System Dynamics (22.554 & 24.509)

VII. Introduction to the Design and Simulation of Controlled Systems

Introduction

As indicated several times already, the focus of this course is not on the design of control systems. Instead we have been emphasizing the mathematics and implementation techniques associated with the modeling and simulation of general systems. However, control concepts are very important, and it does make sense to at least introduce this subject within the context of general systems analysis.

In particular, there are two general approaches to developing controller designs - the so-called ***classical method*** and the ***modern method***. Some of the Case Studies in Section VI already introduced the classical approach which includes PID controllers, lead-lag compensators, etc.. The classical design method relies on the root locus technique (see Case Study E for example) and/or the various frequency response representations of LTI systems (Bode, Nyquist, and Nichols plots) to assist in the selection of the free variables introduced within the controller transfer function. The tuning of the control variables is usually performed via an educated or guided trial-and-error approach, and the creativity and experience of the designer plays an important role in the overall design process. The classical design method is well suited to single-input-single-output (SISO) systems.

Modern control theory, in contrast, employs a more formal mathematical approach for the design of control systems. Matrix methods are usually applied which allows the treatment of multiple-input-multiple-output (MIMO) systems. The objective of the design can often be stated precisely in quantitative terms in the form of a performance index, and the control variables are determined via application of a rigorous set of mathematical procedures. The trial-and-error aspect of the classical design method is considerably reduced and, in some cases, eliminated completely.

This section of notes introduces the subject of controller design using modern control methods. In particular, state feedback control, with and without a full state observer, is introduced and illustrated with some numerical examples. The idea of a simple classical proportional controller is also revisited, and the combination of classical and modern control is used to help explain what is really happening with state control.

The development of these subjects is broken into two parts. The first challenge deals with the design of the control system. The root locus method is used to obtain the appropriate controller gain for the simple proportional controller and the pole placement method is used to obtain the gain matrix within the state control formulation (a similar method is used to obtain the observer gains). Once the design parameters are known, our focus then turns to addressing the actual simulation of the system with preset control parameters. In the simulation mode we address both linear and nonlinear systems (note that the controller design step is usually performed with a linear model of the system).

Finally, as an illustration of the benefits of the state-space design method, a sequence of Matlab examples is given for the so-called ***inverted pendulum*** problem. This system is highly unstable, and a robust control system is required for stable performance. Additionally, since the dynamics

of the inverted pendulum is governed by a set of nonlinear equations, we can also explore the impact of using linear models to design a controller for a nonlinear system.

Design of Control Systems

A linear model of the plant or system of interest is usually used in the design of control systems. For linear time invariant (LTI) systems the state-space formulation of the plant model is given by

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}\,\underline{u} \qquad \text{and} \qquad \underline{y} = \underline{\underline{C}}\,\underline{x} \tag{7.1}$$

where we have assumed that the output of the plant is not a direct function of the input (i.e. $\underline{\underline{D}} = \underline{\underline{0}}$). This can also be represented in terms of transfer functions, or

$$\underline{Y}(s) = \underline{\underline{G}}(s)\underline{U}(s) \qquad \text{with} \qquad \underline{\underline{G}}(s) = \underline{\underline{C}}(s\underline{\underline{I}} - \underline{\underline{A}})^{-1}\underline{\underline{B}} \tag{7.2}$$

A block diagram of the LTI model of the plant is shown in Fig. 7.1.

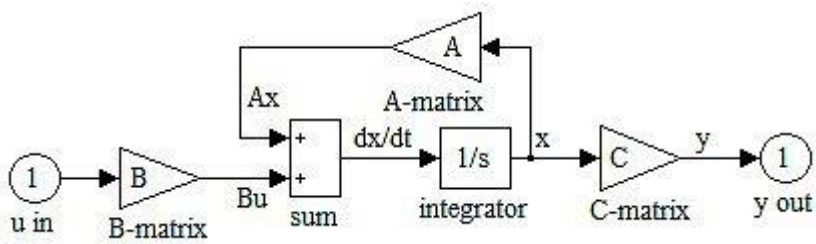


Fig. 7.1 Linear state-space model of the plant.

Classical Proportional Control (SISO)

As an example of classical control, consider the simple closed loop system shown in Fig. 7.2. For a SISO system, the plant transfer function, G(s), is simply the SISO representation of eqn. (7.2) -- with a single u and single y -- and this is embedded within the plant block in Fig. 7.2. The feedback loop contains the sensor transfer function, H(s), and the controller block simply contains the scalar gain, K_c. r_d is the setpoint or desired response of the closed loop system.

The closed loop transfer function for the system in Fig. 7.2 can be written as

$$Y(s) = G(s)K_cE(s) = G(s)K_c\left(R_d(s) - H(s)Y(s)\right)$$

$$Y(s) = \frac{K_cG(s)}{1 + K_cG(s)H(s)}R_d(s) = G_c(s)R_d(s) \tag{7.3}$$

where $G_c(s)$ is the closed loop transfer function and K_c is the classical proportional gain. For unity feedback, $H(s) = 1$, and $G_c(s)$ simplifies to

$$G_c(s) = \frac{K_cG(s)}{1 + K_cG(s)} \tag{7.4}$$

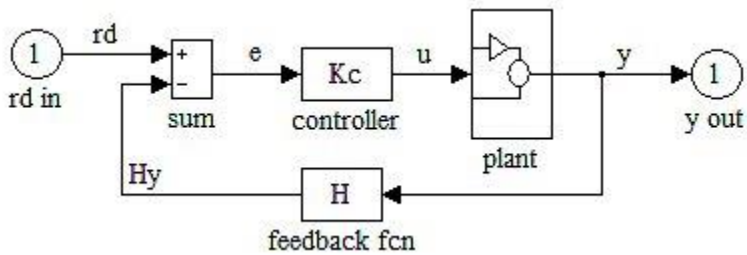


Fig. 7.2 Classical proportional control of a SISO system.

In this case we can write the time domain representation of the scalar input function as

$$u(t) = K_c\left(r_d(t) - y(t)\right) = K_c\left(r_d(t) - \underline{\underline{C}}\,\underline{x}(t)\right) \tag{7.5}$$

and with this u(t), eqn. (7.1) for the SISO system becomes

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + K_c\underline{\underline{B}}\,r_d - K_c\underline{\underline{B}}\,\underline{\underline{C}}\,\underline{x} = (\underline{\underline{A}} - K_c\underline{\underline{B}}\,\underline{\underline{C}})\underline{x} + K_c\underline{\underline{B}}\,r_d \tag{7.6}$$

where the setpoint value r_d is now recognized as the independent input to the system. Also note that, for a SISO system, $\underline{\underline{B}}$ is a column vector and $\underline{\underline{C}}$ is a row vector (both of length N which represents the order of the system).

The controller design for this system is particularly straightforward since there is only a single scalar gain, K_c, that needs to be determined. The transient response of the closed loop system is determined by the eigenvalues of the state matrix or by the poles of the overall transfer function. In the time domain, one would choose K_c to force the eigenvalues of $(\underline{\underline{A}} - K_c\underline{\underline{B}}\,\underline{\underline{C}})$ to give the desired transient response (rise time, settling time, maximum overshoot, etc.). Similarly, in the frequency domain or transfer function representation, one needs to find the K_c that gives the desired pole locations for the transfer function in eqn. (7.4). These are equivalent statements of the same design problem. Also recall that the poles of $G_c(s)$ are the roots of $1 + K_c G(s)$, where G(s) is the plant transfer function. Thus, for the case of simple proportional control, the most common method for determining the best value of gain is to plot the location of the poles or eigenvalues as a function of K_c. Then one selects the gain that comes closest to the desired pole locations as determined from the specified step response characteristics. This procedure, which is referred to as the ***root locus method***, was illustrated in Case Study E for the ***Light Tracking Servo System*** and it will not be repeated here.

State Feedback Control (SISO)

The primary disadvantage of the classical control strategy given above is that there is only a single free variable, K_c, that can be adjusted. However, for an Nth order system, there are N eigenvalues of the open loop state matrix or N poles of the open loop system transfer function given by

$$\det\,(\underline{\underline{A}} - \lambda\underline{\underline{I}}) = 0 \qquad \text{or} \qquad \det\,(s\underline{\underline{I}} - \underline{\underline{A}}) = 0$$

If the design goal for the controlled system is to move these N poles to more appropriate locations, it certainly makes sense that additional degrees of freedom (i.e. more free variables)

should allow more freedom in placing the closed loop poles as desired. In particular, if there were N control variables for an Nth order system, we conceivably could arbitrarily place all the poles of the system in any desired location.

With this discussion in mind, instead of feeding back a single output variable, let's feed back the full state vector, $\underline{x}$, multiplied by a matrix of gains, $\underline{\underline{K}}_s$, where the s subscript refers to state feedback (above the K_c referred to the classical gain). $\underline{\underline{K}}_s$ is known as the state feedback gain matrix. For the case of a SISO plant, the manipulated variable becomes

$$u(t) = r_d(t) - \underline{\underline{K}}_s \underline{x}(t) \tag{7.7}$$

where the state feedback gain matrix is simply a row vector of length N, or

$$\underline{\underline{K}}_s \underline{x} = \begin{bmatrix} k_1 & k_2 & \cdots & k_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

which is a scalar quantity.

With this expression for the input, u(t), the state-space equations for the closed loop system become

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}\, r_d - \underline{\underline{B}}\underline{\underline{K}}_s \underline{x} = \left(\underline{\underline{A}} - \underline{\underline{B}}\underline{\underline{K}}_s\right)\underline{x} + \underline{\underline{B}}\, r_d \qquad \text{and} \qquad y = \underline{\underline{C}}\underline{x} \tag{7.8}$$

This system is pictured in Fig. 7.3. This is quite different from the block diagram in Fig. 7.2; the gain block is in the feedback loop, it contains a vector of gains instead of just a scalar gain, and the state vector is being feed back instead of the output, y(t). In addition, the plant model is modified slightly to include two output paths -- the desired response, y(t), and the state vector, $\underline{x}(t)$. This new plant model is shown in Fig. 7.4 (it is only slightly different from Fig. 7.1).

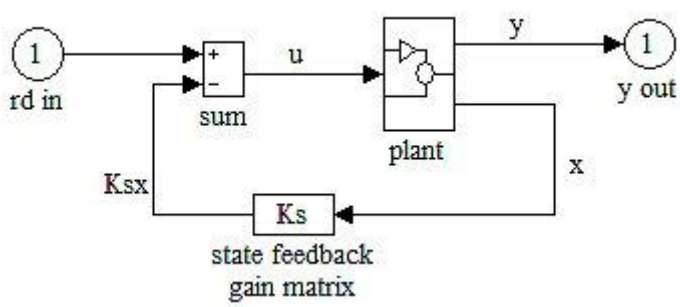


Fig. 7.3 State feedback control of a SISO plant.

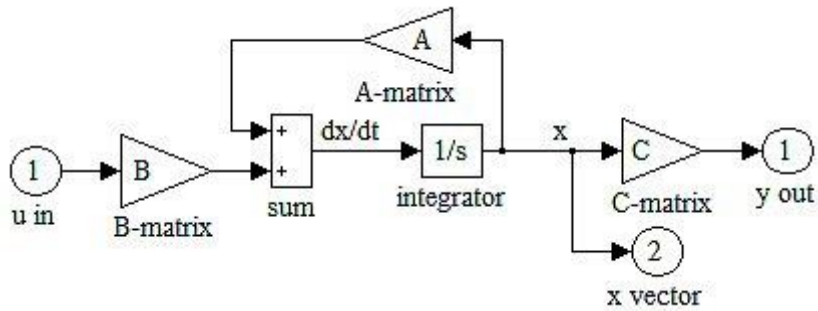


Fig. 7.4 Linear state-space model of the plant with two outputs.

The transient response of this state controlled system is determined by the eigenvalues of $\left(\underline{\underline{A}} - \underline{\underline{B}}\underline{\underline{K}}_s\right)$. Varying the feedback gains can alter the pole locations and can help achieve the desired transient response. In fact, if the system is ***completely state controllable*** (see below), then the N elements of the gain matrix can be specified to give any desired location for the N eigenvalues of $\left(\underline{\underline{A}} - \underline{\underline{B}}\underline{\underline{K}}_s\right)$.

Complete State Controllability

Given the SISO LTI system defined by

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}u \qquad \text{and} \qquad y = \underline{\underline{C}}\underline{x} \tag{7.9}$$

the system is said to be state controllable at $t = t_0$ if it is possible to construct an unconstrained control signal, u(t), that will transfer an initial state to ***any*** final state in a finite time interval $t_0 \le t \le t_f$. If every state is controllable, then the system is said to be ***completely state controllable***.

Without loss of generality, let's assume $\underline{x}(t_f) = 0$ and $t_0 = 0$, then

$$\underline{x}(t) = e^{\underline{\underline{A}}t}\underline{x}(0) + \int_0^t e^{\underline{\underline{A}}(t-\tau)}\underline{\underline{B}}u(\tau)d\tau$$

and applying the definition of complete state controllability, we have

$$\underline{x}(t_f) = 0 = e^{\underline{\underline{A}}t_f}\underline{x}(0) + \int_0^{t_f} e^{\underline{\underline{A}}(t_f-\tau)}\underline{\underline{B}}u(\tau)d\tau$$

or $$\underline{x}(0) = -\int_0^{t_f} e^{-\underline{\underline{A}}\tau}\underline{\underline{B}}u(\tau)d\tau$$

Now using Sylvester's Interpolation Formula (derived in Ogata's Modern Control Engineering textbook)

$$e^{\underline{\underline{A}}\tau} = \sum_{k=0}^{N-1}\alpha_k(\tau)\underline{\underline{A}}^k \tag{7.10}$$

we have

$$\underline{x}(0) = -\sum_{k=0}^{N-1} \underline{\underline{A}}^k \underline{\underline{B}} \int_0^{t_f} \alpha_k(\tau) u(\tau) d\tau = -\sum_{k=0}^{N-1} \underline{\underline{A}}^k \underline{\underline{B}}\, \beta_k$$

or $$\underline{x}(0) = \begin{bmatrix} \underline{\underline{B}} & \underline{\underline{A}}\,\underline{\underline{B}} & \underline{\underline{A}}^2\underline{\underline{B}} & \cdots & \underline{\underline{A}}^{N-1}\underline{\underline{B}} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{N-1} \end{bmatrix}$$

This system of simultaneous equations can be satisfied if and only if the vectors $\underline{\underline{B}}, \underline{\underline{A}}\,\underline{\underline{B}}, \cdots$ are linearly independent (recall that for a single input system, $\underline{\underline{B}}$ is simply a column vector), or

$$\underline{\underline{M}} = \begin{bmatrix} \underline{\underline{B}} & \underline{\underline{A}}\,\underline{\underline{B}} & \underline{\underline{A}}^2\underline{\underline{B}} & \cdots & \underline{\underline{A}}^{N-1}\underline{\underline{B}} \end{bmatrix} \tag{7.11}$$

has rank N (i.e. is nonsingular). The N×N matrix defined in eqn. (7.11) is called the ***controllability matrix***.

Note that ***Sylvester's Interpolation Formula*** is significantly different from the basic infinite series definition of the matrix exponential, which is given by

$$e^{\underline{\underline{A}}t} = \sum_{k=0}^{\infty} \frac{\underline{\underline{A}}^k t^k}{k!} = \underline{\underline{I}} + \underline{\underline{A}}t + \frac{\underline{\underline{A}}^2 t^2}{2!} + \cdots$$

The essential difference, of course, is that the standard infinite series expression includes an infinite number of terms and the expression in eqn. (7.10) only has a finite number, N. This representation is essential for obtaining the finite matrix formulation given above.

Pole Placement

The easiest strategy for implementing state feedback control is via the so-called ***pole placement*** technique. The basic idea is simply to specify the desired location of all N poles in the closed loop system, and then determine the N elements of the state gain matrix to achieve these poles. If the system is fully state controllable (as defined above), the equality of the closed loop characteristic equation and the characteristic equation formed from the specified pole locations gives a linearly independent system of N equations and N unknowns. Solution of this system of equations gives the required elements of the gain matrix.

For small systems the pole placement method can be implemented via hand calculation. The basic procedure (whether implemented by hand or in an automated fashion within a computer code) is as follows:

1. Check that the rank of the controllability matrix is N.
2. Specify the desired poles of the closed loop system, $\mu_1, \mu_2, \cdots, \mu_N$.

3. With the desired poles given, one can develop the desired characteristic equation, $(s-\mu_1)(s-\mu_2)\cdots(s-\mu_N) = s^N + \alpha_1 s^{N-1} + \cdots + \alpha_N = 0$.
4. Finally, one develops the characteristic equation for the closed loop system, which is given by $\det\left(s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{B}}\underline{\underline{K}}_s\right)\right) = 0$, and equates the coefficients of like powers of s from the desired characteristic equation. This gives N equations for the N unknown elements of $\underline{\underline{K}}_s$.

An illustration of this procedure for a low order system is given in Example 7.1. For higher order systems, one usually uses a different algorithm (see Ogata's Modern Control Engineering textbook, for example) that can be automated within a more efficient overall computational scheme. In Matlab, for example, the **place** command is used to automatically determine the required feedback gain matrix using the pole placement method. A Matlab example that illustrates the use of this function is given later in this section of notes (after we discuss the State Observer).

Example 7.1 Pole Placement Method for a Simple 2nd Order System

Problem Statement:

Given the SISO LTI plant defined by

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}u \qquad \text{where} \qquad \underline{\underline{A}} = \begin{bmatrix} 0 & 1 \\ 20.6 & 0 \end{bmatrix} \qquad \underline{\underline{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$y = \underline{\underline{C}}\underline{x} \qquad \underline{\underline{C}} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

find the elements of the state feedback gain matrix such that the closed loop poles are located at $\mu_{1,2} = -1.8 \pm j2.4$.

Problem Solution:

As a preliminary step, we should find the poles of the open loop system, or

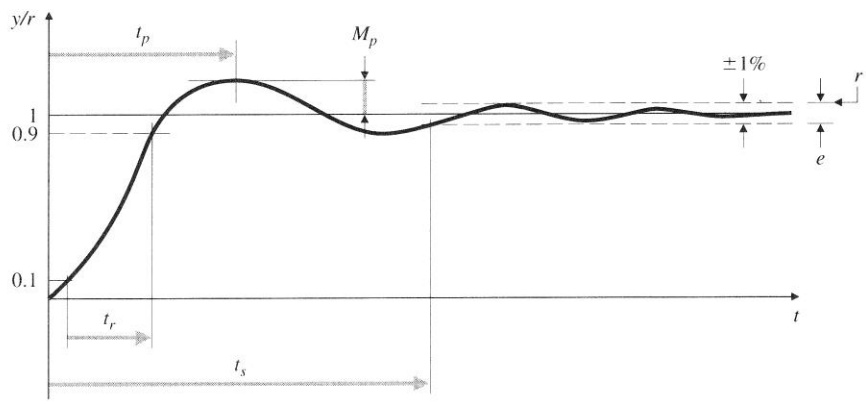
$$\left|s\underline{\underline{I}} - \underline{\underline{A}}\right| = \begin{vmatrix} s & -1 \\ -20.6 & s \end{vmatrix} = s^2 - 20.6 = 0$$

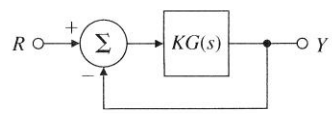
Thus, $s_{1,2} = \pm 4.539$, and we see that the open loop system is unstable, with a real pole quite far into the right half side of the complex plane.

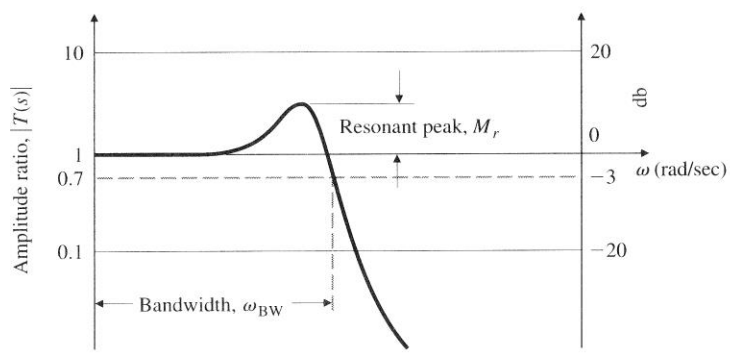
Also, we should note that the choice of the closed loop poles was determined from the desire to have a step response with a relatively fast rise time and settling time and a maximum overshoot of about 10%. We can get a rough estimate of these quantities using the "Design Aids" cover page from **Feedback Control of Dynamic Systems** by Franklin, Powell, and Emami-Naeini (see below):

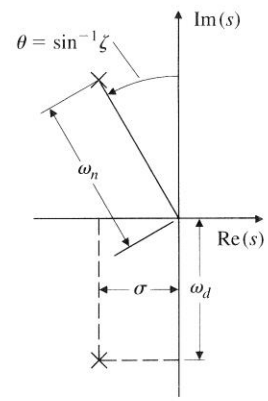
Design Aids

Closed Loop

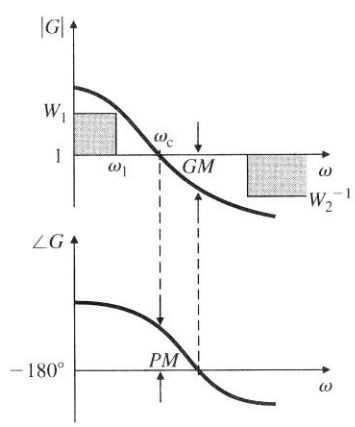








Open Loop



Design Relations

$$t_s = \frac{4.6}{\sigma} \qquad t_r = \frac{1.8}{\omega_n}$$

$$\sigma = \zeta\omega_n \qquad \omega_d = \omega_n\sqrt{1-\zeta^2}$$

$$e_{ss} = \frac{1}{1+K_0}, \quad K_0 = |G(j\omega)|_{\omega=0}$$

$$|E| < \frac{1}{1+W_1}, \quad \omega < \omega_1$$

$$\omega_{Bw} = \omega_c \qquad \text{for } PM = 90^\circ$$

$$\omega_{Bw} = 2\omega_c \qquad \text{for } PM = 45^\circ$$

$$M_r \cong \frac{1}{2\sin(PM/2)}$$

$$M_p = 5\%, \qquad \zeta = 0.7$$

$$M_p = 15\%, \qquad \zeta = 0.5$$

$$M_p = 35\%, \qquad \zeta = 0.3$$

$$\zeta \cong \frac{PM}{100} \quad \text{for } PM < 70^\circ$$

Pole locations $\mu_{1,2} = -\zeta\omega_n \pm j\omega_d$ where $\omega_d = \omega_n\sqrt{1-\zeta^2}$

$$|\mu| = \sqrt{(\zeta\omega_n)^2 + \omega_n^2(1-\zeta^2)} = \omega_n$$

Natural frequency $\omega_n = \sqrt{(1.8)^2 + (2.4)^2} = 3.0$

Damping ratio $\zeta = \dfrac{\sigma}{\omega_n} = \dfrac{1.8}{3.0} = 0.6$

Now from the Design Aids sheet, the maximum overshoot, rise time, and settling time are given by

$$M_p \approx 10\% \qquad t_r = \frac{1.8}{\omega_n} = \frac{1.8}{3.0} \approx 0.6\text{ sec} \qquad t_s = \frac{4.6}{\sigma} = \frac{4.6}{1.8} \approx 2.6\text{ sec}$$

With these values, it appears that a reasonable transient response should result for the closed loop response. These criteria are quite arbitrary, however, and a different set of specifications would lead to a different set of pole locations.

Now with the pole locations specified (and rationalized), we can proceed with finding the gains of the state feedback matrix using the procedure outlined above.

Step 1: Check the rank of the controllability matrix.

$$\underline{\underline{M}} = \begin{bmatrix} \underline{\underline{B}} & \underline{\underline{A}}\,\underline{\underline{B}} \end{bmatrix} = \left[\begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ 20.6 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Thus, the rank is N = 2 and arbitrary pole placement is possible.

Steps 2 & 3: Specify desired poles and develop desired characteristic equation.

$$(s-\mu_1)(s-\mu_2) = (s+1.8-j2.4)(s+1.8+j2.4) = s^2 + 3.6s + 9 = s^2 + \alpha_1 s + \alpha_2 = 0$$

Step 4: Develop the characteristic equation for the closed loop system and equate the coefficients of like powers of s from the desired characteristic equation.

$$\det\left(s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s\right)\right) = \left|s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s\right)\right| = 0$$

and $\underline{\underline{B}}\,\underline{\underline{K}}_s = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\begin{bmatrix} k_1 & k_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ k_1 & k_2 \end{bmatrix}$ and $\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s = \begin{bmatrix} 0 & 1 \\ 20.6 - k_1 & -k_2 \end{bmatrix}$

Therefore,

$$\left|s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s\right)\right| = \begin{vmatrix} s & -1 \\ -20.6 + k_1 & s + k_2 \end{vmatrix} = s^2 + k_2 s - 20.6 + k_1 = 0$$

Finally, equating the coefficients for like powers of s for this polynomial with those from Steps 2 & 3 gives $k_1 = 29.6$ and $k_2 = 3.6$. Thus, the state feedback gain matrix needed to achieve the desired closed loop transient response characteristics is $\underline{\underline{K}}_s = \begin{bmatrix} 29.6 & 3.6 \end{bmatrix}$.

State Feedback with a Full State Observer

The problem with state feedback control is that every element of the state vector is used in the feedback path and, clearly, many states in realistic systems are not easily measurable. In many cases, only a few states are readily available from physical or economical concerns. Without the full state vector, the above development is not possible.

One way around this dilemma is to use an estimate of the unmeasurable states using a mathematical simulation of the system. With this approach, we need to implement a state estimation routine or ***state observer*** into the overall system model, being sure to account for the fact that some states are measurable and may be used to improve the computed estimate.

In the following development, we assume a SISO LTI system. This means that there is a single manipulated variable and a single measurable quantity. This assumption is not necessary in general, but the equations and notation become more complicated for the general case. Thus, in the following development the only measurable quantity is the desired output, y(t), and this will be used within the state observer to help improve the state estimation process. Here we use the notation $\underline{\hat{x}}(t)$ to represent the estimate of the state, $\underline{x}(t)$, at any time t.

Consider the state observer pictured in Fig. 7.5 (note that the variable xc refers to $\underline{\hat{x}}(t)$, etc.). This observer uses u(t) and y(t) as input quantities and it outputs an estimate of the state vector versus time. From the diagram, we have

$$\frac{d}{dt}\underline{\hat{x}} = \underline{\underline{A}}\,\underline{\hat{x}} + \underline{\underline{B}}u + \underline{\underline{L}}\left(y - \underline{\underline{C}}\,\underline{\hat{x}}\right) = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}u + \underline{\underline{L}}y \tag{7.12}$$

where $\underline{\underline{L}}$ is a matrix of unknown gains that is determined based on the desired transient response characteristics for this subsystem. This quantity is known as the ***state observer gain matrix***. For a SISO system, $\underline{\underline{L}}$ is a column vector of length N.

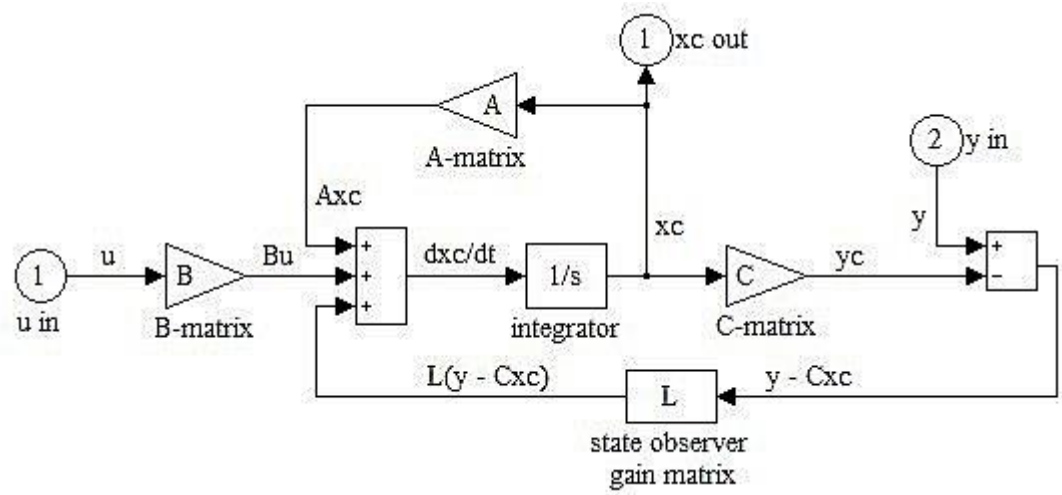


Fig. 7.5 State observer model for a SISO system.

Note that this design problem is similar to that described above for standard state feedback control. Here the observer gain matrix, $\underline{\underline{L}}$, is chosen such that the ***eigenvalues of the state estimator are stable and fast compared to the dynamics of the closed loop system***. The eigenvalues of the state observer are given by

$$\det\left(s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\right) = 0 \tag{7.13}$$

When the state observer is incorporated within the system with state feedback control, we have the block diagram given in Fig. 7.6. For this system, the input to the plant is given by

$$u(t) = r_d(t) - \underline{\underline{K}}_s\,\underline{\hat{x}}(t) \tag{7.14}$$

Note that this expression is very similar to eqn. (7.7) for the state feedback case without state observer. The only difference here is that $\underline{x}(t)$ is replaced by $\underline{\hat{x}}(t)$.

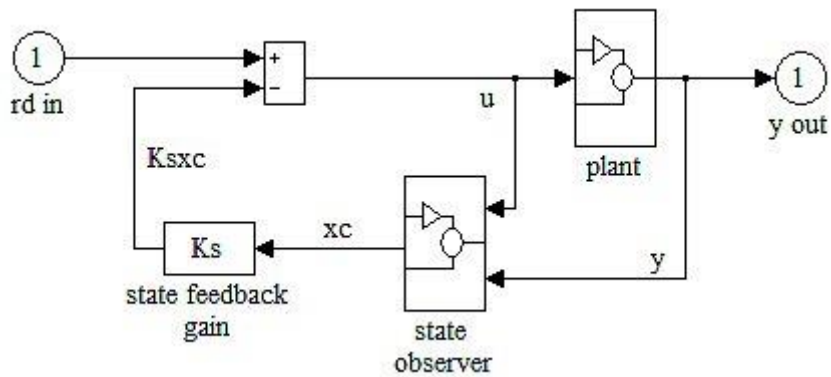


Fig. 7.6 State feedback control with full state observer (SISO system).

Now, if the linearized model of the plant and the state observer are both characterized by the same state space matrices, $\underline{\underline{A}}$, $\underline{\underline{B}}$, and $\underline{\underline{C}}$, then, for the plant, one has

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}u \qquad \text{and} \qquad y = \underline{\underline{C}}\,\underline{x} \tag{7.15}$$

and substituting in the control rule from eqn. (7.14) gives

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}r_d - \underline{\underline{B}}\,\underline{\underline{K}}_s\,\underline{\hat{x}} \qquad \text{(plant dynamics)} \tag{7.16}$$

For the state observer, we can write a similar relationship by substituting eqn. (7.14) into eqn. (7.12), giving

$$\frac{d}{dt}\underline{\hat{x}} = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}r_d - \underline{\underline{B}}\,\underline{\underline{K}}_s\,\underline{\hat{x}} + \underline{\underline{L}}\,\underline{\underline{C}}\,\underline{x}$$

or

$$\frac{d}{dt}\underline{\hat{x}} = \left(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}r_d + \underline{\underline{L}}\,\underline{\underline{C}}\,\underline{x} \qquad \text{(observer dynamics)} \tag{7.17}$$

Now defining an error vector, $\underline{e} = \underline{x} - \underline{\hat{x}}$, and subtracting eqn. (7.17) from eqn. (7.16), gives

$$\frac{d}{dt}\underline{e} = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{e} \qquad \text{(error dynamics)} \qquad (7.18)$$

This expression represents an unforced stable system if the eigenvalues of the state matrix have negative real parts. In this case, $\underline{e}(t)$ approaches zero for large t, which implies that $\hat{\underline{x}}(t) \to \underline{x}(t)$. If the dynamics of $\underline{e}(t)$ are also quite fast compared to the dynamics of $\underline{x}(t)$, then $\hat{\underline{x}}(t)$ becomes a good estimate of the state at any time t.

Since the time domain behavior of the error vector is determined by the eigenvalues of $\left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)$, the N elements of the observer gain matrix, $\underline{\underline{L}}$, can be varied to give the desired transient response time for the error dynamics. In fact, if the system is ***completely state observable*** (see below), then the N elements of the gain matrix (which is really a vector for a SISO system) can be specified to give any desired location for the N eigenvalues of $\left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)$.

Complete State Observability

A system is said to be ***completely state observable*** if every state, $\underline{x}(t_0)$, can be determined from the observation of y(t) over a finite time interval, $t_0 \le t \le t_f$. To develop a test for observability, let's consider the SISO LTI system defined in eqn. (7.9). The time domain solution for this system can be written as

$$y(t) = \underline{\underline{C}}\,e^{\underline{\underline{A}}t}\,\underline{x}(0) + \underline{\underline{C}}\int_0^t e^{\underline{\underline{A}}(t-\tau)}\,\underline{\underline{B}}u(\tau)d\tau \qquad (7.19)$$

If we let u(t) = 0, for convenience (since for known u(t), the second term in eqn. (7.19) is known precisely), then this expression reduces to

$$y(t) = \underline{\underline{C}}\,e^{\underline{\underline{A}}t}\,\underline{x}(0) \qquad (7.20)$$

Recall here that $\underline{\underline{C}}\,e^{\underline{\underline{A}}t}$ is known and y(t) can be measured. Therefore, the statement of observability concerns the determination of $\underline{x}(0)$ from the observation of y(t) over some period of time.

For the SISO case, eqn. (7.20) represents an underdetermined system of equations with only one equation and N unknowns (for the general case with M outputs or measurable quantities, the system is still underdetermined since M is usually less than N). However, the time dependent nature of the process allows one to make many measurements and, when integrated over time, the full information obtained in y(t) may be sufficient to uniquely determine the initial state, $\underline{x}(0)$.

Continuing our efforts to develop a test for complete state observability, consider the following manipulations of eqn. (7.20). First, let's multiply both sides by the transpose of the known coefficient matrix (assuming real elements of the matrix), or

$$\left(\underline{\underline{C}}\,e^{\underline{\underline{A}}t}\right)^T y = \left(\underline{\underline{C}}\,e^{\underline{\underline{A}}t}\right)^T \underline{\underline{C}}\,e^{\underline{\underline{A}}t}\,\underline{x}(0)$$

Now, rewriting the transposed matrix as

$$\left(\underline{\underline{C}}e^{\underline{\underline{A}}t}\right)^T = \left(e^{\underline{\underline{A}}t}\right)^T \underline{\underline{C}}^T = e^{\underline{\underline{A}}^T t}\underline{\underline{C}}^T$$

gives the more manageable expression,

$$e^{\underline{\underline{A}}^T t}\underline{\underline{C}}\,y = e^{\underline{\underline{A}}^T t}\underline{\underline{C}}^T\underline{\underline{C}}e^{\underline{\underline{A}}t}\underline{x}(0)$$

Integrating both sides of this relationship over the observation time gives

$$\underline{Q} = \underline{\underline{W}}\underline{x}(0) \tag{7.21}$$

where $\underline{Q} = \int_0^{t_f} e^{\underline{\underline{A}}^T\tau}\underline{\underline{C}}\,y(\tau)d\tau$ and $\underline{\underline{W}} = \int_0^{t_f} e^{\underline{\underline{A}}^T\tau}\underline{\underline{C}}^T\underline{\underline{C}}e^{\underline{\underline{A}}\tau}d\tau$ (7.22)

Finally, solving eqn. (7.21) for $\underline{x}(0)$ gives

$$\underline{x}(0) = \underline{\underline{W}}^{-1}\underline{Q} \tag{7.23}$$

If $\underline{\underline{W}}$ is nonsingular, then $\underline{x}(0)$ can be uniquely determined from observation of y(t), and the system is said to be completely state observable.

To put the observability test into final form, we again use Sylvester's Interpolation Formula given in eqn. (7.10). Using this representation for the matrix exponential gives

$$\underline{\underline{C}}e^{\underline{\underline{A}}t} = \sum_{k=0}^{N-1}\alpha_k(t)\underline{\underline{C}}\,\underline{\underline{A}}^k = \begin{bmatrix}\alpha_0 & \alpha_1 & \cdots & \alpha_{N-1}\end{bmatrix}\begin{bmatrix}\underline{\underline{C}} \\ \underline{\underline{C}}\,\underline{\underline{A}} \\ \underline{\underline{C}}\,\underline{\underline{A}}^2 \\ \vdots \\ \underline{\underline{C}}\,\underline{\underline{A}}^{N-1}\end{bmatrix}$$

and

$$e^{\underline{\underline{A}}^T t}\underline{\underline{C}}^T = \sum_{k=0}^{N-1}\gamma_k(t)\underline{\underline{A}}^{T^k}\underline{\underline{C}}^T = \begin{bmatrix}\underline{\underline{C}}^T & \underline{\underline{A}}^T\underline{\underline{C}}^T & \underline{\underline{A}}^{T^2}\underline{\underline{C}}^T & \cdots & \underline{\underline{A}}^{T^{N-1}}\underline{\underline{C}}^T\end{bmatrix}\begin{bmatrix}\gamma_0 \\ \gamma_1 \\ \vdots \\ \gamma_{N-1}\end{bmatrix}$$

Let's define

$$\underline{\underline{H}} = \begin{bmatrix}\underline{\underline{C}} \\ \underline{\underline{C}}\,\underline{\underline{A}} \\ \underline{\underline{C}}\,\underline{\underline{A}}^2 \\ \mathrm{M} \\ \underline{\underline{C}}\,\underline{\underline{A}}^{N-1}\end{bmatrix} \quad \text{and} \quad \underline{\underline{H}}^T = \begin{bmatrix}\underline{\underline{C}}^T & \underline{\underline{A}}^T\underline{\underline{C}}^T & \underline{\underline{A}}^{T^2}\underline{\underline{C}}^T & \mathrm{L} & \underline{\underline{A}}^{T^{N-1}}\underline{\underline{C}}^T\end{bmatrix} \tag{7.24}$$

The matrix $\underline{\underline{H}}^T$ (or sometimes $\underline{\underline{H}}$) is referred to as the ***observability matrix*** for a SISO system (assuming real matrices). If the rank of $\underline{\underline{H}}^T$ or $\underline{\underline{H}}$ is N (notice that $\underline{\underline{H}}^T$ is simply the transpose of $\underline{\underline{H}}$), then the coefficient matrix, $\underline{\underline{W}}$, in eqn. (7.21) will be nonsingular and the system is

completely state observable (see Ogata's **Modern Control Engineering** text for justification of this last argument, for example).

As a simple example to help explain this result, consider a SISO system defined by the 2x2 state matrix, $\underline{\underline{A}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Let's identify two cases; one whose output $y_A(t)$ is the first state, $x_1(t)$, and another whose output is the second state, or $y_B(t) = x_2(t)$. For these two cases we have

Case A: $\underline{\underline{C}} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ then $\underline{\underline{H}}_A^T = \begin{bmatrix} 1 & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$ which has Rank = 2

Case B: $\underline{\underline{C}} = \begin{bmatrix} 0 & 1 \end{bmatrix}$ then $\underline{\underline{H}}_B^T = \begin{bmatrix} 0 & \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$ which has Rank = 1

Notice that from the state matrix, $\underline{\underline{A}}$, we see that x_1 is a function of x_2. Therefore, observation of the first state, $y = x_1$, gives information about both states. Thus, Case A is completely state observable. However, the state matrix also indicates that the second state, x_2, is independent of x_1. Therefore, observation of $y = x_2$ cannot give information about x_1, and Case B is not completely state observable. Checking the rank of the observability matrix simply gives a formal methodology for evaluating the observability condition.

Determining the Observer Gains

The procedure for finding the elements of the observer gain matrix is based on the same pole placement method that was used for determining the state feedback gains. In this case, however, one specifies the pole locations for the error dynamics of the state estimator. The selection here is somewhat arbitrary, but the overall dynamics should be relatively fast compared to the plant dynamics. If the system is completely state observable, the specification of the N eigenvalues for $\left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}}\right)$ should allow a unique determination of the N elements of the observer gain matrix, $\underline{\underline{L}}$ (which, of course, is just a column vector of length N for the case of an SISO system).

The procedure can be summarized as follows:

1. Check that the rank of the observability matrix is N.
2. Specify the desired pole locations for the error vector, $\underline{e} = \underline{x} - \underline{\hat{x}}$ (the poles, $\mu_1, \mu_2, \cdots, \mu_N$, should be further into the left hand side of the complex plane than the dominant poles associated with the plant dynamics).
3. With the desired poles given, one can develop the desired characteristic equation, $(s-\mu_1)(s-\mu_2)\cdots(s-\mu_N) = s^N + \alpha_1 s^{N-1} + \cdots + \alpha_N = 0$.
4. Finally, one develops the characteristic equation for the state error vector, which is given by $\det\left(s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}}\right)\right) = 0$, and equates the coefficients of like powers of s from the desired characteristic equation. This gives N equations for the N unknown elements of $\underline{\underline{L}}$.

A sample problem showing this procedure for a low order system is given in Example 7.2. This problem is based on the same system used in Example 7.1. For the present case, the observer dynamics are chosen to be three times faster than the plant dynamics. This example gives a good illustration of the hand calculations required in the design of a state observer. The procedure is very similar to the steps required for finding the state feedback gain matrix.

Example 7.2 An Observer Design Example for a Simple 2nd Order System

Problem Statement:

Given the SISO LTI plant defined by

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}u \qquad \text{where} \qquad \underline{\underline{A}} = \begin{bmatrix} 0 & 1 \\ 20.6 & 0 \end{bmatrix} \qquad \underline{\underline{B}} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$y = \underline{\underline{C}}\underline{x} \qquad \underline{\underline{C}} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

find the elements of the state observer gain matrix such that the closed loop poles associated with the error dynamics are three times faster than the closed loop plant poles specified in Example 7.1.

Problem Solution:

If the system is completely state observable and the dynamics of $\underline{e}(t)$ are fast compared to the dynamics of $\underline{x}(t)$, then the observer will give a good estimate of the plant states. To obtain the required observer gains, let's follow the procedure given above:

Step 1: Check the rank of the observability matrix.

$$\underline{\underline{H}}^T = \begin{bmatrix} \underline{\underline{C}}^T & \underline{\underline{A}}^T\underline{\underline{C}}^T \end{bmatrix} = \begin{bmatrix} 1 & \begin{bmatrix} 0 & 20.6 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Thus, the rank is N = 2 and the system, as specified, is fully state observable.

Steps 2 & 3: Specify desired poles and develop the corresponding characteristic equation.

Let's choose the roots of $\det\left[s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}}\right)\right] = 0$ to be three times faster than the roots of the closed loop system. Note that if we simply multiply the closed loop poles by 3, the damping ratio and peak overshoot will be unchanged and the rise time and settling time will be 3 times faster. Therefore, since $\mu_{clp} = -1.8 \pm j2.4$, let's choose $\mu_{obs} = -5.4 \pm j7.2$. Thus, the desired characteristic equation is

$$(s-\mu_1)(s-\mu_2) = (s+5.4-j7.2)(s+5.4+j7.2) = s^2 + 10.8s + 81 = s^2 + \alpha_1 s + \alpha_2 = 0$$

Step 4: Develop the characteristic equation for the error dynamics and equate the coefficients of like powers of s from the desired characteristic equation.

$$\det\left(s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}}\right)\right) = \left|s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}}\right)\right| = 0$$

and $\quad \underline{\underline{L}}\underline{\underline{C}} = \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} l_1 & 0 \\ l_2 & 0 \end{bmatrix} \quad$ and $\quad \underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}} = \begin{bmatrix} -l_1 & 1 \\ 20.6 - l_2 & 0 \end{bmatrix}$

Therefore,

$$\left| s\underline{\underline{I}} - \left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}} \right) \right| = \begin{vmatrix} s + l_1 & -1 \\ -20.6 + l_2 & s \end{vmatrix} = s^2 + l_1 s - 20.6 + l_2 = 0$$

Finally, equating the coefficients for like powers of s for this polynomial with those from Steps 2 & 3 gives $\lambda_1 = 10.8$ and $\lambda_2 = 101.6$. Thus, the observer gain matrix needed to achieve the desired transient response for the error dynamics is $\underline{L} = \begin{bmatrix} 10.8 & 101.6 \end{bmatrix}^T$.

A Matlab Example

As was the case for determining the state feedback gains, Matlab also has an automated procedure for computing the elements of the observer gain matrix. In fact, Matlab's **place** command is used again for this purpose. To see this, first recall that the eigenvalues of a matrix and its transpose are identical. Therefore, we have

$$\left(\underline{\underline{A}} - \underline{\underline{L}}\underline{\underline{C}} \right)^T = \underline{\underline{A}}^T - \left(\underline{\underline{L}}\underline{\underline{C}} \right)^T = \left(\underline{\underline{A}}^T - \underline{\underline{C}}^T \underline{\underline{L}}^T \right)$$

which has the same form as the state matrix for the feedback gain design problem, $\left(\underline{\underline{A}} - \underline{\underline{B}}\underline{\underline{K}}_s \right)$. Thus, the same function can be used to find the state feedback gains and the state observer gains. For the feedback gain problem, one passes the $\underline{\underline{A}}$ and $\underline{\underline{B}}$ matrices into the **place** function, and for the observer gain design problem, one passes the $\underline{\underline{A}}^T$ and $\underline{\underline{C}}^T$ matrices into **place**.

Table 7.1 contains a listing of a sample Matlab file for simulating the same hand calculations performed in Examples 7.1 and 7.2. This file, **sfsotest1.m**, simply illustrates how Matlab might be used to design a plant with state feedback control using a full state observer. It highlights the use of Matlab's **place** command (as discussed above) and it also uses the **ctrb** and **obsv** functions for checking the controllability and observability of the system. Finally, the **sfsotest1.m** file also illustrates how to simulate the closed loop system once the desired gains have been determined (this is discussed in more detail in the next subsection).

Summary output from this test simulation is given in Table 7.2 and in Figs. 7.7 and 7.8. The diary file listed in Table 7.2 gives some of the intermediate results for the state feedback and state observer gain calculations, and it helps the reader follow the individual steps in the **sfsotest1.m** listing in Table 7.1.

The simulation results from the state feedback case and the state feedback with observer case are compared in Fig. 7.7. Note that the desired output is $x_1(t)$, and we specified a design with about a 10% overshoot and a settling time of about 2.6 seconds. Notice that both cases give essentially identical results and they do indeed give the desired transient response characteristics. The cases with and without the full observer are identical because the observer dynamics were chosen to be very fast relative to the plant dynamics. Thus, the actual error, $\underline{e}(t) = \underline{x}(t) - \hat{\underline{x}}(t)$, should be quite

small. This fact is highlighted in Fig. 7.8, which plots the difference between the plant and observer states versus time. Clearly the errors here are negligibly small (magnitudes are scaled by 1.0e-15)!

Table 7.1 Listing of Matlab example file sfsotest1.m.

```
%
%   SFSOTEST1.M  Test Problem for State Feedback & State Observer Design
%
%   This is a Matlab equivalent of the simple 2x2 test problem done in the
%   Lecture Notes.
%
%   This file is broken into three sections, as follows:
%     Part I. Setup base data for the linear model and show that the base open loop
%             plant is unstable.
%    Part II. Add state feedback control to stabilize the system and simulate the
%             system behavior for a step change in the state 1 reference point.
%   Part III. Add state feedback control and a full observer to stabilize the
%             system. Simulate system behavior for a step change in the state 1
%             reference point. This should give the same simulation as Part II.
%
%   File prepared by J. R. White, UMass-Lowell (last update: Feb. 2020)
%

      clear all,  close all, nfig = 0;
      format compact
%
%   Part I. Setup base data for the linear model and show that the open loop
%           plant is unstable.
%
%   create state space matrices for plant (output state 1 value)
      A = [0  1;  20.6  0];   B = [0 1]';   C = [1  0];   D = [0];
      disp('   *** Results for SFSOTest1 ***'),   disp('  ')
      disp('State Space Matrices for the Plant')
      A,  B,  C,  D
%
%   compute eigenvalues of state matrix for open loop plant
      disp('Eigenvalues of the "Open Loop Plant"');     ev = eig(A)
%
%
%   Part II. Add state feedback control to stabilize the system and simulate
%            system behavior for a step change in the state 1 reference point.
%
%   check for full state controllability
      disp('Controllability Matrix for this system'),   M = ctrb(A,B)
      disp('Rank of Controllability Matrix'),   rank(M)
%
%   calculate state feedback gains for specified closed loop poles
      clp = [-1.8+2.4j  -1.8-2.4j];
      Ks = place(A,B,clp);
      disp('Desired closed loop poles for state feedback controller');   clp
      disp('State feedback gains needed to give desired poles');         Ks
      disp('Calculated eigenvalues of system with state feedback'); eig(A-B*Ks)
%
%   calculate Nv for zero SS error  (see derivation in notes - next subsection)
      Nv = -1.0/(C*inv(A-B*Ks)*B);
      disp('Setpoint gain for zero SS error');    Nv
%
%   simulate linear plant + controller
      to = 0;  tf = 5;
      t = linspace(to,tf,201);
      syscl1 = ss(A-B*Ks,B*Nv,C,D);
      [y1,t,x1] = step(syscl1,t);
%
%   plot results from state feedback case
      nfig = nfig+1;      figure(nfig)
      subplot(2,1,1),plot(t,x1(:,1),'r-',t,x1(:,2),'g--','LineWidth',2),grid,
      title('SFSOTest1:  States for State Feedback Test Case')
      xlabel('Time (sec)'),ylabel('State Variables')
      legend('x1(t)','x2(t)','Location','East')
%
%   Part III. Add state feedback control and a full observer to stabilize the
%             system. Simulate system behavior for a step change in the state 1
```

```
%             reference point. This should give the same simulation as Part II.
%
%   check for full state observability
      disp('Observability Matrix for this system'),   H = obsv(A,C)
      disp('Rank of Observability Matrix'),   rank(H)
%
%   calculate estimator gains for specified observer poles
      op = 3*clp;   % estimator dynamics is 3 times faster than closed loop poles
      L = place(A',C',op);   L = L';
      disp('Desired observer poles for state feedback controller');   op
      disp('Estimator gains needed to give desired poles');           L
      disp('Calculated eigenvalues of estimator system');      eig(A-L*C)
%
%   setup matrices for plant + controller model
      A11 = A;     A12 = -B*Ks;         B1 = B*Nv;
      A21 = L*C;   A22 = A-L*C-B*Ks;    B2 = B*Nv;
      zz = 0;
      AB = [A11 A12; A21 A22];   BB = [B1; B2];
      CB = [C zz*C];
%
%   simulate linear plant + controller
      syscl2 = ss(AB,BB,CB,D);    [y2,t,x2] = step(syscl2,t);
%
%   separate plant and estimator states
      nn = max(size(A));
      xp2 = x2(:,1:nn);     xe2 = x2(:,nn+1:2*nn);
%
%   plot results from case with full observer
      subplot(2,1,2),plot(t,xp2(:,1),'r-',t,xp2(:,2),'g--','LineWidth',2),grid,
      title('SFSOTest1:  States for State Feedback with Full Observer ')
      xlabel('Time (sec)'),ylabel('State Variables')
      legend('x1(t)','x2(t)','Location','East')
%
%   also plot error for plant vs estimator
      nfig = nfig+1;      figure(nfig)
      plot(t,xp2(:,1)-xe2(:,1),'r-',t,xp2(:,2)-xe2(:,2),'g--','LineWidth',2),grid,
      title('SFSOTest1:  Error (State - Observer) Dynamics')
      xlabel('Time (sec)'),ylabel('Error in State Variables')
      legend('e1(t)','e2(t)')
%
%   end of simulation
```

Table 7.2 Diary file from sfsotest1.m.

```
>> sfsotest1
   *** Results for SFSOTest1 ***

State Space Matrices for the Plant
A =
         0    1.0000
   20.6000         0
B =
     0
     1
C =
     1     0
D =
     0

Eigenvalues of the "Open Loop Plant"
ev =
    4.5387
   -4.5387
Controllability Matrix for this system
M =
     0     1
     1     0
Rank of Controllability Matrix
ans =
     2
Desired closed loop poles for state feedback controller
clp =
  -1.8000 + 2.4000i  -1.8000 - 2.4000i
State feedback gains needed to give desired poles
Ks =
   29.6000    3.6000
Calculated eigenvalues of system with state feedback
ans =
  -1.8000 + 2.4000i
  -1.8000 - 2.4000i
Setpoint gain for zero SS error
Nv =
    9.0000

Observability Matrix for this system
H =
     1     0
     0     1
Rank of Observability Matrix
ans =
     2
Desired observer poles for state feedback controller
op =
  -5.4000 + 7.2000i  -5.4000 - 7.2000i
Estimator gains needed to give desired poles
L =
   10.8000
  101.6000
Calculated eigenvalues of estimator system
ans =
  -5.4000 + 7.2000i
  -5.4000 - 7.2000i >>
```

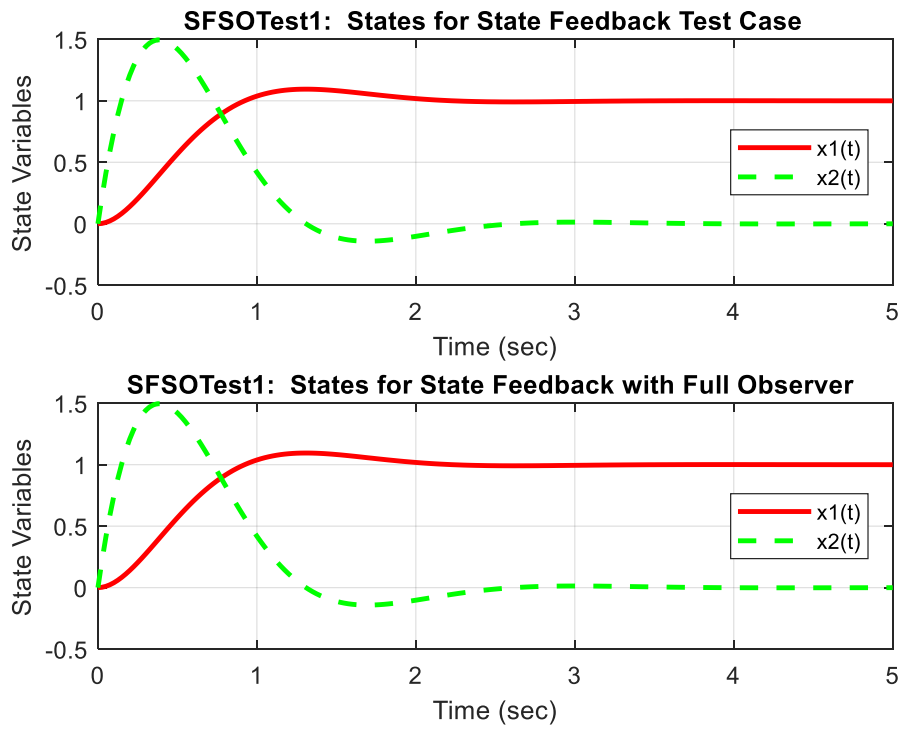


Fig. 7.7 Illustration of the plant and estimator dynamics from sfsotest1.m.

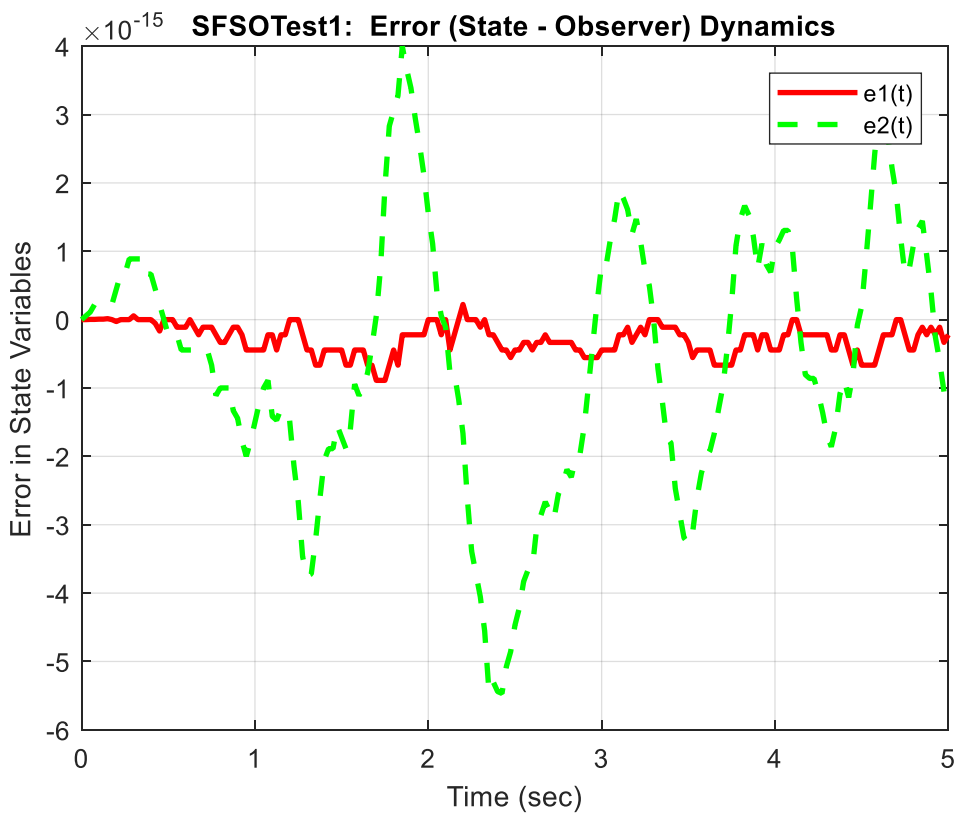


Fig. 7.8 Illustration of the error dynamics from sfsotest1.m.

Principle of Duality

As a final point for this subsection of notes, it should be highlighted that our previous discussions of controllability and observability certainly share a common theme. The final statement of controllability, which requires the rank of the controllability matrix to be N, involves a relationship between the state matrix, $\underline{\underline{A}}$, and the input matrix, $\underline{\underline{B}}$. Similarly, for complete state observability, the rank of the observability matrix, which involves a relationship between $\underline{\underline{A}}$ and the output matrix, $\underline{\underline{C}}$, also must be N. The interrelationship of these two tests is summarized with the definition of duality.

The ***Principle of Duality*** can be stated as follows (here we also generalize to MIMO systems and use the asterisk notation to imply the complex conjugate transpose operation):

System 1 $\quad \frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}\,\underline{u}$ $\qquad$ **System 2** $\quad \frac{d}{dt}\underline{z} = \underline{\underline{A}}^*\,\underline{z} + \underline{\underline{C}}^*\,\underline{v}$ $\qquad$ (7.25)

$\underline{y} = \underline{\underline{C}}\,\underline{x}$ $\qquad$ $\underline{w} = \underline{\underline{B}}^*\,\underline{z}$ $\qquad$ (7.26)

System 2 is said to be the dual of System 1 where $\underline{\underline{A}}^*$, $\underline{\underline{B}}^*$, and $\underline{\underline{C}}^*$ are the complex conjugate transposes of $\underline{\underline{A}}$, $\underline{\underline{B}}$, and $\underline{\underline{C}}$.

System 1

A necessary and sufficient condition for complete state controllability is that the controllability matrix $\underline{\underline{M}}$ has a rank of N, where

$$\underline{\underline{M}}_1 = \left[\underline{\underline{B}} \quad \underline{\underline{A}}\,\underline{\underline{B}} \quad \underline{\underline{A}}^2\underline{\underline{B}} \quad \cdots \quad \underline{\underline{A}}^{N-1}\underline{\underline{B}}\right] \tag{7.27}$$

A necessary and sufficient condition for complete state observability is that the observability matrix $\underline{\underline{H}}^*$ has rank N, where

$$\underline{\underline{H}}_1^* = \left[\underline{\underline{C}}^* \quad \underline{\underline{A}}^*\underline{\underline{C}}^* \quad \underline{\underline{A}}^{*2}\underline{\underline{C}}^* \quad \cdots \quad \underline{\underline{A}}^{*N-1}\underline{\underline{C}}^*\right] \tag{7.28}$$

System 2

The same conditions apply to System 2, where the controllability matrix is

$$\underline{\underline{M}}_2^* = \left[\underline{\underline{C}}^* \quad \underline{\underline{A}}^*\underline{\underline{C}}^* \quad \underline{\underline{A}}^{*2}\underline{\underline{C}}^* \quad \cdots \quad \underline{\underline{A}}^{*N-1}\underline{\underline{C}}^*\right] \tag{7.29}$$

and the observability matrix is

$$\underline{\underline{H}}_2^* = \left[\underline{\underline{B}} \quad \underline{\underline{A}}\,\underline{\underline{B}} \quad \underline{\underline{A}}^2\underline{\underline{B}} \quad \cdots \quad \underline{\underline{A}}^{N-1}\underline{\underline{B}}\right] \tag{7.30}$$

The observability of a given system can be checked by testing the state controllability of its dual. Similarly, the controllability a given system can be checked by testing the state observability of its dual. These statements are referred to as the ***Principle of Duality***.

Time Domain Simulation of Controlled Systems

The previous subsection addressed the process of obtaining the proper controller design parameters using examples of both classical and state feedback control. The focus was on the time domain representation, where the eigenvalues of the closed loop state matrix determines the transient response of the closed loop system. The goal of the controller design problem is to choose the controller gains such that the N eigenvalues of the closed loop state matrix correspond to the desired pole locations.

Once the controller design parameters have been determined, the emphasis shifts towards simulation of the closed loop system. In the design mode, the plant model is usually a linear representation of the real system. However, in the simulation mode, one usually tries to simulate the plant dynamics as accurately as possible. This often requires the simulation of a time varying or nonlinear system.

In this section, we focus on the simulation problem assuming that the controller gains have already been determined. In keeping with the examples in the previous subsection, we also restrict the current analysis to SISO systems. The equations for the closed loop dynamics are written for both linear and nonlinear plant models. The plant models used in the subsequent development are:

Linear Plant $$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}u \quad \text{and} \quad y = \underline{\underline{C}}\,\underline{x} \tag{7.31}$$

Nonlinear Plant $$\frac{d}{dt}\underline{x} = \underline{f}(\underline{x}, u, t) \quad \text{and} \quad y = \underline{\underline{C}}\,\underline{x} \tag{7.32}$$

Classical Control (with present gain)

A block diagram for simple proportional control with unity feedback is shown in Fig. 7.9.

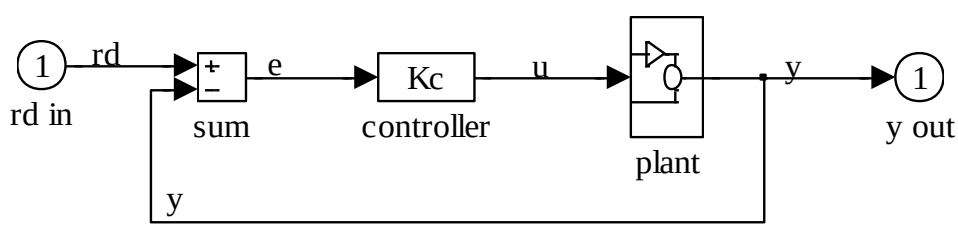


Fig. 7.9 Classical proportional control with unity feedback for a SISO system.

The control law for this system is

$$u = K_c\left(r_d - y\right) \tag{7.33}$$

The closed loop simulation equation using a ***linear plant model*** was given previously in eqn. (7.6) as

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + K_c\underline{\underline{B}}\,r_d - K_c\underline{\underline{B}}\,\underline{\underline{C}}\,\underline{x} = (\underline{\underline{A}} - K_c\underline{\underline{B}}\,\underline{\underline{C}})\underline{x} + K_c\underline{\underline{B}}\,r_d$$

Writing this in standard state-space form for simulation in Matlab, for example, gives

$$\frac{d}{dt}\underline{x} = \underline{\underline{\bar{A}}}\,\underline{x} + \underline{\underline{\bar{B}}}\,r_d \qquad \text{and} \qquad y = \underline{\underline{\bar{C}}}\,\underline{x} \tag{7.34}$$

with $\underline{\underline{\bar{A}}} = \underline{\underline{A}} - K_c\underline{\underline{B}}\,\underline{\underline{C}}$ $\qquad \underline{\underline{\bar{B}}} = K_c\underline{\underline{B}}$ $\qquad \underline{\underline{\bar{C}}} = \underline{\underline{C}}$ (7.35)

If the plant simulation uses a ***nonlinear plant representation***, the complete simulation must be performed using a standard ODE solver (like Matlab's **ode23** or **ode45** routines, for example). In this case, the user-defined function file called by the ODE solver would include the following algorithm and equations (for known state vector, $\underline{x}(t)$, at time t):

1. specify the set point value at time t, $r_d(t)$
2. evaluate the output at time t, $y(t) = \underline{\underline{C}}\,\underline{x}(t)$
3. determine the manipulated input at time t, $u(t) = K_c\left(r_d(t) - y(t)\right)$
4. evaluate the state derivatives at time t, $\frac{d}{dt}\underline{x}(t) = \underline{f}\left(\underline{x}(t), u(t), t\right)$

Finally, upon return from the ODE solver, one can re-compute the output function for all time by simply evaluating $y(t) = \underline{\underline{C}}\,\underline{x}(t)$.

State Feedback Control (with preset feedback and observer gains)

A block diagram for state feedback control with a full state observer is given in Fig. 7.10. This diagram is very similar to that given in Fig. 7.6, except this model has an additional steady state gain block containing a normalizing gain, N_r. This gain is determined so that the steady-state error vanishes (see below).

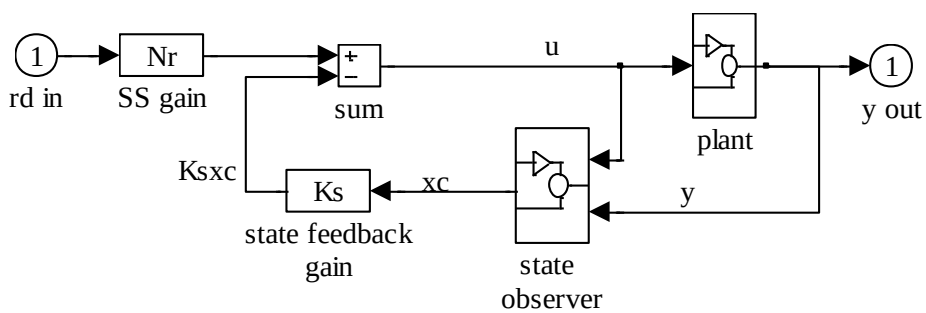


Fig. 7.10 State feedback controller with full state observer.

Again, we can look at the simulation equations for this system for both linear and nonlinear plant models and preset controller gains (i.e. $\underline{\underline{K}}_s$, $\underline{\underline{L}}$, and N_r have known values). For this control scheme, the control law can be written as

$$u = N_r r_d - \underline{\underline{K}}_s\,\underline{\hat{x}} \tag{7.36}$$

The closed loop dynamics for this system contains both the actual plant states, $\underline{x}(t)$, and the estimated plant states, $\underline{\hat{x}}(t)$. Thus, we have 2N unknowns (N plant states and N estimated states). With a ***linear plant model***, the fully coupled dynamics for this system are written as

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}N_r r_d - \underline{\underline{B}}\,\underline{\underline{K}}_s\,\underline{\hat{x}} \qquad \text{(plant dynamics from eqn. (7.16))}$$

$$\frac{d}{dt}\underline{\hat{x}} = \left(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}r_d + \underline{\underline{L}}\,\underline{\underline{C}}\,\underline{x} \qquad \text{(observer dynamics from eqn. (7.17))}$$

This coupled set of matrix equations can be written in standard state form as

$$\frac{d}{dt}\underline{z} = \underline{\underline{\bar{A}}}\,\underline{z} + \underline{\underline{\bar{B}}}r_d \qquad \text{and} \qquad y = \underline{\underline{\bar{C}}}\,\underline{z} \tag{7.37}$$

$$\text{with} \quad \underline{z} = \begin{bmatrix} \underline{x} \\ \underline{\hat{x}} \end{bmatrix} \qquad \underline{\underline{\bar{A}}} = \begin{bmatrix} \underline{\underline{\bar{A}}}_{11} & \underline{\underline{\bar{A}}}_{12} \\ \underline{\underline{\bar{A}}}_{21} & \underline{\underline{\bar{A}}}_{22} \end{bmatrix} \qquad \underline{\underline{\bar{B}}} = \begin{bmatrix} \underline{\underline{B}}N_r \\ \underline{\underline{B}}N_r \end{bmatrix} \qquad \underline{\underline{\bar{C}}} = \begin{bmatrix} \underline{\underline{C}} & \underline{\underline{0}} \end{bmatrix} \tag{7.38}$$

$$\text{and} \quad \underline{\underline{\bar{A}}}_{11} = \underline{\underline{A}} \qquad \underline{\underline{\bar{A}}}_{12} = -\underline{\underline{B}}\,\underline{\underline{K}}_s \qquad \underline{\underline{\bar{A}}}_{21} = \underline{\underline{L}}\,\underline{\underline{C}} \qquad \underline{\underline{\bar{A}}}_{22} = \underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s - \underline{\underline{L}}\,\underline{\underline{C}} \tag{7.39}$$

As before, if the plant simulation uses a nonlinear plant representation, the complete simulation must be performed with a standard ODE solver. For state feedback with a ***nonlinear plant model,*** the user-defined function file called by the ODE solver would include the following algorithm and equations (for known state vector, $\underline{z}(t)$, at time t):

1. specify the set point value at time t, $r_d(t)$
2. extract the plant and estimated states (for convenience), where

   $$\underline{x} = \begin{bmatrix} z_1 & z_2 & \cdots & z_N \end{bmatrix}^T \qquad \text{and} \qquad \underline{\hat{x}} = \begin{bmatrix} z_{N+1} & z_{N+2} & \cdots & z_{2N} \end{bmatrix}^T$$
3. evaluate the output at time t, $y(t) = \underline{\underline{C}}\,\underline{x}(t)$
4. determine the manipulated input at time t, $u(t) = N_r r_d(t) - \underline{\underline{K}}_s\,\underline{\hat{x}}(t)$
5. evaluate the plant state derivatives at time t, $\frac{d}{dt}\underline{x}(t) = \underline{f}\left(\underline{x}(t), u(t), t\right)$
6. evaluate the observer state derivatives at time t, $\frac{d}{dt}\underline{\hat{x}}(t) = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}}(t) + \underline{\underline{B}}u(t) + \underline{\underline{L}}y(t)$
7. form derivative of full state vector at time t, $\frac{d}{dt}\underline{z}(t) = \left[\frac{d}{dt}\underline{x}^T(t) \quad \frac{d}{dt}\underline{\hat{x}}^T(t)\right]^T$

Finally, upon return from the ODE solver, one can re-compute the desired output function for all time by simply extracting the plant and estimated states (as above) and evaluating $y(t) = \underline{\underline{C}}\,\underline{x}(t)$.

State Feedback Assisted Classical Control (with present gains)

Various authors have tried to explain the logic behind state feedback control, either physically or mathematically (or both). One interesting way to view things is to consider state feedback control as a means of supplying a modified setpoint signal to a classically controlled plant. For example, if the performance of the system shown in Fig. 7.9 was not really satisfactory for the chosen K_c, one might consider manually altering the setpoint, $r_d(t)$, to give a better transient response. Alternatively, one might develop a control system, such as that shown in Fig. 7.11, to automatically supply the modified setpoint demand signal, $r_{dm}(t)$, for a given change in the actual demand signal, $r_d(t)$. Note that the output of the left sum block in Fig. 7.11 is the modified setpoint signal and that, after this point, the system looks like a classically controlled plant with unity feedback and a classical proportional gain, K_c. In this diagram, the goal of the state observer and the state feedback path is to provide a modified demand signal, $r_{dm}(t)$, and help the classically controlled system (shaded portion of block diagram) to perform more effectively. This configuration is sometimes referred to as ***State Feedback Assisted Classical Control*** (SFACC).

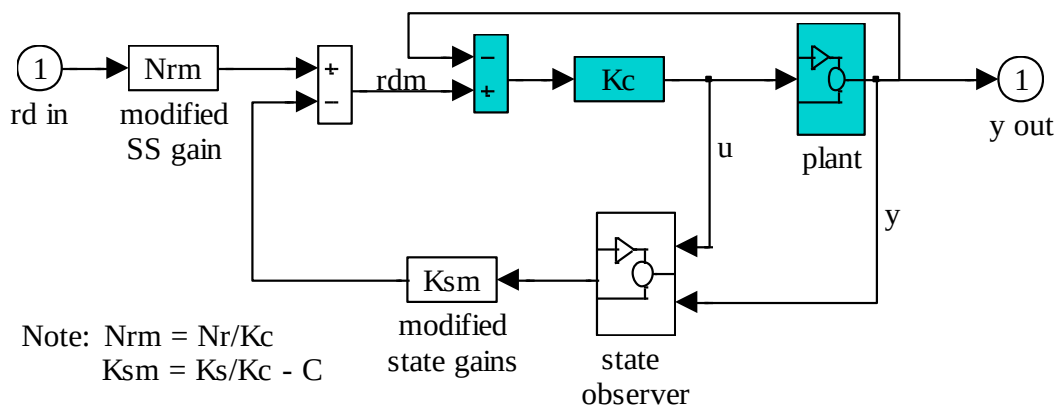


Fig. 7.11 State feedback control with an embedded classical control loop (SFACC scheme).

The control system shown in Fig. 7.11 is just the state-feedback control scheme discussed previously. This block diagram can be developed from the standard formulation shown in Fig. 7.10 by performing a sequence of block diagram manipulations. In particular, Figs. 7.12 - 7.14 summarize the steps involved in going from Fig. 7.10 to Fig. 7.11. This four-step sequence is outlined as follows:

1. Introduce a proportional gain block just prior to the plant input and cancel its effect in each of the incoming paths (as shown in Fig.7.12).

2. Introduce a classical negative feedback path from the system output, y, which is canceled by a positive feedback loop from the estimated output, $\hat{y}$ (as shown in Fig. 7.13).

3. Separate the summing junction as shown in Fig. 7.14 and identify the classically controlled plant (shaded portion of Fig. 7.14).

4. Finally, we can identify a modified state feedback gain,

$$\underline{\underline{K}}_{sm} = \underline{\underline{K}}_{s} / K_c - \underline{\underline{C}} \tag{7.40}$$

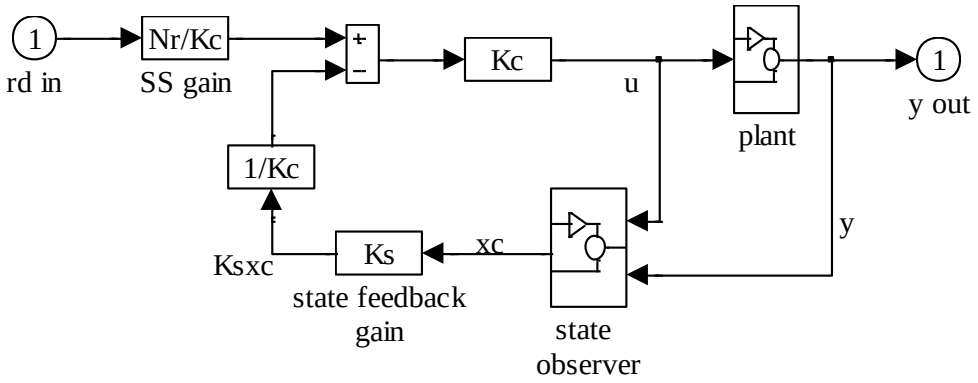


Fig. 7.12 Introduce classical controller gain into SFC scheme.

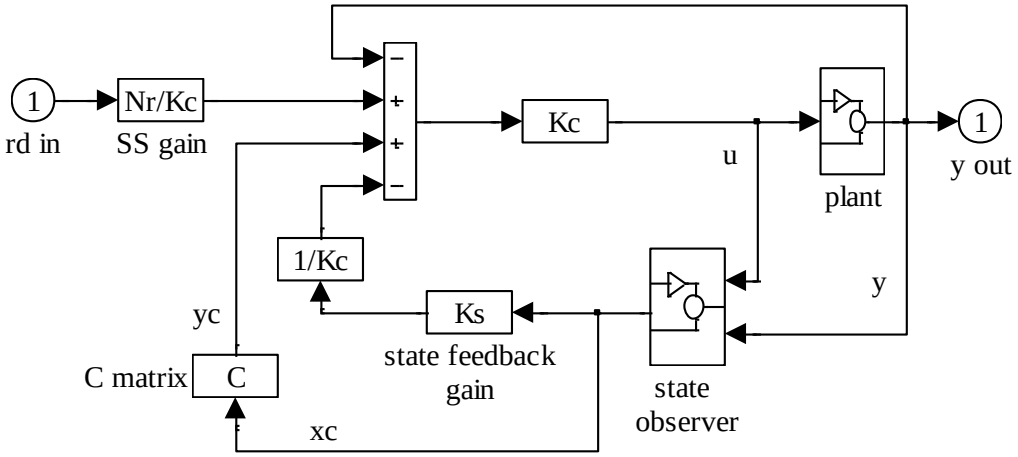


Fig. 7.13 Introduce classical negative feedback canceled by estimated output positive feedback.

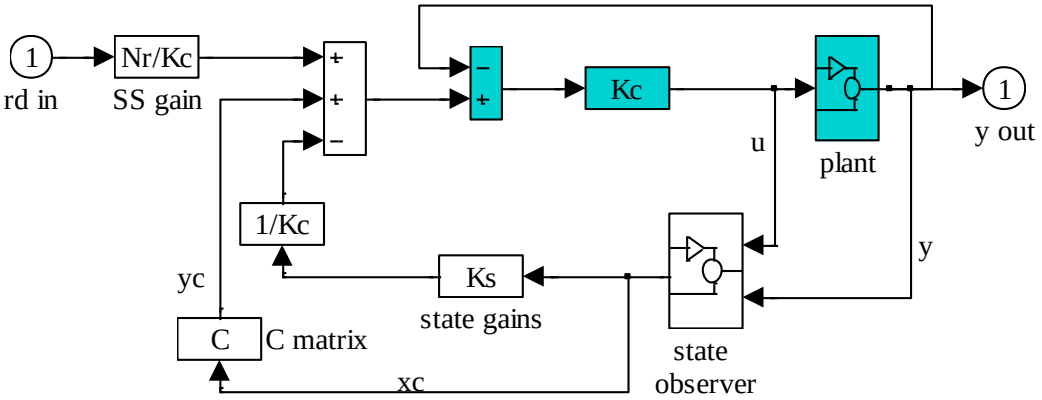


Fig. 7.14 Separate summing junction and identify classical proportional controller.

and a modified steady state normalization,

$$N_{rm} = N_r / K_c \tag{7.41}$$

and produce the block diagram shown in Fig. 7.11.

The block diagram in Fig. 7.11 for state feedback assisted classical control (SFACC) is functionally identical with the traditional state feedback control (SFC) representation given in Fig. 7.10. The only approximation made here was in Step 2, where we assumed that the estimated output canceled exactly with the actual measured output. If the state observer is properly designed to have a fast response time relative to the plant, this becomes an excellent approximation.

Thus, from the above discussion, we see that SFC and SFACC give the same transient performance. However, the SFACC algorithm offers a different perspective, and it may, in some cases, allow a better interpretation of how state feedback affects system performance. It is certainly a different viewpoint from the mathematical approach presented earlier.

Finally, it should be noted that the SFACC scheme might also be suitable for implementation of SFC into a real plant that already is outfitted with a classical control scheme. In this way, existing control schemes may be upgraded to use modern control strategies (if needed), without removing existing systems (however, additional software and/or hardware are often required). This also represents a smooth transition from classical to modern control, since one can simply interpret the modern state-feedback strategy as an enhanced classical control scheme (with automatic adjustment of the modified setpoint to achieve improved plant performance).

To simulate the state feedback assisted classical control scheme, we identify the control law as

$$u = K_c(r_{dm} - y) \tag{7.42}$$

However, the modified setpoint is given in terms of the estimated state, or

$$r_{dm} = N_{rm} r_d - \underline{\underline{K}}_{sm} \underline{\hat{x}} \tag{7.43}$$

Substitution of eqn. (7.43) into eqn. (7.42) gives the final form of the control law, or

$$u = K_c N_{rm} r_d - K_c \underline{\underline{K}}_{sm} \underline{\hat{x}} - K_c y \tag{7.44}$$

As for the SFC scheme, the closed loop dynamics for this SFACC system with a linearized plant model contains both the actual plant states, $\underline{x}(t)$, and the estimated plant states, $\underline{\hat{x}}(t)$. Thus, we again have 2N unknowns. With a ***linear plant model***, the fully coupled dynamics for this SFACC system can be written as:

Plant Dynamics

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}u = \underline{\underline{A}}\underline{x} + \underline{\underline{B}}\left(K_c N_{rm} r_d - K_c \underline{\underline{K}}_{sm} \underline{\hat{x}} - K_c y\right)$$

or

$$\frac{d}{dt}\underline{x} = \left(\underline{\underline{A}} - K_c \underline{\underline{B}}\underline{\underline{C}}\right)\underline{x} - K_c \underline{\underline{B}}\underline{\underline{K}}_{sm} \underline{\hat{x}} + K_c \underline{\underline{B}} N_{rm} r_d \tag{7.45}$$

Observer Dynamics

$$\frac{d}{dt}\underline{\hat{x}} = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}u + \underline{\underline{L}}y = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}\left(K_c N_{rm} r_d - K_c \underline{\underline{K}}_{sm}\underline{\hat{x}} - K_c y\right) + \underline{\underline{L}}\,\underline{\underline{C}}\,\underline{x}$$

or
$$\frac{d}{dt}\underline{\hat{x}} = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}} - K_c \underline{\underline{B}}\,\underline{\underline{K}}_{sm}\right)\underline{\hat{x}} + \left(\underline{\underline{L}}\,\underline{\underline{C}} - K_c \underline{\underline{B}}\,\underline{\underline{C}}\right)\underline{x} + K_c \underline{\underline{B}} N_{rm} r_d \tag{7.46}$$

This coupled set of matrix equations can be written in standard state form as

$$\frac{d}{dt}\underline{z} = \underline{\underline{\bar{A}}}\,\underline{z} + \underline{\underline{\bar{B}}}\,r_d \qquad \text{and} \qquad y = \underline{\underline{\bar{C}}}\,\underline{z} \tag{7.47}$$

with
$$\underline{z} = \begin{bmatrix} \underline{x} \\ \underline{\hat{x}} \end{bmatrix} \qquad \underline{\underline{\bar{A}}} = \begin{bmatrix} \underline{\underline{\bar{A}}}_{11} & \underline{\underline{\bar{A}}}_{12} \\ \underline{\underline{\bar{A}}}_{21} & \underline{\underline{\bar{A}}}_{22} \end{bmatrix} \qquad \underline{\underline{\bar{B}}} = \begin{bmatrix} K_c \underline{\underline{B}} N_{rm} \\ K_c \underline{\underline{B}} N_{rm} \end{bmatrix} \qquad \underline{\underline{\bar{C}}} = \begin{bmatrix} \underline{\underline{C}} & \underline{\underline{0}} \end{bmatrix} \tag{7.48}$$

and
$$\underline{\underline{\bar{A}}}_{11} = \underline{\underline{A}} - K_c \underline{\underline{B}}\,\underline{\underline{C}} \qquad \underline{\underline{\bar{A}}}_{12} = -K_c \underline{\underline{B}}\,\underline{\underline{K}}_{sm} \tag{7.49}$$

$$\underline{\underline{\bar{A}}}_{21} = \underline{\underline{L}}\,\underline{\underline{C}} - K_c \underline{\underline{B}}\,\underline{\underline{C}} \qquad \underline{\underline{\bar{A}}}_{22} = \underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}} - K_c \underline{\underline{B}}\,\underline{\underline{K}}_{sm}$$

Once the above simulation is complete for a specified setpoint variation, $r_d(t)$, one can compute several other parameters of interest. For example, one can easily determine the following quantities:

1. The plant and estimated states can be extracted from the full state vector, where

$$\underline{x} = \begin{bmatrix} z_1 & z_2 & \cdots & z_N \end{bmatrix}^T \qquad \text{and} \qquad \underline{\hat{x}} = \begin{bmatrix} z_{N+1} & z_{N+2} & \cdots & z_{2N} \end{bmatrix}^T$$

2. One can also determine the error associated with the state estimator,

$$\underline{e} = \underline{x} - \underline{\hat{x}}$$

3. The modified demand signal can be determined from

$$r_{dm} = N_{rm} r_d - \underline{\underline{K}}_{sm}\underline{\hat{x}}$$

This is the signal that drives the classical control loop to give better overall behavior (modified setpoint).

4. One can also compute the manipulated input at each time point,

$$u = K_c(r_{dm} - y)$$

This is the actual input to the plant.

5. Etc.

The algorithm for the simulation of the SFACC system with a ***nonlinear plant model*** is almost identical to the SFC algorithm given previously. For completeness, we will repeat the sequence of steps here, which includes the slight modifications for the SFACC system. As above, the user-defined function file called by the ODE solver would include the following algorithm and equations (for known state vector, $\underline{z}(t)$, at time t):

1. specify the set point value, r_d
2. extract the plant and estimated states (for convenience), where

   $\underline{x} = \begin{bmatrix} z_1 & z_2 & \cdots & z_N \end{bmatrix}^T$ and $\underline{\hat{x}} = \begin{bmatrix} z_{N+1} & z_{N+2} & \cdots & z_{2N} \end{bmatrix}^T$
3. evaluate the output, $y = \underline{c}^T \underline{x}$
4. determine the modified setpoint, $r_{dm} = N_{rm} r_d - \underline{\underline{K}}_{sm} \underline{\hat{x}}$
5. determine the manipulated input, $u = K_c (r_{dm} - y)$
6. evaluate the plant state derivatives, $\frac{d}{dt}\underline{x} = \underline{f}\left(\underline{x}, u, t\right)$
7. evaluate the observer state derivatives, $\frac{d}{dt}\underline{\hat{x}} = \left(\underline{\underline{A}} - \underline{\underline{L}}\,\underline{\underline{C}}\right)\underline{\hat{x}} + \underline{\underline{B}}u + \underline{\underline{L}}y$
8. form derivative of full state vector, $\frac{d}{dt}\underline{z} = \begin{bmatrix} \frac{d}{dt}\underline{x}^T & \frac{d}{dt}\underline{\hat{x}}^T \end{bmatrix}^T$

Upon return from the ODE solver, since $\underline{z} = \begin{bmatrix} \underline{x}^T & \underline{\hat{x}}^T \end{bmatrix}^T$ is known, one can re-compute any quantity of interest to make it available for plotting and further analysis.

Note: In an actual application, the "plant" would be a physical device or process. The sensors in the plant monitor y(t) for the given u(t). Thus, the computational model embedded with the actual controller hardware only requires determination of the estimated states, $\underline{\hat{x}}(t)$, and the automatic control of the manipulated input, u(t), so that the output will follow the regulator setpoint, $r_d(t)$, as designed.

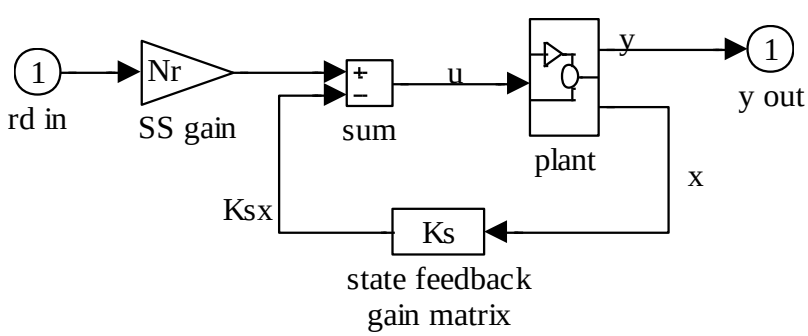


Fig. 7.15 State feedback control with SS gain block.

Adjusting the Reference Input Gain for Zero SS Error

In some of the above block diagrams, an input normalization block was included to help minimize the steady state error associated with the proportional controller (i.e. state feedback control). The value for this gain can be determined as follows (refer to the simplified block diagram in Fig. 7.15):

Linear Plant $\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}u$ and $y = \underline{\underline{C}}\,\underline{x}$

Control Law $u = N_r r_d - \underline{\underline{K}}_s \underline{x}$

Now, at steady state, we desire that y_{ss} be the same as the setpoint, r_{dss}, for any value of r_{dss}. Simply evaluating the state equation at steady state conditions gives

$$0 = \underline{\underline{A}}\,\underline{x}_{ss} + \underline{\underline{B}}u_{ss} = \underline{\underline{A}}\,\underline{x}_{ss} + \underline{\underline{B}}N_r r_{dss} - \underline{\underline{B}}\,\underline{\underline{K}}_s \underline{x}_{ss} = (\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s)\underline{x}_{ss} + \underline{\underline{B}}N_r r_{dss}$$

Solving this expression for the state vector at steady state gives

$$\underline{x}_{ss} = -(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s)^{-1}\underline{\underline{B}}N_r r_{dss}$$

and the steady state outputs becomes

$$y_{ss} = -\underline{\underline{C}}(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s)^{-1}\underline{\underline{B}}N_r r_{dss}$$

In this last expression, we simply substitute the desired result, $y_{ss} = r_{dss}$, and solve the resultant expression for N_r, giving

$$N_r = \frac{-1}{\underline{\underline{C}}(\underline{\underline{A}} - \underline{\underline{B}}\,\underline{\underline{K}}_s)^{-1}\underline{\underline{B}}} \tag{7.50}$$

This value of N_r should force the steady state error to zero.

A Detailed Example - The Inverted Pendulum

As a somewhat realistic example of the design and simulation of controlled systems we now focus on the development and analysis of an inverted pendulum mounted on a motor driven cart. A sketch of this system is shown in Fig. 7.16 (from Ogata's text, **Modern Control Engineering**). Realistically, this simple mechanical system is representative of a class of attitude control problems whose goal is to maintain the desired vertically oriented position at all times.

Our particular goal here is to illustrate some of the techniques introduced earlier in this section, with specific focus on the state feedback methodology. Since the inverted pendulum is a nonlinear system, we first develop the basic balance equations for the system, put these nonlinear equations into standard state form, and then generate a linearized model of the nonlinear "plant". The open loop plant is shown to be highly unstable. A simplistic approach to classical control of this system is attempted with the end result showing rather ineffective performance. To achieve better control and a more desirable closed loop response, state feedback control is implemented, with considerable improvement in the response of the system due to a setpoint change associated with the cart's position.

Several variants of this state-controlled system are illustrated in a series of Matlab simulations. The system with and without a state observer is compared and the use of a linear versus nonlinear plant model is highlighted. Finally, a wind disturbance input is added to the mathematical model, and the effect of this additional random force on system performance is addressed. This series of applications implements and illustrates many of the topics discussed in the early part of this section. The Matlab examples should give the reader a better understanding of the design and simulation methods discussed earlier, and they also provide the student with a set of functional tools that can easily be modified to fit other situations of interest.

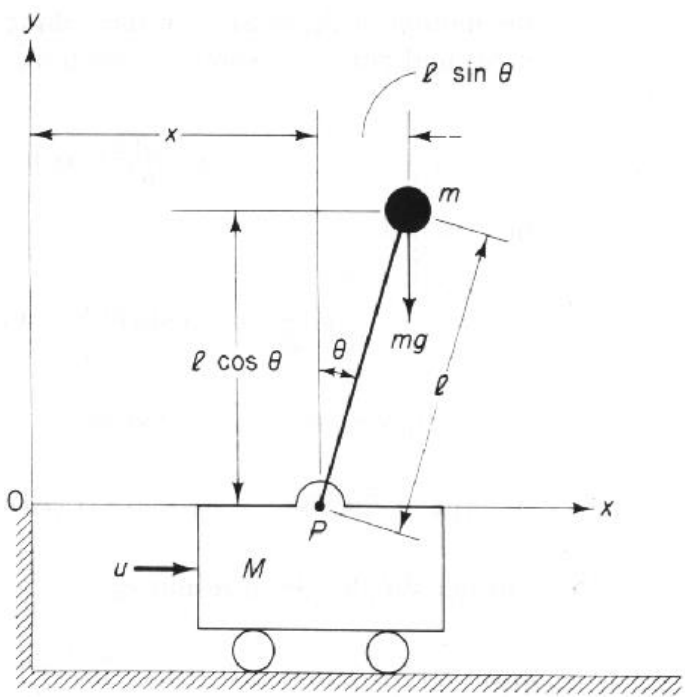


Fig. 7.16 Inverted pendulum on a cart (from Modern Control Engineering by Ogata).

Basic Equations (Inverted Pendulum)

Given an inverted pendulum mounted on a motor driven cart as shown in Fig. 7.16, the defining nonlinear equations can be derived as follows. First, we assume that the rod has negligible mass and that the cart mass and the point mass at the upper end of the inverted pendulum are denoted as M and m, respectively. There is an externally x-directed force on the cart, u(t), and a gravity force acts on the point mass at all times. The coordinate system chosen is defined in Fig. 7.16, where x(t) represents the cart position and θ(t) is the tilt angle referenced to the vertically upward direction.

A force balance in the x-direction gives that the mass times acceleration of the cart plus the mass times the x-directed acceleration of the point mass must equal the external force on the system. This can be written as

$$M\frac{d^2}{dt^2}x + m\frac{d^2}{dt^2}x_G = u \tag{7.51}$$

where the time-dependent center of gravity of the point mass is given by the coordinates, (x_G, y_G). For the point mass assumed here, the location of the center of gravity of the pendulum mass is simply

$$x_G = x + \ell\sin\theta \qquad \text{and} \qquad y_G = \ell\cos\theta \tag{7.52}$$

where ℓ is the pendulum rod length. Substitution of eqn. (7.52) into (7.51) gives

$$M\frac{d^2}{dt^2}x + m\frac{d^2}{dt^2}\left(x + \ell\sin\theta\right) = u$$

Noting the following definitions,

$$\frac{d}{dt}\sin\theta = (\cos\theta)\dot{\theta} \qquad \text{and} \qquad \frac{d^2}{dt^2}\sin\theta = -(\sin\theta)\dot{\theta}^2 + (\cos\theta)\ddot{\theta} \tag{7.53}$$

$$\frac{d}{dt}\cos\theta = -(\sin\theta)\dot{\theta} \qquad \text{and} \qquad \frac{d^2}{dt^2}\cos\theta = -(\cos\theta)\dot{\theta}^2 - (\sin\theta)\ddot{\theta} \tag{7.54}$$

we have

$$(M+m)\ddot{x} - m\ell\sin\theta\,\dot{\theta}^2 + m\ell\cos\theta\,\ddot{\theta} = u \tag{7.55}$$

In a similar manner, we perform a torque balance on the system, where torque is the product of the perpendicular component of the force and the distance to the pivot point (lever arm length, ℓ). In this case, the torque on the mass due to the acceleration force is balanced by the torque on the mass due to the gravity force. The force components are identified in Fig. 7.17 and the resultant balance can be written as

$$(F_x\cos\theta)\ell - (F_y\sin\theta)\ell = (mg\sin\theta)\ell \tag{7.56}$$

where the force components, F_x and F_y, are determined from eqns. (7.52) - (7.54) and written as

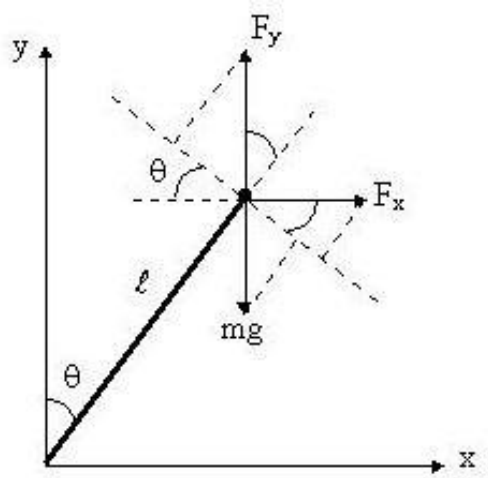


Fig. 7.17 Vector diagram for force components in torque balance.

$$F_x = m\frac{d^2}{dt^2}x_G = m\left[\ddot{x} - \ell\sin\theta\dot{\theta}^2 + \ell\cos\theta\ddot{\theta}\right]$$

$$F_y = m\frac{d^2}{dt^2}y_G = -m\left[\ell\cos\theta\dot{\theta}^2 + \ell\sin\theta\ddot{\theta}\right]$$

Substituting these expressions into eqn. (7.56) and noting that ℓ cancels, gives

$$m\ddot{x}\cos\theta - m\ell\sin\theta\cos\theta\dot{\theta}^2 + m\ell\cos^2\theta\ddot{\theta} + m\ell\sin\theta\cos\theta\dot{\theta}^2 + m\ell\sin^2\theta\ddot{\theta} = mg\sin\theta$$

or $$m\ddot{x}\cos\theta + m\ell\ddot{\theta} = mg\sin\theta \tag{7.57}$$

Therefore, the defining equations for this system are given by eqns. (7.55) and (7.57). These equations definitely represent a nonlinear system which is relatively complicated from a mathematical viewpoint. However, since the goal of this particular system is to keep the inverted pendulum upright around $\theta = 0$, one might consider linearization about the upright equilibrium point. We will do this later and actually compare the linear and nonlinear dynamics of the system; but first, we need to put the nonlinear equations into standard state space form.

Nonlinear State Equations (Inverted Pendulum)

To numerically simulate the nonlinear model for the inverted pendulum we need to put it into standard state form,

$$\frac{d}{dt}\underline{z} = \underline{f}(\underline{z}, u, t) \tag{7.58}$$

To put eqns. (7.55) and (7.57) into this form, let's first manipulate the equations algebraically to only have a single second derivative term in each equation. From eqn. (7.57), we have

$$m\ell\ddot{\theta} = mg\sin\theta - m\ddot{x}\cos\theta$$

and putting this into eqn. (7.55) gives

$$(M+m)\ddot{x} - m\ell\sin\theta\,\dot{\theta}^2 + mg\cos\theta\sin\theta - m\ddot{x}\cos^2\theta = u$$

or
$$\left(M+m-m\cos^2\theta\right)\ddot{x} = u + m\ell\sin\theta\,\dot{\theta}^2 - mg\cos\theta\sin\theta \tag{7.59}$$

Similarly, from eqn. (7.57) we have

$$\ddot{x} = \frac{g\sin\theta - \ell\ddot{\theta}}{\cos\theta}$$

and putting this into eqn. (7.55) gives

$$\frac{(M+m)\left(g\sin\theta - \ell\ddot{\theta}\right)}{\cos\theta} - m\ell\sin\theta\,\dot{\theta}^2 + m\ell\cos\theta\,\ddot{\theta} = u$$

or
$$(M+m)\left(g\sin\theta - \ell\ddot{\theta}\right) - m\ell\cos\theta\sin\theta\,\dot{\theta}^2 + m\ell\cos^2\theta\,\ddot{\theta} = u\cos\theta$$

and
$$\left(m\ell\cos^2\theta - (M+m)\ell\right)\ddot{\theta} = u\cos\theta - (M+m)g\sin\theta + m\ell\cos\theta\sin\theta\,\dot{\theta}^2 \tag{7.60}$$

Finally, dividing by the lead coefficients of eqns. (7.59) and (7.60) gives

$$\ddot{x} = \frac{u + m\ell(\sin\theta)\dot{\theta}^2 - mg\cos\theta\sin\theta}{M+m-m\cos^2\theta} \tag{7.61}$$

$$\ddot{\theta} = \frac{u\cos\theta - (M+m)g\sin\theta + m\ell(\cos\theta\sin\theta)\dot{\theta}^2}{m\ell\cos^2\theta - (M+m)\ell} \tag{7.62}$$

Now, to put these equations into state form, we make the following substitutions

$$z_1 = \theta \qquad z_2 = \dot{\theta} = \dot{z}_1 \qquad z_3 = x \qquad z_4 = \dot{x} = \dot{z}_3 \tag{7.63}$$

and write the final state space equations for the inverted pendulum as

$$\frac{d}{dt}\underline{z} = \frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} z_2 \\ \dfrac{u\cos z_1 - (M+m)g\sin z_1 + m\ell\left(\cos z_1\sin z_1\right)z_2^2}{m\ell\cos^2 z_1 - (M+m)\ell} \\ z_4 \\ \dfrac{u + m\ell\left(\sin z_1\right)z_2^2 - mg\cos z_1\sin z_1}{M+m-m\cos^2 z_1} \end{bmatrix} \tag{7.64}$$

This expression is now in the desired form given in eqn. (7.58). If both the pendulum angle θ and the cart position x are of interest, we have

$$\underline{y} = \underline{\underline{C}}\underline{z} \qquad \text{or} \qquad \underline{y} = \begin{bmatrix} \theta \\ x \end{bmatrix} = \underline{\underline{C}}\underline{z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \tag{7.65}$$

Equations (7.64) and (7.65) give a complete state space representation of the nonlinear inverted pendulum. This is the system implemented in the subsequent Matlab examples.

Linear State Equations (Inverted Pendulum)

If we desire a linearized system around the upright stationary point, we simply "linearize" the nonlinear system given in eqn. (7.64) as discussed previously in Section III of these notes. Since the usual $\underline{\underline{A}}$ and $\underline{\underline{B}}$ matrices are zero for this case (i.e. everything is put into the nonlinear vector function, $\underline{f}(\underline{z}, u, t)$), the linearized form for the system becomes

$$\frac{d}{dt}\delta\underline{z} = \underline{\underline{J}}_z(\underline{z}_o, u_o)\delta\underline{z} + \underline{\underline{J}}_u(\underline{z}_o, u_o)\delta u \tag{7.66}$$

where the reference state is defined with the pendulum stationary and upright with no input force. Under these conditions, $\underline{z}_o = \underline{0}$ and $u_o = 0$.

Since the nonlinear vector function is rather complicated, let's determine the components of the Jacobian matrices systemically, term by term. The elements of the first column of $\underline{\underline{J}}_z(\underline{z}_o, u_o)$ are given by $\partial f_i / \partial z_1 \big|_{\underline{z}_o, u_o}$. However, since the first and third functions are not directly related to z_1, these contributions are identically zero. For the second term, we have

$$\frac{\partial f_2}{\partial z_1} = \frac{-u\sin z_1 - (M+m)g\cos z_1 + m\ell\left(-\sin^2 z_1 + \cos^2 z_1\right)z_2^2}{m\ell\cos^2 z_1 - (M+m)\ell}$$
$$- \frac{u\cos z_1 - (M+m)g\sin z_1 + m\ell\left(\cos z_1 \sin z_1\right)z_2^2}{\left[m\ell\cos^2 z_1 - (M+m)\ell\right]^2}\left[2m\ell\cos z_1 \sin z_1\right]$$

and $$\left.\frac{\partial f_2}{\partial z_1}\right|_{\underline{z}_o, u_o} = \frac{(M+m)g}{M\ell}$$

Similarly, for the fourth term we have

$$\frac{\partial f_4}{\partial z_1} = \frac{m\ell\left(\cos z_1\right)z_2^2 - mg\left(-\sin^2 z_1 + \cos^2 z_1\right)}{M + m - m\cos^2 z_1}$$
$$- \frac{u + m\ell\left(\sin z_1\right)z_2^2 \ - mg\cos z_1 \sin z_1}{\left[M + m - m\cos^2 z_1\right]^2}\left[-2m\cos z_1 \sin z_1\right]$$

and $\left.\dfrac{\partial f_4}{\partial z_1}\right|_{\underline{z}_o,u_o} = \dfrac{-mg}{M}$

The second column of $\underline{\underline{J}}_z\left(\underline{z}_o,u_o\right)$ is given by $\partial f_i/\partial z_2\big|_{\underline{z}_o,u_o}$. Again, referring to eqn. (7.64), we see that the first element is unity and the third element is zero. For the second and fourth components we have

$$\frac{\partial f_2}{\partial z_2} = \frac{m\ell\left(\cos z_1 \sin z_1\right)2z_2}{m\ell\cos^2 z_1 - (M+m)\ell} \qquad \text{and} \qquad \left.\frac{\partial f_2}{\partial z_2}\right|_{\underline{z}_o,u_o} = 0$$

and $$\frac{\partial f_4}{\partial z_2} = \frac{m\ell\left(\sin z_1\right)2z_2}{M+m-m\cos^2 z_1} \qquad \text{and} \qquad \left.\frac{\partial f_4}{\partial z_2}\right|_{\underline{z}_o,u_o} = 0$$

For the third and fourth columns of $\underline{\underline{J}}_z\left(\underline{z}_o,u_o\right)$, the only nonzero term is $\partial f_3/\partial z_4\big|_{\underline{z}_o,u_o} = 1$. Thus, combining all these separate terms gives

$$\underline{\underline{J}}_z\left(\underline{z}_o,u_o\right) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{(M+m)g}{M\ell} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{mg}{M} & 0 & 0 & 0 \end{bmatrix} \tag{7.67}$$

For the derivative of the nonlinear terms with respect to u, we have

$$\underline{J}_u\left(\underline{z}_o,u_o\right) = \begin{bmatrix} \partial f_1/\partial u \\ \partial f_2/\partial u \\ \partial f_3/\partial u \\ \partial f_4/\partial u \end{bmatrix}_{\underline{z}_o,u_o} = \begin{bmatrix} 0 \\ \dfrac{\cos z_1}{m\ell\cos^2 z_1 - (M+m)\ell} \\ 0 \\ \dfrac{1}{M+m-m\cos^2 z_1} \end{bmatrix}_{\underline{z}_o,u_o} = \begin{bmatrix} 0 \\ \dfrac{-1}{M\ell} \\ 0 \\ \dfrac{1}{M} \end{bmatrix} \tag{7.68}$$

Finally, after all these manipulations we can write eqn. (7.66) explicitly as

$$\frac{d}{dt}\delta\underline{z} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{(M+m)g}{M\ell} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{mg}{M} & 0 & 0 & 0 \end{bmatrix}\delta\underline{z} + \begin{bmatrix} 0 \\ \dfrac{-1}{M\ell} \\ 0 \\ \dfrac{1}{M} \end{bmatrix}\delta u \tag{7.69}$$

which is in the standard LTI state form needed for implementation into Matlab (written in perturbation form). Equation (7.69) along with the response definition given in eqn. (7.65) represents the final linear model of the inverted pendulum.

Case 1 - Base Matlab Simulations

Matlab and Simulink were used to simulate both the open loop and closed loop response of the inverted pendulum. In this first demonstration, three aspects of the system are highlighted:

1. Comparison of the unstable behavior of the nonlinear and linearized models.

2. Demonstration that the linearized model can be obtained directly using Simulink's **linmod** function and an S-function representation of the nonlinear equations.

3. Implementation of several rather unsuccessful classical feedback control schemes.

These simulations are contained in a series of M-files (**invpn1.m**, **invpnnl1.m**, **invpnnl2.m**, etc.), many of which are listed in the Appendix to this section of notes. The **invpn1.m** file is the main program that controls the calculational flow and directs Matlab to perform the above analyses. The first comparison examines the step responses of the linearized system given by eqn. (7.69) and the nonlinear system defined by eqn. (7.64). The nonlinear equations are implemented in the **invpnnl1.m** function file which is called by Matlab's **ode45** routine to evaluate the state derivatives at any time point. Since the system is unstable, only a short 1 second simulation is performed. The results are summarized in Fig. 7.18. With a step change in the force, u(t), on the cart, we expect the cart's position to increase and the pendulum to fall in the counterclockwise direction. This is exactly as observed in Fig. 7.18, where we see, towards the end of the simulation, an increasing difference between the linear and nonlinear models. Again, this is expected, since the linear model was linearized around the state point $\underline{z}_o = \underline{0}$ and effectively assumes only small deviations from the reference linearization point.

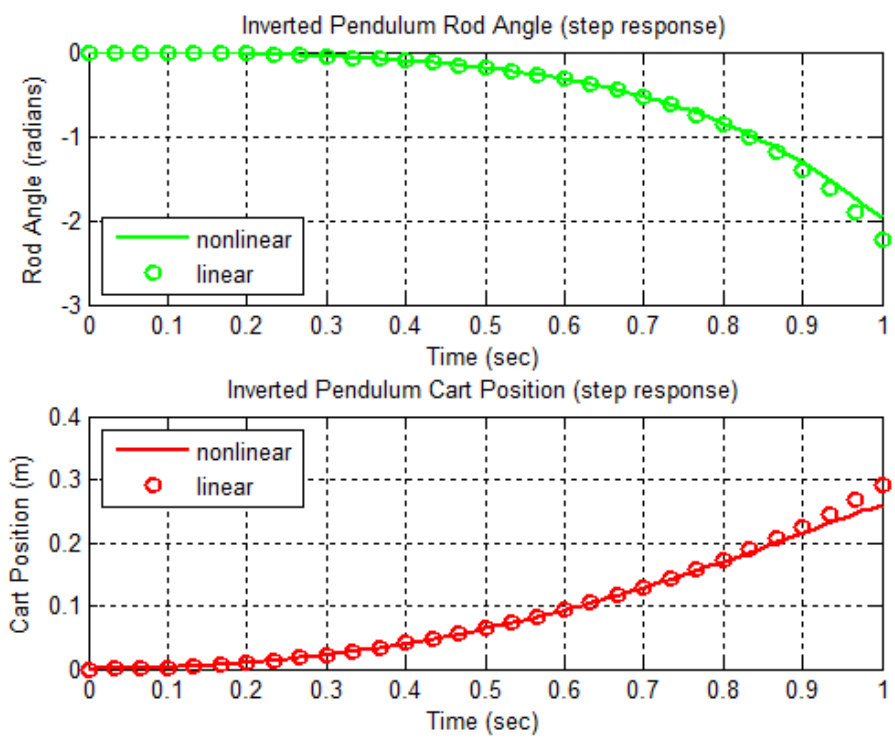


Fig. 7.18 Step response of open loop inverted pendulum.

The second task in **invpn1.m** focuses on the generation of the linear model directly within the Matlab/Simulink environment. Within a Matlab simulation, a Simulink graphical model is converted into an S-function that has all the simulation equations needed for computation of the state derivatives, the output function, and the initial state of the system. This S-function can then be used for simulation (with Matlab's **sim** command, for example) or a linearized state space model can be extracted (using **linmod**). In this example, the S-function was developed directly as **invpnnl2.m** (see the M-file listing in the Appendix). Although the syntax is a little different, the basic equations implemented within the S-function file, **invpnnl2.m**, are identical to those used in the **ode45** function file, **invpnnl1.m**. The reader is referred to the Matlab and Simulink documentation and help files for more information on the use of S-functions.

In any case, once an S-function is available, the numerical integration and linearization routines available with Simulink can be used as desired. In this demo, **sim** was used to simulate the step response of the nonlinear model and **linmod** was used to extract a linearized model around the $\underline{z}_o = \underline{0}$ and $u_o = 0$ point. Comparison of the dynamic responses and the state matrices show that the use of the S-function gives identical results with the standard Matlab analyses performed previously. Thus, the use of the S-function format represents an alternative to the traditional ODE solver used in Matlab. This new capability also offers additional flexibility -- especially with respect to the numerical evaluation of a linearized model.

In this example, the linearized model was originally obtained analytically by determining the desired Jacobian matrices. However, this was quite tedious, and the analytical process might even become completely impractical for some really complicated systems. Now that Simulink's **linmod** function has been demonstrated, one might consider a numerical approach for developing the linearized state equations in such cases.

The unstable open loop response of the inverted pendulum dictates the need for a robust control system to stabilize the pendulum and to improve the system response to setpoint changes and to disturbances in the system. Figures 7.19 - 7.21 outline a first attempt at designing a classical control system for the inverted pendulum. The problem here is that the open loop system has double roots at the origin in addition to an unstable root well into the right side of the complex plane. Additionally, there are two quantities of interest, the rod position and the cart position, and there is only a single input force on the cart.

A classical control scheme can take on many forms. Three relatively simple-minded approaches are identified in Figs. 7.19 - 7.21. The three schemes feed back the rod position, cart position, and both rod and cart position, respectively, in an attempt to control the system. For the first two schemes, a range of gains was examined, but the system could not be stabilized (see the **invpn1.out** output file in the Appendix). With both major and minor feedback loops as shown in Fig. 7.21, a trial and error sensitivity study on the gains, k_1 and k_2, finally gave a combination that stabilized the system. However, the performance of the system with this control scheme is far from satisfactory. For example, with k_1 = -50 and k_2 = -2, the behavior of the system for a step change in the cart position is illustrated in Fig. 7.22. Although the linearized closed loop system has a bounded response, it is only marginally stable, and it continuously oscillates around the new setpoint.

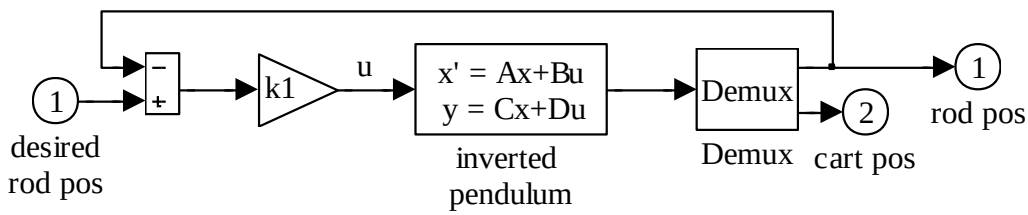


Fig. 7.19 Classical control of the rod position.

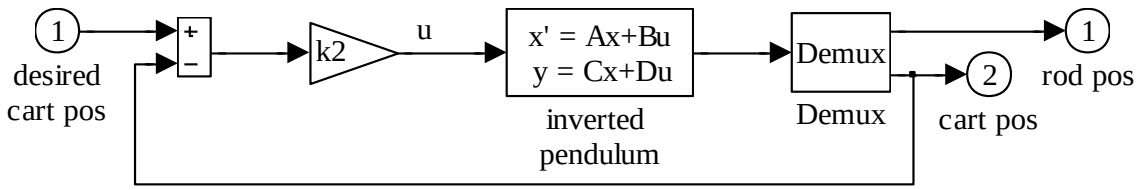


Fig. 7.20 Classical control of the cart position.

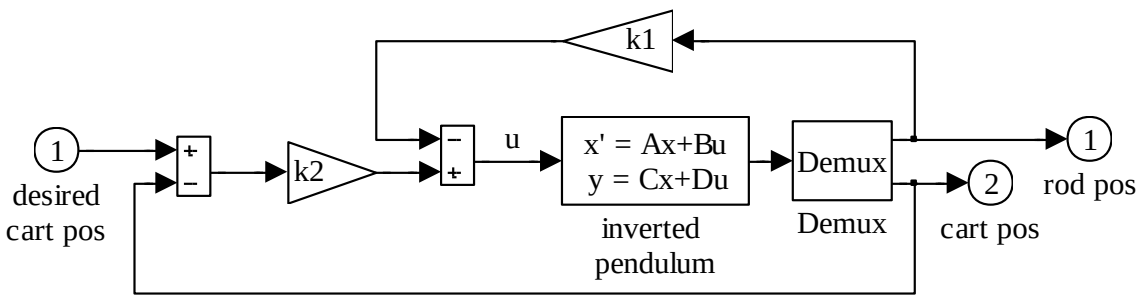


Fig. 7.21 Classical control of both cart position and rod position.

Case 2 - Matlab Simulations with State Feedback

The poor performance of the classical control schemes illustrated above could be improved with a modified controller design. However, instead of pursuing the trial and error classical design methodology, we decided to implement a state feedback control scheme. The development here follows closely the procedure used previously in Examples 7.1 and 7.2 and in the Matlab implementation of these cases (see **sfsotest1.m**, for example). Several somewhat redundant steps are performed in **invpn2.m** to illustrate various aspects of the state feedback controlled inverted pendulum system. In particular, we compare the linearized system performance with and without a state observer and also implement the state feedback controller and full observer with the nonlinear plant model. An optional frequency domain analysis is also performed if desired.

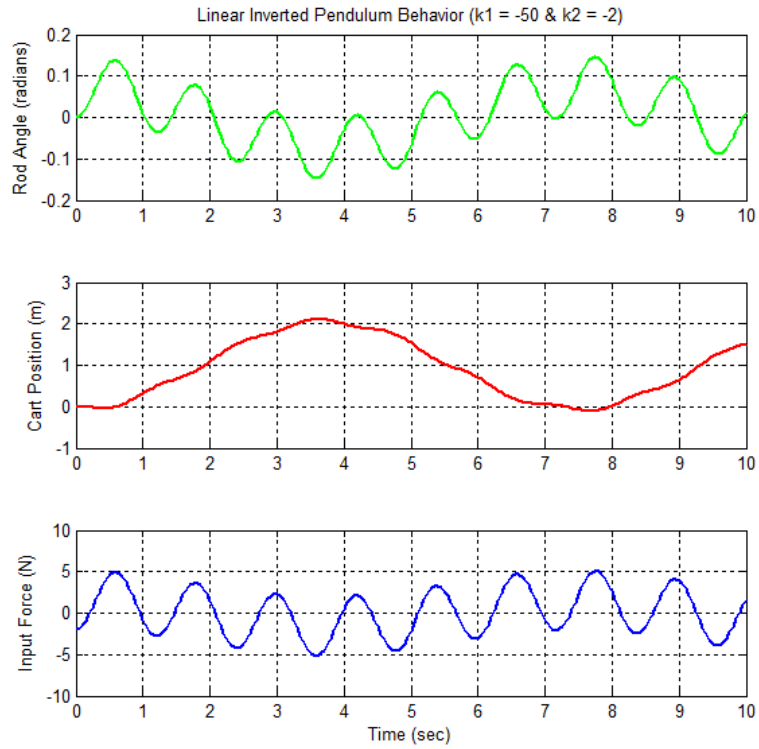


Fig. 7.22 Marginally stable behavior of the classically controlled inverted pendulum.

The first few steps in **invpn2.m** set up the basic design parameters for the inverted pendulum and the linearized model, determine the state feedback gains needed to achieve closed loop poles at $-1.5 \pm 3j$, -5, and -6, and finally simulate the closed loop response due to a step change in the cart's desired position.

The choice of the pole locations is somewhat arbitrary, but the dominant poles at $-1.5 \pm 3j$ should give reasonable transient behavior. From the ***Design Aids*** sheet (see pg. 8), we see that dominant poles at $-1.5 \pm 3j$ give the following dynamic quantities:

Pole locations $\mu_{1,2} = -1.5 \pm 3j = -\zeta\omega_n \pm j\omega_d$ where $\omega_d = \omega_n\sqrt{1-\zeta^2}$

$$|\mu| = \sqrt{(\zeta\omega_n)^2 + \omega_n^2(1-\zeta^2)} = \omega_n$$

Natural frequency $\omega_n = \sqrt{(1.5)^2 + (3.0)^2} = 3.354$

Damping ratio $\zeta = \dfrac{\sigma}{\omega_n} = \dfrac{1.5}{3.354} = 0.447$

Maximum overshoot, rise time, and settling time:

$$M_p \approx 20\% \qquad t_r = \frac{1.8}{\omega_n} = \frac{1.8}{3.354} \approx 0.54\,\text{sec} \qquad t_s = \frac{4.6}{\sigma} = \frac{4.6}{1.5} \approx 3.1\,\text{sec}$$

Thus, with a settling time of about 3 sec and a maximum overshoot of about 20%, a reasonable transient response should result.

With the desired closed loop poles specified, Matlab's **place** command is used to determine the required feedback gains. The closed loop system is then simulated, with the summary results given in Figs. 7.23 and 7.24. The cart's position is first highlighted in Fig. 7.23 and then all four states are displayed in Fig. 7.24. The system behavior is exactly as expected, where the initial negative displacement of the cart is required to get the rod moving in a clockwise direction. After a short time, the cart starts moving in the +x direction and eventually settles at x = 1 m after about 3 to 4 seconds. Notice that the maximum overshoot and settling time are exactly as designed. Note also that all the other states return to zero as x(t) approaches its equilibrium value of unity (see Fig. 7.24).

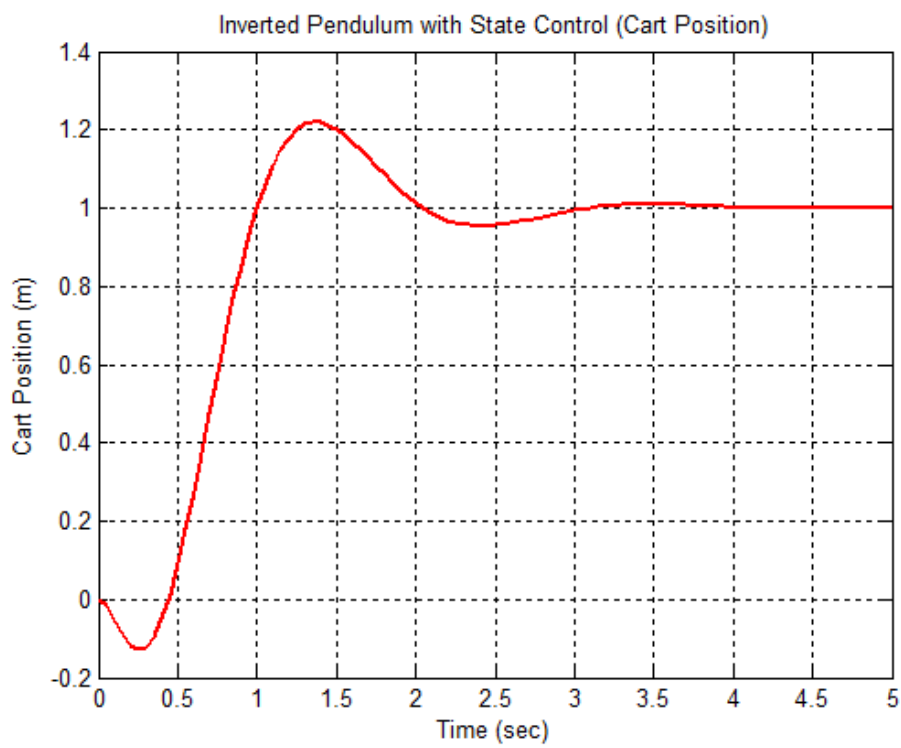


Fig. 7.23 Cart position with state feedback (step change in setpoint).

The next step in **invpn2.m** adds a state observer to the closed loop system, where the observer poles are designed to be a factor of two faster that the plant poles. With this specification, Matlab's **place** command is used again, this time to determine the necessary observer gains. The complete system with state feedback and a full state observer is then simulated with essentially identical results to those already displayed in Figs. 7.23 and 7.24. Figure 7.25 emphasizes the negligible differences that are observed between the plant (linear model) and observer states (linear model).

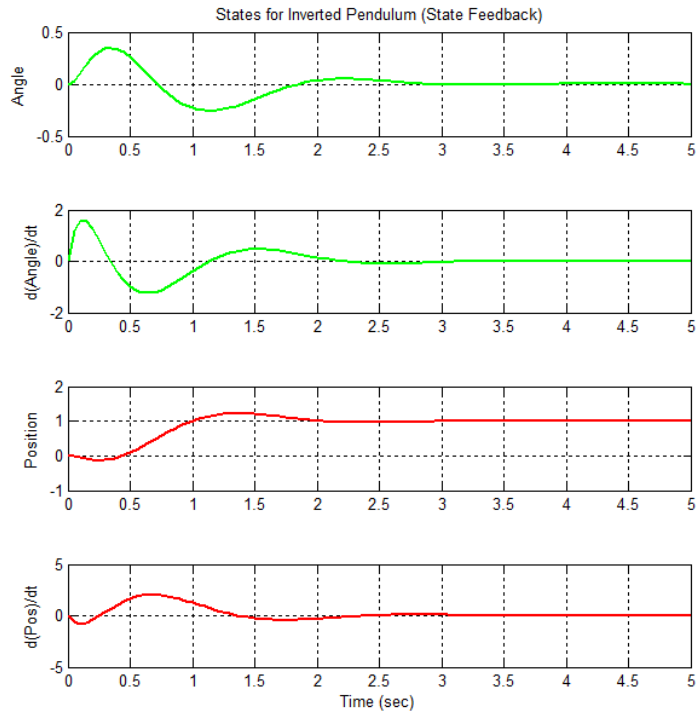


Fig. 7.24 State vector versus time for state feedback (step change in setpoint).

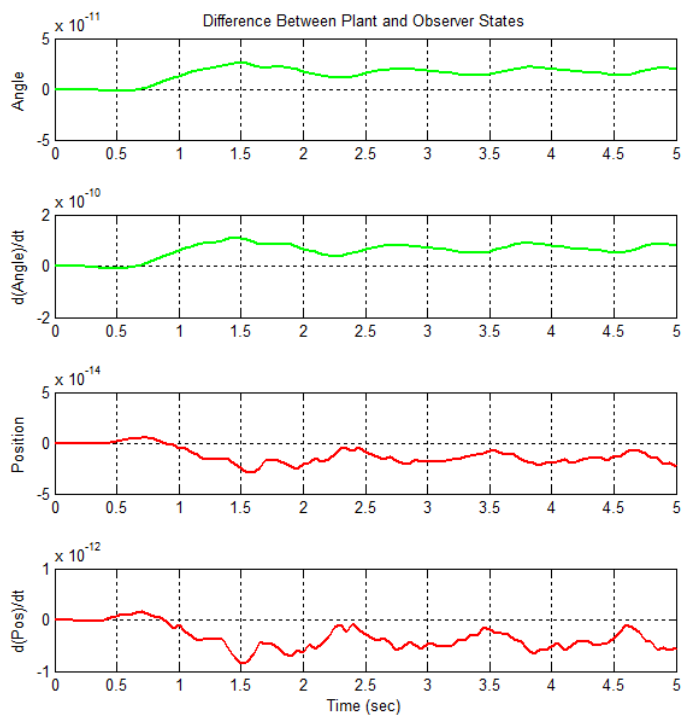


Fig. 7.25 Typical error vector for a linear plant model and linear observer model.

As a test of the robustness of the state feedback controller, **invpn2.m** also combines this same control system with the nonlinear plant model (which is contained in ODE function file **invpnnl3.m**). The essential results from the step response of this closed loop nonlinear system are highlighted in Figs. 7.26 - 7.28. These figures show an expanded view of the cart's position versus time, a composite representation of all the states, and a summary plot with the errors between the nonlinear plant and linear state estimator, respectively.

The approach to equilibrium for this case is a little more erratic, it takes a little longer to get there, and it has a somewhat larger overshoot than the linear plant simulation. However, in all, the step response of the nonlinear inverted pendulum is fairly well-behaved, and no redesign of the controller parameters is needed. Note, however, that since the plant is nonlinear and the observer is linear, we might expect to see a larger error vector than for the linear plant - linear observer case.

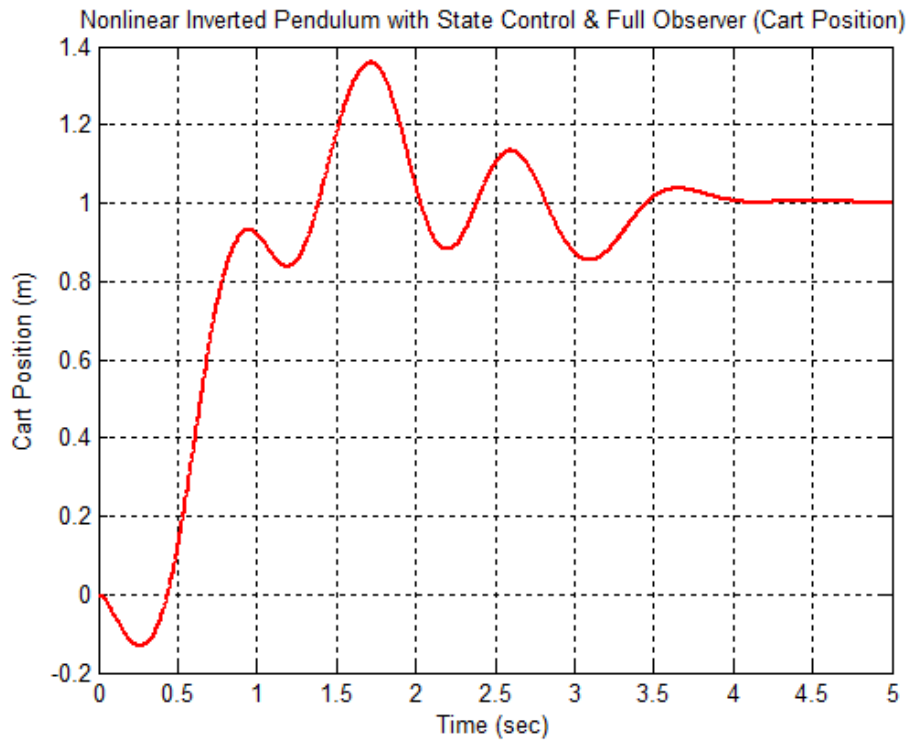


Fig. 7.26 Cart position versus time for a nonlinear plant and linear observer.

This expectation proved to be correct as clearly shown in Fig. 7.28. The errors here all tend to zero with time but, except for the cart position (which is used to drive the observer), the individual errors are on the same order of magnitude as the actual states (compare the magnitudes in Figs. 7.27 and 7.28). This mismatch in the predicted state versus the actual state is what causes the more erratic behavior seen in Fig. 2.26. Though definitely not negligible, the error vector is small enough to give reasonable overall performance of the closed loop nonlinear inverted pendulum.

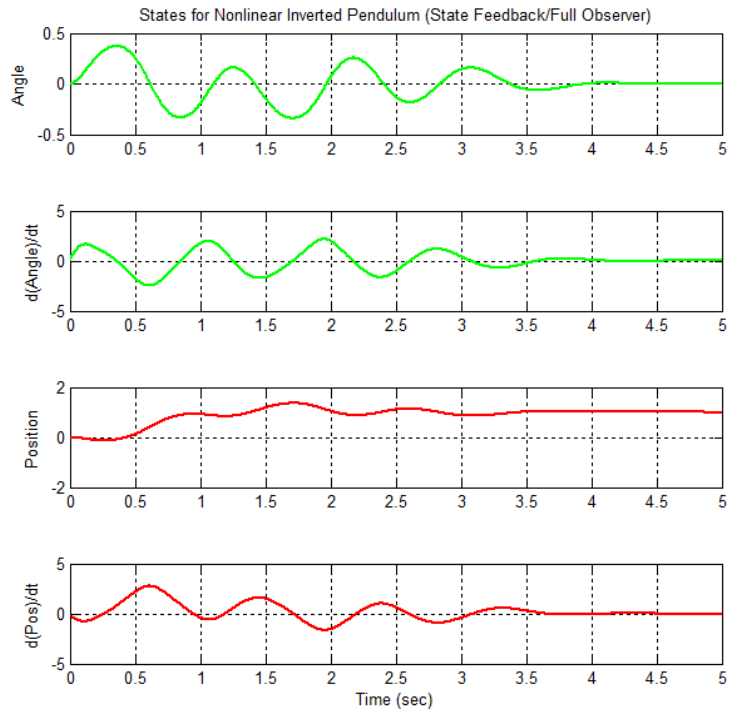


Fig. 7.27 State vector transient response for a nonlinear plant and linear observer.

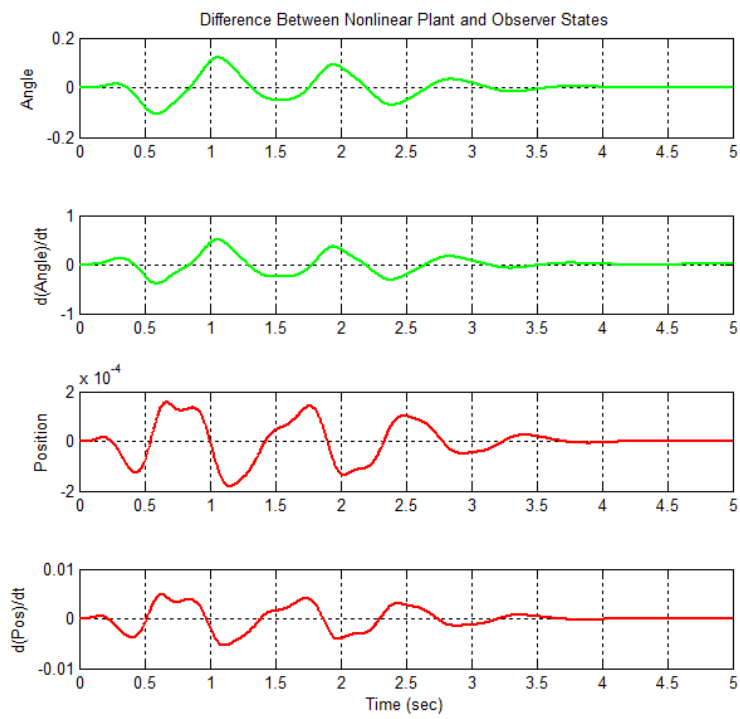


Fig. 7.28 Error vector versus time for a nonlinear plant and linear observer.

The reader is certainly encouraged to study this Matlab demonstration in some detail. The base **invpn2.m** program and a typical output file, **invpn2.out**, are listed in the Appendix. These files, along with **invpnnl3.m** for the nonlinear simulation with state feedback, represent a good illustration of how to use state feedback with a full observer for both linear and nonlinear plant models. The algorithms implemented here follow the procedures developed in the early parts of this section. Thus, they represent concrete examples of the theoretical equations developed previously. Simulations of this type may be quite useful in other areas of personal interest.

Case 3 - Matlab Simulations with Disturbance Input

As a final demonstration of the inverted pendulum, consider the inverted pendulum model with state feedback with an additional disturbance input due to wind effects. Let F_w represent the horizontal wind force on the pendulum point mass. With this additional force component, the original force balance given in eqn. (7.51) becomes

$$M\frac{d^2}{dt^2}x + m\frac{d^2}{dt^2}x_G = u + F_w \tag{7.70}$$

which can be manipulated as before to give

$$(M+m)\ddot{x} - m\ell\sin\theta\,\dot{\theta}^2 + m\ell\cos\theta\,\ddot{\theta} = u + F_w \tag{7.71}$$

Similarly, the torque in the clockwise direction caused by the horizontal wind disturbance is $\left(F_w\cos\theta\right)\ell$, and this term is added to the torque balance in eqn. (7.56) to give

$$(F_x\cos\theta)\ell - (F_y\sin\theta)\ell = (mg\sin\theta)\ell + \left(F_w\cos\theta\right)\ell \tag{7.72}$$

which again can be modified to give

$$m\ddot{x}\cos\theta + m\ell\ddot{\theta} = mg\sin\theta + F_w\cos\theta \tag{7.73}$$

Equations (7.71) and (7.73) are the defining equations for this new system which includes a horizontal wind disturbance. These two nonlinear equations can be manipulated as before and put into standard state form. Some of the steps in the process are:

1. Solve the torque balance expression for $m\ell\ddot{\theta}$ and put this into the force balance equation, giving

$$m\ell\ddot{\theta} = mg\sin\theta + F_w\cos\theta - m\ddot{x}\cos\theta$$

and $$\left[M + m - m\cos^2\theta\right]\ddot{x} = u + m\ell\sin\theta\,\dot{\theta}^2 - mg\sin\theta\cos\theta + F_w\sin^2\theta$$

where the last term comes from the relation

$$F_w\sin^2\theta = F_w - F_w\cos^2\theta$$

2. Solve the torque balance equation for $\ddot{x}$ and put this into the force balance, giving

$$\ddot{x} = \frac{mg\sin\theta + F_w\cos\theta - m\ell\ddot{\theta}}{m\cos\theta}$$

and $$(M+m)\frac{mg\sin\theta + F_w\cos\theta - m\ell\ddot{\theta}}{m\cos\theta} - m\ell\sin\theta\dot{\theta}^2 + m\ell\cos\theta\ddot{\theta} = u - F_w$$

Multiplying both sides by $\cos\theta$ gives

$$\left[m\ell\cos^2\theta - (M+m)\ell\right]\ddot{\theta} = u\cos\theta - (M+m)g\sin\theta + m\ell\cos\theta\sin\theta\dot{\theta}^2 - \left(\frac{M+m}{m}\right)F_w\cos\theta + F_w\cos\theta$$

3. Now, defining the elements of the state vector as before gives

$$\frac{d}{dt}\underline{z} = \frac{d}{dt}\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} z_2 \\ \dfrac{u\cos z_1 - (M+m)g\sin z_1 + m\ell\left(\cos z_1\sin z_1\right)z_2^2 - \dfrac{M}{m}F_w\cos z_1}{m\ell\cos^2 z_1 - (M+m)\ell} \\ z_4 \\ \dfrac{u + m\ell\left(\sin z_1\right)z_2^2 - mg\cos z_1\sin z_1 + F_w\sin^2 z_1}{M + m - m\cos^2 z_1} \end{bmatrix} \quad (7.74)$$

Notice that this expression is very similar to that given in eqn. (7.64). An additional term containing F_w is included in both the force and torque balances.

A linearized model can also be developed as before. When the resultant Jacobian matrices are evaluated at the reference point $\underline{z}_o = \underline{0}$ and $\underline{u}_o = 0$, we get

$$\frac{d}{dt}\delta\underline{z} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \left(\dfrac{M+m}{M\ell}\right)g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\dfrac{Mg}{M} & 0 & 0 & 0 \end{bmatrix}\delta\underline{z} + \begin{bmatrix} 0 \\ -\dfrac{1}{M\ell} \\ 0 \\ \dfrac{1}{M} \end{bmatrix}\delta u + \begin{bmatrix} 0 \\ -\dfrac{1}{m\ell} \\ 0 \\ 0 \end{bmatrix}\delta F_w \quad (7.75)$$

This is the open loop linearized model for the inverted pendulum with a cart force, $\delta u(t)$, and a horizontal wind disturbance, $\delta F_w(t)$. The two inputs have been separated for convenience. In this format the LTI system can be written as

$$\frac{d}{dt}\delta\underline{z} = \underline{\underline{A}}\delta\underline{z} + \underline{\underline{B}}_1\delta u + \underline{\underline{B}}_2\delta F_w \quad (7.76)$$

The **invpn3.m** file simulates the closed loop behavior associated with the linearized plant model defined by eqn. (7.76) with a variety of disturbance inputs. First, the wind force is assumed to be zero and a state feedback control system (without a state estimator) is designed to give the same closed loop response as observed for the **invpn2.m** simulations. Note that this system is an SISO LTI system, so the standard test for controllability and the procedure for determining the

feedback gains can be applied. During this development in **invpn3.m**, the B1matrix is simply the matrix (4×1 column vector in this case) that operates on $\delta u(t)$ as given in eqn. (7.76).

Once the state controller has been designed, we now move into a simulation mode. Here we define another B matrix (called BB in **invpn3.m**) which is the matrix containing both $\underline{\underline{B}}_1$ and $\underline{\underline{B}}_2$ from eqns. (7.75) and (7.76). Note also that the size of the null D matrix is also redefined for consistency.

The open and closed loop equations are summarized as follows:

Open Loop Plant: $$\frac{d}{dt}\delta\underline{z} = \underline{\underline{A}}\delta\underline{z} + \underline{\underline{B}}_1\delta u + \underline{\underline{B}}_2\delta F_w \tag{7.76}$$

Control Law for State Feedback: $$\delta u = N_r r_d - \underline{\underline{K}}_s\delta\underline{z} \tag{7.77}$$

Closed Loop System: $$\frac{d}{dt}\delta\underline{z} = \left(\underline{\underline{A}} - \underline{\underline{B}}_1\underline{\underline{K}}_s\right)\delta\underline{z} + \begin{bmatrix}\underline{\underline{B}}_1 & \underline{\underline{B}}_2\end{bmatrix}\begin{bmatrix}N_r r_d \\ \delta F_w\end{bmatrix} \tag{7.78}$$

A series of four simulations is performed in **invpn3.m** using eqn. (7.78). The first case represents a step change in the cart position setpoint r_d with the wind force set to zero. The expectation here is that we should get identical results relative to the **invpn2.m** simulations. This is indeed the case as seen in Fig. 7.29. Note here that we also plot the disturbance input, $\delta F_w(t)$, and the controlled input, $\delta u(t)$, to the plant. In this case, $\delta F_w(t) = v(t) = 0$ and the controlled input is the force required to give the desired setpoint change, yet keep the pendulum upright.

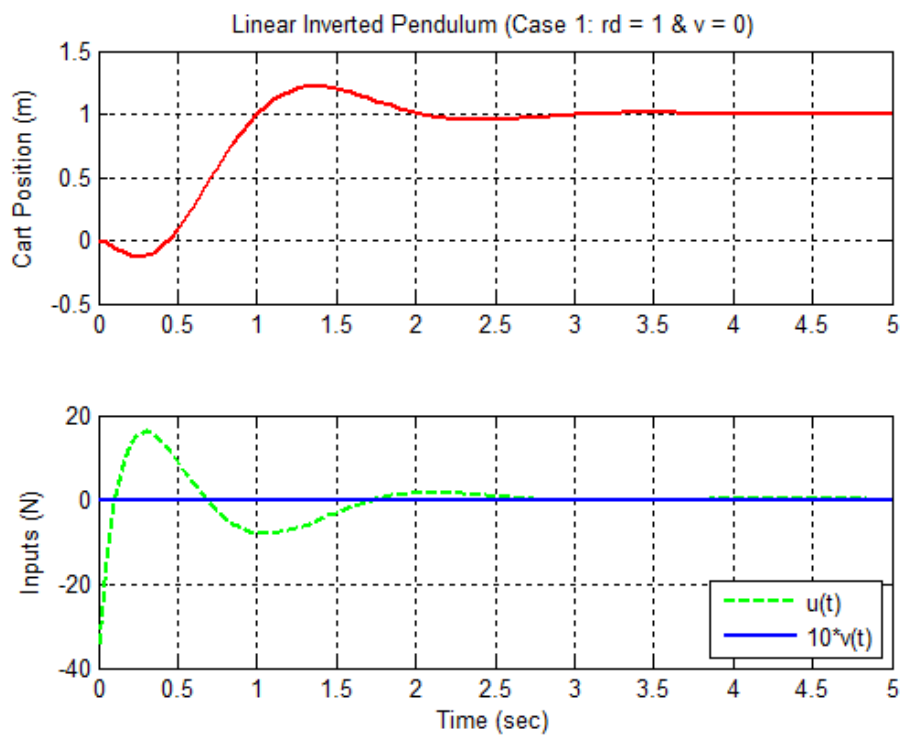


Fig. 7.29 Cart response and inputs for step change in setpoint position (no wind).

The second simulation models a step change in the disturbance input of 0.2 N with no change in the setpoint (i.e. $r_d = 0$). Figure 7.30 shows the results from this simulation. Here we see that an offset in the cart position of about 0.65 m occurs due to the step force of the wind. This is because a nonzero input force is required to keep the pendulum upright and, in the process, the cart is moved to the right by some finite distance. Recall that the setpoint normalization gain was derived to give zero steady state error due to a cart position setpoint change, not a disturbance input. Thus, although the dynamics due to wind effects is well behaved, a constant wind force does lead to a finite steady state error in the cart's position. Note that, at steady state, the constant wind force of +0.2 N is just offset by a -0.2 N cart force (notice that the wind disturbance v(t) is increased by a factor of 10 in all the plots for consistency so that it can be plotted and observed on the same scale as the controlled input force u(t)).

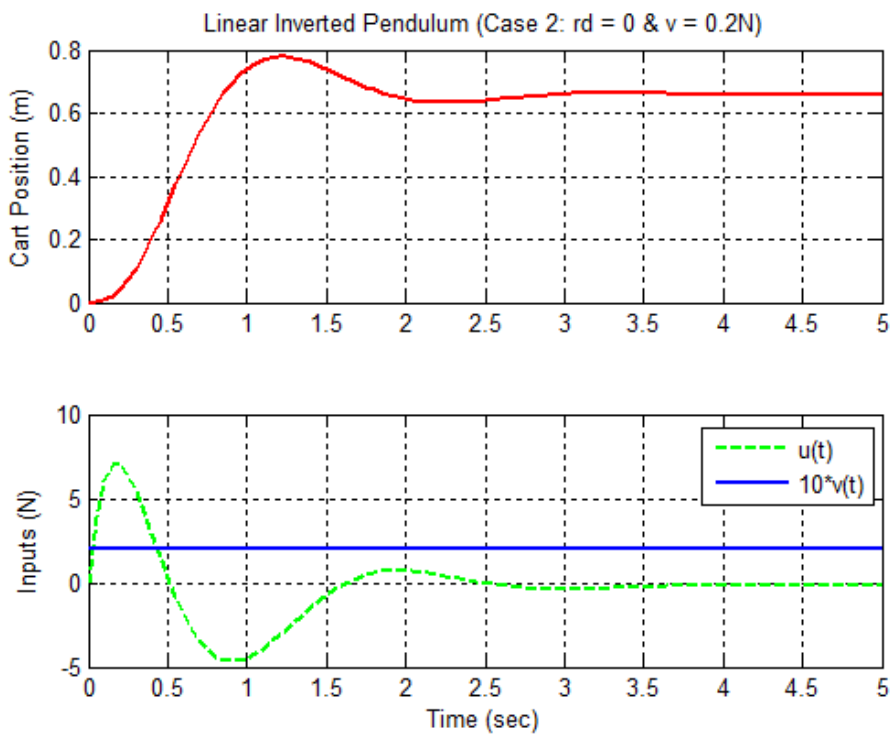


Fig. 7.30 Cart response and inputs for step change in wind force (no setpoint change).

Simulation Case #3 simply combines the inputs from Cases #1 and #2. Since the system is linear, we expect the behavior of this case to be the sum of the previous two runs. This is exactly what happened as shown in Fig. 7.31. The new steady state value still has an offset of about 0.65 m from the desired value of unity. In addition, the cart position, controlled input, and disturbance input are exactly the sum of the previous two cases -- no real surprises were observed.

The final simulation looks at a step change in the cart position as well as a random noise disturbance due to wind effects. The disturbance input is generated by selecting a random number between zero and one at each time point and scaling this to give a random disturbance input between ± 0.2 N. The random noise sequence has a mean value near zero.

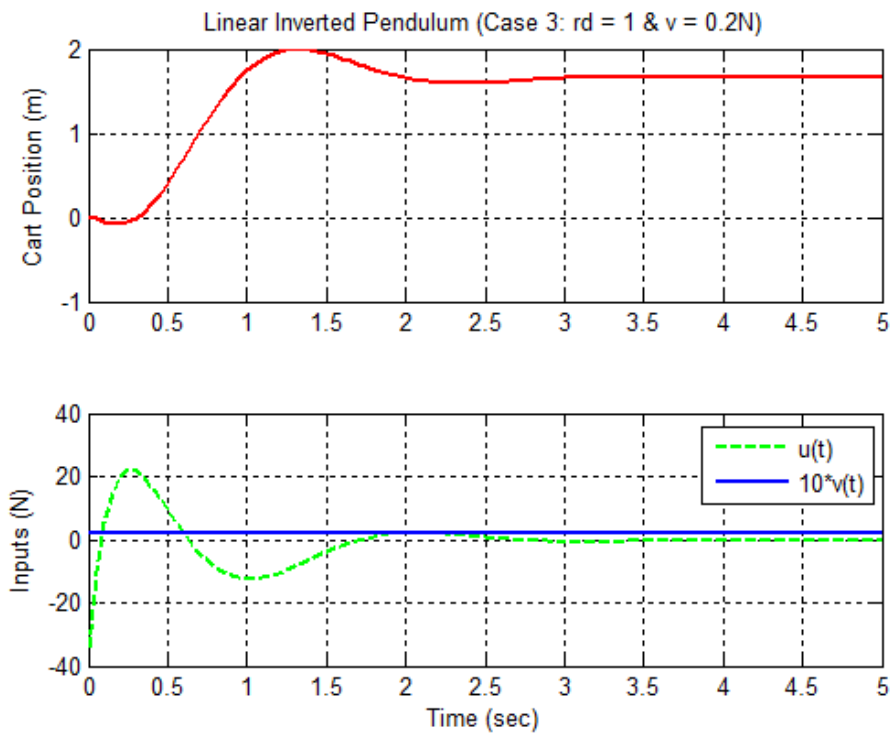


Fig. 7.31 Cart response and inputs for step change in setpoint and wind force.

Results from the Case #4 simulation are highlighted in Fig. 7.32. Here we see the typical cart movement due to the setpoint change but, this time, there is a small variation in position riding on the general movement towards the new equilibrium value. This modulation is due to the random wind force trying to topple the pendulum. However, the state feedback controller does a good job of keeping the pendulum upright and the cart close to the desired setpoint condition. In this simulation case, the average steady state error is near zero because the average wind force is also near zero. The controlled input has a high frequency random change that corresponds to and counteracts the effect of the random change in wind direction and magnitude. Again, it appears that the state controller, as designed, is sufficient to handle small variations in wind effects. If the magnitude of the wind disturbance is increased significantly, then the pendulum and cart overall performance is degraded.

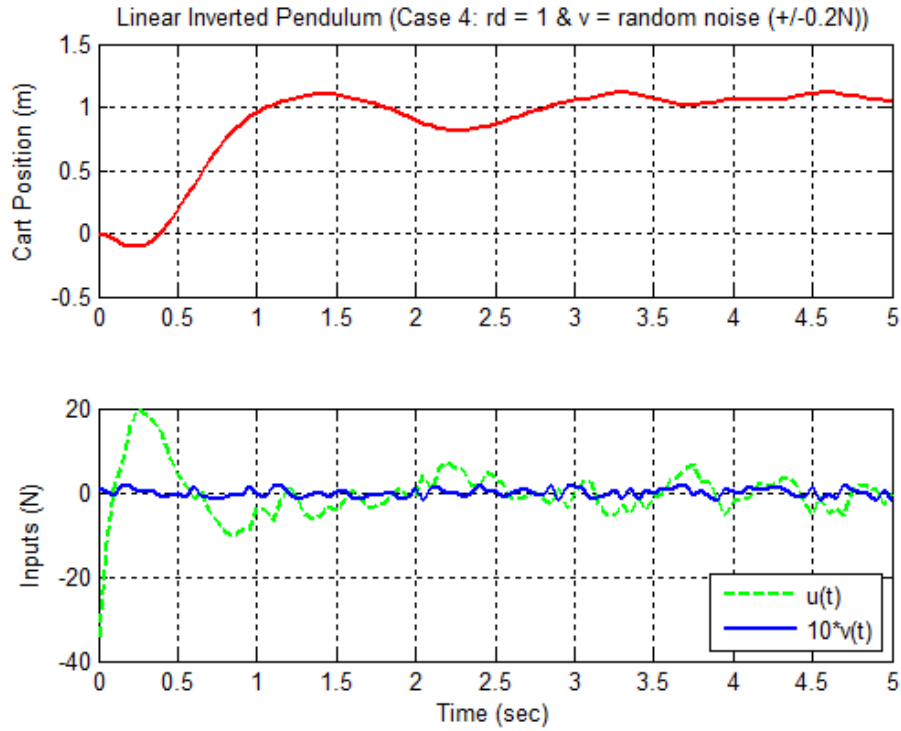


Fig. 7.32 Cart response and inputs for step change in setpoint and random wind force.

Summary

Well, this set of inverted pendulum examples completes the formal lecture notes for this section on the Introduction to the Design and Simulation of Controlled Systems and for the entire System Dynamics course -- any time left in the semester will be devoted to a variety of special topics. We have come a long way covering a lot of material, and I hope you have gained some real knowledge and experience in this area. Certainly, there is a lot more that could be done if time permitted, but you should now have a pretty good introduction into this subject -- so you are well-qualified for further self-study or other specialized courses in Dynamic Systems or Control Theory, as desired. Good Luck…

Appendix

This appendix contains listings of the Matlab programs used as part of the Inverted Pendulum Case Study. Note that the Simulink files that contain the block diagrams for part of the Case 1 studies are not included here. However, these ***.slx** files are certainly needed if one desires to run the **invpn1** case in Matlab (all the needed files are included within the **inverted_pendulum.zip** archive). The m-files and sample output files listed here are contained in several tables as follows:

Table #	File Names	File Description
7.3	**invpn1.m**	Main simulation file for the Case 1 inverted pendulum example.
7.4	**invpnnl1.m**	Function file used by **invpn1** to simulate the nonlinear inverted pendulum (ODE function file).
7.5	**invpnnl2.m**	S-function file used within **invpn1** and **invpnsl.slx** to simulate the nonlinear inverted pendulum and to extract the linearized model (used by Simulink's **sim** and **linmod** commands).
---	**invpnsl.slx**	Simulink file with simple block diagram containing only a single S-function block between a single input port and output port (called within **invpn1.m** and uses **invpnnl2.m**).
---	**invpnsl1.slx** **invpnsl2.slx** **invpnsl3.slx**	Simulink files containing the block diagrams for the classical control cases shown Figs. 7.19 - 7.21. These are used within the **invpn1.m** file.
7.6	**invpn1.out**	Sample output from a typical **invpn1** run.
7.7	**invpn2.m**	Main simulation file for the Case 2 inverted pendulum example.
7.8	**invpnnl3.m**	Function file used by **invpn2** to simulate the nonlinear inverted pendulum with state feedback (ODE function file).
7.9	**invpn2.out**	Sample output from a typical **invpn2** run.
7.10	**invpn3.m**	Main simulation file for the Case 3 inverted pendulum example.

Table 7.3 Listing of the invpn1.m file.

```
%
%   INVPN1.M      Inverted Pendulum Demonstration #1
%            (basic model simulations & classical control test)
%
%   This file simulates the inverted pendulum under several scenarios.  The
%   nonlinear and linear models are compared.  The Simulink linmod function is
%   used to generate the linear model and this is compared to the analytically
%   determined linear model.  A simple classical control scheme is attempted.
%   The choice of a controller is done by trial and error.
%
%   This file is broken into three sections, as follows:
%     Part I.  Setup base data for the nonlinear & linear models and show that the
%              base plant is unstable.  Determine step response and compare the
%              linear and nonlinear models.
%    Part II.  Use a Simulink S-function to represent the nonlinear equations.
%              Use Simulink's SIM function to simulate nonlinear model and
%              Simulink's LINMOD command to extract the linear model around the
%              desired reference point.
%   Part III.  Try to find a simple classical control scheme that stabilizes this
%              system.  Nothing fancy here - just trial and error.  Then simulate
%              the closed loop system behavior for a step change in the cart's
%              position.
%
%   Related files:
%     invpnnl1.m   - contains basic nonlinear model for use in ODE45
%     invpnnl2.m   - contains nonlinear model written as S-function for Simulink
%     invpnsl.slx  - Simulink model containing link to S-function of nonlinear model
%     invpnsl1.slx - Simulink linear model with classical control of rod position
%     invpnsl2.slx - Simulink linear model with classical control of cart position
%     invpnsl3.slx - Simulink linear model with classical control of rod & cart position
%
%   Note: Additional M-files (INVPN2.M & INVPN3.M) look at this system with
%         state feedback and a random disturbance as input.
%
%   File prepared by J. R. White, UMass-Lowell (last update: March 2020).
%

      clear all,   close all,   nfig = 0;
      format compact
      disp(' ')
      disp(' *** Summary Data from INVPN1.M ***')
      disp(' ')
%
%   Part I.  Setup base data for the nonlinear & linear models and show that the
%            base plant is unstable.  Determine step response and compare the
%            linear and nonlinear models.
%
%   basic data
      M = 2.0;  m = 0.1;     % mass of cart and mass at end (Kg)
      len = 0.5;             % length of pendulum rod (m)
      g = 9.81;              % gravitational acceleration (m/s^2)
%
%  simulate a step change in input force u = 1 in nonlinear model
      u = 1;    to = 0;   tf = 1.0;
      zo = [0 0 0 0]';    tol = 1.0e-6;
      options = odeset('RelTol',tol);
      ftz = @(t,z) invpnnl1(t,z,u,M,m,g,len);
      [tnl1,znl1] = ode45(ftz,[to tf],zo,options);
%
%  create state space matrices for linear model (from analytical development)
      c1 = M*len;  c2 = m*len;  c3 = m*g;   c4 = (M+m)*g;
      A1 = [0  1  0  0;  c4/c1  0  0  0; 0  0  0  1; -c3/M  0  0  0];
      B1 = [0 -1/c1 0 1/M]';
      C1 = [1 0 0 0; 0  0  1  0];      D1 = [0 0]';
      disp('State Space Matrices for the Analytically Determined Linear Model')
      A1,  B1,  C1,  D1
%
%   compute eigenvalues of state matrix for analytical linear system
      disp('Eigenvalues of the "Analytical Linear Model"');     ev = eig(A1)
%
%   determine step response of the analytical linear system
      tl = linspace(to,tf,31);
      sysl1 = ss(A1,B1,C1,D1);   [yl1,tl,zl1] = step(sysl1,tl);
```

```
%
%   plot results from nonlinear & linear models (no feedback)
      nfig = nfig+1;   figure(nfig);
      subplot(2,1,1),plot(tnl1,znl1(:,1),'g-',tl,zl1(:,1),'go','LineWidth',2),grid
      title('Inverted Pendulum Rod Angle (step response)')
      xlabel('Time (sec)'),ylabel('Rod Angle (radians)')
      legend('nonlinear','linear','Location','SouthWest')
%
      subplot(2,1,2),plot(tnl1,znl1(:,3),'r-',tl,zl1(:,3),'ro','LineWidth',2),grid
      title('Inverted Pendulum Cart Position (step response)')
      xlabel('Time (sec)'),ylabel('Cart Position (m)')
      legend('nonlinear','linear','Location','NorthWest')
%
%   Part II.  Use a Simulink S-function to represent the nonlinear equations.
%             Use Simulink's SIM function to simulate nonlinear model and
%             Simulink's LINMOD command to extract the linear model around the
%             desired reference point.
%
%  simulate a step change in input force u = 1 in nonlinear model
      invpnsl      % shows Simulink model with link to the nonlinear S-function model
      ut = [to  u; tf  u];   % this sets the step input vector (SIM interpolates)
      options = simset('RelTol',tol);
      [tnl2,znl2,ynl2] = sim('invpnsl',[to tf],options,ut);
%
%  create state space matrices for linear model (from Simulink numerical development)
      [A2,B2,C2,D2] = linmod('invpnsl',zo,0);
      disp('State Space Matrices for the Numerically Determined Linear Model')
      A2,  B2,  C2,  D2
%
%   compute eigenvalues of state matrix for numerical linear system
      disp('Eigenvalues of the "Numerical Linear Model"');     ev = eig(A2)
%
%   determine step response of the numerical linear system
      tl = linspace(to,tf,31);
      sysl2 = ss(A2,B2,C2,D2);   [yl2,tl,zl2] = step(sysl2,tl);
%
%   plot results from nonlinear & linear models from S-function (no feedback)
      nfig = nfig+1;   figure(nfig);
      subplot(2,1,1),plot(tnl2,znl2(:,1),'g-',tl,zl2(:,1),'go','LineWidth',2),grid
      title('S-Fun Inverted Pendulum Rod Angle (step response)')
      xlabel('Time (sec)'),ylabel('Rod Angle (radians)')
      legend('nonlinear','linear','Location','SouthWest')
%
      subplot(2,1,2),plot(tnl2,znl2(:,3),'r-',tl,zl2(:,3),'ro','LineWidth',2),grid
      title('S-Fun Inverted Pendulum Cart Position (step response)')
      xlabel('Time (sec)'),ylabel('Cart Position (m)')
      legend('nonlinear','linear','Location','NorthWest')
%
%   Part III.  Try to find a simple classical control scheme that stabilizes this
%              system.  Nothing fancy here - just trial and error.  Then simulate
%              the closed loop system behavior for a step change in the cart's
%              position.
%
%   Note: MATLAB's root locus command is very specialized and can only treat
%         SISO systems.
%
%   Initial matrices for use in Simulink models
      A = A1;  B = B1;  C = C1;  D = D1;
%
%   Rod position feedback
      invpnsl1      % shows simulink model (rod position feedback)
      disp('Following data for ROD POSITION feedback:');
      K1 = [-500 -100 -10 -1 -0.1 0.1 1 10 100 500];
      for j = 1:length(K1)
        k1 = K1(j),  [A1,B1,C1,D1] = linmod('invpnsl1');  eig(A1)
      end
%
%   Cart position feedback
      invpnsl2      % shows simulink model (cart postion feedback)
      disp('Following data for CART POSITION feedback:');
      K2 = [-500 -100 -10 -1 -0.1 0.1 1 10 100 500];
      for j = 1:length(K2)
        k2 = K2(j),  [A2,B2,C2,D2] = linmod('invpnsl2');  eig(A2)
      end
%
%   now try adding two feedback loops (cart position outside and rod position inside)
```

```
      invpnsl3      % shows simulink model (both feedbacks)
      disp('Closed loop system with two feedback loops');
      k1 = -50,    k2 = -2      % default values of gains
      nfig = nfig+1;    cont = 'y';
      while cont == 'y'
        [A3,B3,C3,D3] = linmod('invpnsl3');      eig(A3)
%
%   step response of linear system with two feedback loops (cart goes from 0 -> 1)
        to = 0;   tf = 10;    tlf = linspace(to,tf,201);
        sysl3 = ss(A3,B3,C3,D3);   [ylf,tlf,zlf] = step(sysl3,tlf);
%
%   plot results from linear models (with two feedbacks)
        figure(nfig);
        subplot(3,1,1),plot(tlf,ylf(:,1),'g-','LineWidth',2),grid
        title(['Linear Inverted Pendulum Behavior (k1 = ', ...
           num2str(k1),' & k2 = ',num2str(k2),')'])
        ylabel('Rod Angle (radians)')
%
        subplot(3,1,2),plot(tlf,ylf(:,2),'r-','LineWidth',2),grid
        ylabel('Cart Position (m)')
%
        u = k2*(1-ylf(:,2)) - k1*ylf(:,1);
        subplot(3,1,3),plot(tlf,u,'b-','LineWidth',2),grid
        xlabel('Time (sec)'),ylabel('Input Force (N)')
%
        cont = input('Select different gains? (y/n): ', 's');
        if isempty(cont);   cont = 'n';    end
        if cont == 'y'
          disp('Input values for gains (k1 & k2)')
          k1 = input('k1 = ');     k2 = input('k2 = ');
        end
      end
%
%   now draw full size plots for last gains used
      nfig = nfig+1;    figure(nfig);
      plot(tlf,ylf(:,1),'g-','LineWidth',2),grid
      title(['Inverted Pendulum Rod Position (k1 = ', ...
              num2str(k1),' & k2 = ',num2str(k2),')'])
      xlabel('Time (sec)'),ylabel('Rod Angle (radians)')
%
      nfig = nfig+1;    figure(nfig);
      plot(tlf,ylf(:,2),'r-','LineWidth',2),grid
      title(['Inverted Pendulum Cart Position (k1 = ', ...
              num2str(k1),' & k2 = ',num2str(k2),')'])
      xlabel('Time (sec)'),ylabel('Cart Position (m)')
%
      nfig = nfig+1;    figure(nfig);
      plot(tlf,u,'b-','LineWidth',2),grid
      title(['Inverted Pendulum Input Force (k1 = ', ...
              num2str(k1),' & k2 = ',num2str(k2),')'])
      xlabel('Time (sec)'),ylabel('Input Force (N)')
%
      disp('End of simulation')
%
%   end of simulation
```

Table 7.4 Listing of the invpnnl1.m file.

```
%
%   INVPNNL1.M   Nonlinear model of inverted pendulum (ODE function file format)
%
%   See documentation of equations in Section VII of Dynamic Systems Notes
%
%   File prepared by J. R. White, UMass-Lowell (last update: March 2020).
%

      function zdot = invpnnl1(t,z,u,M,m,g,len)
      zdot = zeros(size(z));
      c1 = (M+m);  c2 = m*len;  c3 = m*g;  c4 = (M+m)*len;  c5 = (M+m)*g;
      zdot(1) = z(2);
      top2 = u*cos(z(1)) - c5*sin(z(1)) + c2*cos(z(1))*sin(z(1))*z(2)^2;
      zdot(2) = top2/(c2*cos(z(1))^2 - c4);
      zdot(3) = z(4);
      top4 = u + c2*sin(z(1))*z(2)^2 - c3*cos(z(1))*sin(z(1));
      zdot(4) = top4/(c1-m*cos(z(1))^2);
%
%   end of routine
```

Table 7.5 Listing of the invpnnl2.m file.

```
%
%   INVPNNL2.M   Nonlinear model of inverted pendulum (Simulink S-function format)
%
%   See documentation of equations in Section VII of Dynamic Systems Notes
%   The interpretation of the flag variable is obtained by typing: help sfuntmpl
%
%   File prepared by J. R. White, UMass-Lowell (last update: March 2020).
%

      function [zdot,zo] = invpnnl2(t,z,u,flag,M,m,g,len)
%
%   return parameter size and initial conditions
      if flag == 0,    zdot = [4 0 2 1 0 0];    zo = zeros(4,1);
%
%   return continuous state derivatives (as a column vector)
      elseif flag == 1
        c1 = (M+m);  c2 = m*len;  c3 = m*g;  c4 = (M+m)*len;  c5 = (M+m)*g;
        zdot(1) = z(2);
        top2 = u*cos(z(1)) - c5*sin(z(1)) + c2*cos(z(1))*sin(z(1))*z(2)^2;
        zdot(2) = top2/(c2*cos(z(1))^2 - c4);
        zdot(3) = z(4);
        top4 = u + c2*sin(z(1))*z(2)^2 - c3*cos(z(1))*sin(z(1));
        zdot(4) = top4/(c1-m*cos(z(1))^2);
        zdot = zdot';
%
%   return the output vector (rod and cart position)  (as a column vector)
      elseif flag == 3
        zdot(1) = z(1);   zdot(2) = z(3);   zdot = zdot';
%
%   other options not used
      else
        zdot = [];
      end
%
%   end of routine
```

Table 7.6 Listing of typical output from running the invpn1 program.

```
>> invpn1

 *** Summary Data from INVPN1.M ***

State Space Matrices for the Analytically Determined Linear Model
A1 =
         0    1.0000         0         0
   20.6010         0         0         0
         0         0         0    1.0000
   -0.4905         0         0         0
B1 =
         0
   -1.0000
         0
    0.5000
C1 =
     1     0     0     0
     0     0     1     0
D1 =
     0
     0
Eigenvalues of the "Analytical Linear Model"
ev =
         0
         0
    4.5388
   -4.5388
State Space Matrices for the Numerically Determined Linear Model
A2 =
         0    1.0000         0         0
   20.6010         0         0         0
         0         0         0    1.0000
   -0.4905         0         0         0
B2 =
         0
   -1.0000
         0
    0.5000
C2 =
    1.0000         0         0         0
         0         0    1.0000         0
D2 =
     0
     0
Eigenvalues of the "Numerical Linear Model"
ev =
         0
         0
    4.5388
   -4.5388
Following data for ROD POSITION feedback:
k1 =
  -500
ans =
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 +21.8952i
   0.0000 -21.8952i
k1 =
  -100
ans =
   0.0000 + 0.0000i
   0.0000 + 0.0000i
   0.0000 + 8.9106i
   0.0000 - 8.9106i
k1 =
   -10
ans =
         0
         0
    3.2559
   -3.2559
```

```
k1 =
    -1
ans =
         0
         0
    4.4273
   -4.4273
k1 =
   -0.1000
ans =
         0
         0
    4.5278
   -4.5278
k1 =
    0.1000
ans =
         0
         0
    4.5498
   -4.5498
k1 =
     1
ans =
         0
         0
    4.6477
   -4.6477
k1 =
    10
ans =
         0
         0
    5.5318
   -5.5318
k1 =
   100
ans =
         0
         0
   10.9818
  -10.9818
k1 =
   500
ans =
         0
         0
   22.8167
  -22.8167
Following data for CART POSITION feedback:
k2 =
  -500
ans =
  -15.8450
   -4.4200
    4.4200
   15.8450
k2 =
  -100
ans =
   -7.1821
   -4.3609
    4.3609
    7.1821
k2 =
   -10
ans =
   -4.5727
    4.5727
   -2.1660
    2.1660
k2 =
    -1
ans =
   -4.5415
    4.5415
```

```
   -0.6897
    0.6897
k2 =
   -0.1000
ans =
   -4.5391
    4.5391
   -0.2182
    0.2182
k2 =
    0.1000
ans =
  -4.5386 + 0.0000i
   4.5386 + 0.0000i
   0.0000 + 0.2182i
   0.0000 - 0.2182i
k2 =
     1
ans =
  -4.5363 + 0.0000i
   4.5363 + 0.0000i
  -0.0000 + 0.6905i
  -0.0000 - 0.6905i
k2 =
    10
ans =
   4.5175 + 0.0000i
  -4.5175 + 0.0000i
   0.0000 + 2.1925i
   0.0000 - 2.1925i
k2 =
   100
ans =
  -4.4609 + 0.0000i
   4.4609 + 0.0000i
  -0.0000 + 7.0213i
  -0.0000 - 7.0213i
k2 =
   500
ans =
  -4.4375 + 0.0000i
   4.4375 + 0.0000i
   0.0000 +15.7826i
   0.0000 -15.7826i
Closed loop system with two feedback loops
k1 =
   -50
k2 =
    -2
ans =
  -0.0000 + 5.2622i
  -0.0000 - 5.2622i
   0.0000 + 0.8418i
   0.0000 - 0.8418i
Select different gains? (y/n): n
End of simulation
>>
```

Table 7.7 Listing of the invpn2.m file.

```
%
%   INVPN2.M    Inverted Pendulum Demonstration #2
%
%   This file simulates the inverted pendulum (linear and nonlinear model) with
%   state feedback control.  File invpnnl3.m is used for the nonlinear plant model.
%
%   This file is broken into five sections, as follows:
%     Part I. Setup base data for the linear model and show that the base
%             plant is unstable.
%    Part II. Add state feedback control to the linear model to stabilize the system.
%             Simulate system behavior for a step change in the cart's position.
%   Part III. Add state feedback control and a full observer to stabilize the system.
%             Simulate system behavior for a step change in the cart's position.
%             This should give the same simulation as Part II.
%    Part IV. Same as Part III using nonlinear model for the plant and the
%             linear model for the state observer.
%     Part V. Compute and plot (ie. Bode plots) the transfer function for the
%             closed loop system.  Here we are interested in the dynamics of the
%             cart's position relative to a change in the set point (desired position).
%             Since the closed loop dynamics are identical, we will use the case
%             without the state estimator.
%
%   File prepared by J. R. White, UMass-Lowell (last update: March 2020).
%

      clear all,   close all,   nfig = 0;
      format compact
      disp(' ')
      disp(' *** Summary Data from INVPN2.M ')
      disp(' ')
%
%   Part I. Setup base data for the linear model and show that the base
%           plant is unstable.
%
%   basic data
      M = 2.0;  m = 0.1;     % mass of cart and mass at end (Kg)
      len = .5;              % length of pendulum rod (m)
      g = 9.81;              % gravitational acceleration (m/s^2)
%
%  create state space matrices for linear model (output cart position)
      c1 = M*len;  c2 = m*len;  c3 = m*g;   c4 = (M+m)*g;
      A = [0  1  0  0;  c4/c1  0  0  0; 0  0  0  1; -c3/M  0  0  0];
      B = [0 -1/c1 0 1/M]';
      C = [0  0  1  0];      D = [0];
      disp('State Space Matrices for the Linear Model')
      A,  B,  C,  D
%
%   compute eigenvalues of state matrix for linear system
      disp('Eigenvalues of the "Linear Model"');     ev = eig(A)
%
%   Part II. Add state feedback control to stabilize the system.  Simulate system
%            behavior for a step change in the cart's position.
%
%   check for full state controllability
      disp('Controllability Matrix for this system'),   CM = ctrb(A,B)
      disp('Rank of Controllability Matrix'),   rank(CM)
%
%   calculate state feedback gains for specified closed loop poles
      clp = [-1.5+3.0j  -1.5-3.0j  -5.0  -6.0];
      Ks = place(A,B,clp);
      disp('Desired closed loop poles for state feedback controller');   clp
      disp('State feedback gains needed to give desired poles');         Ks
      disp('Calculated eigenvalues of system with state feedback'); eig(A-B*Ks)
%
%   calculate Nr for zero SS error  (see derivation in notes)
      Nr = -1.0/(C*inv(A-B*Ks)*B);
      disp('Setpoint gain for zero SS error');   Nr
%
%   simulate linear plant + controller
      tto = 0;  ttf = 5;   t = linspace(tto,ttf,101);
      syscl1 = ss(A-B*Ks,B*Nr,C,D);   [y1,t,x1] = step(syscl1,t);
%
%   plot results from state feedback case
```

```
      nfig = nfig+1;      figure(nfig)
      plot(t,y1,'r-','LineWidth',2),grid
      xlabel('Time (sec)'),ylabel('Cart Position (m)')
      title('Inverted Pendulum with State Control (Cart Position)')
%
%   also plot state variables
      nfig = nfig+1;      figure(nfig)
      subplot(4,1,1),plot(t,x1(:,1),'g-','LineWidth',2),grid,ylabel('Angle')
      title('States for Inverted Pendulum (State Feedback)')
      subplot(4,1,2),plot(t,x1(:,2),'g-','LineWidth',2),grid,ylabel('d(Angle)/dt')
      subplot(4,1,3),plot(t,x1(:,3),'r-','LineWidth',2),grid,ylabel('Position')
      subplot(4,1,4),plot(t,x1(:,4),'r-','LineWidth',2),grid,ylabel('d(Pos)/dt')
      xlabel('Time (sec)')
%
%   Part III. Add state feedback control and a full observer to stabilize the system.
%             Simulate system behavior for a step change in the cart's position.
%             This should give the same simulation as Part II.
%
%   check for full state observability
      disp('Observability Matrix for this system'),   OM = obsv(A,C)
      disp('Rank of Observability Matrix'),   rank(OM)
%
%   calculate estimator gains for specified observer poles
      op = 2*clp;    % estimator dynamics is 2 times faster than closed loop poles
      L = place(A',C',op);   L = L';
      disp('Desired observer poles for state feedback controller');   op
      disp('Estimator gains needed to give desired poles');           L
      disp('Calculated eigenvalues of estimator system');      eig(A-L*C)

%   setup matrices for linear plant + controller model
      A11 = A;     A12 = -B*Ks;         B1 = B*Nr;
      A21 = L*C;   A22 = A-L*C-B*Ks;    B2 = B*Nr;
      AB = [A11 A12; A21 A22];   BB = [B1; B2];
      zz = 0;   CB = [C zz*C];
%
%   simulate linear plant + controller
      syscl2 = ss(AB,BB,CB,D);   [y2,t,x2] = step(syscl2,t);
%
%   separate plant and estimator states
      nn = max(size(A));
      xp2 = x2(:,1:nn);     xe2 = x2(:,nn+1:2*nn);
%
%   plot results from case with full observer
      nfig = nfig+1;      figure(nfig)
      plot(t,y2,'r-','LineWidth',2),grid
      xlabel('Time (sec)'),ylabel('Cart Position (m)')
      title(['Inverted Pendulum with State Control & Full Observer (Cart Position)'])
%
%   also plot state variables
      nfig = nfig+1;      figure(nfig)
      subplot(4,1,1),plot(t,xp2(:,1),'g-','LineWidth',2),grid,ylabel('Angle')
      title(['States for Inverted Pendulum (State Feedback/Full Observer)'])
      subplot(4,1,2),plot(t,xp2(:,2),'g-','LineWidth',2),grid,ylabel('d(Angle)/dt')
      subplot(4,1,3),plot(t,xp2(:,3),'r-','LineWidth',2),grid,ylabel('Position')
      subplot(4,1,4),plot(t,xp2(:,4),'r-','LineWidth',2),grid,ylabel('d(Pos)/dt')
      xlabel('Time (sec)')
%
%   also plot error for plant vs estimator
      nfig = nfig+1;      figure(nfig)
      subplot(4,1,1),plot(t,xp2(:,1)-xe2(:,1),'g-','LineWidth',2),grid,ylabel('Angle')
      title(['Difference Between Plant and Observer States'])
      subplot(4,1,2),plot(t,xp2(:,2)-xe2(:,2),'g-','LineWidth',2),grid,ylabel('d(Angle)/dt')
      subplot(4,1,3),plot(t,xp2(:,3)-xe2(:,3),'r-','LineWidth',2),grid,ylabel('Position')
      subplot(4,1,4),plot(t,xp2(:,4)-xe2(:,4),'r-','LineWidth',2),grid,ylabel('d(Pos)/dt')
      xlabel('Time (sec)')
%
%   Part IV. Same as Part III using nonlinear model for the plant and the
%            linear model for the state observer.
%
%   use state feedback & estimator gains from above linear simulations
%
%   simulate linear plant + controller
      rd = 1;   % unit step in cart position setpoint
      tt0 = 0;  ttf = 5;   Zo = zeros(8,1);   tol = 1.0e-6;
      options = odeset('RelTol',tol);
      ftz = @(t,Z) invpnnl3(t,Z,M,m,g,len,A,B,C,D,Ks,Nr,L,rd);
```

```
      [tt,Z] = ode45(ftz,[tto ttf],Zo,options);
%
%   separate plant and estimator states
      nn = max(size(A));
      zp =Z(:,1:nn);     ze = Z(:,nn+1:2*nn);
%
%   plot results from nonlinear case with full observer
      nfig = nfig+1;      figure(nfig)
      plot(tt,zp(:,3),'r-','LineWidth',2),grid
      xlabel('Time (sec)'),ylabel('Cart Position (m)')
      title('Nonlinear Inverted Pendulum with State Control & Full Observer (Cart Position)')
%
%   also plot state variables
      nfig = nfig+1;      figure(nfig)
      subplot(4,1,1),plot(tt,zp(:,1),'g-','LineWidth',2),grid,ylabel('Angle')
      title('States for Nonlinear Inverted Pendulum (State Feedback/Full Observer)')
      subplot(4,1,2),plot(tt,zp(:,2),'g-','LineWidth',2),grid,ylabel('d(Angle)/dt')
      subplot(4,1,3),plot(tt,zp(:,3),'r-','LineWidth',2),grid,ylabel('Position')
      subplot(4,1,4),plot(tt,zp(:,4),'r-','LineWidth',2),grid,ylabel('d(Pos)/dt')
      xlabel('Time (sec)')
%
%   also plot error for plant vs estimator
      nfig = nfig+1;      figure(nfig)
      subplot(4,1,1),plot(tt,zp(:,1)-ze(:,1),'g-','LineWidth',2),grid,ylabel('Angle')
      title(['Difference Between Nonlinear Plant and Observer States'])
      subplot(4,1,2),plot(tt,zp(:,2)-ze(:,2),'g-','LineWidth',2),grid,ylabel('d(Angle)/dt')
      subplot(4,1,3),plot(tt,zp(:,3)-ze(:,3),'r-','LineWidth',2),grid,ylabel('Position')
      subplot(4,1,4),plot(tt,zp(:,4)-ze(:,4),'r-','LineWidth',2),grid,ylabel('d(Pos)/dt')
      xlabel('Time (sec)')
%
%  Part V. Compute and plot (ie. Bode plots) the transfer function for the
%          closed loop system.  Here we are interested in the dynamics of the
%          cart's position relative to a change in the set point (desired position).
%          Since the closed loop dynamics are identical, we will use the case
%          without the state estimator.
%
%   evaluate freqency response
      freqp = input('Perform frequency domain analysis? (y/n) [n]:  ','s');
      if isempty(freqp);   freqp = 'n';    end
      if freqp == 'y'
%
%   create Bode plots
        nfig = nfig+1;   figure(nfig)
        w = logspace(-1,2,100);   bode(syscl1,w),  grid
        hd = findobj(gcf,'Color','blue'); nhd = length(hd);
        for i = 1:nhd; set(hd(i),'LineWidth',2); end
        title('G(s) = X(s)/R(s)  (Cart Position)')
%
%   special note
        disp('  ')
        disp('Note that this system is non minimum phase because of the zero')
        disp('in the right portion of the complex plane')
%
%   convert from state-space to transfer function form
        disp('  ')
        disp('G(s) for Cart Position (Transfer Function form)')
        [num1,den1] = tfdata(syscl1,'v')
        printsys(num1,den1)
%
%   convert from state-space to zero-pole-gain form
        disp('  ')
        disp('G(s) for Cart Position (Zero-Pole form)')
        [z1,p1,k1] = zpkdata(syscl1,'v')
        zpk(syscl1)
%
%   finally, compute the damping ratio and natural frequency for the
%   closed loop system and make a pole zero plot
        disp('  ')
        disp('Damping and natural frequencies for closed loop system'), damp(p1)
        nfig = nfig+1;   figure(nfig)
        pzmap(syscl1),sgrid
        hd = findobj(gcf,'Color','blue'); nhd = length(hd);
        for i = 1:nhd; set(hd(i),'LineWidth',2); end
        title('Pole-Zero Map for Closed Loop System')
      end     %  generate above plots if freqp = 'y'
%   end of simulation
```

Table 7.8 Listing of the invpnnl3.m file.

```
%
%   INVPNNL3.M   Nonlinear model of closed loop (state controlled) inverted pendulum
%
%   File prepared by J. R. White, UMass-Lowell (last update: March 2020)
%

      function Zdot = invpnnl3(t,Z,M,m,g,len,A,B,C,D,Ks,Nr,L,rd)
%
%   separate plant and estimator states
      nn = max(size(A));
      zp = Z(1:nn,1);     ze = Z(nn+1:2*nn,1);
%   now calc derivative of full state vector (plant states and estimator states)
      yp = C*zp;   u = Nr*rd - Ks*ze;
      c1 = (M+m);  c2 = m*len;  c3 = m*g;  c4 = (M+m)*len;  c5 = (M+m)*g;
%   plant states
      zpdot(1) = zp(2);
      top2 = u*cos(zp(1)) - c5*sin(zp(1)) + c2*cos(zp(1))*sin(zp(1))*zp(2)^2;
      zpdot(2) = top2/(c2*cos(zp(1))^2 - c4);
      zpdot(3) = zp(4);
      top4 = u + c2*sin(zp(1))*zp(2)^2 - c3*cos(zp(1))*sin(zp(1));
      zpdot(4) = top4/(c1-m*cos(zp(1))^2);
%   estimator states
      zedot = A*ze + L*(yp - C*ze) + B*u;
%   put full state vector back together
      Zdot = [zpdot'; zedot];
%
%  end of routine
```

Table 7.9 Listing of typical output from running the invpn2 program.

```
>> invpn2

 *** Summary Data from INVPN2.M

State Space Matrices for the Linear Model
A =
         0    1.0000         0         0
   20.6010         0         0         0
         0         0         0    1.0000
   -0.4905         0         0         0
B =
         0
   -1.0000
         0
    0.5000
C =
     0     0     1     0
D =
     0
Eigenvalues of the "Linear Model"
ev =
         0
         0
    4.5388
   -4.5388
Controllability Matrix for this system
CM =
         0   -1.0000         0  -20.6010
   -1.0000         0  -20.6010         0
         0    0.5000         0    0.4905
    0.5000         0    0.4905         0
Rank of Controllability Matrix
ans =
     4
Desired closed loop poles for state feedback controller
clp =
  -1.5000 + 3.0000i  -1.5000 - 3.0000i  -5.0000 + 0.0000i  -6.0000 + 0.0000i
State feedback gains needed to give desired poles
Ks =
 -112.0528  -24.8945  -34.4037  -21.7890
```

```
Calculated eigenvalues of system with state feedback
ans =
  -6.0000 + 0.0000i
  -5.0000 + 0.0000i
  -1.5000 + 3.0000i
  -1.5000 - 3.0000i
Setpoint gain for zero SS error
Nr =
  -34.4037
Observability Matrix for this system
OM =
         0         0    1.0000         0
         0         0         0    1.0000
   -0.4905         0         0         0
         0   -0.4905         0         0
Rank of Observability Matrix
ans =
     4
Desired observer poles for state feedback controller
op =
  -3.0000 + 6.0000i  -3.0000 - 6.0000i -10.0000 + 0.0000i -12.0000 + 0.0000i
Estimator gains needed to give desired poles
L =
   1.0e+04 *
   -0.4662
   -2.4348
    0.0028
    0.0318
Calculated eigenvalues of estimator system
ans =
 -12.0000 + 0.0000i
 -10.0000 + 0.0000i
  -3.0000 + 6.0000i
  -3.0000 - 6.0000i
Perform frequency domain analysis? (y/n) [n]:  y

Note that this system is non minimum phase because of the zero
in the right portion of the complex plane

G(s) for Cart Position (Transfer Function form)
num1 =
         0         0  -17.2018   -0.0000  337.5000
den1 =
    1.0000   14.0000   74.2500  213.7500  337.5000

num/den =
       -17.2018 s^2 - 3.0557e-14 s + 337.5
   -------------------------------------------
   s^4 + 14 s^3 + 74.25 s^2 + 213.75 s + 337.5

G(s) for Cart Position (Zero-Pole form)
z1 =
    4.4294
   -4.4294
p1 =
  -6.0000 + 0.0000i
  -5.0000 + 0.0000i
  -1.5000 + 3.0000i
  -1.5000 - 3.0000i
k1 =
  -17.2018

ans =
   -17.202 (s-4.429) (s+4.429)
  ------------------------------
  (s+6) (s+5) (s^2 + 3s + 11.25)

Continuous-time zero/pole/gain model.

Damping and natural frequencies for closed loop system
         Pole              Damping       Frequency       Time Constant
                                       (rad/TimeUnit)     (TimeUnit)
 -6.00e+00                 1.00e+00       6.00e+00          1.67e-01
 -5.00e+00                 1.00e+00       5.00e+00          2.00e-01
 -1.50e+00 + 3.00e+00i     4.47e-01       3.35e+00          6.67e-01
 -1.50e+00 - 3.00e+00i     4.47e-01       3.35e+00          6.67e-01
```

Table 7.10 Listing of the invpn3.m program.

```
%
%   INVPN3.M    Inverted Pendulum Demonstration #3 (with disturbance input)
%
%   This file simulates the inverted pendulum (linear model) with state
%   feedback control.  It is similar to INVPN2.M except this simulation contains
%   a disturbance input.  We will use the state gains from the standard design
%   with this case.  Let's see how this works...
%
%   File prepared by J. R. White, UMass-Lowell (last update: March 2020)
%

      clear all,   close all,   nfig = 0;
%
      disp(' ')
      disp(' *** Summary Data from INVPN3.M ***')
      disp(' ')
%
%   basic data
      M = 2.0;  m = 0.1;     % mass of cart and mass at end (kg)
      len = .5;              % length of pendulum rod (m)
      g = 9.81;              % gravitational acceleration (m/s^2)
%
%  create state space matrices for linear model (output cart position)
      c1 = M*len;  c2 = m*len;  c3 = m*g;   c4 = (M+m)*g;
      A = [0  1  0  0;  c4/c1  0  0  0; 0  0  0  1; -c3/M  0  0  0];
      B1 = [0 -1/c1 0 1/M]';   B2 = [0 1/c2 0 0]';
      C = [0  0  1  0];      D = [0];
      disp('State Space Matrices for the Linear Model')
      A,  B1,  B2,  C
%
%   check for full state controllability (SISO system)
     disp('Controllability Matrix for this system'),   CM = ctrb(A,B1)
     disp('Rank of Controllability Matrix'),   rank(CM)
%
%   calculate state feedback gains for specified closed loop poles (SISO system)
      clp = [-1.5+3.0j  -1.5-3.0j  -5.0  -6.0];
      Ks = place(A,B1,clp);
      disp('Desired closed loop poles for state feedback controller');     clp
      disp('State feedback gains needed to give desired poles');           Ks
      disp('Calculated eigenvalues of system with state feedback'); eig(A-B1*Ks)
%
%   calculate Nr for zero SS error with no disturbance  (see derivation in notes)
      Nr = -1.0/(C*inv(A-B1*Ks)*B1);
      disp('Setpoint gain for zero SS error');    Nr
%
%   simulate linear plant + controller (use lsim for consistency in all cases)
      BB = [B1 B2];   D = [0 0];   % two inputs and one output
      syscl = ss(A-B1*Ks,BB,C,D);
%
%   Case 1: unit step change in cart position with no disturbance
%
      to = 0;  tf = 5;   Nt = 101;   t = linspace(to,tf,Nt)';
      u1 = zeros(size(t));      % controlled input (initialize)
      rd = ones(size(t));       % step change in setpoint
      v1 = zeros(size(t));      % disturbance input (zero for Case 1)
      w1 = [Nr*rd v1];
      [y1,t,x1] = lsim(syscl,w1,t);
      for i = 1:Nt,  u1(i) = Nr*rd(i)-Ks*x1(i,:)';   end    % controlled input
%
%   plot results from Case 1
      nfig = nfig+1;      figure(nfig)
      subplot(2,1,1),plot(t,y1,'r-','LineWidth',2),grid,ylabel('Cart Position (m)')
      title('Linear Inverted Pendulum (Case 1: rd = 1 & v = 0)')
      subplot(2,1,2),plot(t,u1,'g--',t,10*v1,'b-','LineWidth',2),grid
      ylabel('Inputs (N)'),xlabel('Time (sec)')
      legend('u(t)','10*v(t)','Location','SouthEast')
%
%   Case 2: no change in cart position with constant disturbance (0.2 N)
%
      u2 = zeros(size(t));          % controlled input (initialize)
      rd = zeros(size(t));          % no change in setpoint
      v2 = 0.2*ones(size(t));       % disturbance input (0.2N)
      w2 = [Nr*rd v2];
```

```
      [y2,t,x2] = lsim(syscl,w2,t);
      for i = 1:Nt,  u2(i) = Nr*rd(i)-Ks*x2(i,:)';   end    % controlled input
%
%   plot results from Case 2
      nfig = nfig+1;      figure(nfig)
      subplot(2,1,1),plot(t,y2,'r-','LineWidth',2),grid,ylabel('Cart Position (m)')
      title('Linear Inverted Pendulum (Case 2: rd = 0 & v = 0.2N)')
      subplot(2,1,2),plot(t,u2,'g--',t,10*v2,'b-','LineWidth',2),grid
      ylabel('Inputs (N)'),xlabel('Time (sec)')
      legend('u(t)','10*v(t)')
%
%   Case 3: unit step change in cart position with constant disturbance (0.2 N)
%
      u3 = zeros(size(t));          % controlled input (initialize)
      rd = ones(size(t));           % step change in setpoint
      v3 = 0.2*ones(size(t));       % disturbance input (0.2N)
      w3 = [Nr*rd v3];
      [y3,t,x3] = lsim(syscl,w3,t);
      for i = 1:Nt,  u3(i) = Nr*rd(i)-Ks*x3(i,:)';   end    % controlled input
%
%   plot results from Case 3
      nfig = nfig+1;      figure(nfig)
      subplot(2,1,1),plot(t,y3,'r-','LineWidth',2),grid,ylabel('Cart Position (m)')
      title('Linear Inverted Pendulum (Case 3: rd = 1 & v = 0.2N)')
      subplot(2,1,2),plot(t,u3,'g--',t,10*v3,'b-','LineWidth',2),grid
      ylabel('Inputs (N)'),xlabel('Time (sec)')
      legend('u(t)','10*v(t)')
%
%   Case 4: unit step change in cart position with random disturbance (+/- 0.2 N)
%
      u4 = zeros(size(t));        % controlled input (initialize)
      rd = ones(size(t));         % step change in setpoint
      rn = rand(size(t));   a = -0.2;   b = 0.2;
      v4 = (b-a)*rn + a;          % should be uniformly distributed between +/- 0.2 N
      w4 = [Nr*rd v4];
      [y4,t,x4] = lsim(syscl,w4,t);
      for i = 1:Nt,  u4(i) = Nr*rd(i)-Ks*x4(i,:)';   end    % controlled input
%
%   plot results from Case 4
      nfig = nfig+1;      figure(nfig)
      subplot(2,1,1),plot(t,y4,'r-','LineWidth',2),grid,ylabel('Cart Position (m)')
      title('Linear Inverted Pendulum (Case 4: rd = 1 & v = random noise (+/-0.2N))')
      subplot(2,1,2),plot(t,u4,'g--',t,10*v4,'b-','LineWidth',2),grid
      ylabel('Inputs (N)'),xlabel('Time (sec)')
      legend('u(t)','10*v(t)','Location','SouthEast')
%
%   end of simulation
```