12 Cruise and Headway Control

One of the most widely adopted and visible control systems available on contemporary vehicles sold in the United States is the cruise control, which automatically regulates the vehicle longitudinal velocity by throttle adjustments. Typically, a vehicle cruise-control system is activated by a driver who wants to maintain a constant speed during long highway driving. This relieves the driver from having to continually adjust the throttle. The driver activates the cruise controller while driving at a particular speed, which then is recorded as the desired (or set-point) speed to be maintained by the controller.

Intelligent cruise control systems – also known as autonomous or adaptive cruise control (ACC) systems – are the next-generation product for cruise control. When no lead vehicle is within sight, an ACC vehicle behaves like a conventional cruise-control vehicle by maintaining a constant (i.e., target) speed. However, an ACC vehicle also has a headway-control mode. When the vehicle, using a range sensor, detects that it is close to the in front vehicle, the controller switches to headway-control mode and adjusts the speed to maintain a desired (i.e., safe) headway. Many field tests have been conducted to assess the real-life performance of ACC vehicles and consumers generally are receptive to the convenience provided by them. A rapidly growing number of luxury vehicles now offer ACC as an option.

12.1 Cruise-Controller Design

The main goal in designing the cruise-control algorithm is to maintain vehicle speed smoothly but accurately, even under large variations in plant parameters and road grade. In the case of cruise control for heavy trucks, the vehicle mass can change by up to several hundred percentage points (e.g., from a tractor with no trailer to towing a fully loaded trailer). Furthermore, a truck transmission may have numerous gears and frequent gear shifting may occur. Therefore, powertrain behavior may vary significantly, which causes additional complexity.

In the case of passenger automobiles or SUVs/light trucks, vehicle mass still may change noticeably but it is within a much smaller range. Also, the cruise-control function usually is activated only above a threshold speed (e.g., 30 mph); therefore, gear shifting is less of a problem. In this chapter, we assume that the engine and transmission characteristics can be inverted easily through engine-map inversion.

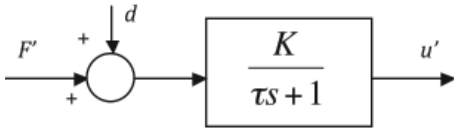


Figure 12.1. Block diagram of linearized vehicle longitudinal dynamics.

In other words, the desired traction force can be obtained easily through table lookup and commanding proper throttle angle or fuel-injection rate. In addition, we assume that tire friction is not a limiting factor. Therefore, we assume that we can directly manipulate traction force. Under these assumptions, a cruise-control design can be based on the equation of motion for longitudinal vehicle dynamics presented in Chapter 3 and the free-body diagram in Figure 4.5. Thus, from Eq. (4.4), the vehicle longitudinal motion is described by:

$$m\frac{du}{dt} = F_x - mg\sin\Theta - fmg\cos\Theta - 0.5\rho AC_d(u+u_w)^2 \tag{12.1}$$

Equation (12.1) is nonlinear in the forward velocity, *u(t)*, but otherwise is a simple dynamic system: it only has one state variable. So, what are the main challenges in cruise-control design problems? The difficulties arise mainly from two factors: (1) plant uncertainty due to change of vehicle weight, and (2) external disturbances due to road grade. Thus, a good cruise-control algorithm must work well under these uncertainties. We can linearize Eq. (12.1) about a nominal operating condition. At equilibrium (i.e., when $du/dt = 0$), Eq. (12.1) can be solved for:

$$F_{x0} = mg\sin\Theta_0 + fmg\cos\Theta_0 + 0.5\rho AC_d(u_0+u_w)^2 \tag{12.2}$$

For example, using the numerical values:

$$g = 9.81\,\text{m/s},\ u_0 = 20\,\text{m/s},\ \Theta_0 = 0,\ \text{m} = 1000\,\text{kg},$$
$$\rho = 1.202\,\text{kg/m}^3,\ A = 1\,\text{m}^2,\ C_d = 0.5,\ f = 0.015,\ u_w = 2\,\text{m/s};$$

this yields $F_{x0} = 292.6\,N$. Linearizing Eq. (12.1) about the specified operating (i.e., equilibrium) state by using a Taylor series expansion yields the linearized equation:

$$\tau\dot{u}' + u' = K(F' + d) \tag{12.3}$$

where the incremental, or perturbed, variables are defined as:

$$u = u_0 + u';\quad F_x = F_{x0} + F';\quad \Theta = \Theta_0 + \Theta'\quad d = mg(f\sin\Theta_0 - \cos\Theta_0)\Theta';$$
$$\tau = (m/(\rho C_d A(u_0+u_w)));\quad K = (1/(\rho C_d A(u_0+u_w)));$$

Using the parameter values given here, we obtain $\tau = 75.632$ sec and $K = 0.0756$ (m/s)/N. Based on the linearized equations, the process model for control-system design can be represented as shown in the block diagram in Figure 12.1.

Using this model, a PI controller can be designed, which has the transfer function:

$$G_c(s) = \frac{F'(s)}{e(s)} = K_P + K_I/s \tag{12.4}$$

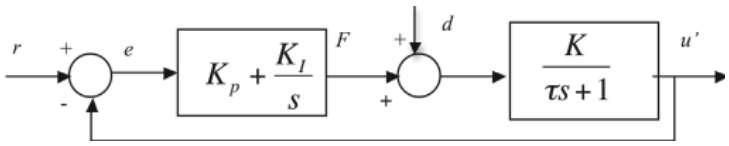


Figure 12.2. Closed-loop system with PI controller.

As shown in the block diagram in Figure 12.2, the closed-loop-system equations become:

$$u'(s) = \frac{K(K_I + K_P s)}{\tau s^2 + (KK_P + 1)s + KK_I} r + \frac{Ks}{\tau s^2 + (KK_P + 1)s + KK_I} d \qquad (12.5)$$

Thus, the controller gains (i.e., parameters K_P and K_I) must be selected to achieve good performance for the closed-loop system. This requires consideration of a number of factors, such as stability, steady-state errors, and transient response, which are discussed in Example 12.1.

EXAMPLE 12.1: PI CRUISE CONTROLLER DESIGN (FIRST-ORDER PLANT). A PI cruise-controller design requires consideration of several performance requirements (e.g., stability, steady-state error, and transient response) in selecting the control gains, K_P and K_I. Various control-design techniques, including pole placement, root-locus, and Bode plots, can be used. These techniques are made considerably easier to apply by the availability of computer-aided control-system design programs such as those available in MATLAB. The controller design is based on the plant-model parameters $K = 0.0756$ (m/s)/N and $\tau = 75.632$ seconds given previously. The closed-loop system has a second-order denominator, and its performance can be designed by manipulating the poles of the closed-loop characteristic equation. For example, the controller gains can be selected to achieve no overshoot ($M_p = 0$ or $\zeta = 1$) and a settling time of $t_s = 46$ seconds. The program also generates for the fixed value of $T_I = (K_p/K_I) = 18.686$ a root-locus plot with K_P as a parameter (see Figure 12.3). Finally, for the selected

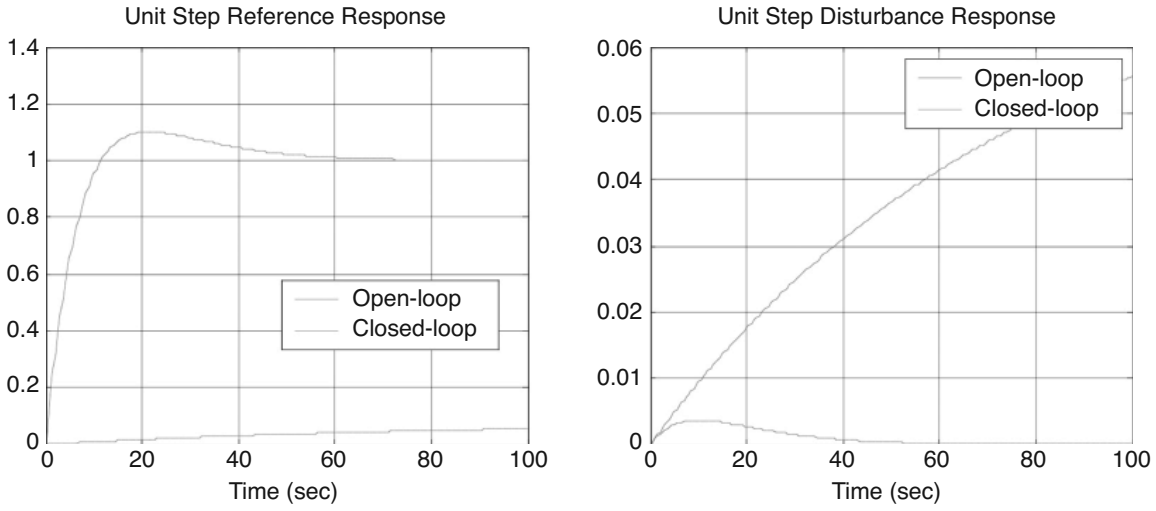


Figure 12.3. Response of PI control system for Example 12.1.

$K_P = 186.86$ and $K_I = 10$ values, the closed-loop system performance is simulated for step changes in reference speed and disturbance inputs, as shown in Figure 12.3. For these selected gains, the closed-loop system exhibits zero steady-state error and a settling time of approximately 45 seconds, as expected. However, despite the value of $\zeta = 1$ the response exhibits some overshoot. This is because the settling time and overshoot criteria used here are for second-order systems with no zeros; the closed-loop system is second-order but has a zero at -0.0535. If the response of the closed-loop system was determined to be not fast enough, a smaller T_I could be selected. For example, the step response of the system with $T_I = 9.34$ and $K_P = 401$ shows a 50 percent reduction in rising and settling times.

```
% Ex12_1.m
% Define the linearized model parameters
u0=20; g=9.81; m=1000; f=0.015; Theta=0;
rho=1.202; A=1; Cd=0.5; uw=2;
% Calculate the equilibrium force, Fx0:
Fx0 = m*g*sin(Theta) + f*m*g + ...
          0.5*rho*A*Cd*(u0+uw)^2;
% The time constant and dc gain are:
Tau=(m/(rho*A*Cd*(u0+uw))); K=Tau/m;
% Fix the Ti gain (Ki=Kp/Ti) for the PI control:
Ti=186.86/10.0;
% The loop transfer function for PI control is
% Kp*(num/den) where num/dec = (1+1/Ti*s)*K/(1+Tau*s)
num_o=K*[Ti 1]; den_o=[Ti*Tau Ti 0];
% Calculate open loop zeros and poles:
z=roots(num_o);
p=roots(den_o);
% Obtain the root locus for Kp:
K_root=[0:10:300];
R=rlocus(num_o,den_o,K_root);
plot(R,'*'); title('RootLocus Plot');
axis([-0.2 0 -0.05 0.05]), pause;
[K_root, POLES]=rlocfind(num_o,den_o)

% Simulate the reference unit step response
% with and w/o control:
t=[0:0.5:100];
sys = tf([K],[Tau 1]);
y=step(sys,t);
Kp=186.86; KI = Kp/Ti;
numc=K*[Kp KI]; denc=[Tau K*Kp+1 K*KI];
% Calculate the closed-loop zeros and poles:
zc=roots(numc)
pc=roots(denc)
sys_c = tf(numc, denc);
yc=step(sys_c, t);
```

```
plot(t,y,'-r',t,yc,'--b'); grid;
xlabel('Time (sec)')
title('Unit Step Reference Response');
legend('Open-loop', 'Closed-loop'); pause;
% Simulate the disturbance unit step response
% with and w/o control:
yd=step(sys,t);
sys_disturbance = tf([K 0], denc);
ycd=step(sys_disturbance,t);
plot(t,yd,'-r',t,ycd,'--b'); xlabel('Time (sec)'),grid;
title('Unit Step Disturbance Response');
legend('Open-loop', 'Closed-loop');
```

The simple model used for the cruise-control design in Example 12.1 has several shortcomings. First, it was assumed that the tractive force can be changed instantaneously by the controller – that is, the throttle/engine system dynamics has been neglected. Second, the controller design is based on linearized equations of longitudinal motion, and the time constant and gain are assumed to be constant. It reality, the vehicle-forward dynamics are nonlinear and the corresponding linear dynamics vary depending on operating conditions. Both issues have been addressed by various studies in the literature (e.g., Liubakka et al. 1991; Oda et al. 1991; and Tsujii et al. 1990). Example 12.2 illustrates the design of a PID cruise controller based on a more complex but still linear model, which accounts for the throttle/engine dynamics. It also has been proposed to develop adaptive versions of the cruise controller to estimate the actual parameters K and τ under various operating conditions. An adaptive version of the PI cruise-control system is discussed in Example 12.3.

The closed-loop transfer function of the vehicle shown in Figure 12.4 is:

$$u'(s) = \frac{KK_a(K_Ds^2 + K_Ps + K_I)}{s^2(\tau s+1)(\tau_a s+1) + KK_a(K_Ds^2 + K_Ps + K_I)} r + \frac{Ks^2(\tau_a s+1)}{s^2(\tau s+1)(\tau_a s+1) + KK_a(K_Ds^2 + K_Ps + K_I)} d$$

EXAMPLE 12.2: PID CRUISE-CONTROLLER DESIGN (SECOND-ORDER ACTUATOR). The process model in Eq. (12.3) neglects any dynamics associated with the actuator, which adjusts the throttle angle. In Tsujii et al. (1990), a *dc* motor throttle actuator model is given as:

$$G_a(s) = \frac{K_a}{s(\tau_a s+1)}$$

where the input is the motor-drive duty cycle (percent) and the output is the tractive force. Parameter values used here are the same as those in Example 12.1, plus

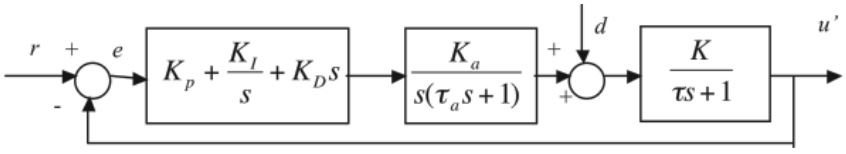


Figure 12.4. Closed-loop system with PID controller and throttle dynamics.

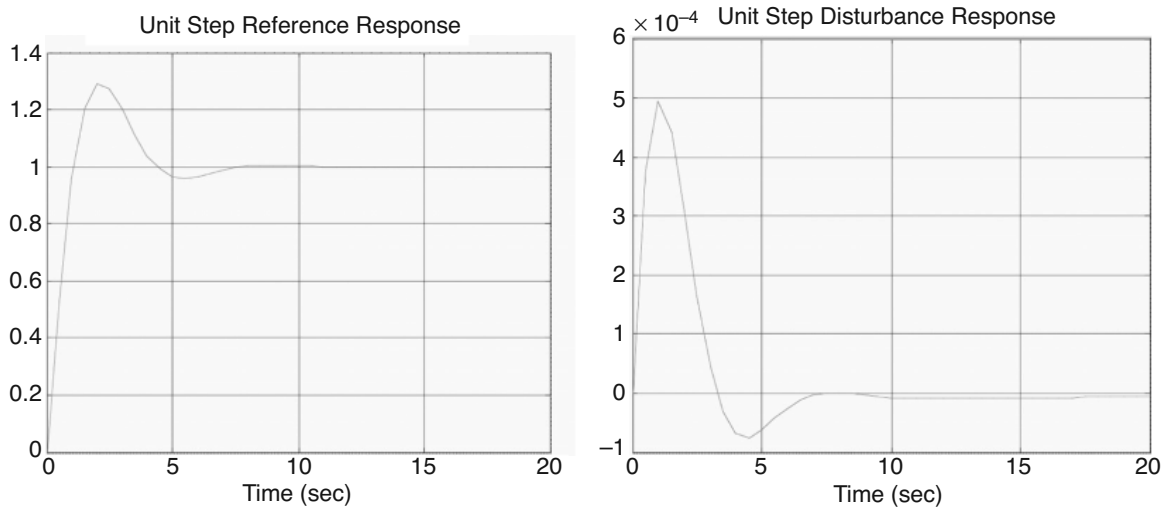


Figure 12.5. Response of PID control system for Example 12.2.

$K_a = 10$ (Newton-sec/%) and $\tau_a = 0.05$ seconds. Clearly, the actuator dynamics usually is much faster than the vehicle dynamics (i.e., $\tau_a \ll \tau$), and it is reasonable to neglect actuator dynamics as in Example 12.1. However, in this example, the throttle dynamics is considered and the controller is a PID controller rather than the PI controller used in Example 12.1. The closed-loop block diagram is shown in Figure 12.4. With the controller gains, $K_P = 120$, $K_I = 12$, and $K_D = 120$, the closed-loop-system block diagram and responses to step reference and disturbance inputs are illustrated in Figure 12.5. The response is faster than the PI controller shown in Example 12.1 but has a larger overshoot. The P gain is selected for a damping ratio of approximately 0.707.

```
% Ex12_2.m
% Define the linearized model parameters
clear
u0=20; g=9.81; m=1000; f=0.015; Theta=0;
rho=1.202; A=1; Cd=0.5; uw=2;
% Calculate the equilibrium force, Fx0:
Fx0 = m*g*sin(Theta) + f*m*g + ...
            0.5*rho*A*Cd*(u0+uw)^2;
% Time constant & dc gain are then:
Tau=(m/(rho*A*Cd*(u0+uw))); K=Tau/m;
% The throttle actuator is first-order with integrator:
Ka=10; TauA=0.05;
% Fix the Ti gain (Ki=Kp/Ti) for the PID control:
Ti = 100.0;
% Fix the Td gain (Td=Kd/Kp) for PID control:
Td = 1.0;
% The open loop transfer function for PID control is
% Kp*(num/den) where:
num_o=K*Ka*[Td 1 1/Ti];
den_o=[Tau*TauA Tau+TauA 1 0 0];
```

```
% Calculate and display open loop zeros and poles:
z=roots(num_o)
p=roots(den_o)
% Obtain the root locus for Kp:
K_locus=0:5:160;
R=rlocus(num_o,den_o,K_locus);
plot(R,'*'); title('Root locus'), pause
% Simulate the unit step response with PID control:
t=[0:0.5:20];
Kp=120.0;
numc=Kp*num_o;
denc=den_o+[0 0 numc];
% Calculate the closed-loop zeros and poles:
zc=roots(numc)
pc=roots(denc)
sys_c = tf(numc, denc);
yc=step(sys_c, t);
plot(t,yc,'r'); xlabel('Time (sec)'), grid;
title('Unit Step Reference Response'); pause;
% Simulate disturbance unit step
% response with and w/o control:
sys_disturbance = tf([K*TauA K 0 0], denc);
ycd=step(sys_disturbance,t);
plot(t,ycd,'--b'); xlabel('Time (sec)'),grid;
title('Unit Step Disturbance Response');
```

EXAMPLE 12.3: ADAPTIVE PI CRUISE-CONTROLLER DESIGN. As mentioned previously, the gain, K, and time constant, τ, vary with operating conditions (e.g., mass, speed, and wind velocity). In this example, the model in Eq. (12.3) is used (i.e., actuator dynamics is neglected) and an estimation algorithm is developed to identify the parameters. In other words, the parameters K and τ are estimated on-line from the measured signals. The controller gains then are updated by using the estimated parameters. This "adaptive" cruise-control (ACC) system structure is illustrated in Figure 12.6, which consists of two major

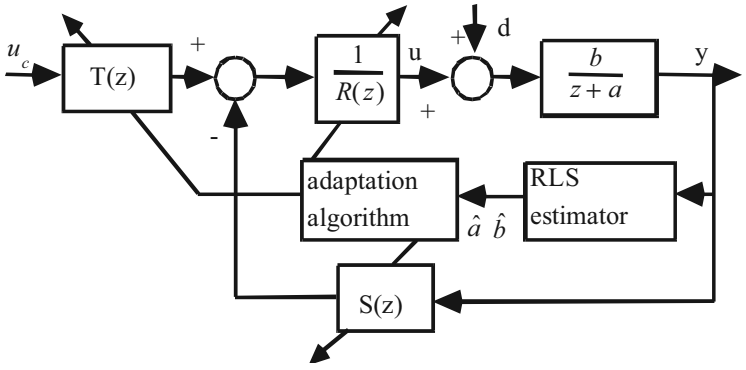


Figure 12.6. Structure of the adaptive cruise control system.

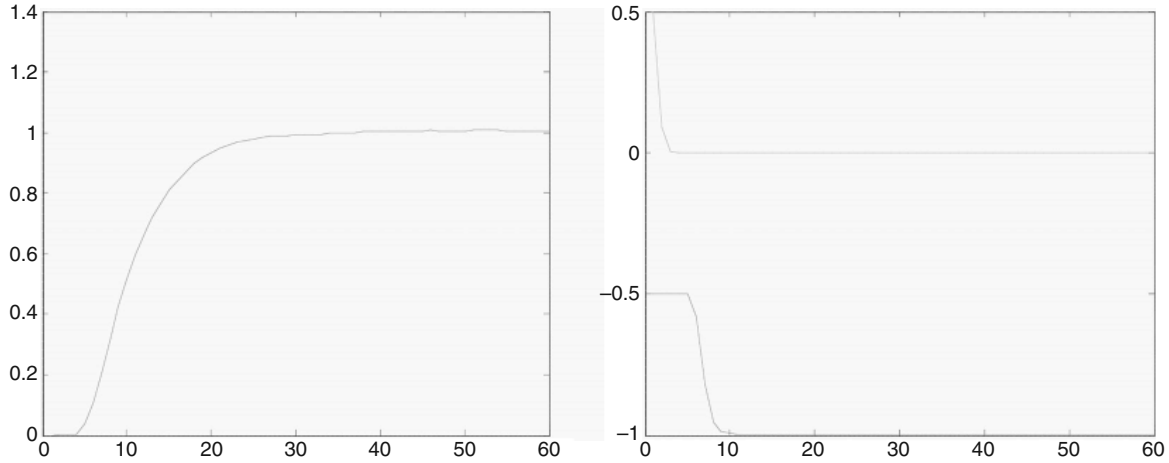


Figure 12.7. Adaptive PI control for Example 12.3.

parts: (1) a recursive least squares (RLS) parameter-estimation algorithm; and (2) an indirect (or self-tuning) PI controller strategy. The MATLAB program shown here is used to generate the results shown in Figure 12.7. The implementation is digital, so a discrete-time process model is used, obtained for a sampling rate of 1 second. The parameter estimates converge quickly (i.e., in about 5 to 10 seconds) and the response is comparable to that shown in Figure 12.2 (somewhat slower and with more overshoot), despite the fact that the unknown parameters are initialized to zero. In the MATLAB program, the RLS estimation algorithm is used to estimate the plant parameters. The basic equations of the RLS algorithm are summarized as follows:

$$
\begin{aligned}
y(k) &= \phi^T(k)\theta \\
\hat{y}(k) &= \phi^T(k)\hat{\theta} \\
\hat{\theta}(k+1) &= \hat{\theta}(k) + K(k)[y(k+1) - \phi^T(k+1)\hat{\theta}] \\
P(k+1) &= [I - K(k)\phi^T(k+1)]P(k)/\lambda \\
K(k) &= P(k)\phi(k+1)[\lambda + \phi^T(k+1)P(k)\phi(k+1)]^{-1}
\end{aligned}
$$

In this example,

$$y(k) = -ay(k-1) + bu(k-1)$$

Therefore,

$$\phi^T(k) = [-y(k-1)\, u(k-1)] \qquad \hat{\theta} = [\hat{a} \quad \hat{b}]^T$$

```
% Ex12_3.m
  K = 0.0756; Tau = 75.632;
  A=-(1/Tau); B=(K/Tau); T=1.0;
  [F,G]=c2d(A,B,T);
  C=1; D=0; [num,den]=ss2tf(F,G,C,D,1);
  b=num(2)/den(1); a=den(2)/den(1);
% initialization:
```

```
  kmax=60; n=2; lambda=0.995;
  theta = [-0.5;0.5]; P = 1000*eye(n);
  u_old = 0.0; y_old = 0.0; uc_old = 1.0;
  phi = [-y_old; u_old];
  uc=1.0+0.01*randn(1);
  Y = zeros(kmax+1,n+3);
  Y(1,:)=[u_old uc y_old theta'];
% System with a ''PI'' controller
% designed for given z, wn
% and parameter estimation using RLS:
  z=0.95; wn=0.1;
% Controller Ru = Tuc - Sy
for k=1:kmax,
  uc = 1.0 + 0.02*randn(1);
  y = - a*y_old + b*u_old;
  [theta,P] = rls(y,theta,phi,P,lambda);
  a1 = theta(1); b1=theta(2);
  % we use estimated values
  % (a1, b1) of the
  % process for the controller
  % gains and control:
  s0 = (exp(-2*z*wn*T)+a1)/b1;
  s1 = (1-a1-2*cos(wn*T* ...
       (1-z^2)^0.5)*exp(-z*wn*T))/b1;
  t0 = s0 + s1;
  u = u_old + t0*uc_old - ...
      s0*y_old -s1*y;
  % update the measurement vector phi
  % for next iteration:
  phi = [-y;u];
  u_old = u; y_old = y; uc_old = uc;
  Y(k+1,:)=[u uc y a1 b1];
end
% Display the results:
k=[0:kmax]'; u_uc_y_ai_bi = Y;
plot(k,Y(:,3),'r'); title('Response');
xlabel('Sampling Interval, k'); ylabel('y'); pause;
plot(k,Y(:,4:n+3)); title('Parameter Estimates');
xlabel('Sampling Interval, k'); ylabel('Estimated a & b');
% function saved in
% rls.m
function [theta, P] = rls(y, theta,
              phi, P, lambda)
K=P*phi*inv(lambda+phi'*P*phi);
theta=theta+K*(y-phi'*theta);
P=(eye(size(P))-K*phi')*P/lambda;
```

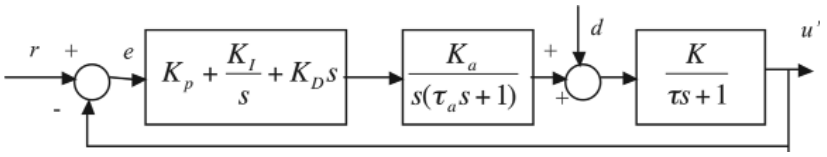


Figure 12.8. Block diagram of the adaptive cruise control system.

EXAMPLE 12.4: ADAPTIVE PID CRUISE-CONTROLLER DESIGN. In this example, we revisit Example 12.2, assuming that some kind of parameter-identification method has been implemented and that we have access to accurate estimation of vehicle parameters (i.e., τ and K). How then can we take full advantage of the three design variables (i.e., the PID gains: K_p, K_I, and K_D) for superior performance? In previous examples, we used the root-locus technique, which helps to select one control parameter; the other two parameters were specified in advance. Thus, a form of trial and error or engineering judgment is needed to find a set of good control parameters. In this design, we show an alternative design method based on pole placement. First, we derive the closed-loop transfer function:

$$\frac{u'}{r} = \frac{K_D K_a K s^2 + K_P K_a K s + K_I K_a K}{\tau\tau_a s^4 + (\tau + \tau_a)s^3 + (1 + K_D K_a K)s^2 + K_P K_a K s + K_I K_a K}$$

The closed-loop poles are the roots of the denominator polynomial (i.e., the characteristic equation), which can be rewritten in the following monic form:

$$s^4 + \frac{\tau + \tau_a}{\tau\tau_a}s^3 + \frac{1 + K_D K_a K}{\tau\tau_a}s^2 + \frac{K_P K_a K}{\tau\tau_a}s + \frac{K_I K_a K}{\tau\tau_a} = 0$$

Because we only have three design variables, we can place only three of the four closed-loop poles. The remaining pole then is determined by the fixed coefficient ($\frac{\tau+\tau_a}{\tau\tau_a}$) of the closed-loop characteristic equation. Notice that the actuator-time constant is much faster than the vehicle-time constant (i.e., 0.2 versus 55$\sim$140). If we select three slow (i.e., dominant) closed-loop poles, the last pole, which may move a little when τ changes, will be far from the imaginary axis and thus does not noticeably affect the system performance. Based on the damping ratio and bandwidth of the complex conjugate pairs, we determine that the three poles should locate at $-0.3 \pm 0.1j$ and -0.4. The closed-loop characteristic equation then becomes:

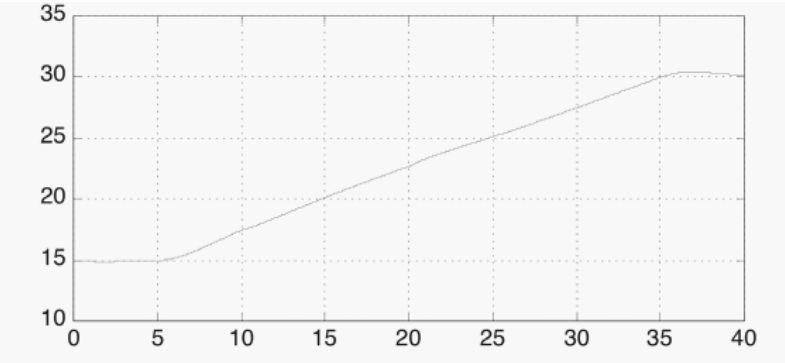
$$\begin{aligned}
&(s^2 + 0.6s + 0.1)(s + 0.4)\left(s + \frac{\tau + \tau_a}{\tau\tau_a} - 1\right) \\
&\quad = s^4 + \frac{\tau + \tau_a}{\tau\tau_a}s^3 + \left(\frac{\tau + \tau_a}{\tau\tau_a} - 0.66\right)s^2 + \left(0.34\frac{\tau + \tau_a}{\tau\tau_a} - 0.3\right)s \\
&\qquad + 0.04\left(\frac{\tau + \tau_a}{\tau\tau_a} - 1\right)
\end{aligned}$$

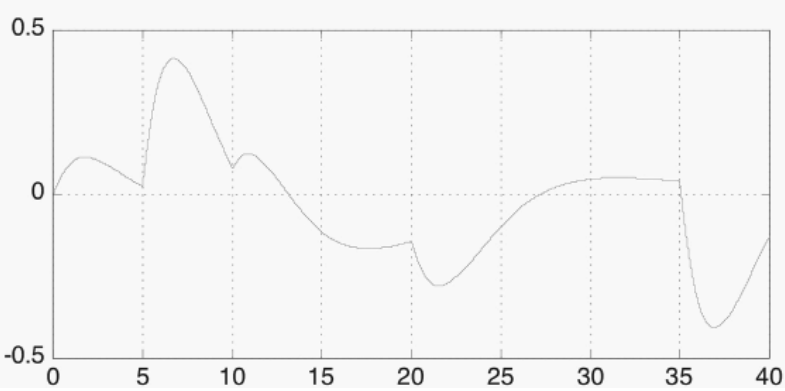
The PID gains to achieve these pole assignments are then:

$$K_D = \frac{\tau + \tau_a - 0.66\tau\tau_a - 1}{K_a K} \quad K_P = \frac{0.34(\tau + \tau_a) - 0.3\tau\tau_a}{K_a K}$$

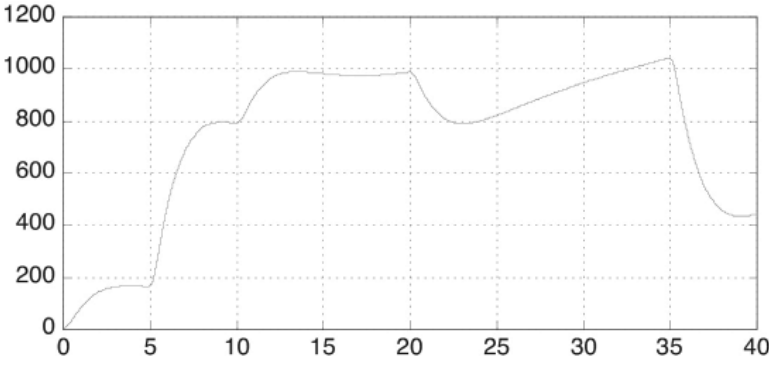
$$K_I = \frac{0.04(\tau + \tau_a) - 0.04\tau\tau_a}{K_a K}$$

The unplaced pole will be located at $-\frac{1}{\tau_a} - \frac{1}{\tau} + 1$, which is about 10 times faster than the three slower poles. In other words, the performance of the adaptive controller is not affected much even when the vehicle-operation conditions vary. In the following example program, we assume that the vehicle is accelerating from 15 to 30 m/sec in 30 seconds and a 2 percent road gradient also is assumed to be present. It can be seen that the adaptive PID control algorithm is tracking the desired speed command with small error under these two uncertainties.





In this MATLAB program, we used the Euler approximation for integration. This may introduce some error but it is more flexible because all of the intermediate variables (e.g., engine force F and command to actuator c) can be examined. Sometimes it is desirable to keep track of these intermediate variables. For example, the force (F) in the following figure shows that the controller design is reasonable because the force command is not excessively large:

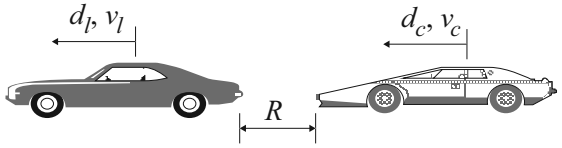


```
% Ex12_4.m
g=9.81; m=1000; f=0.015; Theta=0;
rho=1.202; A=1; Cd=0.5; uw=0;
Ka=10; TauA=0.2; T=0.01; u(1)=15.0;
error=0.0; error_old=0.0; int_error=0.0;
F_dot=0.0; F=0.0; t=0:T:40;
r=[15*ones(1,5/T+1),15+(1:30/T)*0.5*T,30*ones(1,5/T)];
d=[zeros(1,10/T+1), -0.02*m*g*ones(1,10/T), zeros(1,20/T)];
for i=1:40/T,
   Tau=(m/(rho*A*Cd*(u(i)+uw))); K=Tau/m;
   Kd=(TauA+Tau-0.66*TauA*Tau-1)/(Ka*K);
   Kp=(0.34*(TauA+Tau)-0.3*TauA*Tau)/(Ka*K);
   Ki=(0.04*(TauA+Tau)-0.04*TauA*Tau)/(Ka*K);
   error_old=error;
   error=r(i)-u(i);
   int_error=int_error+error*T;
   c=Kd*(error-error_old)/T+Kp*error+Ki*int_error;
   F_dotdot=(Ka*c-F_dot)/TauA;
   F_dot=F_dot+F_dotdot*T;
   F=F+F_dot*T;
   u_dot=(K*(F+d(i))-u(i))/Tau;
   u(i+1)=u(i)+u_dot*T;
end
plot(t(1:40/T),u(1:40/T),'r'), grid
xlabel('time (sec)'); title('True speed (m/sec)')
pause; plot(t(1:40/T),r(1:40/T)-
u(1:40/T),'r'); grid
xlabel('time (sec)'); title('Speed error (m/sec)')
```

12.2 Autonomous Cruise Control: Speed and Headway Control

A vehicle cruise-control system regulates vehicle longitudinal velocity. As an intermediate step between platooning (see Chapter 19) and cruise control, the control of the vehicle longitudinal motion (i.e., velocity and position) is considered. The idea is to allow the vehicle to regulate speed when there are no automobiles in front of it. However, when a vehicle is detected in front, then the controller regulates relative distance between the two of them (i.e., "headway"). Such a system (i.e., ACC or intelligent cruise control) can be effective in stop-and-go traffic and on the highway and can be used in conjunction with the auto-cruise system described in Chapter 19. The vehicle cruise-control system uses the desired speed set by the driver (e.g., the speed limit) as a reference input when there is no vehicle in sight. When a vehicle is detected in front, the system switches to the headway-control mode until the lead vehicle exceeds the reference speed or vanishes from sight, when it then switches back to speed control. In this scenario, the driver does not need to be concerned about longitudinal-motion control under conditions ranging from stop-and-go traffic to highway driving. A formulation of the control-system design problem for ACC is presented in the following example.

Figure 12.9. Headway control problem.



EXAMPLE 12.5: AUTONOMOUS CRUISE CONTROL SYSTEM DESIGN. To differentiate from the adaptive cruise control problem presented in the previous section, we name the range-regulation enhanced cruise-control system autonomous cruise control (ACC). In the literature, this system is commonly referred as intelligent cruise control (ICC), adaptive cruise control (ACC), or autonomous intelligent cruise control (AICC). When no lead vehicle is in sight, the performance of an ACC vehicle is similar to a traditional cruise-control vehicle. When a lead vehicle is present, we must model the interaction with the lead vehicle and keep track of the range variable, which is one of the most important performance variables.

It is assumed that the measurement of the vehicle longitudinal velocity (v_c) as well as of the distance (R) between the first vehicle (i.e., controlled vehicle) and the vehicle in front (i.e., lead vehicle) are available. This is illustrated in Figure 12.9. Using a linearized description of the longitudinal velocity of each vehicle, as discussed in a previous section, we can obtain the following state equations:

$$\begin{aligned}
\dot{x}_1 &= x_3 - x_2 \\
\dot{x}_2 &= -(1/\tau_c)x_2 + (K_c/\tau_c)u + (K_c/\tau_c)w_c \\
\dot{x}_3 &= -(1/\tau_l)x_3 + (K_l/\tau_l)w_l
\end{aligned}$$

where $x_1 = R$, $x_2 = v_c$, $x_3 = v_l$ and u is the control input (i.e., traction force) applied to the controlled vehicle (i.e., Vehicle c). The inputs w_c and w_l represent disturbances (e.g., grade disturbances for both vehicles and driver inputs for the lead vehicle). The open-loop-system block diagram is shown in Figure 12.10a, and an equivalent block diagram is shown in Figure 12.10b. The two block diagrams result in a different number of state variables because of the difference in the selection of disturbance inputs and the fact that the vehicle displacements are ignored in the second representation. When an ACC algorithm is implemented using differential GPS, for example, then it may be interesting to keep vehicle displacements as state variables. That extension is readily achieved by simply integrating the vehicle speeds.

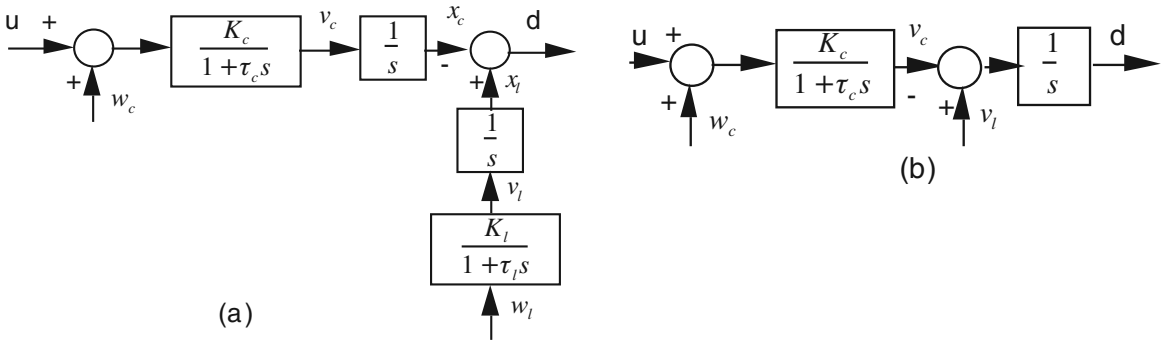


Figure 12.10. Open-loop block diagram for ACC.

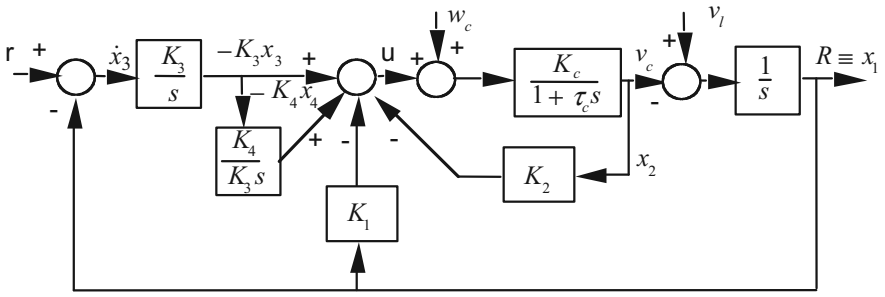


Figure 12.11. Closed-loop block diagram of the ACC system.

Consider the design of a controller to maintain a desired headway, r, based on measurements of R and v_c only. This requires only a range sensor mounted on the controlled vehicle, in addition to measurement of the controlled variable (i.e., vehicle longitudinal speed) and no information from other vehicles or from the roadway infrastructure. The control problem is illustrated in the block diagram shown in Figure 12.11. The reference input, r, should be computed as a function of the vehicle forward velocity to maintain safe headway (e.g., $r = t_h v_c$, where t_h is the desired headway time). Also, the velocity of the lead vehicle (i.e., Vehicle l) is not known and acts as a disturbance input. The control, u, on Vehicle c must be calculated to maintain the desired headway. A feedback controller can be designed for this system based on the following state equations:

$$\begin{aligned}
\dot{x}_1 &= -x_2 + v_l \\
\dot{x}_2 &= -(1/\tau_c)x_2 + (K_c/\tau_c)(u + w_c) \\
\dot{x}_3 &= (R - r) = (x_1 - r) \\
\dot{x}_4 &= x_3 \\
u &= -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4
\end{aligned}$$

This controller feeds back the measured values $x_1 = R$ and $x_2 = v_c$; x_3, the integral of the error, $e = d - r$; and x_4, the double integral of the error. The two integrals are used because it is assumed that the lead vehicle velocity (which acts as a disturbance to be rejected by the control law) can be modeled as a ramp-type disturbance input. With the double integrator built into the control algorithm, the controlled vehicle can follow the lead vehicle even when it accelerates. The controller gains, k_i, can be determined by a variety of methods, and a pole-placement design technique is used in the MATLAB program. The closed-loop poles are placed at $s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2}\,j$, and $s_{3,4} = -\alpha\zeta\omega_n$. The values $\zeta = 0.9$, $\omega_n = 0.4$, and $\alpha = 3.0$ are used in the MATLAB simulations for Example 12.5.

In this setup, the lead-vehicle speed (v_l) is treated as a disturbance to be rejected. It may be desired to treat w_l (instead of v_l) as the unknown disturbance and include v_l as a state variable (x_5), which can be used for the control/warning purposes. In this case, the dynamic equations are as follows:

$$\begin{aligned}
\dot{x}_1 &= x_5 - x_2 \\
\dot{x}_2 &= -(1/\tau_c)x_2 + (K_c/\tau_c)u + (K_c/\tau_c)w_c \\
\dot{x}_3 &= (d - r) = (x_1 - r)
\end{aligned}$$

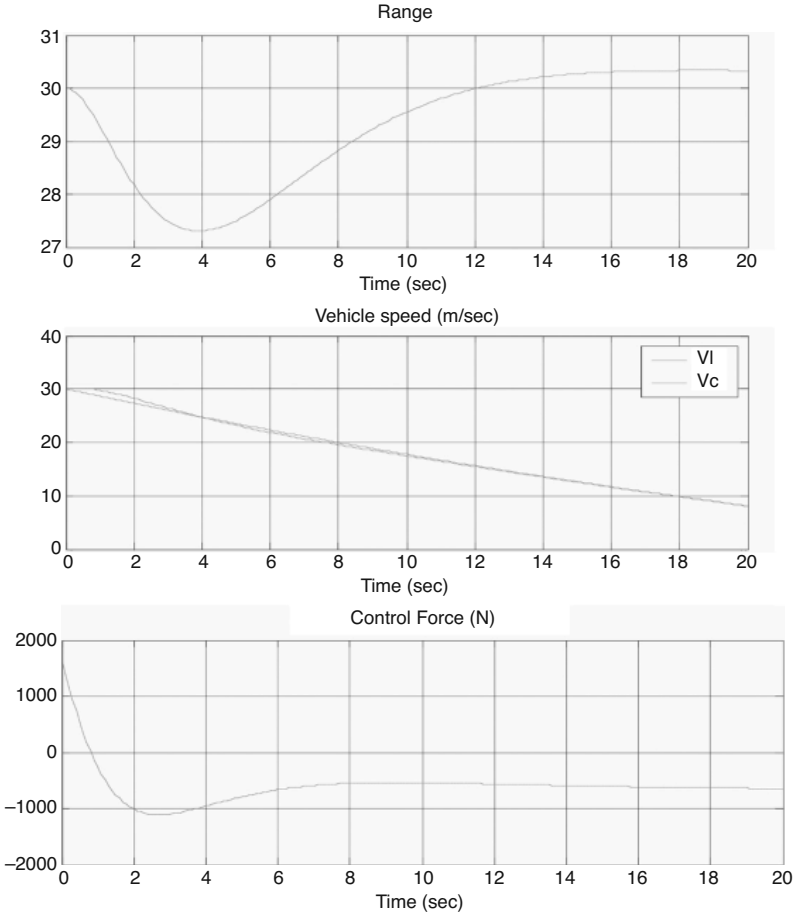


Figure 12.12. Simulation results of the ACC system.

$$\dot{x}_4 = x_3$$
$$\dot{x}_5 = -(1/\tau_c)x_5 + (K_c/\tau_c)w_l$$
$$u = -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4$$

which, for the closed-loop system, can be written in state-space form as:

$$\mathbf{x} = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 \\ -k_1K_c/\tau_c & -(1+k_2K_c)/\tau_c & -k_3K_c/\tau_c & -k_4K_c/\tau_c & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/\tau_l \end{bmatrix} \mathbf{x} + \begin{Bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 0 \end{Bmatrix} r + \begin{Bmatrix} 0 \\ K_c/\tau_c \\ 0 \\ 0 \\ 0 \end{Bmatrix} w_c + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ K_l/\tau_l \end{Bmatrix} w_l$$

The simulation results are shown in Figure 12.12 for a moderate deceleration case. The controller maintains the desired range (i.e., 30 meters), despite changes in the lead-vehicle velocity.

```
% Ex12_5.m
g=9.81; uw=0.0; u0=30.0; rho =1.202;
Theta=0.0; ThetaPrime=0.0;
% Controlled vehicle parameters:
mc=1000.0; Cdc=0.5; Arc=1.5; fc=0.015;
Kc=(1/(rho*Cdc*Arc*(u0+uw)));Tc=mc*Kc;
wc0=mc*g*(fc*sin(Theta)-cos(Theta))*ThetaPrime; Fc=(u0/Kc);
% Lead vehicle parameters (typical):
ml=1500.0;Cdl=0.6;Arl=1.95;fl=0.015;
Kl=(1/(rho*Cdl*Arl*(u0+uw)));Tl=ml*Kl;
Fl=(u0/Kl);
wl0=(ml*g*(fl*sin(Theta)-cos(Theta))*ThetaPrime);
t=[0:0.1:20]';
U0=Fc*ones(size(t));          % Nominal control force
wl= -800*(1+0.01*t);          % Ramp function
wc=wc0*ones(size(t));
disturbance=[U0 wc wl];
% 4-state system for controller-design:
Aa=[0 -1 0 0;
    0 -1/Tc 0 0;
    1 0 0 0;
    0 0 1 0];
Ba=[0;Kc/Tc;0;0];
% Controller design:
pc=[roots([1 2*0.9*0.4 0.4^2]); -1.08; -1.18];
K=place(Aa,Ba,pc);
% Closed-loop simulation (5 states, keep track of vl (x5)):
Ac=[0 -1 0 0 1;
    -K(1)*Kc/Tc -(1+K(2)*Kc)/Tc -K(3)*Kc/Tc -K(4)*Kc/Tc 0;
    1 0 0 0 0;
    0 0 1 0 0;
      0 0 0 0 -1/Tl];
Bc=[0 0 0;
    0 Kc/Tc 0;
      -1 0 0;
      0 0 0;
    0 0 Kl/Tl];
% outputs: x1 (range), vc and vl
Cc=[1 0 0 0 0;0 1 0 0 0; 0 0 0 0 1]; Dc=zeros(3,3);
r=30.0*ones(size(t));
disturbance=[r U0+wc wl]; xc0=[30 u0 0 -
(u0/Kc+K(1)*30+K(2)*u0)/K(4) u0];
[yc,xc]=lsim(Ac,Bc,Cc,Dc,disturbance,t,xc0);
subplot(211), plot(t,yc(:,1)); title('Range');
xlabel('Time (sec)'); grid;
subplot(212), plot(t,yc(:,3), 'r',t,yc(:,2),'-.b');
title('Vehicle speed (m/sec)');
```

```
xlabel('Time (sec)'); grid;
legend('Vl', 'Vc'); pause;
clf, subplot(211)
u= U0-K(1)*xc(:,1)-K(2)*xc(:,2)-K(3)*xc(:,3)-K(4)*xc(:,4);
plot(t, u'); title('Control Force (N)');
xlabel('Time (sec)'); grid
```

PROBLEMS

1. Grades on interstate highways usually are limited to about 4 percent or less (i.e., rise over run, or the tan θ is about 0.04). On major roads, grades may occasionally reach 10 to 20 percent. In the simulation in Example 12.1, a unit-step disturbance input was used and caused a transient change in the speed with the PI control. Rerun the simulation in Example 12.1 for the PI cruise controller with grade-disturbance inputs of (a) 15 percent, and (b) −15 percent. Use the following general expression for the grade-disturbance input:

$$d = mg(f \sin\theta_0 - \cos\theta_0)\theta'$$

Use the same parameter values as in Example 12.1. Are the observed deviations from the desired speed "acceptable" for this range of extreme disturbance inputs?

2. In this problem, carry out several calculations to assess whether an adaptive version of the fixed-gain PI controller in Example 12.1 is desirable. Based on the results of your calculations in Parts (a) and/or (b), do you recommend the design of an adaptive PI cruise controller?

(a) Consider the fixed-gain PI cruise controller in Example 12.1. This was designed for $m = 1{,}000$ kg, $u_w = 2$ m/s, and $u_o = 20$ m/s. Apply this controller to the system with $m = 1{,}250$ kg, $u_w = 5$ m/s, and $u_o = 30$ m/s. Compared to the performance specifications used in Example 12.1, how does this controller perform?

(b) For cruise control (see Example 12.1), linearization was used to obtain the following time constant and gain:

$$\tau = (m/(\rho C_d A(u_o + u_w)));$$
$$K = (1/(\rho C_d A(u_o + u_w)));$$

For the nominal values given in this chapter and following Eq. (12.2), plot the values of τ and K for each of the following cases:

(1) $m = [1{,}000{:}1{,}250]$ (i.e., 25 percent variation in m from curb weight to loaded weight)
(2) $u_w = [-10{:}10]$ (i.e., ±10m/s [about ±22 mph] head/tail wind)
(3) $u_o = [20{:}30]$ (i.e., 10m/s change in forward velocity)

3. Example 12.5 considers an ACC to maintain a desired headway based on measurements of d and v_c only (i.e., no inputs from other vehicles or from the roadway infrastructure). The control problem is illustrated in block-diagram form in Figure 12.11. Note that the reference input, r, would be computed as a function of the

vehicle forward velocity to maintain safe headway. Also note that the velocity, v_l, of the lead vehicle is not known and acts as a disturbance input. The control u on Vehicle c must be calculated to maintain the desired headway. A feedback controller is designed for this system based on the following state equations:

$$\begin{aligned}
\dot{x}_1 &= -x_2 + v_l \\
\dot{x}_2 &= -(1/\tau_c)x_2 + (K_c/\tau_c)(u + w_c) \\
\dot{x}_3 &= (d - r) = (x_1 - r) \\
\dot{x}_4 &= x_3 \\
u &= -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4
\end{aligned}$$

This controller feeds back the measured values $x_1 = d$ and $x_2 = v_c$, as well as x_3, the integral of the error, $e = d - r$, and x_4, the double integral of the error. The two integrals of error are used because it is assumed that the lead-vehicle velocity, v_l, (which acts as a disturbance to be rejected by the control law) is well represented as a ramp-type function in time. The controller gains, k_i, can be determined by a variety of methods, and a pole-placement design technique is used in the MATLAB program in Example 12.5. The closed-loop poles are placed at $s_{1,2} = -\zeta\omega_n \pm \omega_n\sqrt{1-\zeta^2}\,j$ and $s_{3,4} = -\alpha\zeta\omega_n$. The values $\zeta = 0.9$, ω_n =0.4, and $\alpha = 3.0$ are used in the MATLAB simulations in Example 12.5.

Repeat Example 12.5 but with a simpler control law that does not feed back the double integral of the error (i.e., x_4) and has the form $u = -k_1x_1 - k_2x_2 - k_3x_3$. Compare the results for a ramp-disturbance input, v_l, to those obtained using the previous control law, $u = -k_1x_1 - k_2x_2 - k_3x_3 - k_4x_4$.

REFERENCES

Druzhinina, M., A. G. Stefanopoulou, and L. Moklegaard, 2002a, "Speed Gradient Approach to Longitudinal Control of Heavy-Duty Vehicles Equipped with Variable Compression Brake," *IEEE Transactions on Control System Technology*, Vol. 10, No. 2, March, pp. 209–21.

Druzhinina, M., A. G. Stefanopoulou, and L. Moklegaard, 2002b, "Adaptive Continuously Variable Compression Braking Control for Heavy-Duty Vehicles," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 124, No. 3, September.

Fancher, P., H. Peng, and Z. Bareket, 1996, "Comparison of Three Control Algorithms on Headway Control for Heavy Trucks," *Vehicle System Dynamics*, Vol. 25 Suppl., pp. 139–51.

Fancher, P., H. Peng, Z. Bareket, C. Assaf, and R. Ervin, 2001, "Evaluating the Influences of Adaptive Cruise Control Systems on the Longitudinal Dynamics of Strings of Highway Vehicles," *Proceedings of the 2001 IAVSD Conference*, Copenhagen, Denmark, August.

Himmelspach, T., and W. Ribbens, 1989, "Radar Sensor Issues for Automotive Headway Control Applications," *Fall 1989 Summary Report for the Special Topics Course in IVHS, Appendix G*, University of Michigan.

Ioannou, P. A., and C. C. Chien, 1993, "Autonomous Intelligent Cruise Control," *IEEE Transactions on Vehicular Technology*, Vol. 42, No. 4, pp. 657–72.

Klein, R. H., and J. R. Hogue, 1980, "Effects of Crosswinds on Vehicle Response – Full-Scale Tests and Analytical Predictions," SAE Paper No. 800848. Detroit, MI.

Liang, C., and H. Peng, 1999, "Optimal Adaptive Cruise Control with Guaranteed String Stability," *Vehicle System Dynamics*, Vol. 32, No. 4–5, November, pp. 313–30.

Liubakka, M. K., D. S. Rhode, J. R. Winkelman, and P. V. Kokotovic, 1993, "Adaptive Automotive Speed Control," *IEEE Transactions on Automatic Control*, Vol. 38, No. 7, July, pp. 1011–20.

Liubakka, M. K., J. R. Winkelman, and P. V. Kokotovic, 1991, "Adaptive Automotive Speed Control," *Proceedings of the American Control Conference*, Boston, MA, June, pp. 439–40.

Oda, K., H. Takeuchi, M. Tsujii, and M. Ohba, 1991, "Practical Estimator for Self-Tuning Automotive Cruise Control," *Proceedings of the American Control Conference*, Boston, MA, June, pp. 2066–71.

Okuno, A., A. Kutami, and K. Fujita, 1990, "Towards Autonomous Cruising on Highways," SAE Paper No. 901484.

Rajamani, R., and Chunyu Zhu, "Semi-Autonomous Adaptive Cruise Control Systems," 1999, *Proceedings of the American Control Conference*, Vol. 2, June, pp. 1491–5.

Swaroop, D., and K. R. Rajagopal, 1999, "Intelligent Cruise Control Systems and Traffic Flow Stability," *Transportation Research*, Part C, Vol. 7, pp. 329–52.

Tsujii, M., H. Takeuchi, K. Oda, and M. Ohba, 1990, "Application of Self-Tuning to Automotive Cruise Control," *Proceedings of the American Control Conference*, San Diego, CA, May, pp. 1843–8.

Xu, Z., and P. A. Ioannou, 1994, "Adaptive Throttle Control for Speed Tracking," *Vehicle System Dynamics*, Vol. 23, No. 4, May, pp. 293–306.

Yoshimoto, K., H. Tanabe, and M. Tanaka, 1994, "Speed Control Algorithm for an Automated Driving Vehicle," *Proceedings of the AVEC '94*, pp. 408–13.