Lab #7 DC-Motor PI Velocity Control

1. Description

In this laboratory we are implementing a PI closed loop system to control the velocity of a DC motor. We learned about performing tasks in the main thread which interacts with the timer interrupt to adjust the controller gains of the DC motor. The K_i and K_p, BTI, and v_ref is assigned by a table that is shown on the LCD screen and editable through the keypad.

The hierarchy of the routines in this project is described below.

```
main()
|_ myRio_Open()
|_ Irq_RegisterTimerIrq()
|_ pthread_create()
    |_Timer_Irq_Thread()
        |_ Irq_Wait()
        |_ NiFpga_UWrite32()
        |_ NiFpga_WriteBool()
        |_ Aio_Read()
        |_ Aio_Write()
        |_ vel()
        |_ cascade()
        |_ Irq_Acknowledge()
        |_ pthread_exit()
|_ ctable2()
|_ pthread_join()
|_ Irq_UnregisterDiIrq()
```

Like the previous labs, the main program is responsible for registering the timer interrupt and the table editor. The timer interrupt contains one extra field containing a pointer to the table in order to modify the values in the table. This allows us to perform edits of the control parameters in main and send them to the timer interrupt which is responsible of doing the velocity control. Editting the values in the table is done by calling ctable2() in the main thread.

As mentioned previously, the closed-loop velocity control is done in the timer interrupt thread. The timer interrupt thread is implemented similar to that of Lab 6 with the additional feature of being able to modify the coefficients in the biquad structure. The parameters described in the biquad is described in the transfer function description below.

In the figure above, we are able to modify K_p (proportional gain), K_i (integral gain), and T (sampling rate). These modifiable parameters can be used to calculate the biquad structure's parameters like a0, a1, b0, and b1. Lastly, we can also change the v_ref for the motor, which specifies the error of the motor velocity used as the input of the transfer function. This determines the voltage output sent to the motor which sets the torque of the motor. All

$$\begin{aligned} T(z) &= \frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1}}{a_0 + a_1 z^{-1}} \\ a_0 &= 1, \quad b_0 = K_p + \frac{1}{2}K_i T \\ a_1 &= -1, \quad b_1 = -K_p + \frac{1}{2}K_i T \end{aligned}$$

Figure 1: Biquad Coefficients Formula

the coefficients calculated must be scaled properly to the correct units, just like how we calculated the actual velocity of the motor in RPM in Lab4. (please check the source code to see the unit conversion implementation)

Lastly, to import the measured data to MATLAB, I triggered the measurement only when there is a change in vref, which indicates the beginning of a step. Once there is a step input, the value of K_i, K_p, vref, and the BTI will be updated according to our input. Next, the actual velocity (vact) and the output voltage (VDAout) will be taken until the buffer is full. The torque measurement can be calculated by multiplying VDAout with the motor gain, k.

2. Testing

To test the program, run main.c and follow the procedure below.

1. After main.c is executed, the LCD screen should print the table containing the editable and measured parameters. If main.c is executed after the previous program has not been terminated, it should stop the motor.

2. Check if all the editable parameters are actually editable. Use the up or down button to naviagate to all the options. The editable parameters are: vref, K_p, K_i, and BTI. Change all of them. The recommended setting of the testing parameter is described below: If setting the variables

V_{ref}	±200	rpm
BTI length	5	ms
K_p	0.1	V-s/r
K_i	2.0	V/r

Figure 2: Testing Parameter

to move the motor does not work, the most common error is from the miscalculation of the VDAout. Please check if all the table parameters are actually updated by ensuring that the timer interrupt contains the *a_table pointer to access the table. In addition, please check the unit conversion and ensure that the timeout value is the BTI value multiplied by 1000 (default is 5 ms). A suggestion to check if there is something wrong in the cascade() function or the Aio_Write() is by passing in 1 as the first parameter of cascade() and passing in the result to Aio_Write(), which will force the motor to turn.

3. If the motor runs after the enter key is pressed upon modifying each parameter, test the response of the motor. The LCD screen should show that the vact (actual velocity) will eventually reach the vref value without disturbance. The oscilloscope is a great way to observe the voltage the controller is sending to the motor.

4. To test if our system adapts to disturbance, apply friction to the circumference of the flywheel carefully with your finger. You should see that VDAout increases in the LCD screen and the oscilloscope reads higher Voltage reading. Furthermore, VDAout should saturate at 7500 mV and -7500 mV.

5. Check if the motor is able to perform negative velocity control by specifying negative vref. This will cause the motor to rotate counterclockwise. Also, check if you can hold the motor and turn it to the counterclockwise direction.

6. We should also be able to change every parameter on the go. Note that changing the controller gains will cause the motor to have different response performance. Generally, increasing K_p will cause the motor to response more rigorously, while sacrificing overshoot. Increasing K_i will reduce the steady state error which shows up when the disturbance is continually increasing.

7. To record the torque and vact data to be analyzed in MATLAB, input the parameter described in the Figure 2. Enter -200 rpm as the vref. Once the velocity has reached steady state, immediately change vref to 200 rpm. The code should automatically take measurements.

8. After each procedure is tested, press backspace (DEL key) and the motor should stop and the MATLAB file is available to be imported. Check if the vact and the torque imported in the MATLAB file is meaningful result. When I imported my data, the torque and vact value stays flat at 0, and I fixed it by initializing the variable pointers once the end of the buffer is reached.

3. Results

The program runs as intended. The error I encountered was importing the MATLAB file, and I fixed it by initializing the pointer to the beginning of the buffer once the buffers are full. Another bug I encountered was the velocity of the motor blowing up when negative rpm is entered or when I forcefully turned the motor to the counter clockwise directly. This issue is easily fixed by changing the counter types to signed integers.

Compare the tachometer reading with vact. How close is it?

When 200 rpm is entered as vref, vact reading in the table fluctuates between 198 and 204 rpm, which is due to the quantization error. The tachometer reading shows 198 rpm, which is a really accurate reading compared to the reading in Lab 3. This result is achieved because the control is implemented in a separate thread.

When the motor is at steady speed and a steady load torque is applied, what happens?

When a steady load torque is applied by using my finger to hold the motor a little, the actual speed drops for a quick moment, then the motor compensates until the speed returns back to normal (200 rpm in my case). The voltage reading initially rose to a steady value and stays there as long as the load is kept the same. If the steady load is too large, the voltage saturates at 7.5 V and the motor can't compensate to the load.

Explore the effect of varying K_p

When high K_p of 0.2 is entered, indeed, the settling time of the velocity is smaller, but with oscillations. While when low K_p of 0.05 is entered, the settling time is longer but there are less oscillations to the response. This confirms the transfer function parameters described in the figure below, where K_p is directly proportional with the damping ratio.

Explore the effect of varying K_i

On the other hand, when low K_i of 1 is entered, the settling time is quick, but with oscillations. Entering high K_i of 10 makes the settling time longer but with less oscillations on the VDAout. This confirms the transfer function parameters described in the figure below, where the square root of K_i is inversely proportional to the damping ratio.

the damping ratio: $\zeta = \frac{K_p}{2}\sqrt{\frac{K}{JK_i}}$

Figure 3: Damping Ratio Equation

With the base parameters, record the control torque and actual velocity responses for vref = -200 rpm to 200 rpm

The figure below describes the measurement versus the theoretical step response of vref change from -200 rpm to 200 rpm. The theoretical step response is calculated by using the step function in MATLAB in which T1 is used for the transfer function to get the velocity and T3 is used for the transfer function to get the torque command.

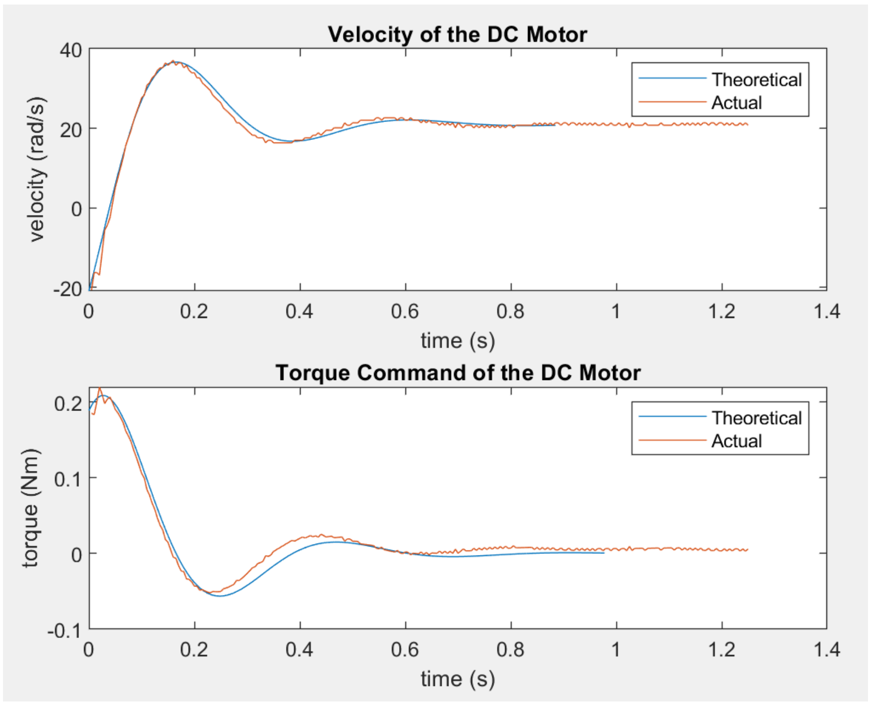

Figure 4: Performance of the Response

$$T_1(s) = \frac{V_{act}(s)}{V_{ref}(s)} = \frac{\tau s + 1}{\frac{s^2}{{\omega_n}^2} + \frac{2\zeta}{\omega_n}s + 1}, \quad T_2(s) = \frac{V_{act}(s)}{T_d(s)} = \frac{sK_d}{\frac{s^2}{{\omega_n}^2} + \frac{2\zeta}{\omega_n}s + 1}, \quad T_3(s) = \frac{U(s)}{V_{ref}(s)} = \frac{sK_u(\tau s + 1)}{\frac{s^2}{{\omega_n}^2} + \frac{2\zeta}{\omega_n}s + 1}$$

Figure 5: Transfer Functions of the System

The measured velocity and torque does not deviate too much from those derived from the theoretical model. The measured data has a wiggly trend throughout the data, and this is called by quantization in both the feedback information (encoder count), and also the quantization in the control output (VDAout). It can be observed that the measured velocity and torque command is slightly delayed from the theoretical curve, which is most likely caused by transport lag. Another takeaway is that the torque command plot is less accurately predicted by the linear model, where we can see higher delay, and that the final steady state value is slightly above zero. This error makes sense since the DC motor probably has higher damping than expected due to aging, which causes the motor to always have to apply additional torque to maintain constant velocity.

With more time, this algorithm can be extended such that the user can input the performance specifications such as the percent overshoot or the settling time, which automatically tunes the PI controller used to control the velocity of the DC motor.