Auto-Tune PID Controller
Final Project Report

Project Sponsor:

Ben Bales (TA)

Introduction:

The purpose of this project is to create a VI that can automatically tune the speed and position PID constants with a motor plant program. The VI allows users to input requirements to adjust the performance of the motor.

Method:

Two methods are used to tune PID gains. The first one simulates the general manual tuning method (*Manual Method VI*), in which the controller tunes Kp first until oscillation, then tunes Ki until oscillation, and repeat decreasing and increasing Kp and Ki until the end condition is met. The end condition can be maximum overshoot or settling time.

The second method is a type of optimization searching (*Optimizing Method VI*). The controller first combines the overshoot and settling time and multiplies some factors, then uses this sum as a reference to optimize gains. When plotting the sum and corresponding gains, part of the plot will be a parabola. The controller will first test the 3 gains specified in the main VI as the range, then test the middle points of the 3 gains. Thus, these 5 gains will have 3 combinations of 3 different gains. If one combination appears to be V shape (lower high, middle low, upper high), it means the optimal point is located at this range. The controller moves to this combination and iterates optimization again.

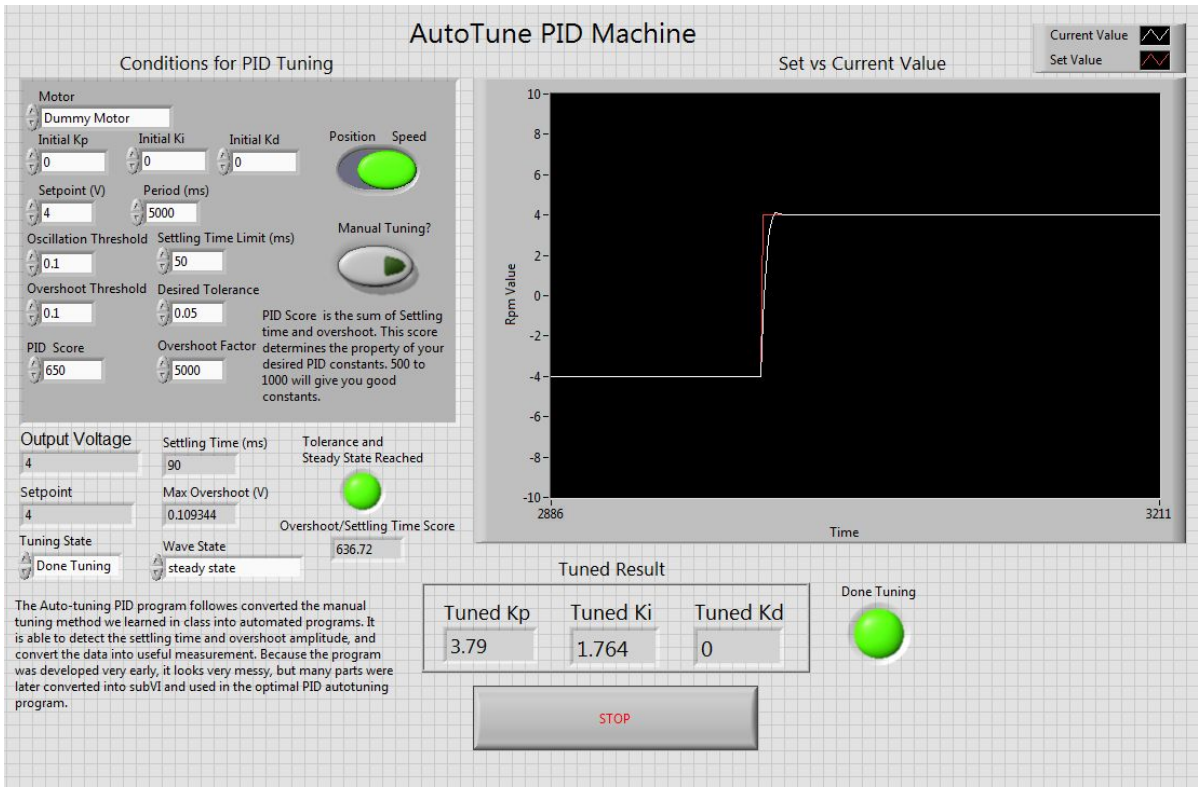
Since the controller needs to switch between speed and position, and optimization needs to switch among tuning Kp, Ki, and Kd. The general structure used is State Machine. A queue is

used in the *Optimizing Method VI* to feed in user inputs, but it doesn't really show the full function of Producer-Consumer.

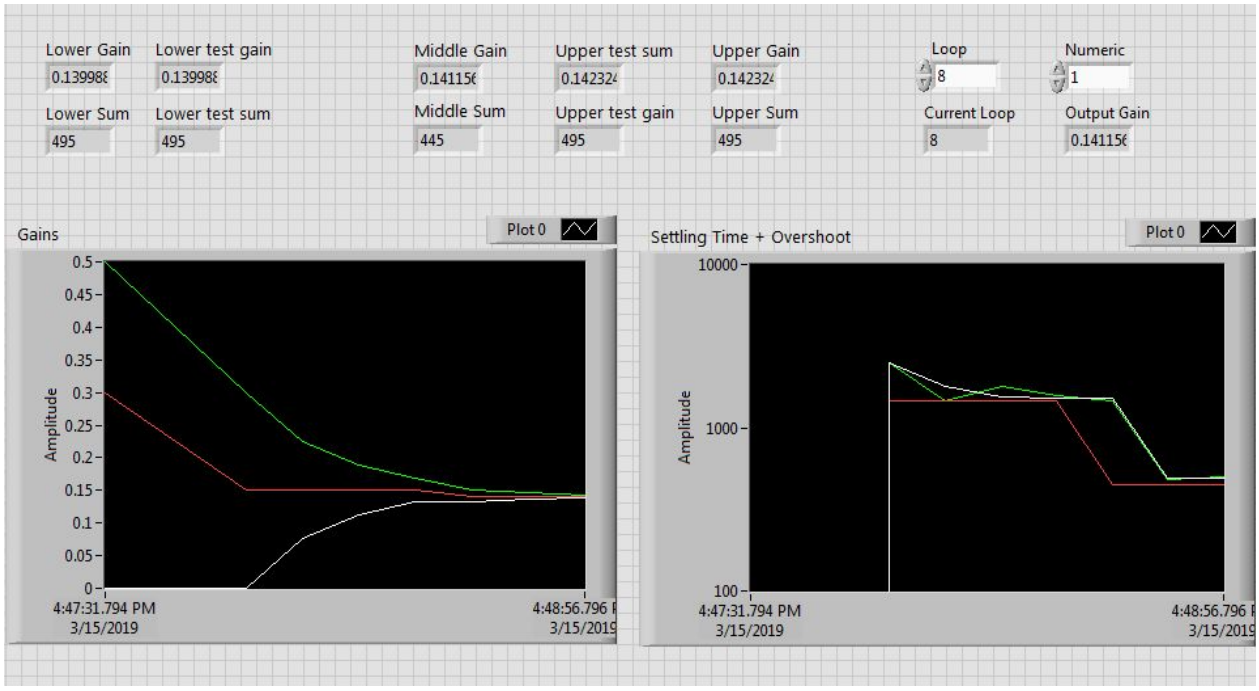
Instruction to Use:

In both VIs, there are speed and position tuning options. Users first need to put in the initial gains or a range of gain they want. Since the program will generate a square wave to the motor plant, user also needs to specify the voltage setpoint (amplitude) and period of the wave. User also needs to input the oscillation and overshoot threshold since they can decide if there really is oscillation or overshoot. Once it's running, the LED indicator will be on if all the requirements are met, then the tuning process will stop.

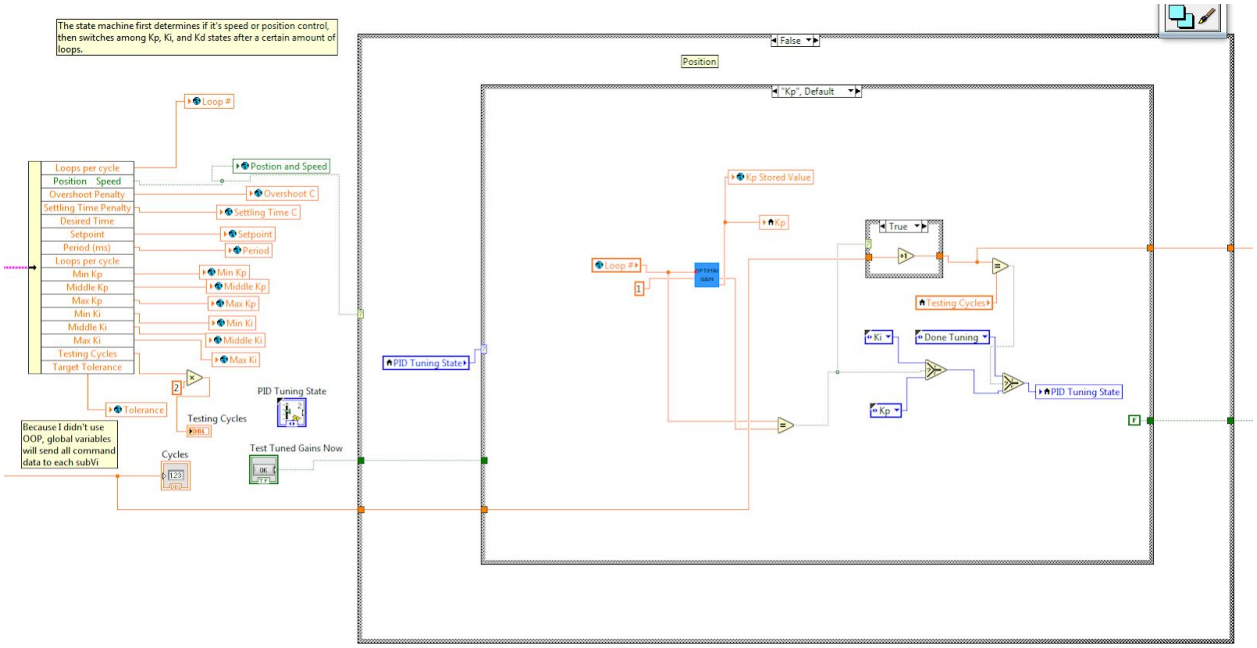
One thing is different between two VIs is that in *Optimizing Method VI*, user must go to a subVI called *Optimizing for finding V* to view the waveform chart, and the main VI cannot directly show the chart (I will explain this below in Problems).



The front panel of *Manual Method VI*



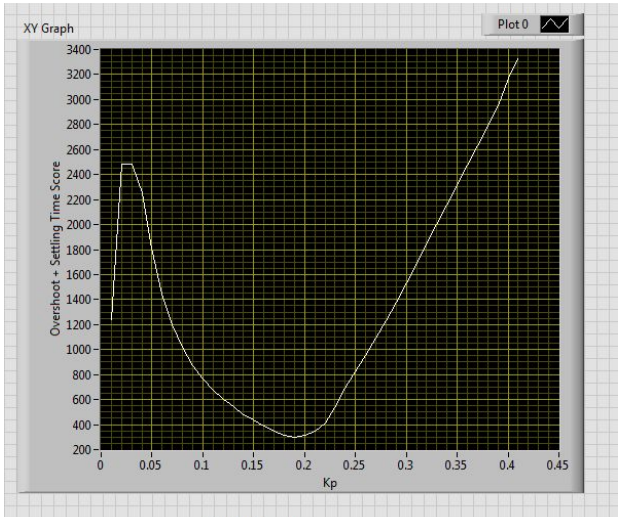
The front panel of *Optimizing for finding V*
(Here the gain converges to a number while settling time + overshoot is decreasing)



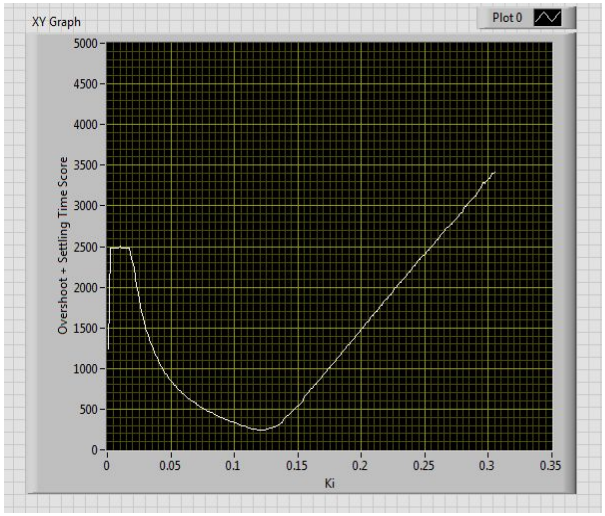
The State Machine in *Optimizing Method VI*

Known Problems:

Major Problems: for the *Manual Method VI*, the controller is changing gains while motor is running. Although it's more efficient in term of total runtime, the controller can only decide the max overshoot and settling time after each square wave pulse. Thus, the controller should adjust the gain between each step pulse. However, the *Optimizing Method VI* uses this way, but it made the tuning process become extremely slow since the gain can only change after each cycle (about 8-10 seconds). Its main VI also can't show graph since the only motor plant is running in a SubVI. Another problem is that the *Manual Method VI*, can tune Kp, Ki except for Kd since derivative's trend is too strange to find. the *Optimizing Method VI* can only locate the optimal Kp in position and Ki in speed since they have the parabola feature.



Kp Position vs [Overshoot + Settling Time]



Ki Speed vs [Overshoot + Settling Time]

Minor Problems: In the *Manual Method VI,* the wiring is very messy since it was built to test ideas. Sometimes the process doesn't start with tuning Kp first even it's already initialized. In the *Optimizing Method VI*, too many global variables are used, which slow down the speed. Also, the data is not accessible in the main VI (only for inputting data).

Future Goals:

For *Manual Method VI,* a case of derivative can be developed by examining its trend. However, the *Optimizing Method VI* is the correct path to really do optimization. One way to improve the speed is to use a math model to simulate motor instead of plant program (although plant is more similar to the real motor), which doesn't require time to "run motor." This can also allow the main VI to generate waveform chart. A method like finding the minimum value of parabola may be helpful in some specific cases, but PID is a complex, dynamic system (especially when it has two non-zero gains), and it requires a more advanced skill maybe until grad school.