**E105 "Mini Lab": PID Control via Arduino**

ENGR 105: Feedback Control Design
Winter Quarter 2013

**Solutions**

## System setup (5 pts.)

## 1. Proportional control (5 pts.)

a. What do you think is the form of the plant model for the device you are controlling? I.e., what is the general form of the transfer function?

The system is a one-degree-of freedom inverted pendulum. The device is most naturally described in rotational coordinates, but the translation of handle makes a useful coordinate system. There is an inertia ($m$), some friction in the joints and motor (really a nonlinear friction, but can be modeled as linear damping, $b$), and the effect of gravity (which for small angles can be approximated as a stiffness, $k$, where $k > 0$ but there will be a negative in front in the equation of motion).

$$m\ddot{x} + b\dot{x} - kx = f \qquad \frac{X(s)}{F(s)} = \frac{1}{ms^2+bs-k}$$

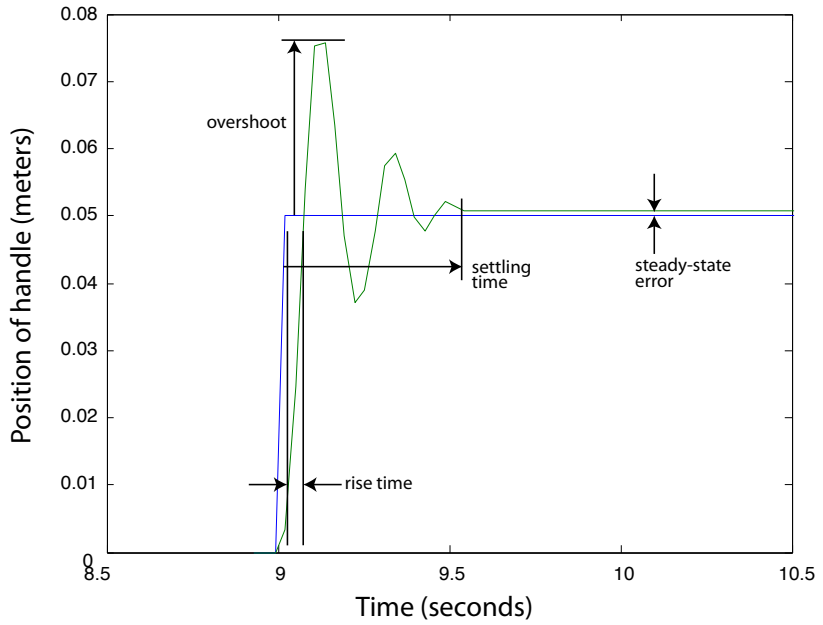
b. Is the plant inherently stable or not? Explain.

It is not, since there is a destabilizing "negative stiffness" effect of gravity. You might not have seen your device fall over if the nonlinear friction in the device was high enough, though.

c. What is the value of `kp` that you selected? (There is not a right answer here; every device is a little different.)

The `kp` gain (in this example) was increased from 10 N/m to 40 N/m.

d. Annotate on your plot (just with arrows, no numbers needed) the rise time, overshoot, settling time, and steady-state error.



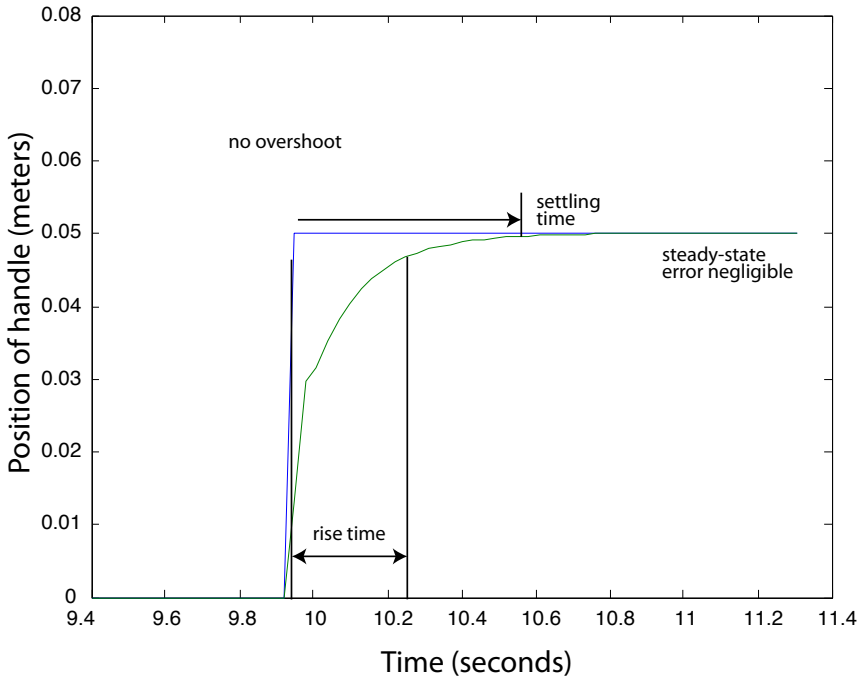

## 2. Proportional-derivative (PD) control (5 pts.)

The new line of code is: `f = kp*e +kd*dedt;`

a. What is the value of `kd` that you selected?

The `kd` gain selected was 5 N-s/m.

b. Annotate on your plot (just with arrows, no numbers needed) the rise time, overshoot, settling time, and steady-state error.



c. What performance metrics did adding derivative control improve? Did anything degrade?

The derivative control decreased overshoot (an improvement) and increased rise time (degradation). There is a decrease in steady-state error, but this is not an expected effect of adding derivative control – this is due to the fact that there is nonlinear friction in the device. Ideally, derivative control should decrease rise time since it adds a zero that should cause the output to try to rise quickly (like the step input). However, the controller does not behave ideally because there is a strong low-pass filter on the velocity signal.

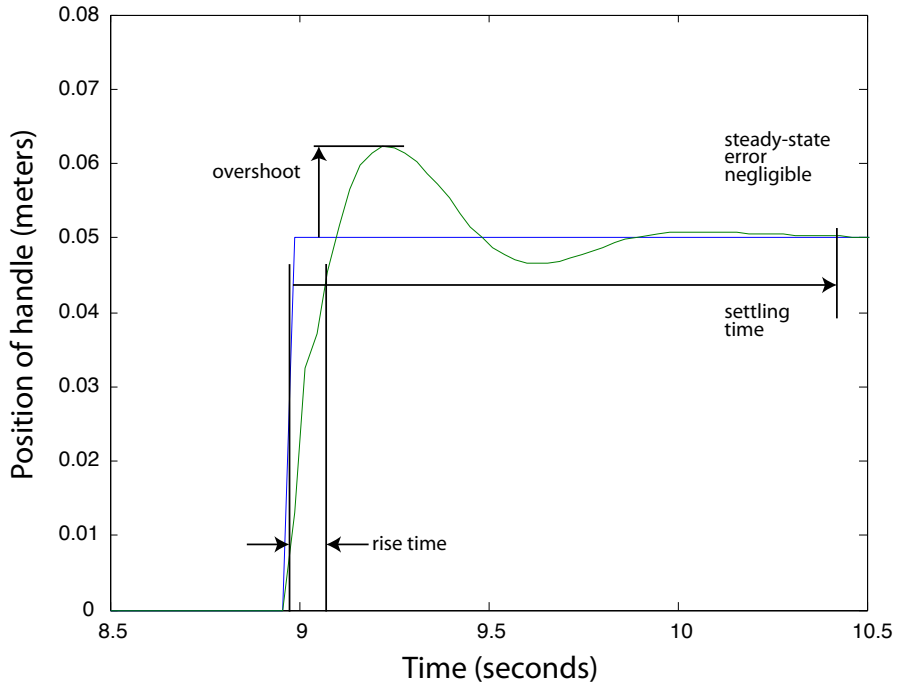
## 3. Proportional-integral-derivative (PID) control (5 pts.)

The new line of code is: `f = kp*e +kd*dedt + ki*eint;`

a. What is the value of `ki` that you selected?

The `ki` gain selected was 0.5 m-s.

b. Annotate on your plot (just with arrows, no numbers needed) the rise time, overshoot, settling time, and steady-state error.




d. What performance metrics did adding integral control improve? Did anything degrade?

The goal of adding integral control is to remove steady-state error. However, even a little too much integral control (a high `ki`), will result in overshoot and oscillations as seen above. A good choice of `ki` should not affect the response much and also remove any steady-state error. (Many of you will not see much steady-state error for PD control alone, though, since most groups used a high enough `kp` to make the steady-state error not visible on these relatively low-resolution plots.)