PID Control with Derivative Filtering and Integral Anti-Windup for a DC Servo

Nicanor Quijano and Kevin M. Passino
The Ohio State University
Department of Electrical Engineering
2015 Neil Avenue, Columbus Ohio, 43210

April 8, 2002

Abstract

In this lab you are going to use the main concepts that you saw in the prelab for PID controllers. For that, we use the DC motor for the SRV-02ET to do position control. You are also going to learn about another sensor. Instead of the using the tachometer to measure shaft angular velocity as you did in the Lab # 2, you are going to use an optical encoder to measure shaft angular position.

Contents

1 Introduction

In the pre-lab you studied some basic ideas about PID control. In this lab you are going to apply these ideas to position control for the SRV-02ET DC motor using a proportional controller, then a PID controller with derivative filtering. Finally we are going to design an integral anti-windup scheme for the PID controller.

2 Laboratory Procedures

2.1 Necessary Equipment

- 1 dSPACE software.
- 1 DS1104 interface card.
- 1 Universal Power Module, UPM-2405.
- 1 SRV 02ET servo DC motor.
- 1 Encoder cable.
- 1 To load Cable.
- 1 From D/A Cable.
- 1 Inertial disk.
- 3 72 teeth gears (low gear ratio).

2.2 Connections

We are going to connect the motor using the encoder to perform a position control experiment. For that, follow the next steps:

1. Take the "To Load Cable" and connect one side to the Universal Power Module, and the other to the motor.
2. Take the "From D/A" cable and connect the RCA termination to the analog output 0 in the DS1104 Interface Board (remember that this analog output is # 1 in the software). The connector at the other end goes to the UPM-2405.
3. Take the "Encoder cable" and connect one side to the encoder that is located on the motor (as you saw in the document "Interfacing dSPACE to the Quanser Rotary Series of Experiments (SRV02ET)"), and the other end to the DS1104 Interface Card.
4. Remember that you have to put the motor in the low gear ratio, with its inertia in the top, as you can see in Figure 1.

2.3 Proportional Controller

First, we are going to implement a proportional controller. For that, follow the next steps:

1. Open dSPACE and Matlab. Remember that you have to change the path where you are working (you will work under C:\EE758\LAB3\).
2. You are going to build a model that has one input sequence described in the next step, the controller, and the input from the encoder. This model should be saved as "model3.mdl."
3. Construct one output that has the following characteristics:

Figure 1: Motor with its inertial disk.

- It must be zero between 0 and 10 seconds.
- It must be 45 between 10 and 40 seconds.
- It must be 0 between 40 and 60 seconds.
- The signal must look like the one that you see in Figure 2.
- Since the input wave that you just constructed must be smooth, you can use a rate limiter with a rising slew rate of 1000, and a falling slew rate of -1000 to process it before it is used as a reference input.
- Label the output of the rate limiter as "Input."

4. Now you can construct a simple proportional controller using the gain that you found in the prelab. This gain should be able to be changed by a dSPACE user.

5. Limit the output of the controller between 5 and -5.

6. You have to use the encoder to acquire position of the DC servo shaft. For that you have to use two basic elements in the library of dSPACE that you have in Simulink: the encoder setup block, and the encoder position block. You have to configure these elements, such that you work with channel # 1, and you have a single-ended (TTL) signal type.

7. The output of the encoder position block is in "cycles," and the encoder that we have has 4 *counts/cycle*. Also, the encoder has $\frac{360}{4096}$ *deg/count*. You have to be able to figure out the value of the gain that converts from "cycles" to *deg*, such that you can subtract the reference input that you configured above (in deg.) from this input (*Hint: You can figure out directly the value that you have to use via the previous information, but if you are not able to do that, try to find the number of degrees per count using dSPACE. For that you have to find heuristically the number of counts once you turn the motor shaft one whole revolution by hand.*).

8. Add the "Safety Stop time" block that you built in the last laboratory. Fix its value at 60 seconds.

9. Change whatever you need to change to compile the experiment, and run it using the DS1104 card.

10. Now that you already compiled the model, start a dSPACE experiment, so that the user can see the input sequence that you are applying, the real position that you are obtaining, and finally so the user

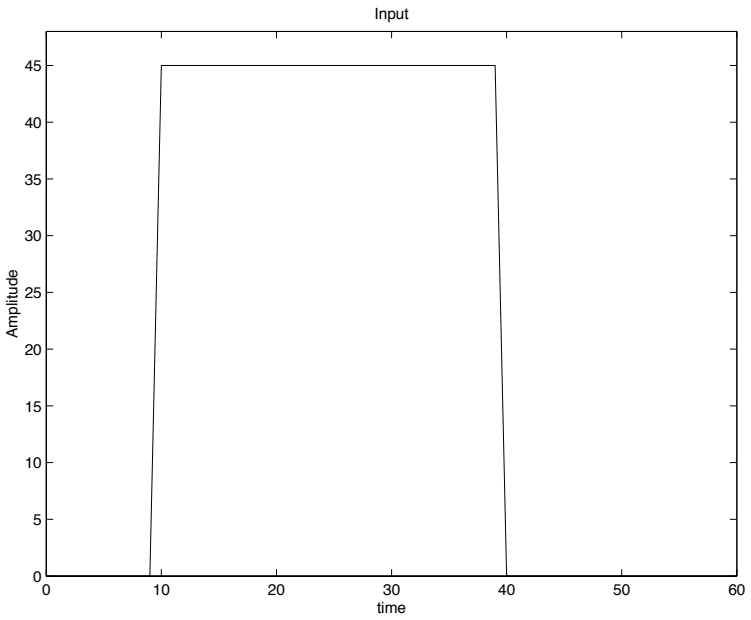

Figure 2: Signal input for the experiment.

can change the proportional gain, and the set point. Finally, save the experiment, so that in the future it can be opened (use your own folder under EE758).

11. Take some closed-loop response data for the gain that you obtained in the prelab. Is this gain appropriate? If not, try to tune the controller to obtain a better result, and save the resulting closed-loop data (*Hint: To save the data, you have to follow the same steps that you use in Lab # 2, or you can just autosave the values using the steps described in the tutorial of Lab # 1.*).

2.4 PID Controller with Derivative Filtering

Now, building on the simple proportional controller, you are going to use a more sophisticated one: a PID controller with "derivative filtering."

1. Use the same input that was described in Section 2.3.
2. Now you are going to build a subsystem called "PID with filter" that has the following characteristics:
   - The block has two inputs: (i) the error that is going to be used by the proportional and the integral part, and (i) the output of the motor that is going to be passed through a filtered derivative to create an approximation to the derivative.
   - One input should be named "Error," the other "Output", and the output must be named "PID output."
   - When the user double clicks this block he/she should be able to change the P, I, and D gains and the value α of the filter (see the prelab). This subsystem should be like the one in Figure 3.
3. Now, using the previous subsystem, and the input wave that you already built, construct a model that you are going to save as "model4.mdl" that controls the position of the motor. For that you have to use the values of P, I, and D gains and α that you found in prelab # 3.
4. After you follow the steps to compile, and build the model, you have to run the experiment, and **IF YOU HEAR A "BUZZ" IN THE MOTOR, TURN THE EXPERIMENT OFF IMMEDIATELY AS THIS CAN DAMAGE THE EQUIPMENT, AND RECONSIDER YOUR DESIGN.** (See the warning in the handout on the DC servo from Quanser.)

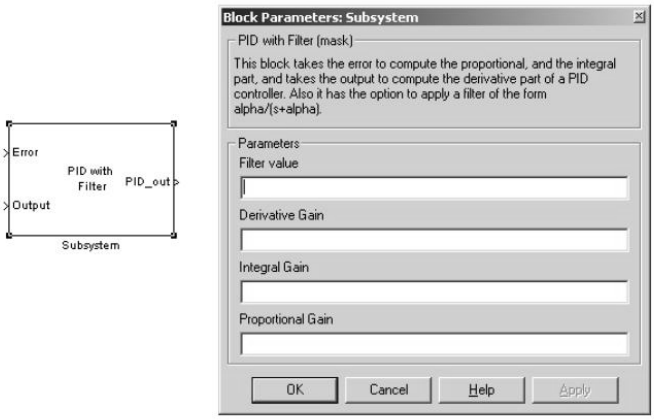

Figure 3: PID controller with derivative filtering subsystem.

5. Build an experiment so that the user can change the values of P, I, and D gains and α. Besides that, the user should be able to see in the same plot the input that you are applying to the system, and the output that you are obtaining. Finally, in a separate plot show the error that you are getting. If you want, you can put two displays so that in one you can see the value in degrees of the DC motor, and in the other you can see the value of the error.

6. Are the values found in the prelab ones that lead to good closed-loop performance? Explain what you found, and save the data, so that you can use it later in the post-lab.

7. Tune the controller, so that you will obtain zero steady state error and a good transient response (you define what you mean by "good"). Which values did you change? Why? Explain in detail the criteria that you used to change the values of P, I, D, and α, if you changed any of these values.

8. Again, save the closed-loop response data that you obtained for different values of the P, I, and D gains and α.

2.5 PID Controller with Derivative Filtering and Integral Anti-Windup

Here, you have to build a PID controller using the same derivative filter that you used before, but instead of using the integral part as you had in the Section 2.4, use the block diagram that you have in the Figure 2.5.B in the handout of the prelab. In this part we will need to use the experiment that you saved in Section 2.4, and a new experiment that you will build right now.

1. Use the same output that you built in Section 2.3.

2. Using practically the same values that you found in Section 2.4 for the P, I, and D gains, and α. Find a value that can saturate the output of the motor. (*Hint: You may want to use a value that is 18 times less than the original value that you had.*)

3. Choose a value for the gain in the anti-windup loop that you built. This value is going to be tuned, once the experiment is running.

4. Compile the new model that you create using the block diagram in Figure 2.5.B in the Appendix A, using the filter for the derivative part.

5. In the control desk of dSPACE, put two plotters that are going to be able to show, in one of them the input and the real position of the motor, and in the other plotter you will put the output that enters in the saturation block. These two plotters will allow you to save these values so that you will plot the results in the post-lab exercises.
6. Put 7 displays that will allow the user to see the values of the parameters of the controller (including the value of the anti-windup gain), plus the actual position, and the error.
7. Save the experiment, and run it. What happens?
8. Now, you are going to open the dSPACE experiment that you saved in the Section 2.4, and you will change the values of the controller, and the saturation blocks, so that you will have the same as in the case of the anti-windup scheme.
9. Run the experiment, and save the data of the input, output, and the output of the "PID with filter" block.
10. Tune the values, until you will observe something interesting, and you can show the difference between the two controllers.

3 Post-Laboratory Exercises

1. What was the value of the gain used at the output of the encoder? How did you find it? Explain.
2. Plot on the same graph the input wave and the response of the actual DC servo for the proportional controller gain value that you found in the prelab.
3. What is the steady state error? Explain what is happening.
4. Now, plot on the same graph the input wave and the response of system for the tuned proportional controller that you found in the lab. What is the value of the gain that you finally obtained in V/deg?
5. What was the criteria that you used to change the proportional value?
6. What is the value of the overshoot? Rise-time?
7. Why do you think that you obtained different values for the proportional controller in theory and in the practice?
8. Explain how you built the subsystem called "PID with filter." Give the code, and clearly state which steps you followed to "mask" the system.
9. Plot on the same graph the input wave and the response of the actual DC servo for the values you found from prelab and the final tuned values you used for the PID controller with derivative filtering.
    - Are the values found in the prelab good ones? Explain.
    - Do you find a steady state error in the response? Why?
    - What is your rise-time and overshoot? How did you tune the controller?
    - What are the values of the gains that you finally picked? Is the value α of the filter affecting the performance of the controller?
10. What are the values for P, I, D, anti-windup gain, saturation, and α that you found in the Section 2.5. Why did you choose these values?
11. Plot in the same graph, the output of your two different controllers developed using the same gains in the Section 2.5. What do you observe? Is that correct?
12. Plot in the same graph the output of the motor for the case where you used a PID with filter, and the anti-windup scheme. Besides that, plot also the output. What do you observe? Give your conclusions.