**Self-balancing Inverted Pendulum Using Intelligently Autotuned PID**

Brian Searle, bsearle@ucsd.edu, a10660812
and
Will Zegers, wzegers@eng.ucsd.edu, a09291938

## 1 Proposal

The goal of our project is to design and implement an inverted pendulum-type system that can autonomously self balance, and possibly even be able to move forward while maintaining balance. To accomplish this, we propose a system that will use control logic, namely PID, to keep the pendulum at an upright (i.e., straight up and down) angle and compensates for the pendulum leaning by moving the platform on which it rests forward and backwards, moving directly back underneath the pendulum.

Furthermore, we propose tuning the PID control using machine learning techniques to constantly "learn" the ideal K parameters for PID and keep the system upright.

## 2 Hardware Components

1) Small mobile platform options
    a) Four-wheeled to balance a separate inverted pendulum system
    b) Two-wheeled, effectively making the vehicle itself the inverted pendulum-type
3) Embedded microcoontroller options
    a) Raspberry Pi 2
    b) Beaglebone
    c) mbed 1768 (simple, but use may be limited)
4) Breadboard
5) Jumper wires
6) Components to provide analog input to encode angle of pendulum
    a) Mid- to high-resolution encoders
    b) Potentiometer
    c) Accelerometer / gyroscope

## 3 Software

Our current, bird's-eye-view of the software specifies the following systems:

1) Sensor-to-system
    a) A-D logic to read and process signal coming in from pendulum angle.
2) PID system to read in the angle (process variable) and update the control variable (forward/backward momentum of platform) to keep system balanced.
3) Driver to move the platform, on input provided by the PID controller
    a) Communication software between microcontroller and platform (exact method - UART, GIPIO, etc. - will depend on selected platform)
4) Machine learning-based tuning algorithm to update PID parameters.

More than likely, the software will be implemented in 1 or two languages catered to the individual systems (e.g., Python for scripting and sensor interfacing logic, MATLab/Numpy for ML, possibly C for lower-level drivers).

## 4 Criteria for Success

Success of the project will be based on two criteria, and a possibly an optional third.

1) Most importantly, the PID controller should be able to respond to the pendulum "falling over" in real-time and quickly enough to keep the system balanced. Degree of success here will be based on the duration that the pendulum system is able to stay upright (which is, ideally, as long as the system is turned and on the PID is able to keep it upright. In this case, duration spent upright will be the quantity we can measure to gauge how well the

system is able to balance itself.

2) Assuming the first criteria can be met and the system can balance upright indefinitely, success of the PID tuning will be based on how quickly the system can converge at an optimal set of parameters that keep the system balanced upright, and minimize oscillation (i.e., wobble).

3) Finally, we'd like the system to be able to do more than just balance in place. Provided the first two criteria can be met successfully, we'd like to be able to push the PID controller further and make it possible to move the system forward and back, while still maintaining balance. Whether this criteria can be met will be based upon how successful we were with meeting the first two, as well as being able to address the additional complexity of being able to send the system moving while still balancing at the same time.

**5 Proposed Experiments**

1) Will need to check movements of mobile platforms and speed for full varying control.
2) Experiment that the PID control mechanism can work on its own with manually tested PID configurations.
**3**) Test if machine learning works to configure the PID parameters using different machine learning techniques. In order to do this the weight at the top of the pendulum will change and we will check to see if the mobile platform can correctly configure the PID controller as well as keep the inverted pendulum balanced.

**6 Deliverables**

- Source code
    1) c code to query and control sensors
    2) script of machine learning algorithms

**7 Proposed Demo and Report**

For the demo we will showcase the auto balancing control of the inverted pendulum by letting it run to show the automatic balance. To display the machine learning code we will change the weights of the inverted pendulum to show the adaptive features of the system and auto learning PID configurations. If the project is fully successful we will then be able to move the drive the inverted pendulum forward and backward without losing balance.