Nathan Isaman ME 477: Lab 7 4 Mar 17

1. Description

The purpose of this lab was to code and implement a Proportional plus Integral (PI) Controller to control a DC motor. The block diagram for this system can be found below:

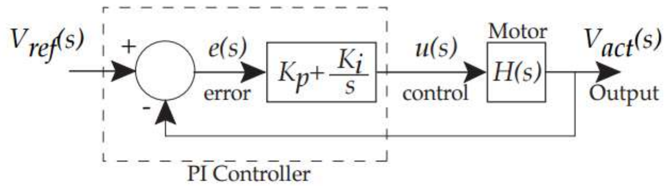

Figure 1: DC Motor/Controller Block Diagram

This feedback control system was implemented using interrupts and three threads: main, timer, and update. The main function thread initialized I/O, created and registered interrupt threads, and finally unregistered and terminated the interrupt threads. The Timer IRQ was pivotal in controlling the motor and was responsible for implementing the controller seen in Figure 1 through calls to *vel()*, *cascade()*, and *Aio_Write()*. The Update IRQ was used to update the control table on the LCD screen and allow the user to input new gain values for the control law without interrupting the motor's operation.

Hierarchy:

```
main()
|_MyRio_Open()                                   // Opens a session with the MyRio
|_AIO_initialize()                               // Initializing Analog Input/Output
|_Aio_Write()                                    // Further initialization
|_EncoderC_initialize()                          // Initializes the encoder interface
|_Irq_RegisterTimerIrq()                         // Register the timer interrupt
|_pthread_create()                               // Create a new thread for the interrupt
|       |_Timer_Irq_Thread()   (argument)        // Interrupt service function
|_pthead_create()                                // Register the update thread
|       |_Table_Update_Thread() (argument)       // IRQ that updates the control table
|_ctable()                                       // Displays the control Table on the LCD
|_pthread_join()                                 // Terminates the timer IRQ
|_pthread_join()                                 // Terminates the update IRQ
|_Irq_UnregisterDiIrq()                          // Unregister the timer IRQ
|_printf_lcd()                                   // Confirmation message on LCD
|_MyRio_Close()                                  // Closes the session with the MyRio

Timer_Irq_Thread()                               // Timer interrupt service function
|_while()                                        // Interrupt until thread stopped by main
|       |_Irq_Wait()                             // wait for IRQ to assert or time out
|       |_NiFpga_WriteU32()                      // Scheduling next interrupt
|       |_NiFpga_WriteBool()                     // More timer interrupt parameter setting
|       |_if()                                   // if the IRQ is asserted
|               |_vel()                          // calculates the BDI
|               |_cascade()                      // call cascade and process input/output
|               |_Aio_Write()                    // transmit y0 to analog output
|               |_Irq_Acknowledge()              // Interrupt acknowledged to the scheduler
|_pthread_exit()                                 // Terminate the new thread
```

```
Table_Update_Thread()                    // Update interrupt service function
|_while()
|       |_nanosleep()                    // sleeps for a designated # of nanosec
|       |_update()                       // updates the control table
|_pthread_exit()                         // Terminate the thread

cascade()                                // Uses a biquad cascade
|_for()                                  // difference eq to calculate y0

vel()                                    // Function calculates the BDI
|_Encoder_Counter()                      // gets the current count from the encoder
```

2. Testing

The code was initially tested using recommended values for V_ref, Kp, Ki, and BTI (200/-200, 0.1, 2, and 5 respectively). The system was then tested using different gain, rpm, and BTI values to ensure the response was as expected for a PI controller. The motor speed displayed on the LCD screen was also compared with the value read by a tachometer. Most of the analysis was done qualitatively as there was no data collected and returned in a MATLAB file for this lab.

3. Results

The basic code components provided for this lab functioned as expected. Update() allowed the user to successfully change the system parameters and these changes were implemented by the control system while operating. The ctable() function displayed the table values and allowed the user to shift the table as expected.

The system responded as expected with the recommended values for V_ref, Kp, Ki, and BTI. When 200 rpm was used for V_ref, the motor turned in the clockwise direction and counter-clockwise when -200 rpm was used. This verifies that negative values will drive the motor in the opposite direction compared to positive values of rpm. When V_ref was set to 200rpm, values for V_act stayed within roughly 5-10 rpm of this value. The V_act value displayed was verified using a tachometer and the values between the LCD and the tachometer agreed (with some reasonable error). The system was also tested using higher gain values for Kp and Ki and an external disturbance. The expected response with higher gain would involve more overshoot compared to lower levels of gain and this was seen in the motor's response. At one point, the system was set to a V_ref of 500 rpm, Ki of 10, and Kp of ~5 and a large disturbance was introduced to the system. The system response was large enough that the current amplifier faulted, despite the saturation macro used on the output voltage, and the test was terminated. Future tests should be formulated with the current limitations inherent in the system in mind. Another test was done with a higher BTI and it was observed that the fluctuations in V_act were larger than those seen with a lower BTI. This is expected as the system is not comparing the measured and reference velocities as frequently and the system has more time to diverge from the reference value. This larger divergence requires a large response from the system to return it to normal, which leads to more overshoot prior to correction in the opposite direction.

Overall, this experiment was a success and the feedback control for the DC motor acted as desired, with a few supply-related hiccups.