Ducati Multistrada 1200 PID Cruise Control Analysis

Gregory P. Starr

Professor Emeritus
Department of Mechanical Engineering
University of New Mexico
Albuquerque, New Mexico 87131

Abstract. This paper documents the simplified modeling and analysis of a PID-based cruise control for a 2010 Ducati Multistrada motorcycle. The motorcycle "speed dynamics" are modeled as a first-order system, with parameters determined from experimental observation. It is shown that proportional (P) control action is necessary to give the system some disturbance (*e.g.* hill or headwind) resistance. Integral (I) control action is also necessary to achieve zero steady-state error, but too much I action can cause undesirable behavior. Derivative (D) action is not really necessary for this application. It is recommended that the integral (I) gain be reduced to 50% of its current setting in the Tuneboy cruise control system, with further gain tuning from that point.

1 Introduction

Vehicle "cruise control," or speed regulation systems, are useful for many reasons, which will not be discussed here. The purpose of this document is to examine the analysis of an example cruise control system, and the tuning of its control gains.

It is important to clarify the distinction between the "tuning" of engine parameters (fuel metering, ignition, *etc.*) and the "tuning" of controller parameters. Engine tuning is a discipline unto itself, and is a separate issue from the topic of this document. "Tuning" as applied to control systems typically means the setting of the controller gain parameters. Tuning of a PID controller is often not straightforward, and much literature has been devoted to it. Thankfully, the cruise control application is not one of the more difficult scenarios.

This document will first discuss modeling of the vehicle whose speed is to be controlled, simulation of the vehicle speed in response to a "hill" disturbance, then introduction of a closed-loop PID speed controller, and simulations at various controller gain values.

2 Vehicle Modeling

Since we are interested in controlling the speed of a vehicle, and the throttle setting is the only "input" to the vehicle which affects this variable, we can use a fairly simple model.

Note that SI (metric) units are generally used here, unless otherwise specified. In particular, speed is measured in m/s (meters/second)—which is the SI unit—but later for ease of visualization speed will be converted to MPH (miles/hour) for plotting.

2.1 Free-Body Diagram

A free-body diagram of a simplified vehicle is shown in Figure 1.

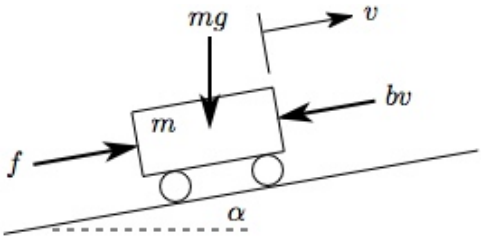

Figure 1: Free-body diagram of vehicle on a hill of angle α.

The nomenclature of Figure 1 is as follows:

m = mass of vehicle (kg)
b = viscous friction coefficient (N-s/m)
v = speed of vehicle (m/s)
f = tractive force on vehicle from engine (N)
g = gravitational constant (m/s^2)
α = angle of road surface (rad)

2.2 Equation of Motion

Applying a Newton-Euler force balance in the direction of the road surface, we obtain the vehicle equation of motion as

$$f - bv - mg\sin\alpha = m\dot{v} \tag{1}$$

Since the angle of the road surface will generally be small, we can employ the "small-angle" approximation $\sin\alpha \approx \alpha$, and equation (1) becomes linear, and can be written with dependent variables on the left and inputs on the right as:

$$\dot{v} + \frac{b}{m}v = \frac{1}{m}f + g\alpha \tag{2}$$

2.3 Vehicle Block Diagram and Transfer Functions

Using standard Laplace Transform techniques, a block diagram representing (2) is shown in Figure 2.

There is one new parameter in Figure 2, and one new variable. The new variable is:

θ_t = twistgrip angle (deg),

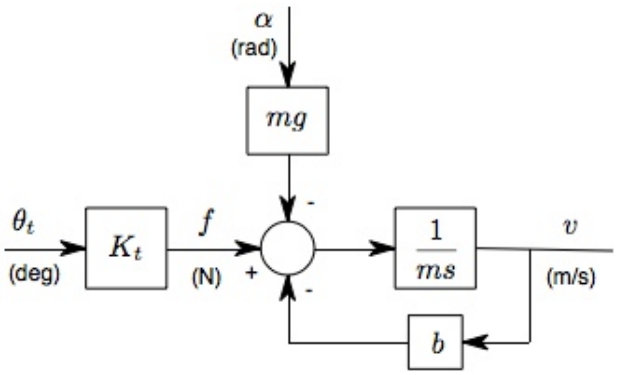

Figure 2: Vehicle block diagram.

and the new parameter is

$$K_t = \text{tractive force gain (N/deg)},$$

Here the non-SI unit of "degree" was used for twistgrip angle, since radians seemed a bit arcane.

Also, the assumption was made that tractive force f is linearly proportional to twistgrip angle θ_t. This ignores all "engine dynamics," but is hopefully adequate for a simplified linear analysis such as this.

Two transfer functions can be extracted from Figure 2: one relating vehicle speed to throttle opening, and one relating vehicle speed to road gradient.

Setting the "hill" input α to zero, one can find the transfer function relating vehicle speed to throttle opening θ_t as

$$\frac{V(s)}{\theta_t(s)} = \frac{K_t}{ms+b} = \frac{K_t/m}{s+b/m} \ \frac{\text{m/s}}{\text{deg}} \tag{3}$$

Likewise, seting the throttle opening θ_t to zero, the transfer function relating vehicle speed to road gradient α is

$$\frac{V(s)}{\alpha(s)} = \frac{-mg}{ms+b} = \frac{-g}{s+b/m} \ \frac{\text{m/s}}{\text{rad}} \tag{4}$$

Simulations of motorcycle speed behavior in response to both of these inputs will be shown in Section ???

2.4 Numerical Parameter Values

For design and simulation, we must have numerical values for all parameters. Some numerical parameter values were found directly, while others were found *via* experimentation. Each will be discussed in turn.

1. Mass m. This is easy—the motorcycle weight is 520 lb + rider weight of 160 lb results in a total mass of
$$\boxed{m = 310 \text{ kg}} \tag{5}$$
2. Viscous friction coefficient b. This parameter is primarily air drag, with some bearing friction added. I performed a "coast-down" test (with clutch disengaged) and estimated that it takes 5 seconds to decrease from 70 mph to 60 mph.

Since the motorcycle equation of motion is a linear first-order differential equation, when "coasting," down from a speed of v_o, its solution will be of the form

$$v(t) = v_o e^{-t/\tau}, \tag{6}$$

where τ is the first-order "time constant," and from the denominator of either transfer function (3) or (4) is given by

$$\tau = \frac{m}{b} \text{ sec} \tag{7}$$

Letting $v_o = 70$ mph (here speed units don't matter), we know that at $t = 5$ the speed is 60 mph, thus

$$v(5) = 70e^{-5/\tau} = 60 \implies e^{-5/\tau} = 0.86 \tag{8}$$

Taking the natural logarithm of both sides reveals that

$$-\frac{5}{\tau} = -0.15 \implies \tau = 32 \text{ sec} \tag{9}$$

This means that is you started coasting from 100 mph, at the end of 32 seconds you would be down to 63% of the way to zero, or 37 mph. Doesn't sound too unreasonable...

And finally, knowing τ we can find viscous friction coefficient b from (7):

$$\boxed{b = \frac{m}{\tau} = \frac{310}{32} = 9.7 \frac{\text{N-s}}{\text{m}}} \tag{10}$$

3. Throttle gain K_t. I estimated that a 30° throttle opening (about half-throttle) causes the motorcycle speed to increase from 60 mph to 70 mph in about 2 sec. Now the vehicle speed in response to throttle opening has an exponential behavior, just like equation (6). With a little iteration (details omitted here), I approximated the throttle gain to be

$$\boxed{K_t = 24 \frac{\text{N}}{\text{deg}}} \tag{11}$$

That relates the tractive force (N) to twistgrip opening (deg). Hard to "visualize" if that's correct, but it's the best I could do.

4. Gravitation constant g. Well, *everybody* knows that

$$\boxed{g = 9.8 \frac{\text{m}}{\text{s}^2}} \tag{12}$$

That completes the numerical parameter determination.

2.5 Effect of Road Grade on Uncontrolled Vehicle

Before we start looking at speed controllers, it is worthwhile to simulate the effect of a hill on the vehicle speed. Everyone knows that if you approach a hill at, say, 70 mph, and keep the throttle setting constant, that once on the hill you slow down. Our model should predict that behavior.

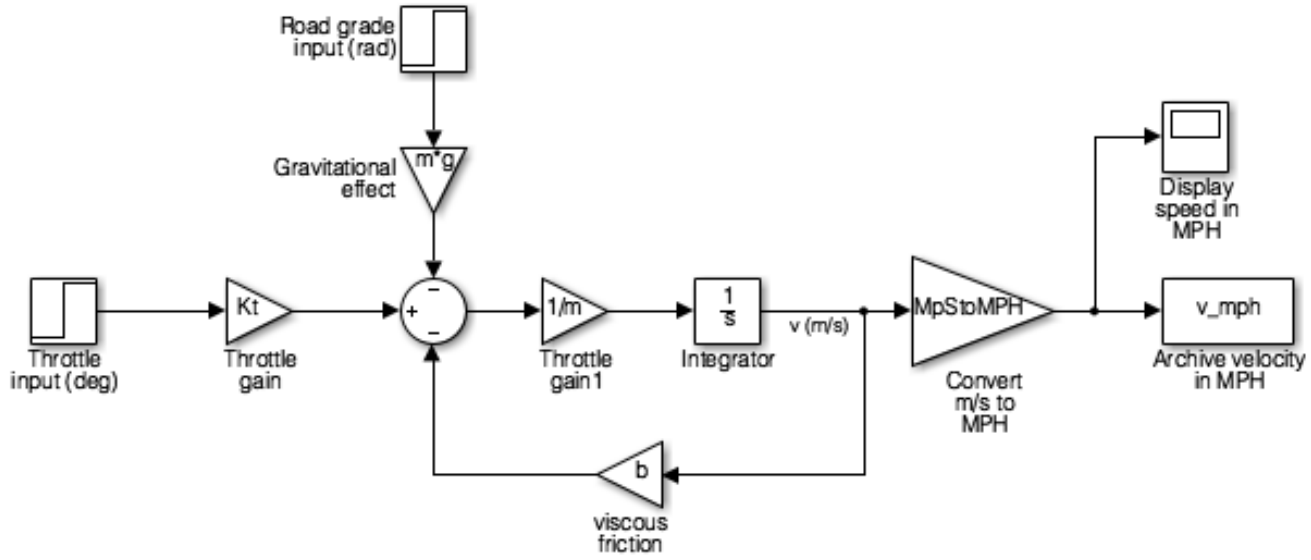

Figure 3: Vehicle block diagram in Simulink format.

Given the numerical values of all parameters, we can use the MATLAB computational dynamic modeling software package to perform a simulation. Actually, we'll use the Simulink package to allow the use of block diagram modeling.

The vehicle block diagram of Figure 2 cast into the Simulink format is shown in Figure 3.

The two block diagrams are really the same, although the Simulink version does have a conversion from m/s to MPH, along with some archiving blocks. Also, the mass scaling and the integrator were separated to allow an initial condition (70 mph in m/s) to be imposed on the integrator.

Simulink has the ability to define "subsystems," encapsulating several blocks and signals into larger subsystems for clarity. Such as "consolidated" model of the vehicle is shown in Figure 4.

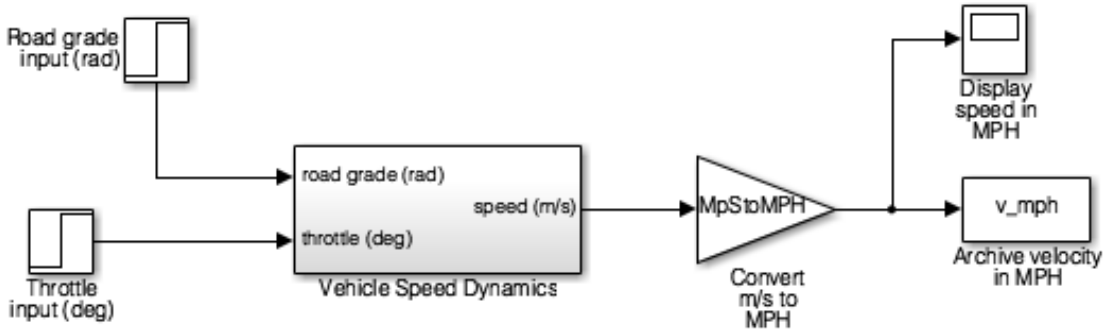

Figure 4: Simulink vehicle model using subsystem.

As before, the vehicle has two inputs: throttle setting (deg) and road grade angle (rad). The "Vehicle Speed Dynamics" output is speed in m/s, but a conversion block to MPH is provided so we can visualize the response easier.

Figure 5 shows the effect of a 5% grade on the vehicle speed, with throttle setting and no cruise control.

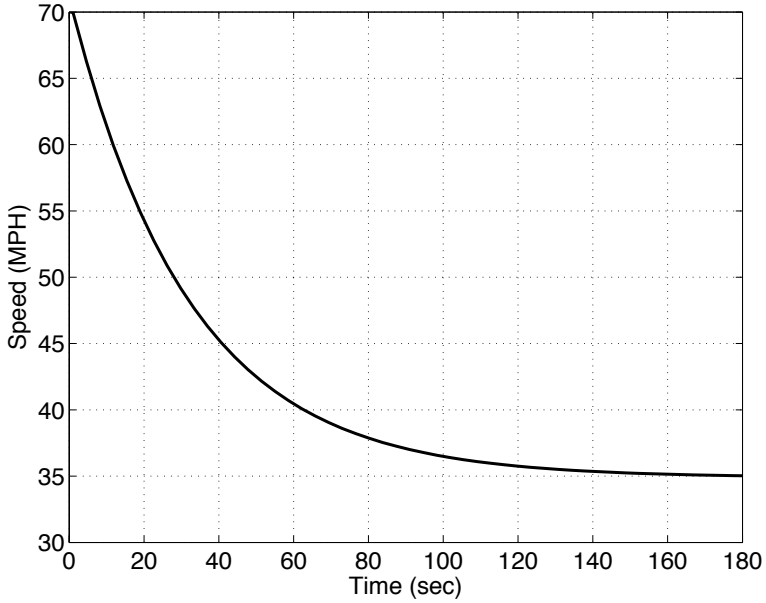

Figure 5: Effect of 5% grade, no cruise control, throttle setting constant.

Although the speed ultimately slows down to 35 mph, note that at 10 seconds the speed has decreased from 70 mph to about 60 mph. That actually seems reasonable. So perhaps our vehicle model is not too bad.

This concludes the vehicle modeling. Now it is time to add the cruise control.

3 Development of PID-Based Cruise Control

This is really the heart of this work. However, to develop a controller for a given process, one must first have a model of that process. That is what we just accomplished. By the way, the thing you're trying to control is called the *plant*. That's just standard terminology.

3.1 What is PID Control Action?

You can't hang around automatic control systems very long before you hear of the ubiquitous "PID" controller. Following is a very brief explanation.

You can't control a variable (quantity) unless you can measure it. Given the measurement, you can subtract the measurement from the *desired* value of the quantity to form an error e. The plant has some input; here it is throttle angle θ_t.

A PID controller is a "three-mode" controller, with the three modes being P, I, and D. Here's what they are:

1. **P action.** P stands for proportional, and the control command to the plant is *proportional* to the error. The proportionality constant (P gain) is K_p. In mathematical form, P action is defined by

$$\theta_t(t) = K_p e(t) \tag{13}$$

Proportional control is what *you* do when you see the speed drop on a hill: you roll on the throttle a little more. Controllers *always* have P action.

2. **I action.** This is a little harder to explain. The I stands for "integral" action, and integral means there is an integrator. An integrator is kind of like a "cumulative memory," it keeps track of what has happened in the past. Integral action is necessary in a cruise control to "reset" the speed back to the desired value when there is a disturbance (we'll see that in a little while). The *bad* part of integral action is that it introduces a 90° phase lag (can't explain that here, but it's bad), which de-stabilizes the system. Above all, you want a control system to be *STABLE.* A little integral action is a good thing; too much can be catastrophic. Mathematically, I action is given by

$$\theta_t(t) = K_i \int e(t)\, dt \tag{14}$$

where the swoopy symbol is "math" for integral...K_i is the I or integral gain.

3. **D action.** D stands for "derivative," and this means that the throttle setting depends on the "rate of change" of the error. Sometimes D action can add damping, and improve stability; sometimes it can speed up the response to changes in the "desired" input. In the case of the cruise control it doesn't really do much. Mathematically,

$$\theta_t = K_d \frac{d}{dt} e(t) = K_d \dot{e}(t) \tag{15}$$

where the "dot" means time differentiation, and K_d is the derivative gain.

3.2 Simulink Block Diagram of PID Cruise Control

Figure 6 shows a PID controller added to the Simulink block diagram model of the plant we had previously.

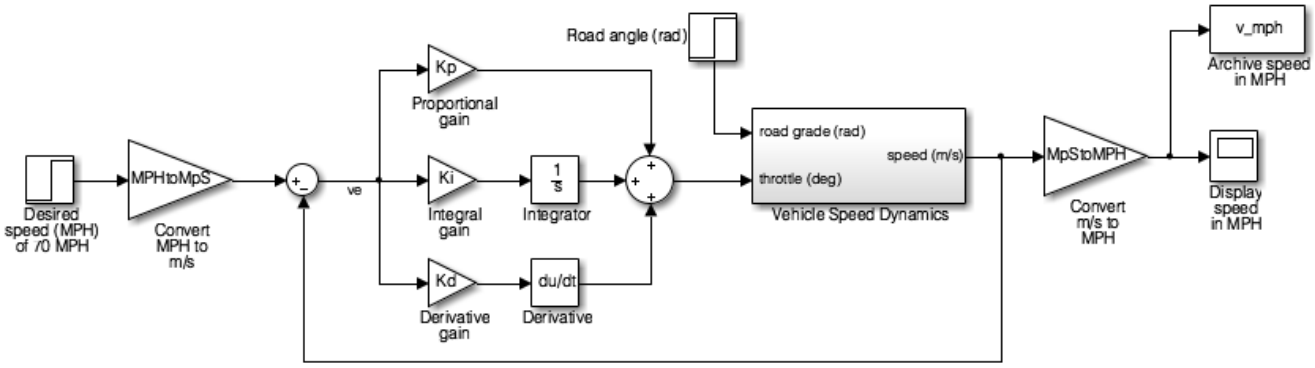

Figure 6: Simulink model of PID Controller added to Vehicle.

The PID controller is expressed *via* the three parallel signal paths just preceding the "Vehicle Speed Dynamics." Also note the "feedback path" with a measurement of the speed being "fed back" and subtracted from the desired speed. Since we subtract, this is "negative feedback;" this is now a *closed-loop system.*

Addition of closed-loop control can make things better—or worse. It's all in how you set the parameters. The three "PID Gains:" K_p, K_i, and K_d are visible in the three parallel signal paths.

As before, we can use Simulink "subsystems" to clean things up a bit—a slightly tidier model is shown in Figure 7.

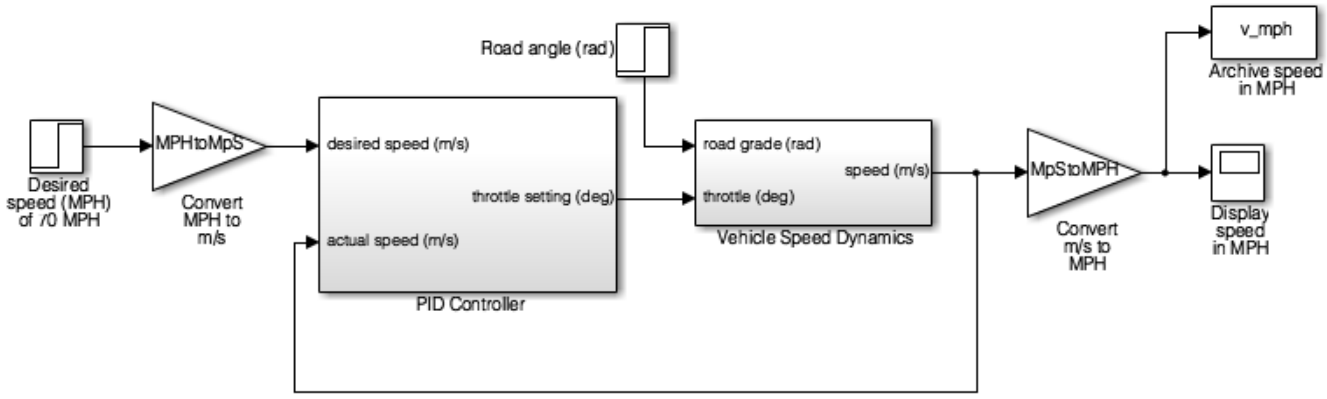

Figure 7: Simulink model of PID controll using subsystems.

The PID gains are not explicitly visible in Figure 7; they are "embedded" in the PID Controller subsystem.

3.3 P Control Only—How Well Does That Work?

Recall that in Figure 5 when encountering a 5% grade at 70 mph at constant throttle, after 10 seconds the speed dropped to 60 mph—finally dropping to 35 mph after a couple minutes. That's pretty bad.

Let's try two values of proportional gain $K_p = 10$ and $K_p = 20$. The response of these two systems is shown in Figure 8.

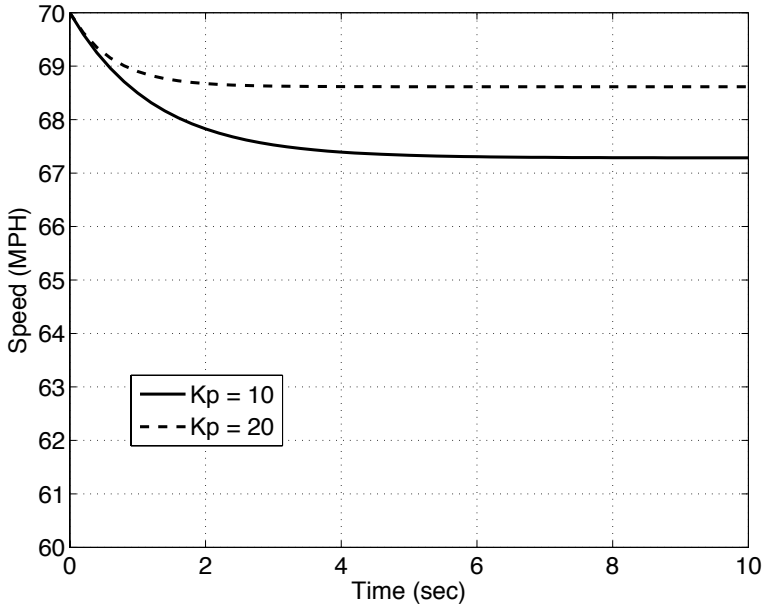

Figure 8: Response of P-controlled systems to 5% grade.

Wow, **MUCH BETTER!** With both gains, the speed only drops a few MPH. With the higher gain, the

speed drops less. Also, the speed settles after only a couple seconds.

However, we would really like for the cruise control to "regain" the speed it lost on the hill—and for that we need to add integral (I) action.

3.4 PI Control.

Keeping the value of $K_p = 20$, add a small amount of integral action, $K_i = 5$. The plot of this response is shown in Figure 9.

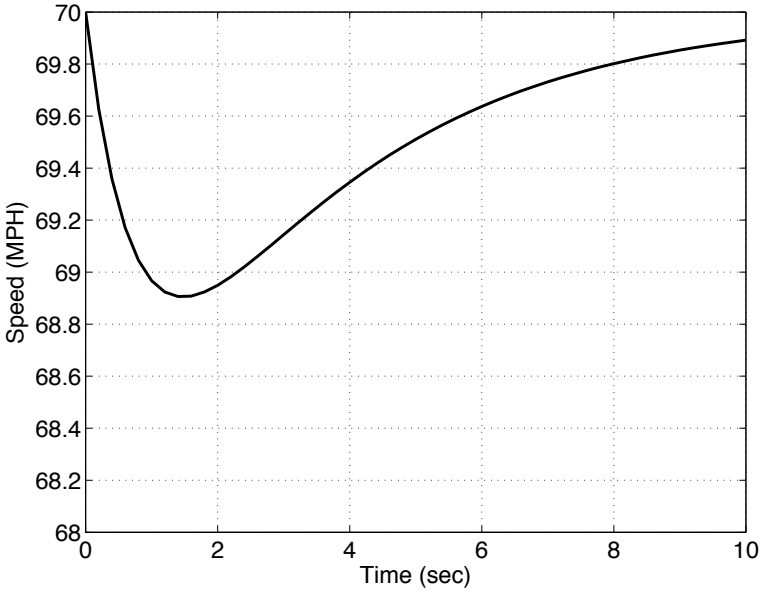

Figure 9: Response of PI system to road grade with $Ki = 5$.

As the speed begins to drop from the effect of the hill, the integrator begins to "kick in" and crank it back up again. Unlike the P-controlled system, the integrator will relentlessly drive the speed back to the desired value. Note that it takes quite a while—after 10 seconds it still isn't back yet. We need more gain!!

I can't fit the next two plots on the space remaining on this page, so I'll just go on to the next page.

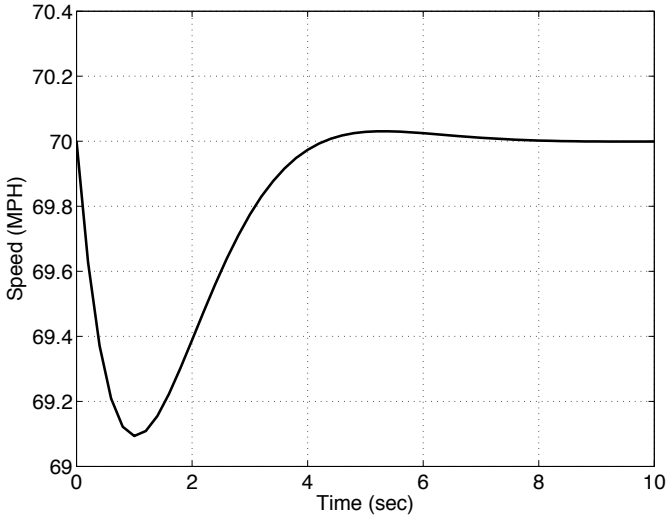

Figure 10: Response of PI system to road grade with $Ki = 15$.

Figure 10 shows the response with $K_i = 15$. This looks *VERY NICE!* After 1 second the integrator takes over and cranks the speed back up to the original value by about 4 seconds. Then it settles down nicely. This looks good to me.

3.5 What If the Integral Gain is Too High?

Here is my final point. If some is good, more is better, right? Let's crank the integral gain K_i up to 50, and see what happens. Figure 11 shows what happens. The system responds even faster, but overshoots, and...

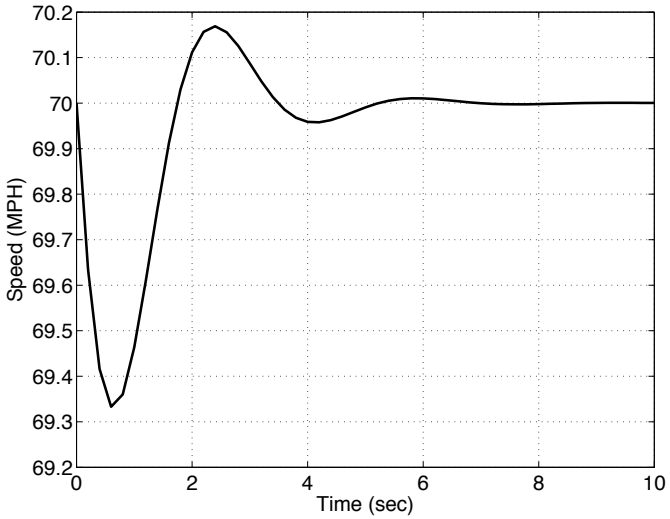

Figure 11: Response of PI system to road grade with $K_i = 50$.

Figure 11 is what the Tuneboy cruise control feels like to me. It corrects a little *too fast* and oscillates ("hunts") a lot. I feel that Wayne should reduce the integral gain to about 50% of its current value and see what happens.