# Abstract

For this project, a PID controller that runs on QNX, is developed to control a plant simulator that runs on the development station. Data Translation DT9812 data acquisition board is used to provide input to plant simulation and get output signal from plant simulation. The program is run with different values for the PID controller constants and results are captured for each set and plot of plant output signal against time is included.

# Design Methodology

A QNX c program is developed to implement a PID controller which controls a plant simulator that runs on the development station and generates the plant output signal through the Data Translation DT9812 data acquisition board. This program uses multiple threads. The input scan thread reads the plant output signal from analog to digital converter. Input channel 4 which is connected to analog output channel 0 of the DT9812, is used as analog input to ADC. PID controller thread calculates the new plant control input based on the previous plant output signal. Based on the PID controller constants (Kp, Ki, Kd) and last output from plant, a correction value is calculated which is the next plant control input. Plant simulator output thread writes the plant control input to Digital to Analog converter registers and the output of DAC is connected to analog input 0 of Data Translation DT9812 data acquisition board. Kill all thread kills all threads after 15 sec to end the application after 15 sec. Main thread spawns all 4 threads and waits on completion of each thread. Three semaphores are used for synchronization among Input scan, PID controller and Plant simulator output threads to ensure proper correct flow of program. PID controller threads waits on the semaphore syncPID, which is to be posted by Input scan thread. Input scan thread reads the converted ADC value and posts the semaphore syncPID. And Input scan thread waits on the semaphore syncADC, which is to be posted by Plant simulator output thread. Plant simulator output thread writes the 16 bit plant control input to registers and posts the semaphore syncADC. Plant simulator output thread waits on syncDAC semaphore, which is to be posted by PID controller thread. PID controller thread calculates the plant control input value and posts the semaphore syncDAC. Simplified, the program flow is to receive plant output signal from plant simulator, calculate the plant control input, and send the plant control input to plant simulator. The set point, i.e. the desired plant output voltage is set to 2.3 volts. After experimenting with different sets of PID controller constants, steady plant output within a short period of time is achieved with the following PID controller constants: (Kp = 2.5, Ki = 0.03, Kd = 2.5).

# Results

The program is run on 4 sets of PID controller constants, which includes one parameter set that gives optimal control and three other interesting parameter sets. For each parameter set a plot of plant output against time is shown. Also percentage overshoot and the rise time are calculated and listed out for each parameter set.

1. Kp = 2.5, Ki = 0.03, Kd = 2.5

Figure 1. Plot of plant output signal and plant input signal against time, with PID control

Figure 1 above shows the final PID waveform. It can be seen that the wheel and car settle about 2.5 seconds after the input starts to change.

Figure 2. Plot of plant output signal and plant input signal against time, without PID control.

Figure 2 above shows the waveform of step function without and PID. It can be seen that the wheel and car settle about 4.5 seconds after the input starts to change. This is nearly twice the amount of time it takes to settle compared to using the PID controller.

Figure 3. Comparison of plant output in both cases, i.e. with PID control and without PID control

The red line shows the signal with the PID controller and the blue line shows the signal with the step function. The red line clearly settles much faster than the blue line.

From the results, Rise time = 0.4 seconds and percentage overshoot = 32.84 %

1. Kp = 2.5, Ki = 0, Kd = 0

Figure 4. Plot of plant output signal and plant input signal against time, with PID control

From the results, Rise time = 0.5 seconds and percentage overshoot = 5.48 %

In this case, the plant output does not reach the set point of 2.3 volts.

1. Kp = 0, Ki = 0.03, Kd = 0

Figure 5. Plot of plant output signal and plant input signal against time, with PID control

From the results, Rise time = 0.7 seconds and percentage overshoot = 66.74 %

Here, the plant output keeps oscillating between two values.

1. Kp = 2.5, Ki = 0.03, Kd = 10

Figure 6. Plot of plant output signal and plant input signal against time, with PID control

From the results, Rise time = 0.4 seconds and percentage overshoot = 34.77 %

Derivative term should ideally improve settling time and stabilize the system. But in this case where high derivative constant is used, system becomes unstable and takes more time to settle.

# Experimental approach

The approach taken to determine the PID values was very simple. First, the P term was solved for and the I and the D term were set to 0. When the P term got the input signal to roughly 75% of the desired set point it was set. Then the Integral term was solved for using that P term and setting the D term to 0. This took more of a guess and check approach until a reasonable value was chosen. What was considered reasonable was a signal that settled at the desired set point quicker than a step function. Initially values around 1 were chosen but determined to have way too much impact so they were dramatically dropped. Then finally the Derivative term was added to smooth out the signal and reduce the overshoot. This was again guess and check. The D term did not have that great of an impact however.

# Conclusion

The goal was to have the input changing to allow the response to settle at a set point faster than if it had not varied at all. This was accomplished and a comparison can be seen in figures 1, 2, and 3 above. The PID allowed the signal to settle at the set point in about 2.5 seconds instead of 4.5 seconds for an unvarying signal. Unfortunately the signal still produces an overshoot and ringing affect. For future improvements that ringing should be eliminated. This is usually done with the Derivative term of the PID controller. The PID controller was still successful in settling the set point at a signal faster than without it.