ME 601 Advanced System Analysis & Control

<u>Dr. Bob's Super-Detailed Lecture Notes</u>

Dr. Bob Williams
Mechanical Engineering
Ohio University
phone: (740) 593-1096
fax: (740) 593-0476
bobw@bobcat.ent.ohiou.edu
http://www.ent.ohiou.edu/~bobw

Table of Contents

1. Introduction

1.1 History and Context

1.2 State-Space Description of Dynamical Systems

State-space Concept

Transfer functions classically limited to linear SISO systems
- Matrix of transfer functions to extend to MIMO

State-space formulation

- linear or non-linear
- SISO or MIMO
- time-invariant or varying

Basic system diagram: *r* inputs, *m* outputs

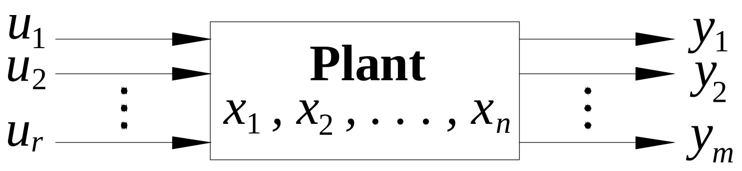

Figure 2.1a

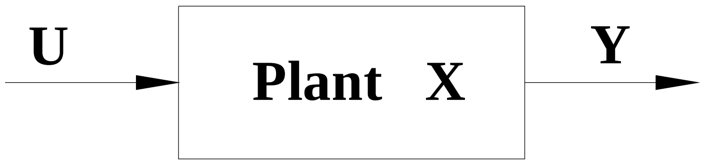

Figure 2.1b

State Vector X(t)

State vector is composed of state variables: minimum set of parameters which uniquely describe the future response of a system given the current state, input, and dynamics equations.

There are infinite choices for state variables and hence infinite state-space representations for the same system.

State **X**(t) is not the same as output **Y**(t). Output is a physical quantity; we assume each output has an associate sensor to measure its value over time. State can be anything, not even recognizable quantity always.

Basic motivation - convert all dynamics and control system models to 1st order ODEs.

State-space representation converts a single n^{th} order ODE into a system of n coupled 1^{st} order ODEs. Matrix of differential equations. In principle, easier to solve (standardized methods which we shall present later).

Can also convert a system of k n^{th} order ODEs into a matrix system of kn coupled 1^{st} order ODEs.

State-space description

- State differential equations
- Output algebraic equations

Example - linear SISO 1-dof *m*-*c*-*k* mechanical translational system (see Figure 1.2a).

$$m\ddot{y}(t)+c\dot{y}(t)+ky(t)=u(t)$$

$y(t)$ output (displacement); $u(t)$ input (force)

Single 2^{nd} order ODE - need to select $2x1$ state vector: $\mathbf{X}(t)=\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}$

Define:
$$x_1(t)=y(t)$$
$$x_2(t)=\frac{dy(t)}{dt}=\frac{dx_1(t)}{dt}=\dot{x}_1(t)$$
so
$$\dot{y}(t)=x_2(t)$$
$$\ddot{y}(t)=\dot{x}_2(t)$$

Substitute into original equation:

$$m\dot{x}_2(t)+cx_2(t)+kx_1(t)=u(t)$$

Original single 2^{nd} order dynamics ODE can be written as a system of two 1^{st} order dynamics ODEs:

$$\dot{x}_1(t)=x_2(t)$$
$$\dot{x}_2(t)=-\frac{c}{m}x_2(t)-\frac{k}{m}x_1(t)+\frac{1}{m}u(t)$$

And the output is: $y(t)=x_1(t)$

Write these equations in matrix-vector form to get

State-space description

- State differential equations

$$\begin{Bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \{u(t)\}$$

- Output algebraic equation

$$y(t) = [1 \quad 0] \begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix} + [0]\{u(t)\}$$

In this example, the state vector is composed of the position and velocity of the output $y(t)$ (the output and its time derivative - not the acceleration!). There are two states required since we started with one 2*nd* order ODE.

The state variables are not always physically identifiable; for instance we could transform the above state differential equations into another basis (there are infinite choices for basis) so that the state variables are each some strange combination of the more-logical $y(t), \dot{y}(t)$.

General Form of State-Space Description Differential-Algebraic Equations

r	number of inputs
m	number of outputs
n	number of state variables (order of ODE for SISO; combination of ODE orders and numbers of equations for MIMO)

- State differential equations: Matrix of 1st order ODEs - represents system dynamics. Solution of this equation yields the state vector $\mathbf{X}(t)$.

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$

A	*n*x*n*	System Dynamics Matrix
X	*n*x1	State Vector
B	*n*x*r*	Input Matrix
U	*r*x1	Input Vector

- Output algebraic equations - calculates output vector $\mathbf{Y}(t)$ given the state vector and possibly the input vector.

$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$$

Y	*m*x1	Output Vector
C	*m*x*n*	Output Matrix
D	*m*x*r*	Direct Transmission Matrix

$\mathbf{D} = [\mathbf{0}]$ for most physical systems because dynamics appear in all paths from input to output!

Example 2nd order SISO system from above: $r = m = 1,\ n = 2$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{B}u & \qquad \{2x1\} &= [2x2]\{2x1\} + [2x1]\{1x1\} \\ y &= \mathbf{CX} + \mathbf{D}u & \qquad \{1x1\} &= [1x2]\{2x1\} + [1x1]\{1x1\} \end{aligned}$$

State-Space Description of General *n*th order ODE (SISO)

$$\frac{d^n y}{dt^n} + a_{n-1}\frac{d^{n-1}y}{dt^{n-1}} + \cdots + a_2\frac{d^2 y}{dt^2} + a_1\frac{dy}{dt} + a_0 y = u(t)$$

Define state variables:

$$\begin{aligned} x_1 &= y \\ x_2 &= \frac{dy}{dt} \\ x_3 &= \frac{d^2 y}{dt^2} \\ &\cdots \\ x_n &= \frac{d^{n-1}y}{dt^{n-1}} \end{aligned}$$

Substitute the state variable definitions into the original ODE:

$$\dot{x}_n = -a_0 x_1 - a_1 x_2 - a_2 x_3 - \cdots - a_{n-1}x_n + u(t)$$

State and output equations:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \vdots \\ \dot{x}_n \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \{u(t)\}$$

$$y(t) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} + [0]\{u(t)\}$$

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{B}u & \qquad \{nx1\} &= [nxn]\{nx1\} + [nx1]\{1x1\} \\ y &= \mathbf{CX} + \mathbf{D}u & \qquad \{1x1\} &= [1xn]\{nx1\} + [1x1]\{1x1\} \end{aligned}$$

This form for the state-space description (with 0s and 1s in the first *n*-1 rows of [*A*], the interesting coefficients only in the *n*th row of [*A*]) is called the ***Phase-Variable Canonical Form***; it is also called the

Controllable Canonical Form We will consider other canonical forms when we present similarity transformations in Chapter 6.

Another SISO Example - parallel RLC circuit with current $i(t)$ input and voltage $v(t)$ output (see Figure 1.3).

$$C\frac{dv(t)}{dt}+\frac{1}{R}v(t)+\frac{1}{L}\int v(t)dt=i(t)$$

Integro-differential equation with single integral and time derivative - need to select 2*x*1 state vector:

Define:
$$\begin{aligned}x_1(t)&=\int v(t)dt\\ x_2(t)&=\dot{x}_1(t)=v(t)\end{aligned}$$

Substitute into original equation: $C\dot{x}_2+\frac{1}{R}x_2+\frac{1}{L}x_1=i(t)$

Output $y(t)=v(t)$. Write these equations in matrix-vector form to get

State-space description

- State differential equations

$$\begin{Bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{Bmatrix}=\begin{bmatrix}0 & 1\\ -\frac{1}{LC} & -\frac{1}{RC}\end{bmatrix}\begin{Bmatrix}x_1(t)\\ x_2(t)\end{Bmatrix}+\begin{bmatrix}0\\ \frac{1}{C}\end{bmatrix}\{i(t)\}$$

- Output algebraic equation

$$y(t)=\begin{bmatrix}0 & 1\end{bmatrix}\begin{Bmatrix}x_1(t)\\ x_2(t)\end{Bmatrix}+[0]\{i(t)\}$$

$$\mathbf{A}=\begin{bmatrix}0 & 1\\ -\frac{1}{LC} & -\frac{1}{RC}\end{bmatrix}\qquad \mathbf{B}=\begin{bmatrix}0\\ \frac{1}{C}\end{bmatrix}\qquad \mathbf{C}=\begin{bmatrix}0 & 1\end{bmatrix}\qquad \mathbf{D}=[0]$$

Recall Force-Current analogy.

Block Diagram for General MIMO State-Space Description

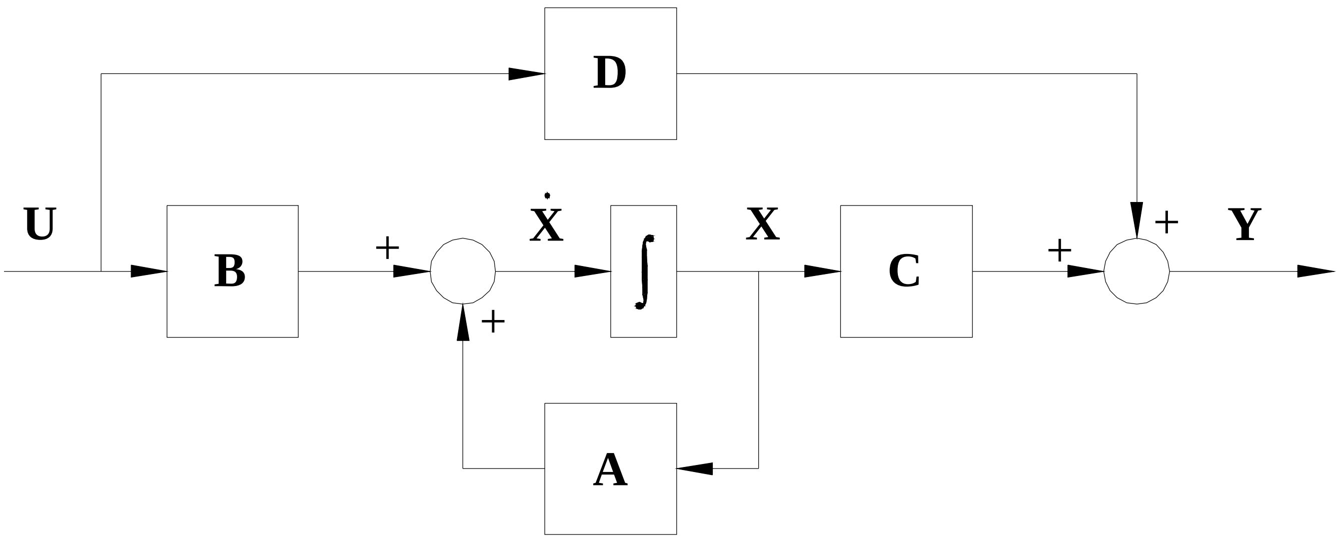

Figure 2.2 State-Space Open-Loop System Block Diagram

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU} \qquad \{nx1\} = [nxn]\{nx1\} + [nxr]\{rx1\}$$
$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU} \qquad \{mx1\} = [mxn]\{nx1\} + [mxr]\{rx1\}$$

MIMO Example: 3-dof linear translational mechanical system (CUT??!?? – Cont Ex I)

Diagram: 2 inputs u_1, u_2
3 outputs y_1, y_2, y_3

Free-body diagrams:

Write 3 equations of motion:

$$k(y_2 - y_1) + c(\dot{y}_2 - \dot{y}_1) + u_1 = m\ddot{y}_1$$
$$k(y_3 - y_2) + c(\dot{y}_3 - \dot{y}_2) - k(y_2 - y_1) - c(\dot{y}_2 - \dot{y}_1) + u_2 = m\ddot{y}_2$$
$$-k(y_3 - y_2) - c(\dot{y}_3 - \dot{y}_2) = m\ddot{y}_3$$

Define state variables:

$$x_1 = y_1 \qquad x_3 = y_2 \qquad x_5 = y_3$$
$$x_2 = \dot{y}_1 \qquad x_4 = \dot{y}_2 \qquad x_6 = \dot{y}_3$$

- State differential equations

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{k}{m} & -\frac{c}{m} & \frac{k}{m} & \frac{c}{m} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k}{m} & \frac{c}{m} & -\frac{2k}{m} & -\frac{2c}{m} & \frac{k}{m} & \frac{c}{m} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k}{m} & \frac{c}{m} & -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_1 \end{Bmatrix}$$

- Output algebraic equation

$$\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_1 \end{Bmatrix}$$

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU} \qquad \{nx1\} = [nxn]\{nx1\} + [nxr]\{rx1\}$$

$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU} \qquad \{mx1\} = [mxn]\{nx1\} + [mxr]\{rx1\}$$

r=2; *m*=3; *n*=6 (3 - 2nd order ODEs)

State-Space Description for System with Zero(s)

Example

$\ddot{y}(t)+2\dot{y}(t)+10y(t)=\dot{u}(t)+3u(t)$ Transfer function: $\dfrac{Y(s)}{U(s)}=\dfrac{s+3}{s^2+2s+10}$

Separate transfer function with intermediate variable *w*:

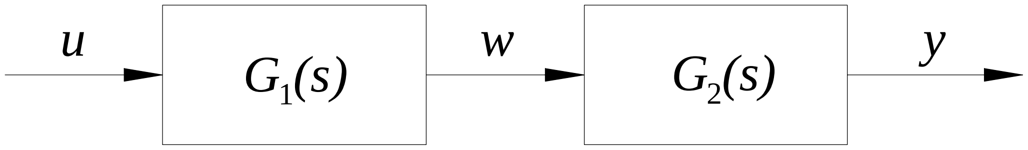

Figure 2.3 Separate Transfer Functions

Where $G_1(s)=\dfrac{1}{s^2+2s+10}$ and $G_2(s)=s+3$. Now, two differential equations:

$$\ddot{w}(t)+2\dot{w}(t)+10w(t)=u(t)$$
$$y(t)=\dot{w}(t)+3w(t)$$

Still 2nd order system - need to select 2*x*1 state vector: $X(t)=\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}$

Define: $\begin{aligned} x_1(t)&=w(t) \\ x_2(t)&=\dot{w}(t)=\dot{x}_1(t) \end{aligned}$

$\ddot{w}(t)=-2\dot{w}(t)-10w(t)+u(t)$, which leads to the state equations:

$$\begin{aligned} \dot{x}_1(t)&=x_2(t) \\ \dot{x}_2(t)&=-2x_2(t)-10x_1(t)+u(t) \end{aligned}$$

Output equation: $y(t)=x_2(t)+3x_1(t)$

$$\begin{Bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{Bmatrix}=\begin{bmatrix} 0 & 1 \\ -10 & -2 \end{bmatrix}\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}+\begin{bmatrix} 0 \\ 1 \end{bmatrix}\{u(t)\} \qquad y(t)=\begin{bmatrix} 3 & 1 \end{bmatrix}\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}+[0]u(t)$$

$$\mathbf{A}=\begin{bmatrix} 0 & 1 \\ -10 & -2 \end{bmatrix} \qquad \mathbf{B}=\begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{C}=\begin{bmatrix} 3 & 1 \end{bmatrix} \qquad \mathbf{D}=[0]$$

1.3 State-Space and Transfer Function Relationships

State-Space Equations

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$
$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$$

Given State-Space Matrices **A**, **B**, **C**, **D**, find the equivalent transfer function description for the same linear system. Matrix of transfer functions (assume MIMO case):

$$\mathbf{Y}(s) = \mathbf{G}(s)\mathbf{U}(s)$$

Output equals matrix of transfer functions times input; no explicit state vector involved. The scalar transfer function for SIS systems is:

$$G(s) = \frac{Y(s)}{U(s)}$$

Laplace Transforms of State-Space Equations

$$s\mathbf{X}(s) - \mathbf{X}(0) = \mathbf{AX}(s) + \mathbf{BU}(s)$$
$$\mathbf{Y}(s) = \mathbf{CX}(s) + \mathbf{DU}(s)$$

For transfer functions - zero initial conditions so $\mathbf{X}(0) = \{\mathbf{0}\}$

$$[s\mathbf{I} - \mathbf{A}]\mathbf{X}(s) = \mathbf{BU}(s)$$
$$\mathbf{X}(s) = [s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{BU}(s)$$

Substitute this **X**(s) into output equation to eliminate explicit state dependence.

$$\mathbf{Y}(s) = \mathbf{C}[sI - A]^{-1}\mathbf{BU}(s) + \mathbf{DU}(s) = \left[\mathbf{C}[sI - A]^{-1}\mathbf{B} + \mathbf{D}\right]\mathbf{U}(s)$$

And so:

$$\mathbf{G}(s) = \mathbf{C}[sI - A]^{-1}\mathbf{B} + \mathbf{D}$$

In the general MIMO case, this is a matrix of transfer functions where $G_{ij}(s)$ is the scalar transfer function giving the contribution of input j to output i.

Example – SISO linear 1-dof *m*=1, *c*=0.1, *k*=10 mechanical translational system

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -10 & -0.1 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D} = [0]$$

$$G(s) = \mathbf{C}[sI - A]^{-1}\mathbf{B} + \mathbf{D}$$

$$s\mathbf{I} - \mathbf{A} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ -10 & -0.1 \end{bmatrix} = \begin{bmatrix} s & -1 \\ 10 & s+0.1 \end{bmatrix}$$

$$[s\mathbf{I} - \mathbf{A}]^{-1} = \frac{1}{s(s+0.1)+10} \begin{bmatrix} s+0.1 & 1 \\ -10 & s \end{bmatrix}$$

$$G(s) = \frac{1}{s^2 + 0.1s + 10} \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} s+0.1 & 1 \\ -10 & s \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + [0]$$

$$G(s) = \frac{1}{s^2 + 0.1s + 10}$$

Check:

$$m\ddot{y}(t) + c\dot{y}(t) + ky(t) = u(t)$$

$$ms^2Y(s) + csY(s) + kY(s) = U(s)$$

$$G(s) = \frac{Y(s)}{U(s)} = \frac{1}{ms^2 + cs + k} = \frac{1}{s^2 + 0.1s + 10}$$ Agrees!

1.4 Linearization of Nonlinear Systems

1.5 Matlab for State-Space Description

There a three primary ways to describe a dynamic system to Matlab: 1. state-space realizations **ABCD**; 2. transfer functions with (*num*,*den*), where *num* is the array of polynomial coefficients for the transfer function numerator and *den* is the array of polynomial coefficients for the transfer function denominator; and 3. transfer functions with (*zeros*,*poles*), where *zeros* is the array of numerator polynomial roots and *den* is the array of denominator polynomial roots.

To convert between these various system representations, we have the following Matlab functions:

ss2tf	Covert state-space realization **ABCD** to transfer function (*num*,*den*).
tf2ss	Covert transfer function (*num*,*den*) to state-space realization **ABCD**.
ss2zp	Covert state-space realization **ABCD** to transfer function (*zeros*,*poles*).
zp2ss	Covert transfer function (*zeros*,*poles*) to state-space realization **ABCD**.
tf2zp	Covert transfer function (*num*,*den*) to transfer function (*zeros*,*poles*).
zp2tf	Covert transfer function (*zeros*,*poles*) to transfer function (*num*,*den*).
poly	Find the system characteristic polynomial coefficients from **A** or from *den*.

<u>Continuing Matlab Example: State-Space Description</u>

Derive a valid state-space description for the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ). That is, specify the state variables and derive matrices **A**, **B**, **C**, and **D**. Write out the results in full matrix-vector form. Explicitly give the system order and complete matrix/vector dimensions of the results.

We start with the 2nd-order ODE (1) in the Chapter 1 Matlab Example. Since we have a 2nd-order ODE, we need to define two state variables x_i (n=2). A good set of choices is:

$$\begin{aligned} x_1 &= \theta \\ x_2 &= \dot{\theta} = \dot{x}_1 \end{aligned} \tag{2}$$

We will have two 1st-order ODEs, derived from the original 2nd-order ODE (1), and from $\dot{x}_1 = x_2$ above. The state differential equations $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ are:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{-k_R}{J} & \dfrac{-b}{J} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \dfrac{1}{J} \end{bmatrix} \{\tau\} \tag{4}$$

The output algebraic equation $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$ comes from $y = \theta = x_1$:

$$\{y\} = [1 \quad 0] \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + [0]\{\tau\} \tag{5}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k_R}{J} & -\frac{b}{J} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D} = [0]$$

3rd-order SISO system:

$r = 1$ # inputs
$m = 1$ # outputs
$n = 2$ # states

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{BU} & \{2x1\} &= [2x2]\{2x1\} + [2x1]\{1x1\} \\ \mathbf{Y} &= \mathbf{CX} + \mathbf{DU} & \{1x1\} &= [1x2]\{2x1\} + [1x1]\{1x1\} \end{aligned}$$

Let us assume the following constant, lumped parameters for the Continuing Matlab Example:

$$J = 1\ kg\text{-}m^2 \qquad b = 4\ Nms/rad \qquad k_R = 40\ Nm/rad$$

$$\mathbf{A} == \begin{bmatrix} 0 & 1 \\ -40 & -4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D} = [0]$$

Chapter-by-chapter we will present actual Matlab code and results dealing with the topics at hand for the Continuing Matlab Example. These will be complete only if taken together over all chapters (i.e. ensuing code portions may require previously-defined variables in earlier chapters to run without errors). To get started, we need to define the state-space matrices **A**, **B**, **C**, **D** to Matlab. Then we can find the system characteristic polynomial and transfer function description.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 2. State-Space Description
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

J = 1;
b = 4;
kR = 40;

A = [0 1;-kR/J -b/J];               % Define the state-space realization A, B, C, D
B = [0;1/J];
C = [1 0];
D = [0];

CharPoly = poly(A);                 % Find the system characteristic polynomial from A

[num,den] = ss2tf(A,B,C,D);         % Find SISO transfer function
printsys(num,den);

[A1,B1,C1,D1] = tf2ss(num,den);     % Check - find A,B,C,D from transfer function
printsys(A1,B1,C1,D1);
```

The *poly* and *printsys* (transfer function and state-space, respectively) commands result in:

```
CharPoly =
    1.0000    4.0000   40.0000


num/den =
            1
   -----------------
     s^2 + 4 s + 40


a =
                       x1           x2
           x1     -4.00000    -40.00000
           x2      1.00000            0

b =
                       u1
           x1      1.00000
           x2            0

c =
                       x1           x2
           y1           0      1.00000

d =
                       u1
           y1            0
```

Note in the *tf2ss* command, Matlab chooses a different order for the state variables than our examples thus far:

$$x_2 = y$$
$$x_1 = \dot{y} = \dot{x}_2$$

This swaps the columns of **A** and **C** compared to our example above. Also, Matlab reverses the order of equations; this swaps the rows **A** and **B** compared to our example above.

1.6 Continuing Examples

1.6.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

This example will continue throughout each chapter of this book, building chapter-by-chapter to demonstrate the important topics.

1.6.1.1 Modeling

A complex nonlinear, time-varying mechanical system is greatly simplified to the 2 degree-of-freedom (dof) linear, constant-coefficient, lumped-parameter system shown in Figure 1.5a. There are two inputs $u_i(t)$ and two outputs $y_i(t)$, $i = 1,2$. The constant lumped parameters are point masses m_i, linear spring coefficients k_i, and linear damping coefficients c_i, $i = 1,2$. Derive the linear model for this system, i.e. draw the free-body diagrams and write the correct number of independent ordinary differential equations. All motion is constrained to be horizontal as shown in Figure 1.5a. Outputs y_i are each measured from the neutral spring equilibrium location of each point mass m_i.

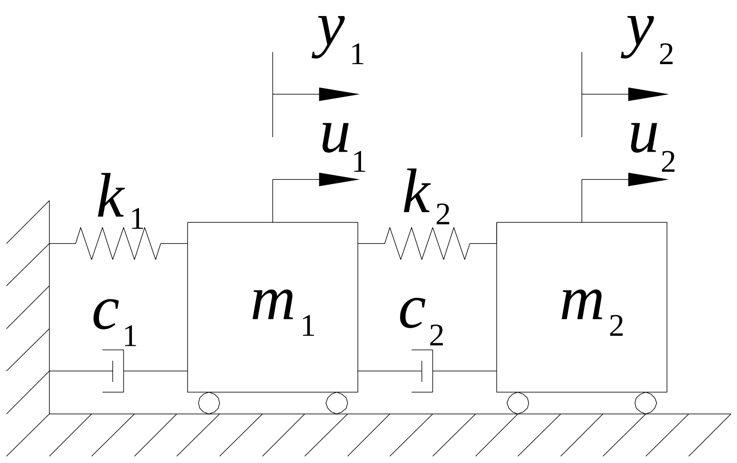

Figure 1.5a 2-dof Lumped, Linear, Constant-Coefficient m-c-k System

<u>Solution</u>

First, assume all y_i are small positive displacements; further assume $y_2 > y_1$. During the ensuing vibratory motion these will all take +/- directions at different times, but the general equations derived with the positive assumption hold good for all motion. Figure 1.5b shows the two free-body diagrams:

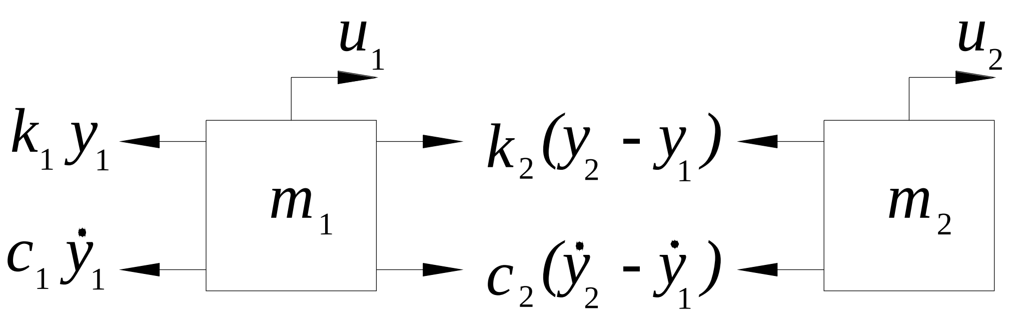

Figure 1.5b 2-dof m-c-k System Free-Body Diagrams

Now use Newton's 2nd law twice, once for each mass, to derive the two independent 2nd order dynamic equations of motion.

$$\begin{aligned}\sum F_1 &= m_1\ddot{y}_1 = k_2(y_2 - y_1) + c_2(\dot{y}_2 - \dot{y}_1) - k_1 y_1 - c_1\dot{y}_1 + u_1 \\ \sum F_2 &= m_2\ddot{y}_2 = -k_2(y_2 - y_1) - c_2(\dot{y}_2 - \dot{y}_1) + u_2\end{aligned} \tag{1}$$

Rewrite equations (1) so the output terms y_i appear on the left side along with their derivatives and the input forces u_i are on the right. Also, combine y_i terms:

$$\begin{aligned}m_1\ddot{y}_1 + (c_1 + c_2)\dot{y}_1 + (k_1 + k_2)y_1 - c_2\dot{y}_2 - k_2 y_2 &= u_1 \\ m_2\ddot{y}_2 + c_2\dot{y}_2 + k_2 y_2 - c_2\dot{y}_1 - k_2 y_1 &= u_2\end{aligned} \tag{2}$$

In (1) and (2) note that inputs u_i outputs y_i are functions of time, while m_i, k_i, and c_i, are constants, i = 1,2.

Equations (2) are two linear, coupled, 2nd-order ordinary differential equations (ODEs). In this type of vibrational system, you will always find that for equation i, the i^{th} displacement and its derivatives are always positive in the equations of motion; i.e. $m_i\ddot{y}_i + c_i\dot{y}_i + k_i y_i = u_i$. Note then the cross-coupling terms all always negative.

Example I is a MIMO (multiple-input, multiple output) system with two inputs u_i and two outputs y_i.

We can express the two 2nd-order ODEs of (2) in standard 2nd-order matrix vector form, $\mathbf{M\ddot{Y}} + \mathbf{C\dot{Y}} + \mathbf{KY} = \mathbf{U}$:

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{Bmatrix} + \begin{bmatrix} c_1 + c_2 & -c_2 \\ -c_2 & c_2 \end{bmatrix}\begin{Bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{Bmatrix} + \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \tag{3}$$

1.6.1.2 State-Space Description

Derive a valid state-space description for Continuing Example I (Two-mass MIMO Translational Mechanical System). That is, specify the state variables and derive matrices **A**, **B**, **C**, and **D**. Write out the results in full matrix-vector form. Explicitly give the system order and complete matrix/vector dimensions of the results. Do two distinct cases:

i. Full MIMO; two inputs, two outputs
ii. SISO; one input $u_2(t)$, one output $y_1(t)$

We start with the two coupled 2nd order ODEs derived in the Chapter 1 Example I; it will be convenient to use the equations (1) since the highest-order derivatives $\ddot{y}_i(t)$ are isolated in those forms. For case i, we use both inputs u_i; for case ii, we must set u_1=0. For both cases the choice of state variables and the resulting **A** system dynamics matrix will be identical. This will always be true; i.e. **A** is fundamental to the system and does not change with different choices for inputs and outputs.

<u>Solution, Case i: Full MIMO</u>

Since we have two 2nd-order ODEs, we need to define four state variables x_i (n=4). A good set of choices is:

$$\begin{aligned} x_1 &= y_1 & x_3 &= y_2 \\ x_2 &= \dot{y}_1 = \dot{x}_1 & x_4 &= \dot{y}_2 = \dot{x}_3 \end{aligned} \tag{4}$$

We will have four 1st-order ODEs, derived from the original two 2nd-order ODEs; two are $\dot{x}_i = x_{i+1}$ from the state variable definitions above, for i = 1, 3. The remaining two come from the original 2nd-order ODEs, rewritten as follows (substituting the state variable definitions in place of outputs y and their derivatives; also, collect terms of state variables and divide by m_i to normalize each equation):

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{-(k_1+k_2)x_1-(c_1+c_2)x_2+k_2x_3+c_2x_4+u_1}{m_1} \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \frac{k_2x_1+c_2x_2-k_2x_3-c_2x_4+u_2}{m_2} \end{aligned} \tag{5}$$

The state differential equations $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ are:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{-(k_1+k_2)}{m_1} & \dfrac{-(c_1+c_2)}{m_1} & \dfrac{k_2}{m_1} & \dfrac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_2}{m_2} & \dfrac{c_2}{m_2} & \dfrac{-k_2}{m_2} & \dfrac{-c_2}{m_2} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ \dfrac{1}{m_1} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{m_2} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \qquad (6)$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \dfrac{-(k_1+k_2)}{m_1} & \dfrac{-(c_1+c_2)}{m_1} & \dfrac{k_2}{m_1} & \dfrac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \dfrac{k_2}{m_2} & \dfrac{c_2}{m_2} & \dfrac{-k_2}{m_2} & \dfrac{-c_2}{m_2} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ \dfrac{1}{m_1} & 0 \\ 0 & 0 \\ 0 & \dfrac{1}{m_2} \end{bmatrix}$$

The output algebraic equations $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$ come from the state definitions (4):

$$\begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} \qquad (7)$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

4th-order MIMO system:

$r = 2$ # inputs
$m = 2$ # outputs
$n = 4$ # states

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU} \qquad \{4x1\} = [4x4]\{4x1\} + [4x2]\{2x1\}$$
$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU} \qquad \{2x1\} = [2x4]\{4x1\} + [2x2]\{2x1\}$$

Solution, Case ii SISO: one input u_2, one output y_1

Remember, **A** does not change by considering different inputs and outputs to the system; **A** is fundamental to the system itself. For SISO case ii, only **B**, **C**, and **D** change. The state differential equations $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ are now:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{-(k_1+k_2)}{m_1} & \frac{-(c_1+c_2)}{m_1} & \frac{k_2}{m_1} & \frac{c_2}{m_1} \\ 0 & 0 & 0 & 1 \\ \frac{k_2}{m_2} & \frac{c_2}{m_2} & \frac{-k_2}{m_2} & \frac{-c_2}{m_2} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_2} \end{bmatrix} \{u_2\} \qquad (8)$$

A is the same as that in (6) $\qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_2} \end{bmatrix}$

The output algebraic equation $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$ is now:

$$\{y_1\} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} + [0]\{u_2\} \qquad (9)$$

$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = [0]$

Still a 4th-order system, now SISO:

$r = 1$ # inputs
$m = 1$ # outputs
$n = 4$ # states (n is the same as Case i)

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{BU} & \{4x1\} &= [4x4]\{4x1\} + [4x1]\{1x1\} \\ \mathbf{Y} &= \mathbf{CX} + \mathbf{DU} & \{1x1\} &= [1x4]\{4x1\} + [1x1]\{1x1\} \end{aligned}$$

1.6.2 Continuing Example II: Rotational Electromechanical System

This example will also continue throughout each chapter of this book, building chapter-by-chapter to demonstrate the important topics.

1.6.2.1 Modeling

A simplified 1-dof DC servomotor model is shown in Figure 1.6. The single input is armature voltage $v(t)$ and the output is motor shaft angle $\theta(t)$. The constant lumped parameters are armature circuit inductance and resistance L and R, respectively, and motor shaft rotational inertia and rotational viscous damping coefficient J and b, respectively. The intermediate variables are armature current $i(t)$, motor torque $\tau(t)$, and $\omega(t) = d\theta/dt$. In this continuing Example II, we have simplified the model: we ignore back emf voltage, there is no gear ratio or load inertia, and the numbers (see Chapter 3) are chosen for a simple integer characteristic polynomial rather than for realism. For improvements on each of these issues, please see Continuing Exercise 3.

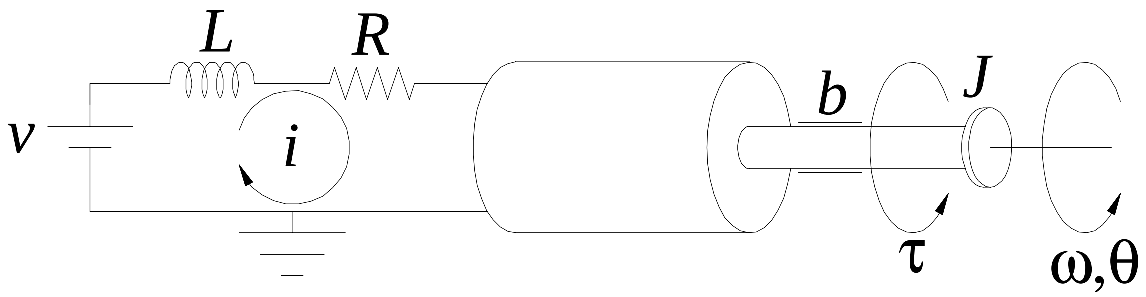

Figure 1.6 Simplified DC Servomotor Model

Solution

We can derive the dynamic model of this system in three steps: circuit model, electromechanical coupling, and rotational mechanical model. For the circuit model, Kirchoff's voltage law yields a first-order ODE with voltage input and current output:

$$L\frac{di(t)}{dt} + Ri(t) = v(t) \tag{1}$$

Motor torque is proportional to the circuit current, so the electromechanical coupling equation is:

$$\tau(t) = k_T i(t) \tag{2}$$

For the rotational mechanical model, $\sum M = J\alpha$ results in a second-order ODE with torque input and motor shaft angle output:

$$J\ddot{\theta}(t)+b\dot{\theta}(t)=\tau(t) \tag{3}$$

To derive the overall system model we need to use voltage input and motor shaft angle output; the intermediate variables $i(t)$ and $\tau(t)$ must be eliminated. It is convenient to use Laplace transforms and transfer functions for this purpose, due to the differential terms. We have:

$$\frac{I(s)}{V(s)}=\frac{1}{Ls+R} \qquad \frac{T(s)}{I(s)}=k_T \qquad \frac{\Theta(s)}{T(s)}=\frac{1}{Js^2+bs} \tag{4}$$

Multiplying the transfer functions (4) together we eliminate the intermediate variables to generate the overall transfer function:

$$\frac{\Theta(s)}{V(s)}=\frac{k_T}{(Ls+R)(Js^2+bs)} \tag{5}$$

Simplifying, cross multiplying, and taking the inverse Laplace transform yields the single, third-order, linear, constant-coefficient ordinary differential equation (6):

$$LJ\dddot{\theta}(t)+(Lb+RJ)\ddot{\theta}(t)+Rb\dot{\theta}(t)=k_T v(t) \tag{6}$$

(6) is the dynamic model for the system of Figure 1.6. Note there is no rotational mechanical spring term in this equation, i.e. the coefficient of the $\theta(t)$ term is zero.

1.6.2.2 State-Space Description

Derive a valid state-space description for Continuing Example II (1-dof rotational electromechanical system; SISO: input v, output θ). That is, specify the state variables and derive matrices **A**, **B**, **C**, and **D**. Write out the results in full matrix-vector form. Explicitly give the system order and complete matrix/vector dimensions of the results.

We start with the 3rd-order ODE (1) in the Chapter 1 Example II.

<u>Solution</u>

Since we have a 3rd-order ODE, we need to define three state variables x_i (n=3). A good set of choices is:

$$\begin{aligned} x_1 &= \theta \\ x_2 &= \dot{\theta}=\dot{x}_1 \\ x_3 &= \ddot{\theta}=\dot{x}_2 \end{aligned} \tag{7}$$

We will have three 1st-order ODEs, derived from the original 3rd-order ODE (6); two are $\dot{x}_i = x_{i+1}$ from the state variable definitions above, for i = 1, 2. The remaining 1st-order ODE comes from the original

3rd-order ODE, rewritten as follows (substituting the state variable definitions in place of output θ and its derivatives; also, collect terms of state variables and divide the third equation by *LJ* to normalize it:

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= x_3 \\
\dot{x}_3 &= \frac{-(Lb+RJ)}{LJ}x_3 - \frac{Rb}{LJ}x_2 + \frac{k_T v(t)}{LJ}
\end{aligned} \tag{8}$$

The state differential equations $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ are:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{-Rb}{LJ} & \frac{-(Lb+RJ)}{LJ} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{k_T}{LJ} \end{bmatrix} \{v\} \tag{9}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{-Rb}{LJ} & \frac{-(Lb+RJ)}{LJ} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \frac{k_T}{LJ} \end{bmatrix}$$

The output algebraic equation $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$ comes from $y = \theta = x_1$:

$$\{y\} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + [0]\{v\} \qquad (10)$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad\qquad \mathbf{D} = [0]$$

3rd-order SISO system:

- $r = 1$ # inputs
- $m = 1$ # outputs
- $n = 3$ # states

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU} \qquad \{3x1\} = [3x3]\{3x1\} + [3x1]\{1x1\}$$

$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU} \qquad \{1x1\} = [1x3]\{3x1\} + [1x1]\{1x1\}$$

1.7 Homework Assignments

1.7.1 Mathematical Homework Assignments

1.7.2 Matlab Homework Assignments

1.7.3 Continuing Homework Assignments

CE1.1a
A complex nonlinear, time-varying mechanical system is greatly simplified to the 3 degree-of-freedom (dof) linear, constant-coefficient, lumped-parameter system shown in Figure CE1. There are three input forces $u_i = f_i(t)$ and three output displacements $y_i = y_i(t)$, $i = 1,2,3$. The constant lumped parameters are point masses m_i, $i = 1,2,3$; linear spring coefficients k_j, and linear damping coefficients c_j, $j = 1,2,3,4$. Derive the linear model for this system, i.e. draw the free-body diagrams and write the correct number of independent ordinary differential equations. All motion is constrained to be horizontal. Outputs y_i are each measured from the neutral spring equilibrium location of each point mass m_i. Also express the results in matrix-vector form $[M]\{\ddot{Y}\}+[C]\{\dot{Y}\}+[K]\{Y\}=\{F\}$.

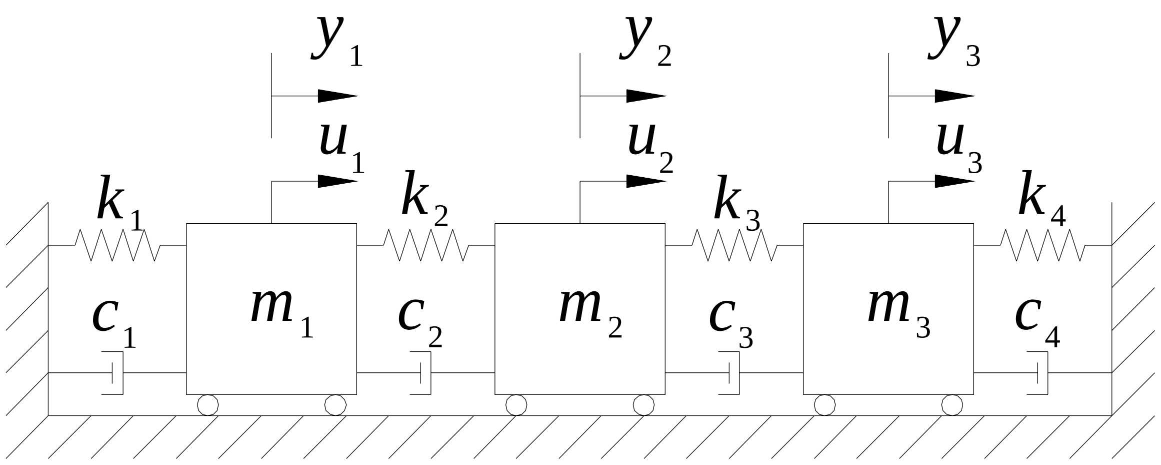

Figure CE1. 3-dof Linear m-c-k System

CE1.1b
Derive a valid state-space realization for the CE1.1a system. That is, specify the state variables and derive matrices **A**, **B**, **C**, and **D**. Write out your results in full matrix-vector form. Explicitly give the system order and complete matrix/vector dimensions of your result. Do three distinct cases:

i. Full MIMO; three inputs, three displacement outputs.
ii. MIMO; two inputs ($u_1(t)$ and $u_3(t)$ only), three displacement outputs.
iii. SISO; input $u_2(t)$ and output $y_3(t)$.

The solution for CE1.1a is given in 2nd-order matrix-vector form below:

$$m_1\ddot{y}_1(t)+(c_1+c_2)\dot{y}_1(t)+(k_1+k_2)y_1(t)-c_2\dot{y}_2(t)-k_2y_2(t)=u_1(t)$$
$$m_2\ddot{y}_2(t)+(c_2+c_3)\dot{y}_2(t)+(k_2+k_3)y_2(t)-c_2\dot{y}_1(t)-k_2y_1(t)-c_3\dot{y}_3(t)-k_3y_3(t)=u_2(t)$$
$$m_3\ddot{y}_3(t)+(c_3+c_4)\dot{y}_3(t)+(k_3+k_4)y_3(t)-c_3\dot{y}_2(t)-k_3y_2(t)=u_3(t)$$

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}\begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \\ \ddot{y}_3 \end{Bmatrix}+\begin{bmatrix} c_1+c_2 & -c_2 & 0 \\ -c_2 & c_2+c_3 & -c_3 \\ 0 & -c_3 & c_3+c_4 \end{bmatrix}\begin{Bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{Bmatrix}+\begin{bmatrix} k_1+k_2 & -k_2 & 0 \\ -k_2 & k_2+k_3 & -k_3 \\ 0 & -k_3 & k_3+k_4 \end{bmatrix}\begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix}=\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix}$$

CE2.1a

The classic nonlinear, inherently unstable, 2-dof inverted pendulum is shown in Figure CE2. The goal is to maintain pendulum angle $\theta = 0$ by using a feedback controller with a sensor (encoder or potentiometer) for θ and input force f. The cart point mass is m_1; the lumped pendulum mass is m_2. There are two possible outputs, pendulum angle θ and cart displacement y (the primary concern is θ). The classical inverted pendulum has only one input, the force f. We will consider a second case, adding a torque motor providing a second input τ (not shown) at the pin joint shown in the figure. For both cases (they will be very similar), derive the nonlinear model for this system, i.e. draw the free-body diagrams and write the correct number of independent ordinary differential equations. Alternately, you may use the Lagrangian dynamics approach that does not require FBDs. Assume our controller will be good enough to use the small angle approximation and linearize your models.

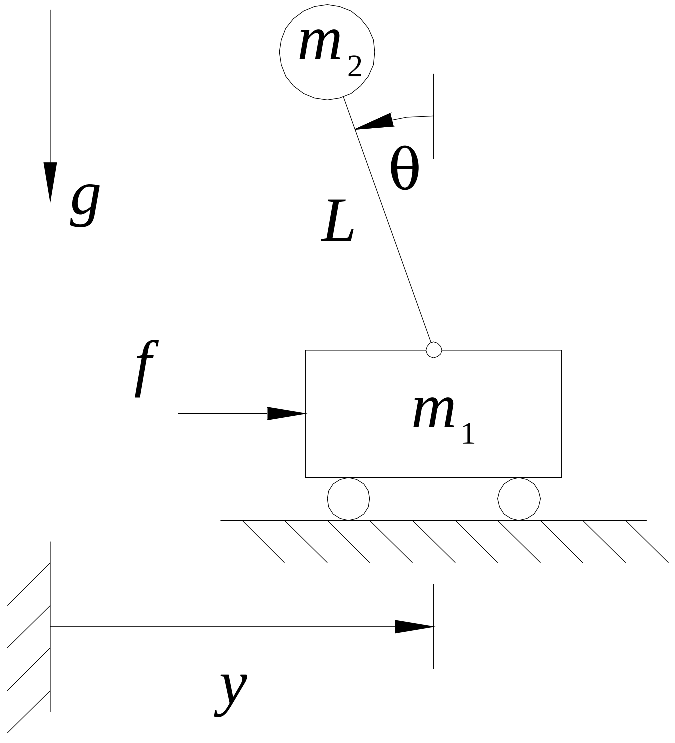

Figure CE2. Inverted Pendulum

CE2.1b

Derive a valid state-space description for the CE2.1 system. That is, specify the state variables and derive matrices **A**, **B**, **C**, and **D**. Write out your results in full matrix-vector form. Explicitly give the system order and complete matrix/vector dimensions of your result. Do three distinct cases:

i. The classical SISO case; input $f(t)$ and output $\theta(t)$.
ii. SIMO; one input $f(t)$, two outputs $y(t)$ and $\theta(t)$.
iii. MIMO; two inputs $f(t)$ and $\tau(t)$ (add a torque motor to the inverted pendulum revolute joint, traveling with the cart), two outputs $y(t)$ and $\theta(t)$.

The solution for CE2.1 is given below:

Coupled non-linear ODEs

$$(m_1+m_2)\ddot{y}(t)-m_2L\cos\theta(t)\ddot{\theta}(t)+m_2L\sin\theta(t)\dot{\theta}(t)^2 = f(t)$$
$$m_2L^2\ddot{\theta}(t)-m_2L\cos\theta(t)\ddot{y}(t)-m_2gL\sin\theta(t)=0$$

Coupled linearized ODEs

$$(m_1+m_2)\ddot{y}(t)-m_2L\ddot{\theta}(t)= f(t)$$
$$-m_2\ddot{y}(t)+m_2L\ddot{\theta}(t)-m_2g\theta(t)=0$$

Coupled linearized ODEs with torque motor included

$$(m_1+m_2)\ddot{y}(t)-m_2L\ddot{\theta}(t)= f(t)$$
$$-m_2L\ddot{y}(t)+m_2L^2\ddot{\theta}(t)-m_2gL\theta(t)=\tau(t)$$

CE3.1a

Figure CE3 shows a 1-dof single robot joint/link driven through a gear ratio n by an armature-controlled DC servomotor as shown. The input is armature voltage $v(t)$ (DC, but you can change the DC level) and the output is load-shaft angle $\theta_L(t)$. Derive the linearized model for this system, i.e. develop the circuit ODE, the electromechanical coupling equations, and the rotational mechanical ODE. Eliminate intermediate variables and simplify; it will be convenient to use a transfer function approach. Assume the mass-moment of inertia of all outboard links plus any load, $J_L(t)$, is a constant (a reasonable assumption when the gear ratio $n = \omega_M/\omega_L$ is much greater than 1 as it is in the case of industrial robots). The parameters in Figure CE3 are named below.

$v(t)$	armature voltage	L	armature inductance	R	armature resistance
$i_A(t)$	armature current	$v_B(t)$	back emf voltage	k_B	motor back emf constant
J_M	lumped motor inertia	b_M	motor viscous damping	$\tau_M(t)$	motor torque
k_T	torque constant	$\omega_M(t)$	motor shaft velocity	$\theta_M(t)$	motor shaft angle
n	gear ratio	$J_L(t)$	total load inertia	b_L	motor viscous damping
$\tau_L(t)$	load shaft torque	$\omega_L(t)$	load shaft velocity	$\theta_L(t)$	load shaft angle

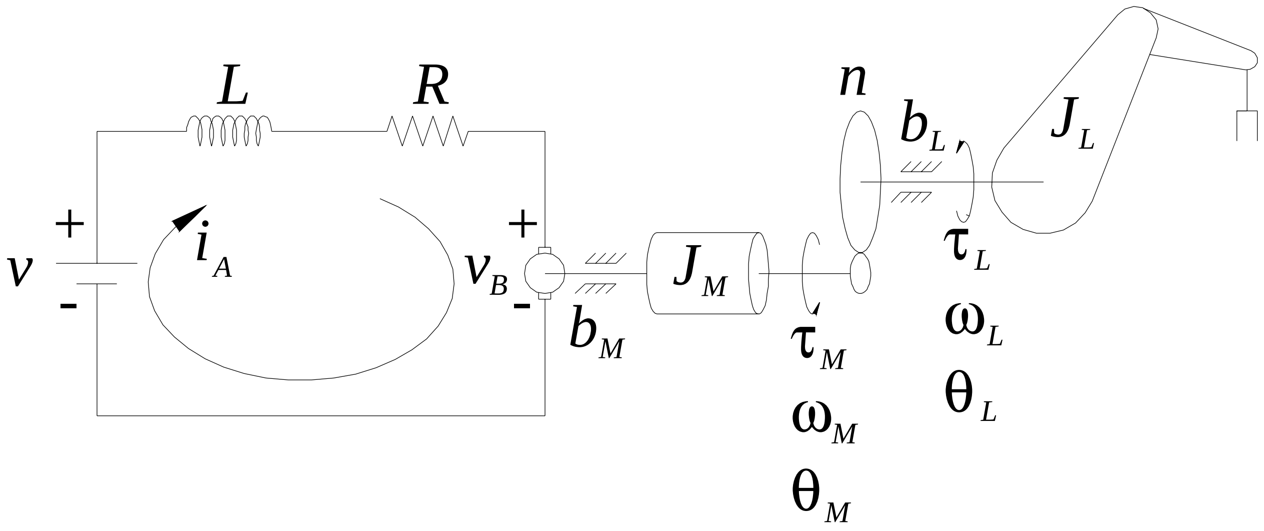

Figure CE3. Robot Joint/Link driven by Armature-controlled DC servomotor

CE3.1b
Derive a valid state-space description for the CE3.1 system. That is, specify the state variables and derive matrices **A**, **B**, **C**, and **D**. Write out your results in full matrix-vector form. Explicitly give the system order and complete matrix/vector dimensions of your result. Do two distinct cases:

i. SISO; input armature voltage $v(t)$ and output robot load shaft angle $\theta_L(t)$.
ii. SISO; input armature voltage $v(t)$ and output robot load shaft angular velocity $\omega_L(t)$.

The solution for CE3.1 is given below; the overall transfer function is:

$$G(s)=\frac{\Theta_L(s)}{V(s)}=\frac{k_T/n}{LJs^3+(Lb+RJ)s^2+(Rb+k_Tk_B)s}$$

Where $J \equiv J_E = J_M + \frac{J_L}{n^2}$ and $b \equiv b_E = b_M + \frac{b_L}{n^2}$ are the effective rotational inertias and viscous damping coefficients, reflected to the motor shaft. The associated SISO ODE is:

$$LJ\dddot{\theta}_L(t)+(Lb+RJ)\ddot{\theta}_L(t)+(Rb+k_Tk_B)\dot{\theta}_L(t)=\frac{k_T}{n}v(t)$$

2. Simulation of State-Space Systems

2.1 Solution of State-Space Equations

State-Space Dynamics Differential Equations

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$

Solve this coupled system of 1*st* order ODEs, then output is found from the linear combination:

$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$$

2.1.1 Solution of Scalar 1st-Order Differential Equations

First, further review **Scalar 1*st* order ODEs**

Physical systems with scalar 1*st* order ODEs

- Translational mechanical w/o mass $c\dot{x}(t) + kx(t) = F(t)$
- Rotational mechanical w/o torsional spring $J\dot{\omega}(t) + B\omega(t) = \tau(t)$
- Series *R*-*L* circuit w/o capacitance $L\dot{i}(t) + Ri(t) = v(t)$

Solution methods

- Slow ME way (homogeneous and particular)
- Laplace Transform
- Matlab *lsim*

Example: solve $\dot{x}(t) + 50x(t) = u(t)$ subject to input $u(t)$ (step input of magnitude 5) and initial condition $x(0) = 0$.
use Laplace transform method:

$$\left(sX(s) - x(0)\right) + 50X(s) = \frac{5}{s}$$

$$(s+50)X(s) = \frac{5}{s}$$

$$X(s) = \frac{5}{s(s+50)} = \frac{C_1}{s} + \frac{C_1}{s+50}$$

$5 = C_1(s+50) + C_2 s$ $\quad\begin{matrix} s^1 : 0 = C_1 + C_2 \\ s^0 : 5 = 50C_1 \end{matrix}$ so $\begin{matrix} C_1 = 0.1 \\ C_2 = -0.1 \end{matrix}$ and thus

$$x(t) = L^{-1}\{X(s)\} = L^{-1}\left\{\frac{0.1}{s} - \frac{0.1}{s+50}\right\} \qquad x(t) = 0.1 - 0.1e^{-50t} = 0.1\left(1 - e^{-50t}\right) \quad \text{(Plot)}$$

Starts at the zero initial condition on $x(t)$, transient goes to zero by 0.15 *sec*, steady-state value is 0.1 (5/50). Verify initial and steady-state values using the Initial- and Final-Value theorems:

Initial Value Theorem $\lim\limits_{t\to 0} x(t) = \lim\limits_{s\to\infty} sX(s) = \lim\limits_{s\to\infty} s\left(\frac{5}{s(s+50)}\right) = 0$ Agrees!

Final Value Theorem $\lim\limits_{t\to\infty} x(t) = \lim\limits_{s\to 0} sX(s) = \lim\limits_{s\to 0} s\left(\frac{5}{s(s+50)}\right) = 0.1$ Agrees!

Time constant τ: $Ae^{-t/\tau}$ So in this example τ=1/50
After 3 time constants (0.06 *sec*), the first order system rises to 95% of its final value of 0.1 (see Figure 3.1):

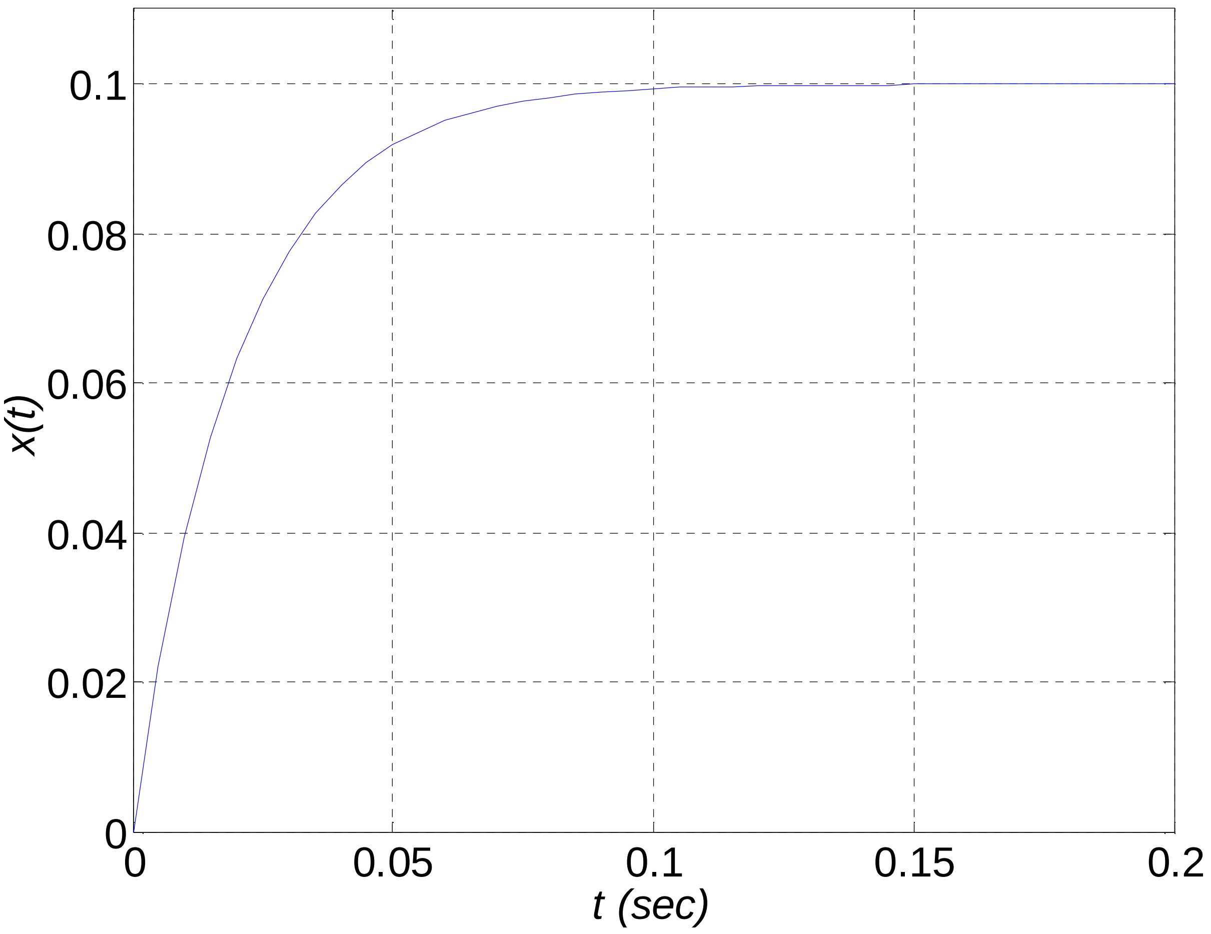

Figure 3.1 Single 1st-Order ODE Solution Plot

$\tau = \frac{c}{k}$ for translational mechanical; $\tau = \frac{J}{B}$ for rotational mechanical ; $\tau = \frac{L}{R}$ for series R-L circuit

General Solution Form to Scalar 1*st* order ODEs

Solve: $\dot{x}(t) = ax(t) + bu(t)$ (Form of matrix-vector equations $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$):
subject to input $u(t)$ and initial condition $x(0)$.

$$sX(s) - x(0) = aX(s) + bU(s)$$
$$(s-a)X(s) = x(0) + bU(s)$$
$$X(s) = \frac{x(0)}{(s-a)} + \frac{bU(s)}{(s-a)}$$

$$x(t) = L^{-1}\{X(s)\} = e^{at}x(0) + \int_0^t e^{a(t-\tau)} bu(\tau)d\tau$$

First part is homogeneous: transient response to initial conditions $x(0)$
Second part is particular: steady-state response to forcing function $u(t)$

Recall: $e^{at} = 1 + at + \frac{1}{2}a^2t^2 + \frac{1}{6}a^3t^3 + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!}a^k t^k$

2.1.2 Solution of Matrices of 1st-Order Differential Equations

Generalize Scalar Solution Form to Matrix of 1*st* order ODEs $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$

$$s\mathbf{X}(s) - \mathbf{X}(0) = \mathbf{AX}(s) + \mathbf{BU}(s)$$
$$[s\mathbf{I} - \mathbf{A}]\mathbf{X}(s) = \mathbf{X}(0) + \mathbf{BU}(s)$$
$$\mathbf{X}(s) = [s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{X}(0) + [s\mathbf{I} - \mathbf{A}]^{-1}\mathbf{BU}(s)$$

$$\mathbf{X}(t) = L^{-1}\{\mathbf{X}(s)\} = \mathbf{\Phi}(t)\mathbf{X}(0) + \int_0^t \mathbf{\Phi}(t-\tau)\mathbf{BU}(\tau)d\tau$$

First part is homogeneous: transient response to initial conditions $\mathbf{X}(0)$
Second part is particular: steady-state response to forcing functions $\mathbf{U}(t)$

$\Phi(t)$ State Transition Matrix

- Solution to homogeneous case, transient response to initial conditions with zero forcing

$$\dot{\mathbf{X}} = \mathbf{A}\mathbf{X}, \qquad \mathbf{X}(0) = \mathbf{X_0}, \qquad \mathbf{U}(t) = \{\mathbf{0}\}$$

- That is, for the homogeneous case, given any initial conditions, the future state vector at any time *t* is:

$$\mathbf{X}(t) = \mathbf{\Phi}(t)\,\mathbf{X_0}$$

- $\phi_{ij}(t)$ is the response of the i^{th} state variable due to an initial condition on the j^{th} state variable with zero initial conditions on all other state variables (linear superposition).

- $\mathbf{\Phi}(t) = L^{-1}\left\{[s\mathbf{I} - \mathbf{A}]^{-1}\right\}$ (see general solution form and last Laplace line above)

- Also, $\mathbf{\Phi}(t) = e^{\mathbf{A}t} = \mathbf{I} + \mathbf{A}t + \frac{1}{2}\mathbf{A}^2t^2 + \frac{1}{6}\mathbf{A}^3t^3 + \cdots = \sum_{k=0}^{\infty}\frac{1}{k!}\mathbf{A}^k t^k$. **A** is the System Dynamics Matrix, *t* is scalar time.

Properties of State Transition Matrix $\mathbf{\Phi}(t)$

1) $\mathbf{\Phi}(0) = \mathbf{I}$
2) $\mathbf{\Phi}^{-1}(t) = \mathbf{\Phi}(-t)$
3) $\mathbf{\Phi}(t_2 - t_1)\mathbf{\Phi}(t_1 - t_0) = \mathbf{\Phi}(t_2 - t_0)$
4) $\left[\mathbf{\Phi}(t)\right]^k = \mathbf{\Phi}(kt)$ For any positive integer *k*

Example: Linear SISO 2nd order System

Solve $\ddot{y}(t)+7\dot{y}(t)+12y(t)=u(t)$

Subject to $u(t)$ (step input of magnitude 3) and initial conditions $\begin{matrix} y(0)=0.10 \\ \dot{y}(0)=0.05 \end{matrix}$.

Characteristic polynomial: $s^2+7s+12=(s+3)(s+4)=0$ $\qquad s_{1,2}=-3,-4$

Distinct, negative real roots - overdamped system

By either the slow ME way or using Laplace transforms, we can find the solution to be:

$$y(t)=0.25-0.55e^{-3t}+0.40e^{-4t}$$

Let us derive this same solution from the state-space description. Define state vector:

$$\mathbf{X}(t)=\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}=\begin{Bmatrix} y(t) \\ \dot{y}(t) \end{Bmatrix}$$

Then the system dynamics differential equations, input, and initial state vector are:

$$\begin{Bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{Bmatrix}=\begin{bmatrix} 0 & 1 \\ -12 & -7 \end{bmatrix}\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}+\begin{bmatrix} 0 \\ 1 \end{bmatrix}\{u(t)\}$$

$u(t)$ is a step input of magnitude 3 and the initial conditions are $\mathbf{X}(0)=\{0.10 \quad 0.05\}^T$.

Solve in Laplace frequency domain

$$\mathbf{X}(s)=[s\mathbf{I}-\mathbf{A}]^{-1}\mathbf{X}(0)+[s\mathbf{I}-\mathbf{A}]^{-1}\mathbf{BU}(s) \qquad \text{then} \qquad \mathbf{X}(t)=L^{-1}\{\mathbf{X}(s)\}$$

$$[s\mathbf{I}-\mathbf{A}]=\begin{bmatrix} s & -1 \\ 12 & s+7 \end{bmatrix} \qquad [s\mathbf{I}-\mathbf{A}]^{-1}=\frac{1}{\Delta}\begin{bmatrix} s+7 & 1 \\ -12 & s \end{bmatrix}$$

Where $\qquad \Delta=s^2+7s+12=(s+3)(s+4) \qquad$ is $|s\mathbf{I}-\mathbf{A}|$, the characteristic polynomial

$$\mathbf{X}(s)=\frac{1}{\Delta}\begin{bmatrix} s+7 & 1 \\ -12 & s \end{bmatrix}\begin{Bmatrix} 0.10 \\ 0.05 \end{Bmatrix}+\frac{1}{\Delta}\begin{bmatrix} s+7 & 1 \\ -12 & s \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\left(\frac{3}{s}\right)$$

Where the Laplace transform of the unit step function is $\frac{1}{s}$

$$\mathbf{X}(s) = \frac{1}{\Delta}\begin{Bmatrix} 0.10s + 0.75 + \frac{3}{s} \\ 0.05s + 1.80 \end{Bmatrix} = \begin{Bmatrix} x_1(s) \\ x_2(s) \end{Bmatrix}$$

$$x_1(s) = \frac{0.10s + 0.75 + \frac{3}{s}}{(s+3)(s+4)} = \frac{0.10s^2 + 0.75s + 3}{s(s+3)(s+4)} = \frac{C_1}{s} + \frac{C_2}{(s+3)} + \frac{C_3}{(s+4)}$$ (partial fraction expansion)

$$0.10s^2 + 0.75s + 3 = C_1(s+3)(s+4) + C_2 s(s+4) + C_3 s(s+3)$$

Match like powers of *s*, then solve:

$$C_1 = 0.25$$
$$C_2 = -0.55$$
$$C_3 = 0.40$$

$$y(t) = x_1(t) = L^{-1}\{x_1(s)\} = L^{-1}\left\{\frac{0.25}{s} - \frac{0.55}{(s+3)} + \frac{0.40}{(s+4)}\right\} = 0.25 - 0.55e^{-3t} + 0.40e^{-4t}$$ Agrees!

Now find solution for second state variable $x_2(t)$:

$$x_2(s) = \frac{0.05s + 1.8}{(s+3)(s+4)} = \frac{C_1}{(s+3)} + \frac{C_2}{(s+4)}$$ (partial fraction expansion)

$$0.05s + 3 = C_1(s+4) + C_2(s+3)$$

Match like powers of *s*, then solve:

$$C_1 = 1.65$$
$$C_2 = -1.60$$

$$x_2(t) = L^{-1}\{x_2(s)\} = L^{-1}\left\{\frac{1.65}{(s+3)} - \frac{1.60}{(s+4)}\right\} = 1.65e^{-3t} - 1.60e^{-4t}$$

Check: $x_2(t) = \dot{x}_1(t)$??

$$\dot{x}_1(t) = -(-3)0.55e^{-3t} + (-4)0.40e^{-4t} = 1.65e^{-3t} - 1.60e^{-4t}$$ Agrees!

Figure 3.2 shows the state responses over time for this example.

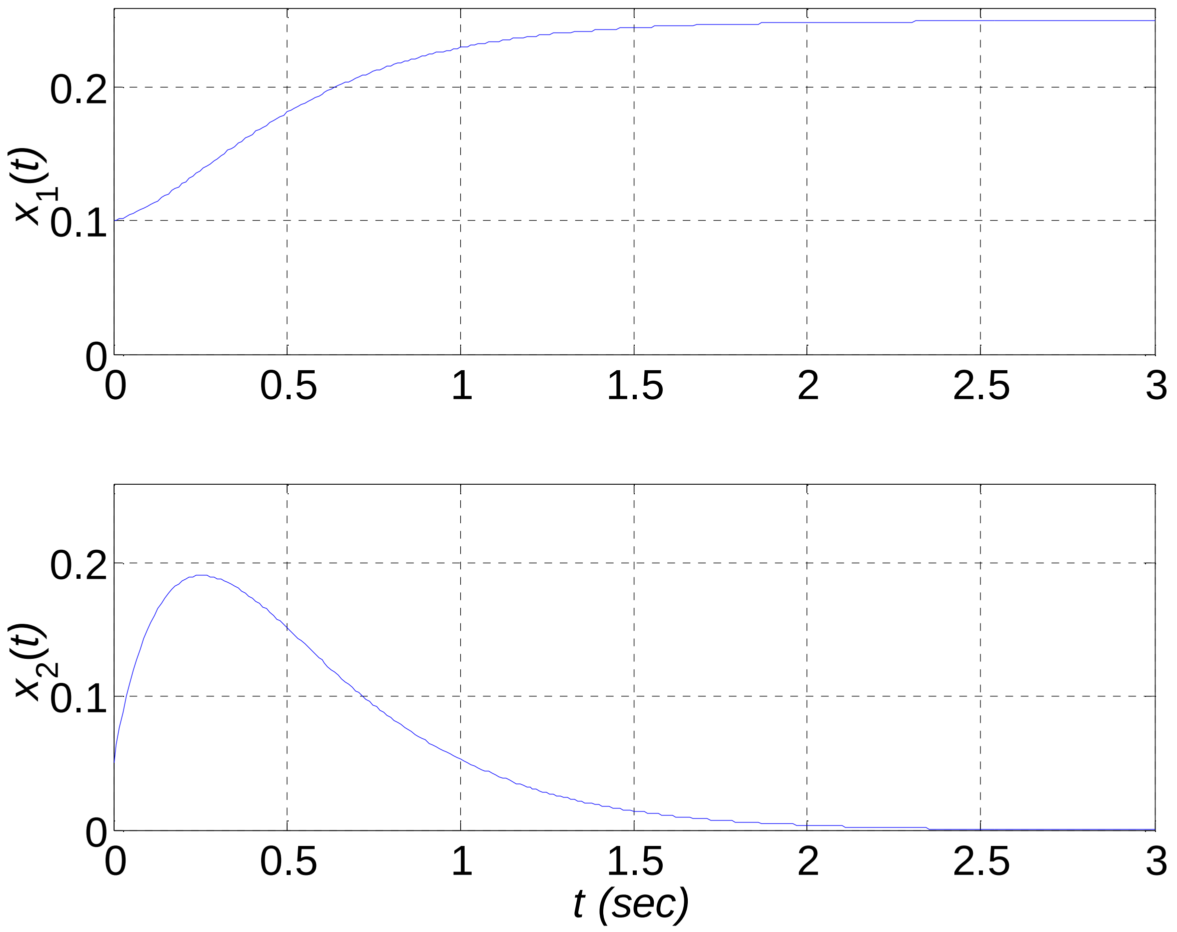

Figure 3.2 2nd-Order State Responses

We can see that the initial conditions $\mathbf{X}(0)=\{0.10 \quad 0.05\}^T$ are met and the steady state values of 0.25 for x_1 and 0 for x_2 (from the functions, with $t \to \infty$, or by using the Final Value Theorem for x_1 and x_2).

2.2 Similarity Transformations

Linear state vector coordinate transformation

$$\mathbf{X} = \mathbf{T}\mathbf{Z}$$

Where	$\mathbf{X}$	original state vector
	$\mathbf{Z}$	new state vector
	$\mathbf{T}$	non-singular transformation matrix

$$\mathbf{Z} = \mathbf{T}^{-1}\mathbf{X}$$

$$\dot{\mathbf{Z}} = \mathbf{T}^{-1}\dot{\mathbf{X}} = \mathbf{T}^{-1}\left(\mathbf{AX} + \mathbf{BU}\right)$$

State-space description in terms of new state vector **Z**:

$$\begin{aligned} \dot{\mathbf{Z}} &= \mathbf{T}^{-1}\mathbf{ATZ} + \mathbf{T}^{-1}\mathbf{BU} \\ \mathbf{Y} &= \mathbf{CTZ} + \mathbf{DU} \end{aligned} \qquad \begin{aligned} \dot{\mathbf{Z}} &= \overline{\mathbf{A}}\mathbf{Z} + \overline{\mathbf{B}}\mathbf{U} \\ \mathbf{Y} &= \overline{\mathbf{C}}\mathbf{Z} + \overline{\mathbf{D}}\mathbf{U} \end{aligned}$$

$$\begin{aligned} \overline{\mathbf{A}} &= \mathbf{T}^{-1}\mathbf{AT} \\ \overline{\mathbf{B}} &= \mathbf{T}^{-1}\mathbf{B} \\ \overline{\mathbf{C}} &= \mathbf{CT} \\ \overline{\mathbf{D}} &= \mathbf{D} \end{aligned}$$

This set of linear transformations is called a similarity transformation because new system has the same:

- characteristic equation
- eigenvalues
- transfer function
- but eigenvectors are different!

Proof:

$$\begin{aligned} \left|s\mathbf{I} - \overline{\mathbf{A}}\right| &= \left|s\mathbf{I} - \mathbf{T}^{-1}\mathbf{AT}\right| = \left|s\mathbf{T}^{-1}\mathbf{T} - \mathbf{T}^{-1}\mathbf{AT}\right| \\ &= \left|\mathbf{T}^{-1}\left[s\mathbf{I} - \mathbf{A}\right]\mathbf{T}\right| = \left|\mathbf{T}^{-1}\right|\left|s\mathbf{I} - \mathbf{A}\right|\left|\mathbf{T}\right| = \left|\mathbf{T}^{-1}\right|\left|\mathbf{T}\right|\left|s\mathbf{I} - \mathbf{A}\right| \end{aligned}$$

So $\left|s\mathbf{I} - \overline{\mathbf{A}}\right| = \left|s\mathbf{I} - \mathbf{A}\right|$

Therefore, $\overline{\mathbf{A}}$ and **A** have the same characteristic polynomial and eigenvalues.

2.3 Diagonal Canonical Form (DCF)

modal form, decoupled ODEs.

- solve n 1st order ODEs independently
- In SISO *DCF*, if all B elements are non-zero, system is completely controllable
- In SISO *OCF*, if all C elements are non-zero, system is completely observable

Any state-space realization **A**, **B**, **C**, **D** can be transformed to *DCF* by:

$$\mathbf{X} = \mathbf{TZ}, \qquad \text{where} \qquad \mathbf{T} = \begin{bmatrix} \mathbf{v_1} & \mathbf{v_2} & \mathbf{L} & \mathbf{v_n} \end{bmatrix} \quad (n \text{ x } n)$$

and $\mathbf{v}_i$ is the eigenvector of **A** associated with eigenvalue λ_i.

DCF:

$$\overline{\mathbf{A}} = \mathbf{T^{-1}AT} = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \lambda_n \end{bmatrix}$$

$$\overline{\mathbf{B}} = \mathbf{T^{-1}B}, \quad \overline{\mathbf{C}} = \mathbf{CT}, \quad \text{and} \quad \overline{\mathbf{D}} = \mathbf{D} \quad \text{have no particular form}$$

If **A** is *CCF* and **A** has distinct eigenvalues λ_i then **T** for *DCF* is the Vandermonde matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \cdots & \lambda_n \\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2 & \cdots & \lambda_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \lambda_3^{n-1} & 0 & \lambda_n^{n-1} \end{bmatrix}$$

***DCF* Example**   NEED A NEW DCF EXAMPLE. THIS IS FROM ANOTHER BOOK??!!??

Given the system from the *CCF* and *OCF* examples, calculate the *Observable Canonical Form*.

$$[A]=\begin{bmatrix}-1 & -2/3 & -3\\ 0 & 1 & 3\\ -1 & -5/3 & -3\end{bmatrix} \qquad [B]=\begin{bmatrix}1\\0\\1\end{bmatrix} \qquad [C]=\begin{bmatrix}1 & 1 & 1\end{bmatrix} \qquad [D]=[0]$$

$$T=\begin{bmatrix}v_1 & v_2 & v_3\end{bmatrix}=\begin{bmatrix}0.707 & -0.21+0.52i & -0.21-0.52i\\ -0.424 & -0.50-0.56i & -0.50+0.56i\\ 0.566 & 0.35+0.02i & 0.35-0.02i\end{bmatrix}$$

$$[\overline{A}]=T^{-1}AT=\begin{bmatrix}-3 & 0 & 0\\ 0 & i & 0\\ 0 & 0 & -i\end{bmatrix} \qquad [\overline{B}]=T^{-1}B=\begin{bmatrix}2.12\\ -0.24+0.58i\\ -0.24-0.58i\end{bmatrix}$$

$$[\overline{C}]=CT=\begin{bmatrix}0.85 & -0.35-0.02i & -0.35+0.02i\end{bmatrix} \qquad [\overline{D}]=[D]=[0]$$

$\overline{B}$ and $\overline{C}$ completely populated (non-zero).
Therefore, this system is completely controllable and observable. NOT INTRODUCED YET??

$$eig(A)=eig(\overline{A})=roots([3 \quad 1 \quad 3])=-3,\pm i$$

If start with *DCF*, $T=I$.

Same example, start with *CCF*:

$$A_{CCF}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -3 & -1 & -3\end{bmatrix} \qquad B_{CCF}=\begin{bmatrix}0\\0\\1\end{bmatrix} \qquad C_{CCF}=\begin{bmatrix}3 & 1 & 2\end{bmatrix} \qquad D_{CCF}=[0]$$

$$T=\begin{bmatrix}1 & 1 & 1\\ \lambda_1 & \lambda_2 & \lambda_3\\ \lambda_1^2 & \lambda_2^2 & \lambda_3^2\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\ i & -i & -3\\ -1 & -1 & 9\end{bmatrix}$$

$$[\overline{A}]=T^{-1}AT=\begin{bmatrix}i & 0 & 0\\ 0 & -i & 0\\ 0 & 0 & -3\end{bmatrix} \qquad [\overline{B}]=T^{-1}B=\begin{bmatrix}-0.05-0.15i\\ -0.05+0.15i\\ 0.1\end{bmatrix}$$

$$[\overline{C}]=CT=\begin{bmatrix}1+i & 1-i & 18\end{bmatrix} \qquad [\overline{D}]=[D]=[0]$$

2.4 Matlab

2.4.1 Matlab for Simulation of State-Space Systems

Some of the Matlab functions that are useful for simulation of state-space systems (to solve $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ given the input **U** and initial conditions **X**(0)) are:

eig(*A*)	Find the eigenvalues of **A**, which are the system poles.
roots(*den*)	Find the roots of the characteristic polynomial, which are the same as the system poles.
damp(*A*)	Calculate the 2nd-order system ξ and ω_n (for each mode if *n*>2), from the system dynamics matrix **A**.
damp(*den*)	Calculate the 2nd-order system ξ and ω_n (for each mode if *n*>2), from the coefficients den of system characteristic polynomial.
impulse(*A*,*B*,*C*,*D*) or *impulse*(*num*,*den*)	Determine numerically the unit impulse response for a system.
step(*A*,*B*,*C*,*D*) or *step*(*num*,*den*)	Determine numerically the unit step response for a system.
lsim(*A*,*B*,*C*,*D*,*U*,*t*,*X0*)	General linear simulation; calculate the output *Y* and state *X* given the state-space description *A*,*B*,*C*,*D*, inputs *U*, evenly-spaced time vector *t*, and initial conditions *X0*.
expm(*A*t*)	Evaluate the state transition matrix at time *t sec*.
plot(*x*,*y*)	Plot dependent variable *y* vs. independent variable *x*.

Continuing Matlab Example: State-Space Simulation

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), simulate the open-loop system response given zero input torque τ and initial state conditions $\mathbf{X}(0) = \{0.4 \quad 0.2\}^T$. The problem is: solve $\mathbf{X}(t)$ from $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ given the zero input **U** and the initial conditions $\mathbf{X}(0)$. Then find $\mathbf{Y}(t)$ from $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$. The following Matlab code, in combination with that in Chapter 2, performs the open-loop system simulation for this example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 3. Simulation of State-Space Systems
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

t  = [0:.01:4];                   % Define array of time values
U  = [zeros(size(t))];            % Define zero single input of proper size to go with t
X0 = [0.4;0.2];                   % Define vector of initial conditions [x1;x2]

PolesO    = eig(A);               % Calculate open-loop system poles
damp(A);                          % Determine zeta and wn from ABCD

[Yo,Xo] = lsim(A,B,C,D,U,t,X0);   % Open-loop response:  zero input torque, given ICs


Xo(101,:)                         %  State vector values at t=1 sec; compare with state transition matrix method
X5 = expm(A*1)*X0

figure;                           % Open-loop State Plots
subplot(211), plot(t,Xo(:,1)); grid; axis([0 4 -0.2 0.5]);
set(gca,'FontSize',18);
ylabel('{\itx}_1 (\itrad)')
subplot(212), plot(t,Xo(:,2)); grid; axis([0 4 -2 1]);
set(gca,'FontSize',18);
xlabel('\ittime (sec)'); ylabel('{\itx}_2 (\itrad/s)');
```

This m-file, combined with the m-file from Chapter 1, generated the following results for open-loop *poles*, ξ and ω_n, and the state vector values at 1 *sec* (both methods yielded the same state):

```
PolesO =
  -2.0000 + 6.0000i
  -2.0000 - 6.0000i

       Eigenvalue           Damping     Freq. (rad/s)
-2.00e+000 + 6.00e+000i     3.16e-001      6.32e+000
 -2.00e+000 - 6.00e+000i     3.16e-001      6.32e+000

X5 =
    0.0457
    0.1293
```

The m-file also generated the open-loop state plots of Figure 3.3.

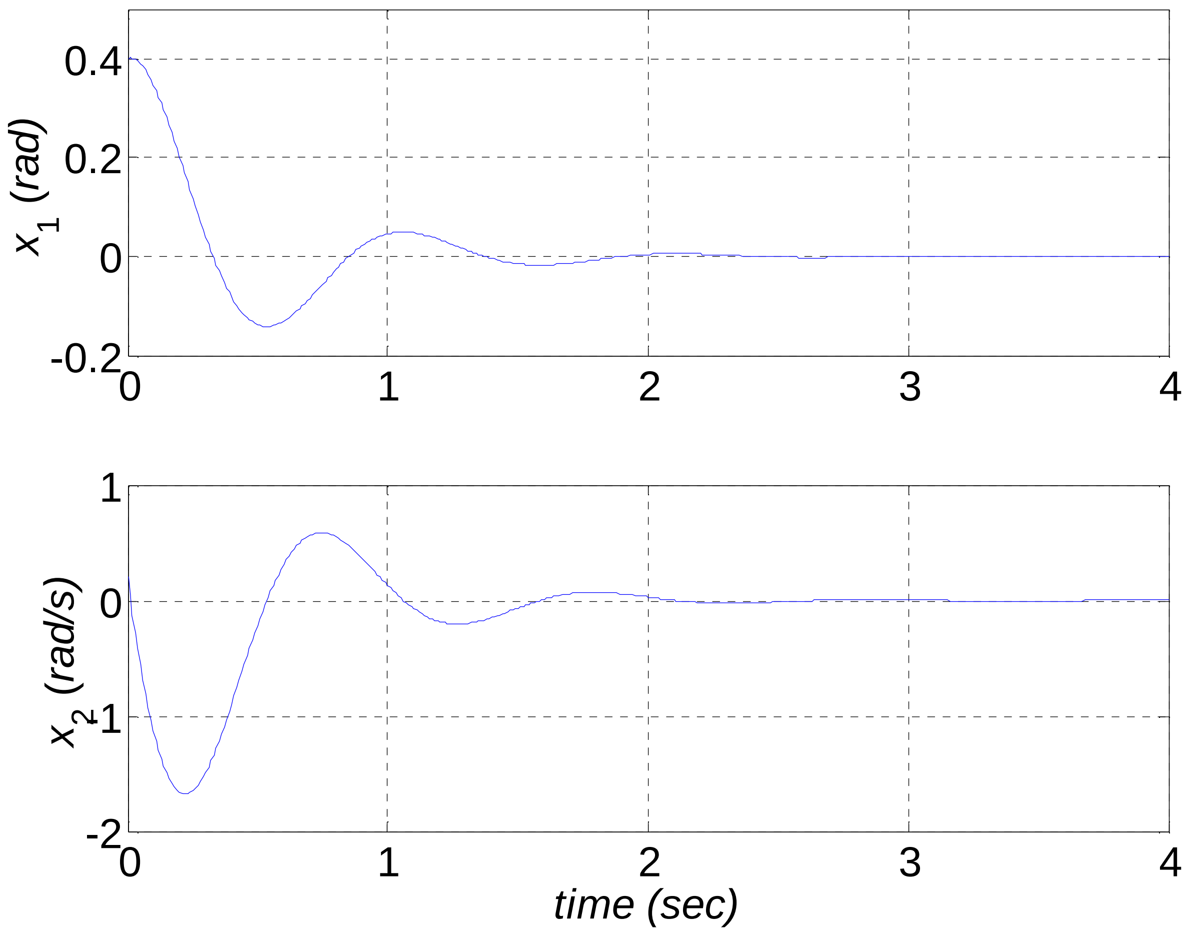

Figure 3.3 Open-Loop State Responses for Matlab Example

System simulation can also be performed using Matlab's Simulink. This is the graphical user interface to Matlab, and it is a fast and powerful tool. At the Matlab prompt simply enter *simulink* and the GUI opens, with menus for building system diagrams and then simulating them. Figures 3.4 show Simulink diagrams for this example; Figure 3.4a shows the high-level diagram and Figure 3.4b shows the detailed diagram for the ABCD Open-loop block mask, implementing $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ and $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$. This could be replaced by the built-in *simulink* ABCD block. To run these diagrams for this example, the step input was set to zero (torque) and the initial conditions were set to those given. The scope shows the output θ plot, identical to the upper plot of Figure 3.3.

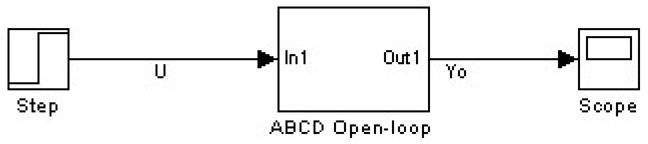

Figure 3.4a Simulink Diagram for Open-Loop Response

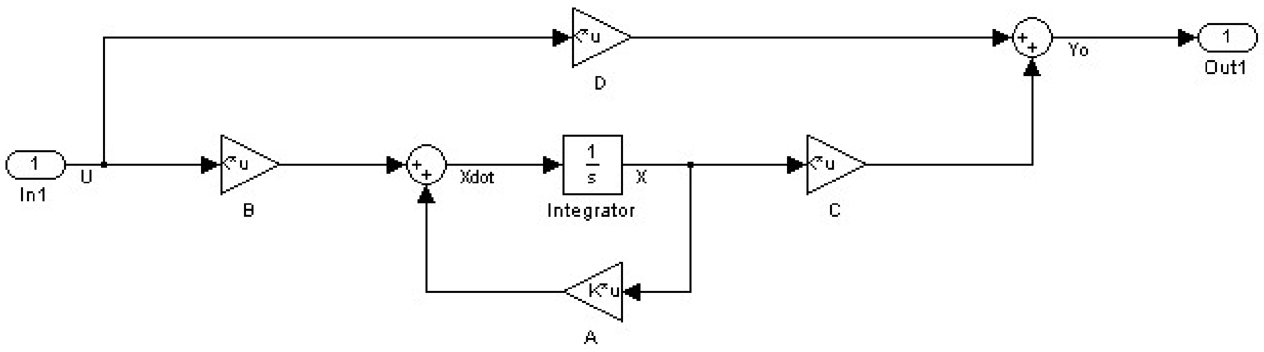

Figure 3.4b ABCD Open-loop Detailed Simulink Diagram

2.4.2 Matlab for Diagonal Canonical Form

The Matlab functions that are useful for similarity transformations and canonical realizations are:

canon	Matlab function for canonical forms (use 'modal' for DCF)
ss2ss	Similarity transformation of one state-space realization to another.

<u>**Continuing Matlab Example: Similarity Transformations and Canonical Realizations**</u>

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), determine the DCF canonical form for the given open-loop system. The following Matlab code, along with code from previous chapters, performs this determination.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 2.  Similarity Transformations and Diagonal Canonical Form
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Td,E] = eig(A);              % Transform to Diagonal canonical form (DCF) via formula
Ad = inv(Td)*A*Td;
Bd = inv(Td)*B;
Cd = C*Td;
Dd = D;

[Am,Bm,Cm,Dm,Tm] = canon(A,B,C,D,'modal');          % Determine DCF using Matlab function canon
```

This m-file, combined with the previous chapter m-files, yielded the following output:

```
Td =
  -0.0494 - 0.1482i  -0.0494 + 0.1482i
   0.9877             0.9877
Ad =
  -2.0000 + 6.0000i        0 - 0.0000i
   0.0000 - 0.0000i  -2.0000 - 6.0000i
Bd =
   0.5062 + 0.1687i
   0.5062 - 0.1687i
Cd =
  -0.0494 - 0.1482i  -0.0494 + 0.1482i
Dd =
     0
```

```
Tm =
         0    1.0124
   -6.7495   -0.3375

Am =
   -2.0000    6.0000
   -6.0000   -2.0000
Bm =
    1.0124
   -0.3375
Cm =
   -0.0494   -0.1482
Dm =
     0

TO =
     4     1
     1     0
AO =
     0   -40
     1    -4
BO =
     1
     0
CO =
     0     1
DO =
     0
```

The modal form using Matlab function *canon* does not agree with DCF from the formula since the formula allows imaginary numbers in the realizations (the open-loop system poles are complex conjugates $-2 \pm 6i$) and Matlab *canon* only allows real numbers. Both ‘diagonal’ forms are valid state-space realizations for this system.

2.5 Continuing Examples for Simulation and Similarity Transformations

2.5.1 Simulation of State-Space Systems

2.5.1.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

The following constant, lumped parameters are given for Continuing Example I (Two-mass MIMO Translational Mechanical System):

$m_1 = 40\ kg$	$c_1 = 20\ N\text{-}s/m$	$k_1 = 400\ N/m$
$m_2 = 20\ kg$	$c_2 = 10\ N\text{-}s/m$	$k_2 = 200\ N/m$

- For Case i, simulate the open-loop system response given zero initial conditions and step inputs of magnitudes 20 and 10 *N*, respectively, for u_1 and u_2.
- For Case ii, simulate the open-loop system response given zero input u_2 and initial conditions $\mathbf{X}(0)=\{0.1 \quad 0 \quad 0.2 \quad 0\}^T$ (initial displacements of 0.1 and 0.2 in y_1 and y_2, respectively, with zero initial velocities).

<u>Solution, Case i</u>

For Case i, the problem is: solve $\mathbf{X}(t)$ from $\dot{\mathbf{X}}=\mathbf{AX}+\mathbf{BU}$ given the input $\mathbf{U}$ and the zero initial conditions $\mathbf{X}(0)$. Then find $\mathbf{Y}(t)$ from $\mathbf{Y}=\mathbf{CX}+\mathbf{DU}$. The state-space matrices, with specific parameters from above, are:

$$\mathbf{A}=\begin{bmatrix} 0 & 1 & 0 & 0 \\ -15 & -0.75 & 5 & 0.25 \\ 0 & 0 & 0 & 1 \\ 10 & 0.5 & -10 & -0.5 \end{bmatrix} \quad \mathbf{B}=\begin{bmatrix} 0 & 0 \\ 0.025 & 0 \\ 0 & 0 \\ 0 & 0.05 \end{bmatrix} \quad \mathbf{C}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{D}=\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Using Matlab numerical simulation (function *lsim*), the plots for outputs y_1 and y_2 over time are given in Figure 3.5.

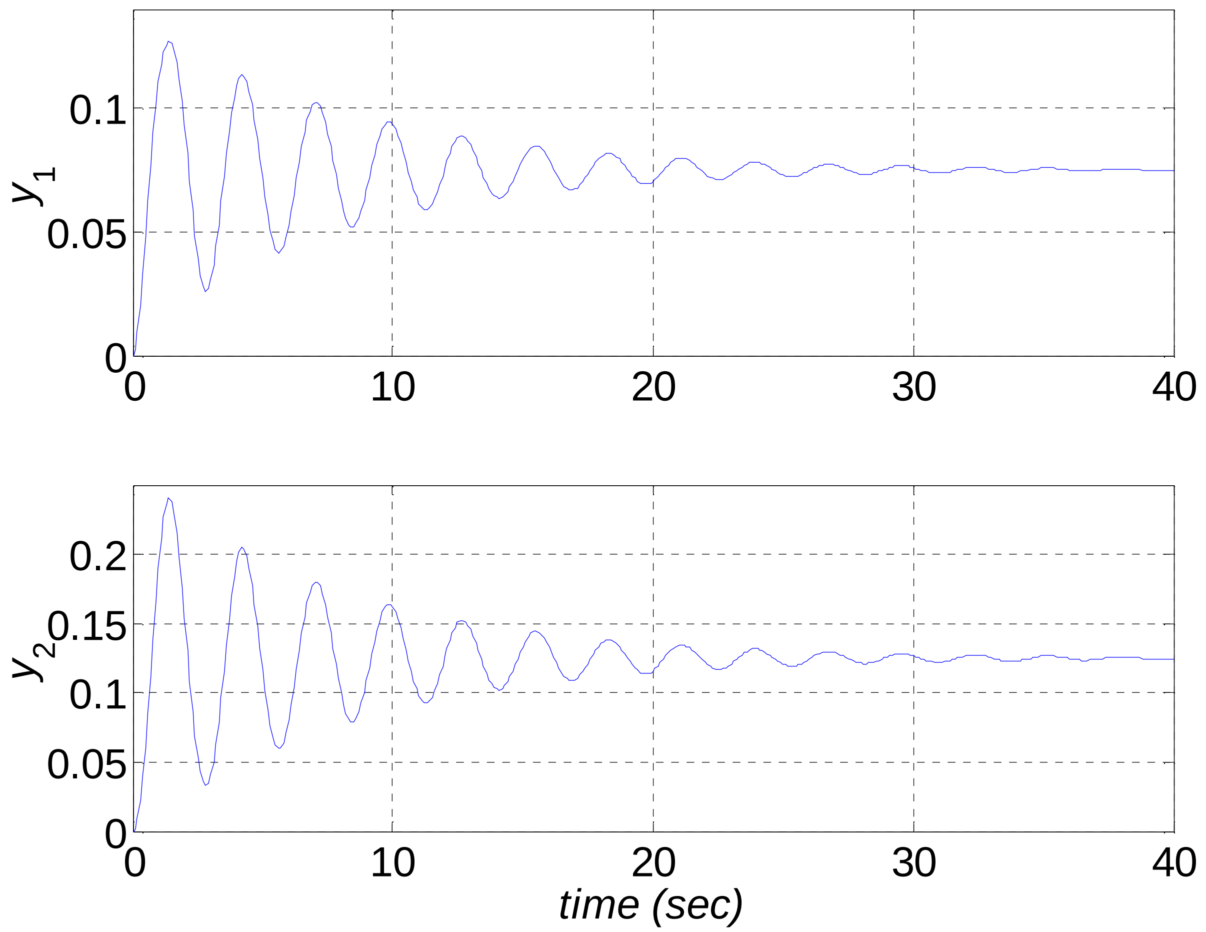

Figure 3.5 Open-Loop Response for Example I, Case i

We see from Figures 3.5 that this system is lightly damped; there are large percent overshoots and the masses are still vibrating past 40 seconds. The vibratory motion is a classical 2nd-order transient response, settling to final non-zero steady-state values due to the step inputs. The four open-loop system poles, found from the eigenvalues of **A**, are:

$$s_{1,2} = -0.5 \pm 4.44i \text{ and } s_{3,4} = -0.125 \pm 2.23i$$

Thus, this open-loop system is stable since all real parts of the four poles are strictly negative. The 4th-order system characteristic polynomial is:

$$\Delta(s) = s^4 + 1.25s^3 + 25.25s^2 + 10s + 100 = 0$$

This was found using the Matlab function poly(**A**); the roots of this polynomial are identical to the system poles. There are two modes of vibration in this 2-dof system; both are underdamped with $\xi_1 = 0.112$ and $\omega_{n1} = 4.48$ *rad/s* for $s_{1,2}$ and $\xi_2 = 0.056$ and $\omega_{n2} = 2.24$ *rad/s* for $s_{3,4}$. Note both modes contribute to both y_1 and y_2 in Figures 3.5. The steady state values are found from

$\mathbf{X_{ss}} = -\mathbf{A}^{-1}\mathbf{BU}$ for step inputs; due to the step inputs u_1 and u_2, y_1 and y_2 do not go back to zero in the steady-state, as the velocities do: $\mathbf{X_{ss}} = \mathbf{X}(\infty) = \{0.075 \quad 0 \quad 0.125 \quad 0\}^{\mathbf{T}}$.

Though we focus on state-space techniques, for completeness the matrix of transfer functions is given below for continuing Example I, Case i (found from Matlab function *ss2tf*):

$$\mathbf{Y}(s) = \mathbf{G}(s)\mathbf{U}(s)$$

$$\mathbf{G}(s) = \begin{bmatrix} \dfrac{0.025s^2 + 0.0125s + 0.25}{\Delta(s)} & \dfrac{0.0125s + 0.25}{\Delta(s)} \\ \dfrac{0.0125s + 0.25}{\Delta(s)} & \dfrac{0.05s^2 + 0.0375s + 0.75}{\Delta(s)} \end{bmatrix}$$

where the system characteristic polynomial is:

$$\Delta(s) = s^4 + 1.25s^3 + 25.25s^2 + 10s + 100 = 0$$

This is identical to the system characteristic polynomial derived from the **A** matrix and presented earlier. Note that the roots of the system characteristic polynomial are identical to the eigenvalues of **A** presented earlier.

Solution, Case ii

For Case ii, the problem is: solve $\mathbf{X}(t)$ from $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ given the input **U** (zero u_2) and the given initial conditions $\mathbf{X}(0)$. Then find $\mathbf{Y}(t)$ from $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$. The state-space matrices, with specific parameters from above, are (**A** is unchanged from Case i):

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.05 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = [0]$$

Using Matlab numerical simulation (function *lsim*, with initial conditions), the plots for states x_1 through x_4 over time are given in Figure 3.6.

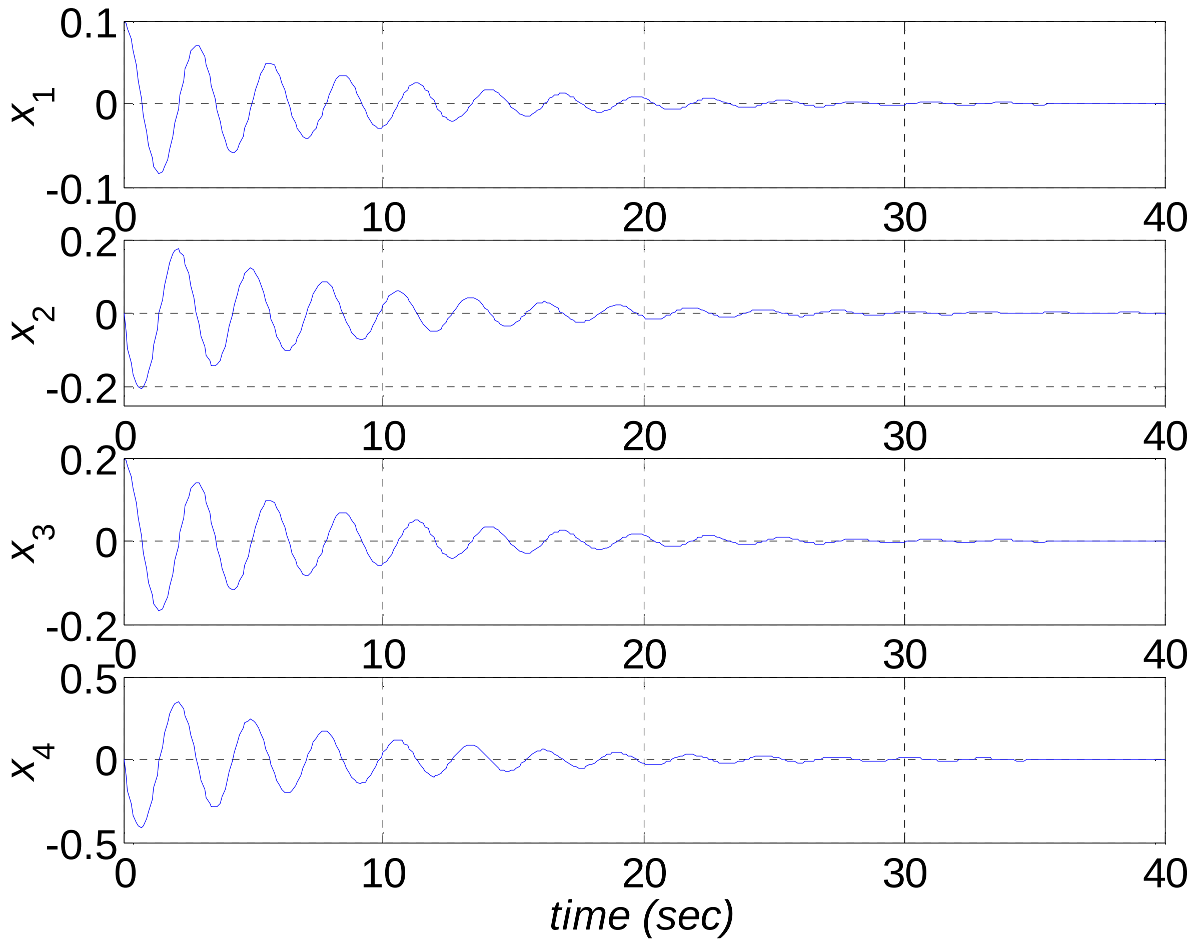

Figure 3.6 Open-Loop State Response for Example I, Case ii

We again see from Figures 3.6 that this system is lightly damped. The vibratory motion is again a classical 2nd-order transient response, to the initial displacement conditions, settling to zero final steady-state values due to the zero input u_2. The open-loop system characteristic polynomial, poles, vibration modes, and stability condition are all identical to the Case i example above.

In Figure 3.6, we see that states x_1 and x_3 start from the given initial values of 0.1 and 0.2, respectively; these are the initial displacements. The given initial velocities were both zero. Note that in this Case ii example, the final values are all zero since after the transient dynamics, the system returns to the equilibrium positions of both spring (because the initial conditions have damped out naturally and the input **U** is zero).

For such transient response problems based on given initial conditions, we can calculate the final state vector value at any desired time value by using the state transition matrix. For instance, at time t=20 *sec*.:

$$\mathbf{X}_{20} = \mathbf{\Phi}(20)\mathbf{X}(0) = \begin{Bmatrix} 0.0067 \\ -0.0114 \\ 0.0134 \\ -0.0228 \end{Bmatrix}$$

These values, though difficult to see at the scale of Figure 3.6 (at least the signs can be seen to be correct), agree perfectly with the Matlab data used in Figure 3.6, at *t*=20 *sec*.

Though we focus on state-space techniques, for completeness the matrix of transfer functions is given below for continuing Example I, Case i (found from Matlab function ss2tf):

$$G(s) = \frac{Y(s)}{U(s)} = \frac{0.0125s + 0.25}{\Delta(s)}$$

where the system characteristic polynomial is again:

$$\Delta(s) = s^4 + 1.25s^3 + 25.25s^2 + 10s + 100 = 0$$

Note that this scalar transfer function giving output y_1 from input u_2 is identical to the (1,2) element of the matrix of transfer functions presented for the full MIMO Case i. This makes sense because the (1,2) element refers to output y_1 caused by input u_2.

2.5.1.2 Continuing Example II: Rotational Electromechanical System

The following constant, lumped parameters are given for Continuing Example II (1-dof rotational electromechanical system; SISO: input *v*, output θ):

$L = 1$	$J = 1$	$k_T = 2$
$R = 2$	$b = 1$	

Simulate the open-loop system response given zero initial conditions and unit step input of voltage.

<u>Solution</u>

The problem is: solve $\mathbf{X}(t)$ from $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ given the input $\mathbf{U}$ and the initial conditions $\mathbf{X}(0)$. Then find $\mathbf{Y}(t)$ from $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$. The state-space matrices, with specific parameters from above, are:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -2 & -3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

Using Matlab numerical simulation (function *lsim*), the plots for the three states x_i over time are given in Figure 3.7.

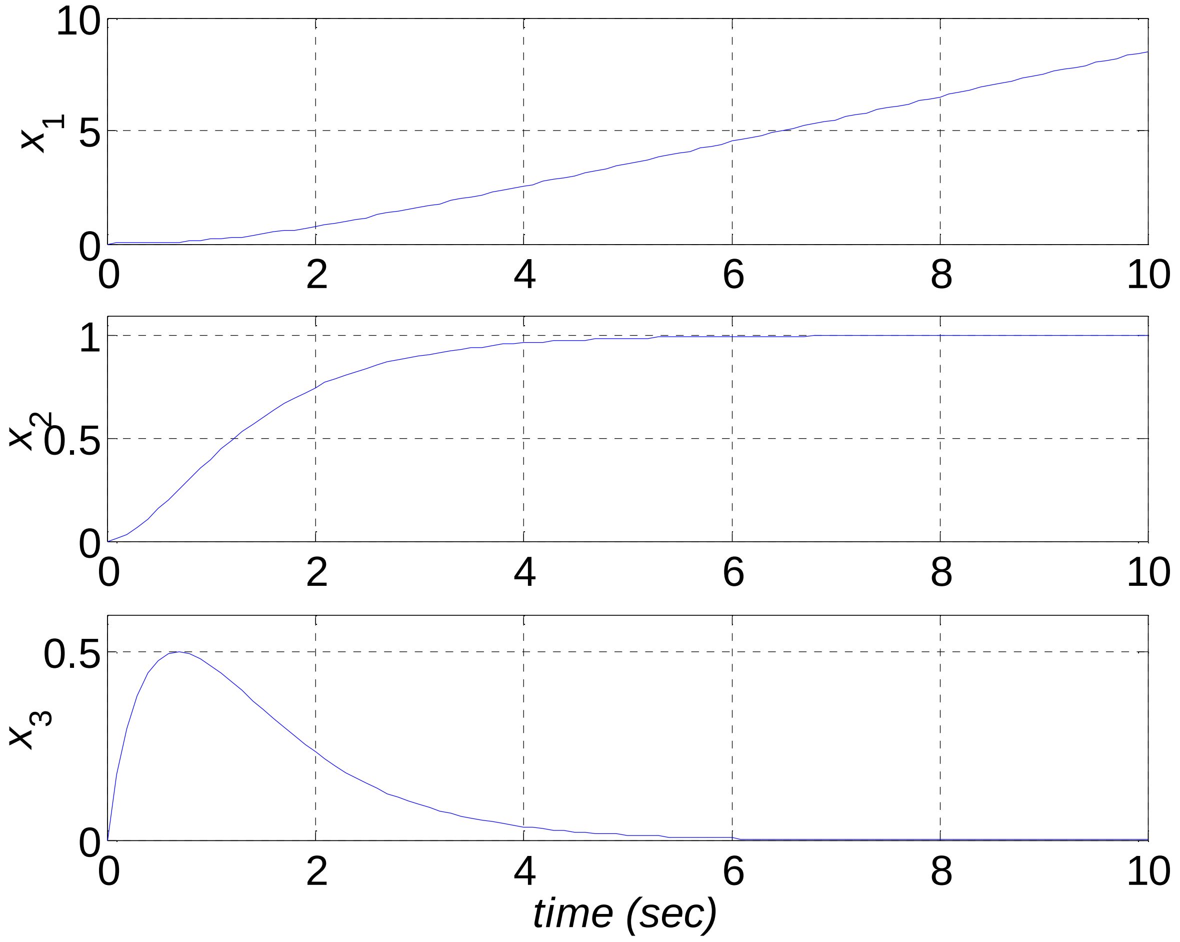

Figure 3.7 Open-Loop Response for Example II

We see from Figure 3.7 (top) that the motor shaft angle $\theta = x_1$ increases linearly in the steady state, after the transient dynamics response. This is desired; if a constant voltage is applied, the motor angle should continue to increase since there is no torsional spring. The steady-state slope of x_1 in Figure 3.7 is the steady-state value of $\omega = d\theta/dt = x_2$, 1 *rad/s*. This x_2 response is a classical 2nd-order overdamped response. The third state response, $\alpha = d\omega/dt = d^2\theta/dt^2 = x_3$, rapidly rises from its zero initial condition to a maximum of 0.5 rad/s^2; in the steady state, α is zero due to the constant angular velocity ω of the motor shaft. The three open-loop system poles, found from the eigenvalues of **A**, are:

$$s_{1,2,3} = 0, -1, -2$$

Thus, this open-loop system is marginally-stable since two of the real poles are negative, but the other is zero. The 3rd-order system characteristic polynomial is:

$$\Delta(s) = s^3 + 3s^2 + 2s = 0$$
$$\Delta(s) = s\left(s^2 + 3s + 2\right) = s(s+1)(s+2) = 0$$

This was found using the Matlab function poly(**A**); the roots of this polynomial are identical to the system poles. The zero pole corresponds to the rigid-body rotation of the motor shaft; the remaining two poles of –1,–2 led to the conclusion that the shaft angular velocity system ω is overdamped. Note that we cannot calculate steady state values from $\mathbf{X_{ss}} = -\mathbf{A}^{-1}\mathbf{BU}$ since the system dynamics matrix A is singular (its rank is 2 due to the column of zeros).

For completeness the scalar transfer function is given below for this example (found from Matlab function *ss2tf*):

$$G(s) = \frac{\theta(s)}{V(s)} = \frac{2}{s^3 + 3s^2 + 2s} = \frac{2}{s(s+1)(s+2)}$$

Note the same characteristic polynomial results from *ss2tf*. The roots of the system characteristic polynomial are the same as the eigenvalues of **A**. The above transfer function *G*(*s*) is for system output motor shaft angle θ given system input voltage *v*. If we wish to consider the motor shaft angular velocity ω as the output instead, we must differentiate θ, which is equivalent to multiplying by *s*, yielding the overdamped 2nd-order system discussed previously:

$$G_2(s) = \frac{\omega(s)}{V(s)} = \frac{2}{(s+1)(s+2)}$$

We could develop an associated 2nd-order state-space realization **A**, **B**, **C**, **D** if we wished to control ω rather than θ as the output:

$$x_1 = \omega$$
$$x_2 = \dot{\omega} = \dot{x}_1$$

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ \dfrac{-Rb}{LJ} & \dfrac{-(Lb + RJ)}{LJ} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \dfrac{k_T}{LJ} \end{bmatrix} \{v\} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} \{v\}$$

$$\{\omega\} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + [0]\{v\}$$

2.5.2 Similarity Transformations and Diagonal Canonical Form

2.5.2.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

Determine the diagonal canonical form (DCF) for Continuing Example I (Two-mass MIMO Translational Mechanical System), for Case ii (SISO, input u_2 and output y_1).

<u>Solution, Case ii</u>

If we allow imaginary numbers in our realization, the transformation matrix for **DCF** is the matrix of column-wise eigenvectors of **A**:

$$\mathbf{T}_{DCF} = \begin{bmatrix} 0.0173+0.1553i & 0.0173-0.1553i & -0.0102-0.1823i & -0.0102+0.1823i \\ -0.6901 & -0.6901 & 0.4082 & 0.4082 \\ -0.0173-0.1553i & -0.0173+0.1553i & -0.0204-0.3646i & -0.0204+0.3646i \\ 0.6901 & 0.6901 & 0.8165 & 0.8165 \end{bmatrix}$$

Using the similarity transformations, we find diagonal canonical form:

$$\mathbf{A_{DCF}} = \mathbf{T_{DCF}^{-1}AT_{DCF}} = \begin{bmatrix} -0.50+4.44i & 0 & 0 & 0 \\ 0 & -0.50-4.44i & 0 & 0 \\ 0 & 0 & -0.125+2.23i & 0 \\ 0 & 0 & 0 & -0.125-2.23i \end{bmatrix}$$

$$\mathbf{B_{DCF}} = \mathbf{T_{DCF}^{-1}B} = \begin{bmatrix} 0.0121+0.0014i \\ 0.0121-0.0014i \\ 0.0204+0.0011i \\ 0.0204-0.0011i \end{bmatrix}$$

$$\mathbf{C_{DCF}} = \mathbf{CT_{DCF}} = \begin{bmatrix} 0.0173+0.1553i & 0.0173-0.1553i & -0.0102-0.1823i & -0.0102+0.1823i \end{bmatrix}$$

$$\mathbf{D_{DCF}} = \mathbf{D} = [0]$$

Note that **DCF** is in the form expected, i.e. the poles of the system show up on the diagonal of diagonal matrix $\mathbf{A_{DCF}}$. Also, $\mathbf{B_{DCF}}$ and $\mathbf{C_{DCF}}$ are fully populated (no zero terms) which means that the system is fully state-controllable, and fully state-observable, respectively NOT YET DEFINED??.

The Matlab *canon* function with the switch ‘modal yields different results from these **DCF** results above, forcing only real numbers in the ‘diagonal’ form:

$$\mathbf{A_{DCF}} = \begin{bmatrix} -0.50 & 4.44 & 0 & 0 \\ -4.44 & -0.50 & 0 & 0 \\ 0 & 0 & -0.125 & 2.23 \\ 0 & 0 & -2.23 & -0.125 \end{bmatrix} \qquad \mathbf{B_{DCF}} = \begin{bmatrix} 0.0242 \\ -0.0027 \\ 0.0408 \\ -0.0023 \end{bmatrix}$$

$$\mathbf{C_{DCF}} = \begin{bmatrix} 0.0173 & 0.1553 & -0.0102 & -0.1823 \end{bmatrix}$$

EXPLAIN??!!??

2.5.2.2 Continuing Example II: Rotational Electromechanical System

Determine the diagonal canonical form (DCF) for Continuing Example II (1-dof rotational electromechanical system; SISO: input v, output θ).

Solution

The transformation matrix for **DCF** is the matrix of column-wise eigenvectors of **A**:

$$\mathbf{T}_{DCF} = \begin{bmatrix} 1 & -0.5774 & 0.2182 \\ 0 & 0.5774 & -0.4364 \\ 0 & -0.5774 & 0.8729 \end{bmatrix}$$

Using the similarity transformations, we find diagonal canonical form:

$$\mathbf{A_{DCF}} = \mathbf{T_{DCF}^{-1} A T_{DCF}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2 \end{bmatrix} \qquad \mathbf{B_{DCF}} = \mathbf{T_{DCF}^{-1} B} = \begin{bmatrix} 1 \\ 3.4641 \\ 4.5826 \end{bmatrix}$$

$$\mathbf{C_{DCF}} = \mathbf{C T_{DCF}} = \begin{bmatrix} 1 & -0.5774 & 0.2182 \end{bmatrix} \qquad \mathbf{D_{DCF}} = \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

Note that **DCF** is in the form expected, i.e. the poles of the system show up on the diagonal of diagonal matrix $\mathbf{A_{DCF}}$. Also, $\mathbf{B_{DCF}}$ and $\mathbf{C_{DCF}}$ are fully populated (no zero terms) which means that the system is fully state-controllable, and fully state-observable, respectively NOT YET DEFINED??.

The Matlab *canon* function with the switch ‘modal yields identical results to these **DCF** results since the system poles are real.

2.6 Homework Assignments

2.6.1 Mathematical Homework Assignments

2.6.2 Matlab Homework Assignments

2.6.3 Continuing Homework Assignments

CE1.2a
Use the following numerical parameters for this and all ensuing CE1 assignments (see Figure CE1).

Table 3.I Numerical Parameters for CE1 System

i	m_i (kg)	c_i (Ns/m)	k_i (N/m)
1	1	0.4	10
2	2	0.8	20
3	3	1.2	30
4		1.6	40

Simulate and plot the resulting open-loop output motion (displacements) for three cases (for this problem use the state-space realizations of CE1.2):

i. Full MIMO; three inputs, three outputs
   a. step inputs of magnitudes u_1=3, u_2=2, and u_3=1 (N). Zero initial conditions.
   b. Zero inputs. Initial displacements $y_1(t)=0.005$, $y_2(t)=0.010$, and $y_3(t)=0.015$ (m); zero initial velocities. Plot all six state components in this case, not just the three displacements.

ii. MIMO; two unit step inputs $u_1(t)$ and $u_3(t)$, three displacement outputs. Zero initial conditions.

iii. SISO; unit step input $u_2(t)$ and output $y_3(t)$. Zero initial conditions. Plot all six state components in this case, not just the one output displacement.

For each case, simulate long enough to demonstrate the steady-state behavior. For all plots, use the Matlab *subplot* function to keep each output on separate plots, aligned vertically with the same time range. What are the system poles? These define the nature of the system transient response. For Case i.b only, check your state vector results at t=10 *sec* using the state transition matrix.

One possible solution for CE1.1 (system dynamics matrix **A** only) is given below. This **A** matrix is the same for all input/output cases, while **B**, **C**, and **D** will change for different input/output cases. First, the state vector choices associated with this **A** are:

$$\begin{aligned} x_1 &= y_1 & x_3 &= y_2 & x_5 &= y_3 \\ x_2 &= \dot{y}_1 = \dot{x}_1 & x_4 &= \dot{y}_2 = \dot{x}_3 & x_6 &= \dot{y}_3 = \dot{x}_5 \end{aligned}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -\frac{(k_1+k_2)}{m_1} & -\frac{(c_1+c_2)}{m_1} & \frac{k_2}{m_1} & \frac{c_2}{m_1} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_2}{m_2} & \frac{c_2}{m_2} & -\frac{(k_2+k_3)}{m_2} & -\frac{(c_2+c_3)}{m_2} & \frac{k_3}{m_2} & \frac{c_3}{m_2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_3}{m_3} & \frac{c_3}{m_3} & -\frac{(k_3+k_4)}{m_3} & -\frac{(c_3+c_4)}{m_3} \end{bmatrix}$$

CE1.2b
For the CE1 system, Case iii only, calculate the *DCF* canonical realization. Comment on the structure of the results.

CE2.2a

Use the following numerical parameters for this and all ensuing CE2 assignments (see Figure CE2):

$m_1 = 2$, $m_2 = 1$ (kg), $L = 0.75$ (m), $g = 9.81$ (m/s^2)

Simulate and plot the resulting open-loop output motion (all four states, not just the output(s)) for three cases (for this problem use the state-space realizations of CE2.2); assume zero initial conditions for all cases (except Case i.b):

i. The classical SISO case; input $f(t)$ and output $\theta(t)$.
   a. unit impulse input $f(t)$ and zero initial conditions.
   b. zero input $f(t)$ and an initial condition of θ=0.1 *rad* (zero initial conditions on all other states).

ii. SIMO; unit impulse input $f(t)$, two outputs $y(t)$ and $\theta(t)$.

iii. MIMO; two unit step inputs $f(t)$ and $\tau(t)$ (add a torque motor as in CE2.2.iii), two outputs $y(t)$ and $\theta(t)$.

Simulate long enough to demonstrate the steady-state behavior. What are the system poles? Based on these poles and the physical system, explain the system responses.

One possible solution for CE2.2 (system dynamics matrix **A** only) is given below. This **A** matrix is the same for all input/output cases, while **B**, **C**, and **D** will change for different input/output cases. First, the state vector choices associated with this **A** are:

$$x_1 = y \qquad x_3 = \theta$$
$$x_2 = \dot{y} = \dot{x}_1 \qquad x_4 = \dot{\theta} = \dot{x}_3$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{m_2 g}{m_1} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{(m_1 + m_2) g}{m_1 L} & 0 \end{bmatrix}$$

CE2.2b

For the CE2 system, Case i only, **TRY** to calculate the *DCF* realizations (*DCF* cannot be found – why?).

CE3.2a

Use the following numerical parameters for this and all ensuing CE3 assignments (see Figure CE3):

L	= 0.0006	H	armature inductance
R	= 1.40	Ω	armature resistance
k_B	= 0.00867	$V/deg/s$	motor back emf constant
J_M	= 0.00844	$lb_f\text{-}in\text{-}s^2$	lumped motor shaft rotational inertia
b_M	= 0.00013	$lb_f\text{-}in/deg/s$	Motor shaft damping constant
k_T	= 4.375	$lb_f\text{-}in/A$	Torque constant
n	= 200	*unitless*	Gear ratio
J_L	= 1	$lb_f\text{-}in\text{-}s^2$	Load shaft polar inertia
b_L	= 0.5	$lb_f\text{-}in/deg/s$	Load shaft damping constant

Simulate and plot the resulting open-loop output motion (all three states, not just the output) for two cases (for this problem use the state-space realizations of CE3.2):

i. SISO; input armature voltage $v(t)$ and output robot load shaft angle $\theta_L(t)$.
   a. unit step input armature voltage $v(t)$; plot all three state responses. Zero initial conditions.
   b. zero input armature voltage $v(t)$; plot all three state responses. Initial conditions $\theta(0)=0, \omega(0)=\dot{\theta}(0)=1, \alpha(0)=\ddot{\theta}(0)=2$.

ii. SISO; unit step input armature voltage $v(t)$ and output robot load shaft angular velocity $\omega_L(t)$; plot both state responses. Zero initial conditions.

Simulate long enough to demonstrate the steady-state behavior. What are the system poles? Based on these poles and the physical system, explain the system responses.

One possible solution for CE3.2 (Case i) is given below. The state vector choices associated with the solution below are:

$$\begin{aligned} x_1 &= \theta_L \\ x_2 &= \dot{\theta}_L = \dot{x}_1 \\ x_3 &= \ddot{\theta}_L = \dot{x}_2 = \ddot{x}_1 \end{aligned}$$

The state differential ($\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$) and the output ($\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$) equations are:

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -\dfrac{(Rb+k_Tk_B)}{LJ} & -\dfrac{(Lb+RJ)}{LJ} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dfrac{k_T}{LJn} \end{bmatrix} \{V\} \qquad \{y\} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + [0]\{V\}$$

The solution for Case ii is similar, but of reduced (2nd) order:

$$x_1 = \omega_L$$
$$x_2 = \dot{\omega}_L = \dot{x}_1$$

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{(Rb + k_T k_B)}{LJ} & -\dfrac{(Lb + RJ)}{LJ} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 0 \\ \dfrac{k_T}{LJn} \end{bmatrix} \{V\} \qquad \{y\} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + [0]\{V\}$$

In the above equations (Case i and Case ii), the effective rotational inertia and rotational damping coefficient, reflected from the load to the motor shafts, are:

$$J = J_{eff} = J_M + \frac{J_L}{n^2}$$
$$b = b_{eff} = b_M + \frac{b_L}{n^2}$$

CE3.2b

For the CE3 system, Case i only, calculate the *DCF* canonical realization. Comment on the structure of the results.

3. Controllability

Conditions of *controllability* and *observability* govern the existence of a complete solution to the control system design problem in state space. Introduced by Kalman (filter guy) - father of linear state-space methods. He developed these concepts as the first step in complete control system design. Kalman - flunked prelims at MIT - ideas to far-fetched at the time (early 1960's), apparently. Developed his famous work at some small college.

- Controllable if all states x_i can be affected by at least one control u_j (actuators)

Although most physical systems are *controllable*, we must ensure that the corresponding mathematical models are also *controllable*.

$$\begin{aligned}\dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{BU}\\ \mathbf{Y} &= \mathbf{CX} + \mathbf{DU}\end{aligned}$$

3.1 Definition of Controllability

The continuous-time linear system

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$

is said to be completely state controllable at $t=t_0$ if there exists an unconstrained control input $\mathbf{U}(t)$ that will change an initial state $\mathbf{X}(t_0)$ to any final state $\mathbf{X}(t_1)$ in a finite time interval $t_0 \le t \le t_1$ (for all states). For example, we can force $\mathbf{X}(t_1) \to \{\mathbf{0}\}$ if desired, if the system is completely state controllable.

- Property of coupling between input and state, so it involves **A** and **B**.
- If the input vector has a connection to each state, system is completely controllable.
- If a system is completely controllable (and observable), we can design a linear state-feedback control law to arbitrarily place the closed-loop eigenvalues (poles) so that an unstable system is stabilized and the transient response can be changed.

3.2 Tests for Controllability

<u>Controllability Criterion</u>

Controllability Matrix
$$\mathbf{P} = \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \mathbf{A^2B} & \cdots & \mathbf{A^{n-1}B}\end{bmatrix}$$

If $rank(\mathbf{P}) = n$, the system is completely state controllable.

recall:	
r	number of inputs
m	number of outputs
n	number of states

$\mathbf{B}, \mathbf{AB}, \mathbf{A^2B}, \cdots$ each have dimension (n x r) so the dimension of **P** is (n x nr).

SISO Case: **b** is a column matrix $\mathbf{P} = \begin{bmatrix} \mathbf{b} & \mathbf{Ab} & \mathbf{A^2b} & \cdots & \mathbf{A^{n-1}b} \end{bmatrix}$, **P** is ($n$ x n) square matrix.

If $|\mathbf{P}| \neq 0$, the system is completely state controllable.

<u>Controllability Examples</u>

1) $G(s) = \dfrac{1}{s^3 + a_2 s^2 + a_1 s + a_0}$ $\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix}$ $\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{Ab} = \begin{bmatrix} 0 \\ 1 \\ -a_2 \end{bmatrix} \qquad \mathbf{A^2b} = \begin{bmatrix} 1 \\ -a_2 \\ a_2^2 - a_1 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{b} & \mathbf{Ab} & \mathbf{A^2b} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -a_2 \\ 1 & -a_2 & a_2^2 - a_1 \end{bmatrix}$$

$$|\mathbf{P}| = -1 \neq 0$$

so the system is completely state controllable.

2) NEED A NEW NON-CONTROLLABLE EXAMPLE. THIS IS FROM ANOTHER BOOK.

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} = \begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix} u$$

$$\mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad \mathbf{Ab} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 2 \\ -1 & -2 \end{bmatrix}$$

Clearly, $|\mathbf{P}| = 0$

so the system is NOT completely state controllable.

Why? $\dot{x}_1 + \dot{x}_2 = 2x_1 + u - x_1 + x_2 - u = x_1 + x_2$

States do not depend on u, so uncontrollable.

3.3 Coordinate Transformations and Controllability

3.4 Matlab for Controllability

Some Matlab functions that are useful for controllability determination are:

$P = ctrb(A,B)$	Calculate the controllability matrix associated with the system A, B.
$rank(M)$	Calculate the rank of matrix M.
$det(M)$	Calculate the determinant of square matrix M.
$size(A,1)$	Determine the system order n.

Continuing Matlab Example: Controllability

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), determine the controllability of the given open-loop system. The following Matlab code performs this determination for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 3.  Controllability
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

P = ctrb(A,B);                    % Calculate controllability matrix P
if (rank(P) == size(A,1))         % Logic to determine controllability
   disp('System is fully state-controllable');
else
   disp('System is NOT fully state-controllable');
end

P1 = [B A*B];                     % Check P via the formula
```

This m-file, combined with the m-file from Chapter 2, determined the controllability condition for the continuing example (the Matlab function yielded identical results to the formula):

```
P =
    0    1
    1   -4

System is fully state-controllable
```

3.5 Continuing Examples

3.5.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

Determine the controllability for Continuing Example I (Two-mass MIMO Translational Mechanical System), for both Case i (full MIMO) and Case ii (input u_2 and output y_1).

Solution, Case i

The 4x8 controllability matrix **P** is:

$$\mathbf{P} = \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \mathbf{A}^3\mathbf{B}\end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.03 & 0 & -0.02 & 0.01 & -0.36 & 0.23 \\ 0.03 & 0 & -0.02 & 0.01 & -0.36 & 0.23 & 0.67 & -0.61 \\ 0 & 0 & 0 & 0.05 & 0.01 & -0.03 & 0.23 & -0.48 \\ 0 & 0.05 & 0.01 & -0.03 & 0.23 & -0.48 & -0.61 & 0.73 \end{bmatrix}$$

This controllability matrix is of full rank, i.e. rank(**P**) = 4, which matches the system order n=4. Therefore, the system is fully state-controllable. This means we can proceed to design a full-state-feedback controller to place any desired poles into the closed-loop system.

Solution, Case ii

In Case ii, the system dynamics matrix **A** is identical to Case i; however, since **B** is different due to different input condition, we must again check for controllability.

The 4x4 controllability matrix **P** is:

$$\mathbf{P} = \begin{bmatrix}\mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} & \mathbf{A}^3\mathbf{B}\end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.01 & 0.23 \\ 0 & 0.01 & 0.23 & -0.61 \\ 0 & 0.05 & -0.03 & -0.48 \\ 0.05 & -0.03 & -0.48 & 0.73 \end{bmatrix}$$

This controllability matrix is of full rank, i.e. rank(**P**) = 4, which matches the system order n=4. Therefore, the system is fully state-controllable. This again means we can proceed to design a full-state-feedback controller to place any desired poles into the closed-loop system for this case.

3.5.2 Continuing Example II: Rotational Electromechanical System

Determine the controllability for Continuing Example II (1-dof rotational electromechanical system; SISO: input v, output θ).

Solution

The 3x3 controllability matrix $\mathbf{P}$ is:

$$\mathbf{P} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \mathbf{A}^2\mathbf{B} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 2 & -6 \\ 2 & -6 & 14 \end{bmatrix}$$

This controllability matrix is of full rank, i.e. rank($\mathbf{P}$) = 3, which matches the system order n=3. Therefore, the system is fully state-controllable. Another way to establish full rank for a square matrix is if the matrix determinant is non-zero: $|\mathbf{P}| = -8 \neq 0$. This means we can proceed to design a full-state-feedback controller to place any desired poles into the closed-loop system.

3.6 Homework Assignments

3.6.1 Mathematical Homework Assignments

3.6.2 Matlab Homework Assignments

3.6.3 Continuing Homework Assignments

CE1.3
For the CE1 system, determine if the system is completely state-controllable, for all three Cases (cases from CE1.2 and specific parameters from CE1.3). Give the mathematical details to justify your answers; explain your results in all cases by looking at the physical problem.

CE2.3
For the CE2 system, determine if the system is completely state-controllable, for all three Cases (cases from CE2.2 and specific parameters from CE2.3). Give the mathematical details to justify your answers; explain your results in all cases by looking at the physical problem.

CE3.3
For the CE3 system, determine if the system is completely state-controllable, for both Cases (cases from CE3.2 and specific parameters from CE3.3). Give the mathematical details to justify your answers; explain your results in all cases by looking at the physical problem.

4. Observability

4.1 Definition of Observability

The continuous-time linear system

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$
$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$$

is said to be completely observable if the all initial states $\mathbf{X}(t_0)$ can be determined from the observation of output $\mathbf{Y}(t)$ over a finite time interval $t_0 \le t \le t_1$ given $\mathbf{U}(t)$.

- Property of coupling between state and output, so it involves **A** and **C**.
- An observable system has an output that possesses a component due to each state variable.
- An observable system can estimate all state variables. A connection exists between each state variable and the output.

4.2 Tests for Observability

Observability Criterion

Observability Matrix
$$\mathbf{Q} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA^2} \\ \vdots \\ \mathbf{CA^{n-1}} \end{bmatrix}$$

If $rank(\mathbf{Q}) = n$, the system is completely observable.

recall:		
	r	number of inputs
	m	number of outputs
	n	number of states

$\mathbf{C}, \mathbf{CA}, \mathbf{CA^2}, \cdots$ each have dimension (m x n) so the dimension of **Q** is (nm x n).

SISO Case: **c** is a row matrix $\mathbf{Q} = \begin{bmatrix} \mathbf{c} \\ \mathbf{cA} \\ \mathbf{cA^2} \\ \vdots \\ \mathbf{cA^{n-1}} \end{bmatrix}$, **Q** is ($n$ x n) square matrix.

If $|\mathbf{Q}| \neq 0$, the system is completely observable.

Observability Examples

1) $G(s)=\dfrac{1}{s^3+a_2 s^2+a_1 s+a_0}$ $\qquad \mathbf{A}=\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \qquad \mathbf{c}=\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$

$$\mathbf{c}=\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$
$$\mathbf{cA}=\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$
$$\mathbf{cA^2}=\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Q}=\begin{bmatrix} \mathbf{c} \\ \mathbf{cA} \\ \mathbf{cA^2} \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Clearly, $|\mathbf{Q}|=1\neq 0$
so the system is completely observable.

2) NEED A NEW NON-OBSERVABLE EXAMPLE. THIS IS FROM ANOTHER BOOK.

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix}=\begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}+\begin{bmatrix} 1 \\ -1 \end{bmatrix}u \qquad y=\begin{bmatrix} 1 & 1 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}+\begin{bmatrix} 0 \end{bmatrix}u$$

$$\mathbf{c}=\begin{bmatrix} 1 & 1 \end{bmatrix}$$
$$\mathbf{cA}=\begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$\mathbf{Q}=\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Clearly, $|\mathbf{Q}|=0$
so the system is NOT completely observable.

Why? $\qquad y=x_1+x_2$ which depends on $x_1(0)$ and $x_2(0)$ so this does not allow us to determine $x_1(0)$ and $x_2(0)$ independently.

Examples summary:
1) Completely state controllable and observable so we *can* design a linear state-feedback controller with closed-loop poles as we specify; and we can design an associated observer.

2) Not state controllable or observable so we *cannot* design a linear state-feedback controller with closed-loop poles as we specify; nor can we design an associated observer.

4.3 Coordinate Transformations and Observability

4.4 Duality and Minimality

4.5 Matlab for Observability

Some Matlab functions that are useful for observability determination are:

$Q = obsv(A,C)$	Calculate the observability matrix associated with the system A, C.
$rank(M)$	Calculate the rank of matrix M.
$det(M)$	Calculate the determinant of square matrix M.
$size(A,1)$	Determine the system order n.

Continuing Matlab Example: Observability

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), determine the observability of the given open-loop system. The following Matlab code performs this determination for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 4.  Observability
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Q = obsv(A,C);                          % Calculate observability matrix P
if (rank(Q) == size(A,1))               % Logic to determine observability
   disp('System is fully state-observable');
else
   disp('System is NOT fully state-observable');
end

Q1 = [C; C*A];                          % Check Q via the formula
```

This m-file, combined with the m-file from Chapter 2, determined the observability condition for the continuing example (the Matlab function yielded identical results to the formula):

```
Q =
    1    0
    0    1

System is fully state-observable
```

4.6 Continuing Examples

4.6.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

Determine the observability for Continuing Example I (Two-mass MIMO Translational Mechanical System), for both Case i (full MIMO) and Case ii (input u_2 and output y_1).

Solution, Case i

The 8x4 observability matrix **Q** is:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \mathbf{CA}^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -15 & -0.75 & 5 & 0.25 \\ 10 & 0.5 & -10 & -0.5 \\ 13.75 & -14.31 & -6.25 & 4.69 \\ -12.5 & 9.38 & 7.50 & -9.63 \end{bmatrix}$$

This observability matrix is of full rank, i.e. rank(**Q**) = 4, which matches the system order n=4. Therefore, the system is fully state-observable. This means that we can proceed to design a full-state-feedback observer to estimate the states for feedback.

Solution, Case ii

In Case ii, the system dynamics matrix **A** is identical to Case i; however, since **C** is different due to different output condition, we must again check for observability.

The 4x4 observability matrix **Q** is:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \mathbf{CA}^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -15 & -0.75 & 5 & 0.25 \\ 13.75 & -14.31 & -6.25 & 4.69 \end{bmatrix}$$

This observability matrix is of full rank, i.e. rank(**Q**) = 4, which matches the system order n=4. Therefore, the system is fully state-observable. This again means that we can proceed to design a full-state-feedback observer to estimate the states for feedback for this case.

4.6.2 Continuing Example II: Rotational Electromechanical System

Determine the observability for Continuing Example II (1-dof rotational electromechanical system; SISO: input v, output θ).

Solution

The 3x3 observability matrix $\mathbf{Q}$ is $\mathbf{I_3}$:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

clearly rank($\mathbf{Q}$) = 3 = n; also $|\mathbf{Q}| = 1 \neq 0$. Therefore, this system is fully state-observable. This means that we can proceed to design a full-state-feedback observer to estimate the states for feedback.

4.7 Homework Assignments

4.7.1 Mathematical Homework Assignments

4.7.2 Matlab Homework Assignments

4.7.3 Continuing Homework Assignments

CE1.4
For the CE1 system, determine if the system is completely observable, for all three Cases (cases from CE1.2 and specific parameters from CE1.3). Give the mathematical details to justify your answers; explain your results in all cases by looking at the physical problem.

CE2.4
For the CE2 system, determine if the system is completely observable, for all three Cases (cases from CE2.2 and specific parameters from CE2.3). Give the mathematical details to justify your answers; explain your results in all cases by looking at the physical problem.

CE3.4
For the CE3 system, determine if the system is completely observable, for both Cases (cases from CE3.2 and specific parameters from CE3.3). Give the mathematical details to justify your answers; explain your results in all cases by looking at the physical problem.

5. Stability

5.1 Definition and Eigenvalue Analysis

A system is stable if the output is bounded for all bounded inputs. Stability is property of a system, independent of input signal.

Equilibrium states can be unstable equilibrium (point *a*), neutral equilibrium (region *b*), or stable equilibrium (point *c*); this is demonstrated in the diagram of Figure 7.1.

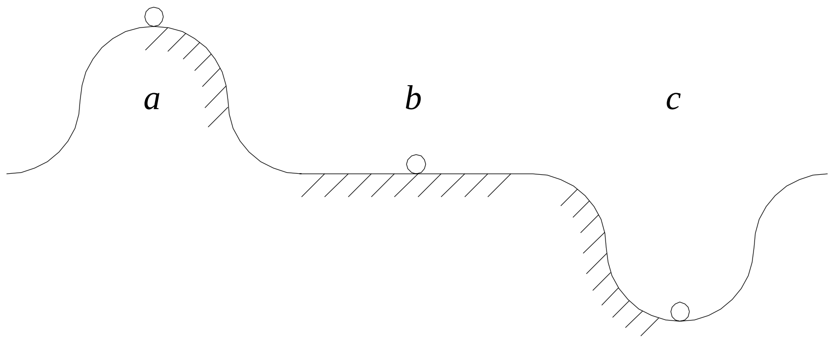

Figure 7.1 Equilibrium States

Simple test for system stability: The real part of all poles must be negative. Poles are eigenvalues of system dynamics matrix **A**.

Characteristic equation $\left|s\mathbf{I}-\mathbf{A}\right|=0$ Poles, eigenvalues: $s_j=\beta_j \pm i\omega_j$

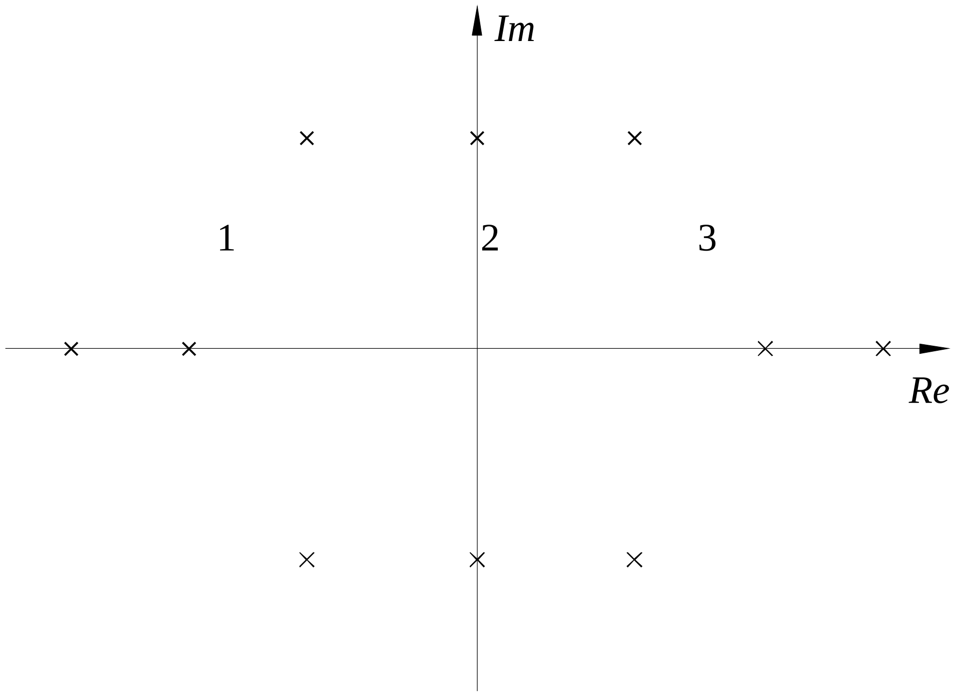

Figure 7.2 Re-Im Pole Map

Transient solution form: $y(t) = Ce^{\beta t}\cos(\omega t + \phi)$

1) If all	$\beta_j < 0$	stable
2) If any	$\beta_j = 0$	marginally stable (assuming the remaining β_j are negative)
3) If any	$\beta_j > 0$	unstable

Classical controls:

- **Routh-Hurwitz criterion** - determine stability based on transfer function coefficients without actually calculating poles.
- **Root-locus method** - graphical method to vary feedback gain *k* to determine ranges for stability and control transient response.

5.2 Lyapunov Stability Analysis

5.2.1 Stability Analysis Based on System Energy

Applies to linear/non-linear and constant/time varying
Can determine stability without solving $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$ (or non-linear system equations).

Based on energy method. E is the total system energy and an equilibrium point is X={0}:

If $\frac{dE}{dt} < 0$ then $\mathbf{X} \to \{\mathbf{0}\}$ and the system is *stable*.
If $\frac{dE}{dt} = 0$ then the system is *marginally stable*.
If $\frac{dE}{dt} > 0$ then the system is *unstable* because something is continuously adding energy.

<u>Example</u>

SISO 1-dof *m*-*c*-*k* linear translational mechanical system.

$$E = \frac{1}{2}m\dot{y}^2 + \frac{1}{2}ky^2$$

$$\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} y \\ \dot{y} \end{Bmatrix}$$

$$E = \frac{1}{2}m\dot{y}^2 + \frac{1}{2}ky^2 = \frac{1}{2}mx_2^2 + \frac{1}{2}kx_1^2$$

If $\frac{dE}{dt} = mx_2\dot{x}_2 + kx_1\dot{x}_1 < 0$, then the system is stable.

This is always true for positive damping.

Stability analysis via phase plots - plot velocity vs. displacement:
- Increasing or decreasing energy
- Stable if orbit converges to a point (constant x_1, x_2=0)
- Show examples, +,0,- damping??!!??
- Initial conditions

5.2.2 Lyapunov Stability Background

Definitions:

Any time varying nonlinear system can be represented as: $\dot{\mathbf{X}} = \mathbf{f}\left(\mathbf{X}, t\right)$

A state is an equilibrium state $\mathbf{X_e}$ if $\mathbf{f}\left(\mathbf{X_e}, t\right) = \mathbf{0}$ for all *t*.

For linear time invariant systems, $\dot{\mathbf{X}} = \mathbf{f}\left(\mathbf{X}, t\right) = \mathbf{AX}$ and there is one unique equilibrium state $\mathbf{X_e}$ if **A** is nonsingular; infinitely many equilibrium states $\mathbf{X_e}$ if **A** is singular.

We can always shift an equilibrium state $\mathbf{X_e}$ to zero by coordinate shifts: $\mathbf{f}\left(\mathbf{0}, t\right) = \mathbf{0}$ for all *t*.

Hyperspherical region of radius *k* about an equilibrium state $\mathbf{X_e}$ (using Euclidean norm):

$\left\|\mathbf{X} - \mathbf{X_e}\right\| \le k$ where $\left\|\mathbf{X} - \mathbf{X_e}\right\| = \left[\left(x_1 - x_{1e}\right)^2 + \left(x_2 - x_{2e}\right)^2 + \cdots + \left(x_n - x_{ne}\right)^2\right]^{1/2}$

Define two such spherical regions: $\left\|\mathbf{X} - \mathbf{X_e}\right\| \le \delta$ and $\left\|\mathbf{X} - \mathbf{X_e}\right\| \le \varepsilon$, with $\delta < \varepsilon$. Graphical representation:

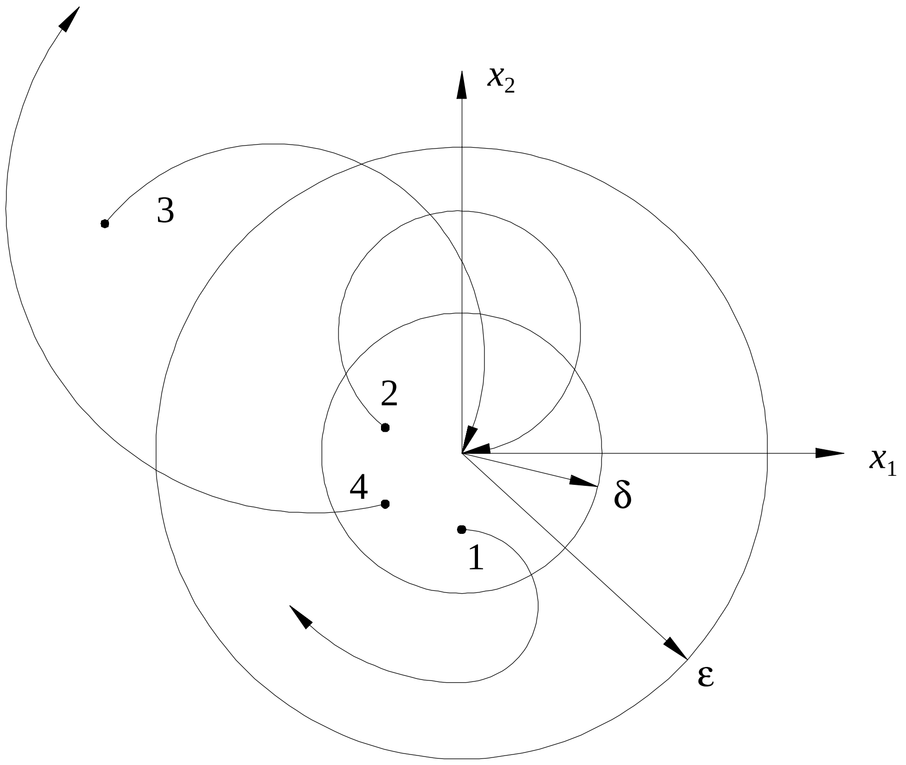

Figure 7.3 Four Lyapunov Stability Conditions

1. An equilibrium state $\mathbf{X_e}$ is said to be *stable in the sense of Lyapunov* (stability *I.S.L.*) if trajectories starting within δ do not leave the ε region as *t* increases indefinitely.

2. An equilibrium state $\mathbf{X_e}$ is said to be *asymptotically stable* if trajectories starting within δ converge to $\mathbf{X_e}$ without leaving the ε region as *t* increases indefinitely. This case is preferable to *stability I.S.L.*

3. An equilibrium state $\mathbf{X_e}$ is said to be *asymptotically stable in the large* asymptotic stability holds for all possible initial states $\mathbf{X_0}$. There must be only one equilibrium state in the whole state space.

4. An equilibrium state $\mathbf{X_e}$ is said to be *unstable* if trajectories starting within δ leaves the ε region as *t* increases.

Stability types:

- Stability in the sense of Lyapunov (*stability I.S.L.*)
- Asymptotic stability
- Bounded input, Bounded state Stability (*BIBS*)
- Bounded input, Bounded output Stability (*BIBO*)

5.2.3 Lyapunov Stability Analysis

Given system $\dot{\mathbf{X}} = \mathbf{AX}$, assume the state vector origin is the equilibrium state $\mathbf{X_e}$:

$$\mathbf{X_e} = \mathbf{0}\text{, or } \mathbf{AX_e} = \mathbf{0}$$

Second method of Lyapunov (1892, Russian):

If a positive-definite function $V(\mathbf{X})$ can be found such that $\dot{V}(\mathbf{X})$ is negative-definite, this equilibrium state is asymptotically stable.

$V(\mathbf{X})$ Lyapunov function: generalized energy function, not unique

$V(\mathbf{X})$ is	positive-definite if	$V(\mathbf{X}) > 0$	for all $\mathbf{X}$
	positive-semi-definite if	$V(\mathbf{X}) \geq 0$	
	negative-definite if	$V(\mathbf{X}) < 0$	

Quadratic form $\mathbf{X^T P X}$ scalar function, $\mathbf{P}$ is real and symmetric.

This form is positive-definite if $\mathbf{P}$ is positive-definite.

Positive-definite matrix: *Sylvester's criterion*:

P is positive-definite if all principal minors are positive. Principal minors are submatrix determinants starting with scalar p_{11} and proceeding (with p_{11} included as the first term in each) until the determinant of the entire **P**.

P is positive-semi-definite if all principal minors are non-negative.

5.2.4 Lyapunov's Direct Method

For linear time-invariant systems, we must find a positive-definite quadratic scalar Lyapunov function $V(\mathbf{X}) = \mathbf{X}^{\mathbf{T}}\mathbf{P}\mathbf{X}$. With **P** real, symmetric, and positive-definite, $V(\mathbf{X})$ is positive-definite.

If $\dot{V}(\mathbf{X}) < 0$ for all *t* (negative-definite), then the system is asymptotically stable.

$$V(\mathbf{X}) = \mathbf{X}^{\mathbf{T}}\mathbf{P}\mathbf{X}$$

$$\dot{V}(\mathbf{X}) = \dot{\mathbf{X}}^{\mathbf{T}}\mathbf{P}\mathbf{X} + \mathbf{X}^{\mathbf{T}}\dot{\mathbf{P}}\mathbf{X} + \mathbf{X}^{\mathbf{T}}\mathbf{P}\dot{\mathbf{X}}$$

If **A** is constant, **P** is constant:

$\dot{\mathbf{P}} = \mathbf{0}$ and using $\dot{\mathbf{X}} = \mathbf{A}\mathbf{X}$,

$$\dot{\mathbf{V}}(\mathbf{X}) = \dot{\mathbf{X}}^{\mathbf{T}}\mathbf{P}\mathbf{X} + \mathbf{X}^{\mathbf{T}}\mathbf{P}\dot{\mathbf{X}} = \mathbf{X}^{\mathbf{T}}\mathbf{A}^{\mathbf{T}}\mathbf{P}\mathbf{X} + \mathbf{X}^{\mathbf{T}}\mathbf{P}\mathbf{A}\mathbf{X} = \mathbf{X}^{\mathbf{T}}\left[\mathbf{A}^{\mathbf{T}}\mathbf{P} + \mathbf{P}\mathbf{A}\right]\mathbf{X}$$

5.2.5 Lyapunov Equation

$\dot{V}(\mathbf{X})$ must be negative-definite, so the matrix in the quadratic form:

$\dot{V}(\mathbf{X}) = \mathbf{X^T}\left[\mathbf{A^T P + PA}\right]\mathbf{X}$ must be negative-definite:

$\mathbf{A^T P + PA = -Q}$ for some positive-definite **Q**.

Starting with an arbitrary positive-definite **Q** yields a unique **P**; however,

Starting with an arbitrary positive-definite **P** may not yield a unique **Q**.

Solve $\mathbf{A^T P + PA = -Q}$

Given **P**, solve **Q** is easy, but may not work this way.

<u>Necessary and sufficient condition:</u>
System represented by dynamics matrix **A** is asymptotically stable if and only if the solution **P** is positive-definite when **Q** is positive-definite. For non-singular constant **A**, equilibrium state $\mathbf{X_e = 0}$ is *asymptotically stable in the large*.

5.2.6 Lyapunov Stability Analysis Example

Determine the stability condition for $\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$

$\mathbf{A^TP + PA = -Q}$; Let $\mathbf{Q} = \mathbf{I}_2$ (positive-definite matrix)

Solve for symmetric $\mathbf{P} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$ and determine its definiteness.

Since **P** is symmetric, we don't have n^2 equations, we have $\dfrac{(n^2+n)}{2}$ equations

$$\begin{bmatrix} 0 & -2 \\ 1 & -3 \end{bmatrix}\begin{bmatrix} a & b \\ b & c \end{bmatrix} + \begin{bmatrix} a & b \\ b & c \end{bmatrix}\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} -2b & -2c \\ a-3b & b-3c \end{bmatrix} + \begin{bmatrix} -2b & a-3b \\ -2c & b-3c \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\begin{bmatrix} -4b & a-3b-2c \\ a-3b-2c & 2b-6c \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

due to symmetry, use either 2,1 or 1,2 (same equations).

$$\begin{bmatrix} 0 & -4 & 0 \\ 1 & -3 & -2 \\ 0 & 2 & -6 \end{bmatrix}\begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{Bmatrix} -1 \\ 0 \\ -1 \end{Bmatrix}$$

actually, these linear equations are decoupled

$$\begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{Bmatrix} 1.25 \\ 0.25 \\ 0.25 \end{Bmatrix} \quad \text{so} \quad \mathbf{P} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} 1.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

Sylvester's criterion for positive definiteness:

check $|a| = +1.25$ and $\begin{vmatrix} a & b \\ b & c \end{vmatrix} = +0.25$

Both principal minors are positive so **P** is positive-definite. Therefore, the linear system represented by **A** is *asymptotically stable in the large*.

Note: eig(**A**) = [-1,-2] All real parts are strictly negative - agrees with the above result.

5.3 Input/Output Stability

5.4 Matlab for Stability Analysis

The following Matlab functions are useful for Lyapunov stability analysis:

lyap(*A'*,*QQ*)	Solve $\mathbf{A}^{\mathbf{T}}\mathbf{PP} + \mathbf{PPA} = -\mathbf{QQ}$ for matrix **PP** given positive-definite matrix **QQ**.

Continuing Matlab Example: Lyapunov Stability Analysis

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), determine the stability condition of the given open-loop system via Lyapunov stability analysis. The following Matlab code performs this determination for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 7.  Lyapunov Stability Analysis
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

QQ = eye(2);                % Given positive definite matrix
PP = lyap(A',QQ);           % Solve for PP

pm1 = PP(1,1);              % Sylvester's method to see if PP is positive definite; principal minors of PP
pm2 = PP(1:2,1:2);
if (det(pm1)>0 & det(pm2)>0)      % Logic to determine stability condition
   disp('System is asymptotically stable');
elseif (det(pm1)==0 | det(pm2)==0)
   disp('System is marginally stable');
else
   disp('System is unstable');
end

figure;                     % Plot phase portraits to enforce stability analysis
plot(Xo(:,1),Xo(:,2)); grid; axis('square'); axis([-1.3 1.0 -1.7 0.6]);
set(gca,'FontSize',18);
xlabel('{\itx}_1 (rad)'); ylabel('{\itx}_2 (rad/s)');
```

This m-file, combined with the previous chapter m-files, yielded the following output, plus the phase portrait plot of Figure 7.4.

```
PP =
   5.1750    0.0125
   0.0125    0.1281

pm1 =
   5.1750

pm2 =
   5.1750    0.0125
   0.0125    0.1281

System is asymptotically stable
```

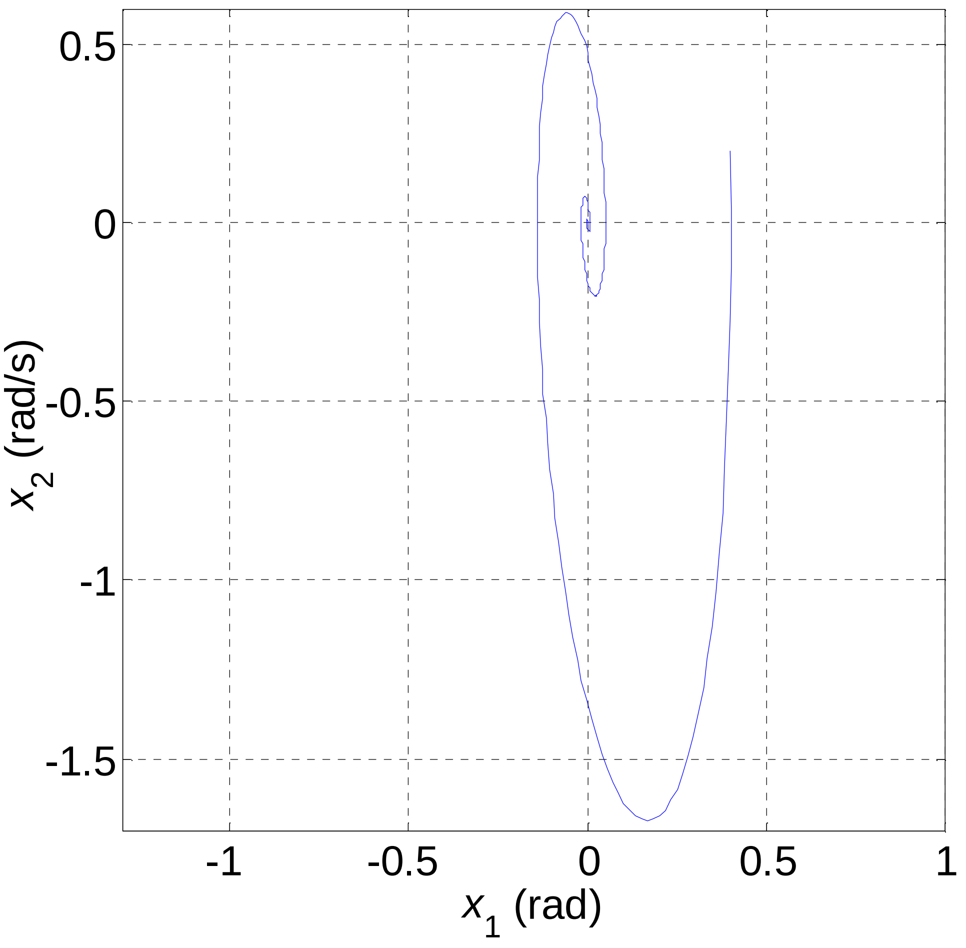

Figure 7.4 Phase Portrait Plot for Continuing Matlab Example

Figure 7.4 plots velocity x2 vs. displacement x1. Since this is an asymptotically-stable system, the phase portrait spirals in from the given conditions $\mathbf{X}(0)=\{0.4 \quad 0.2\}^T$ to the final state vector values of $\mathbf{X}(\infty)=\{0 \quad 0\}^T$. This is another view of the state responses shown in Figure 3.3.

5.5 Continuing Examples: Stability Analysis

5.5.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

Determine the stability condition for the system of Continuing Example I (Two-mass MIMO Translational Mechanical System). Stability is a fundamental property of a system; it is only dependent on the system dynamics matrix **A** and not on matrices **B**, **C**, or **D**. The stability condition is the same regardless of the various possible combinations of input and output choices. Therefore, in this section there is only one stability analysis; it is the same for Case i (full MIMO), Case ii (SISO, input u_2 and output y_1) and all other input/output combinations. However, we will employ two methods to get the same result, simple eigenvalue analysis and Lyapunov stability analysis.

Solution, Simple Eigenvalue Analysis

This case was already presented in Continuing Example I, Chapter 3. If all real parts of all system poles are strictly negative, the system is **stable**. If just one real part of a pole is zero (and the real parts of the remaining system poles are zero or strictly negative), the system is **marginally stable**. If just one real part of a pole is positive (regardless of the real parts of the remaining system poles), the system is **unstable**.

From Chapter 3, the four open-loop system poles for Example I, found from the eigenvalues of **A**, are:

$$s_{1,2} = -0.5 \pm 4.44i \text{ and } s_{3,4} = -0.125 \pm 2.23i$$

Thus, this open-loop system is **stable** since all real parts of the four poles are strictly negative.

Solution, Lyapunov Analysis

The Lyapunov stability equation is:

$$\mathbf{A}^{\mathbf{T}}\mathbf{P} + \mathbf{PA} = -\mathbf{Q}$$

The stability analysis procedure is: for a given positive definite matrix **Q** (**I** is a great choice), solve for **P** in the above equation. If **P** turns out to be positive definite, then the system is **asymptotically stable in the large**; if **P** is positive semi-definite, the system is **marginally stable**; but if **P** is neither positive-definite nor positive semi-definite, the system is **unstable**.

The solution for **P** is:

$$\mathbf{P} = \begin{bmatrix} 15.76 & 0.29 & 1.93 & 0.38 \\ 0.29 & 1.78 & -0.19 & 1.09 \\ 1.93 & -0.19 & 9.16 & -0.04 \\ 0.38 & 1.09 & -0.04 & 1.46 \end{bmatrix}$$

Now we must check the positive-definiteness of **P** using Sylvester's criterion. The four principal minors of **P** are $\mathbf{P_4} = \mathbf{P}$ itself, and:

$$\mathbf{P_3} = \begin{bmatrix} 15.76 & 0.29 & 1.93 \\ 0.29 & 1.78 & -0.19 \\ 1.93 & -0.19 & 9.16 \end{bmatrix} \qquad \mathbf{P_2} = \begin{bmatrix} 15.76 & 0.29 \\ 0.29 & 1.78 \end{bmatrix} \qquad \mathbf{P_1} = [15.76]$$

The determinants of the four principal minors of P are 194.88, 248.58, 27.96, and 15.76, respectively. All principal minor determinants are positive and therefore P is strictly positive definite. Therefore, this system is **asymptotically stable in the large**.

To enforce these stability results, Figures 7.5 present the phase portrait plots for Example I, Case i (full MIMO system with two step inputs of 20 and 10 *N*, respectively, and zero initial conditions).

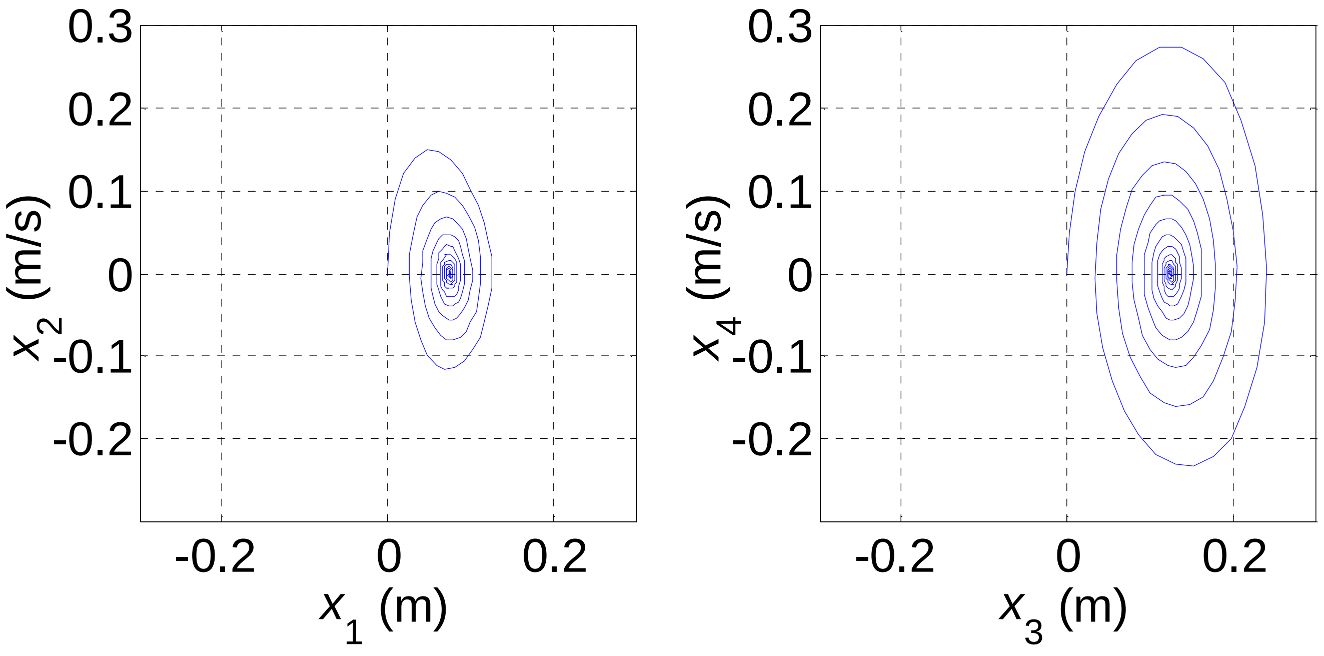

Figure 7.5 Phase Portraits for Case i

Figures 7.5 plot velocity x_2 vs. displacement x_1 on the left and velocity x_4 vs. displacement x_3 on the right. Since this is a stable system, the phase portraits both spiral in from the given zero initial conditions on all state vector components to the final state vector, presented in the Chapter 3 Example I, $\mathbf{X_{ss}} = \mathbf{X}(\infty) = \{0.075 \quad 0 \quad 0.125 \quad 0\}^{\mathbf{T}}$. Since Figures 7.5 are both plotted to the same scale, we see that mass 2 undergoes higher displacement and velocity motions than mass 1. This can also be seen (for displacements only) in Figures 3.5, Chapter 3.

To further enforce the stability results, Figures 7.6 present the phase portrait plots for Example I, Case ii (SISO, with zero input u_2 and output y_1, plus given initial conditions $\mathbf{X}(0) = \{0.1 \quad 0 \quad 0.2 \quad 0\}^T$).

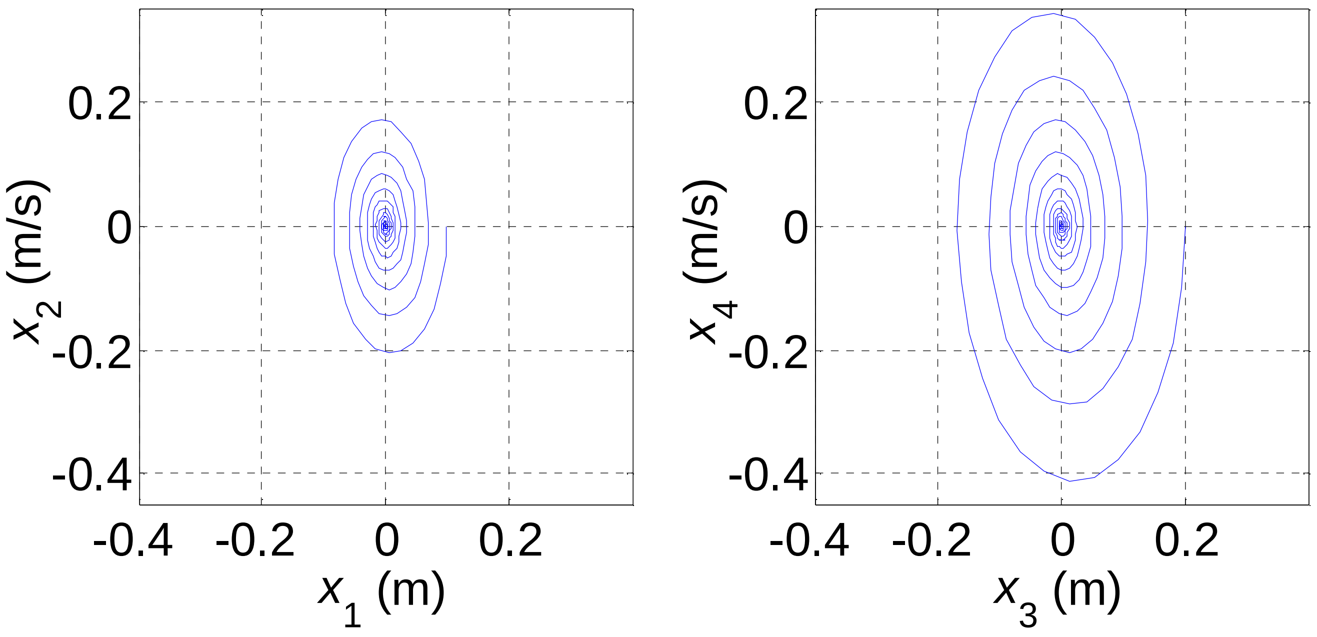

Figure 7.6 Phase Portraits for Case ii

Figures 7.6 again plot velocity x_2 vs. displacement x_1 on the left and velocity x_4 vs. displacement x_3 on the right. Since this is a stable system, the phase portraits both spiral in from the given non-zero initial conditions (displacements; initial velocities are zero) on all state vector components to the final state vector values of all zeros. Since its initial displacement is double that of mass 1, we see that mass 2 undergoes higher displacement and velocity motions than mass 1. This can also be seen, along with the zero final state values in Figures 3.6, Chapter 3.

5.5.2 Continuing Example II: Rotational Electromechanical System

Determine the stability condition for the system of Continuing Example II (1-dof rotational electromechanical system; SISO: input v, output θ). We will attempt to employ two methods, simple eigenvalue analysis and Lyapunov stability analysis.

Solution, Simple Eigenvalue Analysis

This case was already presented in Continuing Example II, Chapter 3. If all real parts of all system poles are strictly negative, the system is **stable**. If just one real part of a pole is zero (and the real parts of the remaining system poles are zero or strictly negative), the system is **marginally stable**. If just one real part of a pole is positive (regardless of the real parts of the remaining system poles), the system is **unstable**.

From Chapter 3, the three open-loop system poles for Example II, found from the eigenvalues of **A**, are:

$$s_{1,2,3} = 0, -1, -2$$

Thus, this open-loop system is **marginally stable** since there is a zero pole and the rest are real and negative. Marginally stable in this example means that the input angle has a rigid body mode; i.e. when voltage is applied, the output shaft angle will increase linearly without bounds in the steady state. This does not pose a problem as this is how a DC servomotor is supposed to behave.

Solution, Lyapunov Analysis

Determining the stability condition using Lyapunov analysis is not possible. Matlab cannot solve the Lyapunov stability equation (Solution does not exist or is not unique), due to the singular system dynamics matrix **A**. This also indicates that the system is marginally stable.

These marginal-stability results can be further demonstrated by the phase portrait plots of Figure 7.7.

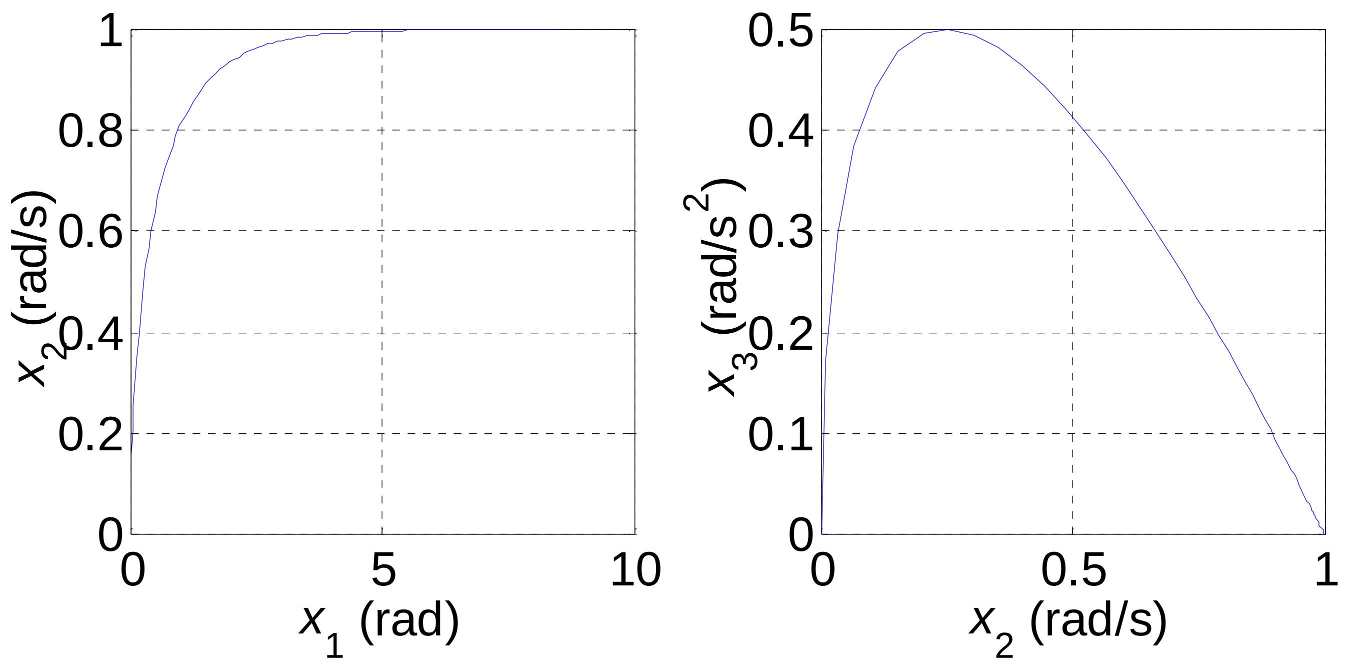

Figure 7.7. Phase Portraits for Rotational Electromechanical System

Figures 7.7 plot motor shaft angular velocity x_2 vs. angle x_1 on the left and angular acceleration x_3 vs. angular velocity x_2 on the right. Since this is a marginally stable system, the phase portraits both start at the given zero initial conditions on all state vector components; they neither spiral back towards the zero equilibrium position nor spiral out of control as an unstable system. This means that the system is marginally stable, approaching a constant angular velocity of 1 *rad*/*s* in steady-state motion.

5.6 Homework Assignments

5.6.1 Mathematical Homework Assignments

5.6.2 Matlab Homework Assignments

5.6.3 Continuing Homework Assignments

CE1.5
Using Lyapunov analysis, determine the stability condition for the CE1 system; any case will do – since the **A** matrix is identical for all input/output cases, the stability condition does not change. Check your results via simple pole analysis.

CE2.5
Using Lyapunov analysis, determine the stability condition for the CE2 system; any case will do – since the **A** matrix is identical for all input/output cases, the stability condition does not change. Lyapunov stability analysis will not succeed (why?); therefore, determine system stability via simple pole analysis.

CE3.5
Using Lyapunov analysis, determine the stability condition for the CE3 system; either case will do – since the **A** matrix is identical for all input/output cases, the stability condition does not change. Check your results via simple pole analysis.

6. Design of Linear State-Feedback Controllers

This chapter presents . . .

In this chapter we will design controllers for the state-space description systems. This is the topic all other chapters have been leading up to. The controller design problem is to calculate a full-state-feedback gain matrix to provide desired behavior in the closed-loop system, masking the original open-loop behavior. This process involves placing desired closed-loop poles into the closed-loop system dynamics matrix. Controller design should be followed by simulation to determine the simulated closed-loop behavior of your controller design.

First, shaping of dynamic response (how controllers should be prescribed to behave).

6.1 Shaping Dynamic Response

If we don't like the open-loop system performance (i.e. responses to step, impulse, and other inputs), we can force system output to perform as desired using feedback control. This is, in the design of controllers we can place the n poles of the closed-loop system in order to:

- ensure stability
- achieve desired transient behavior (shaping dynamic response)

This chapter is another important step on the way to controller design: if we don't like the open-loop response, how should we specify good responses with which to replace it?

6.1.1 Dominant, Augmented Desired System

In this method we will determine good desired system behavior by choosing desired second- or first-order poles and then augmenting them with enough additional poles to obtain n desired poles (we need to specify desired closed-loop poles equal in number to the order of the system (the number of state variables)). We choose all additional augmented poles to be real, negative, and about ten times greater than the real part of the desired dominant poles. If so, the effect of these augmented poles will not be seen much since their transient dynamics will be much faster than the desired dominant system behavior. In this way the specified desired dominant response will still dominate.

6.1.1.1 Second-Order Dominant System

Why study second-order systems?

- many real systems modeled with 2*nd* order ODEs
- design controller so higher-order system mimics desired 2*nd* order system

Let us consider the linear SISO 1-dof *m-c-k* mechanical translational system (see Figure 1.2) with $y(t)$ output displacement and $f(t)$ input force. We can turn this into a generic second-order system, that applies to any physical system that can be modeled with a linear second-order system. First, replace the forcing function *f*(*t*) with the spring *k* times a displacement input *u*. This will in effect normalize the generic system output to 1.0 since the natural frequency squared then shows up on both sides of the ODE:

$$m\ddot{y} + c\dot{y} + ky = f = ku$$
$$\ddot{y} + \frac{c}{m}\dot{y} + \frac{k}{m}y = \frac{k}{m}u$$
$$\ddot{y} + 2\xi\omega_n\dot{y} + \omega_n^2 y = \omega_n^2 u$$

Where: $\omega_n = \sqrt{\frac{k}{m}}$ *rad*/*s* is the natural frequency, *rad*/*s*

$\xi = \frac{c}{2\sqrt{km}}$ is the dimensionless damping ratio

The generic 2nd-order system transfer function is then:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

The associated characteristic polynomial is the denominator of *G*(*s*):

$$s^2 + 2\xi\omega_n s + \omega_n^2 = 0$$

The system poles are found from the roots of the characteristic polynomial:

$$s_{1,2} = -\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}$$

The nature and values of the poles determine the system transient response. We identify five distinct cases, according to the dimensionless damping ratio ξ.

Damping	Unit step response
$\xi > 1$	Overdamped; real distinct negative poles, slow response
$\xi = 1$	Critically damped; real repeated negative poles, fastest response without overshoot
$0 < \xi < 1$	Underdamped; complex conjugate poles, overshoot and oscillation
$\xi = 0$	Undamped; complex conjugate poles with zero real parts, simple harmonic motion (vibrates theoretically forever since there is no damping in the model).
$\xi < 0$	Unstable; at least one pole with a positive real part, exponential term in solution approaches infinity rather than a finite stead-state value.

Second-Order System Performance Specifications

Now, for determining desired closed-loop poles for controller design (Chapter 8), we can choose dominant second-order poles from any of the first four cases above (since we cannot specify unstable poles). Of these, the third is most interesting and most commonly used for dynamic shaping: the underdamped case with $0<\xi<1$. The poles in this case are:

$$s_{1,2} = -\xi\omega_n \pm i\omega_d$$

where: $\omega_d = \omega_n\sqrt{1-\xi^2}$ is the damped frequency.

The unit step response solution for the generic underdamped 2nd-order system is:

$$y(t) = 1 - \frac{e^{-\xi\omega_n t}}{\sqrt{1-\xi^2}}\sin(\omega_d t + \alpha) \qquad \alpha = \sin^{-1}\sqrt{1-\xi^2}$$

This solution has an exponential damping envelope due to the (negative) real part of the poles and an oscillatory sin wave of the damped natural frequency plus a phase angle α. A plot of the form of this solution will be given later in an example, in Figure 4.1.

For this underdamped generic 2nd-order system, there are four performance specifications (see Figure 4.1). Rise time is the relative time between when the output first reaches 10% of the final value to when the output first reaches 90% of the final value. Peak time is the absolute time at which the peak value is reached. Percent overshoot is the maximum output compared to the final steady-state value of 1.0 and converted to a percentage. Settling time would be theoretically infinite; thus for design purposes we define settling time to be when the output enters the $\pm 2\%$ envelope about the final steady-state value and never leaves again. The formulas for rise time, peak time, percent overshoot, and settling time are (DERIVE OR REFER TO Dorf and Bishop ??!!??) given below, as functions of the natural frequency and dimensionless damping ratio. Peak time and percent overshoot are theoretically exact, but rise time and settling time are approximations.

1) Rise Time (10-90%) $t_R \cong \dfrac{2.16\xi + 0.60}{\omega_n}$ (best for $0.3<\xi<0.8$)

2) Peak Time $t_P = \dfrac{\pi}{\omega_n\sqrt{1-\xi^2}}$

3) Percent Overshoot $PO = 100e^{\left(\frac{-\xi\pi}{\sqrt{1-\xi^2}}\right)}$

4) Settling Time ($\pm 2\%$) $t_S \cong \dfrac{4}{\xi\omega_n}$

To control the swiftness of the response we can change the rise time and peak time; to control the error of the response, we can change the percent overshoot and settling time. These are competing requirements.

Let us develop a generic underdamped 2nd-order system example:

$$m = 1\ kg \qquad c = 1\ Ns/m \qquad k = 10\ N/m$$

The natural frequency and dimensionless damping ratio are:

$$\omega_n = \sqrt{\frac{k}{m}} = \sqrt{\frac{10}{1}} = 3.16 \quad rad/s \qquad\qquad \xi = \frac{c}{2\sqrt{km}} = \frac{1}{2\sqrt{10(1)}} = 0.158$$

The damped natural frequency is:

$$\omega_d = \sqrt{1-\xi^2} = 3.12 \quad rad/s$$

The generic 2nd-order system transfer function is:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} = \frac{10}{s^2 + s + 10}$$

The characteristic polynomial is:

$$s^2 + 2\xi\omega_n s + \omega_n^2 = s^2 + s + 10 = 0$$

The system poles (roots of the characteristic polynomial) are complex conjugates:

$$s_{1,2} = -0.5 \pm 3.12i$$

The unit step response solution is:

$$y(t) = 1 - 1.01e^{-0.5t}\sin\left(3.12t + 80.9^\circ\right)$$

From ξ and ω_n we calculate the four performance specifications:

t_r	= 0.30 *sec* (poor approximation since $\xi < 0.3$ - from Matlab simulation data, the value is t_r=0.37 *s*)
t_p	= 1.01 *sec*
PO	= 60.5%
t_s	= 8 *sec*

Figure 4.1 gives a plot of the unit step response with rise time, peak time, % overshoot, and settling time. This plot was obtained from Matlab using the *step* function and right-clicking in the resulting figure window to add the performance specifications. We see that, with the exception of rise time, the formula values agree well with the Matlab numerical values of Figure 4.1 (settling time has some error as well, since this is an approximate equation).

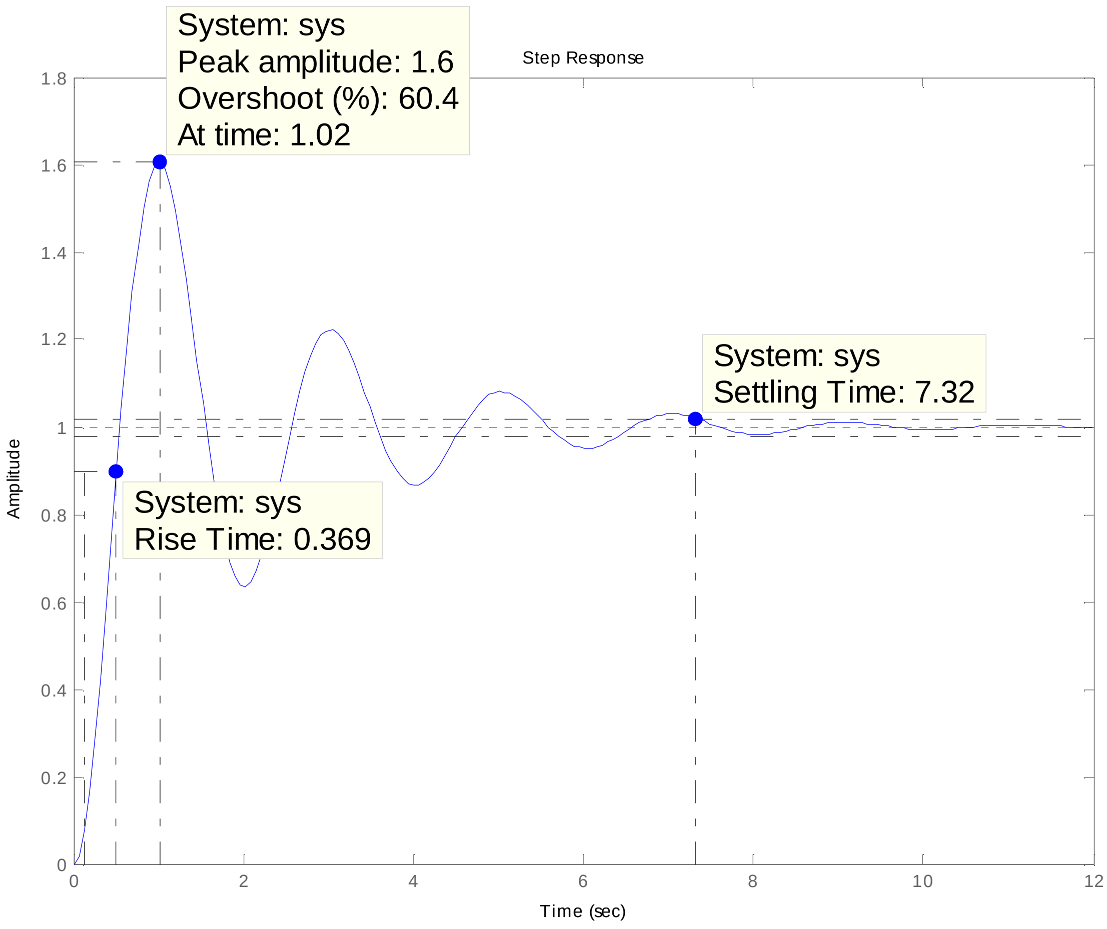

Figure 4.1 Example Generic 2nd-Order System Unit Step Response

The step response in Figure 4.1 is typical of real-world lightly-damped systems such as certain space robots. Let us assume that this is the original given 2nd-order open-loop system and we must design controller poles to improve this response. In the generic 2nd-order system, ξ affects the damping envelope (how much percent overshoot) and ω_n affects the speed of the response, i.e. peak and rise times (in conjunction with ξ). Both ξ and ω_n affect settling time. We will specify a percent overshoot of 4% and a settling time of 2 *sec* to improve the response. Given these desired performance specifications, we can calculate new desired closed-loop system dimensionless damping ratio ξ from the percent overshoot equation and then we can calculate the natural frequency ω_n from the settling time approximate equation (using the new ξ):

$$\xi = \frac{\ln\left(\frac{PO}{100}\right)}{\sqrt{\pi^2 + \left[\ln\left(\frac{PO}{100}\right)\right]^2}} \qquad\qquad \omega_n = \frac{4}{\xi t_S}$$

Given the specified PO = 4% and t_S = 2 *sec*, we find the desired ξ and ω_n to be:

$$\xi = 0.716 \qquad\qquad \omega_n = 2.79$$

The new damped natural frequency is:

$$\omega_d = \sqrt{1-\xi^2} = 1.95 \quad rad/s$$

The new generic 2nd-order system transfer function is:

$$G(s) = \frac{7.81}{s^2 + 4s + 7.81}$$

The new characteristic polynomial is:

$$s^2 + 4s + 7.81 = 0$$

The new, desired system poles are:

$$s_{1,2} = -2 \pm 1.95i$$

The improved unit step response solution is:

$$y(t) = 1 - 1.43e^{-2t}\sin\left(1.95t + 44.3^\circ\right)$$

Matlab *step* gives us the four performance specifications:

t_r	= 0.78 *sec*
t_p	= 1.60 *sec*
PO	= 4%
t_s	= 2.12 *sec*

Note that the percent overshoot was satisfied (theoretically) exactly, while the settling time was close. Figure 4.2 gives a plot of the unit step response for the improved, desired system, compared with the unit step response of the open-loop system from Figure 4.1.

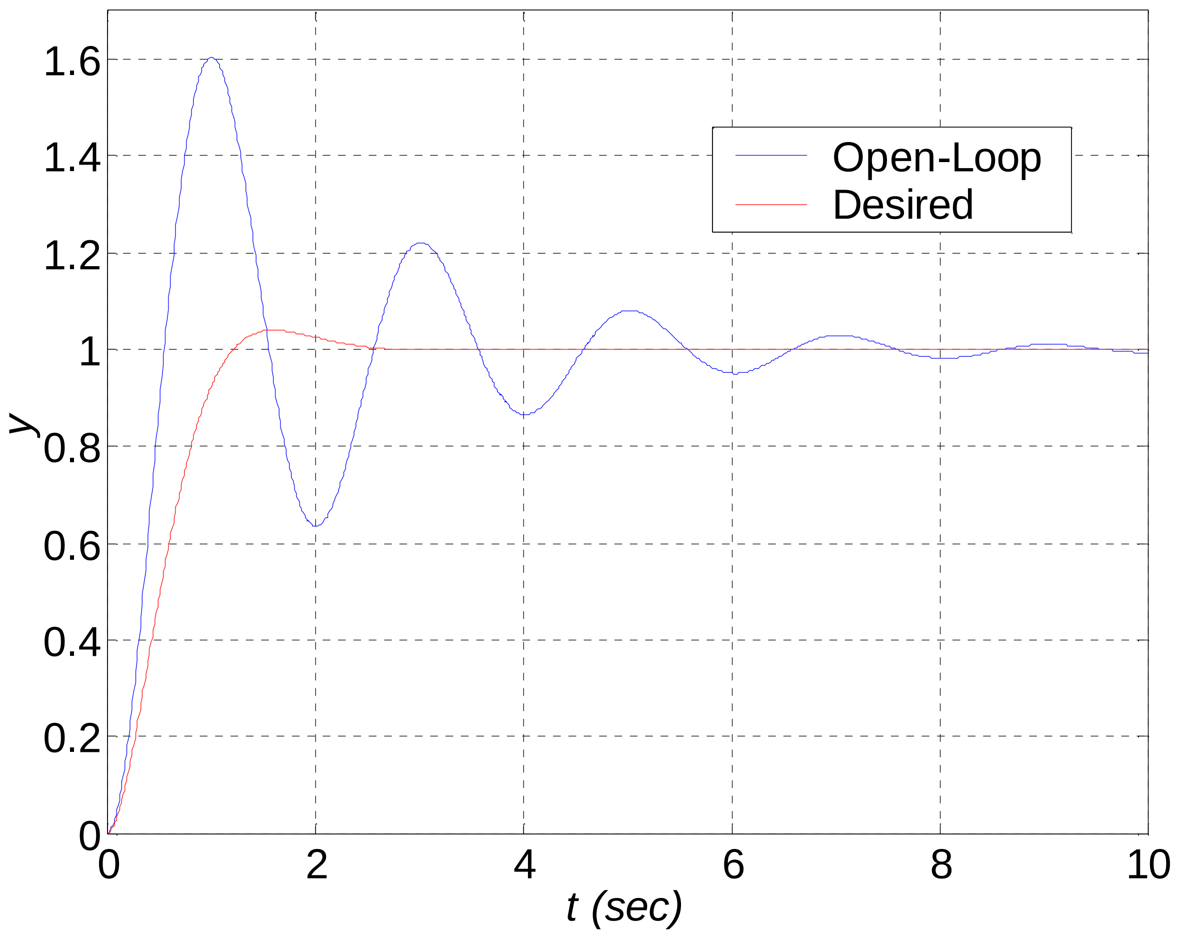

Figure 4.2 Desired vs. Open-Loop System Unit Step Responses

In Figure 4.2 and Table 4.I below we see that the desired, improved system rises slower and peaks slower than the open-loop system, but the error, as measured by the percent overshoot and settling time, is much improved.

Table 4.I Comparison of Open-Loop and Desired Performance Specifications

Specification	Open-Loop Response	Desired Closed-Loop Response
t_r (*sec*)	0.37	0.78
t_p (*sec*)	1.02	1.60
PO (%)	60.4	4
t_s (*sec*)	0.73	2.12

6.1.1.2 First-Order Dominant System

We can also specify desired closed-loop behavior via a dominant 1st-order pole. Figure 3.1 shows a standard 1st-order system step response. Assuming the system is stable and the pole is non-zero, all 1st-order systems driven by unit step inputs behave this way, i.e. rising (or falling) exponentially from the single given initial condition to the steady-state value. Since the system is only 1st-order, we only have 1-dof. So to specify a dominant pole we can set the desired time constant τ (rather than changing two performance specifications to modify ξ and ω_n as in the 2nd-order case above). After three time constants, the 1st-order unit step response is within 95% of its final value. A smaller time constant responds more quickly while a larger time constant responds more slowly. The characteristic polynomial and associated pole for a dominant 1st-order system is:

$$s+\frac{1}{\tau}=0$$
$$s=-\frac{1}{\tau}$$

6.1.1.3 Augmenting Desired Dominant Systems

Now, many MIMO and SISO systems have a system order n greater than 1 or 2. For controller design (Chapter 8), we need to specify as many desired controller poles as the system order n. When we start with a dominant 1st- or 2nd-order desired system, we need to augment the dominant poles with enough addition poles for n total desired poles. All additional augmented poles should be real, negative, and about ten times greater than the real part of the desired dominant poles. Then the effect of these augmented poles will not change the desired behavior significantly and the specified desired dominant response will dominate.

First, let us consider typical 1st- through 4th-order system unit step responses. A stable 1st-order system can only behave like Figure 3.1. There are many options for 2nd-order system responses (we considered the underdamped case in detail); for 3rd- and 4th-order systems there are even more possibilities. Figure 4.3 shows some typical responses for 1st- through 4th-order systems.

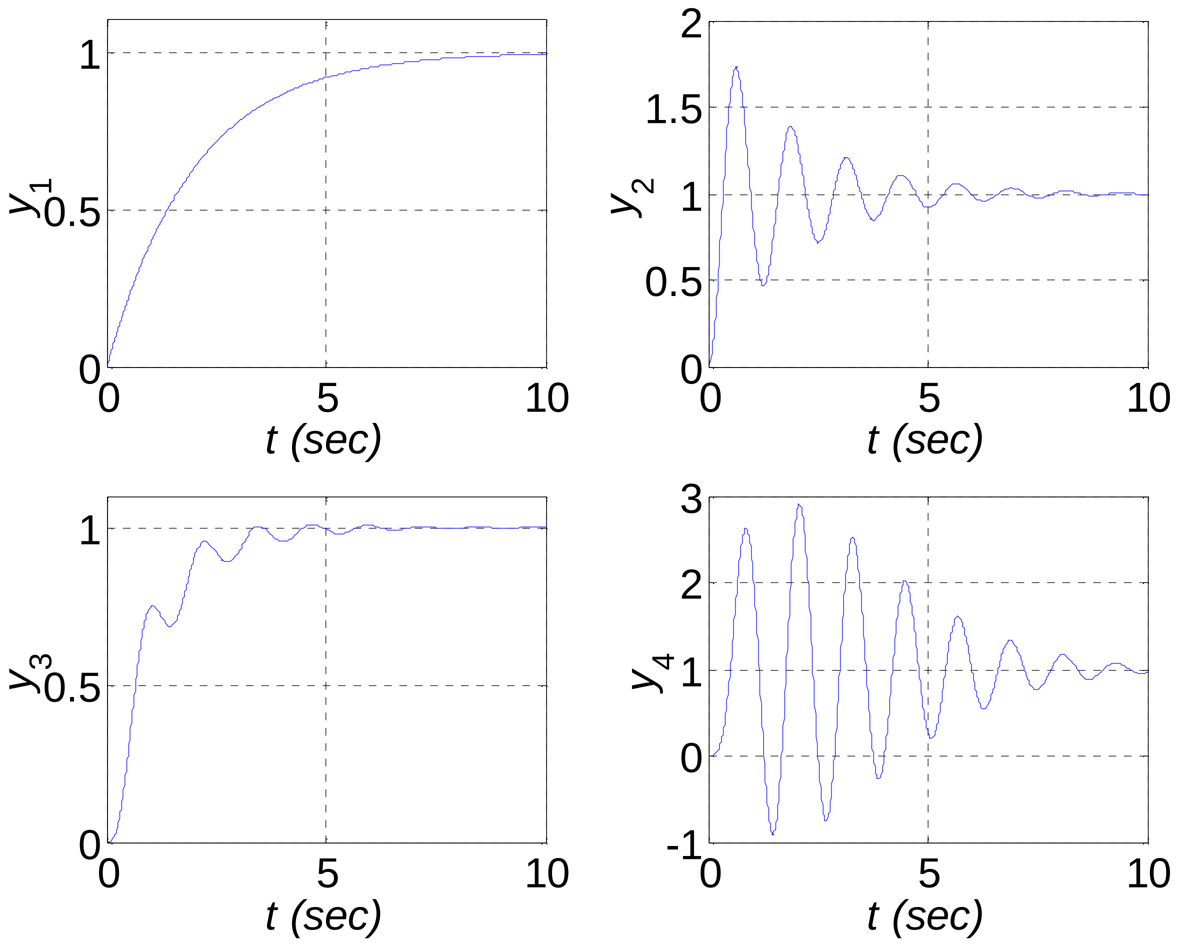

Figure 4.3 1st- through 4th-Order System Unit Step Responses

The 1st- and 2nd-order step responses y_1 and y_2 are similar to what we have seen before. The 3rd-order system response y_3 is a superposition of a typical 1st-order rise and underdamped 2nd-order system. The 4th-order system response y_4 shows a typical 4th-order beat frequency. All responses in Figure 4.3 were normalized for steady-state values of 1.0. The poles associated with each case are given in the table below.

Table 4.II Poles for Figure 4.3

Order *n*	**Poles**
1st	$s_1 = -0.5$
2nd	$s_{1,2} = -0.5 \pm 5i$
3rd	$s_{1,2,3} = -0.5 \pm 5i, -1$
4th	$s_{1,2,3,4} = -0.5 \pm 5i, -0.5 \pm 5.5i$

Figure 4.4 shows the effect of augmenting a dominant 1st-order desired pole ($s_1 = -0.5$) with additional, real, negative poles, at least 10 times higher, for 2nd- through 4th-order systems.

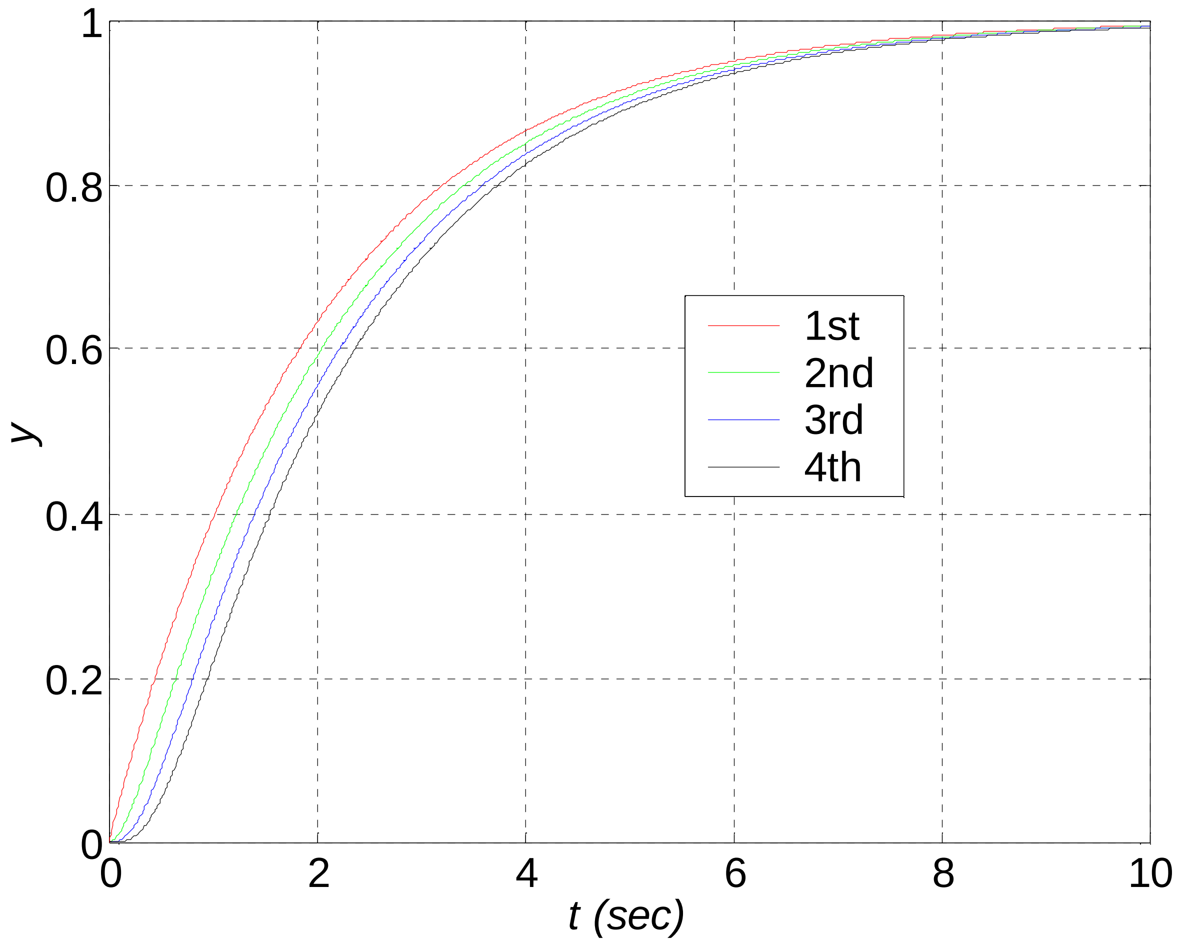

Figure 4.4 2nd- through 4th-Order Systems Mimicking Dominant 1st-Order System

The poles associated with each case are given in the table below.

Table 4.III Poles for Figure 4.4

Order *n*	**Poles**
1st	$s_1 = -0.5$
2nd	$s_{1,2} = -0.5, -5$
3rd	$s_{1,2,3} = -0.5, -5, -6$
4th	$s_{1,2,3,4} = -0.5, -5, -6, -7$

The 2nd- through 4th-order step responses are similar to the desired dominant 1st-order step response. The time lag increases slightly as the system order increases. Also, the 2nd- through 4th-order step responses start at initial conditions of zero displacement and zero derivatives (one for 2nd-, two for 3rd-, and three for 4th-order systems).

Figure 4.5 shows the effect of augmenting dominant 2nd-order desired poles ($s_{1,2} = -2 \pm 1.95i$ from Section 4.1.1) with additional, real, negative poles, at least 10 times higher, for 3rd- and 6th-order systems.

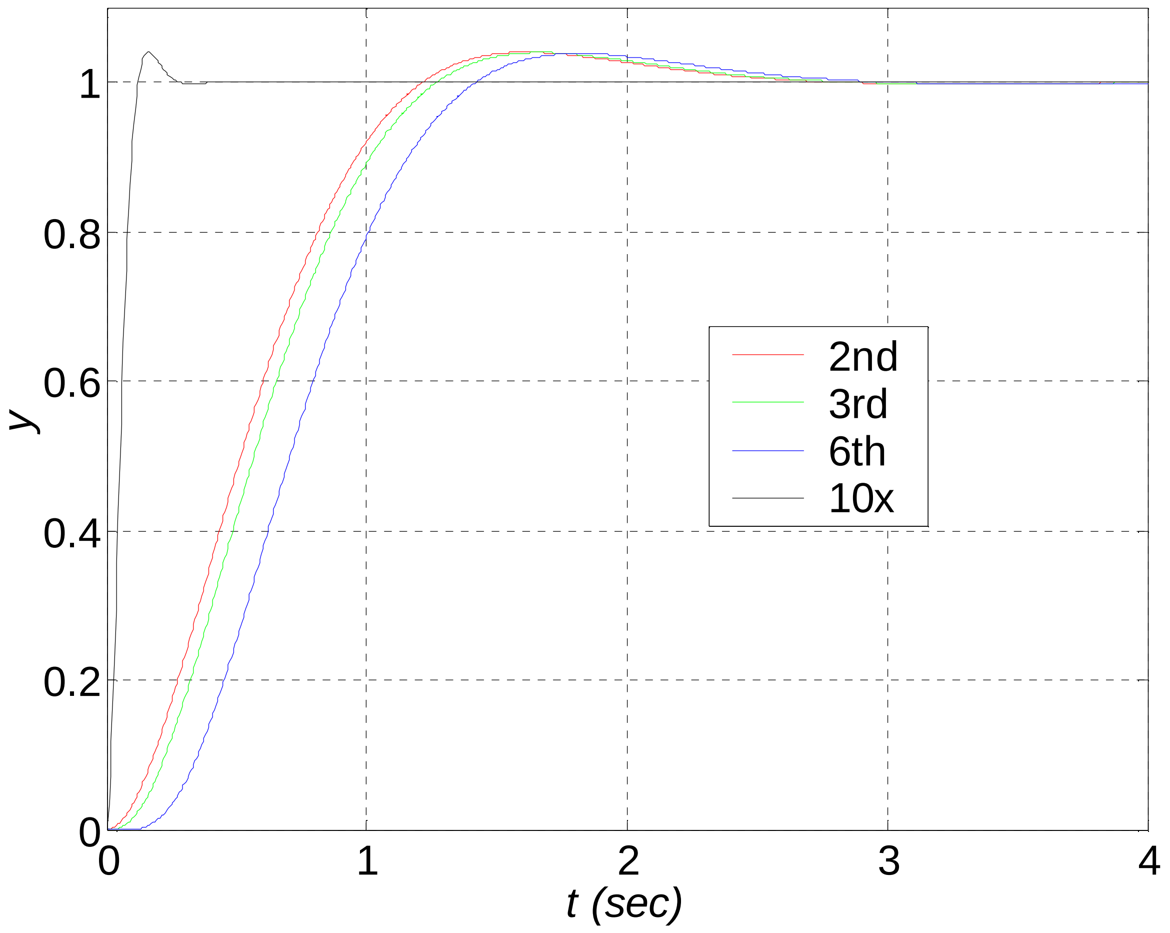

Figure 4.5 3rd- and 6th-Order Systems Mimicking Dominant 2nd-Order System

The poles associated with each case are given in the table below.

Table 4.IV Poles for Figure 4.5

Order *n*	**Poles**
2nd	$s_{1,2} = -2 \pm 1.95i$
3rd	$s_{1,2,3} = -2 \pm 1.95i, -20$
6th	$s_{1,2,3,4,5,6} = -2 \pm 1.95i, -20, -21, -22, -23$
10x	$s_{1,2} = -20 \pm 19.5i$

The 3rd- and 6th-order step responses are similar to the desired dominant 2nd-order step response. The time lag again increases as the system order increases. The 10x curve plots the unit step response for a 2nd-order system with poles 1o times higher than the dominant $s_{1,2} = -2 \pm 1.95i$. This shows why augmenting dominant poles with higher poles works (the transient dynamics occurs much faster for the higher poles).

6.1.2 *ITAE* Dynamic Shaping Method

With Section 4.1, we found that specification of fast response and low errors in system responses are competing factors. The *ITAE* (Integral of time multiplied the absolute value of error) method attempts to accomplish dynamic shaping by balancing both of these competing factors. The *ITAE* objective function is:

$$ITAE = \int_0^\infty t\left|e(t)\right|dt$$

Minimize *ITAE* to simultaneously optimize competing requirements. Minimum *ITAE* means short time and small error at once. Other possible objective functions. For first- through fourth-order systems, the following characteristic polynomials minimize *ITAE* (DERIVE OR REFER TO Dorf and Bishop ??!!??). Design feedback controller to meet one of these specifications and the shaping of the dynamic response will be optimized according to ITAE. The resulting desired characteristic polynomials are given in Table 4.V for 1st- through 4th-order systems. In each case, the engineer must specify the desired natural frequency ω_n (higher values correspond to faster response). Then the desired poles are obtained by finding the roots of the appropriate characteristic polynomial for the specific system order *n*.

Table 4.V *ITAE* Characteristic Polynomials

Order *n*	**Poles**
1st	$s+\omega_n$
2nd	$s^2+1.4\omega_n s+\omega_n^2$
3rd	$s^3+1.75\omega_n s^2+2.15\omega_n^2 s+\omega_n^3$
4th	$s^4+2.1\omega_n s^3+3.4\omega_n^2 s^2+2.7\omega_n^3 s+\omega_n^4$
5th	$s^5+2.8\omega_n s^4+5.0\omega_n^2 s^3+5.5\omega_n^3 s^2+3.4\omega_n^4 s+\omega_n^5$
6th	$s^6+3.25\omega_n s^5+6.6\omega_n^2 s^4+8.6\omega_n^3 s^3+7.45\omega_n^4 s^2+3.95\omega_n^5 s+\omega_n^6$

Note for all cases (except the first order system) some overshoot is required to optimize *ITAE*.

6.2 Feedback Control Law

Given the original open-loop state-space system:

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$

$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$$

Draw the open-loop diagram first. Assume the output $\mathbf{Y}(t)$ is not performing as desired (underdamped, unstable, other issues). Add full-state feedback to the open-loop diagram.

Diagram:

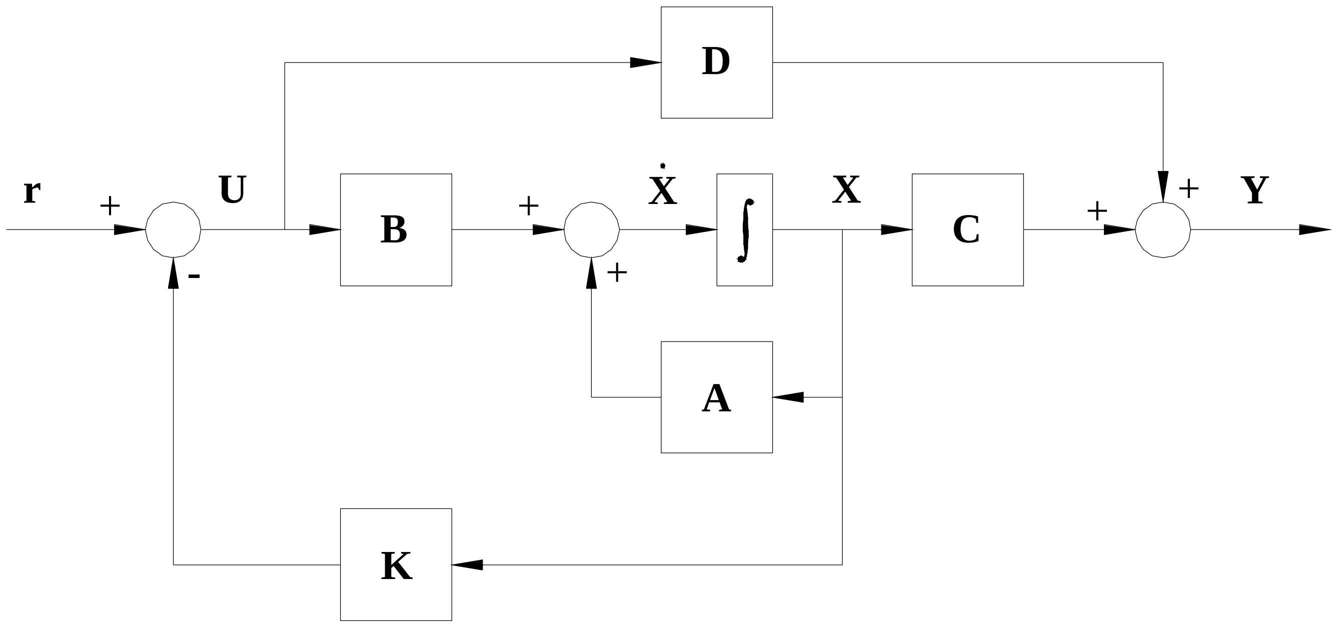

Figure 8.1 State-Space Closed-Loop System Block Diagram

$\mathbf{r}(t)$	reference input	same units and dimensions (r x 1) as $\mathbf{U}(t)$
K	constant full-state feedback gain matrix	dimensions (r x n)

The Feedback Control Law is taken as:

$$\mathbf{U}(t) = \mathbf{r}(t) - \mathbf{KX}(t) \qquad r \times 1 \;=\; r \times 1 \;-\; (r \times n)(n \times 1)$$

For SISO systems, the reference input **r** is a scalar, feedback gain matrix **K** is a row matrix, and the input **u** is a single input. The control law becomes:

$$\mathbf{u} = \mathbf{r} - k_1 x_1 - k_2 x_2 - \cdots - k_n x_n$$

If $\mathbf{r}(t)=\mathbf{0}$, the controller is a *regulator* (reject initial conditions and/or disturbances) - maintain equilibrium state $\mathbf{X_e}=\mathbf{0}$.

6.3 Controller Pole Placement

Now we derive the closed-loop state dynamics equation. Substitute the above control law ($\mathbf{U} = \mathbf{r} - \mathbf{KX}$) into the system dynamics equation $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$; the system output equation $\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$ does not change.

$$\begin{aligned}
\dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{BU} = \mathbf{AX} + \mathbf{B}(\mathbf{r} - \mathbf{KX}) \\
\dot{\mathbf{X}} &= (\mathbf{A} - \mathbf{BK})\mathbf{X} + \mathbf{Br} = \mathbf{A_c X} + \mathbf{B_c r} \\
\mathbf{Y} &= \mathbf{CX} + \mathbf{DU} = \mathbf{C_c X} + \mathbf{D_c U}
\end{aligned}$$

$$\begin{aligned}
\mathbf{A_c} &= \mathbf{A} - \mathbf{BK} \\
\mathbf{B_c} &= \mathbf{B} \\
\mathbf{C_c} &= \mathbf{C} \\
\mathbf{D_c} &= \mathbf{D}
\end{aligned}$$

So, in the closed-loop system, only the system dynamics matrix $\mathbf{A_c} = \mathbf{A} - \mathbf{BK}$ changes compared to the given open-loop system.

If the original system is not controllable, you cannot proceed in this chapter; instead you must determine why the system is not controllable by looking at the physical problem and then re-design your system or re-derive your system model until it is fully state controllable.

If the original open-loop system represented by **A**, **B** is *completely state-controllable* (see Section 5.1), a matrix **K** exists that can give arbitrary eigenvalues in the closed-loop system dynamics matrix $\mathbf{A_c} = \mathbf{A} - \mathbf{BK}$.

That is, we can place the roots (system poles) of the below closed-loop system characteristic equation anywhere we desire.

$$\left| s\mathbf{I} - [\mathbf{A} - \mathbf{BK}] \right| = 0$$

So we can achieve stability and desired transient performance design specifications:

- rise time
- peak time
- percent overshoot
- settling time
- damping, frequency

6.3.1 Bass-Gura Formula

Controller design for systems expressed in Controllable Canonical Form (*CCF*, see Section 6.2.1), also called Phase-Variable Canonical Form, proceeds as follows. Let us further restrict our consideration for now to SISO CCF systems. The **A**, **B** forms of *CCF* are given below. Again, for SISO, **K** is a row matrix with *n* elements.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} k_1 & k_2 & \cdots & k_n \end{bmatrix}$$

The expression for closed-loop system dynamics matrix $\mathbf{A_c = A - BK}$ becomes:

$$\mathbf{A - BK} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ (-a_0 - k_1) & (-a_1 - k_2) & (-a_2 - k_3) & \cdots & (-a_{n-1} - k_n) \end{bmatrix}$$

We see that $\mathbf{A_c = A - BK}$ is still in CCF form. Furthermore, the last row contains the coefficients of the closed-loop characteristic polynomial (with negative signs, appearing in reverse order of *s* powers). The closed-loop system poles are the eigenvalues of $\mathbf{A_c = A - BK}$, the same as the roots of the closed-loop characteristic equation:

$$\Delta_{CL}(s) = \left| s\mathbf{I} - [\mathbf{A - BK}] \right| = s^n + (a_{n-1} + k_n)s^{n-1} + \cdots + (a_2 + k_3)s^2 + (a_1 + k_2)s + (a_0 + k_1) = 0$$

In this form of the closed-loop characteristic equation, a_{i-1}, $i = 1, 2, \cdots, n$ are the original open-loop system characteristic polynomial coefficients (where a_{i-1} is the coefficient for s^{i-1}) and the *n* unknown components of the row matrix **K** are k_i.

The problem statement in controller design is to solve for unknown full-state-feedback gain matrix **K** given the original open-loop system and given the desired closed-loop system poles for the controller to achieve. Let us express the *n* desired closed-loop system poles (chosen via the methods of Chapter 4) as a known, desired characteristic polynomial; we do this by multiplying the *n* factors to obtain a single n^{th} order characteristic polynomial.

$$\Delta_{DES}(s) = s^n + \alpha_{n-1}s^{n-1} + \cdots + \alpha_2 s^2 + \alpha_1 s + \alpha_0 = 0$$

The crux of controller design is to match the closed-loop characteristic polynomial $\Delta_{CL}(s)$ containing the n unknowns k_i with the known, desired closed-loop characteristic polynomial $\Delta_{DES}(s)$. First, ensure that both polynomials are normalized (coefficient of 1.0 for s^n). Then we have a simple, linear, decoupled gains solution for the SISO *CCF* case:

$$k_i = \alpha_{i-1} - a_{i-1} \qquad\qquad i = 1, 2, \cdots, n$$

Implementing this method, the closed-loop system will mask the original open-loop system performance and replace it with the improved behavior associated with the n specified desired poles.

If one or more state component is not controllable, we cannot change the poles associated with these states.

Pole Placement via *K* (constant state feedback gain matrix): Example

$$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18 & -15 & -2\end{bmatrix} \qquad B=\begin{bmatrix}0\\0\\1\end{bmatrix} \qquad C=\begin{bmatrix}1 & 0 & 0\end{bmatrix} \qquad D=0 \qquad K=\begin{bmatrix}k_1 & k_2 & k_3\end{bmatrix}$$

Typical 3^{rd} order lightly damped response (see Figure ??). The open-loop system poles are $s_{1,2,3}=-1.28,-0.36\pm 3.73i$.

This open-loop system is already stable. Let's design a closed-loop state feedback controller to change the poles to improve transient performance. Choose a desirable 2nd-order system. We want the resulting closed-loop system to mimic a standard second-order system with 6% overshoot and 3 sec settling time. The associated damping ratio and natural frequency are $\xi=0.67$ and $\omega_n=2.00 \quad rad/s$. The resulting dominant 2nd-order poles are $s_{1,2}=-1.33\pm 1.49i$. The original is a 3rd-order system, so we need to choose a third desired pole. Make it negative, real, and 10 times higher than the (negative) real part of the dominant 2nd-order poles: $s_3=-13.33$. So the desired characteristic equation is:

$$s^3+\alpha_2 s^2+\alpha_1 s+\alpha_0=s^3+16s^2+39.55s+53.26=0$$

This will lead to better transient performance than the open-loop system. The decoupled CCF solution for the unknown constant full-state-feedback gain matrix is developed below.

$$A-BK=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18 & -15 & -2\end{bmatrix}-\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}k_1 & k_2 & k_3\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18 & -15 & -2\end{bmatrix}-\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ k_1 & k_2 & k_3\end{bmatrix}$$

$$A-BK=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18-k_1 & -15-k_2 & -2-k_3\end{bmatrix}$$

$$\left|sI-(A-BK)\right|=\begin{vmatrix}s & -1 & 0\\ 0 & s & -1\\ k_1+18 & k_2+15 & s+k_3+2\end{vmatrix}=0$$

$$s\left(s\left(s+k_3+2\right)+k_2+15\right)-\left(-1\right)\left(k_1+18\right)=0$$

$$s^3+\left(k_3+2\right)s^2+\left(k_2+15\right)s+\left(k_1+18\right)=0$$

$$s^3+16s^2+39.55s+53.26=0$$

Top characteristic polynomial function of *K*, bottom desired closed-loop characteristic polynomial.

$$k_1 = 53.26 - 18$$
$$k_2 = 39.55 - 15$$
$$k_3 = 16 - 2$$

With *CCF*, the solution for *K* is decoupled (Equation (??)).

$$K = \begin{bmatrix} 35.26 & 24.55 & 14.00 \end{bmatrix}$$

Units? $K \equiv \frac{U}{X}$ $\quad (u(t) = r(t) - Kx(t))$

e.g. $K \equiv \begin{bmatrix} \frac{N}{m} & \frac{Ns}{m} & \frac{Ns^2}{m} \end{bmatrix}$ Units of *k*, *c*, *m* in mechanical system.

The plot below shows a comparison of the open- and closed-loop output responses to a unit step input.

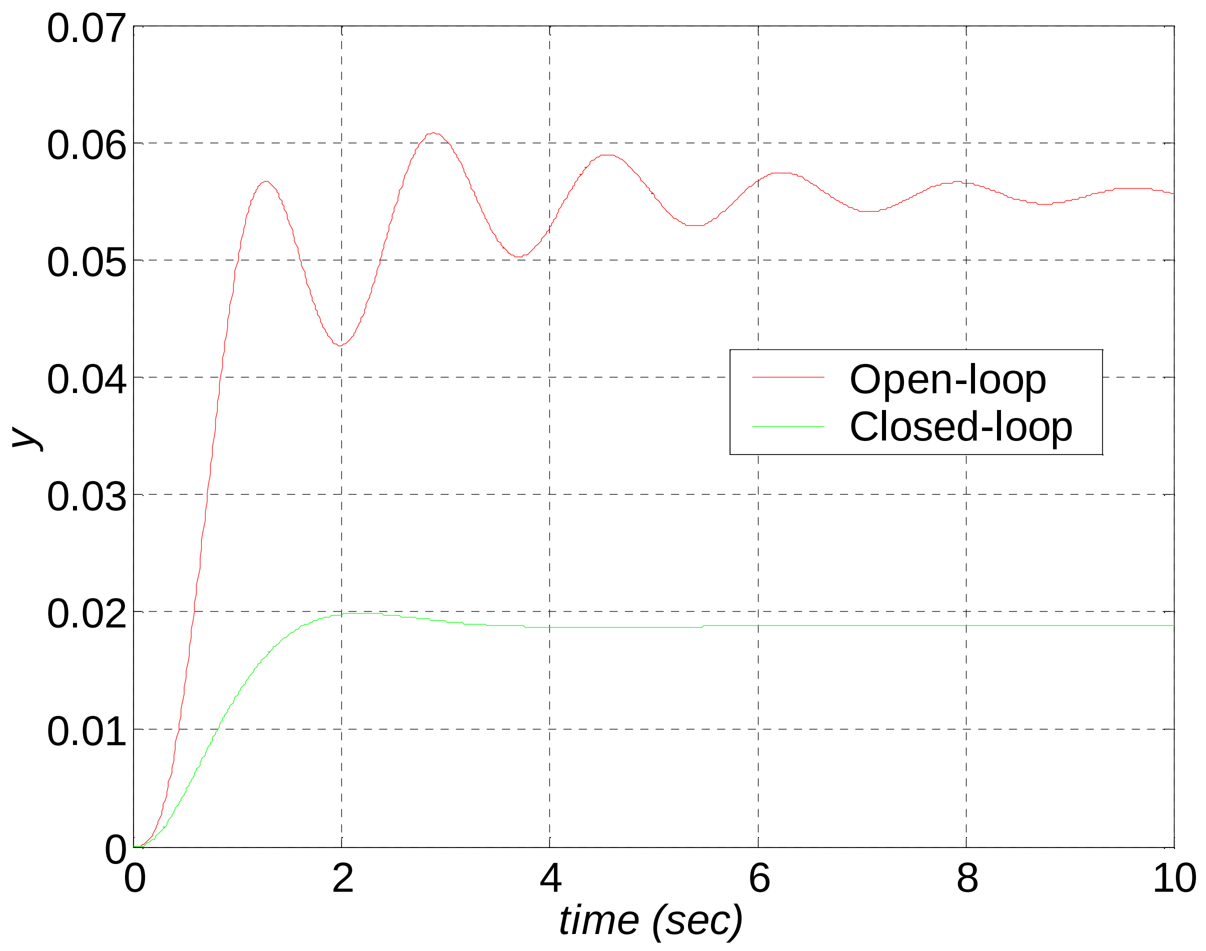

Figure 8.? Open- vs. Closed-Loop Output Response for Example

We will address the output attenuation of Figure 8.? in Section 6.4.

6.3.2 Ackerman's Formula

To calculate **K** for any general SISO system (not just *CCF*), assuming the same linear feedback control law:

$$\mathbf{U}(t)=\mathbf{r}(t)-\mathbf{KX}(t)$$

Ackerman's formula is:

$$\mathbf{K}=\begin{bmatrix}0 & 0 & \cdots & 1\end{bmatrix}\mathbf{P^{-1}}\Delta_{DES}(\mathbf{A})$$

where:

K	calculated constant state-feedback gain matrix
$\mathbf{P}=\begin{bmatrix}\mathbf{B} & \mathbf{AB} & \mathbf{A^2B} & \cdots & \mathbf{A^{n-1}B}\end{bmatrix}$	controllability matrix

And $\Delta_{DES}(\mathbf{A})$ is the specified, desired closed-loop system characteristic polynomial, evaluated with the system dynamics matrix **A**, rather than *s*.

$$\Delta_{DES}(s)=s^n+\alpha_{n-1}s^{n-1}+\cdots+\alpha_2 s^2+\alpha_1 s+\alpha_0$$
$$\Delta_{DES}(\mathbf{A})=\mathbf{A}^n+\alpha_{n-1}\mathbf{A}^{n-1}+\cdots+\alpha_2\mathbf{A}^2+\alpha_1\mathbf{A}+\alpha_0\mathbf{I}$$

For general MIMO systems, controller design can be accomplished with Matlab function *place* (see Section 8.4).

<u>Example</u>

$$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18 & -15 & -2\end{bmatrix}\qquad B=\begin{bmatrix}0\\0\\1\end{bmatrix}$$

$$P=\begin{bmatrix}B & AB & A^2B\end{bmatrix}=\begin{bmatrix}0 & 0 & 1\\ 0 & 1 & -a_2\\ 1 & -a_2 & a_2^2-a_1\end{bmatrix}=\begin{bmatrix}0 & 0 & 1\\ 0 & 1 & -2\\ 1 & -2 & -11\end{bmatrix}$$

$$P^{-1}=\begin{bmatrix}15 & 2 & 1\\ 2 & 1 & 0\\ 1 & 0 & 0\end{bmatrix}$$ (Strange, that looks like *M* matrix from *CCF*)

No, that's not strange! $T=PM=I$ if you start with *CCF*, so $M=P^{-1}$ better hold!

$$q(A)=A^3+16A^2+39.55A+53.26I=\begin{bmatrix}35.26 & 24.55 & 14.00\\ -252.00 & -174.74 & -3.45\\ 62.10 & -200.25 & -167.84\end{bmatrix}$$

$$K = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} P^{-1} q(A) = \begin{bmatrix} 35.26 & 24.55 & 14.00 \end{bmatrix}$$

Agrees with the decoupled CCF solution; for SISO systems the solution for *K* is unique.

6.3.3 MIMO Pole Placement

Matlab place command. Investigate the algorithm. Present (simple) MIMO theory.

6.4 Correction of Output Attenuation

As seen in the controller design simulation results for our example (Figure 8.?), the feedback matrix *K* has achieved the control of transient response as desired. However, the closed-loop steady-state value (assuming a step input) has been changed from the open-loop case, which is undesirable. Therefore we must develop a method so the closed-loop controller yields the original steady-state value (or some specified value). In classical controls, this is referred to as the DC gain.

This phenomenon is called output attenuation; the closed-loop controller provides a virtual spring, usually stiffer than the original open-loop system spring (some open-loop systems even have no spring). Therefore, the steady-state response to a step input will be smaller in the closed-loop case than in the original open-loop case. Let us assume that the steady-state closed-loop response should be identical to the original open-loop steady-state response. This section provides a method to ensure this.

First we study this problem for SISO systems and provide a simple method to correct it (DC gain). Then we extend the simple DC gain approach to MIMO systems. Last we present a more general MIMO output attenuation correction method. This discussion applies to step inputs and controllers without Type I integrators which automatically correct steady-state error problems.

<u>One-dof translational mechanical system:</u>

$m\ddot{y} + c\dot{y} + ky = F$ For constant *F*, $y_{ss} = \dfrac{F}{k}$ ($\ddot{y} = \dot{y} = 0$ at steady-state)

<u>Multiple-dof translational mechanical system:</u> ([*K*] is stiffness, not feedback matrix!)

$[M]\{\ddot{Y}\} + [C]\{\dot{Y}\} + [K]\{Y\} = \{F\}$ For constant {*F*}, $\{Y_{ss}\} = [K]^{-1}\{F\}$ ($\{\ddot{Y}\} = \{\dot{Y}\} = 0$)

<u>State-space description</u>

At steady-state, $\dot{X} = 0 = AX + BU$ so $\{X_{ss}\} = -[A]^{-1}\{B\}\{U\}$

<u>Example</u>

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad u{=}1 \text{ (unit step)}$$

$$\{X_{ss}\} = -\begin{bmatrix} -0.833 & -0.111 & -0.056 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\{1\} = \begin{Bmatrix} 0.056 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} x_{1ss} \\ x_{2ss} \\ x_{3ss} \end{Bmatrix} = \begin{Bmatrix} y_{ss} \\ \dot{y}_{ss} \\ \ddot{y}_{ss} \end{Bmatrix}$$

Check $y_{ss} = \dfrac{F}{k} = \dfrac{1}{18} = 0.056$; also agrees with the simulation results of Figure 8.?. Now,

$$A_c = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -53.26 & -39.55 & -16 \end{bmatrix} \qquad\qquad B_c = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$y_{ss} = \dfrac{F}{k} = \dfrac{1}{53.26} = 0.019$ (agrees with Figure 8.2?, closed-loop response), but we want 0.056. So for the closed-loop system with state feedback controller, we must modify the reference input *r*:

$r = u * corr$ where *u* is the open-loop input magnitude and *corr* is the correction factor. For SISO, this correction factor is a simple ratio:

$$corr = \frac{A_c(3,1)}{A(3,1)}$$

This is the ratio of the effective closed-loop system stiffness with controller to the actual open-loop system stiffness.

For the example, $r = u\dfrac{A_c(3,1)}{A(3,1)} = (1)\dfrac{-53.26}{-18} = 2.96$

This is the reference input (DC gain-adjusted) is used to achieve the plot of Figure 8.?, with the desired steady-state value.

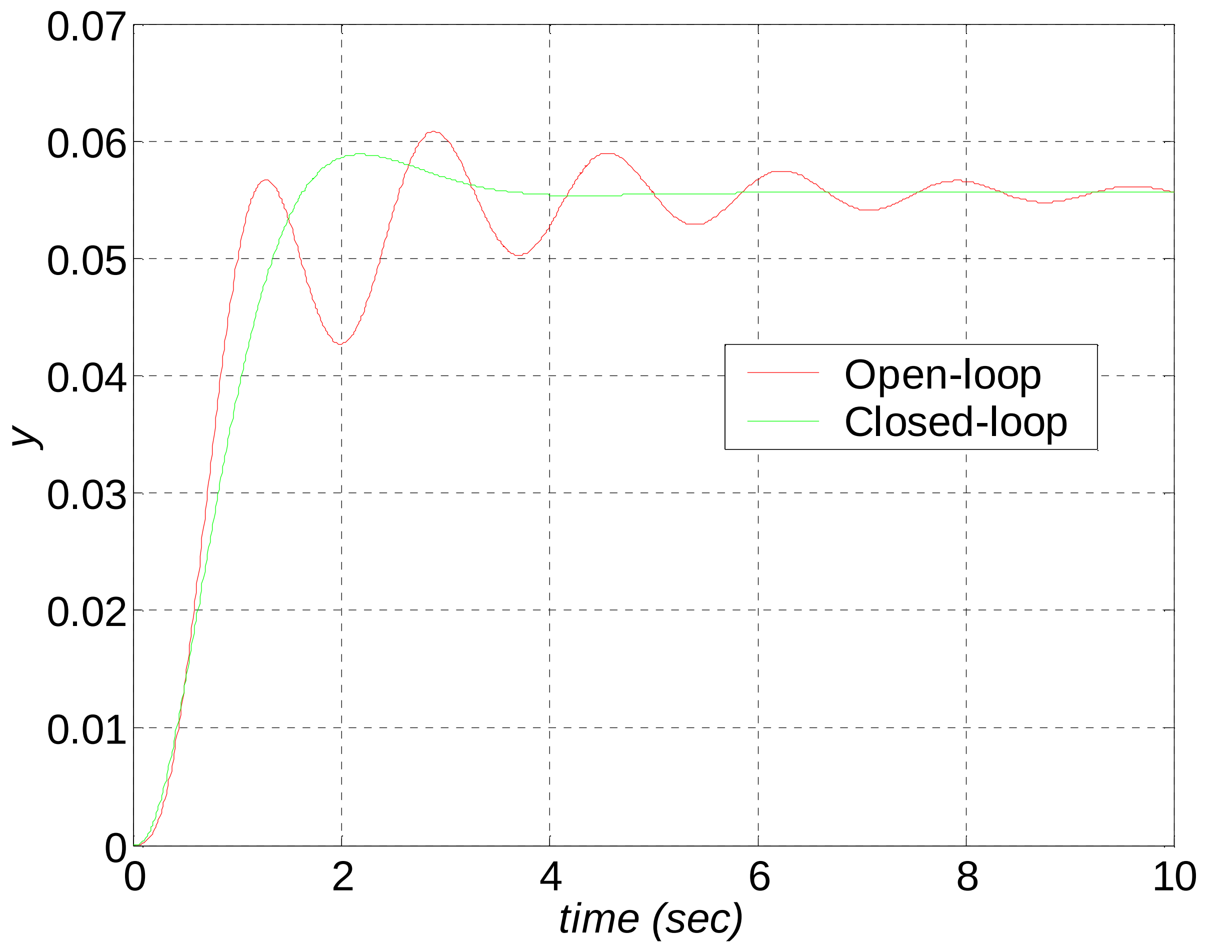

Figure 8.? Open- vs. Closed-Loop Output Response for Example, Corrected

For MIMO systems, we can extend this simple DC gain concept, in order to compare in simulation closed-loop controller system behavior with the original open-loop system behavior. First, determine the steady-state values for the closed loop system using:

$$\dot{X}_{ss} = 0 = A_c X_{ss} + B_c u$$
$$\therefore X_{ss} = -A_c^{-1} B_c u$$

where u is a rx1 vector containing the magnitudes of the r input step functions used for open-loop simulation. For 2nd-order mechanical systems, the closed-loop steady-state displacements will be attenuated compared to the open-loop steady-state displacements, and the steady-state velocities will be 0. Next, simply define an array of displacement correction factors, term-by-term division of the open-loop steady-state displacements by the closed-loop steady-state displacements:

$$corr_i = \frac{y_{sso_i}}{y_{ssc_i}}$$

Finally, when plotting the closed-loop responses, simply multiply all state components (including velocities) by the appropriate correction factor component, in order to compare simulated open- and closed-loop behaviors to the same steady-state values.

Now, this method is good for simulations and open-/closed-loop comparisons, but it is not general for real-world controllers. Therefore, we will now present a more general approach with a slightly modified control law wherein the closed-loop output values will be driven to any desired level with (theoretically) zero steady-state error.

Our existing closed-loop full-state-feedback control law is $u(t) = r(t) - Kx(t)$; the MIMO attenuation method uses the modified control law:

$$u(t) \quad = \quad Nr(t) - Kx(t)$$

where *N* is a constant output attenuation matrix, calculated by:

$$N = -\left[C\left(A - BK \right)^{-1} B \right]^{-1}$$

The matrix $C\left(A - BK \right)^{-1} B$ has dimension *m*x*r*; in order to take the plain inverse of this matrix, it must be square (*r*=*m*; i.e. the number of inputs must match the number of outputs). Using the modified control law, the matrix *N* will guarantee that the closed-loop steady-state output will match that of the reference input *r*(*t*), when simulating the modified closed-loop system:

$$\begin{aligned} \dot{x}(t) &= \left(A - BK \right) x(t) + BNr(t) \\ y(t) &= Cx(t) \end{aligned}$$

That is, we use the familiar $A_c = \left(A - BK \right)$ and a modified $B_c = BN$ (where earlier $B_c = B$). Again, *C* and *D* do not change from the open- to the closed-loop cases.

An example for this output attenuation correction method is given in Section 6.6.2.1, Case i.

Now, requiring the number of inputs to match the number of outputs, *r*=*m*, is a significant restriction on general MIMO systems, where in general $r \neq m$. So we can easily adapt the above procedure by using the Moore-Penrose pseudoinverse in place of the plain square matrix inverse in (??). The pseudoinverse approach will work for all cases, i.e. $r > m, \quad r = m, \quad r < m$.

In this more general case, to make the dimensions work in (??), *r*(*t*) must have the dimension *m*x1 (not *r*x1 as before), since the dimension of *N*, found by the pseudoinverse, is *r*x*m*. So, now *r*(*t*) plays the role of reference output, and should contain the desired steady-state output values for each output component. The modified control law (??) with pseudoinverse-calculated *N* will then yield output steady-state values identical to *r*(*t*). Note that in simulation of the resulting closed-loop system, although *D* is a zero matrix, its dimension must be changed to *m*x*m* (from the original *m*x*r*) to work with the new *BN* and *r*(*t*).

6.5 Stabilizability

Related to controllability, not as strong.

6.6 Matlab for Shaping Dynamic Response and Controller Design

6.6.1 Matlab for Shaping Dynamic Response

The Matlab functions that are useful for dynamic shaping (determine desired controller performance based on determining *n* closed-loop controller poles) are discussed in the Matlab sections of Chapters 1-3. With either dynamic shaping method (dominant/augmented poles and ITAE poles), a useful Matlab capability is provided in conjunction with the step function.

```
figure;
step(numDes,denDes);
```

Where *denDes* are the *n*+1 coefficients of the *n*th-order desired closed-loop characteristic polynomial (which the engineer must determine based on dominant/augmented poles or ITAE poles) and *numDes* is the constant desired behavior numerator, that can be chosen to normalize the final steady-state unit step response value to 1.0. After running the step function with the desired system, one can right-click with the mouse in the figure window to automatically display on the plot the performance measures (rise time, peak time, percent overshoot, and settling time). Matlab determines these values numerically from the response data by applying the rules for each, i.e. they should be accurate even for non-2nd-order systems.

<u>Continuing Matlab Example: Shaping Dynamic Response</u>

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), determine two desired poles for controller design (Chapter 8) to improve the performance relative to the open-loop responses of Figure 3.3. Use a desired percent overshoot and settling time of 3% and 0.7 sec, respectively to determine the desired 2nd-order controller poles. The following Matlab code performs this dynamic shaping for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 4.  Dynamic Shaping
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PO = 3;  ts = 0.7;                               %  Specify percent overshoot and settling time
zeta = log(PO/100)/sqrt(pi^2 + log(PO/100)^2)    % Damping ratio from percent overshoot
wn   = 4/(zeta*ts);                              % Natural frequency from settling time and zeta
num2   = wn^2;                                   % Generic desired 2nd-order system
den2   = [1 2*zeta*wn wn^2];
Poles2 = roots(den2);                            % Desired controller poles

figure;
td = [0:0.01:1.5];
step(num2,den2,td);                              % For right-clicking to place performance measures
```

This m-file generated the following results for desired 2nd-order closed-loop ξ, ω_n, characteristic polynomial, and poles. It also generated the desired closed-loop response shown in Figure

4.6, with the performance specifications displayed via right-clicking. We see that the 3% overshoot is achieved exactly in theory, while the settling time is close to the 0.7 desired since the settling time equation is approximate.

```
zeta =
   0.7448

wn =
   7.6722

den2 =
   1.0000   11.4286   58.8627

Poles2 =
  -5.7143 + 5.1195i
  -5.7143 - 5.1195i
```

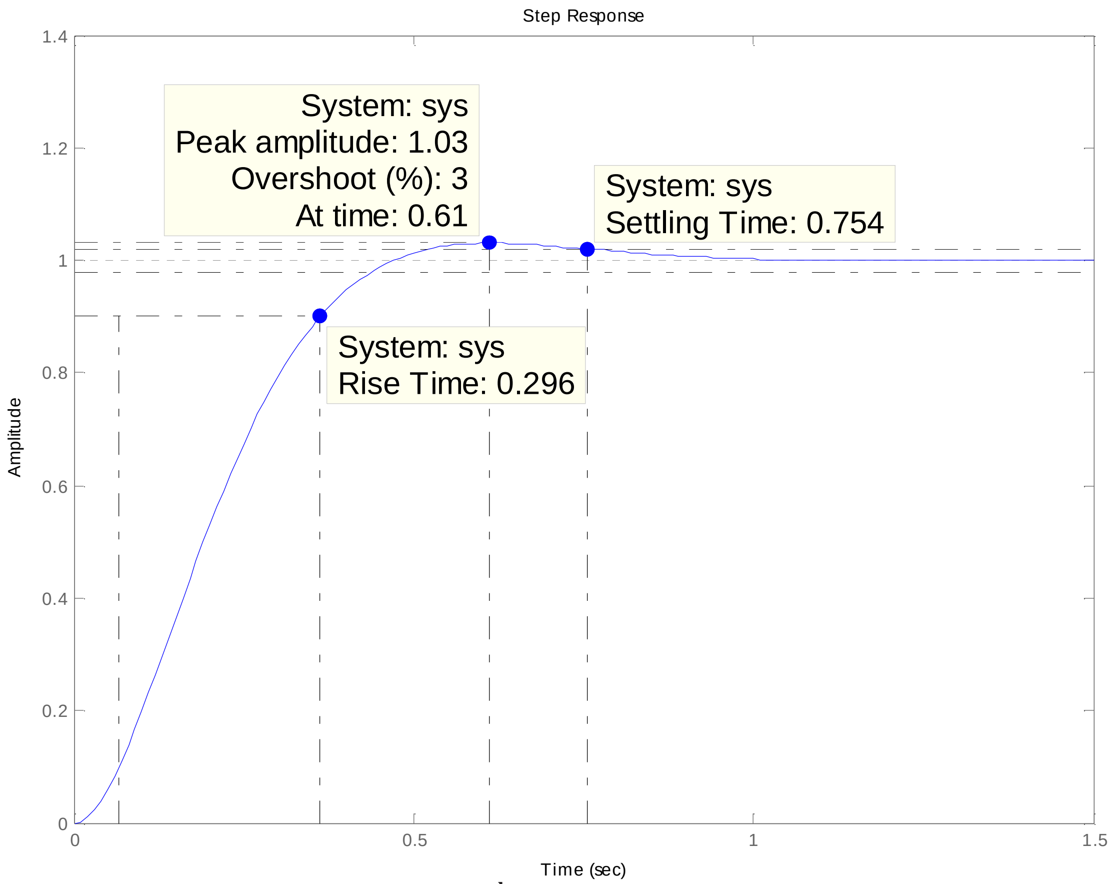

Figure 4.6 Desired 2nd-order Closed-Loop Response

6.6.2 Matlab for Controller Design and Evaluation

The following Matlab functions are useful for design of linear state-feedback controllers:

place(*A*,*B*,*DesPoles*)	Solve for full-state-feedback gain matrix **K** to place the desired controller poles *DesPoles* into closed-loop dynamics matrix $\mathbf{A_c}$=**A–BK**.
acker(*A*,*B*,*DesPoles*)	Solve for full-state-feedback gain matrix **K** to place the desired controller poles *DesPoles* into closed-loop dynamics matrix $\mathbf{A_c}$=**A–BK**, using Ackerman's formula, for SISO systems only.
conv	Multiply factors to obtain a single polynomial.

Continuing Matlab Example: Design of Linear State-Feedback Controllers

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), design the full-state-feedback controller, i.e. determine controller gain matrix **K** given **A**, **B**, and the desired controller poles developed in the Chapter 4 continuing Matlab example. The following Matlab code performs this controller design for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 8.  Design of Linear State-Feedback Controllers
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


K      = place(A,B,Poles2);           % Compute full-state feedback controller gain matrix K
Kack = acker(A,B,Poles2);             % For SISO we can check K via Ackerman's formula

Ac = A-B*K;  Bc = B;  Cc = C;  Dc = D;   % Compute the closed-loop state-feedback system

% Compare open-loop and closed-loop responses: same zero input U, t, and ICs from open-loop above
[Yc,Xc] = lsim(Ac,Bc,Cc,Dc,U,t,X0);

figure;
subplot(211), plot(t,Xo(:,1),'r',t,Xc(:,1),'g'); grid; axis([0 4 -0.2 0.5]);
set(gca,'FontSize',18);
legend('Open-loop','Closed-loop');
ylabel('{\itx}_1')
subplot(212), plot(t,Xo(:,2),'r',t,Xc(:,2),'g'); grid; axis([0 4 -2 1]);
set(gca,'FontSize',18);
xlabel('\ittime (sec)'); ylabel('{\itx}_2');
```

This m-file, combined with the previous chapter m-files, yielded the following output (*place* and *acker* yielded identical results for **K**), plus the comparison of open- vs. closed-loop state responses shown in Figure 8.2.

```
K =
18.86       7.43
```

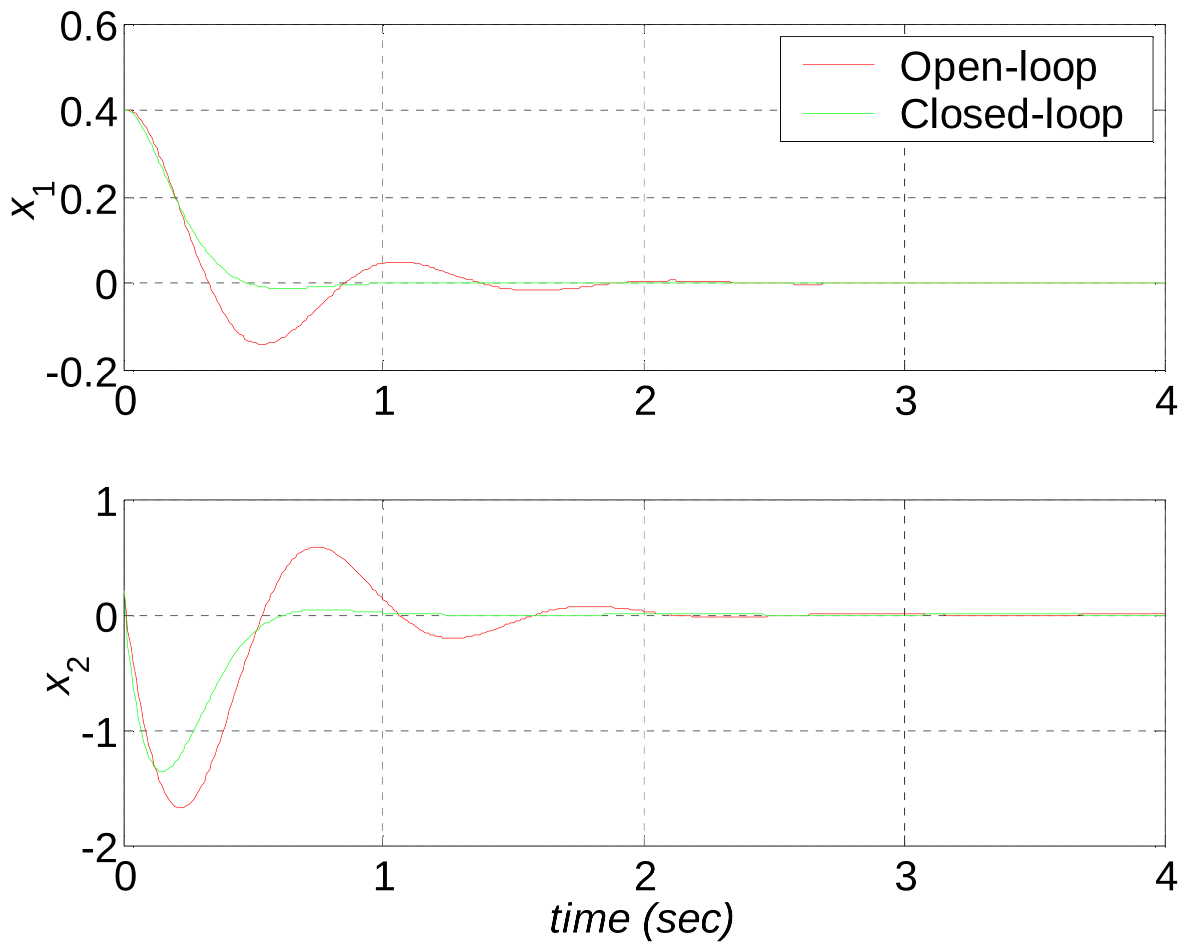

Figure 8.2 Open- vs. Closed-Loop State Responses for Matlab Example

Figure 8.2 shows that the simulated closed-loop system states perform better than the open-loop responses, in terms of lower overshoots and faster rise and settling times. There is less vibration with the controller responses and the steady-state zero values are obtained sooner than in the open-loop case. Closed-loop system simulation can also be performed using Matlab's Simulink. Figure 8.3 shows the high-level Simulink diagram (Figure 3.3b shows the detailed diagram for the ABCD Open-loop blocks; in the closed-loop case, another output is required, **X** for state feedback). To run these diagrams for this example, the step input was again set to zero (torque) and the initial conditions were set to those given. The scope shows the output θ plot, identical to the upper plot of Figure 8.2.

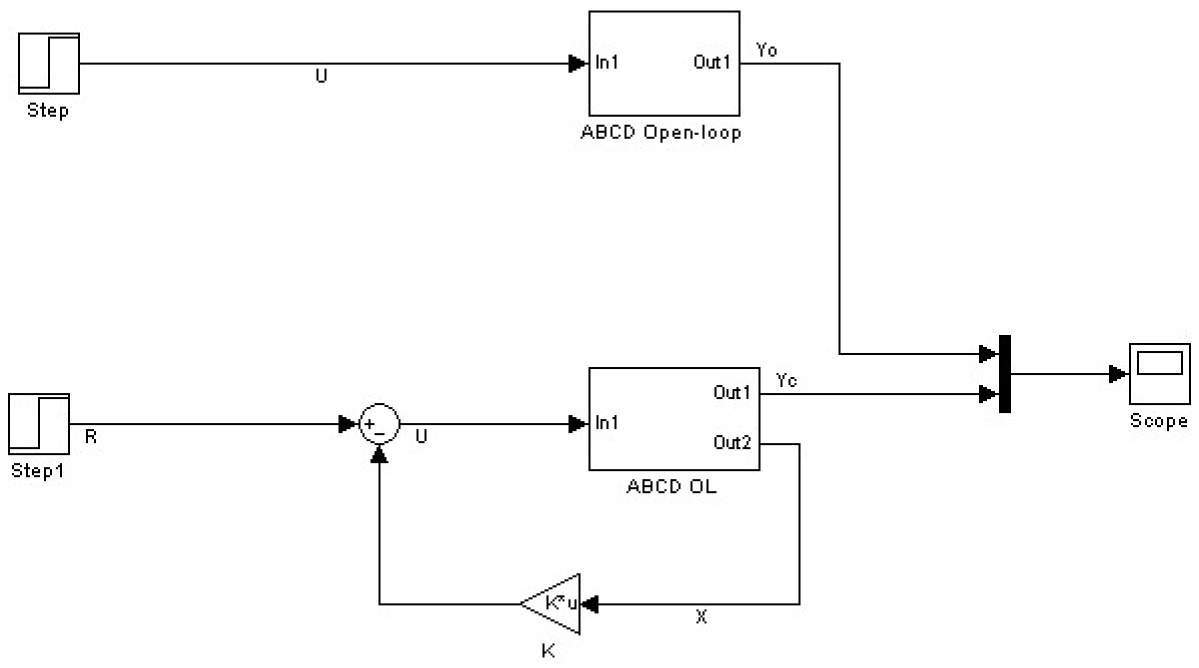

Figure 8.3 Simulink Diagram for Open- vs. Closed-Loop Response

6.7 Continuing Examples: Shaping Dynamic Response and Controller Design

6.7.1 Shaping Dynamic Response

6.7.1.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

For controller design, if we don't like the original open-loop system behavior, we need to specify desired poles which would improve the system performance. There are infinite possibilities; in this example we will consider two distinct methods to be applied later in controller design for Continuing Example I:

- Use a dominant 2nd-order system for desired poles to achieve 5% overshoot and 2 *sec* settling time; augment these two poles with two additional non-dominant poles (we need 4 desired poles for our 4th-order system): real, negative, and at least ten times higher. We will use this specification of poles (arbitrarily) for the full MIMO Case i.
- Use a 4th-order ITAE approach for desired poles, with the same natural frequency ω_n as Case i for easy comparison. We will use this specification of poles (arbitrarily) for the SISO Case ii (input u_2 and output y_1).

<u>Solution, Case i</u>

Percent overshoot is only a function of ξ; substituting the desired 5% overshoot yields dimensionless damping ratio $\xi = 0.69$; with this value, plus the desired 2 *sec* settling time, we then find natural frequency $\omega_n = 2.90$ *rad/s*. This yields a desired dominant generic 2nd-order transfer function:

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} = \frac{8.40}{s^2 + 4s + 8.40}$$

whose dominant, complex conjugate poles are $s_{1,2} = -2 \pm 2.10i$. The response of this desired dominant 2nd-order system is shown in Figure 4.7, complete with the 2nd-order performance measures. This figure was produced using the Matlab step function with the above 2nd-order desired numerator and denominator and then right-clicking to add the performance measures. As seen in Figure 4.7, we have obtained the desired 5% overshoot (at a peak time of 1.5 *sec*) and a settling time of 2.07 (2 *sec* was specified). The 10% to 90% rise time is 0.723 *sec*.

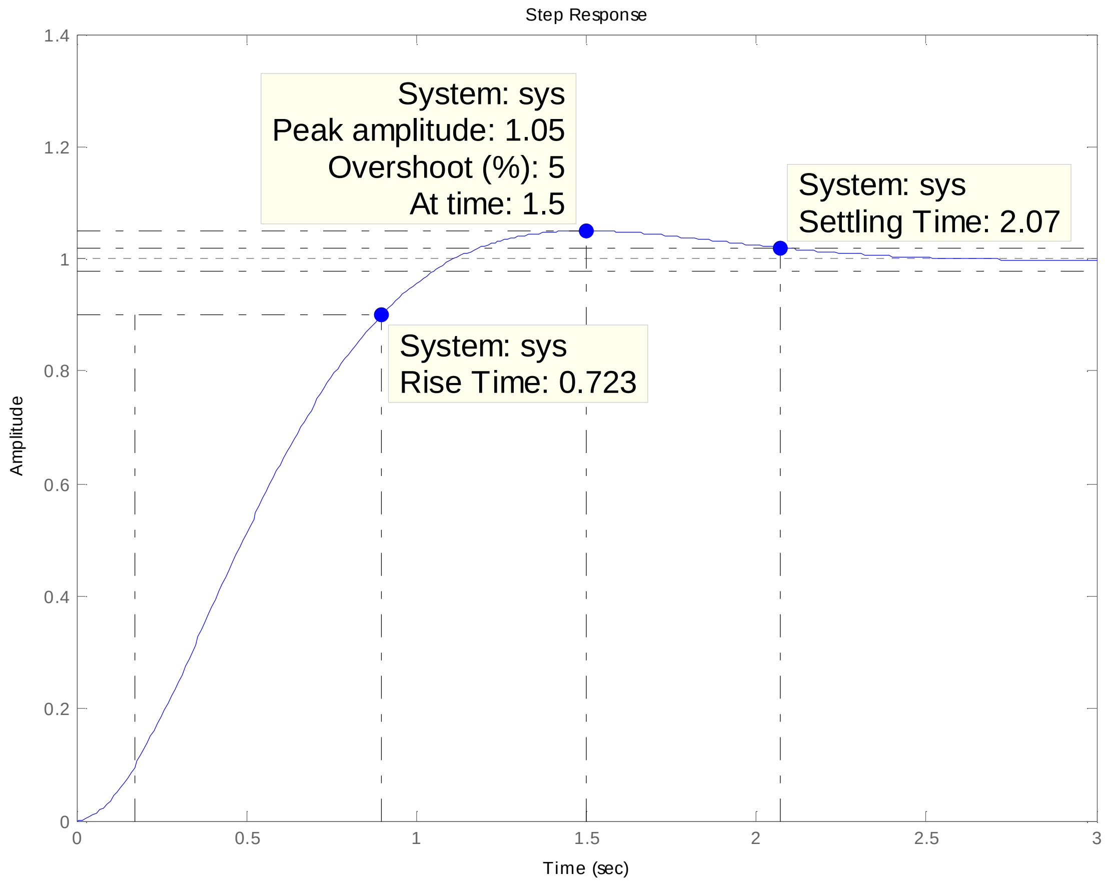

Figure 4.7 Dominant 2nd-Order Response with Performance Measures

Let us augment the dominant 2nd-order poles to fit our 4th-order system desired poles requirements as follows (the real, negative, at least 10x higher poles will not change the dominant behavior much):

$$s_{1,2} = -2 \pm 2.10i \qquad\qquad s_{3,4} = -20,-21$$

Note we do not specify repeated poles for $s_{3,4}$ because that can lead to numerical problems. The transfer function for the 4th-order desired behavior mimicking 2nd-order behavior is (normalizing for a steady-state value of 1):

$$G_4(s) = \frac{3528}{s^4 + 45s^3 + 592s^2 + 2024s + 3528}$$

We will wait to plot this augmented 4th-order desired response until the following subsection, where we will compare all responses on one graph.

Solution, Case ii

For a 4th-order system, the optimal (a balance between the competing response time and error) ITAE characteristic polynomial is:

$$\Delta_{ITAE4}(s) = s^4 + 2.1\omega_n s^3 + 3.4\omega_n^2 s^2 + 2.7\omega_n^3 s + \omega_n^4$$

In this example we will use the same natural frequency from above, i.e. $\omega_n = 2.90 \ rad/s$:

$$\Delta_{ITAE4}(s) = s^4 + 6.09s^3 + 28.56s^2 + 65.72s + 70.54$$

For this 4th-order desired characteristic polynomial, the four desired poles are:

$$s_{1,2} = -1.23 \pm 3.66i \qquad\qquad s_{3,4} = -1.81 \pm 1.20i$$

Figure 4.8 plots the 4th-order ITAE desired response, along with the dominant 2nd-order and augmented 4th-order desired responses from the Case i example above. All are normalized to a steady-state value of 1.0 for easy comparison.

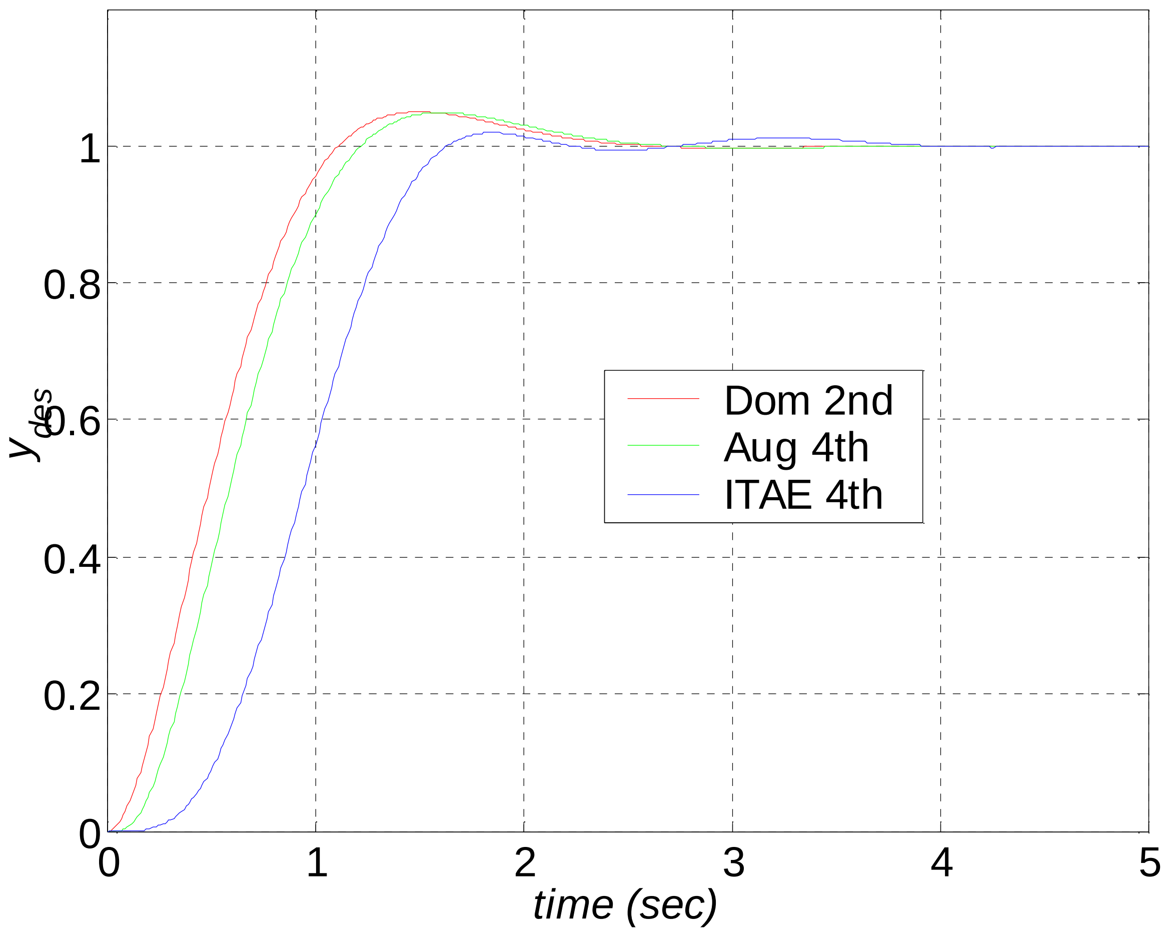

Figure 4.8 Dynamic Shaping Example Results

From Figure 4.8 we see that the augmented 4th-order response (green) mimics the dominant 2nd-order response (red) closely, as desired. The augmented 4th-order response lags the dominant 2nd-order response, but it matches the required 5% overshoot and 2 *sec* settling time well. The 4th-order ITAE response (blue) did not specify percent overshoot or settling time, but we used the same natural frequency as in the dominant 2nd-order response for comparison purposes. The ITAE response lags even further and demonstrates a 4th-order wiggle not present in the augmented 4th-order response.

6.7.1.2 Continuing Example II: Rotational Electromechanical System

For controller design, if we don't like the original open-loop system behavior, we need to specify desired poles which would improve the system performance. There are infinite possibilities; in this example we will use a desired dominant 1st-order system, to be applied later in controller design for Continuing Example II. Use a dominant 1st-order system with time constant τ = ¼ *sec*. Augment this pole with two additional non-dominant poles (we need 3 desired poles for our 3rd-order system): real, negative, and higher so their effect is not seen strongly.

Solution

The relationship between desired dominant pole a and 1st-order time constant τ is $e^{at} = e^{-t/\tau}$; therefore, $a = -1/\tau = -4$. This yields the desired dominant generic 1st-order transfer function:

$$G(s) = \frac{-a}{s-a} = \frac{4}{s+4}$$

Let us augment the dominant 1st-order pole to fit our 3rd-order system desired poles requirement as follows (the real, negative, higher poles should not change the dominant behavior much); we choose additional poles about three times higher:

$$s_{1,2,3} = -4, -12, -13$$

Note we do not specify repeated poles for $s_{2,3}$ because that can lead to numerical problems. The transfer function for the 3rd-order desired behavior mimicking 1st-order behavior is (normalizing for a steady-state value of 1):

$$G_3(s) = \frac{624}{s^3 + 29s^2 + 256s + 624}$$

Figure 4.9 plots the augmented 3rd-order desired response for controller design in a future example, along with the dominant 1st-order system it was derived from. Both are normalized to a steady-state value of 1.0 for easy comparison.

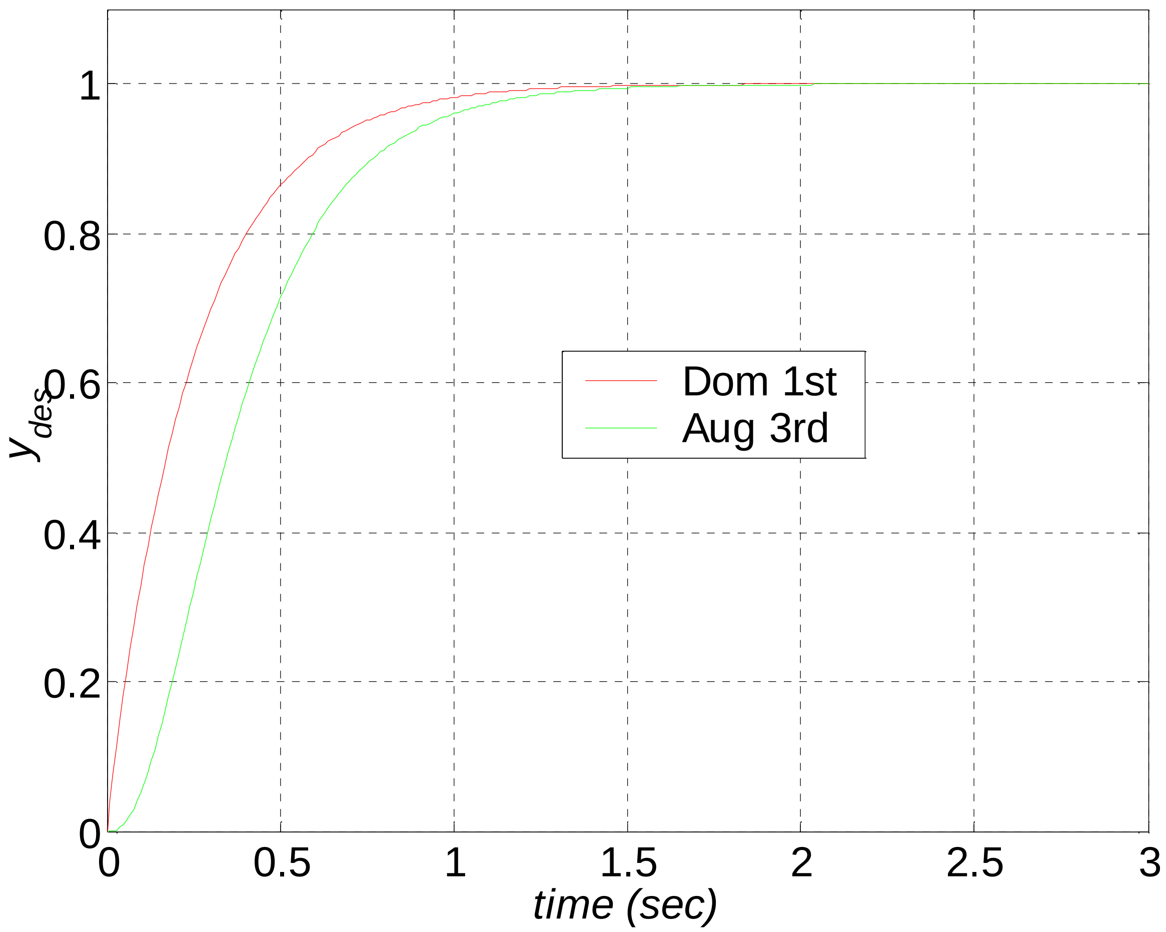

Figure 4.9 Dynamic Shaping for Example II

From Figure 4.9 we see that the augmented 3rd-order response (green) mimics the dominant 1st-order response (red) fairly closely, as desired. We see in the red curve that after three time constants (at t=0.75 sec), the dominant 1st-order response has achieved 95% of the final steady-state value of 1. The augmented 3rd-order response lags the dominant 1st-order response; we can make this arbitrarily close to the red curve, by adding higher poles, much higher than the 3x chosen. However, this may lead to large numerical values for controllers and observers, which is generally to be avoided. This would correspond to high required actuator values, perhaps exceeding physical limits.

6.7.2 Controller Design and Evaluation

6.7.2.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

For both Cases of Continuing Example I (Two-mass MIMO Translational Mechanical System), determine a full-state-feedback gain matrix **K** to move the closed-loop system eigenvalues where desired and hence achieve the control objectives. In both cases, simulate the closed-loop behavior and compare the open-loop system responses with those of the closed-loop system.

- For Case i (full MIMO), design the controller based on the desired poles developed in the Chapter 4 Case i solution and simulate the closed-loop system response given the same conditions as the open-loop simulation of Chapter 3 (zero initial conditions and step inputs of magnitudes 20 and 10 *N*, respectively, for u_1 and u_2).
- For Case ii (SISO, input u_2 and output y_1), design the controller (it will obviously be different from the Case i **K** above, it will not even be of the same size) based on the desired poles developed in the Chapter 4 Case ii solution and simulate the closed-loop system response given the same conditions as the open-loop simulation of Chapter 3 (zero input u_2 and initial conditions $\mathbf{X}(0)=\{0.1 \quad 0 \quad 0.2 \quad 0\}^T$).

<u>Solution, Case i</u>

The control law is $\mathbf{U}=\mathbf{r}-\mathbf{KX}$. The closed-loop system dynamics matrix we will place the desired poles into via **K** is $\mathbf{A_c}=\mathbf{A}-\mathbf{BK}$. From Chapter 4, the four desired poles for this case are attempting to provide 5% overshoot and 2 sec settling time:

$$s_{1,2}=-2\pm 2.10i \qquad\qquad s_{3,4}=-20,-21$$

By using the Matlab function *place* we found the 2x4 full-state-feedback gain matrix **K** to be:

$$\mathbf{K}=\begin{bmatrix} 606 & 858 & 637 & 47 \\ -3616 & -166 & 759 & 446 \end{bmatrix}$$

Upon checking, the eigenvalues of $\mathbf{A_c}=\mathbf{A}-\mathbf{BK}$ are indeed those specified in the *place* command. For the reference input **r** we take the same as the open-loop input **U**, i.e. step inputs of magnitudes 20 and 10 *N*, respectively, for u_1 and u_2. Simulating the closed-loop system response and comparing it to the open-loop system response yields Figure 8.4.

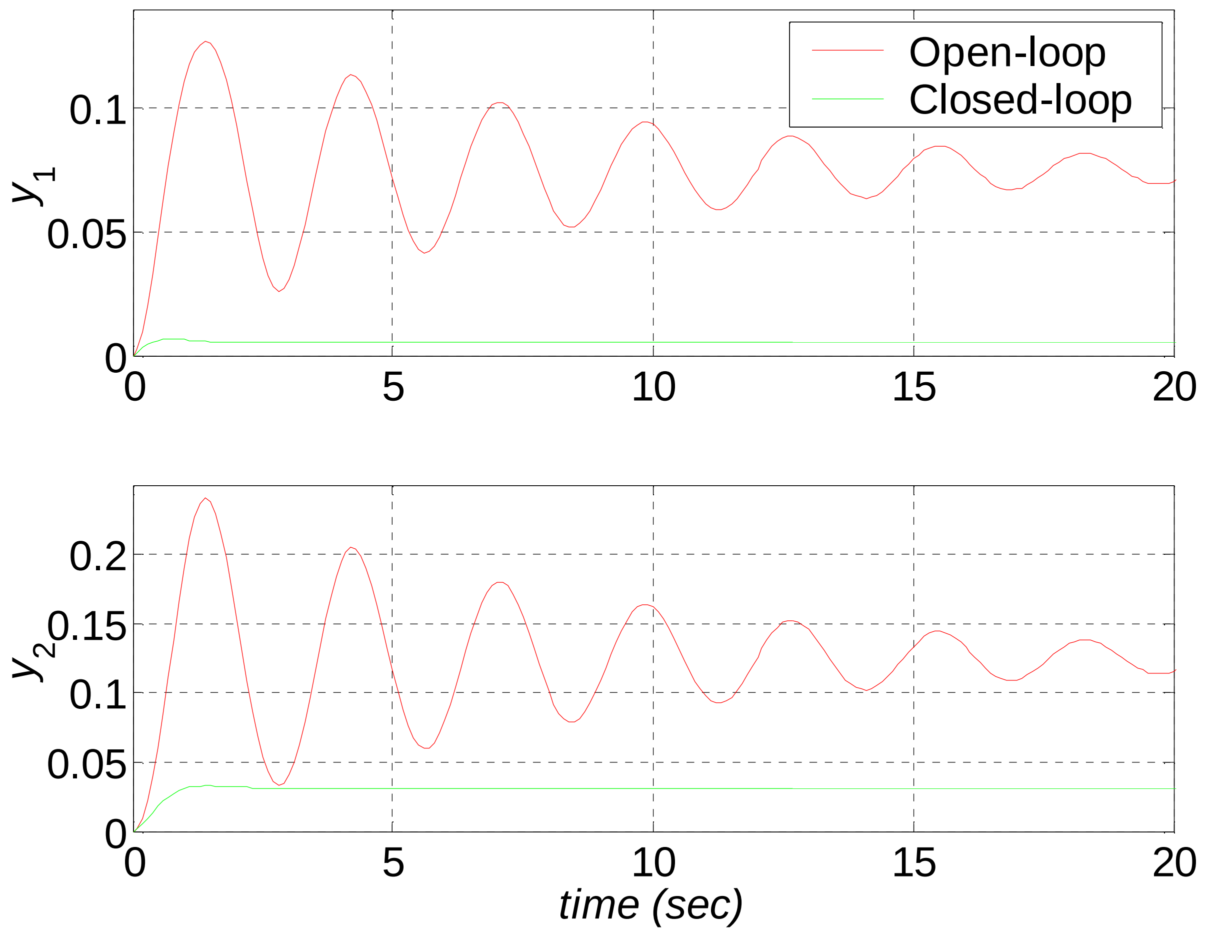

Figure 8.4 Open- vs. Closed-loop Responses for Case i, with output attenuation

The first thing evident in Figures 8.4 is that the closed-loop controller attenuates the output (see Section ??). This is because the controller adds virtual springs in addition to the 'real' springs in the system model; a stiffer set of springs will cause the output attenuation seen in Figures 8.4. Before we can discuss the performance of the controller, we must ensure the level of the closed-loop responses match those of the original open-loop system. This can be done in two ways: a. The output attenuation correction factors (simple term-by-term DC gains) are 14.3 and 3.99 for outputs y_1 and y_2, respectively. The corrected closed-loop responses following this approach are shown in Figures 8.5a. b. Using the modified control law with attenuation matrix N and reference input equal to the open-loop steady state values ($r = \begin{bmatrix} 0.075 & 0.125 \end{bmatrix}^T$), results in the corrected closed-loop responses of Figures 8.5b. The correction matrix N is:

$$N = \begin{bmatrix} 1206 & 437 \\ -3816 & 959 \end{bmatrix}$$

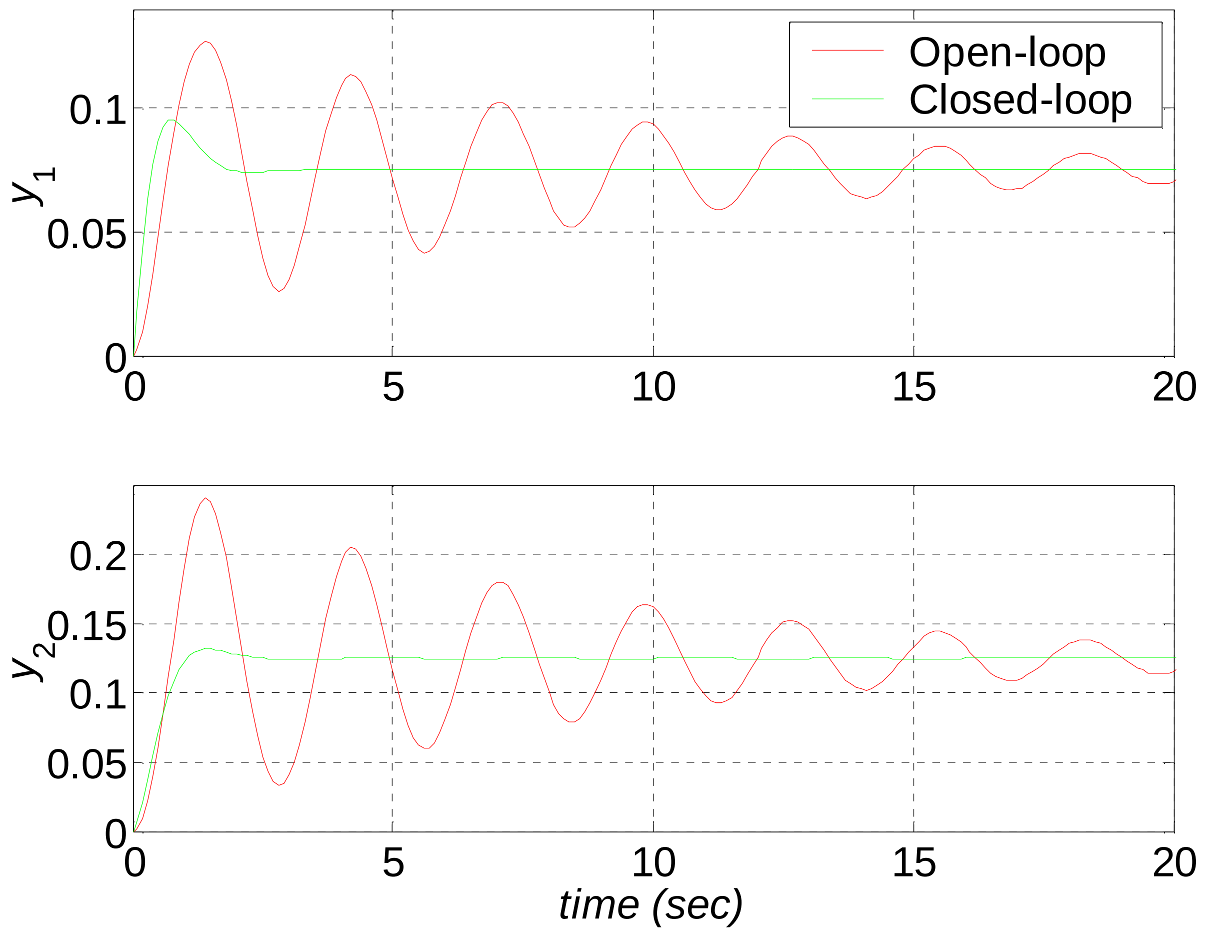

Figure 8.5a Open- vs. Closed-loop Responses for Case i, corrected via DC Gains

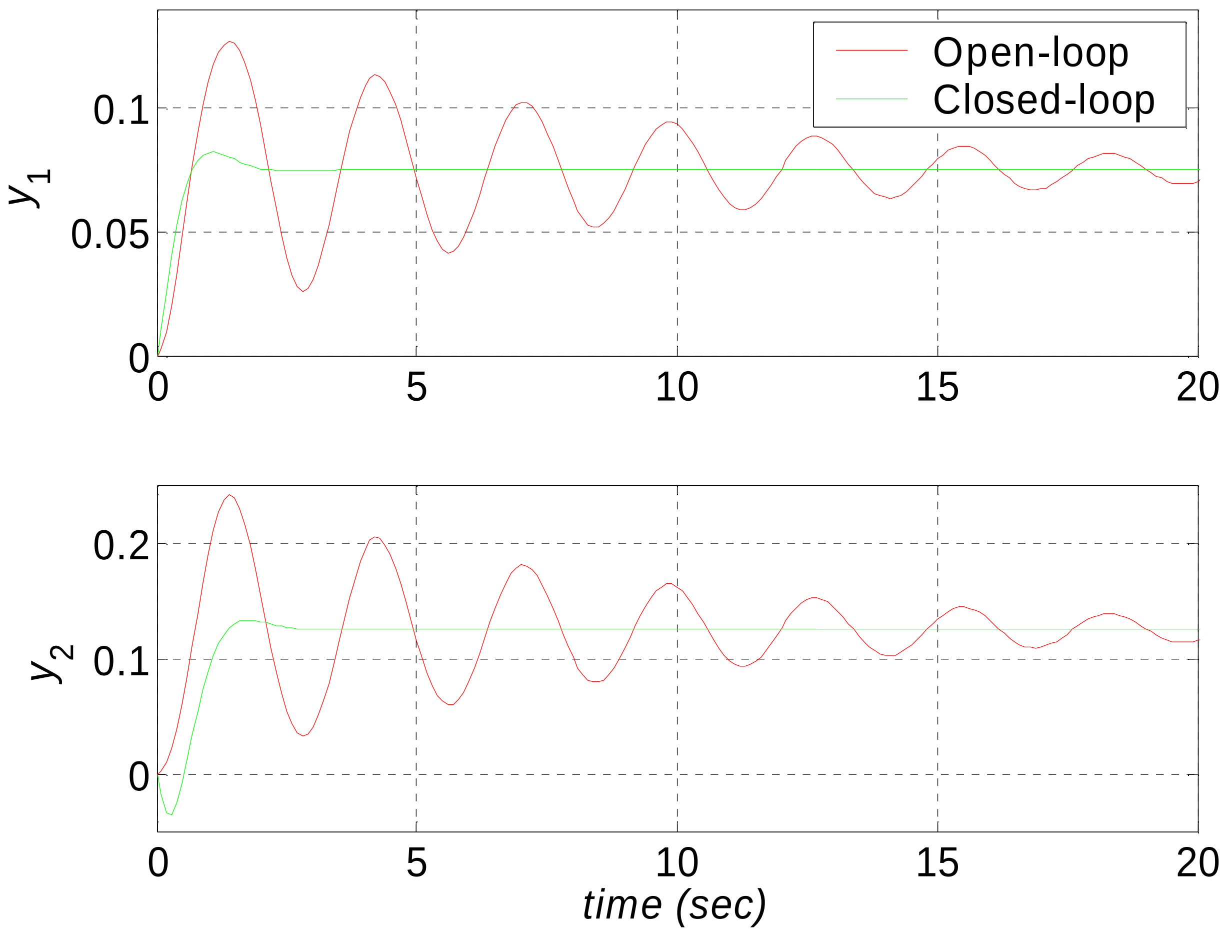

Figure 8.5b Open- vs. Closed-loop Responses for Case i, corrected via *N*

Now we see in both Figures 8.5a and 8.5b that the designed controller has improved the performance of the open-loop system significantly. The settling time of 2 *sec* has been achieved so that the closed-loop system responses meet their steady state values much sooner than those of the open-loop system. However, in the top graph of Figure 8.5a we see that the percent overshoot is much greater than the specified 5%. This was not visible in Figure 8.4 top due to scale. This is a well known problem from classical controls – according to the matrix of transfer functions presented in the Chapter 3, there are zeros (numerator roots) present in this system. The dominant 2nd-order pole specification method does not admit any zeros, thus the results are skewed in the presence of zeros. In classical controls, the way to handle this is pre-shaping the input via filters. HOW TO FIX IN STATE-SPACE??!!??

Note that this overshoot problem is less when using the more general *N*-method shown in Figure 8.5b. However, for the velocity y_2, there is a negative overshoot before it attains the desired value of 0.125.

Solution, Case ii

The control law and closed-loop system dynamics matrix are identical to Case i above. From Chapter 4, the four desired poles for this case are the optimized 4th-order ITAE poles (with the same closed-loop natural frequency $\omega_n = 2.90 \ rad/s$ as in Case i):

$$s_{1,2} = -1.23 \pm 3.66i \qquad\qquad s_{3,4} = -1.81 \pm 1.20i$$

:
By using the Matlab function *place* we found the 1x4 full-state-feedback gain matrix **K** to be:

$$\mathbf{K} = \begin{bmatrix} -145 & -61 & 9 & 97 \end{bmatrix}$$

Since Case ii is SISO we can check this result using Ackerman's formula (Matlab function *acker*); the results are identical. Upon checking, the eigenvalues of $\mathbf{A_c} = \mathbf{A} - \mathbf{BK}$ are indeed those specified in the *place* command. For the reference input **r** we take the same as the open-loop input **U**, i.e. zero u_2. This Case is driven by initial conditions $\mathbf{X}(0) = \{0.1 \quad 0 \quad 0.2 \quad 0\}^T$. Simulating the closed-loop system state responses and comparing it to the open-loop system state responses yields Figure 8.6.

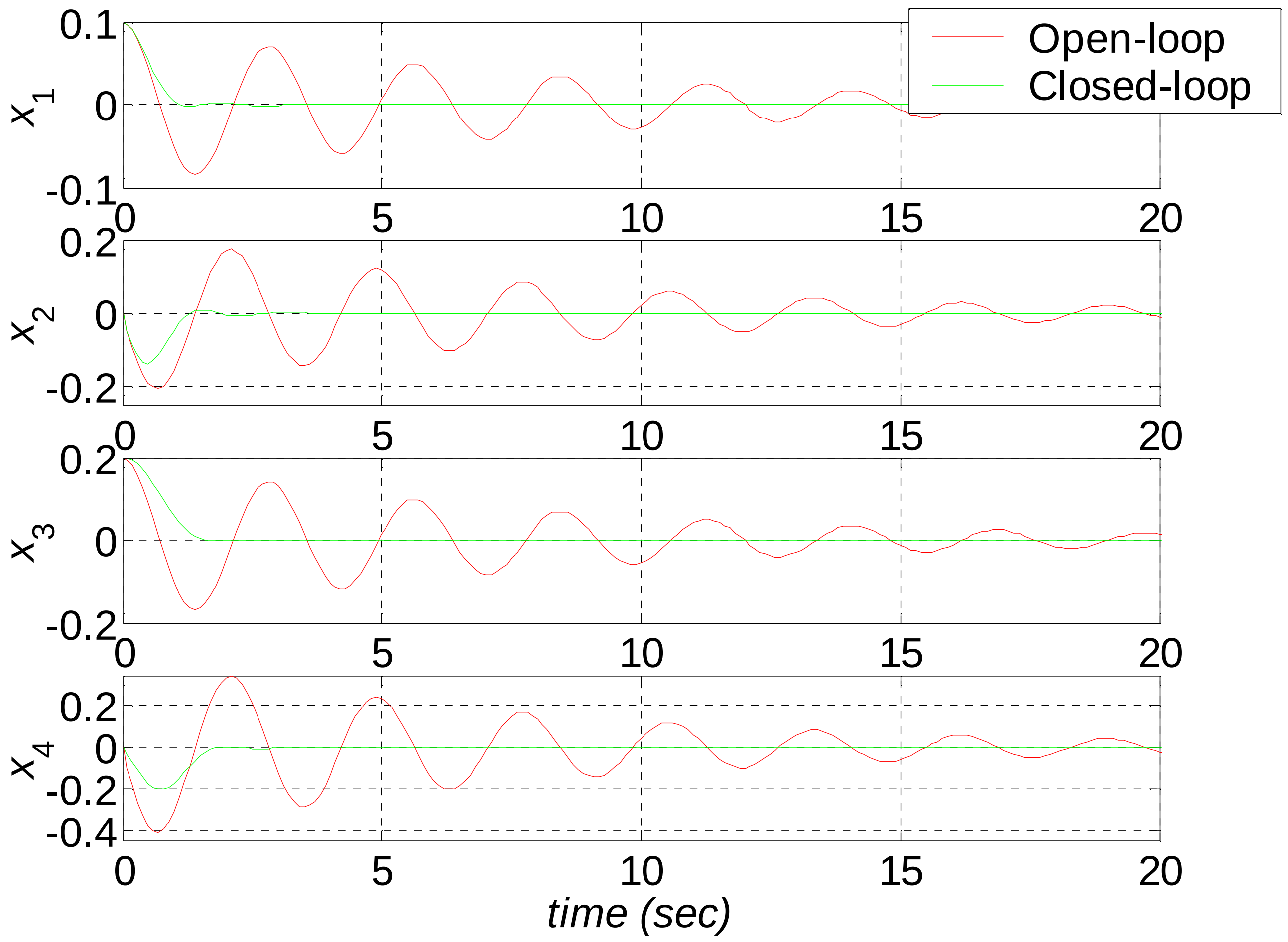

Figure 8.6 Open- vs. Closed-loop Responses for Case ii

Since the closed-loop states all go to zero in the steady-state as the open-loop states do, there is no output attenuation issue. We see in Figures 8.6 that the designed controller has improved the performance of the open-loop system significantly. The closed-loop system responses meet their steady-state zero values much sooner than those of the open-loop system. The displacements do not overshoot significantly and the velocity overshoots are both better than their open-loop counterparts.

6.7.2.2 Continuing Example II: Rotational Electromechanical System

For Continuing Example II (1-dof rotational electromechanical system; SISO: input *v*, output θ), determine a full-state-feedback gain matrix **K** to move the closed-loop system eigenvalues where desired and hence achieve the control objectives. Simulate the closed-loop behavior and compare the open-loop system responses with those of the closed-loop system.

Design the controller based on the desired poles developed in the Chapter 4 Example II solution (dynamic shaping) and simulate the closed-loop system response given the same conditions as the open-loop simulation of Chapter 3 (zero initial conditions and a unit step input in voltage *v*).

Solution

The control law is $\mathbf{U} = \mathbf{r} - \mathbf{KX}$. The closed-loop system dynamics matrix we will place the desired poles into via **K** is $\mathbf{A_c} = \mathbf{A} - \mathbf{BK}$. From Chapter 4, the three desired poles for this case are a dominant 1st-order system augmented by two more real, negative, higher poles:

$$s_{1,2,3} = -4, -12, -13$$

Either by hand, or by using the Matlab functions *place* or *acker*, we find the 1x3 full-state-feedback gain matrix **K** to be:

$$\mathbf{K} = \begin{bmatrix} 312 & 127 & 13 \end{bmatrix}$$

Upon checking, the eigenvalues of $\mathbf{A_c} = \mathbf{A} - \mathbf{BK}$ are indeed those that were specified. For the reference input **r** we take the same as the open-loop input **U**, i.e. a unit step input in voltage *v*. Simulating the closed-loop system response and comparing it to the open-loop system response yields Figure 8.7.

The is no output attenuation issue since the original open-loop response increases linearly after the transient dynamics; this is as expected since there is no torsional spring in the motor model. However, we could use the *N*-method to achieve any desired steady-state output angle value in the closed-loop system.

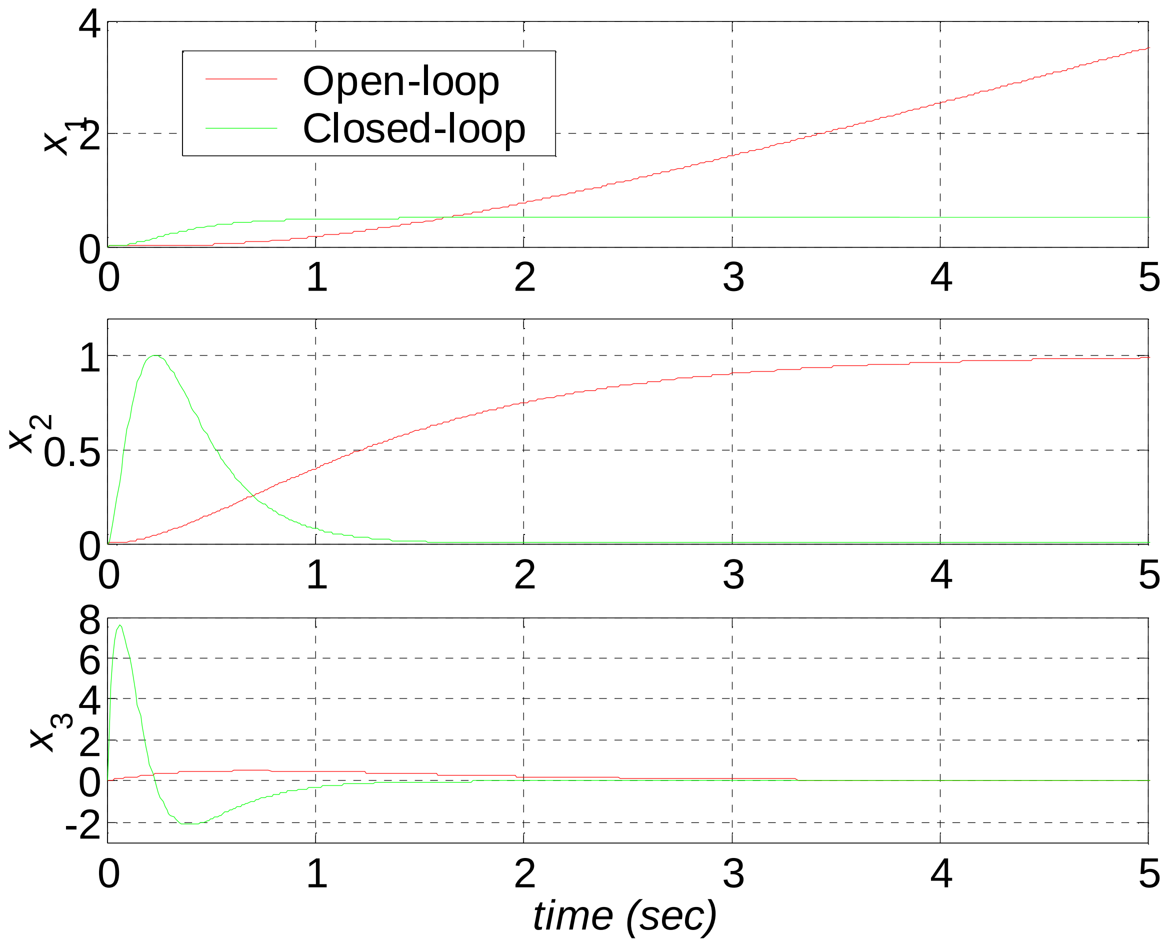

Figure 8.7 Open- vs. Closed-loop Responses for Example II

The closed-loop angle output in Figure 8.7 (top, x_1) was artificially scaled to achieve a steady-state value of 0.5 *rad*. Comparing the open- and closed-loop output motor shaft angle $\theta\,(x_1)$, we see that the full-state feedback controller has effectively added a virtual spring whereby we can servo to commanded angles, rather than having the shaft angle increase linearly without bounds as in the open-loop case. In Figure 8.7, the closed-loop angular velocity and acceleration both experience transient dynamics motion, and then go to zero steady-state values. The open-loop values are identical to those plotted in Figure 3.6 (but plotted to 5 rather than 10 *sec*).

For this example, the closed-loop system dynamics matrix is:

$$\mathbf{A_c} = \mathbf{A} - \mathbf{BK} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -624 & -256 & -29 \end{bmatrix}$$

Now the (3,1) term of $\mathbf{A_c}$ is no longer 0 as it was for the open-loop A; this non-zero term represents the virtual spring of the controller, allowing control to commanded shaft angles. The coefficients of the desired characteristic polynomial can be seen in the third row of $\mathbf{A_c}$, in ascending order of powers of *s*, with negative signs.

The closed-loop case is strictly stable, changed from the marginally stable open-loop system. All three poles are now negative real numbers; now Lyapunov stability analysis would succeed since $\mathbf{A_c}$ is not singular.

6.8 Homework Assignments

6.8.1 Mathematical Homework Assignments

6.8.2 Matlab Homework Assignments

6.8.3 Continuing Homework Assignments

CE1.6a
Given controller design criteria 3% overshoot and 3 *sec* settling time. i) Calculate the natural frequency and damping ratio required for a standard generic second-order system to achieve this (assuming a unit step input). What are the associated system poles? Plot the unit step response for the result and demonstrate how well the design criteria are met (normalize your output to ensure the final value is 1.0). Display the resulting rise time, peak time, settling time, and percent overshoot on your graph. ii) Augment these desired 2nd-order system poles for future controller design in CE1: since this is a 6th-order system you will need four additional poles (real, negative, about ten times higher). For the first additional pole choose exactly ten times the real part of the dominant 2nd-order system poles. For the remaining three, successively subtract one from the first additional pole (to avoid repeated poles). iii) Also determine the optimal ITAE 6th-order coefficients and poles, using a natural frequency twice that from the dominant 2nd-order approach. Plot both the 6th-order ITAE and the augmented 6th-order desired responses with the dominant 2nd-order response of i (normalize to ensure steady-state values of 1.0); compare and discuss.

CE1.6b
For the desired controller poles designed in CE1.6a, design full-state feedback controllers (i.e. calculate **K**) for all three cases from CE1.3. For cases i and ii, use the augmented 6th-order poles based on dominant 2nd-order behavior; for case iii, use the 6th-order ITAE poles. In each case, evaluate your results: compare the simulated open- vs. closed-loop output responses for the same input cases as in CE1.3; use output attenuation correction so that the closed-loop steady-state values match the original open-loop steady-state values for easy comparisons.

CE2.6a
Since this is a 4th-order system we will need four desired poles for future controller design in CE2. Use a 4th-order ITAE approach with natural frequency $\omega_n = 3$ rad/s to generate the four desired poles. Plot the desired system response (normalize your output to ensure the final value is 1.0).

CE2.6b
For the desired controller poles designed in CE2.6a, design full-state feedback controllers (i.e. calculate **K**) for all three cases from CE2.3. In each case, evaluate your results: compare the simulated open- vs.

closed-loop output responses for the same input cases as in CE2.3. Be sure to adjust the *y* scales so that the closed-loop responses are clearly visible.

CE3.6a
Use a dominant 1st-order system with time constant τ=0.5 *sec*. What is the associated desired pole? Augment this desired 1st-order pole for future controller design in CE3: since this is a 3rd-order system you will need two additional poles (real, negative, about ten times higher). For the first additional pole choose exactly ten times the dominant 1st-order pole. For the remaining pole, subtract one from the first additional pole (to avoid repeated poles). Plot this augmented 3rd-order desired response vs. the dominant 1st-order response (normalize to ensure steady-state values of 1.0); compare and discuss.

CE3.6b
For the desired controller poles designed in CE3.6a, design full-state feedback controllers (i.e. calculate **K**) for both cases from CE3.3. In each case, evaluate your results: compare the simulated open- vs. closed-loop output responses for the same input cases as in CE3.3; (for Case ii, use output attenuation correction so that the closed-loop steady-state values match the original open-loop steady-state values for easy comparison).

7. Design of Linear Observers for State Feedback

For the control law $\mathbf{r} = \mathbf{U} - \mathbf{KX}$, controller design straightforward and same for any controllable system. Special *CCF* decoupled solution or Ackerman's formula for SISO, Matlab *place* for MIMO.

Problem: Often in physical systems we cannot measure all of the components in the state vector for feedback. Sometimes state vector components are not even physical quantities, but linear combinations of such. So we can design a linear observer to estimate the states for the full-state feedback control of Chapter 8.

7.1 Observers

Observer - estimate full state vector for use in feedback controller based on measured outputs. Integrate with current state-feedback controller system.

7.1.1 Observer Diagrams

Diagram (high-level):

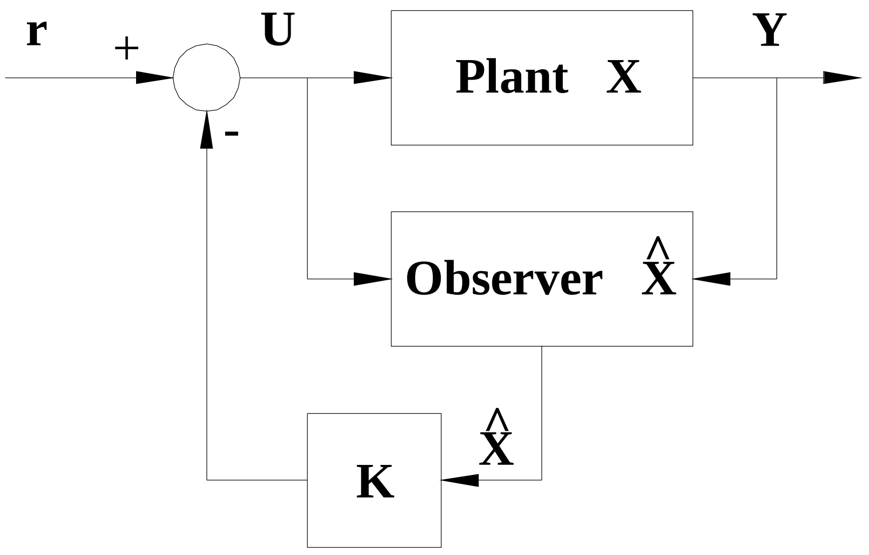

Figure 9.1 High-Level Observer Diagram

$\mathbf{X}$	true current state vector
$\hat{\mathbf{X}}$	estimate for current state vector, from Observer

Want to drive estimation error to zero $\quad \mathbf{e} = \mathbf{X} - \hat{\mathbf{X}} \to \mathbf{0}$

We require this observer error to converge faster than the transient response dynamics of the closed-loop linear system with controller; therefore, choose observer poles about ten times higher than controller poles.

Original open-loop system:

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$
$$\mathbf{Y} = \mathbf{CX} + \mathbf{DU}$$

Form for Observer:

L	observer gain matrix	dimensions (*n x m*)

Choose same form as open-loop plant dynamics. (Use state vector estimate in $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$, add zero (with some error), $\mathbf{L}\left(\mathbf{Y} - \hat{\mathbf{Y}}\right)$.)

$$\dot{\hat{\mathbf{X}}} = \mathbf{A}\hat{\mathbf{X}} + \mathbf{BU} + \mathbf{L}\left(\mathbf{Y} - \hat{\mathbf{Y}}\right)$$
$$\hat{\mathbf{Y}} = \mathbf{C}\hat{\mathbf{X}}$$

(*n x* 1) = (*n x n*) (*n x* 1) + (*n x r*)(*r x* 1) + (*n x m*)(*m x* 1)

Assume **D**=0

Diagram (details):

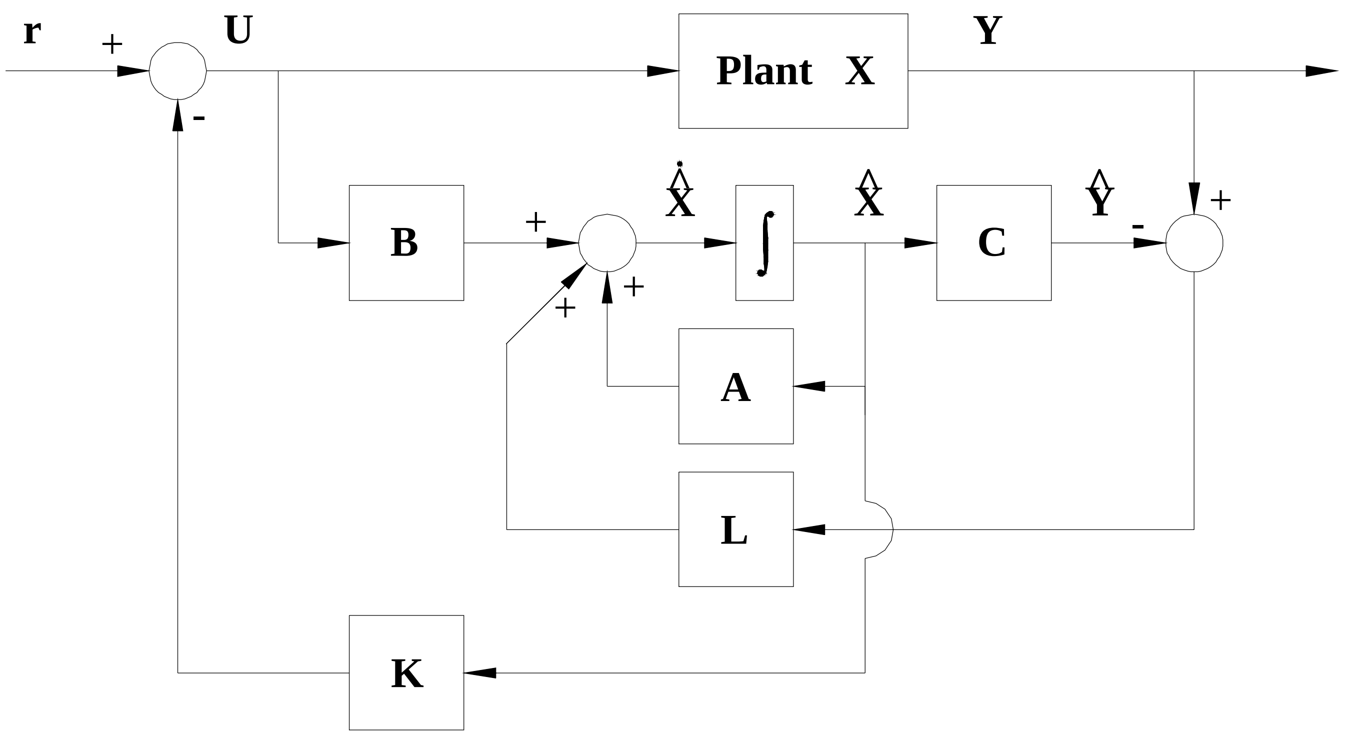

Figure 9.2 State-Space Closed-Loop System with Observer Block Diagram

Derivation of error convergence dynamics equation:

$$\dot{\mathbf{e}} = \dot{\mathbf{X}} - \dot{\hat{\mathbf{X}}} = \mathbf{AX} + \mathbf{BU} - \mathbf{A}\hat{\mathbf{X}} - \mathbf{BU} - \mathbf{L}\left(\mathbf{Y} - \hat{\mathbf{Y}}\right) = \mathbf{AX} - \mathbf{A}\hat{\mathbf{X}} - \mathbf{L}\left(\mathbf{CX} - \mathbf{C}\hat{\mathbf{X}}\right)$$

$$\dot{\mathbf{e}} = \left(\mathbf{A} - \mathbf{LC}\right)\mathbf{X} - \left(\mathbf{A} - \mathbf{LC}\right)\hat{\mathbf{X}}$$

$$\dot{\mathbf{e}} = \left(\mathbf{A} - \mathbf{LC}\right)\mathbf{e}$$

$$\dot{\mathbf{e}} = \hat{\mathbf{A}}\mathbf{e}$$

If all $\mathrm{Re}\left(eig\left(\hat{\mathbf{A}}\right)\right) < 0$ then this is an asymptotically stable error equation and:

$$\mathbf{e} \to \mathbf{0} \qquad (\hat{\mathbf{X}} \to \mathbf{X}) \qquad \text{as} \qquad t \to \infty$$

7.1.2 Observer Design Concept

If the original system is not observable, you cannot proceed in this chapter; instead you must determine why the system is not observable by looking at the physical problem and then re-design your system or re-derive your system model until it is fully state observable.

If the original open-loop system represented by **A**, **C** is *completely observable*, then we can arbitrarily place eigenvalues of $\hat{\mathbf{A}} = \mathbf{A} - \mathbf{LC}$ by selection of observer gain matrix **L**. We can control the rate of convergence $\hat{\mathbf{X}} \to \mathbf{X}$.

Now we derive state-space equations for the overall system with linear full-state feedback controller and full-state observer. We will combine state (actual and estimated) and observer error dynamics. The modified feedback control law (use observer estimate for feedback states) is:

$$\mathbf{U} = \mathbf{r} - \mathbf{K}\hat{\mathbf{X}}$$

The original system, with **D**=0 and adding the observer error convergence dynamics is described as follows:

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$$
$$\mathbf{Y} = \mathbf{CX}$$
$$\dot{\mathbf{e}} = \hat{\mathbf{A}}\mathbf{e}$$

We define a combined state and error vector, of dimension 2*n x* 1:

$$\mathbf{Z} = \begin{Bmatrix} \mathbf{X} \\ \mathbf{e} \end{Bmatrix}$$

(Not *Z* from Canonical!)

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{B}\left(\mathbf{r} - \mathbf{K}\hat{\mathbf{X}}\right) = \mathbf{AX} + \mathbf{B}\left(\mathbf{r} - \mathbf{K}\hat{\mathbf{X}} + \mathbf{KX} - \mathbf{KX}\right)$$

$$\dot{\mathbf{X}} = \left(\mathbf{A} - \mathbf{BK}\right)\mathbf{X} + \mathbf{BK}\left(\mathbf{X} - \hat{\mathbf{X}}\right) + \mathbf{Br} = \left(\mathbf{A} - \mathbf{BK}\right)\mathbf{X} + \mathbf{BKe} + \mathbf{Br}$$

$$\begin{Bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{e}} \end{Bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK} \\ \mathbf{0} & \mathbf{A} - \mathbf{LC} \end{bmatrix} \begin{Bmatrix} \mathbf{X} \\ \mathbf{e} \end{Bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{r} \qquad \mathbf{Y} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{X} \\ \mathbf{e} \end{Bmatrix}$$

$$\mathbf{A_r} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK} \\ \mathbf{0} & \mathbf{A} - \mathbf{LC} \end{bmatrix} \qquad \mathbf{B_r} = \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \qquad \mathbf{C_r} = \begin{bmatrix} \mathbf{C} & \mathbf{0} \end{bmatrix} \qquad \mathbf{D_r} = \mathbf{0}$$

$$\dot{\mathbf{Z}} = \mathbf{A_r Z} + \mathbf{B_r r} \qquad (2n \; x \; 1) = (2n \; x \; 2n)\,(2n \; x \; 1) + (2n \; x \; r)(r \; x \; 1)$$

$$\mathbf{Y} = \mathbf{C_r Z} \qquad (m \; x \; 1) = (m \; x \; 2n)\,(2n \; x \; 1)$$

So, to simulate closed-loop system dynamics with controller and observer, just use our existing MATLAB methods, with $\mathbf{A_r}, \mathbf{B_r}, \mathbf{C_r}, \mathbf{Z}, \mathbf{r}$ in place of $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{X}, \mathbf{U}$.

7.2 Duality and Observer Pole Placement

7.2.1 General Observer Design

First, the closed-loop controller design of Chapter 8 must be complete, i.e. we have found **K** to place the closed-loop system poles to the desired controller poles.

In observer design we must place the eigenvalues of $\mathbf{\hat{A} = A - LC}$ to achieve observer estimation convergence faster than the closed-loop controller transient response. We can take advantage of the controller design mathematics since the forms are similar; the controller and observer design problems are dual to each other: Compare to the controller problem where we placed eigenvalues of *A-BK* to achieve stability and desired transient response design specifications.

$$\mathbf{A_c = A - BK}$$
$$\mathbf{\hat{A} = A - LC}$$

The order is reversed:

$$\mathbf{BK}$$
$$\mathbf{LC}$$

A matrix and its transpose have the same characteristic equation and the same eigenvalues. Therefore, let us select **L** to change the eigenvalues of $\mathbf{\hat{A}^T}$:

$$\mathbf{\left[A - LC\right]^T = A^T - C^T L^T}$$

and then the eigenvalues of $\mathbf{\hat{A} = A - LC}$ will be the same. Comparing again:

$$\mathbf{A_c = A - BK}$$
$$\mathbf{\hat{A}^T = A^T - C^T L^T}$$

So the algorithms for controller design (SISO *CCF* decoupled solution, SISO Ackerman, MIMO *place*) apply directly to observer design if we make the following substitutions:

$$\mathbf{A \rightarrow A^T} \qquad \mathbf{B \rightarrow C^T} \qquad \mathbf{K \rightarrow L^T}$$

The original open-loop system described by **A**, **C** must be *completely observable*. If we replace the controllability matrix **P** with the above substitutions for **A** and **B**, we obtain the transpose of the observability matrix **Q**:

$$\mathbf{P = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}}$$

$$\mathbf{Q^T} = \begin{bmatrix} \mathbf{C^T} & \mathbf{A^T C^T} & \cdots & \left(\mathbf{A^{n-1}}\right)^{\mathbf{T}} \mathbf{C^T} \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA^{n-1}} \end{bmatrix}$$

Full-state observer design is ***DUAL*** to the pole placement problem for design of full-state-feedback controllers. Partial state observers are also possible (more efficient, just estimate those states that don't directly have a sensor, i.e. for all states except those that are also outputs).

<u>Observer Pole Placement via *L* (observer gain matrix): Example - same system as controller example</u>

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \; D = 0 \qquad L = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix}$$

$$C^T = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad L^T = \begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix}$$

In controller design algorithm, replace $A - BK \rightarrow A^T - C^T L^T$

$$A^T - C^T L^T = \begin{bmatrix} 0 & 0 & -18 \\ 1 & 0 & -15 \\ 0 & 1 & -2 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -18 \\ 1 & 0 & -15 \\ 0 & 1 & -2 \end{bmatrix} - \begin{bmatrix} L_1 & L_2 & L_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$A^T - C^T L^T = \begin{bmatrix} -L_1 & -L_2 & -18 - L_3 \\ 1 & 0 & -15 \\ 0 & 1 & -2 \end{bmatrix}$$

$$\left| sI - \left(A^T - C^T L^T\right) \right| = \begin{vmatrix} s + L_1 & L_2 & 18 + L_3 \\ -1 & s & 15 \\ 0 & -1 & s + 2 \end{vmatrix} = 0$$

$$(s + L_1)\left(s^2 + 2s + 15\right) - (-1)\left(L_2 s + 2L_2 + 18 + L_3\right) = 0$$

$$s^3+(L_1+2)s^2+(2L_1+L_2+15)s+(15L_1+2L_2+L_3+18)=0$$

Choose observer poles ten times higher than controller poles.

Controller poles	$s_{1,2,3}=-1.33\pm1.49i,-13.3$
Observer poles	$s_{1,2,3}=-13.3\pm14.9i,-133.3$

So the desired observer characteristic equation is:

$$\begin{gathered}s^3+160s^2+3955s+53260=0\\(\times1)(\times10)(\times100)(\times1000)\end{gathered}$$

Coefficients multiples of *K* case

Match like powers of *s* between the function of *L* and desired characteristic equation.

$$\begin{bmatrix}1&0&0\\2&1&0\\15&2&1\end{bmatrix}\begin{Bmatrix}L_1\\L_2\\L_3\end{Bmatrix}=\begin{Bmatrix}160-2\\3955-15\\53260-18\end{Bmatrix}$$

Coefficient matrix looks like a sort of skew-transpose of P^{-1} from controller design (*DUAL*):

$$P^{-1}=\begin{bmatrix}15&2&1\\2&1&0\\1&0&0\end{bmatrix}$$

Actually we don't need a matrix because solution is found in order 1, 2, 3:

$$L=\begin{bmatrix}L_1\\L_2\\L_3\end{bmatrix}=\begin{bmatrix}158\\3624\\43624\end{bmatrix}$$

Units? $\quad \dot{\hat{X}}=A\hat{X}+BU+L(Y-\hat{Y})$ $\quad \dot{\hat{X}}\begin{bmatrix}\frac{m}{s}\\\frac{m}{s^2}\\\frac{m}{s^3}\end{bmatrix}\equiv L(m)$ so $L\equiv\begin{bmatrix}\frac{1}{s}\\\frac{1}{s^2}\\\frac{1}{s^3}\end{bmatrix}$

Check to ensure that the desired observer poles were placed successfully into $\hat{A}=A-LC$. With observer poles ten times higher than controller poles, we can experience numerical issues (the magnitudes of the *L* terms above increase by an order of magnitude for each component).

Controller & Observer Example Summary

Original given open-loop linear system:

$$A=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18 & -15 & -2\end{bmatrix} \qquad B=\begin{bmatrix}0\\0\\1\end{bmatrix} \qquad C=\begin{bmatrix}1 & 0 & 0\end{bmatrix} \qquad D=0$$

State-Feedback Controller:

$$A_c=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -53.26 & -39.55 & -16\end{bmatrix} \qquad B_c=B \qquad C_c=C \qquad D_c=D=0 \quad K=\begin{bmatrix}35.26 & 24.55 & 14.00\end{bmatrix}$$

Full-State Observer with Controller:

$$A_r=\begin{bmatrix}A-BK & BK\\ 0 & A-LC\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ -53.26 & -39.55 & -16 & 35.26 & 24.55 & 14\\ 0 & 0 & 0 & -158 & 1 & 0\\ 0 & 0 & 0 & -3624 & 0 & 1\\ 0 & 0 & 0 & -43624 & -15 & -2\end{bmatrix} \qquad L=\begin{bmatrix}L_1\\L_2\\L_3\end{bmatrix}=\begin{bmatrix}158\\3624\\43624\end{bmatrix}$$

$$B_r=\begin{bmatrix}B\\0\end{bmatrix}=\begin{bmatrix}0\\0\\1\\0\\0\\0\end{bmatrix} \qquad C_r=\begin{bmatrix}C & 0\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\end{bmatrix} \qquad D_r=D=0$$

$$Z=\begin{Bmatrix}X\\e\end{Bmatrix}$$

Simulation output for Controller & Observer (assuming an initial observer error of 0.0005 on the first state and zero on the other two) is shown in Figure 8.?, for all three states, not just the single output.

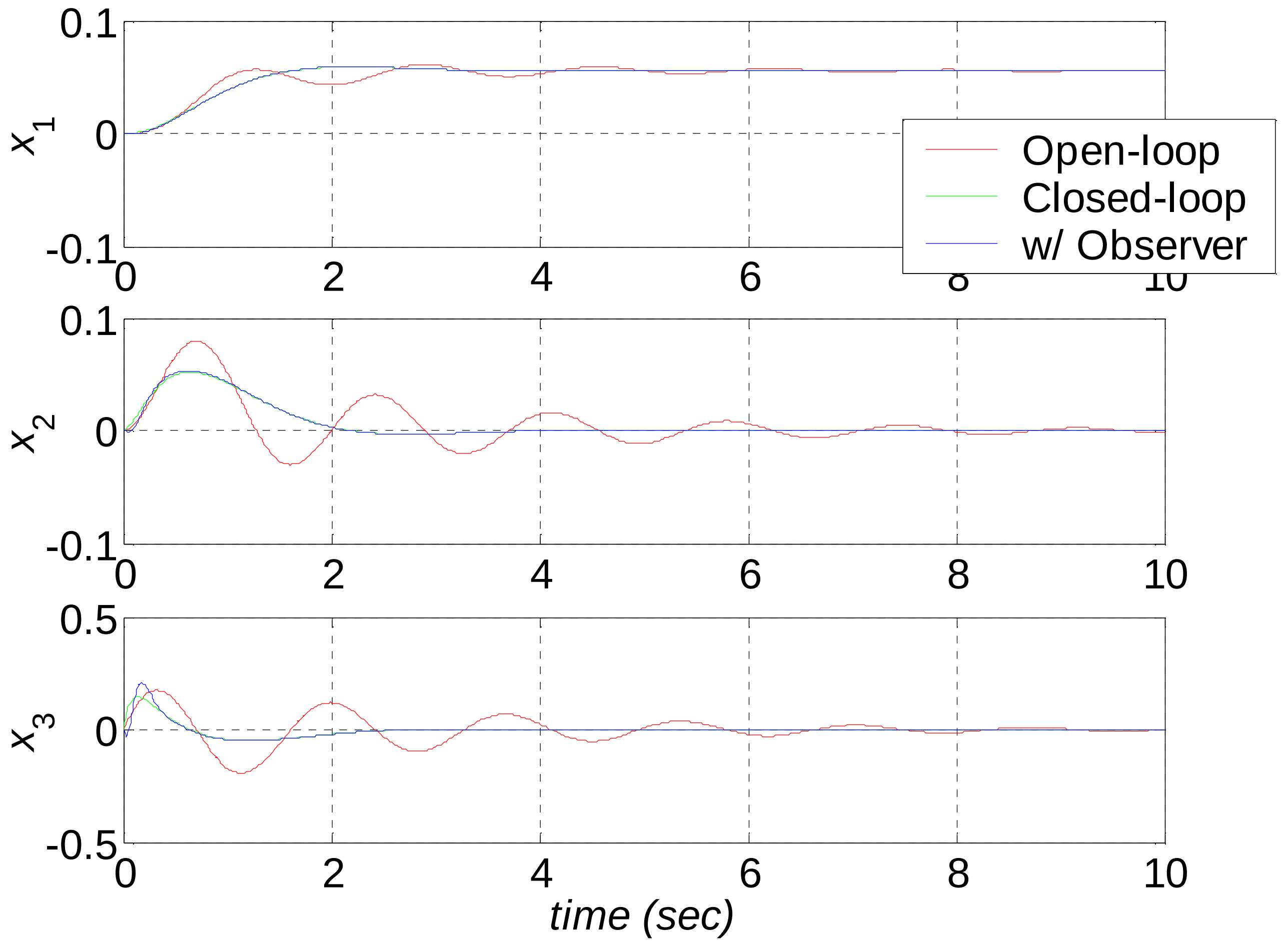

Figure 9.? Open-, Closed-, and Closed-Loop with Observer Loop State Responses for Example

7.2.2 Observer Design for SISO Systems via Ackerman's Formula

To calculate **L** for any general SISO system (not just *OCF*), we can adapt Ackerman's formula from controller design, Section 8.2.3. Ackerman's formula for observer design is:

$$\mathbf{L^T} = [\mathbf{0} \quad \mathbf{0} \quad \cdots \quad \mathbf{1}]\left[\mathbf{Q^T}\right]^{\mathbf{-1}} \Delta_{OBS}\left(\mathbf{A^T}\right)$$

where:

$\mathbf{L^T}$	transpose of constant observer gain matrix
$\mathbf{Q^T} = \left[\mathbf{C^T} \quad \mathbf{A^T C^T} \quad \cdots \quad \left(\mathbf{A^{n-1}}\right)^{\mathbf{T}} \mathbf{C^T}\right]$	transpose of the observability matrix

And $\Delta_{OBS}\left(\mathbf{A^T}\right)$ is the specified, desired observer characteristic polynomial, evaluated with the transpose of the system dynamics matrix **A**, rather than *s*.

Example

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix} \qquad C = [1 \quad 0 \quad 0]$$

$$Q = \begin{bmatrix} C \\ CA \\ CA^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\Delta_{OBS}\left(A^T\right) = A^3 + 160A^2 + 3955A + 53260I = \begin{bmatrix} 53242 & -2844 & -65232 \\ 3940 & 50872 & -57204 \\ 158 & 3624 & 43624 \end{bmatrix}$$

$$L^T = [0 \quad 0 \quad 1]\left(Q^T\right)^{-1} \Delta_{OBS}\left(A^T\right) = [158 \quad 3624 \quad 43624]$$

Agrees with the previous solution; for SISO systems the solution for *L* is unique.

7.2.3 Observer Error

$$\dot{\mathbf{e}} = \hat{\mathbf{A}}\mathbf{e} = [\mathbf{A} - \mathbf{LC}]\mathbf{e}$$

Laplace transform:

$$s\mathbf{E}(s) = [\mathbf{A} - \mathbf{LC}]\mathbf{E}(s)$$
$$(s\mathbf{I} - [\mathbf{A} - \mathbf{LC}])\mathbf{E}(s) = 0$$

If $\left|s\mathbf{I} - [\mathbf{A} - \mathbf{LC}]\right| = 0$ (as it must be for Observer L design), infinite solutions $\mathbf{E}(s)$

If $\left|s\mathbf{I} - [\mathbf{A} - \mathbf{LC}]\right| \neq 0$ unique solution $\mathbf{E}(s)$

7.3 Reduced-Order Observers

7.4 State Estimation and Output Feedback

7.5 Detectability

7.6 Matlab for Observer Design

The following Matlab functions are useful for design of linear observers for full-state feedback:

place(*A'*,*C'*,*ObsPoles*)	Solve for full-state observer gain matrix **L** to place the desired observer poles *ObsPoles* into closed-loop dynamics matrix $\hat{\mathbf{A}} = \mathbf{A} - \mathbf{LC}$.
acker(*A'*,*C'*,*ObsPoles*)	Solve for full-state observer gain matrix **L** to place the desired observer poles *ObsPoles* into closed-loop dynamics matrix $\hat{\mathbf{A}} = \mathbf{A} - \mathbf{LC}$, using Ackerman's formula, for SISO systems only.

Continuing Matlab Example: Design of Linear Observers for State Feedback

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), design the full-state observer, i.e. determine observer gain matrix **L** given **A**, **C**, and reasonable observer poles to go with the controller poles of the Chapter 8 continuing Matlab example. The following Matlab code performs this observer design for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 9.  Design and Simulation of Linear Observers for State Feedback
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

PolesObs = 10*Poles2;          % Select desired observer poles; ten times higher than controller poles

L     = place(A',C',PolesObs)';  % Compute full-state observer controller gain matrix L
Lack = acker(A',C',PolesObs)';   % For SISO we can check L via Ackerman's formula

Ahat = A-L*C;                   % Compute the closed-loop observer estimation error dynamics matrix
eig(Ahat);                      % Check to ensure desired poles are in there

%       Compute and simulate closed-loop system with controller and observer
Xr0 = [0.4;0.2;0.10;0];         % Define vector of initial conditions [x1;x2;e1;e2]
Ar = [(A-B*K) B*K;zeros(size(A)) (A-L*C)];
Br = [B;zeros(size(B))];
Cr = [C zeros(size(C))];
Dr = D;
[Yr,Xr] = lsim(Ar,Br,Cr,Dr,U,t,Xr0);
figure;                         %  Compare Open-, Closed-loop, and Controller/Observer output responses
plot(t,Yo,'r',t,Yc,'g',t,Yr,'b'); grid; axis([0 4 -0.2 0.5]);
set(gca,'FontSize',18);
legend('Open-loop','Closed-loop','w/ Observer');
xlabel('\ittime (sec)'); ylabel('\ity');

figure;                         %  Plot observer errors
plot(t,Xr(:,3),'r',t,Xr(:,4),'g'); grid; axis([0 0.2 -3.5 0.2]);
set(gca,'FontSize',18);
legend('Obs error 1','Obs error 2');
xlabel('\ittime (sec)'); ylabel('\ite');
```

This m-file, combined with the previous chapter m-files, yielded the following output (*place* and *acker* yielded identical results for **L**), plus the output response plot of Figure 9.3 and the observer error plot of Figure 9.4:

```
PolesObs =
-57.1429 +51.1954i
 -57.1429 -51.1954i

L =
110.29
5405.13

Ahat =
   -110.3      1.0
   -5.445.1   -4.0

ans =
-57.1429 +51.1954i
 -57.1429 -51.1954i
```

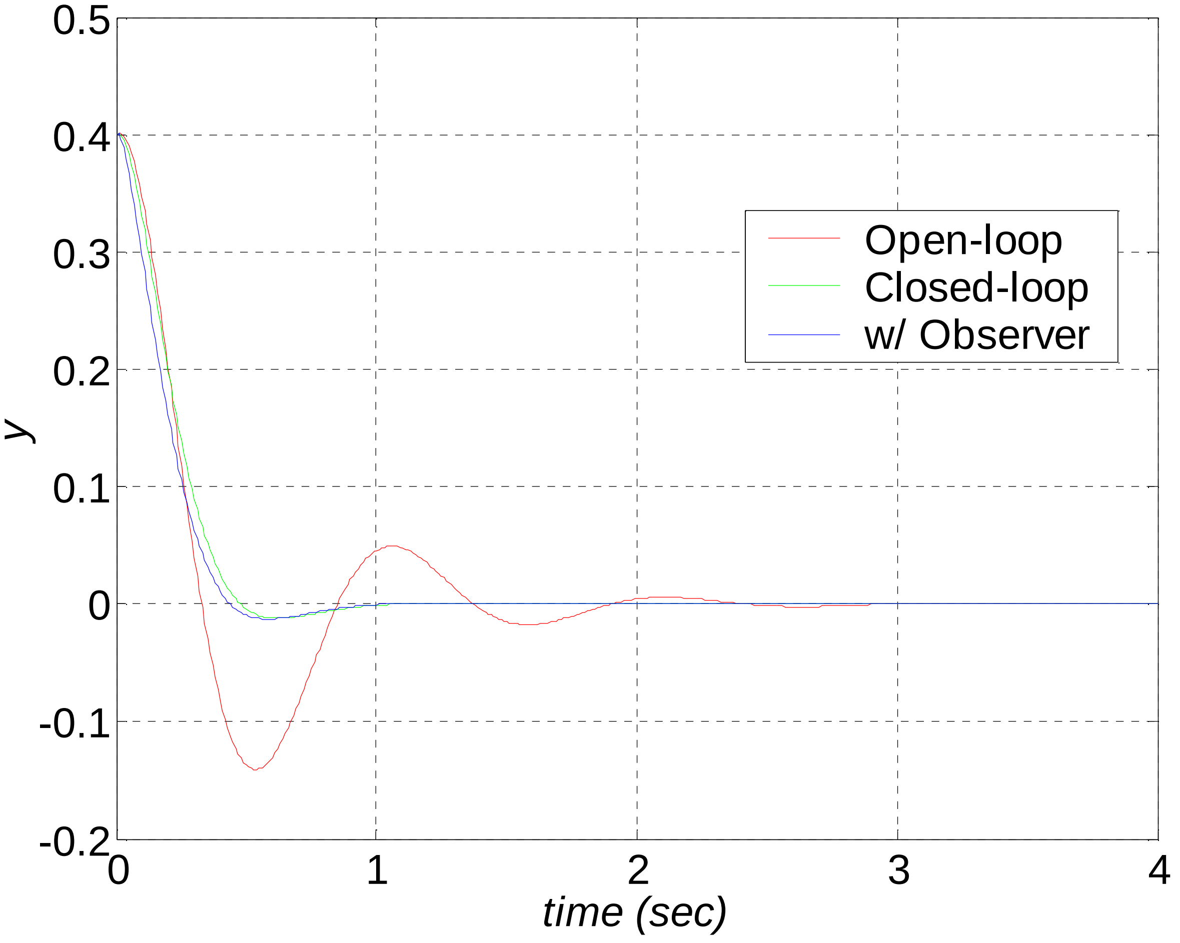

Figure 9.3 Open-, Closed-, and Closed-Loop with Observer Output Responses, Matlab Example

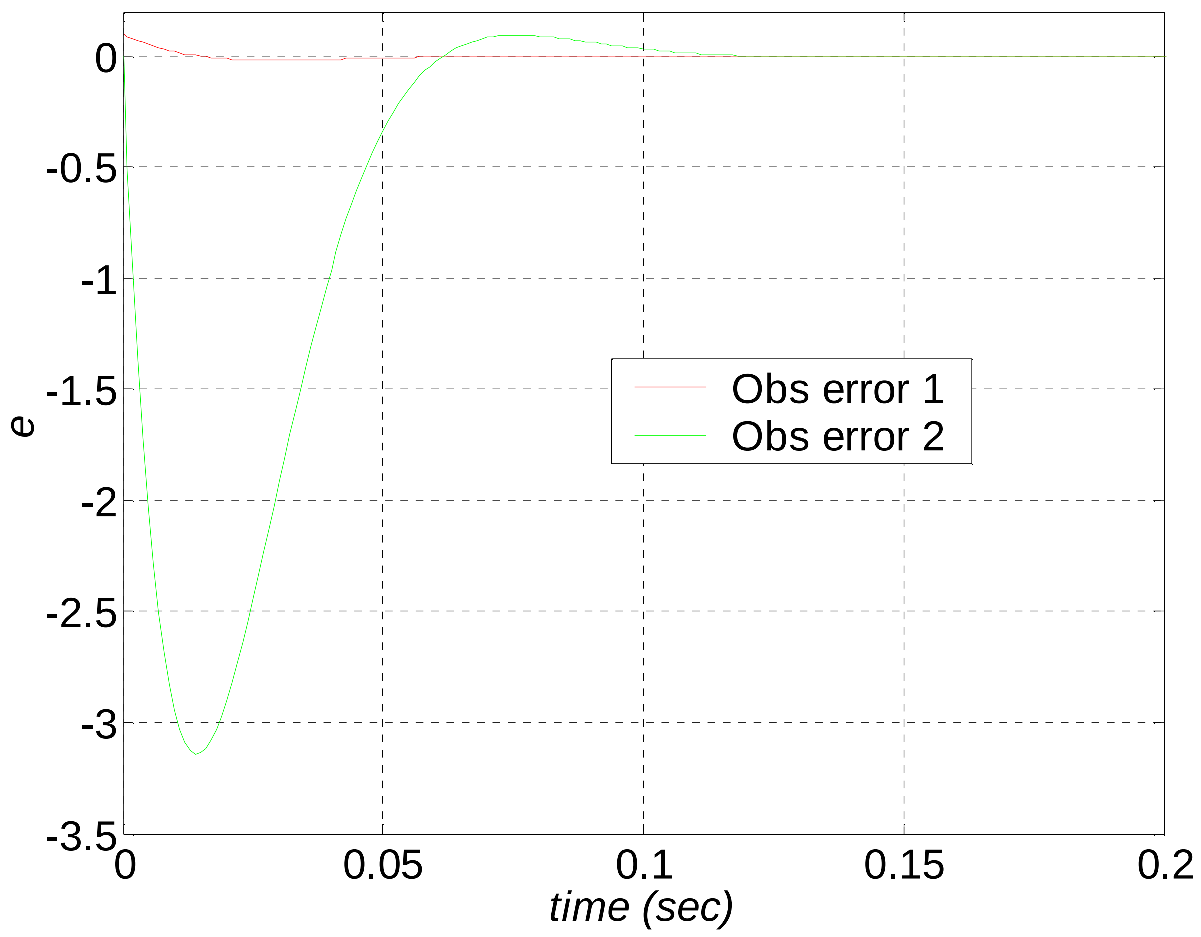

Figure 9.4 Observer Error State Responses, Matlab Example

In simulation we started the observer with an artificial error of 0.1 *rad* in shaft angle θ estimation (and zero error in $\dot{\theta}$ estimation). In Figure 9.4 we see that the observer error for shaft angle starts from the assumed initial value and quickly (the time scale of Figure 9.4 is zoomed in on that of Figure 9.3) goes to zero. The observer velocity error goes to zero soon after, but with an initial large negative peak, even though its assumed initial error was zero. However, this effect is not seen in Figure 9.3, where the closed-loop system with observer (blue) slightly lags the closed-loop system response (green), but then quickly matches (around 1 *sec*). Since the observer poles were chosen to be ten times greater than the controller poles, the observer transient error estimation dynamics goes to zero much faster than the closed-loop system with controller transient dynamics. The red and green responses in Figure 9.3 are identical to those of Figure 8.2.

Closed-loop plus observer system simulation can also be performed using Matlab's Simulink. Figure 9.5 shows the high-level diagram and Figure 9.6 shows the detailed diagram for the observer implementation.

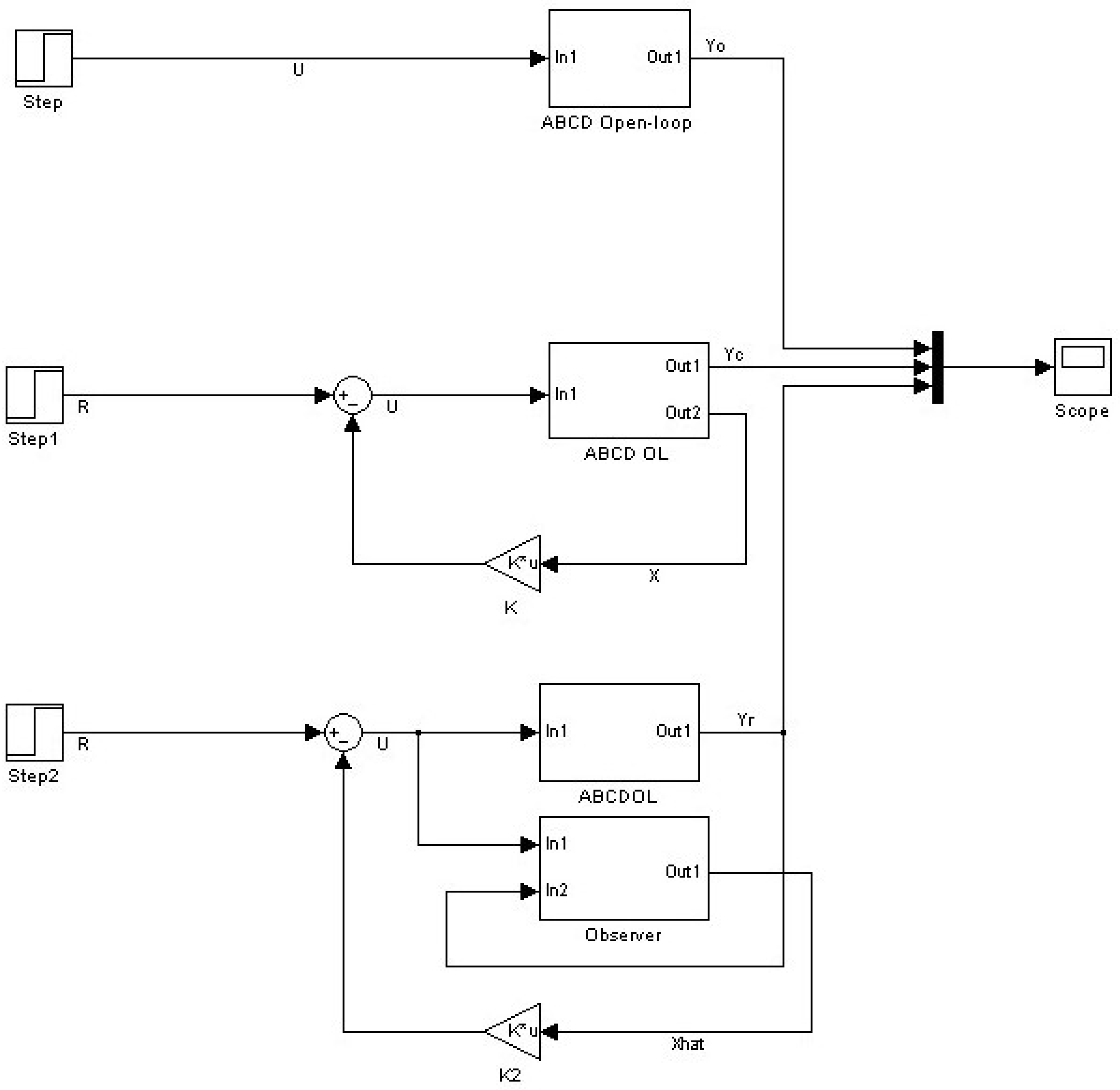

Figure 9.5 Simulink Diagram for Open-, Closed-, plus Closed-Loop with Observer Response

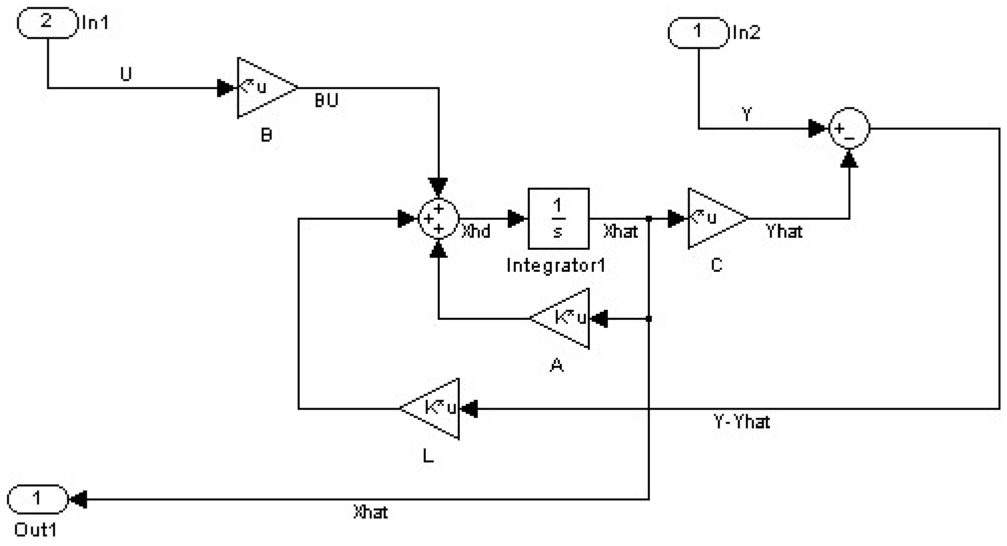

Figure 9.6 Detailed Simulink Diagram for Observer Implementation

7.7 Continuing Examples: Design of Linear State Feedback Observers

7.7.1 Continuing Example I: Two-mass MIMO Translational Mechanical System

For both Cases of Continuing Example I (Two-mass MIMO Translational Mechanical System), determine a full-state observer gain matrix **L** to estimate the states for feedback at every instant in control time.

<u>Solution, Case i</u>

The observer estimation error dynamics matrix we will place the desired observer poles into via **L** is $\hat{\mathbf{A}} = \mathbf{A} - \mathbf{LC}$. Since the observer estimation dynamics must take place faster than the closed-loop controller dynamics, we choose four desired observer poles to be 10 times greater than the four controller poles:

$$s_{1,2} = -20 \pm 21i \qquad\qquad s_{3,4} = -200, -210$$

The polynomial associated with those desired observer poles is:

$$s^4 + 450s^3 + 5.924\times10^4 s^2 + 2.0244\times10^6 s + 3.5276\times10^7$$

Note that this observer polynomial is very similar to the desired controller behavior characteristic polynomial of the Chapter 4 Example I, but the coefficients of the *s* powers are multiplied by:

$$\begin{bmatrix}1 & 10 & 100 & 1000 & 10000\end{bmatrix},$$

since the poles were uniformly multiplied by 10. Therefore, numerical problems may arise; hence one should not take observer poles any higher than the rule-of-thumb 10x greater.

Taking advantage of the duality between controller and observer design and using Matlab function *place*, we found the 4x2 full-state observer gain matrix **L** to be:

$$\mathbf{L} = \begin{bmatrix} 195 & 1073 \\ -978 & 213836 \\ 2 & 254 \\ -101 & 11863 \end{bmatrix}$$

Due to the numerical issue pointed out above, the terms of **L** vary greatly in magnitude. The output response plots for the combined controller/observer system are shown in Figure 9.7 for Case i.

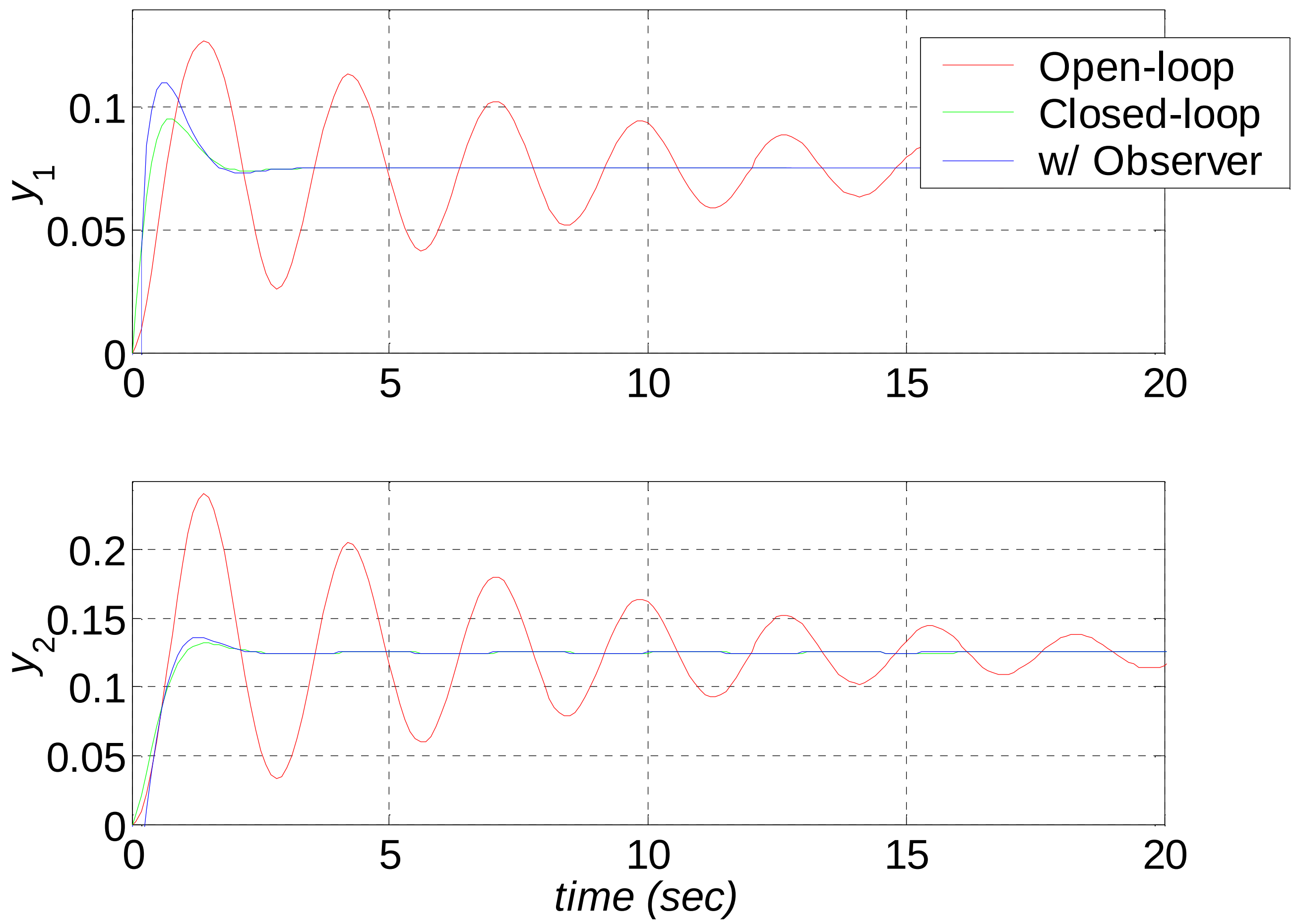

Figure 9.7 Open-, Closed-, and Closed-Loop with Observer Output Responses, Case i

We started the observer with an artificial error of 0.5 and 1 *mm* for y_1 and y_2, respectively (and zero error in both velocity estimations). In Figure 9.7 we see that the assumed initial errors cause both mass displacements to start at a negative initial value. The closed-loop system with observer (blue) overshoots the closed-loop system response (green), but then quickly matches (around 3 *sec*). Since the observer poles were chosen to be ten times greater than the controller poles, the observer transient error estimation dynamics goes to zero much faster than the closed-loop system with controller transient dynamics. The red and green responses in Figure 9.7 are identical to those of Figure 8.5.

Solution, Case ii

The desired observer poles are 10 times greater than the controller poles used in Chapter 8:

$$s_{1,2} = -12.3 \pm 36.6i \qquad\qquad s_{3,4} = -18.1 \pm 12.0i$$

Taking advantage of the duality between controller and observer design and using Matlab function *place*, we found the 4x1 full-state observer gain matrix **L** to be:

$$\mathbf{L} = \begin{bmatrix} 60 \\ 2756 \\ 5970 \\ 135495 \end{bmatrix}$$

Again, we see that the terms of **L** vary greatly in magnitude, due to the fact that the desired observer poles are 10 times greater than the controller poles. The output response plots for the combined controller/observer system are shown in Figure 9.8 for Case ii.

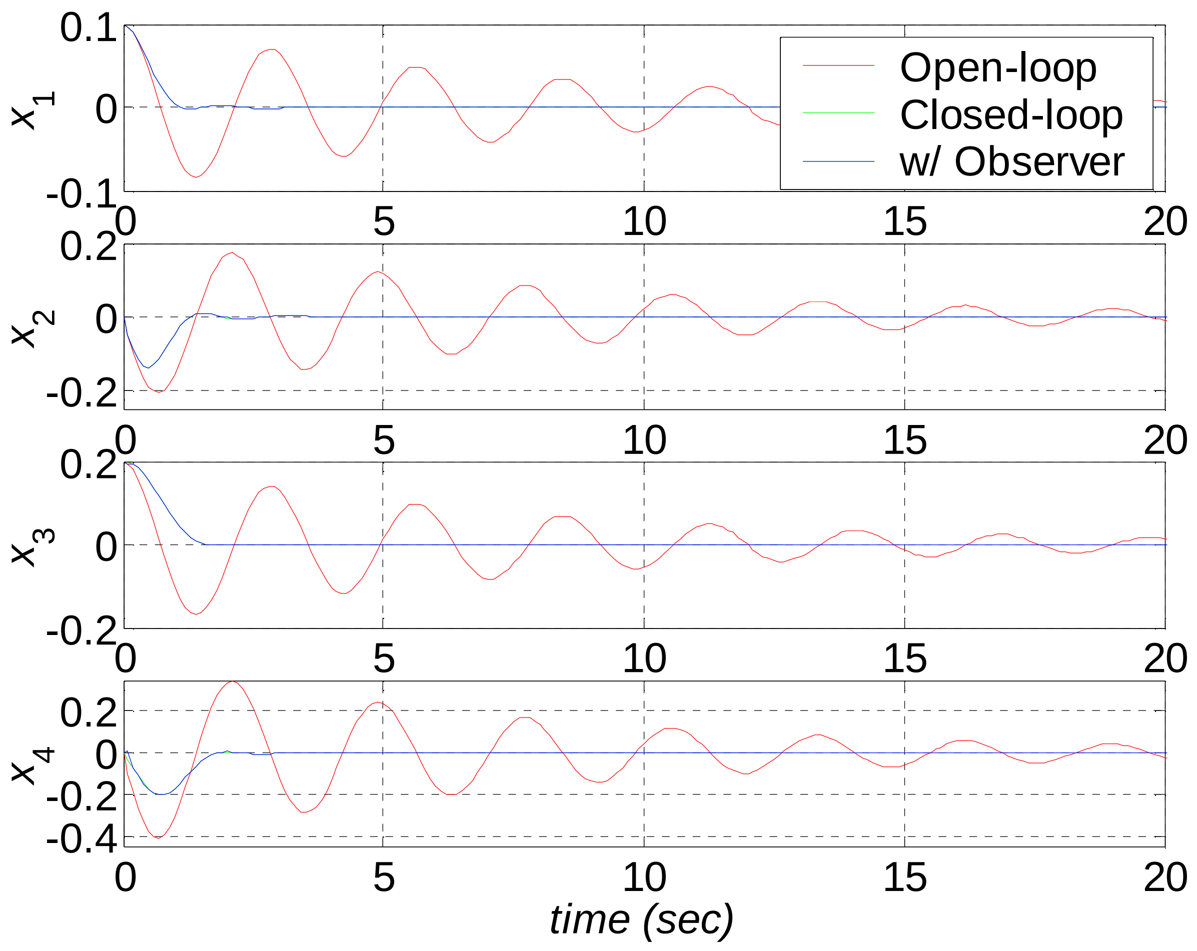

Figure 9.8 Open-, Closed-, and Closed-Loop with Observer Output Responses, Case ii

We again started the observer with an artificial error of 0.5 and 1 *mm* for y_1 and y_2, respectively (and zero error in both velocity estimations). In Figure 9.8 we see that in Case ii the closed-loop system with observer (blue) matches the closed-loop system response (green) very well at this scale. There are observer errors, but they go to zero before 1 *sec*. Since the observer poles were chosen to be ten times greater than the controller poles, the observer transient error estimation dynamics goes to zero much faster than the closed-loop system with controller transient dynamics. The red and green (almost perfectly masked by blue) responses in Figure 9.8 are identical to those of Figure 8.6.

7.7.2 Continuing Example II: Rotational Electromechanical System

For Continuing Example II (1-dof rotational electromechanical system; SISO: input v, output θ), determine a full-state observer gain matrix $\mathbf{L}$ to estimate the states for feedback at every instant in control time.

Solution

The observer estimation error dynamics matrix we will place the desired observer poles into via $\mathbf{L}$ is $\hat{\mathbf{A}} = \mathbf{A} - \mathbf{LC}$. Since the observer estimation dynamics must take place faster than the closed-loop controller dynamics, we choose three desired observer poles to be 10 times greater than the three controller poles:

$$s_{1,2,3} = -40, -120, -130$$

The polynomial associated with those desired observer poles is:

$$s^3 + 290s^2 + 25600s + 624000$$

Note that this observer polynomial is very similar to the desired controller behavior characteristic polynomial of the Chapter 4 Example II, but the coefficients of the s powers are multiplied by:

$$\begin{bmatrix} 1 & 10 & 100 & 1000 \end{bmatrix},$$

since the poles were uniformly multiplied by 10. Therefore, numerical problems may arise; hence one should not take observer poles any higher than the rule-of-thumb 10x greater.

Taking advantage of the duality between controller and observer design and doing the observer design by hand, or using Matlab functions *place* or *acker*, we find the 3x1 full-state observer gain matrix $\mathbf{L}$ to be:

$$\mathbf{L} = \begin{bmatrix} 287 \\ 24737 \\ 549215 \end{bmatrix}$$

The three state response plots for the combined controller/observer system are shown in Figure 9.9 for Example II.

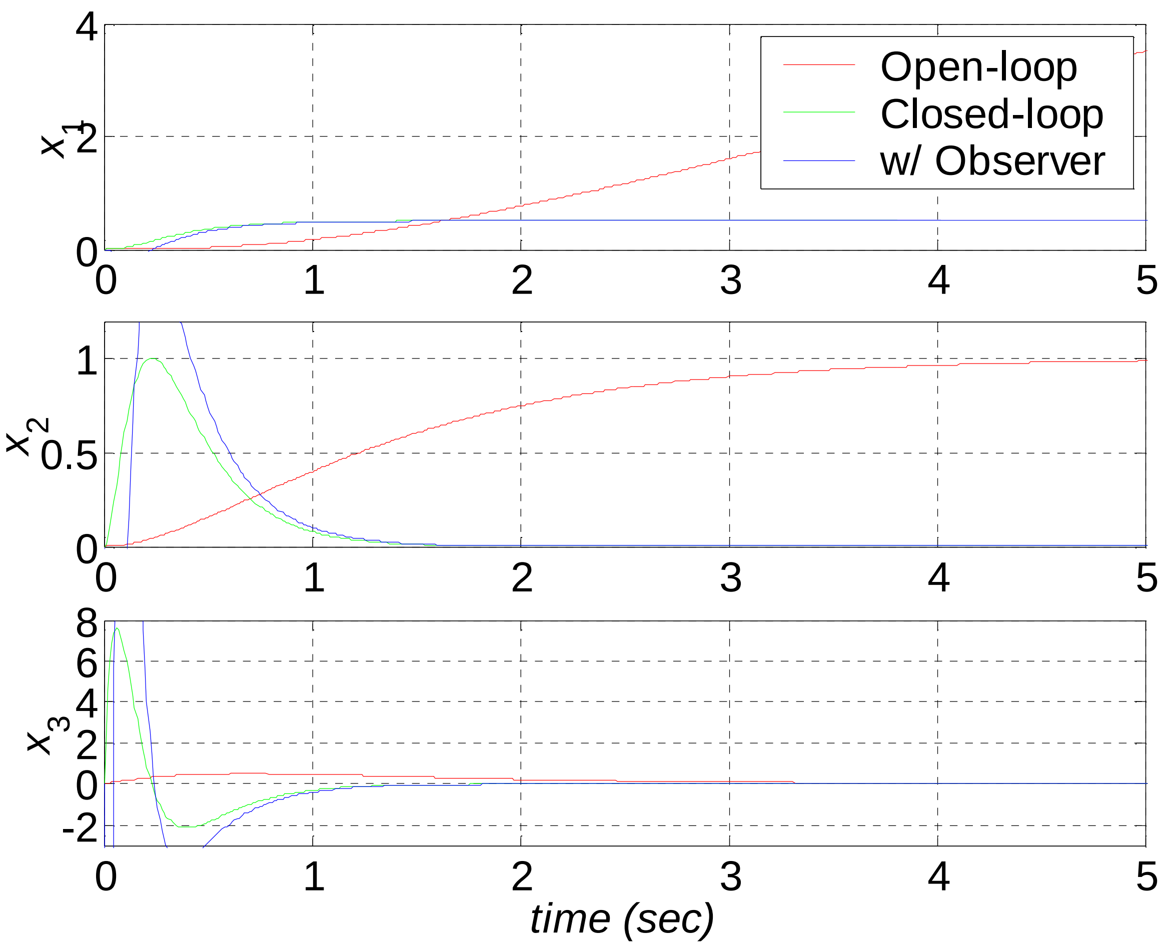

Figure 9.9 Open-, Closed-, and Closed-Loop with Observer Output Responses, Example II

We started the observer with an artificial error of 0.001 *rad* in shaft angle θ estimation (and zero error in $\dot{\theta}$ and $\ddot{\theta}$ estimations). In Figure 9.3, the closed-loop system with observer (blue) slightly lags the closed-loop system output θ response (green); also, there is significant overshooting in the $\dot{\theta}$ and $\ddot{\theta}$ responses for the controller/observer system. All three plots match at around 1.5 *sec*. Since the observer poles were chosen to be ten times greater than the controller poles, the observer transient error estimation dynamics goes to zero much faster than the closed-loop system with controller transient dynamics. The red and green responses in Figure 9.9 are identical to those of Figure 8.7.

7.8 Homework Assignments

7.8.1 Mathematical Homework Assignments

7.8.2 Matlab Homework Assignments

7.8.3 Continuing Homework Assignments

CE1.7
For the controllers designed in CE1.8, design full-state observers (i.e. calculate **L**), for all three cases. Use desired observer poles 10 times higher than the desired controller poles (for Case iii, two times greater works better due to numerical conditioning). In each case, evaluate your results: compare the simulated open-, closed-, and closed-loop with observer output responses for the same input cases as in CE1.3; use the same correction factors from CE1.8. Introduce some initial observer error, otherwise the closed- and closed-loop with observer responses will be identical in simulation.

CE2.7
For the controllers designed in CE2.8, design full-state observers (i.e. calculate **L**), for all three cases (this is possible only for the full-state-observable cases). Use desired observer poles 10 times higher than the desired controller poles. In each case, evaluate your results: compare the simulated open-, closed-, and closed-loop with observer output responses for the same input cases as in CE2.3; use the same correction factors from CE2.8. Introduce some initial observer error, otherwise the closed- and closed-loop with observer responses will be identical in simulation.

CE3.7
For the controllers designed in CE3.8, design full-state observers (i.e. calculate **L**), for both cases. Use desired observer poles 10 times higher than the desired controller poles. In each case, evaluate your results: compare the simulated open-, closed-, and closed-loop with observer output responses for the same input cases as in CE3.3 (for Case ii, use the same correction factor from CE3.8). Introduce some initial observer error, otherwise the closed- and closed-loop with observer responses will be identical in simulation.

8. Introduction to Optimal Control

Linear state-feedback controllers can be designed to stabilize a given system and provide desired transient response.

Can also optimize (usually minimize) objective functions of:

- time
- error
- energy
- combinations
- other

Performance Index (Objective Function):

$$J = \int_0^{t_f} g(x,u,t)dt \tag{1}$$

If $\mathbf{r}(t)=0$, the controller is a *regulator* (reject initial conditions and/or disturbances) - maintain equilibrium state $\mathbf{X}_e=0$. Any deviation $\mathbf{X}$ is the error.

$$\mathbf{U}=\mathbf{r}-\mathbf{KX}=-\mathbf{KX} \tag{2}$$

$$\begin{aligned}\dot{\mathbf{X}} &= (\mathbf{A}-\mathbf{BK})\mathbf{X}+\mathbf{B}r = \mathbf{A_c}\mathbf{X}+\mathbf{B_c}r \\ \dot{\mathbf{X}} &= \mathbf{A_c}\mathbf{X}\end{aligned} \tag{3}$$

8.1 Optimal Controller for Minimum Energy

Replace my XTX.

8.2 Linear Quadratic Regulator (LQR)

Now, in addition to minimizing control efforts *u*(*t*), optimal controllers can also minimize the input effort **U** required in the control. An optimal controller which simultaneously tries to minimize state error and input effort (these are competing factors) is the Linear Quadratic Regulator (LQR), whose objective function is (19):

$$J = \int_0^\infty \frac{1}{2}\left(\mathbf{X}^\mathbf{T}\mathbf{Q}\mathbf{X} + \mathbf{U}^\mathbf{T}\mathbf{R}\mathbf{U}\right)dt \tag{19}$$

where $\mathbf{Q} \in n \times n$ and $\mathbf{R} \in r \times r$ are weighting matrices (could be *I* if both state error and input effort goals are equally important).

We must minimize *J* subject to the system dynamics equations $\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU}$. We start by considering the augmented objective function (where we have added zero ($0 = \dot{\mathbf{X}} - \mathbf{AX} - \mathbf{BU}$) to the integral):

$$J = \int_0^\infty \left[\frac{1}{2}\left(\mathbf{X}^\mathbf{T}\mathbf{Q}\mathbf{X} + \mathbf{U}^\mathbf{T}\mathbf{R}\mathbf{U}\right) + \mathbf{\Lambda}^\mathbf{T}\left(\dot{\mathbf{X}} - \mathbf{AX} - \mathbf{BU}\right)\right]dt \tag{0.1}$$

$\mathbf{\Lambda}$ is a vector of Lagrange multipliers. The integrand of (1.1) is the Hamiltonian:

$$H = \frac{1}{2}\left(\mathbf{X}^\mathbf{T}\mathbf{Q}\mathbf{X} + \mathbf{U}^\mathbf{T}\mathbf{R}\mathbf{U}\right) + \mathbf{\Lambda}^\mathbf{T}\left(\dot{\mathbf{X}} - \mathbf{AX} - \mathbf{BU}\right) \tag{0.2}$$

Now we apply the Euler-Lagrange dynamic equations (REF??) to the variables **X**, **U**, and $\mathbf{\Lambda}$:

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial H}{\partial \dot{\mathbf{X}}}\right) - \frac{\partial H}{\partial \mathbf{X}} &= 0 \\
\frac{d}{dt}\left(\frac{\partial H}{\partial \dot{\mathbf{U}}}\right) - \frac{\partial H}{\partial \mathbf{U}} &= 0 \\
\frac{d}{dt}\left(\frac{\partial H}{\partial \dot{\mathbf{\Lambda}}}\right) - \frac{\partial H}{\partial \mathbf{\Lambda}} &= 0
\end{aligned} \tag{0.3}$$

Substituting (1.2) and simplifying, (1.3) yields three equations to solve:

$$\begin{aligned}
\dot{\mathbf{\Lambda}} &= -\mathbf{QX} - \mathbf{A}^\mathbf{T}\mathbf{\Lambda} \\
\mathbf{U} &= -\mathbf{R}^{-1}\mathbf{B}^\mathbf{T}\mathbf{\Lambda} \\
\dot{\mathbf{X}} &= \mathbf{AX} + \mathbf{BU}
\end{aligned} \tag{0.4}$$

We assume the solution form is:

$$\mathbf{\Lambda} = \mathbf{KX} \tag{0.5}$$

where **K** is the constant LQR feedback gain matrix (different form and role from the full-state-feedback gain matrix of Chapter 8). Substituting this assumed form of (1.5) into equations (1.4), the combined equations become:

$$\dot{\mathbf{K}}\mathbf{X} + \mathbf{K}\left(\mathbf{AX} - \mathbf{BR}^{-1}\mathbf{B}^{T}\mathbf{KX}\right) = -\mathbf{QX} - \mathbf{A}^{T}\mathbf{KX} \tag{0.6}$$

Since K is constant, $\dot{\mathbf{K}} = \mathbf{0}$ and (1.6) becomes:

$$\left(\mathbf{A}^{T}\mathbf{K} + \mathbf{KA} - \mathbf{KBR}^{-1}\mathbf{B}^{T}\mathbf{K} + \mathbf{Q}\right)\mathbf{X} = 0 \tag{0.7}$$

Equation (1.6) must hold for all state vectors X and so:

$$\mathbf{A}^{T}\mathbf{K} + \mathbf{KA} - \mathbf{KBR}^{-1}\mathbf{B}^{T}\mathbf{K} + \mathbf{Q} = \mathbf{0} \qquad \mathbf{K} \in n \times n \tag{20}$$

This is the Matrix-Ricatti Equation for, which must be solved for optimal LQR gain matrix **K**. This can be accomplished by using Matlab function *are* (algebraic Ricatti equation). Note that **K** is different than the full-state-feedback gain matrix **K** for controllers from Chapter 8. The LQR optimal control law from (1.4) is:

$$\mathbf{U} = -\mathbf{R}^{-1}\mathbf{B}^{T}\mathbf{KX} \tag{21}$$

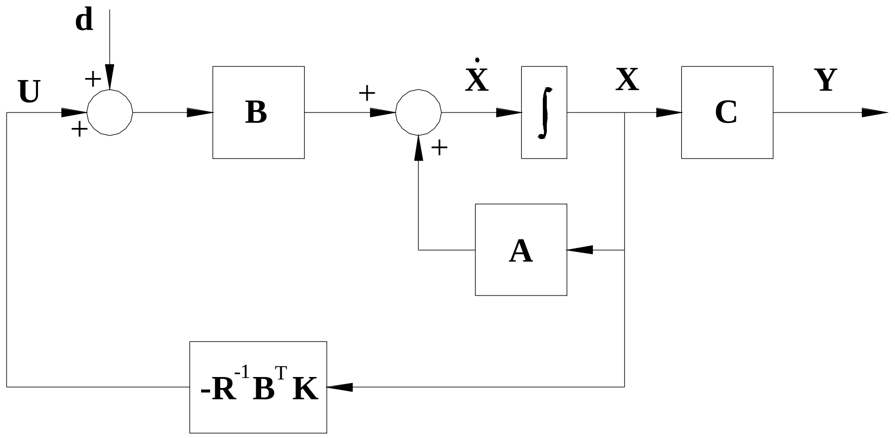

Figure 10.1 shows the closed-loop block diagram for the LQR system. The reference input **r** is zero (compare with Figure 8.1) and **d** is a disturbance which the regulator must reject. Due to the different control law, the LQR closed-loop system dynamics matrix, for control simulation, is no longer $\mathbf{A} - \mathbf{BK}$ as it was in Chapter 8. Instead, it is:

$$\dot{\mathbf{X}} = \mathbf{AX} + \mathbf{BU} = \left[\mathbf{A} - \mathbf{B}\left(\mathbf{R}^{-1}\mathbf{B}^{T}\mathbf{K}\right)\right]\mathbf{X} = \mathbf{A_c X}$$

$$\mathbf{A_c} = \mathbf{A} - \mathbf{BR}^{-1}\mathbf{B}^{T}\mathbf{K}$$

8.3 Matlab for Optimal Control

The following Matlab functions are used for design of LQR feedback controllers:

are(*A*,*BB*,*Q*)	Solve algebraic Ricatti equation to find optimal LQR gain matrix to find gain matrix $\mathbf{K_{LQR}}$, given weighting matrices **Q** (state error) and **R** (input effort), plus $\mathbf{BB}=\mathbf{BR}^{-1}\mathbf{B}^{\mathbf{T}}$. For use with LQR feedback control law $\mathbf{U}=-\mathbf{R}^{-1}\mathbf{B}^{\mathbf{T}}\mathbf{KX}$ and Section 10.2.
lqr(*A*,*B*,*Q*,*R*)	Calculates the optimal LQR gain matrix **K**, given weighting matrices **Q** (state error) and **R** (input effort). For use with standard Chapter 8 feedback control law $\mathbf{U}=-\mathbf{KX}$.

Continuing Matlab Example: Linear Quadratic Regulator Design

For the Continuing Matlab Example (SISO rotational mechanical system: input τ, output θ), design the optimal LQR regulator controller, i.e. determine gain matrix $\mathbf{K_{LQR}}$. The following Matlab code performs this LQR design for the continuing example.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Chapter 10.  Linear Quadratic Regulator Design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Q = 20*eye(2);                          %  Weighting matrix for state error
R = [1];                                %  Weighting matrix for input effort
BB = B*inv(R)*B';

KLQR = are(A,BB,Q);                     %  Solve algebraic Ricatti equation to find optimal LQR gain matrix
ALQR = A-B*inv(R)*B'*KLQR;                      % Compute the closed-loop state-feedback system
[YLQR,XLQR] = lsim(ALQR,Bc,Cc,Dc,U,t,X0);      % Compare open-loop and closed-loop step responses

figure;
subplot(211), plot(t,Xo(:,1),'r',t,Xc(:,1),'g',t,XLQR(:,1),'b'); grid; axis([0 4 -0.2 0.5]);
set(gca,'FontSize',18);
legend('Open-loop','Closed-loop','LQR');
ylabel('{\itx}_1')
subplot(212), plot(t,Xo(:,2),'r',t,Xc(:,2),'g',t,XLQR(:,2),'b'); grid; axis([0 4 -2 1]);
set(gca,'FontSize',18);
xlabel('\ittime (sec)'); ylabel('{\itx}_2');

% Calculate and plot to compare closed-loop and LQR input efforts required
Uc = -K*Xc';                            % Chapter 8 input effort
ULQR = -inv(R)*B'*KLQR*XLQR';           % LQR input effort

figure;
plot(t,Uc,'g',t,ULQR,'b'); grid; axis([0 4 -10 6]);
set(gca,'FontSize',18);
legend('Closed-loop','LQR');
xlabel('\ittime (sec)'); ylabel('\itU');
```

This m-file, combined with the previous chapter m-files, yielded the following output, plus the comparison of open-, closed-loop, vs. LQR state responses shown in Figure 10.2.

```
KLQR =
83.16       0.25
 0.25      2.04
```

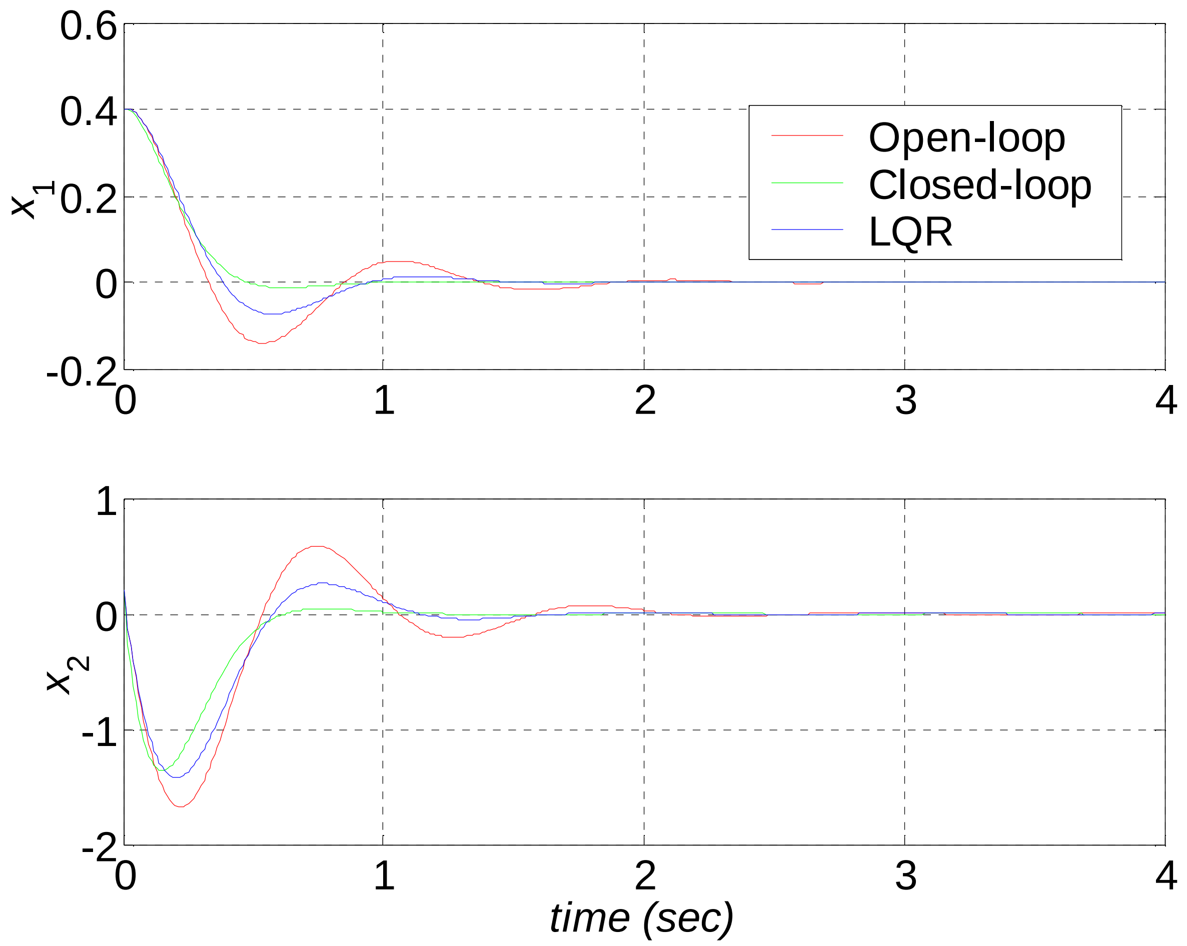

Figure 10.2 Open-, Closed-loop, and LQR State Responses for Matlab Example

We see that the LQR response follows the shape of the original open-loop system responses, but drives the states to zero faster. The error is less for the standard closed-loop system, but that controller was designed without regard to required input effort.

Figure 10.3 compares the single input effort required over time for the standard and LQR controllers in this example. No open-loop input is plotted because it is zero, i.e. no control effort is required at all, the open-loop system returns to zero from the given initial conditions.

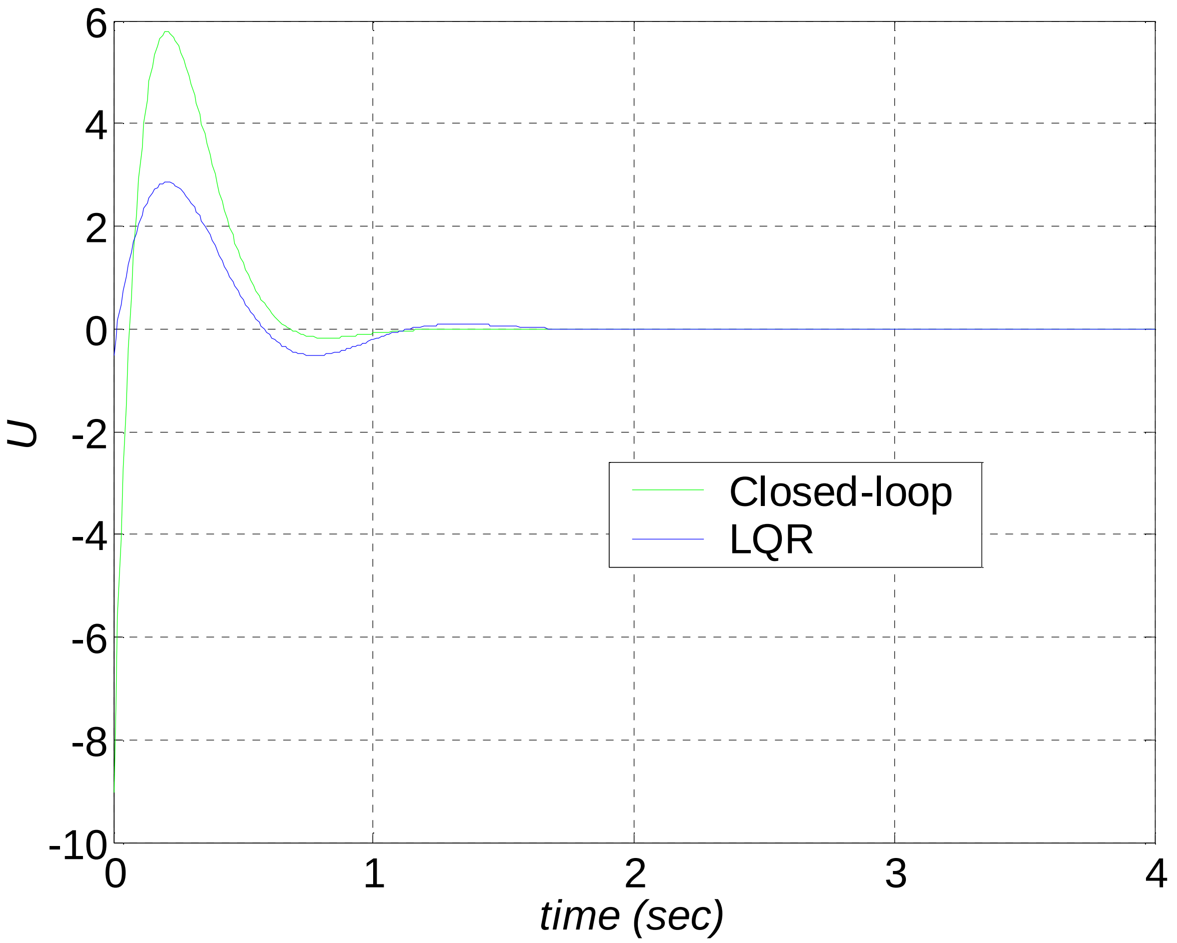

Figure 10.3 LQR Optimal Controller vs. Standard Controller Input Effort

Clearly from Figure 10.3, the LQR optimal controller is superior in terms of less input effort required for control. The standard closed-loop design does not attempt to minimize input efforts. We see that initially a relatively large (negative) input spike are required to get the standard controllers moving; the LQR case does not require this spike. Less energy is required to operate the LQR controller over time compared to the standard controller.

8.4 Continuing Example I: Linear Quadratic Regulator

For Continuing Example I (Two-mass MIMO Translational Mechanical System), design and evaluate in simulation an optimal LQR controller, for Case ii (SISO, zero input u_2, initial conditions $\mathbf{X}(0)=\{0.1 \quad 0 \quad 0.2 \quad 0\}^T$, and output y_1).

Solution, Case ii

Due to scaling in this problem, the weighting matrices for the LQR objective function J were chosen to be:

$$\mathbf{Q}=300\mathbf{I_4} \qquad\qquad \mathbf{R}=1$$

We use Matlab function *are* to solve the algebraic Ricatti equation for the optimal gain matrix **K**. Note that this LQR K plays a different role than the full-state-feedback gain matrix K, due to the changed control law: $\mathbf{U}=-\mathbf{R^{-1}B^TKX}$. Due to the different control law, the LQR closed-loop system dynamics matrix, for control simulation, is:

$$\mathbf{A_c}=\mathbf{A}-\mathbf{BR^{-1}B^TK}$$

The state vector results are plotted in Figure 10.4. We compare the optimal LQR controller responses with the original open-loop system state responses, and the standard controller from Chapter 8 Example I.

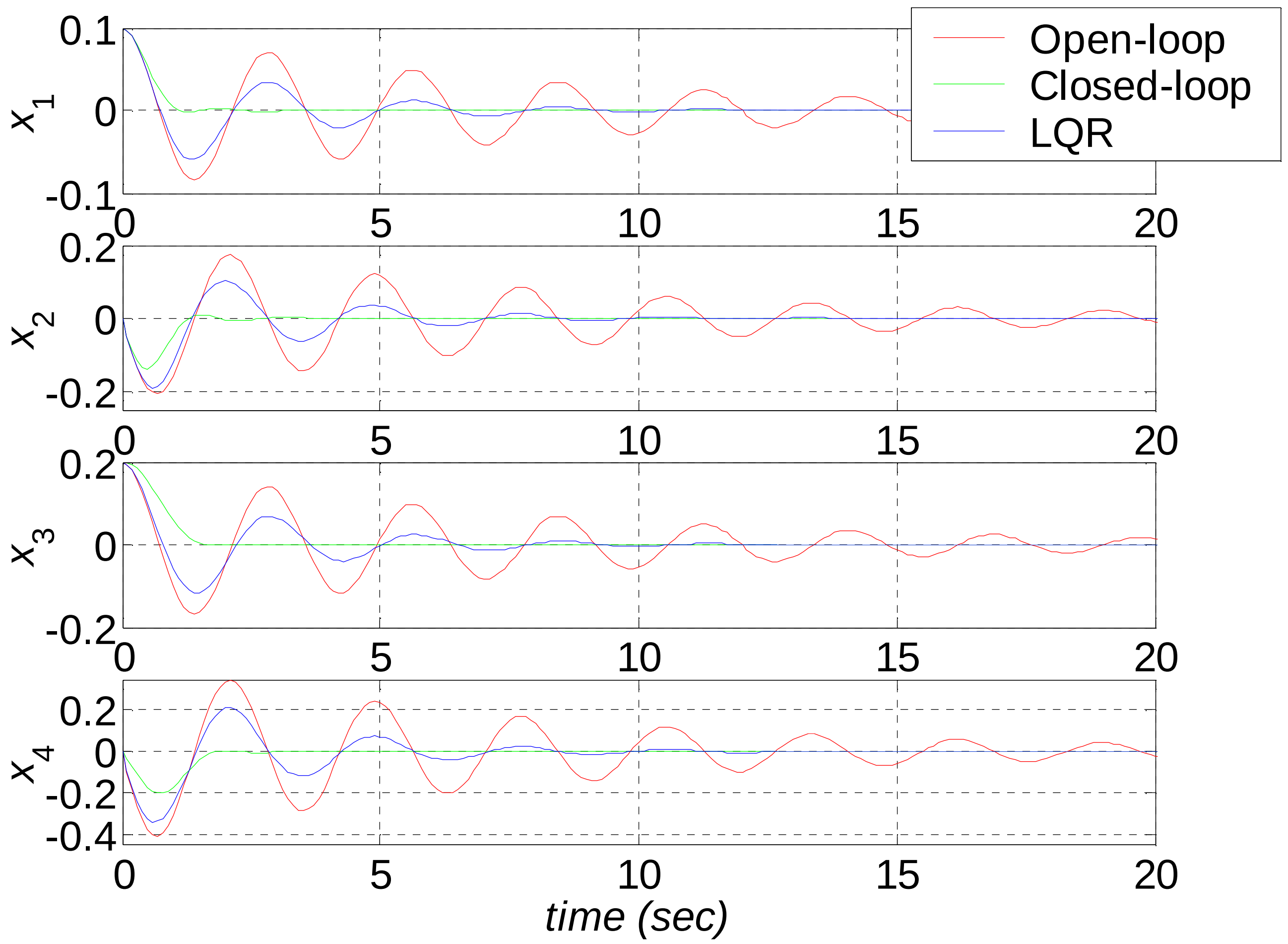

Figure 10.4 Open-, Closed-loop, and LQR Responses for Case ii

If we are used to standard controller design as in Chapter 8, at first glance we may think there is an error in the optimal LQR controller design responses in Figure 10.4. That is, the LQR overshooting and settling time is much greater in Figure 10.4 than we have seen in standard controller designs (such as the green closed-loop responses, from Chapter 8 and seen again in Figure 10.4). However, what is not evident in Figure 10.3 is that the input effort **U** is also reduced in addition to the state error **X**. We will return to this issue to conclude this section. The state error is much improved, upon close inspection of Figures 10.3: though the LQR (blue) is close to the open-loop responses (red) during the first period of motion, we see that the settling time is much better for the LQR than the open-loop.

Now, with regard to state error only, it appears that the LQR optimal controller is second best to the standard closed-loop controller. However, tight, fast responses with low error require a lot of input effort. Figure 10.5 compares the single input effort required over time for the standard and LQR controllers in this example. No open-loop input is plotted because it is zero, i.e. no control effort is required at all, the open-loop system returns to zero from the given initial conditions.

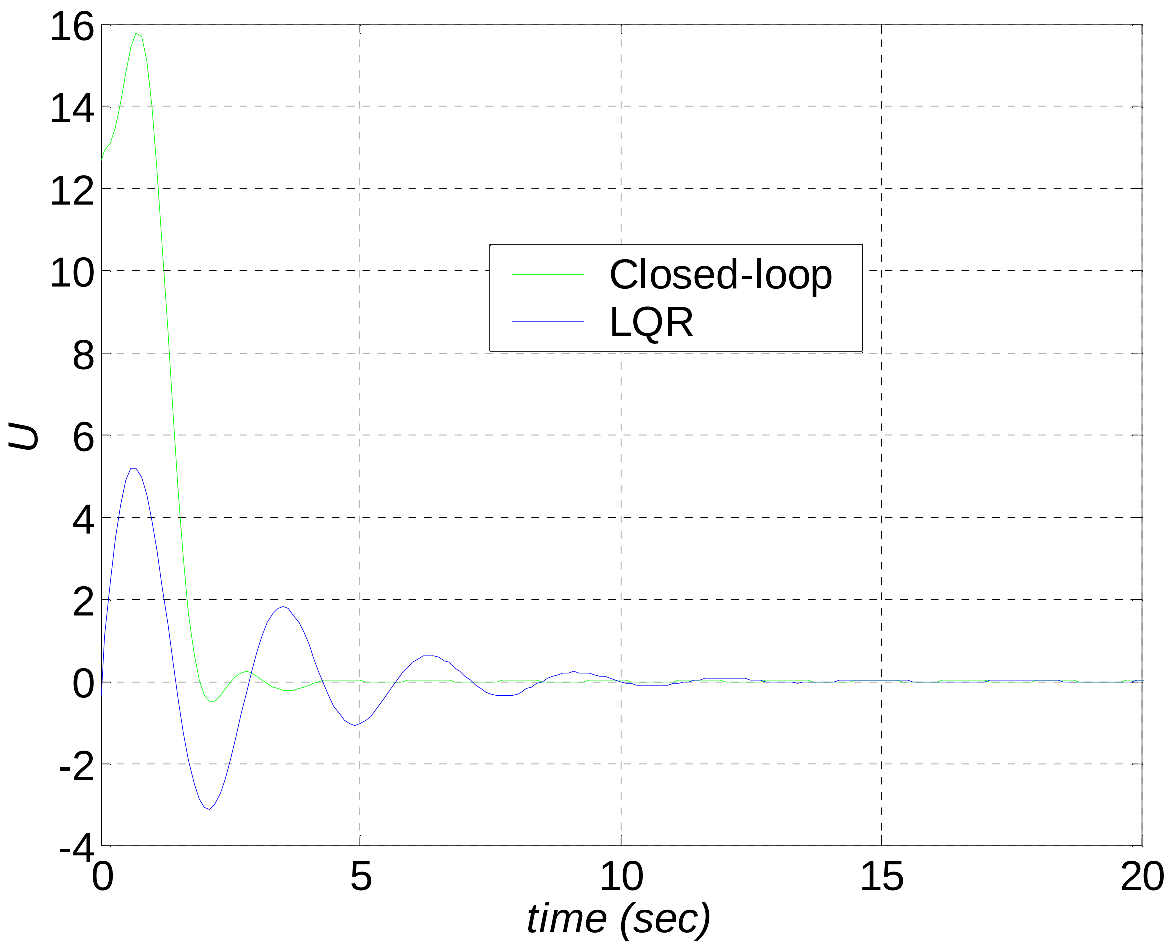

Figure 10.5 LQR Optimal Controller vs. Standard Controller Input Effort

Clearly from Figure 10.5, the LQR optimal controller is superior in terms of less input effort required for control. The standard controller design does not attempt to minimize input efforts. We see that initially a relatively large effort is required to get the standard controller moving; the LQR case starts near zero and clearly requires less energy than the standard controller. While the standard controller input effort goes to zero relatively quickly, the LQR case requires low inputs for longer time.

8.5 Homework Assignments

8.5.1 Mathematical Homework Assignments

8.5.2 Matlab Homework Assignments

8.5.3 Continuing Homework Assignments

CE1.8
Design and evaluate in simulation an LQR regulator for the CE1.3.i.b system. Use equal weighting between state error and input efforts. In addition to plotting the open- and closed-loop LQR responses, separately plot the input efforts required. Compare state responses and inputs efforts to the CE1.8 results.

CE2.8
Design and evaluate in simulation an LQR regulator for the CE2.3.i.b system. Use equal weighting between state error and input efforts. In addition to plotting the open- and closed-loop LQR responses, separately plot the input efforts required. Compare state responses and inputs efforts to the CE2.8 results.

CE3.8
Design and evaluate in simulation an LQR regulator for the CE3.3.i.b system. Use equal weighting between state error and input efforts. In addition to plotting the open- and closed-loop LQR responses, separately plot the input efforts required. Compare state responses and inputs efforts to the CE3.8 results.