MAE 5316 Mechatronics

Project #5

DC Motor Position and Velocity Control by XPC Target

PID Controller

By:

Robert Andringa

Sagar Jagtap

Objective

To create two PID controllers for a DC motor in order to control:

1. Angular position in radians
2. Angular velocity in radians per second

Methods

MATLAB and Simulink are used to set up a system for control. An XPC target is used to link to Simulink to provide real time control. The DC motor includes an optical encoder to read counts which correspond to revolutions. The counts per revolution are converted to radians, which shows the position and direction from zero. The change in radians over sample time shows the velocity in radians per second. The input commands are formed from the step function. For the position control, the step functions start at zero radians away from zero, ramps up to a positive position and then ramps down below zero to a negative position. This is to ensure both directions and the crossover of zero position are taken into account in the PID controller tuning. For the velocity control, the function starts at zero radians per second, ramps up to a positive velocity (clockwise rotation), and then ramps down below zero radians per second to a negative velocity (counterclockwise rotation). This is to ensure both clockwise and counterclockwise rotations are taken into account in the PID controller tuning.

Setup

The system setup is shown in Figure 1 below. The NI 6024E is the interface to the XPC target. The Motor Driver is powered from the output of the NI 6024E by pin 14 for +5V and pin 55 for ground. Pin 16 on the NI 6024E is a digital output P0.6 pin that sends the PWM signal to the Enable pin on the Motor Driver. This signal is created by Simulink and transfers to through the NI6024E block. Pins 47 and 48 on the NI 6024E are digital output P0.3 and P0.7 respectively, and they send a HIGH or LOW signal from Simulink to the IN1 and IN2 Motor Driver pins respectively. This determines the clockwise or counterclockwise direction of the Motor. The Motor Driver accepts 12V from a power source and transfers it to the Motor pins 6 and 5. Pins 8 and 15 on the NI 6024E power the Motor Encoder with +5V and ground respectively to pins 3 and 4. Pins 17 and 49 on the NI 6024E are digital inputs P0.1 and P0.2 and connect with Motor Encoder pins 1 and 2 respectively. This signal is the Motor Encoder counts that is outputted to Simulink to calculate position and velocity.

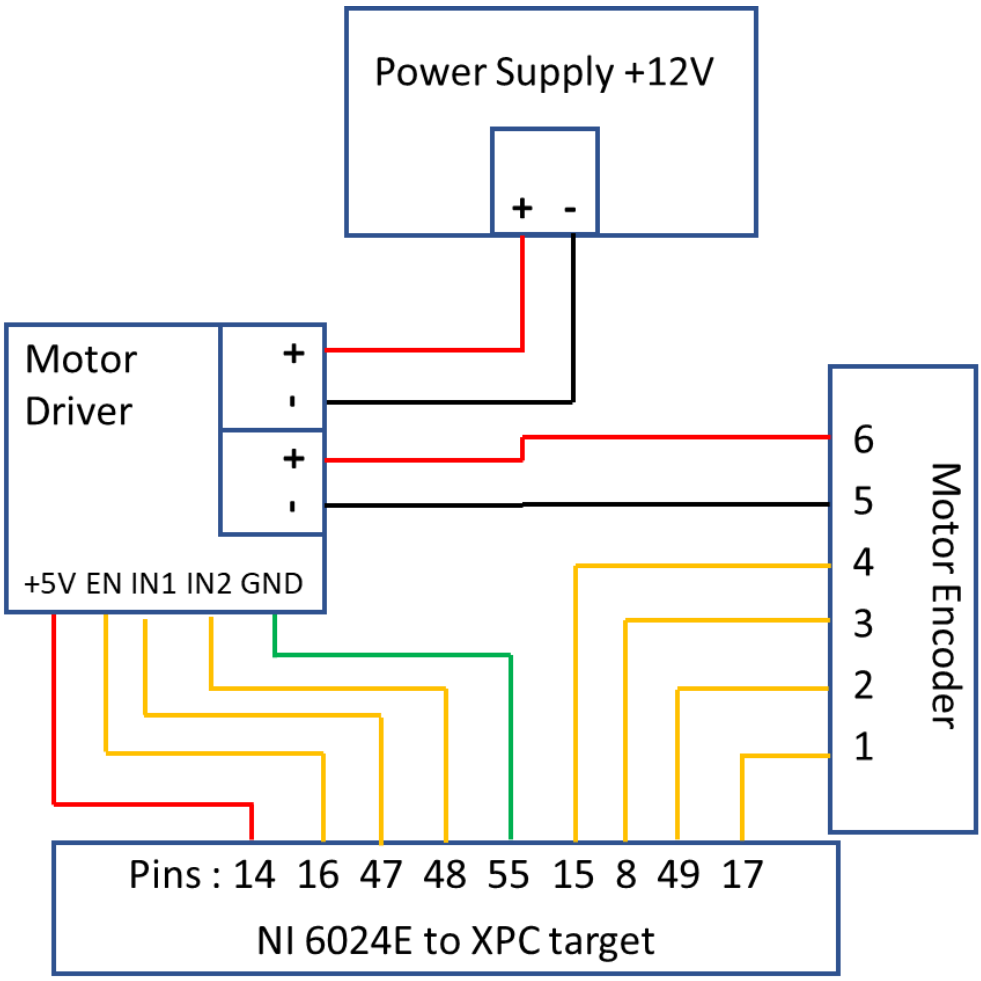

Figure 1 - Component and Wiring Diagram

Position Control

Position value is accomplished by calculating and accumulating the position as it relates to its zero-starting point. The counts and direction are added and converted to radians. This output signal is subtracted from the input signal, which is also converted to radians from volts, in order to find the error of the system. The error is run through a PID controller to attempt to reduce the error to zero. This value is then output back to the Motor Driver PWM. If it is greater than 1 then the Motor will continue its 100% duty cycle. If the error turns negative the compare blocks in Simulink will switch the Motor direction to counterclockwise. This will continue until the input signal is equal to the output position and the error is zero. The PID gains are tuned to provide the fastest route to the correct position with the least overshoot and oscillations. Figure 2 below shows the Simulink model used to tune the PID controller.

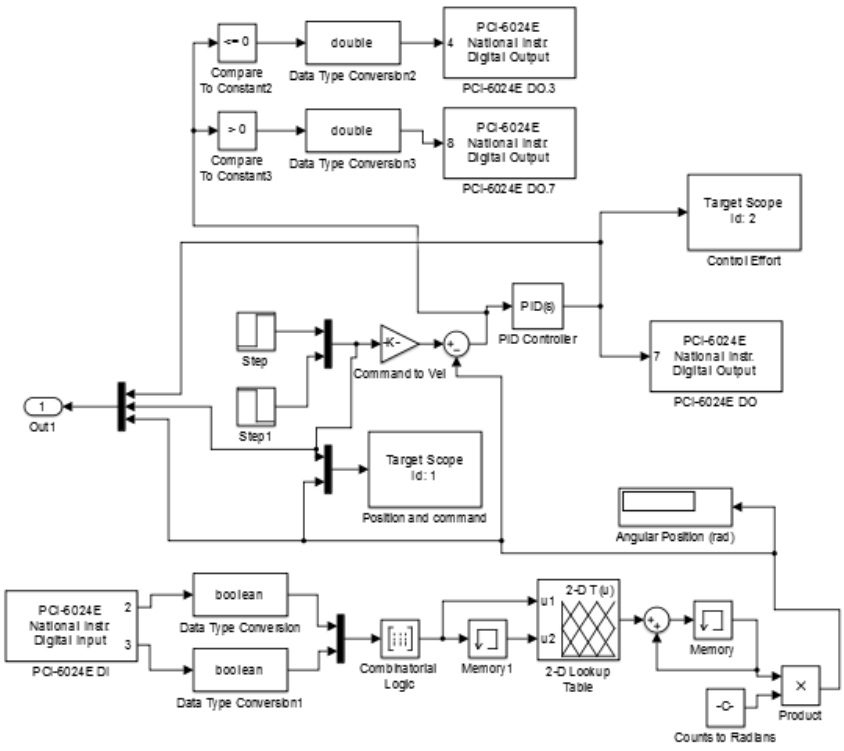

Figure 2 - Simulink Model for Position PID Control

Velocity Control

Velocity value is accomplished by taking the position value, subtracting it from its previous value, and dividing it by the sample time of 0.001s. This is the change of position over time or radians per second. Similarly, to the position control, the velocity is subtracted from the input command signal to find the error and sent to the PID controller. Different from the position control model, the direction of the motor depends on the input command signal and not the error. The PID controller is setup to output a value no higher than 1 to create a duty cycle. The value is split off, one going to a PWM generator and the other going to a compare block to reduce oscillations once the error gets to 0.4. The values are added back together and sent to the output to the Motor Driver. The velocity should increase from zero to the input command signal, varying the duty cycle to achieve zero error. The velocity is the ramped down to below zero to show a negative or counterclockwise velocity. The PID controller is tuned to create the fastest route to the command velocity with the least amount of overshoot and oscillations. Figure 3 below shows the Simulink model used to tune the PID controller.

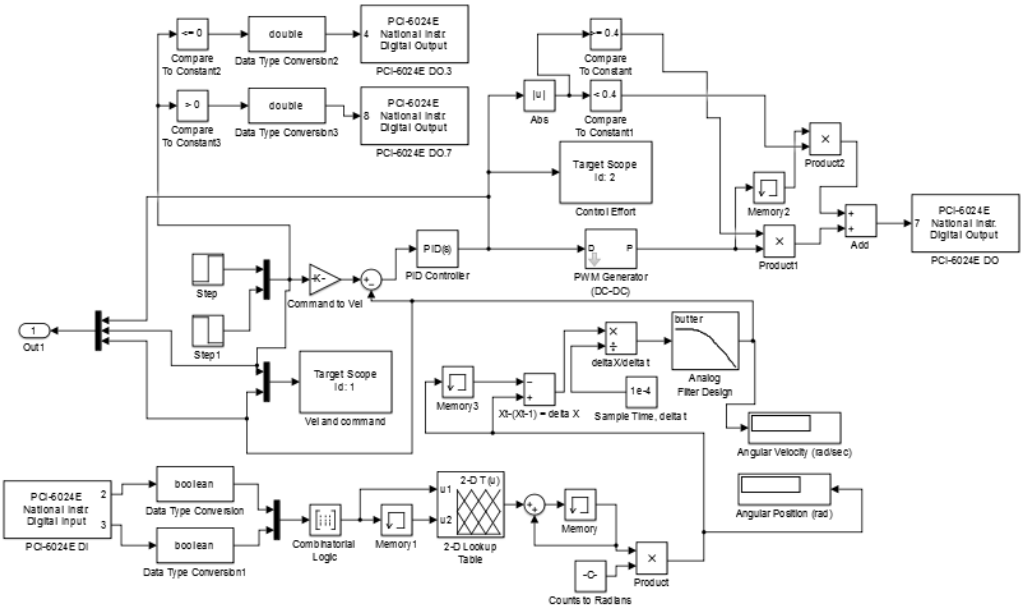

Figure 3 - Simulink Model for PID Velocity Control

Position PID Tuning

A variety of Proportional, Integral, and Derivative gains are used to create a relatively stable simulation. A range of those gains are then simulated and the Command vs Position data is plotted together to determine any differences between each run.

Table 1 – Position PID Tuning Values

	P	I	D
1	1	0	0
2	1	0.01	0.05
3	1	0.01	0.5
4	1	0.05	0.05
5	0.5	0.01	0.05
6	2	0.01	0.05
7	2	0.01	0.1
8	5	0.01	0.05
9	0.1	0.01	0.05

Table 1 above, shows the P, I, and D values used for each run. The results are shows below in Figure 4.

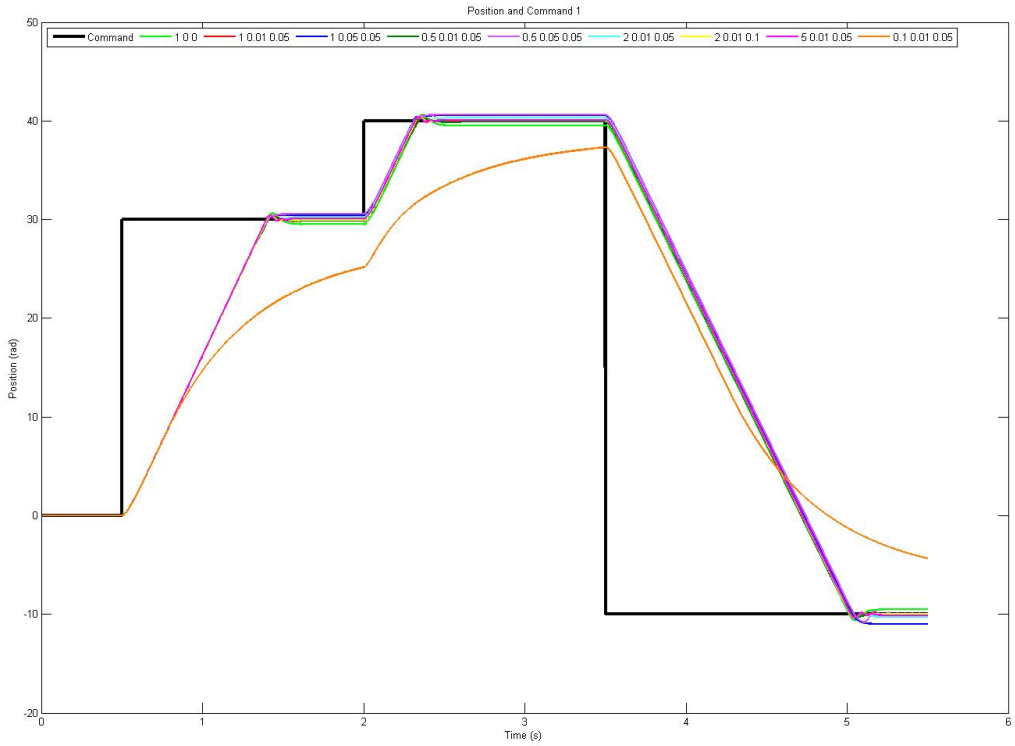

Figure 4 - Position PID Tuning Runs

The values that best fit the input command signal are from run 2 and include P = 1, I = 0.01, and D = 0.05. It is shown in the Figure 4 above as the red line and in Figure 5 below. This run is chosen due to the time it takes to reach its command and the least amount of overshoot and oscillations. Figure 5 below also shows the control effort as it attempts to zero out the error.

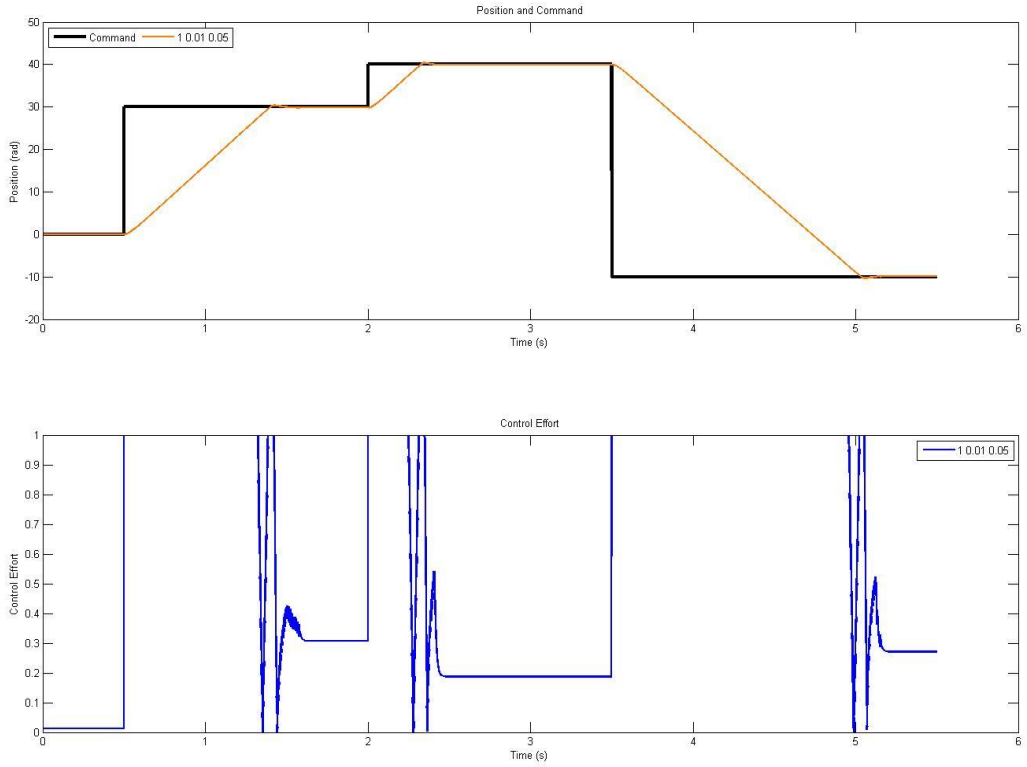

Figure 5 - PID 1 0.01 0.05 vs Command and Control Effort

Velocity PID Tuning

A variety of Proportional, Integral, and Derivative gains are used to create a relatively stable simulation. A range of those gains are then simulated and the Command vs Velocity data is plotted together to determine any differences between each run.

Table 2 – Velocity PID Tuning Values

	P	I	D
1	0.02	0.2	0.002
2	0.02	0.4	0.002
3	0.02	0.2	0.001
4	0.02	0.4	0.001
5	0.03	0.2	0.002
6	0.03	0.2	0.001
7	0.03	0.4	0.001
8	0.04	0.2	0.002
9	0.04	0.4	0.002
10	0.04	0.2	0.001
11	0.04	0.4	0.001

Table 2 above, shows the P, I, and D values used for each run. The results are shown below in Figure 6.

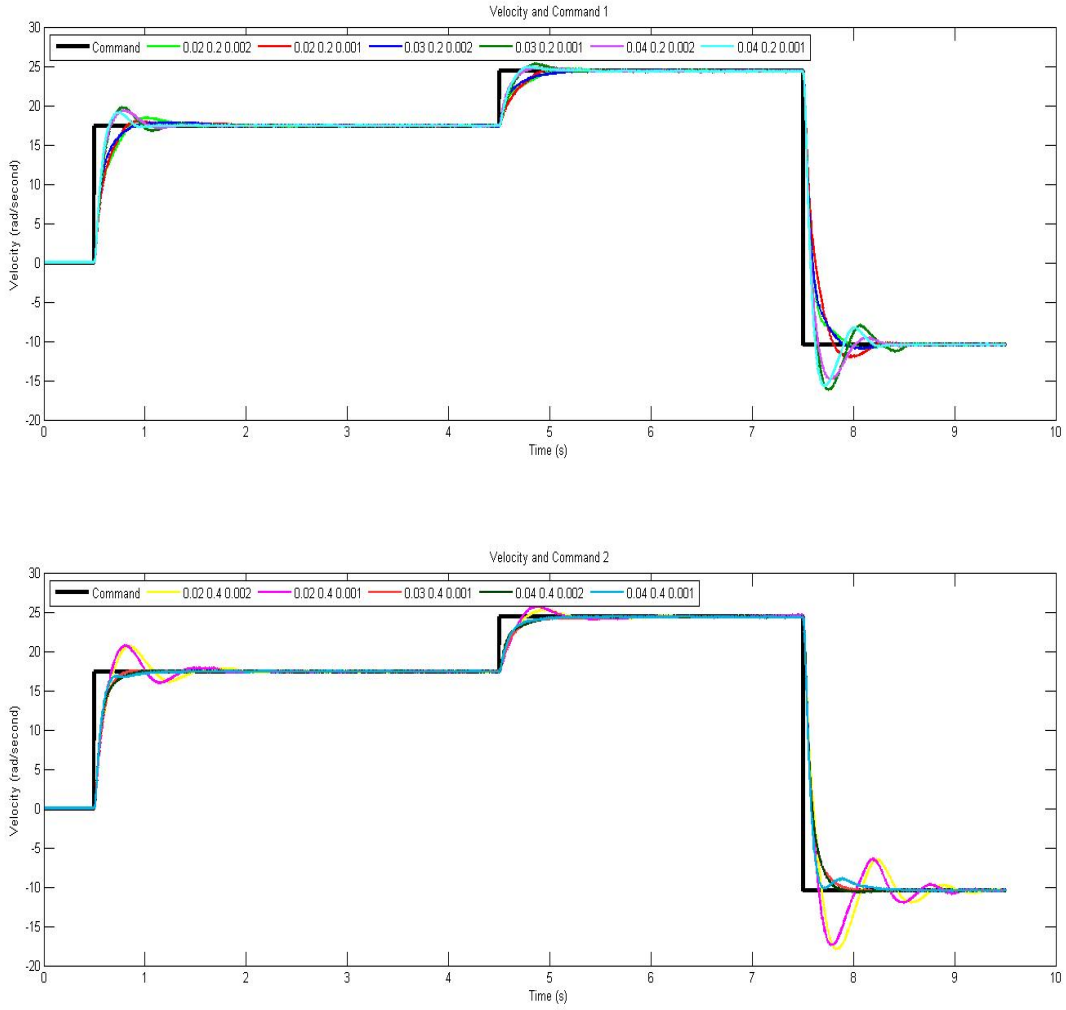

Figure 6 – Velocity PID Tuning Runs

The values that best fit the input command signal are from run 3 and include P = 0.02, I = 0.2, and D = 0.001, run 5 and include P = 0.03, I = 0.2, and D = 0.002, and run 7 and include P =0.03, I = 0.4, and D = 0.001. These runs are chosen due to the time it takes to reach its command and the least amount of overshoot and oscillations. Figure 7 below also shows the control effort as it attempts to zero out the error.

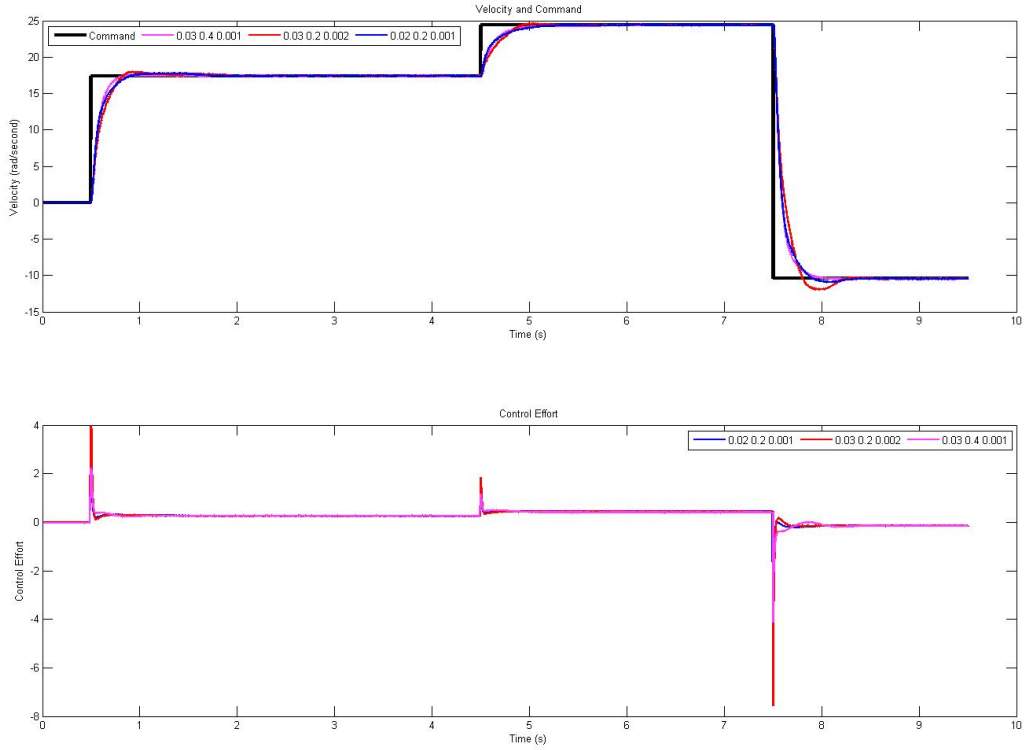

Figure 7 - Run 3 vs Run 5 vs Run 7 and Control Effort

Out of the three runs shown in Figure 7 above, the run 7 shows a fast response with the least amount of overshoot and oscillations. The final PID values are P = 0.03, I = 0.4, and D = 0.001.