Local Control Using the PID Algorithm

Experimental Exercises Using the PIDSim Program – Updated 12/11/03

The PIDSim program is an educational simulation that demonstrates the characteristics of the PID control algorithm. The software is written for Windows 95/98/NT computers and can be downloaded from our web site at: *http://www.usbr.gov/pmts/hydraulics_lab/software/*

The program simulates a very simple flow situation consisting of a submerged turnout pipe and two pools separated by a gate. A constant inflow enters the upstream pool (left hand side of the screen), and a fixed overflow weir controls the flowrate out of the downstream pool. The turnout consists of a submerged orifice in the upstream pool. The gate separating the two pools can be either an overflow gate or an undershot gate.

The gate can be operated manually, or a PID algorithm can be used to control the gate. Three control objectives are possible, upstream water level, downstream water level, or turnout discharge. The P, I, and D coefficients can be varied to provide the user with a feel for how each coefficient affects the controller action. In addition, gate and water level deadbands can be set, noise can be introduced to the water level and discharge measurements, and exponential filters can be applied to the sensor readings. The size of the various pools, gates, weirs, etc. can be adjusted, although the simulation is not intended to be used to accurately reproduce a real-world control problem. The simulation does not consider the effects of waves or canal dynamics, such as the travel time required for a change at one end of a pool to propagate to the other end of the pool. In fact, pool sizes are specified only in terms of the surface area of the pool.

To start the simulation, run PIDSim.EXE from Windows. The simulation begins immediately. The lower display panel shows an elevation schematic of the canal. The two upper panels plot the water levels, gate positions, and turnout discharge as a function of time. The controls on the screen can be used to select control modes, change tuning constants, and activate different control modes or other features.

The initial settings of the program are as follows:

TARGETS:

Upstream Water Level	0.8
Downstream Water Level	0.4
Turnout Q	0.13

The NO AUTOMATIC CONTROL option should be selected

Kp, Ki, and Kd = 0

Noise is disabled
Filtered sensors are disabled
Deadbands are set to 0

Simulation Time Step = 5 sec
Control Interval = 5 sec
Graph Time Span = 1000 sec
Inflow = 0.5 cfs
Pool1Area = 1000 ft^2
Pool2Area = 1000 ft^2

Max Gate Movement Rate = 0.02 ft/sec
Gate Width = 2 ft
Endsill Weir Height = 0.1 ft
Endsill Weir Length = 1 ft

Under Canal Layout, Overshot Gate should be selected

These initial settings are somewhat unrealistic, because the control interval is only 5 seconds (a new position for the gate is being computed every 5 seconds using the PID equation). However, these settings demonstrate the behavior of the PID algorithm very dramatically. We'll use these settings initially, then try a more realistic control interval of 60 seconds later on.

1) Begin by just manually moving the gate up and down to see that the simulation of the gate hydraulics works as you expect it to.
2) Turn on the PID control algorithm by clicking on the Upstream Water Level option in the Targets area of the screen.
3) Set K_p=1 and leave K_i and K_d set at zero. Try setting K_p to larger values. Following each change, press the Reset Gate to Zero button to introduce a disturbance into the system. *Note that following each change the gate position will increase and stabilize at a higher setting, but the target water level will not be reached, (although as K_p is increased it comes pretty close!). This is due to the fact that the entire gate setting is coming from the Kp*error term, and if the error ever reaches zero (i.e., we've reached the target water level), then the gate position would be zero. Obviously, a gate position of zero will not produce the target water level, so it is impossible to reach the target using only the proportional term.*
4) At a K_p value of about 110, the gate begins to oscillate continuously. This value of K_p is called the ultimate gain. The ultimate gain and the period of oscillation at this condition can be used in the Zeigler-Nichols tuning method to obtain a first estimate of good control parameters for a given system. The period of oscillation is approximately 20 seconds (you can determine this by pausing the simulation and counting the number of peaks on the 1000-second graph of the gate position).
5) Now set K_p=0 and K_i = 0.01. This is the purely integral control mode, which will reach the target value, but also overshoots it. Note that it takes about 2500 seconds (42 minutes) for the gate motion to stabilize when starting from a gate position of zero. This seems like a pretty sluggish response for this small system.
6) Now try increasing K_i to get a faster response. Reset the gate position to zero each time. *Note that this an unrealistic operation, but it makes the effect of changing K_i more apparent*. You will see that increasing K_i causes the system to initially reach the target faster, but the overshoot and oscillation are more severe. This is clearly not a desirable operating condition.
7) Now try setting K_p=1 and K_i=0.2. Try adjusting K_i and K_p to get a rapid response without undesirable oscillation. This is a PI controller, which is essentially the same thing as the P+PR controller or the EL-FLO plus reset controller (except that EL-FLO also includes a filter element on the water level sensor). What happens if you make K_i or K_p too large?

Now let's try some more realistic settings. Change the control interval to 60 seconds.

8) Set all of the control constants to zero, then gradually increase K_p. Find the ultimate gain and the period of oscillation.

You can experiment further by changing the pool and gate characteristics, enabling noise, using the exponential filters to damp out noise in the sensor readings, setting gate and water level deadbands, or trying different control modes, such as the derivative mode (K_d)

Zeigler-Nichols Tuning Method

The Zeigler-Nichols method for tuning PID controllers is presented in most control-engineering textbooks. The basic steps required to implement the Zeigler-Nichols method are as follows:

1. Set the integral and derivative coefficients to zero.
2. Gradually increase the proportional coefficient from zero until the system just begins to oscillate continuously. The proportional coefficient at this point is called the "ultimate gain", P_U. The period of oscillation at this point is called the "ultimate period", T_U. The ultimate period should be expressed in the same units used internally in the PID algorithm for calculations involving the time step, Dt. (The PIDSim program uses seconds).
3. The Zeigler-Nichols suggested controller settings are as follows:

Control Action	Desired Performance	K_P	K_I	K_D
P	¼ decay	0.5 P_U		
PI	¼ decay	0.45 P_U	0.54 (P_U/T_U)	
PID	¼ decay	0.6 P_U	1.2 (P_U/T_U)	0.075 (P_U*T_U)
PID	Some overshoot	0.33 P_U	0.66 (P_U/T_U)	0.11 (P_U*T_U)
PID	No overshoot	0.2 P_U	0.606 (P_U/T_U)	0.10 (P_U*T_U)

For our first example (5 second control interval) using the PIDSim program, the ultimate gain was P_U=110, and the ultimate period was T_U = 20. Using these values we can fill in the table.

Control Action	Desired Performance	K_P	K_I	K_D
P	¼ decay	55		
PI	¼ decay	49.5	2.97	
PID	¼ decay	66	6.6	165
PID	Some overshoot	39.6	3.63	242
PID	No overshoot	24	3.33	220

For our second example (60 second control interval) using the PIDSim program, the ultimate gain was P_U = 9, and the ultimate period was T_U = 120. Using these values we can fill in the table.

Control Action	Desired Performance	K_P	K_I	K_D
P	¼ decay	4.5		
PI	¼ decay	4.05	0.0405	
PID	¼ decay	5.4	0.09	81
PID	Some overshoot	3	0.0495	108
PID	No overshoot	1.8	0.04545	118.8

Plug these numbers in and see how they work!