Cruise Control System

In this section, the white-box modeling technique is applied on a cruise control system, the system components are known and the mathematical relations between the input and output are written directly based on this.
There are three major subsystems that need to be understood to build this model successfully from scratch: the car plant model, the terrain model and the controller.

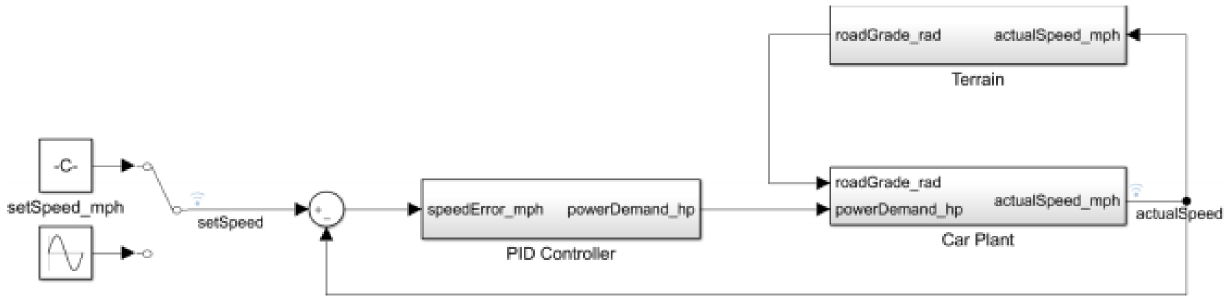

Figure 2-4 Overall car model diagram.

Car Plant Model

The expression for power required is developed as a function of the car's mass, its acceleration/deceleration, the slope of the terrain it's operating on, and its aerodynamic drag. The expression is shown in equation (2-5) with its simulink implementation in Fig. 2-5.

$$P = mV\dot{V} + C_d A \cdot \frac{1}{2}\rho V^3 + mgV \sin\theta \tag{2-5}$$

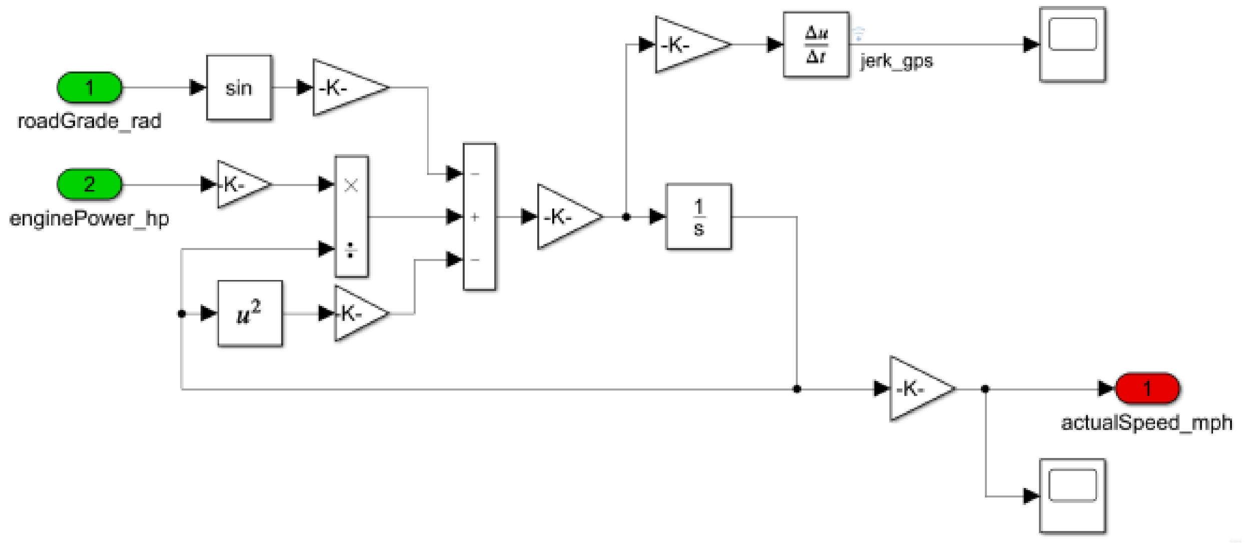

Figure 2-5 Simulink implementation for the car model.

To complete the plant model, a simple engine is made to prevent the unrealistic scenario of power commands resulting from the controller being instantly realized and fed as input power to the car. The easiest way to meet this condition is to model the engine as a saturation block followed by a first order lag. The built engine model is illustrated in Fig. 2-6., with an input of the commanded power and an output of the actual power.

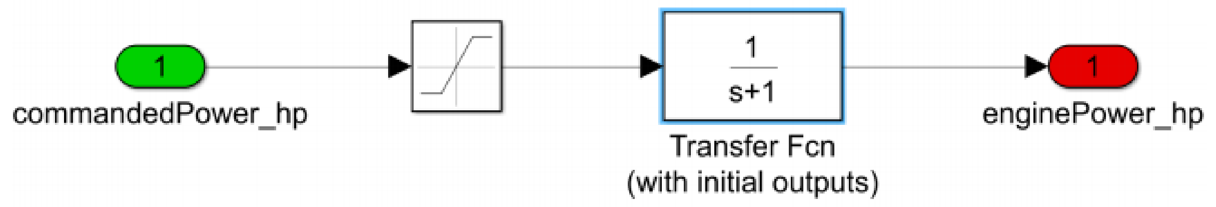

Figure 2-6 Simple first-order engine model with command saturation.

There are two important things to observe about this model. The first one is. The saturation block upper limit is determined by the parameter maxPower hp, that is specified in the initialization script. This prevents any command inputs from the controller in excess of this amount from flowing into the rest of the model. The saturation block lower limit is determined by the parameter -maxPower hp, since the need of slowing down without having a brake model. The braking power of any car should exceed the engine's capacity to move the car, so this is a reasonable limit. The second thing to observe is that an initial output from the first order transfer function needs to be specified in order to make it possible for the model to be initialized at any state other than zero engine power. The methods used to implement initial conditions in time domain simulations are provided by transfer functions and integrators.

The controller

The traditional PID control system includes three gains, one is a proportional gain on the feedback error, another one is a gain on the integral of the feedback error and the last one is a gain on the derivative of the feedback error. In Fig. 2-7., the three gains are shown as kP , kI and kD and they are added together to produce the control law. The problem of selecting the values for these gains is called tuning problem. It takes so much time from people to tune PID gains and it is really very difficult to obtain the three of them each one at his best value by simple trial and error. Fortunately, there are techniques that are often used to solve the gain tuning problem: root locus techniques, frequency domain techniques (Dode diagrams) and state-space time domain approaches but none of these is used here. Instead, a PD controller cascaded to PI controller building a slightly modified form of the PID architecture. This approach is suggested in Automatic Control Systems by Benjamin C. Kuo. Fig. 2-8 shows the PID control architecture. The traditional PID gains shown in Fig. 2-7 can be obtained once the PD-PI gains of Fig. 2-8 are established.

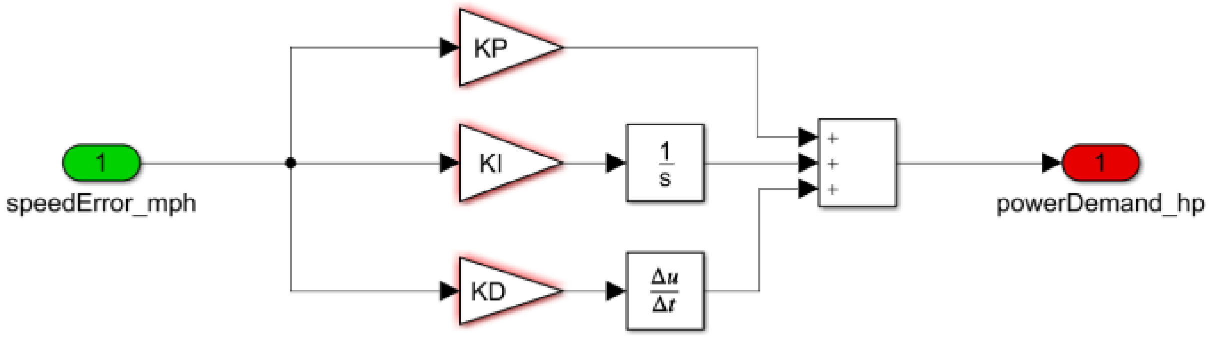

Figure 2-7 Traditional PID controller.

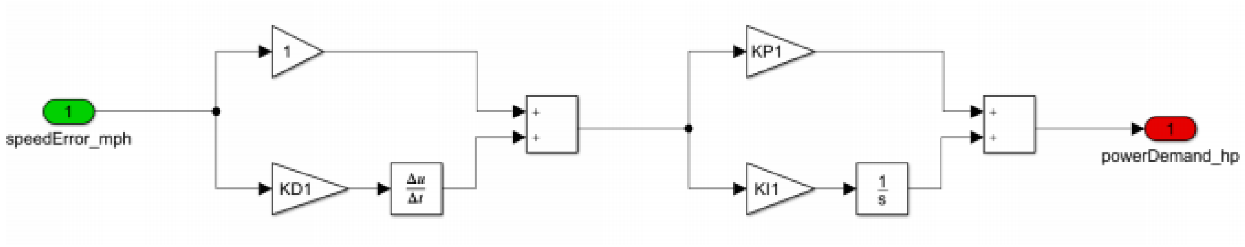

Figure 2-8 Modified PID controller (PD-PI controllers).

Using time domain simulation to design a controller can be achieved by evaluating the closed-loop system response to step input commands. The customer often wants specific values or ranges for rise time, peak overshoot and settling time of the system, all of them are shown in Fig. 2-9. The rise time indicated in the Figure is a common one: the time required for the response to go from 10% to 90% of its final value. The settling time is the time required for the response curve to reach and stay within a range about the final value of size specified by an absolute percentage of the final value, a common percentage is 2% (±1%).

With the PD-PI architecture, the first step is to zero out the derivative gain kD1 so that the result is a traditional PI controller. The values of kP1 and kI1 are then determined to satisfy the requirement on the rise time of the system. The maximum overshoot at this stage is not so much of a concern; it might be large. The PD portion can be used after that to reduce the maximum overshoot. kD1 is selected to satisfy the damping requirements (i.e., the overshoot and settling time requirements).

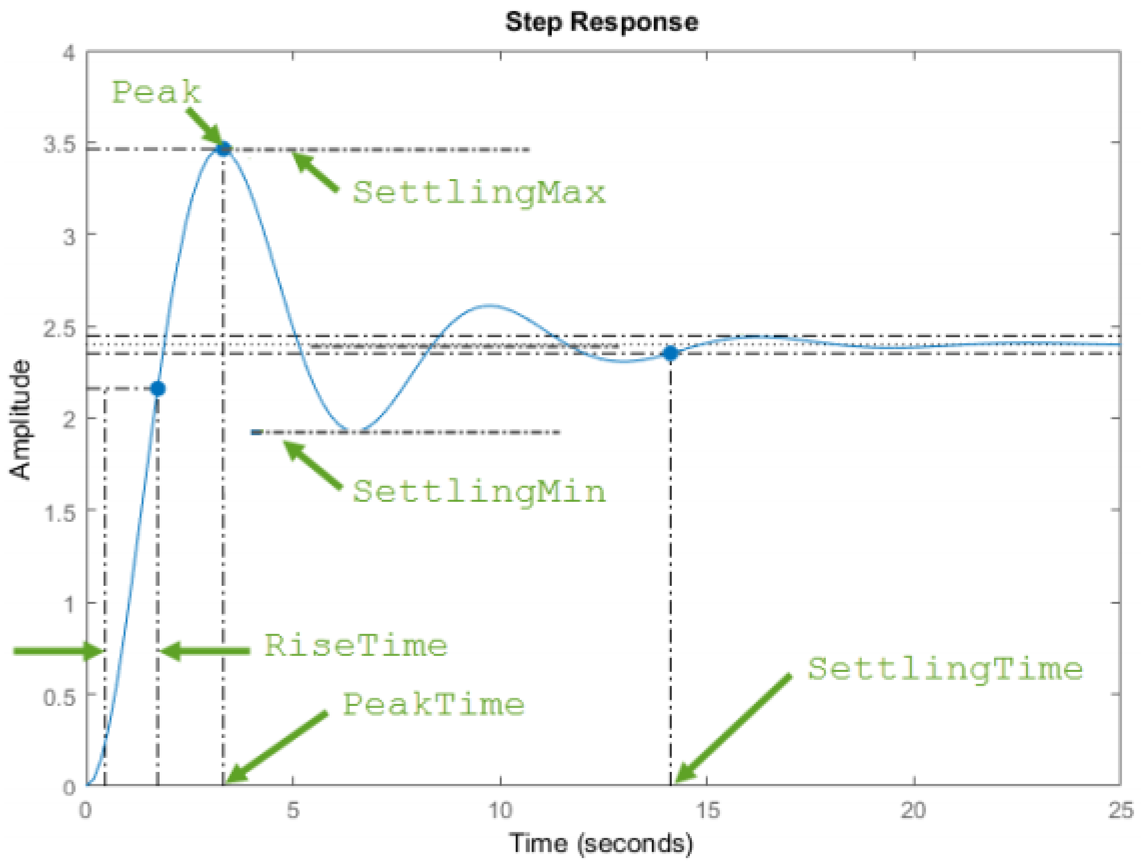

Figure 2-9 Time domain specifications.

There might be a necessity to re-select the original PI gains if the value of kD1 gets too large to be practical and repeat the process. The PID gains values that would be applicable to Fig. 2-7 are then known as functions of kP1, kI1 and kD1 as follows

$$k_P = k_{P1} + k_{D1}\, k_{I1} \tag{2-6}$$

$$k_D = k_{D1}\, k_{P1} \tag{2-7}$$

$$k_I = k_{I1} \tag{2-8}$$

2.4.3 The terrain model

The PID closed loop cruise controller is now designed and its performance with respect to regulating the speed while going up and down steep hills needs to be tested. To this end, the full model is now utilized, including the terrain subsystem.

The hill model defines the terrain height y(x) as

$$y(x) = \frac{A}{2}\left[(1 - \cos\left(\frac{\pi x}{L}\right)\right] \tag{2-9}$$

where A represents the peak height of the hill and L is the distance from the beginning of the hill to its peak at the midpoint. The x-coordinate is a horizontal axis and the y-coordinate represents the terrain height.
The road grade θ at any point x can then be calculated by taking the tangent of the slope,

$$\theta = \tan^{-1}\frac{dy}{dx} = \tan^{-1}\left[\frac{A\pi}{2L}\sin\left(\frac{\pi x}{L}\right)\right] \tag{2-10}$$

At the end points x = 0 and x = 2L, both the height and the slope are zero.

The vehicle speed will always remain tangent to the local terrain. Hence, the rate at which the vehicle advances in the x-coordinate is given by

$$\dot{x} = V\cos\theta \tag{2-11}$$

Hence,

$$x = \int V\cos\theta\, dt \tag{2-12}$$

To resolve the inter-dependence on x and θ that appears in equations (2-10) and (2-12) in the simulink model, initial data for *x* in the integrator are provided. For this case, leaving the initial value of *x* at its default value of zero is sufficient. Implementing these equations on Simulink results in a model like the one in Fig. 2-10.

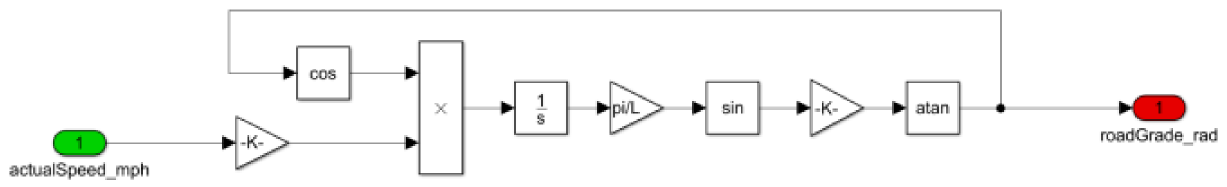

Figure 2-10 : Hill model Simulink implementation to provide regulator disturbances.