Feedback Examples: Cruise Control

Karl J. Åström Richard M. Murray

2019-08-18

This document describes the speed control system that is used as a running example through the text. Material that is pulled from the book is colored in black and marked by a heading indicating where the material came from. Material in green is contained only in this document. Additional supplemental information that may appear in some of textbook materials (extra text, solutions, etc) is colored in blue. Equation and figure numbers of the form *m.nn* refer to the main text. .

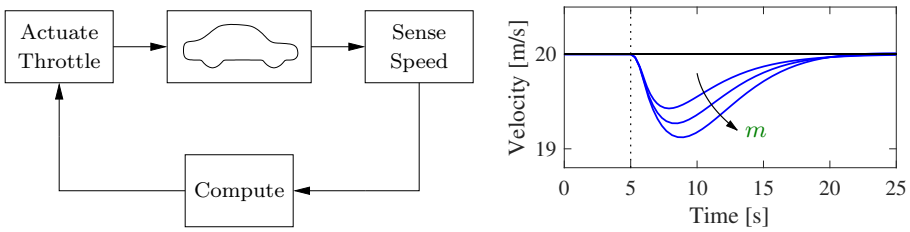
1 Introduction

cruise-intro

Section 1.5 In this system, the velocity of a vehicle is controlled by adjusting the amount of gas flowing to the engine. Simple *proportional-integral* (PI) feedback is used to make the amount of gas depend on both the error between the current and the desired velocity and the integral of that error. The plot on the right shows the effect of this feedback when the vehicle travels on a horizontal road and it encounters an uphill slope. When the slope changes, the car decelerates due to gravity forces and the velocity initially increases. The velocity error is sensed by the controller, which acts to restore the velocity to the desired value by increasing the throttle. The figure also shows what happens when the same controller is used for a different masses of the car, which might result from having a different number of passengers or towing a trailer. Notice that the steady-state velocity of the vehicle always approaches the desired velocity and achieves that velocity within approximately 15 s, independent of the mass (which varies by a factor of $\pm$ 25%), Thus feedback improves both performance and robustness of the system.

cruise-redesign

Exercise 1.4 Download the MATLAB code used to produce simulations for the cruise control system in Figure 1 from the companion web site. Using trial and error, change the parameters of the control law so that the overshoot in



(a) System diagram (b) Response to change in road slope

Figure 1: A feedback system for controlling the velocity of a vehicle. In the block diagram on the left, the velocity of the vehicle is measured and compared to the desired velocity within the "Compute" block. Based on the difference in the actual and desired velocities, the throttle (or brake) is used to modify the force applied to the vehicle by the engine, drivetrain, and wheels. The figure on the right shows how the velocity changes when the car travels on a horizontal road and the slope of the road changes to a constant uphill slope. The three different curves correspond to differing masses of the vehicle, between 1200 and 2000 kg, demonstrating that feedback can indeed compensate for the changing slope and that the closed loop system is robust to a large change in the vehicle characteristics.

speed is not more than 1 m/s for a vehicle with mass $m = 1200$ kg. Does the same controller work if we set $m = 2000$ kg?

Solution. [Cole Lepine, Feb 08] Although there are more than one set of parameters that work, here are ones that do: $k_p = 2$ and $k_i = 0.1$. Figure 2 shows a

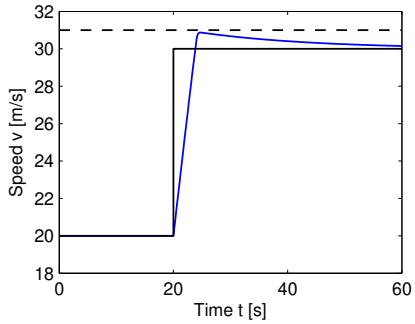


Figure 2: Step response for the redesigned cruise control system in Exercise 1.4. This shows that the overshoot is less than 1 m/s for the given gains.

graph of how the system responds to a change in reference speed from 20 m/s to 30 m/s at $t = 20$ s. The MATLAB code used to generate the plot is

```
plot(Time, Vel);
xlabel('time (sec)');
```

```
ylabel('speed (m/s)');
```

(to be used after running the SIMULINK model).

Exercise 1.9 Download the file "`cruise_ctrl.mdl`" from the companion web site. It contains a SIMULINK model of a simple cruise controller, similar to the one described in Section 1.5. Figure out how to run the example and plot the vehicle's speed as a function of time. cruise-mlintro

(a) Leaving the control gains at their default values, plot the response of the system to a step input and measure the time it takes for the system error to settle to within 5% of commanded change in speed (i.e., 0.5 m/s).

(b) By manually tuning the control gains, design a controller that settles at least 50% faster than the default controller. Include the gains you used, a plot of the closed loop response, and describe any undesirable features in the solution you obtain.

All plots should included a title, labeled axes (with units), and reasonable axis limits.

*Instructor note:*The exercise is a variation of Exercise 1.4 above. The purpose of these problem is to give students some familiarity with MATLAB and SIMULINK. The instructor may want to indicate in the problem that students shouldn't worry if they don't yet know how the control law works or why it does what it does.

Solution. [Caltech CDS 101/110 TAs, 2004–2006; Cole Lepine, Feb 08]

(a) The red curve in Figure 3 depicts speed as a function of time for the default gains. It was created by first running the simulation, then in the MATLAB command window running the command:

```
plot(Time, Vel);
xlabel('time (sec)');
ylabel('speed (m/s)');
```

We calculate the settling time by finding the first time after which the system remains within the five percent bound specified. This occurs at about 34 *second* and can be found after running the simulation with the commands:

```
settle = find (abs(flipud(Vel) - 30) > .5);
timerev = flipud(Time);
timerev(settle(1));
```

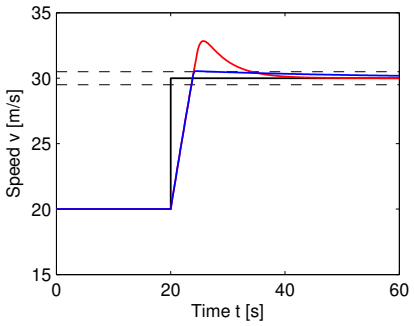


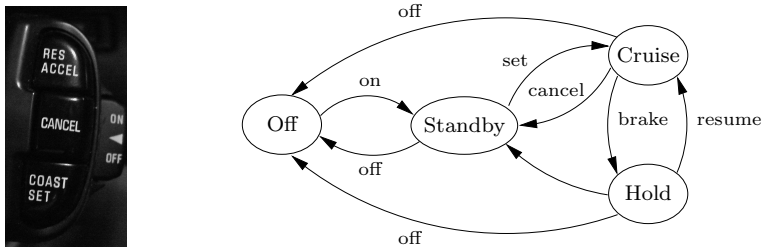
Figure 3: Response of cruise control system to step input. The red curve shows the response using original controller gains for Exercise 1.9. This shows that the settling time is about 15 s for the given gains. The blue curve show the modified gains, which give a settling time is about 7 s.

This searches through the velocity vector backwards to find the first out of bounds value, and then computes the time at which that point occurs. Finally we subtract the time at which the system input began, i.e. $t = 20$ s, and obtain a settling time of 14 s.

(b) We want to modify the control parameters so that the system settles 50% faster, or within 7 s. In this example, increasing the proportional gain resulted in faster, sharper response. Increasing the integral gain does not have much effect on the responsiveness of the system, and for sufficiently large values induces oscillatory behavior, which is undesirable.

A choice of gains that accomplishes the desired behavior is $k_i = 0.1$ and $k_p = 3.3$. This has a settling time (calculated as above) of 7 s. The blue curve in Figure 3 shows the system response for these gains. With the exception of the fact that this performance may be difficult to achieve in reality due to the constraints discussed below, there are no undersirable features of the new response as compared to the default performance.

Of course, we can increase the proportional gain even further, and obtain a faster response time. Extremely large values could result in extremely fast response. This control strategy is limited by physical realism: there's only so much power an engine can provide, only so much traction on any given surface, and only so much acceleration that a passenger (or designer) will tolerate. Certainly, attempting a 10 m/s increase in speed in under a second is an ambitious undertaking, even though it is easy enough to dial up k_p sufficiently in order to achieve this within our simple model.



(a) User interface (b) State machine

Figure 4: Finite state machine for cruise control system. The figure on the left shows some typical buttons used to control the system. The controller can be in one of four modes, corresponding to the nodes in the diagram on the right. Transition between the modes is controlled by pressing one of the four buttons on the cruise control interface: on/off, set, resume, or cancel.

Section 1.7 The basic control function in a cruise controller, such as the one shown in Figure 1, is to keep the velocity constant. It is typically done with a PI controller. The controller normally operates in automatic mode but it is is necessary to switch it off when braking, accelerating, or changing gears. The cruise control system has a human–machine interface that allows the driver to communicate with the system. There are many different ways to implement this system; one version is illustrated in Figure 4a. The system has four buttons: on/off, coast/set, resume/accelerate, and cancel. The operation of the system is governed by a finite state machine that controls the modes of the PI controller and the reference generator, as shown in Figure 4b. cruise-logic

The finite state machine has four modes: off, standby, cruise, and hold. The state changes depending on actions of the driver who can brake, accelerate, and operate using the buttons. The on/off switch moves the states between off and standby. From standby the system can be moved to cruise by pushing the set/coast button. The velocity reference is set as the velocity of the car when the button is released. In the cruise state the operator can change the velocity reference; it is increased using the resume/accelerate button and decreased using the set/coast button. If the driver accelerates by pushing the gas pedal the speed increases, but it will go back to the set velocity when the gas pedal is released. If the driver brakes then the car slows, and the cruise controller goes into hold but it remembers the setpoint of the controller. It can be brought to the cruise state by pushing the resume/accelerate button. The system also moves from cruise mode to standby if the cancel button is pushed. The reference for the velocity controller is remembered. The system goes into off mode by pushing on/off when

the system is engaged.

The PI controller is designed to have good regulation properties and to give good transient performance when switching between resume and control modes.

2 Feedback Principles

Exercise 2.18 Consider the cruise control example discussed in Section 1.5, with cruise-fbkparam

RMM: Nonstandard use of parameters compared to other cruise control examples [RMM, 18 Aug 2019]

$$m\dot{v} = -av + u + w$$

where u is the control input (force applied by engine) and w the disturbance input (force applied by hill, etc.), which will be ignored below ($w = 0$). An *open loop* control strategy to achieve a given reference speed v_{ref} would be to choose

$$u = \hat{a} v_{\text{ref}}$$

where $\hat{a}$ is your estimate of a, which may not be accurate.

(a) Compute the steady-state response for both the open loop strategy above, and for the feedback law

$$u = -k_{\text{p}}(v - v_{\text{ref}})$$

and compare the steady state (with $w = 0$) as a function of $\beta = a/\hat{a}$ when $k_{\text{p}} = 10\hat{a}$. (You should solve the problem analytically, and then plot the response $v_{\text{ss}}/v_{\text{ref}}$ as a function of β.)

(b) Now consider a proportional-integral (PI) control law

$$u = -k_{\text{p}}(v - v_{\text{ref}}) - k_{\text{i}} \int_0^t (v - v_{\text{ref}}) dt$$

and again compute the steady-state solution (assuming stability) and compare the response with the proportional gain case from above. (Note that if you define $q = \int_0^t (v - v_{\text{ref}}) dt$ then $\dot{q} = v - v_{\text{ref}}$.)

Solution. (a) In steady state, the time derivatives must be zero, thus $v_{\text{ss}} = u/a$. For open loop control, then

$$v_{\text{ss}} = \beta^{-1} v_{\text{ref}}$$

For the proportional feedback law, then from lecture notes,

$$v_{\text{ss}} = \frac{k_{\text{p}}}{a + k_{\text{p}}} v_{\text{ref}}$$

If $k_{\text{p}} = 10\hat{a}$, then $v_{\text{ss}} = 10/(10 + \beta) v_{\text{ref}}$

(b) In steady state, the time derivatives must be zero, and $\dot{q} = 0$ implies $v_{ss} = v_{ref}$ in steady state for any value of k_i, k_p, a, m, assuming the system is stable.

Exercise 2.6 A simple model for the relation between speed v and throttle u for a car is given by the transfer function cruisecon

$$G_{vu} = \frac{b}{s+a}$$

where $a = 0.01$ rad/s and $b = 1.32$ m/s^2 (see Section 4.1 and Example 6.11 for more details). The control signal is normalized to the range $0 \le u \le 1$. Design a PI controller for the system that gives a closed loop system with the characteristic polynomial

$$a_{cl}(s) = s^2 + 2\zeta_c\omega_c s + \omega_c^2.$$

What are the consequences of choosing different values of the design parameters ζ_c and ω_c? Use your judgment to find suitable values. Hint: Investigate maximum acceleration and maximum velocity for step changes in the velocity reference.

Solution. The transfer function of the process and the controller are

$$P(s) = G_{vu} = \frac{b}{s+a}, \qquad C(s) = k_p + \frac{k_i}{s} = \frac{k_p s + k_i}{s}$$

The loop transfer function is

$$P(s)C(s) = \frac{b(k_p s + k_i)}{s(s+a)},$$

and the closed loop characteristic polynomial is

$$s(s+a) + b(k_p s + k_i) = s^2 + (a + bk_p)s + bk_i$$

Identification of the coefficients of this polynomial with those of a_{cl} gives

$$a + bk_p = 2\zeta_c\omega_c, \qquad bk_i = \omega_c^2.$$

Solving these equations for the controller gains and inserting the numerical values give

RMM: Wrong numerical values

$$k_p = 2\zeta_c\omega_c - 0.025, \qquad k_i = \omega_c^2$$

To find reasonable values we will first analyze the model

$$\frac{dv}{dt} = -av + bu,$$

with full throttle $u = 1$ the steady-state velocity is $v = \frac{b}{a} = 40$ m/s $= 144$ km/*hour*. The time constant of the open loop system is $T_{\text{ol}} = 1/a = 40$ s.

Assume that the car is running at 20 m/s with half throttle $u = 0.5$. The steady-state change in throttle Δu required to obtain a velocity change Δv is $\Delta u = \frac{a}{b}\Delta v$. The instantaneous change Δu in throttle when commanding a speed increase Δv is $k_{\text{p}}\Delta v$. Requiring that the initial change equals the steady-state change gives $k_{\text{p}} = a/b = 0.025$. To have a system with smooth response we choose critical damping $\zeta_{\text{c}} = 1$ which gives $\omega_{\text{c}} = 0.05$ and $k_{\text{i}} = 0.0025$.

3 System Modeling

Exercise 3.16 [Contributed by D. MacMartin, 2011] Consider the cruise-control example discussed in class,† with cruise-openvspi

RMM: Update to refer to Intro? Update parameters

$$m\dot{v} = -av + u + w$$

where u is the control input (force applied by engine) and w the disturbance input (force applied by hill, etc.), which will be ignored below ($w = 0$). An *open loop* control strategy to achieve a given reference speed v_{ref} would be to choose

$$u = \hat{a}v_{\text{ref}}$$

where $\hat{a}$ is your estimate of a, which may not be accurate.

(a) Compute the steady-state response for both the open loop strategy above, and for the feedback law

$$u = -k_{\text{p}}(v - v_{\text{ref}})$$

and compare the steady-state (with $w = 0$) as a function of $\beta = a/\hat{a}$ when $k_{\text{p}} = 10\hat{a}$. (You should solve the problem analytically, and then plot the response $v_{\text{ss}}/v_{\text{ref}}$ as a function of β.)

(b) Now consider a proportional-integral (PI) control law

$$u = -k_{\text{p}}(v - v_{\text{ref}}) - k_{\text{i}}\int_0^t (v - v_{\text{ref}})dt$$

and again compute the steady-state solution (assuming stability) and compare the response with the proportional gain case from above. (Note that if you define $q = \int_0^t (v - v_{\text{ref}})dt$ then $\dot{q} = v - v_{\text{ref}}$.)

(c) Next, simulate the response of the system (using `ode45` in Matlab or similar) with the PI control law above with $m = 1$, $a = 0.1$, $w = 0$, and "input" to the system of $v_{\text{ref}} = \sin(\omega t)$, for ω=0.01, 0.1, 1, and 10 rad/sec. In each case, you should simulate at least 10 cycles; after some initial transient, the response should be periodic. Compute the peak-to-peak amplitude of the final period for the error $v - v_{\text{ref}}$, and plot this as a function of frequency on a log-log scale, for the following control gains:

i. $k_{\text{p}} = 1$, $k_{\text{i}} = 0$
ii. $k_{\text{p}} = 1$, $k_{\text{i}} = 1$
iii. $k_{\text{p}} = 1$, $k_{\text{i}} = 10$

(If you want to see interesting behaviour, simulate the final case at $\omega = 3.3$ rad/sec as well.)

Solution. (a) In steady state, the time derivatives must be zero, thus $v_{\text{ss}} = u/a$. For open loop control, then

$$v_{\text{ss}} = \beta^{-1} v_{\text{ref}}$$

For the proportional feedback law, then from lecture notes,

$$v_{\text{ss}} = \frac{k_{\text{p}}}{a + k_{\text{p}}} v_{\text{ref}}$$

If $k_{\text{p}} = 10\hat{a}$, then $v_{\text{ss}} = 10/(10 + \beta) v_{\text{ref}}$

(b) In steady state, the time derivatives must be zero, and $\dot{q} = 0$ implies $v_{\text{ss}} = v_{\text{ref}}$ in steady state for any value of k_{i}, k_{p}, a, m, assuming the system is stable.

4 Examples

Section 4.1 The cruise control system of a car is a common feedback system encountered in everyday life. The system attempts to maintain a constant velocity in the presence of disturbances primarily caused by changes in the slope of a road. The controller compensates for these unknowns by measuring the speed of the car and adjusting the throttle appropriately. cruise-modeling

To model the system we start with the block diagram in Figure 5. Let v be the speed of the car and v_{r} the desired (reference) speed. The controller, which typically is of the proportional-integral (PI) type described briefly in Chapter 1,

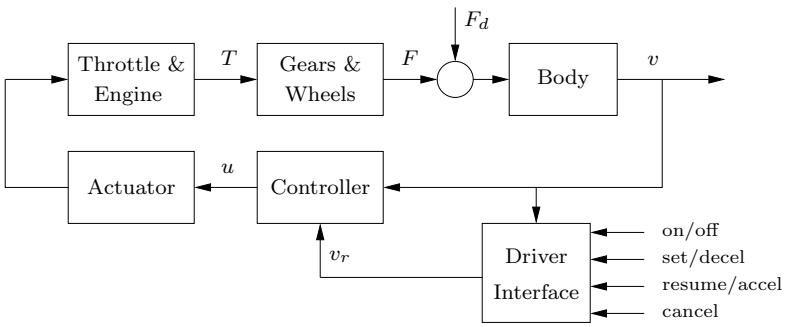


Figure 5: Block diagram of a cruise control system for an automobile. The throttle-controlled engine generates a torque T that is transmitted to the ground through the gearbox and wheels. Combined with the external forces from the environment, such as aerodynamic drag and gravitational forces on hills, the net force causes the car to move. The velocity of the car v is measured by a control system that adjusts the throttle through an actuation mechanism. A driver interface allows the system to be turned on and off and the reference speed v_r to be established.

receives the signals v and v_r and generates a (normalized) control signal u that is sent to an actuator that controls the throttle position. The throttle in turn controls the torque T delivered by the engine, which is transmitted through the gears and the wheels, generating a force F that moves the car. There are disturbance forces F_d due to variations in the slope of the road, the rolling resistance, and aerodynamic forces. The cruise controller also has a human–machine interface that allows the driver to set and modify the desired speed. There are also functions that disconnect the cruise control when the brake is touched.

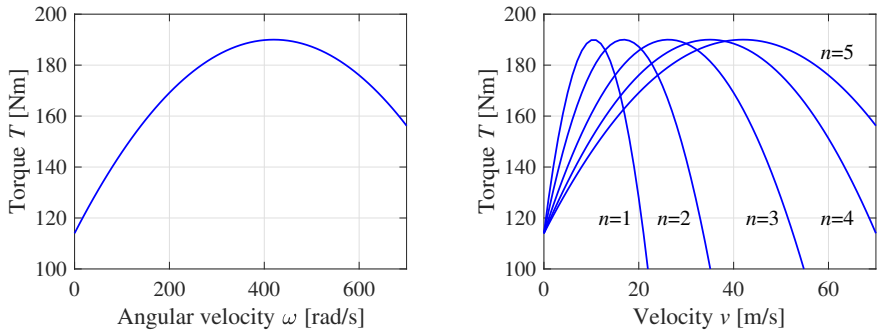
The system has many individual components—actuator, engine, transmission, wheels, and car body—and a detailed model can be very complicated. In spite of this, the model required to design the cruise controller can be quite simple.

To develop a mathematical model we start with a force balance for the car body. Letting m be the total mass of the car (including passengers), the equation of motion of the car is simply

$$m\frac{dv}{dt} = F - F_\mathrm{d}. \tag{1}$$

Typical values for the mass of a car are in the range of 1000–2000 kg (we will use 1600 kg here).

The force F is generated by the engine, whose torque is proportional to the rate of fuel injection, which is itself proportional to a control signal $0 \leq u \leq 1$ that controls the throttle position. The torque also depends on engine speed ω.



(a) Torque versus engine speed (b) Torque versus car speed

Figure 6: Torque curves for typical car engine. The graph on the left shows the torque generated by the engine as a function of the angular velocity of the engine, while the curve on the right shows torque as a function of car speed for different gears.

A simple representation of the torque at full throttle is given by the torque curve

$$T(\omega) = T_{\mathrm{m}} \left(1 - \beta \left(\frac{\omega}{\omega_{\mathrm{m}}} - 1 \right)^2 \right), \tag{2}$$

where the maximum torque T_{m} is obtained at engine speed ω_{m}. Typical parameters are $T_{\mathrm{m}} = 190$ Nm, $\omega_{\mathrm{m}} = 420$ rad/s (about 4000 RPM), and $\beta = 0.4$. Let n be the gear ratio and r the wheel radius. The engine speed is related to the velocity through the expression

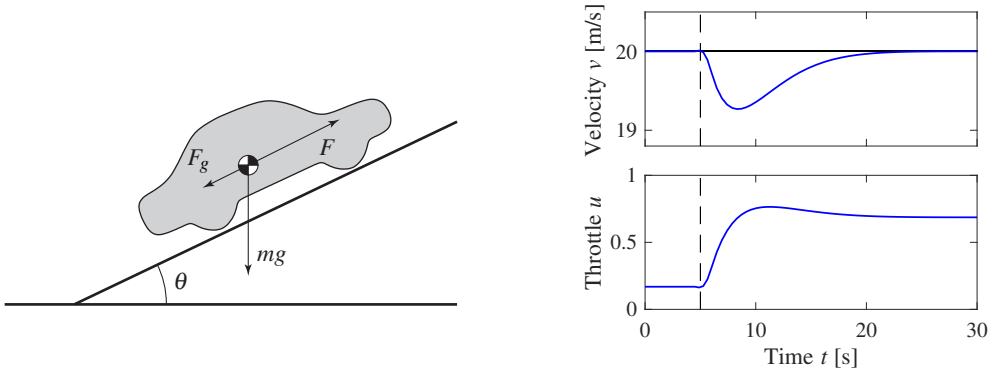
$$\omega = \frac{n}{r} v =: \alpha_n v,$$

and the driving force can be written as

$$F = \frac{nu}{r} T(\omega) = \alpha_n u T(\alpha_n v).$$

Typical values of α_n for gears 1 through 5 are $\alpha_1 = 40$, $\alpha_2 = 25$, $\alpha_3 = 16$, $\alpha_4 = 12$, and $\alpha_5 = 10$. The inverse of α_n has a physical interpretation as the *effective wheel radius*. Figure 6 shows the torque as a function of engine speed and vehicle speed. The figure shows that the effect of the gear is to "flatten" the torque curve so that nearly full torque can be obtained over almost the whole speed range.

The disturbance force F_{d} has three major components: F_{g}, the forces due to gravity; F_{r}, the forces due to rolling friction; and F_{a}, the aerodynamic drag. Letting the slope of the road be θ, gravity gives the force $F_{\mathrm{g}} = mg \sin\theta$, as



(a) Effect of gravitational forces (b) Closed loop response

Figure 7: Car with cruise control encountering a sloping road. A schematic diagram is shown in (a), and (b) shows the response in speed and throttle when a slope of 4° is encountered. The hill is modeled as a net change of 4° in hill angle θ, with a linear change in the angle between $t = 5$ and $t = 6$. The PI controller has proportional gain $k_\mathrm{p} = 0.5$ and integral gain $k_\mathrm{i} = 0.1$.

illustrated in Figure 7a,† where $g = 9.8\ \mathrm{m/s^2}$ is the gravitational constant. A simple model of rolling friction is

RMM: Check page alignment in final printing

$$F_\mathrm{r} = mgC_\mathrm{r}\,\mathrm{sgn}(v),$$

where C_r is the coefficient of rolling friction and $\mathrm{sgn}(v)$ is the sign of v (± 1) or zero if $v = 0$. A typical value for the coefficient of rolling friction is $C_\mathrm{r} = 0.01$. Finally, the aerodynamic drag is proportional to the square of the speed:

$$F_\mathrm{a} = \frac{1}{2}\rho C_\mathrm{d} A|v|v,$$

where ρ is the density of air, C_d is the shape-dependent aerodynamic drag coefficient, and A is the frontal area of the car. Typical parameters are $\rho = 1.3\ \mathrm{kg/m^3}$, $C_\mathrm{d} = 0.32$, and $A = 2.4\ \mathrm{m^2}$.

Summarizing, we find that the car's speed can be modeled by

$$m\frac{dv}{dt} = \alpha_n u T(\alpha_n v) - mgC_\mathrm{r}\,\mathrm{sgn}(v) - \frac{1}{2}\rho C_\mathrm{d} A|v|v - mg\sin\theta, \tag{3}$$

where the function T is given by equation (2). The model (3) is a dynamical system of first order. The state is the car velocity v, which is also the output. The input is the signal u that controls the throttle position, and the disturbance is the force $F_\mathrm{d} = mg\sin\theta$, which depends on the slope of the road. The system

is nonlinear because of the torque curve, the gravity term, and the nonlinear character of rolling friction and aerodynamic drag. There can also be variations in the parameters; e.g., the mass of the car depends on the number of passengers and the load being carried in the car.

We add to this model a feedback controller that attempts to regulate the speed of the car in the presence of disturbances. We use a proportional-integral controller, which has the form

$$u(t) = k_{\mathrm{p}} e(t) + k_{\mathrm{i}} \int_0^t e(\tau)\, d\tau.$$

This controller can itself be realized as an input/output dynamical system by defining a controller state z and implementing the differential equation

$$\frac{dz}{dt} = v_{\mathrm{r}} - v, \qquad u = k_{\mathrm{p}}(v_{\mathrm{r}} - v) + k_{\mathrm{i}} z, \tag{4}$$

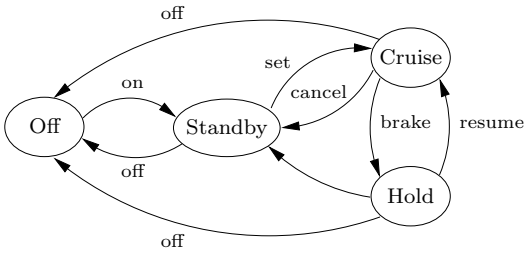
where v_{r} is the desired (reference) speed. As discussed briefly in Section 1.6, the integrator (represented by the state z) ensures that in steady state the error will be driven to zero, even when there are disturbances or modeling errors. (The design of PI controllers is the subject of Chapter 11.) Figure 7b shows the response of the closed loop system, consisting of equations (3) and (4), when it encounters a hill. The figure shows that even if the hill is so steep that the throttle changes from 0.17 to almost full throttle, the largest speed error is less than 1 m/s, and the desired velocity is recovered after 20 s.

Many approximations were made when deriving the model (3). It may seem surprising that such a seemingly complicated system can be described by the simple model (3). It is important to make sure that we restrict our use of the model to the uncertaintyłemon conceptualized in Figure 3.5b. The model is not valid for very rapid changes of the throttle because we have ignored the details of the engine dynamics, neither is it valid for very slow changes because the properties of the engine will change over the years. Nevertheless the model is very useful for the design of a cruise control system. As we shall see in later chapters, the reason for this is the inherent robustness of feedback systems: even if the model is not perfectly accurate, we can use it to design a controller and make use of the feedback in the controller to manage the uncertainty in the system.

The cruise control system also has a human–machine interface that allows the driver to communicate with the system. There are many different ways to implement this system; one version is illustrated in Figure 8. The system has four buttons: on-off, set/decelerate, resume/accelerate, and cancel. The operation of

PUP: Try to find picture that matches terms used in Figure 5 (minor)

(a) Cruise control interface



(b) Finite state machine

Figure 8: Finite state machine for cruise control system. The figure on the left shows some typical buttons used to control the system. The controller can be in one of four modes, corresponding to the nodes in the diagram on the right. Transition between the modes is controlled by pressing one of the five buttons on the cruise control interface: on, off, set, resume, or cancel.

the system is governed by a finite state machine that controls the modes of the PI controller and the reference generator.

Supplement

The controller can operate in two ways: in the normal cruise control mode and in a tracking mode, where the integral is adjusted to match given process inputs and outputs. The tracking mode is introduced to avoid switching transients when the system is controlled manually. The generator for the reference signal has three modes: a normal control mode when the output is controlled by the set/accelerate and resume/decelerate buttons, a tracking mode, and a hold mode where the reference is held constant.

To control the overall operation of the controller and reference generator, we use a finite-state-machine with four states: off, standby, cruise, and hold. The states of the controller and the reference generator in the different modes are given in Figure 8. The cruise mode is the normal operating mode where the speed can be then be decreased by pushing set/decelerate and increased by pushing the resume/accelerate. When the system is switched on it goes to standby mode. The cruise mode is activated by pushing the set/accelerate button. If the brake is touched or if the gear is changed, the system goes into hold mode and the current velocity is stored in the reference generator. The controller is then switched to tracking mode and the reference generator is switched to hold mode, where it holds the current velocity. Touching the resume button then switches the system to cruise mode. The system can be switched to standby mode from any state by pressing the cancel button.

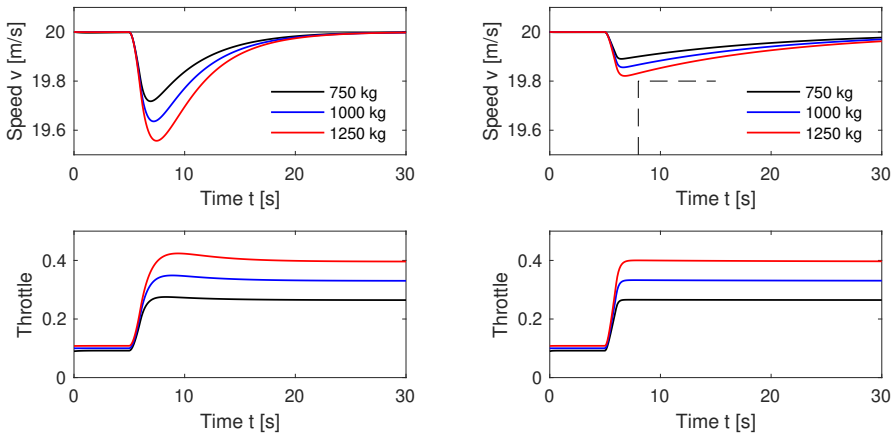
The PI controller should be designed to have good regulation properties and to give good transient performance when switching between resume and control

modes. Implementation of controllers and reference generators will be discussed more fully in Chapter 11.

The use of control in automotive systems goes well beyond the simple cruise control system described here. Applications include emissions control, traction control, power control (especially in hybrid vehicles), and adaptive cruise control. Many automotive applications are discussed in detail in the book by Kiencke and Nielsen [KN00] and in the survey papers by Powers et al. [BP96, PN00]. New vehicles coming on the market also include many "self-driving" features, which represent even more complex feedback systems.

Exercise 4.1 Consider the cruise control example described in Section 4.1. Build a simulation that re-creates the response to a hill shown in Figure 7b and show the effects of increasing and decreasing the mass of the car by 25%. Redesign the controller (using trial and error is fine) so that it returns to within 1% of the desired speed within 3 s of encountering the beginning of the hill. cruise-hillsim

Solution. [Cole Lepine, Feb. 2008] Figure 9a depicts speed as a function of time with the default gains for a hill encounter at $t = 5$ s. The mass of the vehicle



(a) Default games (b) Default games

Figure 9: Solution for Exercise 4.1. (a) The velocity responses for three vehicles that encounter a hill of 4° at $t = 5$ s for Exercise 4.1. This shows that a given set of gains provide decreasing performance for increasing mass. (b) Velocity response of the three different vehicles with a controller that returns the speed to within 1% of the desired speed within 3 s after encountering a hill of 4° at $t = 5$ s.

is plotted in increasing order from $m = 750$ kg to $m = 1250$ kg. It was created using the SIMULINK model `cruise_ctrl.mdl`, available on the companion web

site. Figure 9b depicts speed as a function of time with the gains selected to return the speed to within 1% of 20 m/s for each of the masses. The gains used were $k_p = 1.5$ $k_i = 0.1$.

5 Dynamic Behavior

Exercise 5.3 Consider the cruise control system described in Section 4.1. Generate a phase portrait for the closed loop system on flat ground ($\theta = 0$), in fourth gear, using a PI controller (with $k_\mathrm{p} = 0.5$ and $k_\mathrm{i} = 0.1$), $m = 1600$ kg, and desired speed 20 m/s. Your system model should include the effects of saturating the input between 0 and 1. cruise-phaseplot

(Hint: Keep in mind that when modeling feedback control, additional states can arise that do not appear in the original dynamics. You should include the MATLAB code used to generate your phase portrait.) Supplement

Solution. [CDS 101/110 TAs, 2004–06; Cole Lepine, Feb 08]

Include information on python version of solution (contributed by Scott Livingston)? **RMM**

The dynamics of the system are given in Section 3.1. These can be built into a MATLAB model using the following code: haseplotodel.m
To generate a phase portrait, we use the MATLAB script provided on the course web site: haseplot.m
Figure 10 shows the phase portrait for the system, along with a few trajectories also plotted. The equilibrium point is at $(v, z) = (20, 1)$, but it can be difficult to determine that only using the phase portrait. Note the lack of symmetry due to input saturation.

*Instructor note:*Max points is 10:

(a) 4: for understanding the dynamics

(b) 2: for saturation

(c) 2: for MATLAB code

(d) 2: for correct phase portrait

Common mistakes include: not plotting the full vector field, not including the input saturation, or incorrectly choosing the state variables.

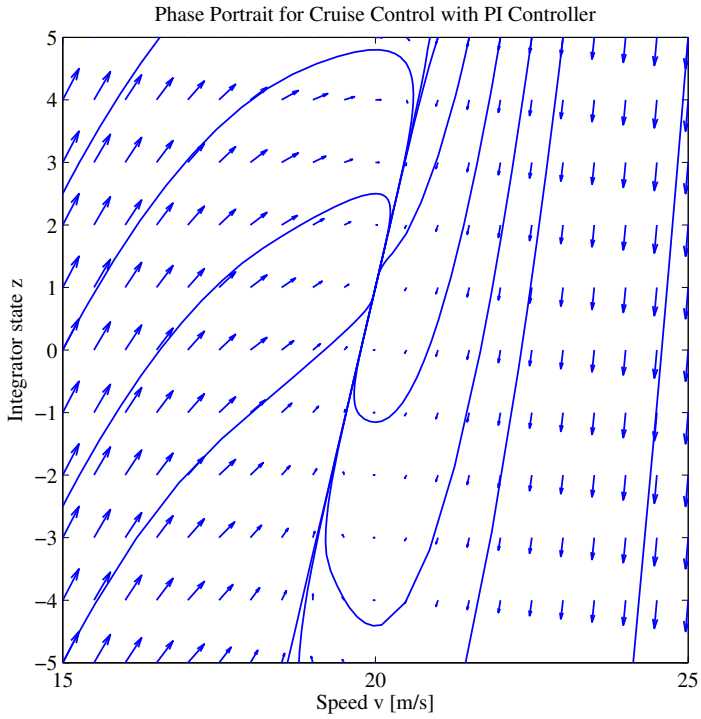


Figure 10: Phase curve of cruise control system for Exercise 5.3.

Exercise 5.29 Find a Lyapunov function for the cruise control system in Exercise 5.3, showing that the system is locally asymptotically stable at the desired speed. If you like, you can use the specific parameters listed above, although it is also possible to solve the problem leaving parameter values unspecified (with some assumptions, which you should state).

cruise-lyapunov

Solution. [Original Contributor, Unknown, Edited by Cole Lepine, Feb 08] The dynamics of the vehicle (on level ground, $\theta = 0$) are given by †

RMM: Update to use standard notation

$$\dot{v} = \frac{\alpha_n}{m} u T(\alpha_n v) - c - bv^2 = f(v, u)$$

where $c = gC_r > 0$, $b = \frac{1}{2m}\rho C_d A > 0$,

$$u = \text{sat}\left(k_p(v_r - v) + k_i \int_0^t (v_r - v)d\tau\right),$$

where

Comment [RMM, 12 Aug 2017]: Changed punctuation; check (and make consistent globally

$$\text{sat}(x) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x > 1, \\ x & \text{otherwise,} \end{cases} \quad \text{and} \quad T(\alpha_n v) = T_m \left(1 - \beta \left(\frac{\alpha_n}{\omega_m} v - 1\right)^2\right).$$

Note that $T(\alpha_n v) > 0$ for $-15 < v < 67$ m/s based on the parameters in the problem. Thus, for v close to v_r, $T(\alpha_n v) > 0$.

The full dynamic system of equations can be expressed as

$$\begin{aligned} \dot{v} &= -c - bv^2 + g(\alpha_n v)\ u \\ \dot{z} &= v_r - v \end{aligned}$$

where $g(\alpha_n v) = \frac{\alpha_n}{m} T(\alpha_n v) > 0$ near v_r and $z = \int_0^t (v_r - v) d\tau + C$ with C a constant of integration. The equilibrium point of this system of equations is at $v^* = v_r$ and $z^* = C = \frac{c + bv_r^2}{k_i g_r} > 0$ with $g_r = g(\alpha_n v_r)$. The positive constant C indicates a constant input from u is required to hold the vehicle at a desired velocity in the presence of friction and drag. Based on the parameters of the problem, at equilibrium $v^* = v_r = 20$ m/s, the input u is not saturated:

$$\dot{v}^* = 0 = -c - bv_r^2 + g(\alpha_n v_r)\ \text{sat}(k_i C) \Rightarrow \text{sat}(k_i C) = 0.1.$$

The dynamics can be rewritten about the equilibrium point by letting $x_1 = v - v^*$ and $x_2 = z - z^*$.

$$\begin{aligned} \dot{x}_1 &= -c - b(x_1 + v^*)^2 + g(\alpha_n(x_1 + v^*))\ \text{sat}(-k_p x_1 + k_i(x_2 + z^*)) \\ \dot{x}_2 &= -x_1 \end{aligned}$$

where

$$g(\alpha_n(x_1 + v^*)) = \frac{\alpha_n}{m} T(\alpha_n(x_1 + v^*)) = g_r - 2kex_1 - kx_1^2$$

with $k = \beta T_m \frac{\alpha_n^3}{m\omega_m^2}$, and $e = v^* - \frac{\omega_m}{\alpha_n}$. Near the equilibrium point, x_1 and x_2 are small, thus the saturation is in the linear region. Utilizing this assumption, the dynamics separate into linear and higher-order terms, and the constants cancel:

$$\begin{aligned} \dot{x}_1 &= -(2bv^* + k_p d_r + 2kek_i z^*)x_1 + k_i g_r x_2 \\ &\quad -bx_1^2 - 2ke(-k_p x_1 + k_i x_2)x_1 - k(-k_p x_1 + k_i x_2 + k_i z^*)x_1^2 \\ \dot{x}_2 &= -x_1. \end{aligned}$$

$$\dot{x} = Ax + \widetilde{F}(x)$$

$$
\begin{aligned}
x &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \\
A &= \begin{bmatrix} -(2bv^* + k_p d_r + 2kek_i z^*) & k_i d_r \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} -a_1 & a_2 \\ -1 & 0 \end{bmatrix}, \\
\widetilde{F}(x) &= \begin{pmatrix} -bx_1^2 - 2ke(-k_p x_1 + k_i x_2)x_1 - k(-k_p x_1 + k_i x_2 + k_i z^*)x_1^2 \\ 0 \end{pmatrix},
\end{aligned}
$$

and $a_1 > 0$, $a_2 > 0$.

The linear portion of the dynamics, represented by A will be used to find a Lyapunov function candidate $V(x) = x^T P x > 0$ with $P = P^T > 0$ from solution of the Lyapunov equation

$$A^T P + PA = Q.$$

For a guess with $Q = -I \in \mathbb{R}^{2\times 2}$,

$$P = \begin{bmatrix} \frac{a_2+1}{2a_1 a_2} & -\frac{1}{2a_2} \\ -\frac{1}{2a_2} & \frac{a_2^2+a_2+a_1^2}{2a_1 a_2} \end{bmatrix}.$$

Further, $\det(P) = \frac{(a_2+1)^2+a_1^2}{4a_l^2 a_2} > 0$ and $\text{trace}(P) = \frac{a_2^2+2a_2+a_1^2+1}{2a_1 a_2} > 0$, thus $P \succ 0$ as required.

A check if the candidate $V(x) = x^T P x$ is a Lyapunov function for the full nonlinear system yields :

$$
\begin{aligned}
\dot{V} &= (Ax + \widetilde{F}(x))^T Px + x^T P(Ax + \widetilde{F}(x)) \\
&= x^T(A^T P + PA)x + \widetilde{F}^T(x)Px + x^T P\widetilde{F}(x) \\
&= x^T Q x + \widetilde{F}^T(x)Px + x^T P\widetilde{F}(x) \\
&= -x^T x + \widetilde{F}^T(x)Px + x^T P\widetilde{F}(x).
\end{aligned}
$$

Since all terms in $\widetilde{F}(x)$ are quadratic and higher order in x, then $\widetilde{F}^T(x)Px + x^T P\widetilde{F}(x)$ has all cubic and higher-order terms in x. Sufficiently close to equilibrium ($x = 0$) implies the quadratic terms are larger than the cubic and higher-order terms, thus sufficiently close to $x = 0$ has $\dot{V} < 0$. Thus, $V(x) = x^T Px$ is a Lyapunov function for the full nonlinear system, and the system is locally asymptotically stable at the desired speed.

6 Linear Systems

cruise-linearize

Example 6.11 The dynamics for the cruise control system are derived in Section 4.1 and have the form

$$m\frac{dv}{dt} = \alpha_n u T(\alpha_n v) - mgC_{\mathrm{r}}\,\mathrm{sgn}(v) - \frac{1}{2}\rho C_{\mathrm{d}} A v|v| - mg\sin\theta, \tag{5}$$

where the first term on the right-hand side of the equation is the force generated by the engine and the remaining three terms are the rolling friction, aerodynamic drag, and gravitational disturbance force. There is an equilibrium point $(v_{\mathrm{e}}, u_{\mathrm{e}})$ when the force applied by the engine balances the disturbance forces.

To explore the behavior of the system near the equilibrium point we will linearize the system. A Taylor series expansion of equation (5) around the equilibrium point gives

$$\frac{d(v - v_{\mathrm{e}})}{dt} = -a(v - v_{\mathrm{e}}) - b_{\mathrm{g}}(\theta - \theta_{\mathrm{e}}) + b(u - u_{\mathrm{e}}) + \text{higher-order terms}, \tag{6}$$

where

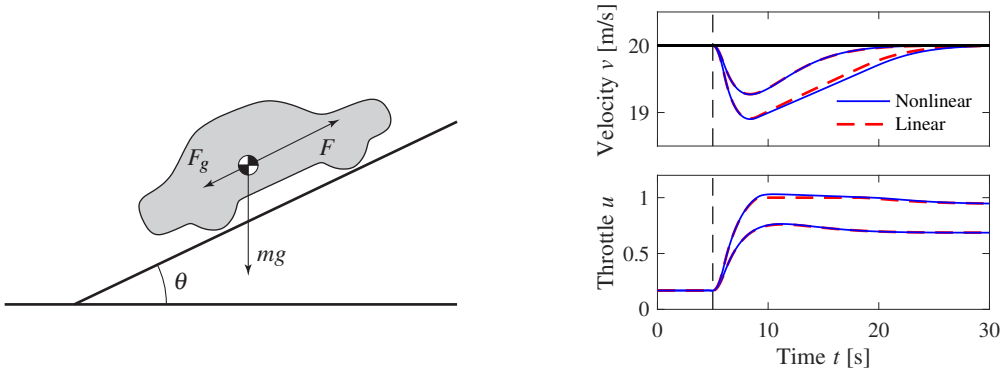
$$a = \frac{\rho C_{\mathrm{d}} A|v_{\mathrm{e}}| - u_{\mathrm{e}}\alpha_n^2 T'(\alpha_n v_{\mathrm{e}})}{m}, \qquad b_{\mathrm{g}} = g\cos\theta_{\mathrm{e}}, \qquad b = \frac{\alpha_n T(\alpha_n v_{\mathrm{e}})}{m}. \tag{7}$$

Notice that the term corresponding to rolling friction disappears if $v > 0$. For a car in fourth gear with $v_{\mathrm{e}} = 20$ m/s, $\theta_{\mathrm{e}} = 0$, and the numerical values for the car from Section 4.1, the equilibrium value for the throttle is $u_{\mathrm{e}} = 0.1687$ and the parameters are $a = 0.01$, $b = 1.32$, and $b_{\mathrm{g}} = 9.8$. This linear model describes how small perturbations in the velocity about the nominal speed evolve in time.

We will later describe how to design a proportional-integral (PI) controller for the system. Here we will simply assume that we have obtained a good controller and we will compare the behaviors when the closed loop system is simulated using the nonlinear model and the linear approximation. The simulation scenario is that the car is running with constant speed on a horizontal road and the system has stabilized so that the vehicle speed and the controller output are constant. Figure 11 shows what happens when the car encounters a hill with a slope of 4° and a hill with a slope of 6° at time t =5 s. The results for the nonlinear model are solid curves and those for the linear model are dashed curves. The differences between the curves are very small (especially for $\theta = 4^\circ$), and control design based on the linearized model is thus validated.

Example 6.12 Consider again the cruise control system from Example 6.11 with θ taken as a constant θ_{e}. We can write the dynamics as a first-order, nonlinear differential equation: cruise-jacobian

$$\begin{aligned}\frac{dx}{dt} &= f(x,u) = \frac{\alpha_n}{m} u T(\alpha_n x) - gC_{\mathrm{r}}\,\mathrm{sgn}(x) - \frac{1}{2}\frac{\rho C_{\mathrm{d}} A}{m} x|x| - g\sin\theta_{\mathrm{e}},\\ y &= h(x,u) = x,\end{aligned}$$



(a) Vehicle diagram (b) Closed loop response

Figure 11: Simulated response of a vehicle with PI cruise control as it climbs a hill with a slope of 4° (smaller velocity deviation/throttle) and unit6° (larger velocity deviation/throttle). The solid line is the simulation based on a nonlinear model, and the dashed line shows the corresponding simulation using a linear model. The controller gains are $k_\mathrm{p} = 0.5$ and $k_\mathrm{i} = 0.1$ and include anti-windup compensation (described in more detail in Example 11.6).

where $x = v$ is the velocity of the vehicle and u is the throttle. We use the velocity as the output of the system (since this is what we are trying to control).

If we linearize the dynamics of the system about an equilibrium point $x = v_\mathrm{e} > 0$, $u = u_\mathrm{e}$, using the formulas above we obtain

$$A = \frac{\partial f}{\partial x}\bigg|_{(x_\mathrm{e},u_\mathrm{e})} = \frac{u_\mathrm{e}\alpha_n^2 T'(\alpha_n x_\mathrm{e}) - \rho C_\mathrm{d} A|x_\mathrm{e}|}{m}, \quad B = \frac{\partial f}{\partial u}\bigg|_{(x_\mathrm{e},u_\mathrm{e})} = \frac{\alpha_n T(\alpha_n x_\mathrm{e})}{m},$$
$$C = \frac{\partial h}{\partial x}\bigg|_{(x_\mathrm{e},u_\mathrm{e})} = 1 \qquad D = \frac{\partial h}{\partial u}\bigg|_{(x_\mathrm{e},u_\mathrm{e})} = 0,$$

where we have used the fact that $\mathrm{sgn}(x) = 1$ for $x > 0$. This matches the results in Example 6.11, remembering that we have used x as the system state (vehicle velocity).

Example 6.14 Consider again the cruise control system from Example 6.11, cruise-fbklin whose dynamics are given in equation (5):

$$m\frac{dv}{dt} = \alpha_n u T(\alpha_n v) - mgC_\mathrm{r}\,\mathrm{sgn}(v) - \frac{1}{2}\rho C_\mathrm{d} A v|v| - mg\sin\theta.$$

If we choose u as a feedback law of the form

$$u = \frac{1}{\alpha_n T(\alpha_n v)} \left(\tilde{u} + mgC_{\mathrm{r}} \operatorname{sgn}(v) + \frac{1}{2} \rho C_{\mathrm{d}} A v |v| \right), \tag{8}$$

then the resulting dynamics become

$$m \frac{dv}{dt} = \tilde{u} + d, \tag{9}$$

where $d(t) = -mg \sin \theta(t)$ is the disturbance force due the slope of the road (which may be changing as we drive). If we now define a feedback law for $\tilde{u}$ (such as a proportional-integral-derivative [PID] controller), we can use equation (8) to compute the final input that should be commanded.

Equation (9) is a linear differential equation. We have essentially "inverted" the nonlinearity through the use of the feedback law (8). This requires that we have an accurate measurement of the vehicle velocity v as well as an accurate model of the torque characteristics of the engine, gear ratios, drag and friction characteristics, and mass of the car. While such a model is not generally available (remembering that the parameter values can change), if we design a good feedback law for $\tilde{u}$, then we can achieve robustness to these uncertainties.

7 State Feedback

cruise-integral

Example 7.8 Consider the cruise control example introduced in Section 1.5 and considered further in Example 6.11 (see also Section 4.1). The linearized dynamics of the process around an equilibrium point v_{e}, u_{e} are given by

$$\frac{dx}{dt} = -ax - b_{\mathrm{g}}\theta + bw, \qquad y = v = x + v_{\mathrm{e}},$$

where $x = v - v_{\mathrm{e}}$, $w = u - u_{\mathrm{e}}$, m is the mass of the car, and θ is the angle of the road. The constants a, b, and b_{g} depend on the properties of the car and are given in Example 6.11.

If we augment the system with an integrator, the process dynamics become

$$\frac{dx}{dt} = -ax - b_{\mathrm{g}}\theta + bw, \qquad \frac{dz}{dt} = y - v_{\mathrm{r}} = v_{\mathrm{e}} + x - v_{\mathrm{r}},$$

or, in state space form,

$$\frac{d}{dt} \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} -a & 0 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{pmatrix} b \\ 0 \end{pmatrix} w + \begin{pmatrix} -b_{\mathrm{g}} \\ 0 \end{pmatrix} \theta + \begin{pmatrix} 0 \\ v_{\mathrm{e}} - v_{\mathrm{r}} \end{pmatrix}.$$

Note that when the system is at equilibrium, we have that $\dot{z} = 0$, which implies that the vehicle speed $v = v_e + x$ should be equal to the desired reference speed v_r. Our controller will be of the form

$$\frac{dz}{dt} = y - v_r, \qquad w = -k_p x - k_i z + k_f v_r,$$

and the gains k_p, k_i, and k_f will be chosen to stabilize the system and provide the correct input for the reference speed.

Assume that we wish to design the closed loop system to have the characteristic polynomial

$$\lambda(s) = s^2 + a_1 s + a_2.$$

Setting the disturbance $\theta = 0$, the characteristic polynomial of the closed loop system is given by

$$\det\big(sI - (A - BK)\big) = s^2 + (bk_p + a)s + bk_i,$$

and hence we set

$$k_p = \frac{a_1 - a}{b}, \qquad k_i = \frac{a_2}{b}, \qquad k_f = -1/\big(C(A - BK)^{-1}B\big) = \frac{a_1}{b}.$$

The resulting controller stabilizes the system and hence brings $\dot{z} = y - v_r$ to zero, resulting in perfect tracking. Notice that even if we have a small error in the values of the parameters defining the system, as long as the closed loop eigenvalues are still stable, then the tracking error will approach zero. Thus the exact calibration required in our previous approach (using k_f) is not needed here. Indeed, we can even choose $k_f = 0$ and let the feedback controller do all of the work. However, k_f does influence the transient response to command signals and setting it properly will generally give a more favorable response.

Integral feedback can also be used to compensate for constant disturbances. Figure 12 shows the results of a simulation in which the car encounters a hill with angle $\theta = 4^\circ$ at $t = 5$ s. The steady-state values of the throttle for a state feedback controller and a controller with integral action are very close, but the corresponding values of the car velocity are quite different. The reason for this is that the zero frequency gain from throttle to velocity is $-b/a = 130$ is high. The stability of the system is not affected by this external disturbance, and so we once again see that the car's velocity converges to the reference speed. This ability to handle constant disturbances is a general property of controllers with integral feedback (see Exercise 7.4).

Exercise 7.23 Build a simulation for the speed controller designed in Example 7.8 and show that even with $k_f = 0$, the system still achieves zero steady-state error. cruise-norefgain

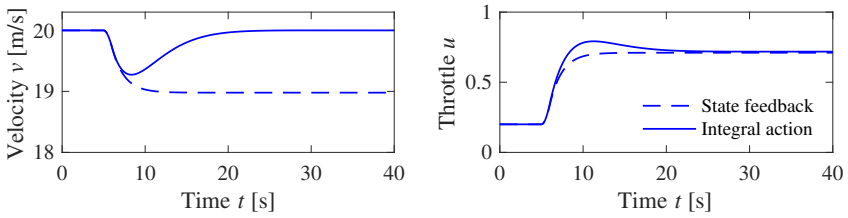


Figure 12: Velocity and throttle for a car with cruise control based on state feedback (dashed) and state feedback with integral action (solid). The controller with integral action is able to adjust the throttle to compensate for the effect of the hill and maintain the speed at the reference value of $v_r = 20$ m/s. The controller gains are $k_p = 0.5$ and $k_i = 0.1$.

8 Output Feedback

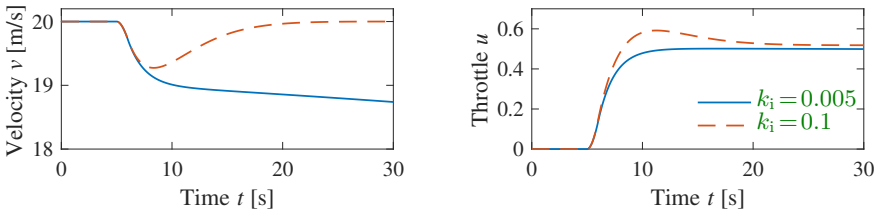
9 Transfer Functions

cruise-pzcancel

Example 9.12 A cruise control system can be modeled by the block diagram in Figure 9.6, where y is the vehicle velocity, r the desired velocity, v the slope of the road, and u the throttle. Furthermore $F(s) = 1$, and the input/output response from throttle to velocity for the linearized model for a car has the transfer function $P(s) = b/(s + a)$. A simple (but not necessarily good) way to design a PI controller is to choose the parameters of the PI controller as $k_i = ak_p$. The controller transfer function is then $C(s) = k_p + k_i/s = k_p(s + a)/s$. It has a zero at $s = -k_i/k_p = -a$ that cancels the process pole at $s = -a$. We have $P(s)C(s) = bk_p/s$ giving the transfer function from reference to vehicle velocity as $G_{yr}(s) = bk_p/(s+bk_p)$, and control design is then simply a matter of choosing the gain k_p. The closed loop system dynamics are of second order with the time constants $1/(bk_p)$ and $1/a$. Notice that the canceled pole $1/a$ is much slower than the other pole.

Figure 13 shows the velocity error when the car encounters an increase in the road slope. A comparison with the controller used in Figure 7b (reproduced in dashed curves) shows that the controller based on pole/zero cancellation has very poor performance. The velocity error is larger, and it takes a long time to settle.

Notice that the control signal remains practically constant after $t = 15$ even if the error is large after that time. To understand what happens we will analyze the system. The parameters of the system are $a = 0.01$ and $b = 1.32$, and the controller parameters are $k_p = 0.5$ and $k_i = 0.005$. The closed loop time constant is $1/(bk_p) = 1.5$ s, and we would expect that the error would settle in about 6 s (4 time constants). The transfer functions from road slope to velocity and control



(a) Velocity error (b) Throttle commands

Figure 13: Car with PI cruise control encountering a sloping road. The velocity error is shown on the left and the throttle is shown on the right. Results for a PI controller with $k_p = 0.5$ and $k_i = 0.005$ are shown by solid lines, and for a controller with $k_p = 0.5$ and $k_i = 0.1$ are shown by dashed lines. Compare with Figure 7b.

signals are

$$G_{yv}(s) = \frac{b_g s}{(s+a)(s+bk_p)}, \qquad G_{uv}(s) = \frac{bk_p}{s+bk_p}.$$

Notice that the slow canceled mode $s = -a = -0.01$ appears in G_{yv} but not in G_{uv}. The reason why the control signal remains constant is that the controller has a zero at $s = -0.01$, which cancels the slowly decaying process mode. Notice that the error would diverge if the canceled pole was unstable.

Exercise 9.23 Consider the following simplified equations of motion for a cruise control system (these are a linearization of the equations from Section 3.1 in Åström and Murray): cruise-pictrl

$$m\frac{dv}{dt} = -cv + b\tau + F_{\text{hill}},$$

where $m = 1000\dagger$ kg is the mass of the vehicle, $c = 50$ N s/m is the viscous damping coefficient, and $b = 25$ is the conversion factor between engine torque and the force applied to the vehicle. We model the engine using a simple first-order equation RMM: 1600?

$$\frac{d\tau}{dt} = a(-\tau + Tu),$$

where $a = 0.2$ is the lag coefficient and $T = 200$ is the conversion factor between the throttle input and the steady-state torque.

The simplest controller for this system is a proportional control, $u = k_p e$, where $e = (r - v)$ (r is the reference speed).

(a) Draw a block diagram for the system, with the engine dynamics and the vehicle dynamics in separate blocks and represented by transfer functions.

Label the reference input to the closed loop system as r, the disturbance due to the hill as d, and the output as y ($= v$).

(b) (MATLAB) Construct the transfer functions H_{er} and H_{yd} for the closed loop system and use MATLAB to generate the step response (`step`) and frequency response (`bode`) for the each. Assume that $k_\mathrm{p} = 0.5$. Make sure to use the transfer function computation.

(c) Consider a more sophisticated control law of the form

$$\frac{dx_\mathrm{c}}{dt} = r - v, \qquad u = k_\mathrm{p} e + k_\mathrm{i} x_\mathrm{c}.$$

This control law contains an "integral" term, which uses the controller state x_c to integrate the error. Compute the transfer functions for this control law and redraw your block diagram from part (a) with the default controller replaced by this one.

(d) (MATLAB) Using the gains $k_\mathrm{p} = 0.5$ and $k_\mathrm{i} = 0.1$, use MATLAB to compute the transfer function from r to y and plot the step response and frequency response for the system.

Solution. (a) We are asked to show an overall block diagram for the system, with the engine and vehicle dynamics separated into two blocks. Each of these subsystems has first-order dynamics, and we have seen in class how to construct transfer functions for these kinds of systems, so we only show the final result in Figure 14. The figure also shows the block diagram for the modified controller. The overall system looks exactly as before except with the new control block swapped in for the old one.

Note that we have moved the torque conversion constant b inside the vehicle dynamics, and have necessarily moved the inverse of this out into the disturbance channel to compensate.

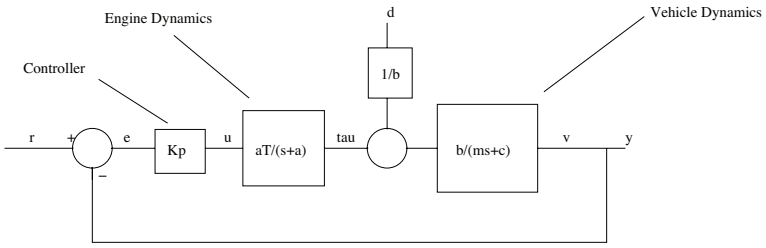
(b) The transfer function from the reference to the error is $1 - H_{yr}$, by construction. The latter is:

$$\frac{k_p K a T}{ms^2 + (ma + b)s + ab + k_\mathrm{p} K a T}$$

Thus, we have:

$$H_{er} = \frac{ms^2 + (ma + c)s + ac}{ms^2 + (ma + c)s + ac + k_\mathrm{p} abT}$$

RMM: Redo figure



Modified Controller Block Diagram

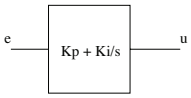


Figure 14: Block diagram of system and modified controller

The Bode plot is shown in Figure 15.

The transfer function from the disturbance to the output can be found by observing that by setting the reference input r to zero, the system is once again a simple feedback loop between d and y. The resulting transfer function is

$$H_{yd} = \frac{s+a}{ms^2 + (ma+c)s + ac + k_{\mathrm{p}}abT}$$

and the step and frequency responses are shown in Figure 16.

(c) The transfer function of the controller is

$$H_{ue} = -k_p - \frac{k_{\mathrm{i}}}{s}.$$

You can replace $-K_p$ by this new H_{ue} in part (a).

(d) Figure 17 shows the new system response. The transfer function is:

$$H_{yr} = \frac{abTk_{\mathrm{p}}s + abTk_{\mathrm{i}}}{ms^3 + (ma+c)s^2 + (ac + abTk_{\mathrm{p}})s + abTk_{\mathrm{i}}}$$

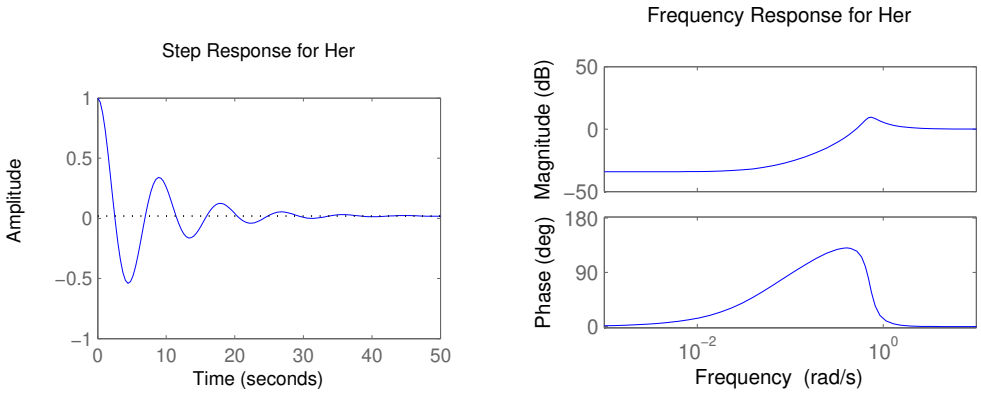


Figure 15: Frequency and step response of H_{er}

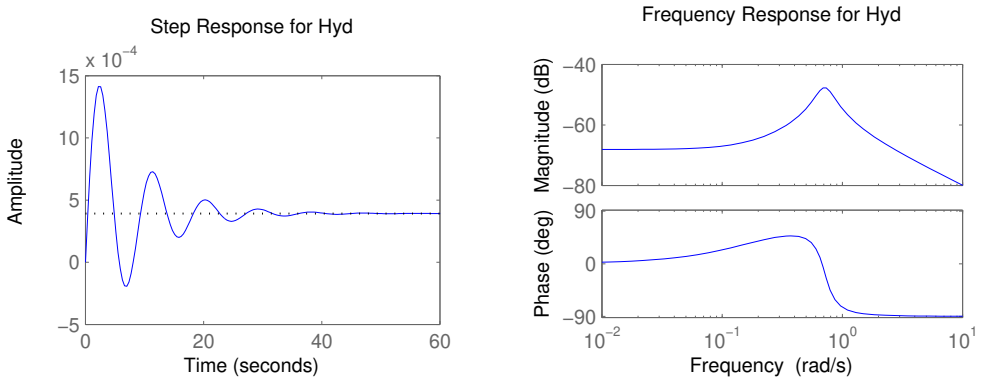


Figure 16: Frequency and step response of H_{yd}

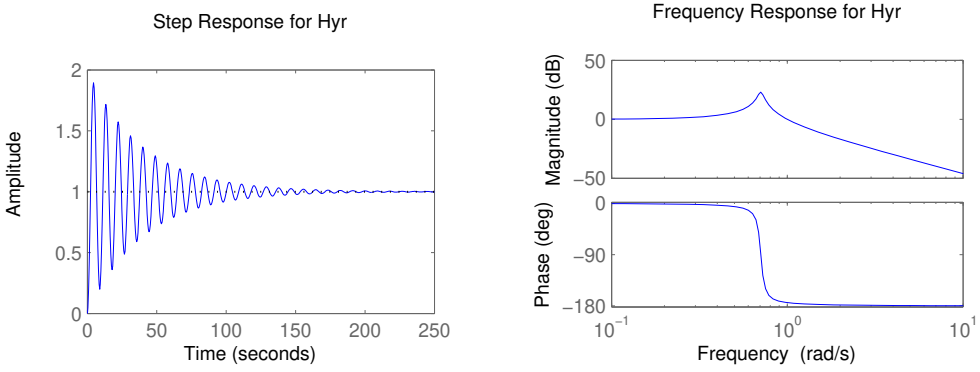


Figure 17: Frequency and Step Response with PI Control

10 Frequency Domain Analysis

Exercise 9.23 In this problem we will design a PI controller for a cruise control system, building on the example shown in class. Use the following transfer function to represent the vehicle and engine dynamics: cruise-pictrl

$$P(s) = \frac{Tba/m}{(s+a)(s+c/m)}$$

where $b = 25$ is the transmission gain, $T = 200$ is the conversion factor between the throttle input and steady-state torque, $a = 0.2$ is the engine lag coefficient, $m = 1000$ kg is the mass of the car, and $c = 50$ N s/m is the viscous damping coefficient.

(a) Consider a proportional controller for the car, $u = k_\mathrm{p}(r - y)$. Assuming a unity gain feedback controller, this gives

$$C(s) = k_\mathrm{p}.$$

Set $k_\mathrm{p} = 0.1$ and compute the steady-state error, gain and phase margins, rise time, overshoot, and poles/zeros for the system. Remember that the gain and phase margins are computed based on the loop transfer function $L(s) = P(s)C(s)$; the remaining quantities should be computed for the closed loop system.

(b) Consider a proportional + integral controller for the car,

$$C(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s}.$$

Fill in the following table (make sure to show your work), where g_m is the gain margin, φ_m the phase margin, SSerr the steady-state error, BW the bandwidth (you can use the bandwidth command in Matlab, but you need to do so for the closed loop system), T_r the rise time, and M_p the overshoot (see Fig. 5.9 on p. 151 of the text, you do not need to be exact).

k_p	k_i	Stable?	g_m	φ_m	SSerr	BW	T_r	M_p
0.5	0.1							
0.05	1							
0.05	0.001							
0.005	0.001							

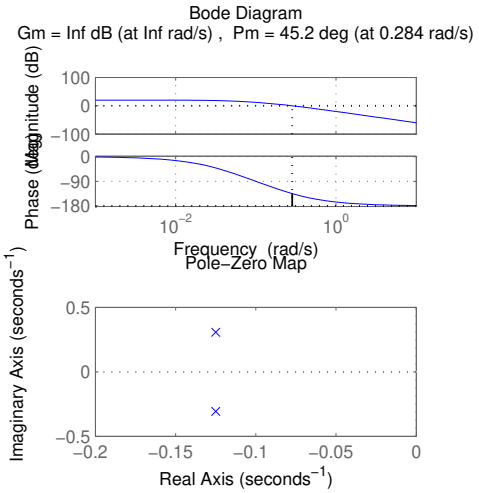


Figure 18: Bode Diagram and Pole-Zero map

For each entry in the table, plot the pole zero diagram (pzmap) for the *closed loop* system and the step response. (Note that the steady-state error is zero in each stable case, due to the integral term in the control law.) (Suggestion: look for relationships between the various quantities you are computing and plotting. This problem should give you some insight into the relationship between some of the quantities.)

Solution. (a) The appropriate open and closed loop transfer functions are constructed via the attached MATLAB m-file. The gain and phase margins are calculated using the command `margin` and are shown in Figure 18. The steady-state error is 0.0909. The closed loop poles are $s = -0.1250 \pm 0.3072i$ and there are no closed loop zeroes (see Figure 18). **Note:** Several people used the transfer function H_{yr} in their computations, which leads to slightly different results throughout the exercise. This was accepted, but if you are required to analyze the steady-state error, this means you need to work with the H_{er} closed loop transfer function!

*Instructor note:*4pts: each feature (ss error, gain margin, phase margin, rise time, overshoot, poles and zeros) is 0.5 pts, 1 pt - demonstrating understanding of material(stability / other discussion)

(b) When a proportional + integral controller is used, the following table results. See attached m-file for the calculations. The pole-zero maps and the step responses for each entry are shown in Figure 19.

Note: Either the bandwidth defined by the loop cross-over frequency (ω_{cg}, e.g. from 'margin') or the -3 dB on the closed loop system (e.g. from the MATLAB command 'bandwidth') is acceptable. The frequency at which the phase crosses -180, ω_{pm} should not be used to determine the bandwidth.

k_p	k_i	St	GM	PM	SSErr	BW		T_r	M_p
.5	.1	Y	∞	4.0498	0	1.0973 (bandwidth)	0.70621 (ω_{cg})	1.5	0.9
.05	1	N	0.0025	-72.8883	NA	1.0892 (bandwidth)	0.9931 (ω_{cg})	1	∞
.05	.001	Y	∞	57.1752	0	0.2845 (bandwidth)	0.1801 (ω_{cg})	8	0.065
.005	.001	Y	∞	38.6908	0	0.1000 (bandwidth)	0.0624 (ω_{cg})	20	0.3

*Instructor note:*8 pts: 2 pts for each row; 0.5 pts each step plot; 0.5 pts each pzmap

Exercise 10.24 Consider the speed control system described in Section 4.1 and analyzed using state space techniques in Example 7.8. Using a modified PI controller

cruise-nyquist

$$C(s) = G_{ue}(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s+\beta} = \frac{k_\mathrm{p} s + k_\mathrm{i} + k_\mathrm{p}\beta}{s+\beta}$$

plot the Nyquist curve for the system and determine the gain, phase, and stability margins.†

RMM: Need to provide the controller parameters

RMM: Add engine dynamics to make i

Solution. This exercise needs to be updated to use the dynamics described in the text

RMM: Interesting?

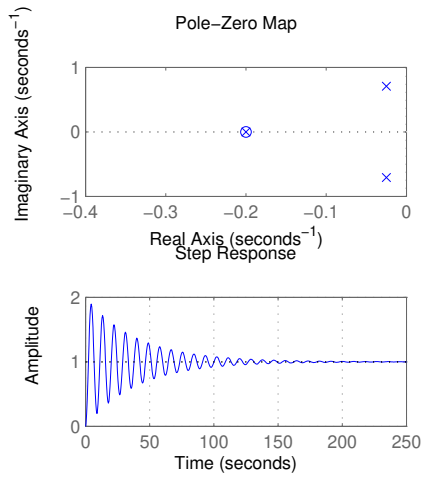
The linearized dynamics around the equilibrium speed v_e and throttle position u_e are given by

$$\dot{\tilde{v}} = a\tilde{v} - b_\mathrm{g}\theta + b\tilde{u}$$
$$y = v = \tilde{v} + v_\mathrm{e},$$

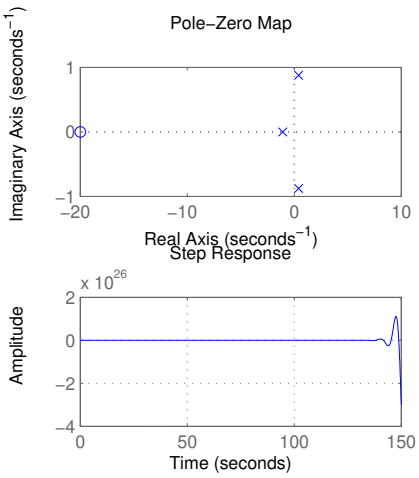
where $\tilde{v} = v - v_\mathrm{e}$, $\tilde{u} = u - u_\mathrm{e}$, m is the mass of the car and θ is the angle of the road. The constant $a < 0$ depends on the throttle characteristic and is given in Example 6.11.

The transfer function from throttle to speed is given by

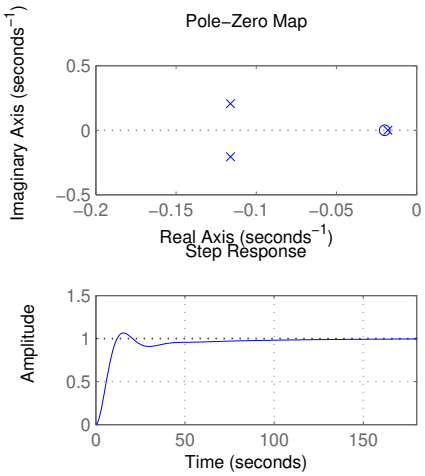
$$P(s) = G_{yu}(s) = \frac{b}{s+a}.$$



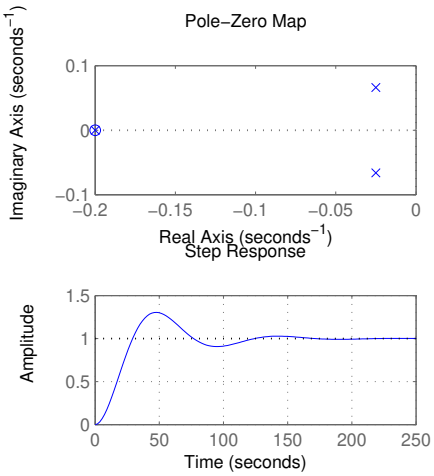
(a) $k_{\mathrm{p}} = .5$ and $k_{\mathrm{i}} = .1$



(b) $k_{\mathrm{p}} = .05$ and $k_{\mathrm{i}} = 1$



(c) $k_{\mathrm{p}} = .05$ and $k_{\mathrm{i}} = .001$



(d) $k_{\mathrm{p}} = .005$ and $k_{\mathrm{i}} = .001$

Figure 19: Pole-zero map and step responses.

We consider a controller that is a modified version of the proportional-integral (PI) controller given previously. Assume that the transfer function of the controller is giving a loop transfer function of

$$L(s) = b\frac{k_{\mathrm{p}}s + k_{\mathrm{i}} + k_{\mathrm{p}}\beta}{(s+a)(s+\beta)}.$$

The Nyquist plot for the system, using $a = 0.01$, $b = 1.32$, $k_{\mathrm{p}} = 0.5$, $k_{\mathrm{i}} = 0.1$,

and $\beta = 0.1$, is shown in Figure 20. We see from the Nyquist plot that the closed

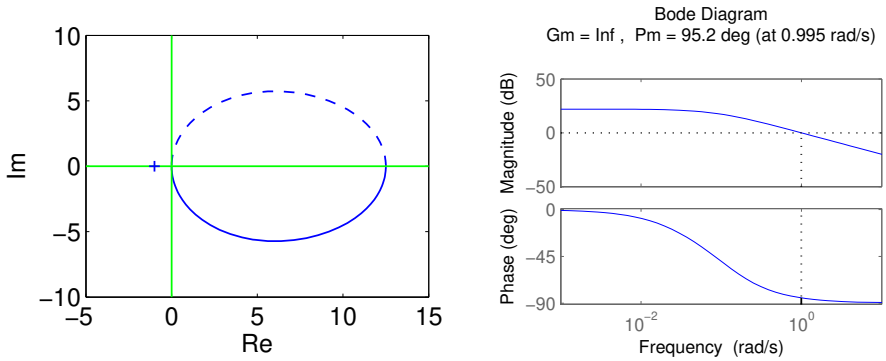


Figure 20: Stability margins for the cruise control system. The Nyquist plot on the left shows the featured required to compute the gain, phase, and stability margins, following Figure 10.11. The gain and phase margin can also be determined from the Bode plot, shown on the right.

loop system is stable, since there are no net encirclements of the -1 point.

To compute the gain, phase, and stability margins, we can use the Nyquist plot as described in Figure 10.11a. This yields the following values:

$$g_m =, \qquad \varphi_m =, \qquad s_m = .$$

The gain and phase margin can also be determined from the Bode plot shown in Figure 10.11b.

Add stability margin computation RMM

Supplemental Exercise Use the Nyquist theorem to analyze the stability of the cruise control system in Example 7.8.† cruise-nyquist-alt

Solution. Figure 21 shows the image of the contour Γ under the map L. The loop

MISSING

Figure 21: The complete Nyquist curve for the loop transfer function $L(s) = \frac{k}{s(s+1)^2}$. The curve is drawn for $k < 2$. The map of the positive imaginary axis is shown in solid lines, the map of the negative imaginary axis and the small semicircle at the origin in dashed lines.

transfer function does not have any poles in the region enclosed by the Nyquist contour. By computing the phase of L, one can show that the Nyquist plot

RMM: Confirm this is the right example to cite.

RMM: Add a Bode plot to the text and update caption to reflect and to [illegible] consistently here.

intersects the real axis for $\omega = 1$ and the intersection is at $-k/2$. It follows from Figure 21 that the number of counterclockwise† encirclements is zero if $k < 2$ and 2 if $k > 2$. We can thus conclude that the closed loop system is stable if $k < 2$ and that the closed loop system has two roots in the right half-plane if $k > 2$. **RMM**: check

Gain, phase, stability computations missing **RMM**

Exercise 10.29 Continuing the previous problem, we will now insert a small amount of time delay into the feedback path of the system. A pure time delay of τ seconds satisfies the equation cruise-delay

$$y(t) = u(t - \tau)$$

This system is a linear input/output system and it can be shown that its transfer function is

$$G(s) = e^{-s\tau}.$$

Unfortunately, MATLAB is not able to perfectly represent a time delay in this form, and so we have to use a "Padé approximation", which gives a constant gain transfer function with phase that approximates a time delay. Using a 2nd order Padé approximation, we can approximate our time delay as

$$G(s) = \frac{1 - \tau s/2 + (\tau s)^2/12}{1 + \tau s/2 + (\tau s)^2/12}$$

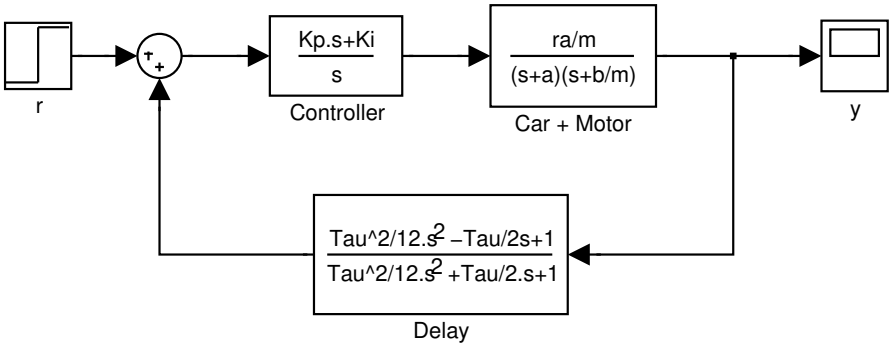
This function can be computed using the `pade` function in MATLAB (although the numerator and denominator are scaled slightly differently).

Assume that there is a time delay of τ seconds, which we will insert between the output of the plant and the controller

(a) For the case $k_\mathrm{p} = 0.05$, $k_\mathrm{i} = 0.001$, insert time delays of $\tau = 0.25$ s and $\tau = 0.75$ s. Using a Padé approximation, compute the resulting gain and phase margin for each case and compute the overshoot and settling time (2%) for the step responses.

(b) Repeat part (a) using $k_\mathrm{p} = 0.02$, $k_\mathrm{i} = 0.0005$, and time delays of 0.75 s and 1.5 s.

Solution. The block diagram for the cruise control system with a time delay is



(a) Using a delay of $\tau = 0.25$ s, we compute the gain margin (18.6 [25.4 dB]) and phase margin (54.59 deg) with the command `margin` (see attached m-file). The overshoot (8%) and settling time (95 s) are calculated directly from the step response of the closed loop system (see Figure 22). For a delay of $\tau = 0.75$ s, the corresponding gain and phase margins are 16.01dB and 49.43 deg while the overshoot is 13% and the settling time is 95 s (see Figure 23).

(b) Now we change the gains on the controller to $k_p = .02$ and $k_i = 0.0005$ and insert a time delay of $\tau = 0.75$ s. The resulting gain and phase margins are 23.77dB and 78.35 deg while the overshoot is 0% and the settling time is 128 s (see Figure **??**) For a delay of $\tau = 1.5$ s, the resulting gain and phase margins are 17.99dB and 74.80 deg while the overshoot is 0% and the settling time is 128 s (see Figure **??**)

```
% Homework #6 Problem #3 (hw6p3.m)
% Abhishek Tiwari 23 Nov 03 (Update by John Carson 19 Nov 2006)
clear all global;
% Define dynamics from prob 2
T = 200; m = 1000; b =50; r= 25; a = 0.2;
% Part (a)
% Construct open and closed loop transfer functions
Kp = .05; Ki =.001;

P = tf([T*r*a/m],conv([1 a],[1 b/m]));
C = tf([Kp Ki],[1 0]);
[n25,d25] = pade(0.25,2);
[n75,d75] = pade(0.75,2);
D25 =tf(n25,d25); D75 =tf(n75,d75);
Lf = series(C,P);
L25 =series(Lf,D25); T25 =feedback(Lf,D25);
```

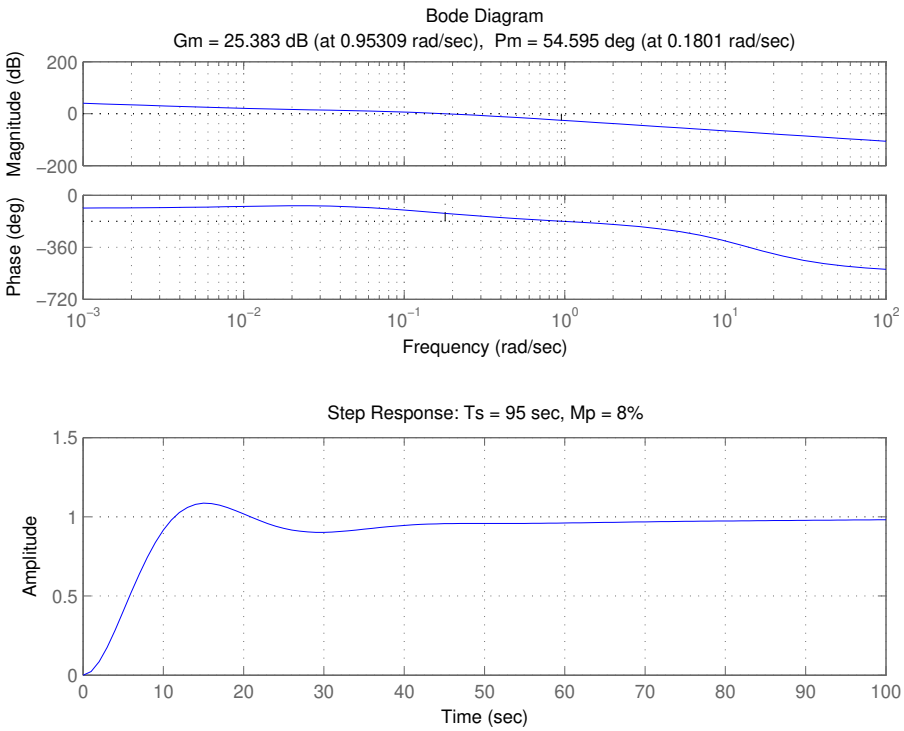


Figure 22: Bode and Step Responses for Gains of $k_{\mathrm{p}} = .05$ and $k_{\mathrm{i}} = .001$ and a Delay of $\tau = 0.25$ s

```
L75 =series(Lf,D75); T75 =feedback(Lf,D75);
% Plot results
figure(1); clf;
subplot(2,1,1); margin(L25); grid;
subplot(2,1,2); step(T25,100); grid;
title('Step Response: Ts = 95 s, Mp = 8%');
figure(2); clf;
subplot(2,1,1); margin(L75); grid;
subplot(2,1,2); step(T75,100); grid;
title('Step Response: Ts = 95 s, Mp = 13%');
% Part (b)
% Construct open and closed loop transfer functions

Kp = .02; Ki =0.0005;
P = tf([T*r*a/m],conv([1 a],[1 b/m]));
C = tf([Kp Ki],[1 0]);
[n150,d150] =pade(1.5,2);
```

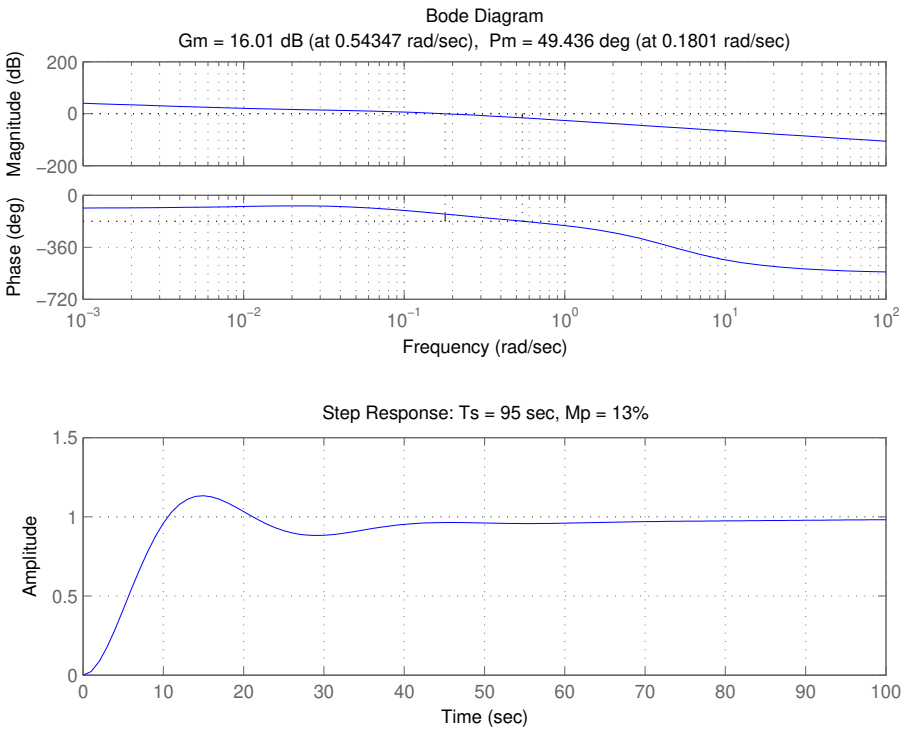


Figure 23: Bode and Step Responses for of $k_{\mathrm{p}} = .05$ and $k_{\mathrm{i}} = .001$ and a Delay of $\tau = 0.75$ s

```
D150 = tf(n150,d150);
Lf = series(C,P);
L150 = series(Lf,D150); T150 =feedback(Lf,D150);
L75 =series(Lf,D75); T75 =feedback(Lf,D75);
% Plot results
figure(3); clf;
subplot(2,1,1); margin(L75); grid;
subplot(2,1,2); step(T75,130); grid;
title('Step Response: Ts = 128 s, Mp = 0%');
figure(4); clf;
subplot(2,1,1); margin(L150); grid;
subplot(2,1,2); step(T150,130); grid;
title('Step Response: Ts = 128 s, Mp = 0%');
```

11 PID Control

Example 11.3 Consider the problem of maintaining the speed of a car as it goes up a hill. In Example 11 we found that there was little difference between the linear and nonlinear models when investigating PI control, provided that the throttle did not reach the saturation limits. A simple linear model of a car was given in Example 6.11: cruise-pid

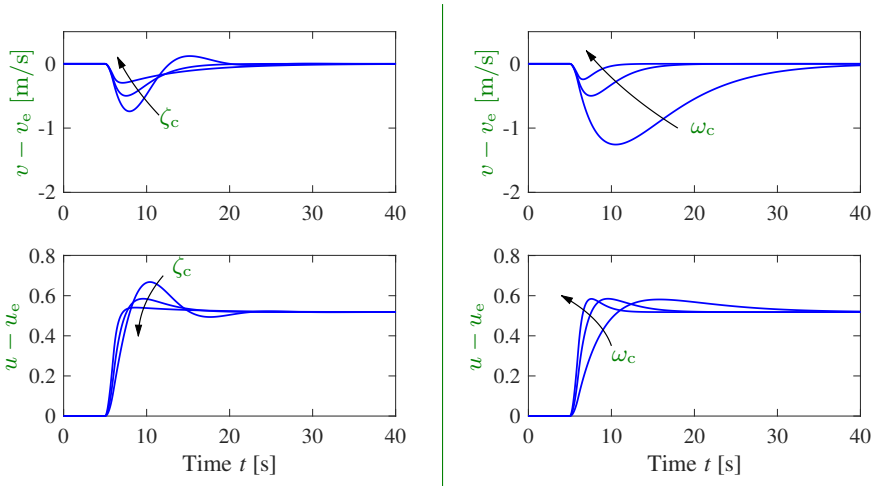
$$\frac{d(v - v_e)}{dt} = -a(v - v_e) + b(u - u_e) - b_g\theta, \tag{10}$$

where v is the velocity of the car, u is the input to the engine (throttle) and θ is the slope of the hill. The parameters were $a = 0.01$, $b = 1.32$, $b_g = 9.8$, $v_e = 20$, and $u_e = 0.1687$. This model will be used to find suitable parameters of a vehicle speed controller. The transfer function from throttle to velocity is a first-order system. Since the open loop dynamics are quite slow ($1/a \approx 100$ s), it is natural to specify a faster closed loop system by requiring that the closed loop system be of second order with damping ratio ζ_c and undamped natural frequency ω_c. The controller gains are given by equation (11.7).

Figure 24 shows the velocity and the throttle for a car that initially moves on a horizontal road and encounters a hill with a slope of 4° at time $t = 5$ s. To design a PI controller we choose $\zeta_c = 1$ to obtain a response without overshoot, as shown in Figure 24a. The choice of ω_c is a compromise between response speed and control actions: a large value gives a fast response, but it requires fast control action. The trade-off is illustrated in Figure 24b. The largest velocity error decreases with increasing ω_c, but the control signal also changes more rapidly. In the simple model (10) it was assumed that the force responds instantaneously to throttle commands. For rapid changes there may be additional dynamics that have to be accounted for. There are also physical limits to the rate of change of the force, which also restricts the admissible value of ω_c. A reasonable choice of ω_c is in the range 0.5–1.0. Notice in Figure 24 that even with $\omega_c = 0.2$ the largest velocity error is only about 1.3 m/s.

With $\omega_c = 0.5$ and $\zeta_c = 1$ we have $k_p = 0.276$ and $k_i = 0.0693$. An analysis of the error equation† shows that the damping term, $0.004 + 3.61k$, is dominated by the term $3.61k \approx 1$ introduced by the controller. The term 0.004, due to the aerodynamic forces, is two orders of magnitude smaller and quite complicated to compute (see Example 6.11). We may therefore conjecture that this term can be neglected for the purpose of design of a speed controller. This observation is an illustration of an important and surprising property of feedback, namely that feedback systems can be designed based on simplified models. This will be discussed in Chapter 13. Supplement

RMM: removed equation reference



(a) $\omega_c = 0.5$, $\zeta_c = 0.5$, 1, 2 (b) $\zeta_c = 1$, $\omega_c = 0.2$, 0.5, 1

Figure 24: Cruise control using PI feedback. The step responses for the error and input illustrate the effect of parameters ζ_c and ω_c on the response of a car with cruise control. The slope of the road changes linearly from 0° to 4° between $t = 5$ and 6 s. (a) Responses for $\omega_c = 0.5$ and $\zeta_c = 0.5$, 1, and 2. Choosing $\zeta_c \geq 1$ gives no overshoot in the velocity v. (b) Responses for $\zeta_c = 1$ and $\omega_c = 0.2$, 0.5, and 1.0.

Another interpretation of the effect of the integral action can be given by returning to the basic force balance model of the car

$$m\frac{dv}{dt} = F - F_d,$$

where m is the mass of the car, F is the applied force (from the engine) and F_d is the disturbance force (aerodynamic drag and force of gravity). Since zero steady-state error implies that v is constant, we see that the PI controller generates an output force F that in steady state is equal to the drag force F_d. Since the error is zero in steady state the controller output equals the output of the integrator of the PI controller. The output of the integrator in the PI controller can thus be interpreted as an estimator of the drag force.

Example 11.5 The windup effect is illustrated in Figure 25a, which shows what happens when a car encounters a hill that is so steep (6°) that the throttle saturates when the cruise controller attempts to maintain speed. When encountering the slope at time $t = 5$, the velocity decreases and the throttle increases to generate more torque. However, the torque required is so large that the throttle cruise-windup

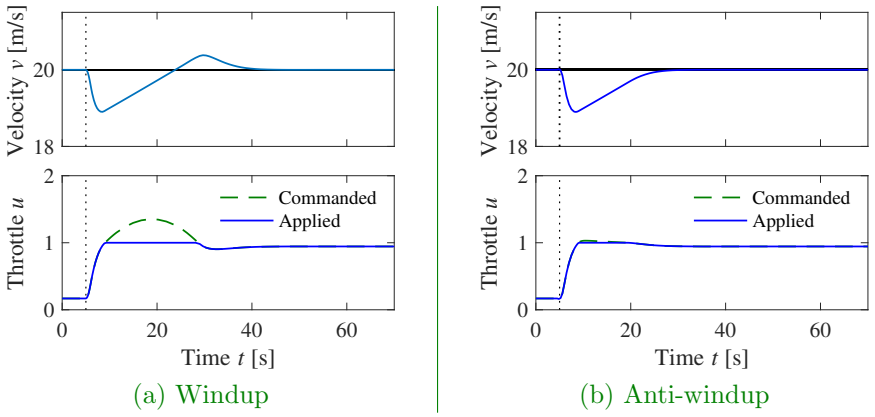
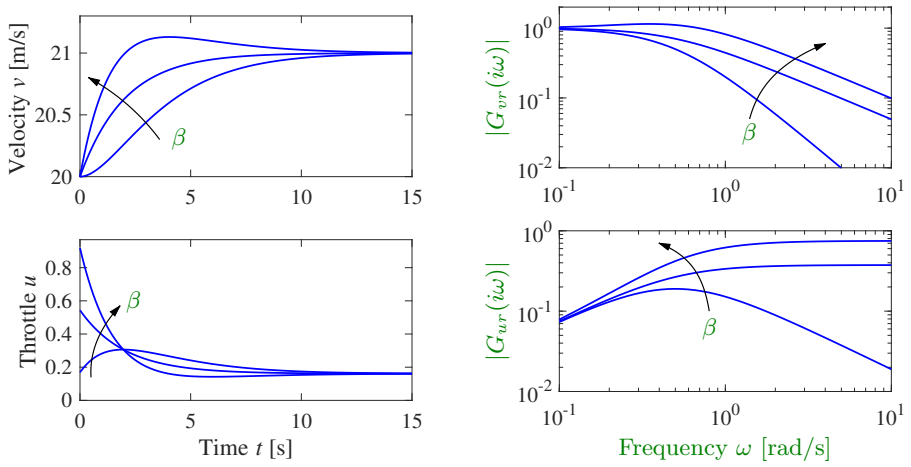


Figure 25: Simulation of PI cruise control with windup (a) and anti-windup (b). The figure shows the speed v and the throttle u for a car that encounters a slope that is so steep that the throttle saturates. The controller output is a dashed line. The controller parameters are $k_\mathrm{p} = 0.5$, $k_\mathrm{i} = 0.1$ and $k_\mathrm{aw} = 2.0$. The anti-windup compensator eliminates the overshoot by preventing the error from building up in the integral term of the controller.

saturates. The error decreases slowly because the torque generated by the engine is just a little larger than the torque required to compensate for gravity. The error is large and the integral continues to build up until the error reaches zero at time 25, but the controller output is still larger than the saturation limit and the actuator remains saturated. The integral term starts to decrease, and the velocity settles to the desired value at time $t = 40$. Also notice the large overshoot.

Example 11.6 Figure 25b shows what happens when a controller with anti-windup is applied to the system simulated in Figure 25a. Because of the feedback from the actuator model, the output of the integrator is quickly reset to a value such that the controller output is at the saturation limit. The behavior is drastically different from that in Figure 25a and the large overshoot is avoided. The tracking gain used in the simulation is $k_\mathrm{aw} = 2$ which is an order of magnitude larger than the integral gain $k_\mathrm{i} = 0.2$. cruise-antiwindup

Example 11.7 Consider the PI controller for the cruise control system derived in Example 11.3. Figure 26 shows the effect of setpoint weighting on the response of the system to a reference signal. With $\beta = 1$ (error-feedback) there is an overshoot in velocity and the control signal (throttle) is initially close to the saturation limit. There is no overshoot with $\beta = 0$ and the control signal is cruise-spweight



(a) Step response (b) Frequency responses

Figure 26: Step and frequency responses for PI cruise control with setpoint weighting. Step responses are shown in (a) and the gain curves of the frequency responses in (b). The controller gains are $k_\mathrm{p} = 0.74$ and $k_\mathrm{i} = 0.19$. The setpoint weights are $\beta = 0$, 0.5 and 1, and $\gamma = 0$.

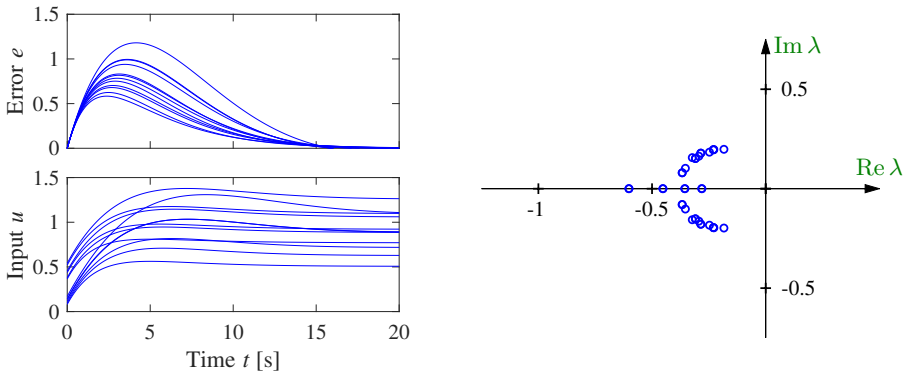
much smaller, clearly a much better drive comfort. The frequency responses gives another view of the same effect. The parameter β is typically in the range 0–1, and γ is normally zero to avoid large transients in the control signal when the reference is changed.

12 Frequency Domain Design

13 Robust Performance

cruise-parametric

Example 13.1 The cruise control problem is described in Section 4.1, and a PI controller was designed in Example 11.3. To investigate the effect of parameter variations, we will choose a controller designed for a nominal operating condition corresponding to mass $m = 1600$ kg, fourth gear ($\alpha = 12$) and speed $v_\mathrm{e} = 20$ m/s; the controller gains are $k_\mathrm{p} = 0.5$ and $k_\mathrm{i} = 0.1$. Figure 27a shows the velocity error e and the throttle u when encountering a hill with a 4° slope with masses in the range $1600 < m < 2000$ kg, gear ratios 3–5 ($\alpha = 10$, 12, and 16), and velocity $10 \leq v \leq 40$ m/s. The simulations were done using models that were linearized around the different operating conditions. The figure shows that there are variations in the response but that they are all quite reasonable. The largest velocity error is in the range of 0.5–1.2 m/s, and the settling time is about 15



(a) Disturbance response (b) Closed loop eigenvalues

Figure 27: Responses of the cruise control system to a slope increase of 4° (a) and the eigenvalues of the closed loop system (b). Model parameters are swept over a wide range. The closed loop system is of second order.

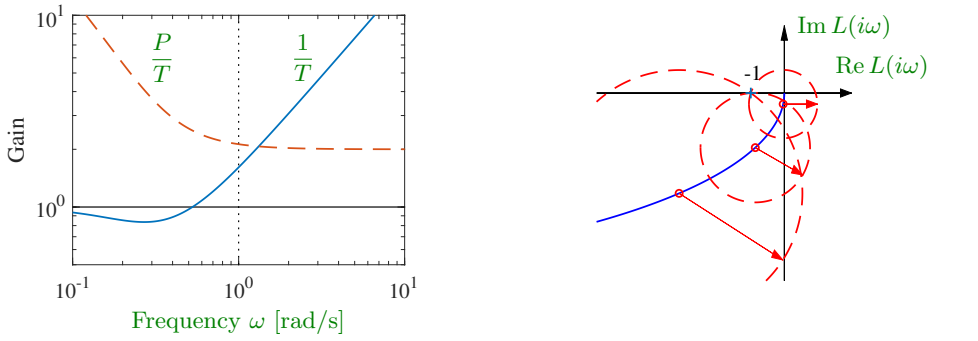
s. The control signal is larger than 1 in some cases, which implies that the throttle is fully open. (A full nonlinear simulation using a controller with windup protection is required if we want to explore these cases in more detail.) The closed loop system has two eigenvalues, shown in Figure 27b for the different operating conditions. We see that the closed loop system is well damped in all cases.

Example 13.7 Using the parameters from Example 6.11, the model of the car in fourth gear at speed 20 m/s is cruise-robstab

$$P(s) = \frac{1.32}{s + 0.01},$$

and the controller is a PI controller with gains $k_\mathrm{p} = 0.5$ and $k_\mathrm{i} = 0.1$. Figure 28 plots the allowable size of the process uncertainty using the bound in equation (13.10).

At low frequencies, $T(0) = 1$ and so the perturbations can be as large as the original process ($|\delta| = |\Delta/P| < 1$). The complementary sensitivity has its maximum $M_\mathrm{t} = 1.17$ at $\omega_\mathrm{mt} = 0.26$, and hence this gives the lowest allowable process uncertainty, with $|\delta| < 0.86$ or $|\Delta| < 4.36$. Finally, at high frequencies, $T \to 0$ and hence the relative error can get very large. For example, at $\omega = 5$ rad/s we have $|T(i\omega)| = 0.264$, which means that the stability requirement is $|\delta| < 3.8$. The analysis clearly indicates that the system has good robustness and that the high-frequency properties of the transmission system are not important for the design of the cruise controller.



(a) Bounds on process uncertainty (b) Nyquist plot representation of bounds

Figure 28: Robustness for a cruise controller. (a) The maximum relative error $1/|T|$ (solid) and the absolute error $|P|/|T|$ (dashed) for the process uncertainty Δ. (b) The Nyquist plot of the loop transfer function L (zoomed in to the region around the critical point) is shown on the right as a solid line. The dashed circles show allowable perturbations in the process dynamics, $|C\Delta| = |CP|/|T|$, at the frequencies $\omega = 0.2$, 0.4, and 2, which are marked with circles.

Another illustration of the robustness of the system is given in Figure 28b, which shows the Nyquist curve of the loop transfer function L along with the allowable perturbations. We see that the system can tolerate large amounts of uncertainty and still maintain stability of the closed loop.

14 Fundamental Limits

15 Architecture