CS148
Building Intelligent Robots

Week 4 : PID Control *Out: 19 Feb 2001*

Preliminary Tasks *before 28 Feb 2001, 1pm*

Before reading this project you should have read the handouts related to PID control that were handed out in class.

In this project you will implement PID control (actually using just the proportional component) with a robot. The goal is to have the robot move in a straight line with a constant speed, which is not as trivial as it might seem since the robot will have to deal with uneven terrain.

PID Review

PID is a term borrowed from the world of engineering–control engineering, dynamical systems etc. It stands for **P**roportional **I**ntegral **D**erivative control (after the formula that uses these components to calculate the control parameter) and is used in a wide variety of real life situations — cruise control in cars is an example that comes to mind. There are several terms that are useful when talking about PID. First there is the process on which we are going to be applying PID control. With the process comes the Process Variable($PV(t)$) which is generally a variable within the process which varies continously over time t. The aim of PID control is to control this PV (we may omit the t parameter since it is implied) so that its value is as close to another chosen value called the set point($SP(t)$) as possible. The error at time t is denoted by $e(t) = SP(t) - PV(t)$. PID control receives its name from the formula:

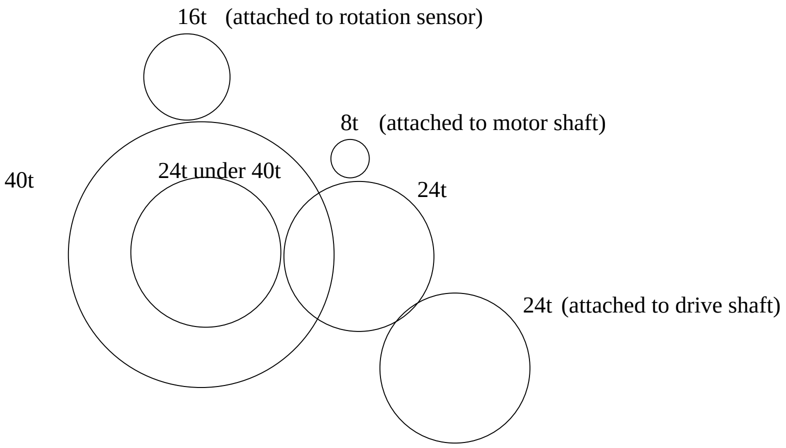
$$input(t) = input(t-1) + K\left[e(t) + \frac{1}{K_{integral}}\int_0^t e(t)\,\mathrm{d}t + \frac{1}{K_{derivative}}\frac{\mathrm{d}}{\mathrm{d}t}PV(t)\right]$$

which is used to control the PV and bring it closer to the SV. The *input* is applied to the PV. The constants K, the proportional constant, $K_{integral}$, the itegral constant, and $K_{derivative}$, the derivative constant are used to determine the extent to which the three different components affect the *input*. For example for some applications the integral component might be needed to a greater extent to control the PV effectively. In this case the value of $K_{integral}$ in the formula above would be comparatively less than another application where the other components are more important.

Project 4: PID Control *due 28 Feb 2001, noon*

Step 1

You have to build the robot. A model for the robot you are supposed to build will be provided. Please be sure to have the correct gear ratios. A schematic diagram for the gears is provided below (you'll probably have to look at the robot anyways):



Step 2

We will choose the velocity of the robot as our PV that we want to control. Now we have to choose a SP, and we will have to determine this before we start. But what does it mean to have a SP of 10? Nothing really if we don't have units for it. We make *clicks/sec* as our unit for velocity where click refers to one reading of the rotation sensor. This makes sense because the only way we can accurately measure the velocity while the robot is running is by using the rotation sensors on it. Use the formula $2\pi r$, where r is the raduis of the robot's wheel, and the gear ratio (between the drive shaft and the rotation sensor), to calculate the distance travelled for one rotation sensor click.

Step 3

What value can we now choose for a SP, obviously not one that is too high or too low i.e. one that cannot be attained. The best way to determine a suitable value, is to make a trial run of the robot. Let the robot travel a distance of say a meter-100cm and record the time it takes. This will give you a velocity that the robot can maintain. Take this value (unit cm/sec) and multiply it with the value you got in the previous step (unit clicks/cm) to get a value in clicks/sec. This is your target SP. Be sure to do the velocity measuring run at a meduim power level. You do not want the power level to be too high because then the robot will not be able to increase the power level to maintain the same velocity when challenged with a ramp.

Step 4

You are given code for the tankbot which you can copy from:

```
cp /course/cs148/labs/4pid/pid.nqc <your directory>
```

Read the code and try to undertand the overall layout. In particular the code for the PID control is implemented by a formula similar to:

$$leftpower(t) = leftpower(t-1) + K_{prop}(SP - PV(t-1))$$

Replace the $TARGET\_VELOCITY$ with the SP calculated in the previous step.

Step 5

As a further adjustment you may now have to tweak the proportional constant.
Now why might we need to tweak the proportional constant? Consider the different scenarios shown in the graphs below:

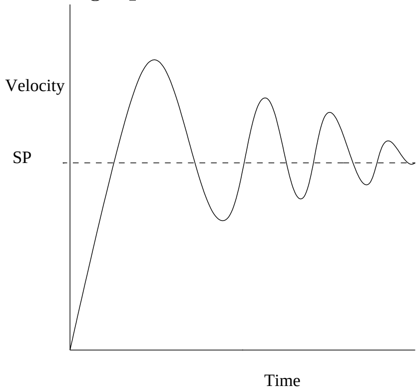


Figure 1

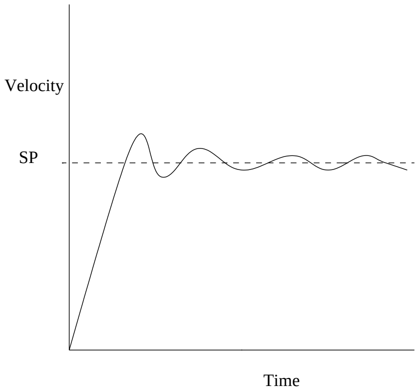


Figure 2

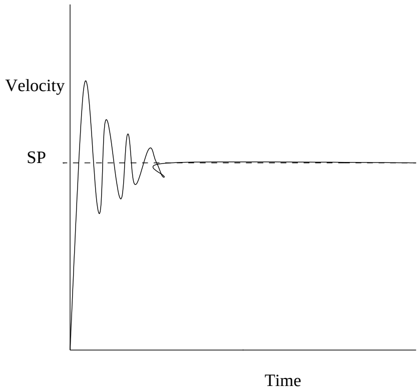


Figure 3

In the first figure the velocity fluctuates wildly for a long time before settling on to the SP. This is clearly not desirable. In the second figure the velocity does not fluctuate that wildly but it takes a long time to settle down and in the third the velocity fluctuates wildly but settles in a short time. We want your robot to display, ideally, behavior which could cause it to quickly reach the SP without much deviation. Practically it will probably be somewhere between figure2 and figure3. The way to manipulate the graphs is by adjusting the proportionality constant. You can narrow your search down to the perfect proportinality constant by doing a kind of a binary search. Start with a large extremes of values (a very large value and a very low value) and keep decreasing the difference between the values till the two extreme values become the same. That will be your desired proportionality constant.
How will you know that you have reached a suitable SP? You have to add code to the existing code to add velocity values (rotation clicks) to the datalog after suitable intervals (use a timer for this). You can then graph these values in Excel or any other spreadsheet program you have to obtain graphs like the ones in figs 1-3. If you have a graph that reaches the SP fast and without much fluctuation, you know you are done.

Step 6

You should now add a separate integral controller in addition to the pid controller that you have already written. *The integral controller will have to control the difference in speed between the left* *and right treads*. You will need to write the code yourself. For minimal extra credit you can also add functionality to the code which allows the robot to turn.

Step 7

Your paper handin should address the following questions:

- A general description of how you went about doing stuff and what problems you faced. This does not have to be an essay. A short description will suffice.
- How did you go about figuring out what the constant values for the integral control should be?
- How did you test your bot with both integral and proportional control and how close to expectations was the velocity. Can you attribute the error to some external factor?

50% of the grade will be on the quality of the analysis in your report. The other 50% will be on functionality and code.