Lab Session 5: Closed loop control of motor Integral control action PID control

Objectives:

1. Appreciate importance of integral control action and implementation details
2. Understand and circumvent some problems associated with practical limits on the actuator output (actuator saturation)
3. Understand differences between theoretical and practical control implementation

Background knowledge required:

Same as in the Previous lab session. Please revise if you need to.

We observed in last lab that there was huge (~300 counts) steady state error. One of the closed loop ways to take care of steady state error is through integral control action. You basically add a term in control input corresponding to scaled integral of error from point the control is implemented on motor.

Things to do?

1. Use given program and make connections and check if ISR set by PIT initiation is running properly and further encoder reading is coming out properly. If not debug using the Easyscope and correct.
2. Setup a reference position square wave with 1 revolution of gear as amplitude and 10 sec as the time period. This reference is to be followed by your motor actual position.
3. Compute proportional control using Error = Desired position - current position and assign resulting computation to PWMDTYx register to control motor with P control. (see that the newduty is the variable to which you assign this computation). Observe what is happening and see effect of proportional gain on control.
4. Next compute derivative by difference formula Error_dot = Error – Old_Error or using difference of position to get velocity and implement only derivative action computation to PWMDTYx register. Now you should feel resistance only when you try to move the motor. 'Feel' this derivative action for different gains.
5. Next compute the integral control action by integrating the ***error*** by using trapezoidal rule. errorSum is the variable used in the program for the same.
6. Finally use combination of P and D and I actions to get control where error is settled within 100 ms and steady state error is withing 5 counts of EnRead. Can you robustly drive it within 5 counts every time now? Can it be possible to drive it to within 2 counts. Argue the hurdles.
7. Use Ziegler-Nichols procedure, given for example at (you may search your own other ways for tuning if you want)
   http://faculty.mercer.edu/jenkins_he/documents/TuningforPIDControllers.pdf
   to tune the gains and see if the results are satisfactory.
8. ***(challenge)*** Another issue is the continuously increasing control computation when there is saturation on actuator implementation. (Think why??). To overcome the same antiwind up strategy is used. Can you see through the material given below and implement the antiwindup?

Anti-windup in PID control:

The continuous increase of integral control action beyond saturation can be taken care of by an anti-wind up strategy. A simple antiwindup strategy would be as follows (from Book: Astrom KJ "Control System Design" 2002):

Chapter 6. PID Control

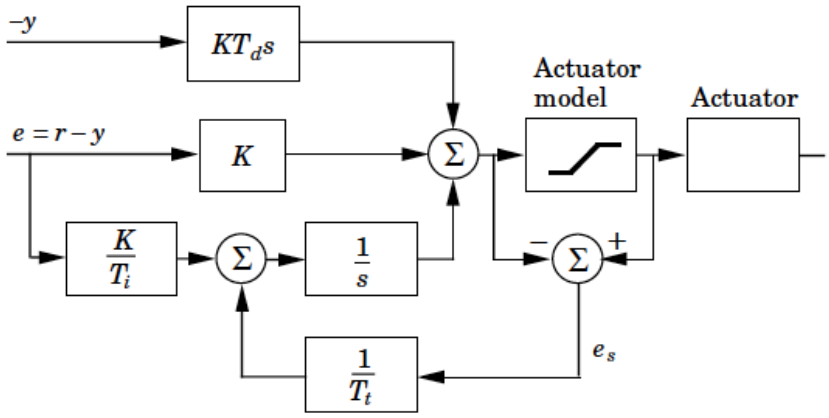

Figure 6.7 Controller with anti-windup where the actuator output is estimated from a mathematical model.

Implement this to see that your steady state error is now lower even with low values of Kp.

You may use the following if you would like to record and see through the data more closely.

Recording the real time data in file:

After compiling the program and running the motor, write the command "cf log_command.txt" in the command window. It is assumed that the file log_command.txt (written below) is placed in project directory (same place where *.mcp of the project sits).

Wait for some time till the log file is written. Now open the log file and try to analyze the results.

Log file to be separately created:

Create a text file named 'log_command.txt' in the operating directory. Type the following in 'log_command.txt' :

```
DEFINE loop = 0
FOR loop = 1..10000,1
fprintf(log2.txt,"%d %d\n", count, En_Read)
ENDFOR
```

In log_command.txt, number 10000 indicates the number of times the logging has to be done, counter and En_Read are variable names (which are to be logged). Each reading is printed 16 times in this file for some reason. Use matlab to import and remove multiple readings and to observe the data carefully. Counter indicates the number of cycles passed before the next data is printed. Develop program to process data to finally get smoothly varying motor angle

in rad as a function of time. You may need to use calibration in part 1. Show the data to TA and verify.