transfer function within Simulink, you will need to rewrite to represent only the velocity.
   - Your original transfer function was in the form $\frac{\dot{\theta}(s)}{v(s)} = ?$, you will need to convert it into $\dot{\theta}(s) = \frac{1}{s} * ?$
   - You will want to isolate the Laplace "s" term from the rest of the equation. This is because Simulink represents integration as a discrete "$\frac{1}{s}$" block.
2. Create a Simulink subsystem model of the transfer function you derived. It should take the coefficients for the moment of inertia, friction, torque constant and voltage.
   - See Figures 7.4 and 7.5 for examples of what the interface and contents of the subsystem should resemble. (Figure 7.5 doesn't represent an actual transfer function. **Don't try to use it!**)
   - Assume that the motor resistance is 1 to simplify the system. ($R = 1$)
   - The following tutorial on subsystems may be helpful: http://www.mathworks.com/help/simulink/ug/creating-subsystems.html
3. Simulate the system using a step function of amplitude 1 for the voltage; with values for the torque constant K, and moment of inertia J set to 1.
   - Simulate the model using different values for the friction parameter.
   - The results should be similar to the second portion of the first exercise when the friction factor is zero. What happens if when the friction factor increases?
4. Perform a simulation with a step function controlling the friction parameter. (so it changes in the middle of the simulation) What happens, and why would friction dynamically change in a physical system?

7.5 Lab Assignment: Designing a PI Controller in Simulink

This exercise simulates portions of a PI (proportional & integral) control system within the Simulink modeling framework. After completing this portion, you should understand the basic operation and be able to tune the modeled system for reasonable performance.

1. Implement a feedback control using only the proportional term.
   - Add a saturation block/module on the output of the control system. Set this limiter to allow a maximum value of 6. This approximates reasonable limitations of your drive hardware which can't increase the output voltage past what is supplied.
   - Study and comment the response of the system.
2. Improve the feedback control loop by adding an integral term.

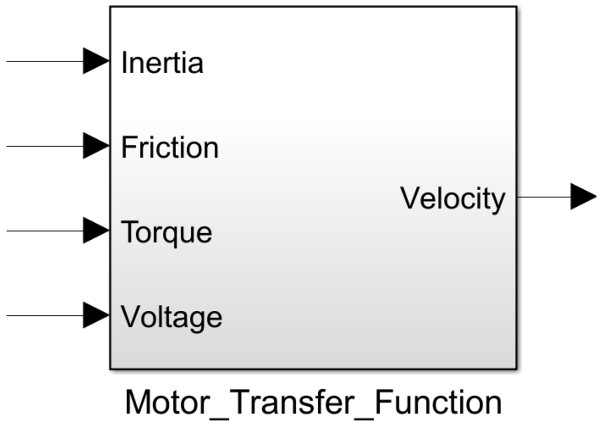

Figure 7.4: Example motor transfer function subsystem interface.

- Study and comment on the response of the system.
- You may need to scale the integral and proportional terms to prevent the system from oscillating.

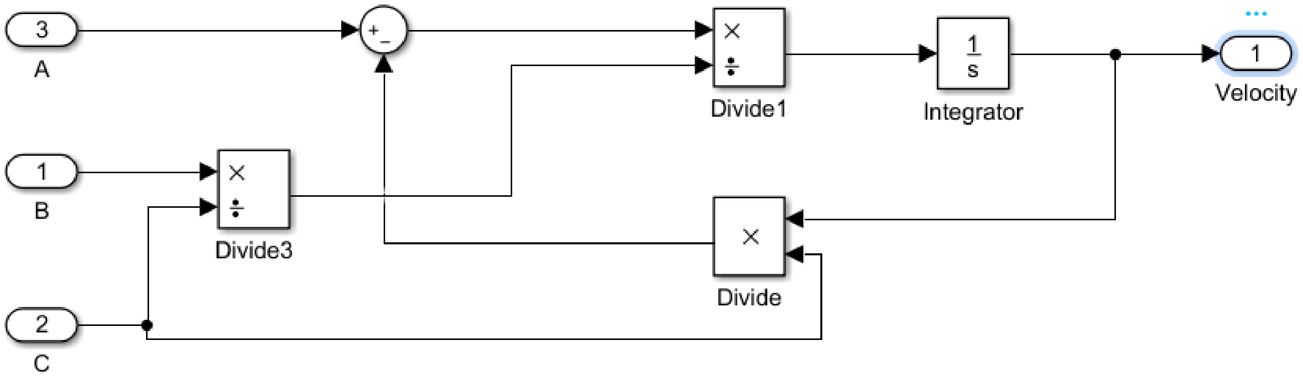

Figure 7.5: Example motor transfer function subsystem contents.