ME 601

Advanced System Analysis & Control

Class Notes

Dr. Bob

Mechanical Engineering

Ohio University

© Dr. Bob Productions

williar4@ohio.edu

oak.cats.ohiou.edu/~williar4

Textbook:

R.L. Williams II and D.A. Lawrence, 2007, Linear State-Space Control Systems, John Wiley & Sons, ISBN 978-0-471-73555-7.

Table of Contents

ME 601 Advanced System Analysis & Control

1. Preliminaries

1.1 Introduction

Linear System

- Modeling
- Simulation (Analysis)
- Control

Systems:

Mechanical, electrical, electromechanical, fluid, thermal, etc. Lumped parameter systems to approximate real systems. Results only as good as your model.

State-space (modern, advanced) vs. Classical

System analysis and control

Classical	Modern
Linear	Non-linear
Continuous-time	Discrete
Analog	Digital
SISO	MIMO
Constant coefficients	Time-varying coefficients
Transfer function	State space matrices
Time or frequency domain	Time domain
Stability	Stability, robustness

We will focus on state-space, linear, continuous, SISO and MIMO, constant coefficients, time domain systems in ME 601.

Review

- Classical controls
- Solution to initial-value ODEs
- Linear algebra
- Modeling
- MATLAB, Simulink

Applications

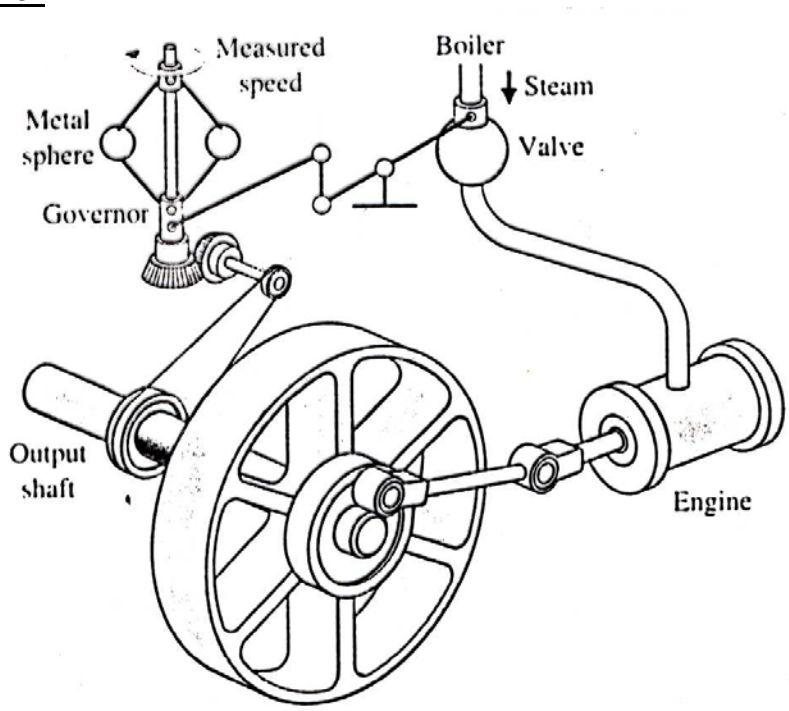
Feedback control systems are hugely widespread: robot systems, vehicles, human body, manufacturing automation, economic systems, predator-prey, etc. Engineering systems: mechanical, electrical, electromechanical, fluid, thermal, etc. Non-engineering systems – same theory applies if linear system.

History

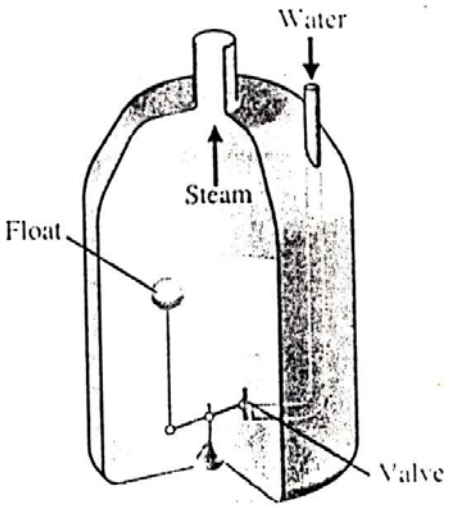
Mechanical engineers originated the field – show Watt and water level examples. But electrical engineers took over with advent of digital computers. Aerospace applications drove need for more capable controllers.

- Watt flyball governor
- Water-level float regulator
- Telephone system, electronic feedback amplifiers
- WWII autopilots, gunning, radar
- Digital computer
- Space age – missiles, spacecraft, robotics: robust, optimal, MIMO, state-space, non-linear

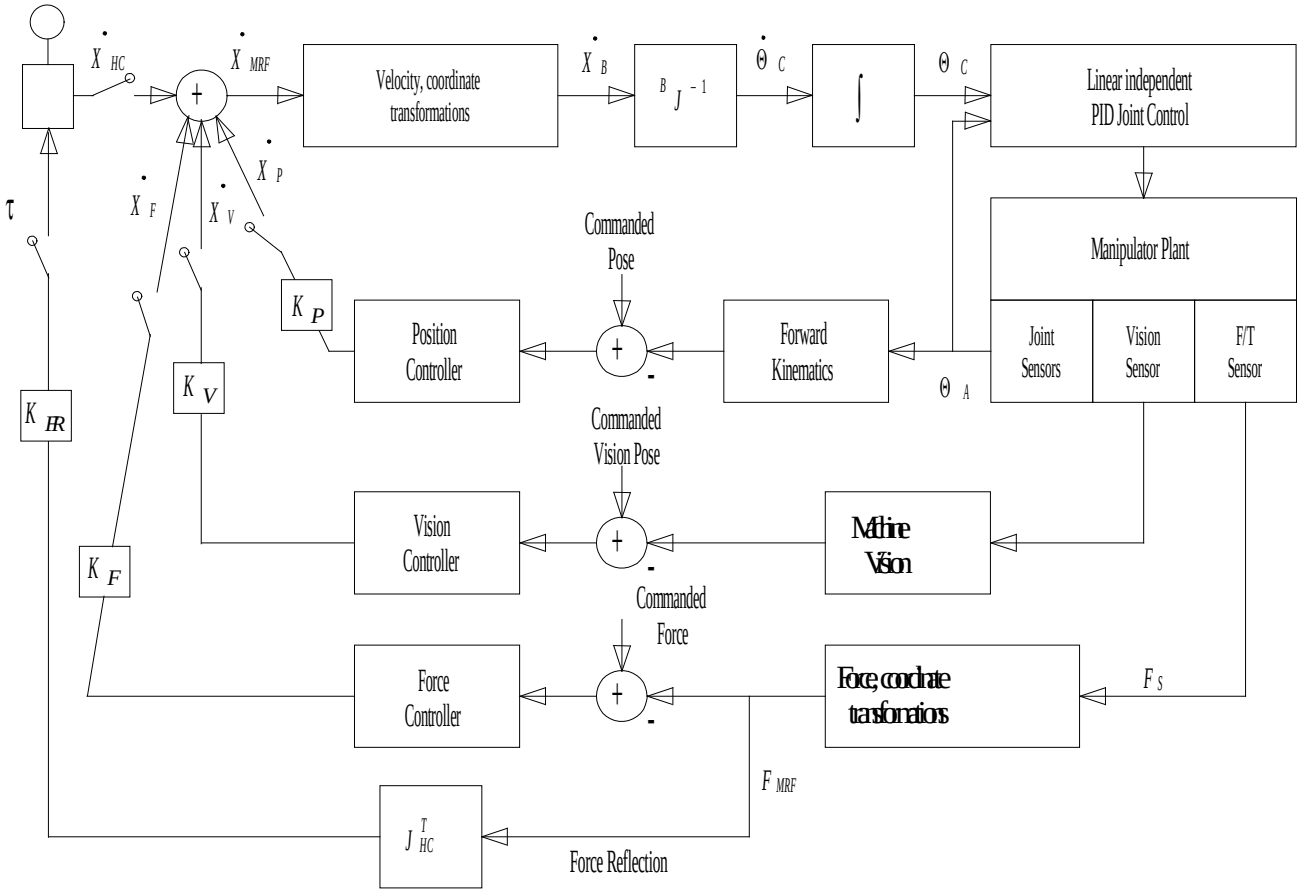
Watt Flyball Governor



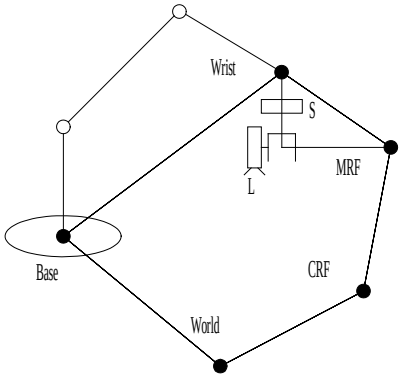
Water-level Float Regulator



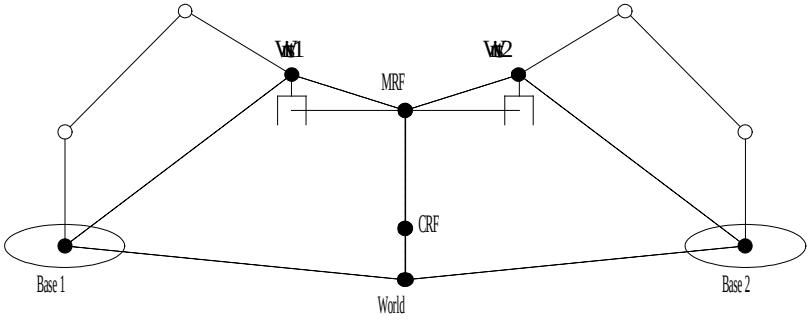
Sensor-based Control of Multiple Manipulator Systems



Sensor-based Telerobotic Control Architecture



Single Manipulator



Cooperating Manipulators

Classical controller block diagram

Why feedback? (it's not free - sensor, complexity, reliability, etc.)

- reduce steady-state error
- robustness for disturbances, unmodeled dynamics, noise, system changes
- decrease sensitivity of output to system changes, uncertainty
- modify transient response of the system

Linear System Definition

All governing equations (differential, algebraic) are linear. Linear systems satisfy the principles of **linear superposition** and **homogeneity**. Let $u(t)$ be the input and $y(t)$ be the output. Further, $u(t) \rightarrow y(t)$ indicates a system yielding output $y(t)$ given input $u(t)$.

1) linear superposition

if $u_1(t) \rightarrow y_1(t)$ and $u_2(t) \rightarrow y_2(t)$ then $u_1(t) + u_2(t) \rightarrow y_1(t) + y_2(t)$

2) homogeneity

if $u(t) \rightarrow y(t)$ then $\beta u(t) \rightarrow \beta y(t)$

where β is any constant.

Examples

linear ODEs:

$$a\dot{y}(t) + by(t) = u(t) \qquad \ddot{y}(t) + 2\xi\omega_n \dot{y}(t) + \omega_n^2 y(t) = u(t)$$

non-linear ODEs:

$$a\dot{y}^2(t) + b\sin y(t) = \sqrt{u(t)} \qquad \ddot{y}^3(t) + 2\xi\omega_n \sqrt{\dot{y}(t)} + \omega_n^2 \cos y(t) = \ln u(t)$$

Almost all real-world systems are non-linear: Coulomb friction, hysteresis, unmodeled dynamics, non-linear stiffness, large-angle dynamics, etc.

However, many engineering systems have

- linear operating ranges, or
- equations may be piecewise linearized in various operating regions

Model vs. actual system – a mathematical model is used to to design controller, which is then run on the actual system, with real-world dynamics, not a math model.

1.2 Classical Control Overview

ME 601 does not require classical control as a prerequisite. However some concepts and terms may enter into ME 601 from classical control so the student is urged to review from undergraduate controls textbooks such as Dorf and Bishop.

Here are some of the important topics for review:

- Modeling of mechanical, electrical, and other engineering systems to yield linear, constant-coefficient, ODEs and integro-differential equations, Force-current analogy
- Simulation: solution of linear ODEs subject to the initial value problem
- Laplace Transforms including table and differential and integral operators, Initial- and Final-Value Theorems
- Zero, impulse (Dirac Delta), unit step, unit ramp, sinusoidal, and other input signals
- First-order ODEs with time constant
- Second-order ODEs with natural frequency and dimensionless damping ratio
- Characteristic polynomials, Stability.
- Transfer Functions, Poles and Zeros, Block Diagrams
- Open- and Closed-loop Systems, Sensor Feedback
- Controller Design, Root-Locus Method
- Frequency Response, Bode Plots, and Nyquist Plots
- Digital Control Systems

1.3 Review of Matrices and Linear Algebra

Matrices are the mathematical vehicle of modern state-space control, in place of transfer functions for classical control. For a review of matrices including basics, arithmetic, determinants, and inversion, please see the Williams and Lawrence textbook, Appendix A.

Linear algebra is the mathematical language of matrices. For a review of linear algebra including vector spaces, subspaces, basis, change of basis, orthogonality, linear transformations, range, null space, eigenvalues, eigenvectors, and norms, please see the Williams and Lawrence textbook, Appendix B.

1.4 MATLAB Introduction

MATLAB is a general engineering analysis and simulation software. MATLAB stands for **MAT**rix **LAB**oratory. It was originally developed specifically for control systems simulation and design engineering, but it has grown over the years to cover many engineering and scientific fields. MATLAB is based on the C language, and its programming is vaguely C-like, but simpler. MATLAB is sold by Mathworks Inc. (www.mathworks.com) and Ohio University has a site license. For an extensive introduction to the MATLAB software, please see Dr. Bob's MATLAB Primer:

oak.cats.ohiou.edu/~williar4/PDF/MATLABPrimer.pdf

2. State-Space Fundamentals

2.1 State-Space Description of Dynamical Systems

Transfer functions are classically limited to linear SISO systems. A matrix of transfer functions can be used for extending classical control to MIMO. We will follow a different path, state-space matrices.

In ME 601 we assume the student already has modeling experience. For an overview of many useful models see oak.cats.ohiou.edu/~williar4/PDF/ModelTFAtlas.pdf.

State-space formulation

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

Basic open-loop system diagram: m inputs, p outputs

State vector $X(t)$

A state vector $X(t)$ is composed of state variables $x_i(t)$, $i = 1,2, \ldots, n$: a 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.

The state vector $X(t)$ is not the same as the output vector $Y(t)$. Outputs are physical quantities, usually measurable by sensors. State variables can be anything, not always recognizable quantities.

Basic motivation

Convert all dynamics and control system models to first-order ODEs.

State-space representation converts a single n^{th}-order ODE into a system of n coupled first-order ODEs, resulting in a coupled matrix of differential equations. In principle, this is easier to solve (via standardized methods which we shall present later). One can also convert a system of k n^{th}-order ODEs into a matrix system of kn coupled first-order ODEs.

State-space description

- State differential equations
- Output algebraic equations

Example

Linear 1-dof *m*-*c*-*k* mechanical translational system:

Where displacement $y(t)$ is the output and force $u(t)$ is the input.

The model is a single second-order ODE – we need to select a 2x1 state vector:

Define: so

Substitute into the original equation:

Original single second-order ODE can be written as a system of two first-order ODEs:

And the output is:

Write these equations in matrix-vector form to get:

State-space description

- State differential equations

- Output algebraic equations

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 including the acceleration). There are two state components required since we started with one second-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 DAEs

m	number of inputs
p	number of outputs
n	number of state variables (order of ODE for SISO)

- State differential equations: Matrix of first-order ODEs which represents the system dynamics. The solution of this set of equations yields the state vector $X(t)$.

- Output algebraic equations – calculates the output vector given the state vector and possibly the input vector.

A	System dynamics matrix	dimension:
B	Input matrix	dimension:
C	Output matrix	dimension:
D	Direct transmission matrix	dimension:

$[D] = [0]$, a zero matrix of appropriate size, for most physical systems because dynamics must appear in all paths from input to output.

The dimensions for the state differential equations and output algebraic equation for the second-order SISO system presented above are (m=1, p=1, n=2):

State-Space Description of a General SISO n^{th} order ODE

Define n state variables:

Substitute into the original ODE:

Derive the state and output equations:

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 *Controller Canonical Form*. We will consider several other canonical forms when we present similarity transformations later.

Block Diagram for Open-Loop MIMO State-Space Description

Another SISO Example

Parallel RLC circuit with current $i(t)$ input and voltage $v(t)$ output.

Integro-differential equation with single integral and time derivative – we need to select a 2x1 state vector:

Substitute these state definitions into the original equation:

Output:

Write these equations in matrix-vector form to get:

State-space description

- State differential equations

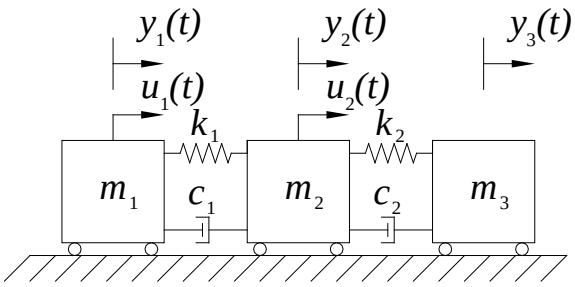
- Output algebraic equations

Recall the **Force-Current Analogy**:

Variable Type	Translational	*R-L-C* Circuit	Rotational
input (through)	$f(t)$	$i(t)$	$t(t)$
output (across)	$v(t)$ (velocity)	$v(t)$ (voltage)	$w(t)$
inertia	m	C	J
damping	c	$1/R$	c_R
stiffness	k	$1/L$	k_R

MIMO Example: 3-dof linear translational mechanical system

m = 2 inputs u_1, u_2 p = 3 outputs y_1, y_2, y_3



Free-body diagrams:

Write 3 equations of motion:

Define state variables:

Write these equations in matrix-vector form to get:

State-space description

- State differential equations

- Output algebraic equation

m=2; p=3; n=6 (three second-order ODEs converted to six first-order ODEs)

MIMO Example Results

Let:

$m_1 = m_2 = m_3 = m$
$c_1 = c_2 = c$
$k_1 = k_2 = k$

State differential equations $\{\dot{X}(t)\} = [A]\{X(t)\} + [B]\{U(t)\}$:

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

$$\{6x1\} = [6x6]\{6x1\} + [6x2]\{2x1\}$$

Output algebraic equation $\{Y(t)\} = [C]\{X(t)\} + [D]\{U(t)\}$:

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

$$\{3x1\} = [3x6]\{6x1\} + [3x2]\{2x1\}$$

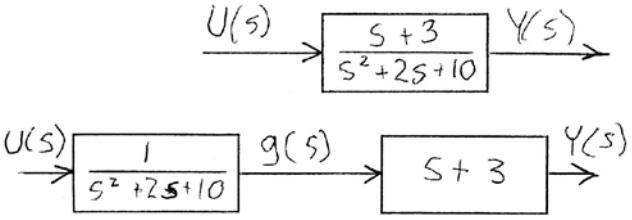
State-Space Description for a System with Zeros

(Zeros: roots of TF numerator)

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 the transfer function with intermediate variable $g(s)$:



Now, write two differential equations:

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

This is still a second-order system – need to select a 2x1 state vector: $X(t)=\begin{Bmatrix}x_1(t)\\x_2(t)\end{Bmatrix}$

Define: $x_1(t)=g(t)$, $x_2(t)=\dot{g}(t)=\dot{x}_1(t)$

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

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

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

$\{\dot{X}(t)\}=[A]\{X(t)\}+[B]\{U(t)\}$:

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

$\{Y(t)\}=[C]\{X(t)\}+[D]\{U(t)\}$:

$$y(t)=\begin{bmatrix}3&1\end{bmatrix}\begin{Bmatrix}x_1(t)\\x_2(t)\end{Bmatrix}+[0]u(t)$$

If the order of the highest derivatives are the same on both sides (i.e. the order of the numerator equals the order of the denominator), see Ogata pp. 78-80 for a state-space representation.

State-Space and Transfer Function Relationships

State-Space Equations:

Given state-space matrices A, B, C, D, find the transfer function description:

Laplace Transforms of State-Space Equations

For transfer functions we have zero initial conditions so the vector $x(0) = x_0 = \{0\}$:

Substitute this $X(s)$ into the output equation to eliminate the explicit state dependence:

Then:

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 – linear 1-dof m=1 c=0.1 k=10 mechanical system.

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

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

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

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

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

Check:

$$m\ddot{y}(t)+c\dot{y}(t)+ky(t)=u(t)$$
$$ms^2Y(s)+csY(s)+kY(s)=U(s)$$

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

Related MATLAB Functions

```
     [num,den] = ss2tf(a,b,c,d)
     [a,b,c,d] = tf2ss(num,den)
     SysName1  = tf(num,den)  % define system with transfer function
     SysName2  = ss(a,b,c,d)  % define system with state-space

%-------------------------------------------------------
%    State-space and transfer function transformations
%-------------------------------------------------------
clc; clear;

num = [1];                   % Transfer function definition,
den = [1 0.1 10];            %       second-order m-c-k system

[a,b,c,d] = tf2ss(num,den);  % State-space from TF
printsys(a,b,c,d)

[num1,den1] = ss2tf(a,b,c,d); % TF from State-space
printsys(num1,den1,'s')

%    MATLAB tf2ss result is different; try a,b,c,d my way
%         TF result should be the same.

a1 = [0 1;-10 -0.1];     b1 = [0;1];    c1 = [1 0];    d1 = [0];

[num2,den2] = ss2tf(a1,b1,c1,d1);  % TF from State-space
printsys(num2,den2,'s')
```

MATLAB Chooses:

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

Also, MATLAB reverses order of equations (swaps rows compared to mine). For multiple inputs $m > 1$, use:

```
[numi,deni] = ss2tf(a,b,c,d,iu);   % TF from State-space
```

where **`iu`** is the input index and **`[numi,deni]`** correspond to the transfer function for the i^{th} input only. **`deni`** is the characteristic polynomial and **`numi`** has one row for each output p.

2.2 Solution of State-Space Equations

State-Space Dynamics Differential Equations

Solve this coupled system of first-order ODEs for state vector $X(t)$, given input $U(t)$ and initial conditions x_0. Then the output is found from the linear combination output equation:

First, review **scalar first-order ODEs**

Physical systems with scalar first-order ODEs

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

See oak.cats.ohiou.edu/~williar4/PDF/ModelTFAtlas.pdf for info on first-order and other models.

Solution methods

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

Example

solve $c\dot{x}(t)+kx(t)=\dot{x}(t)+50x(t)=u(t)$

subject to: $u(t)$ (step input of magnitude 5) and $x(0) = 0$

use Laplace transform method:

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

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

$$X(s)=\frac{5}{s(s+50)}=\frac{C_1}{s}+\frac{C_2}{s+50}=\frac{C_1(s+50)+C_2 s}{s(s+50)}$$

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

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

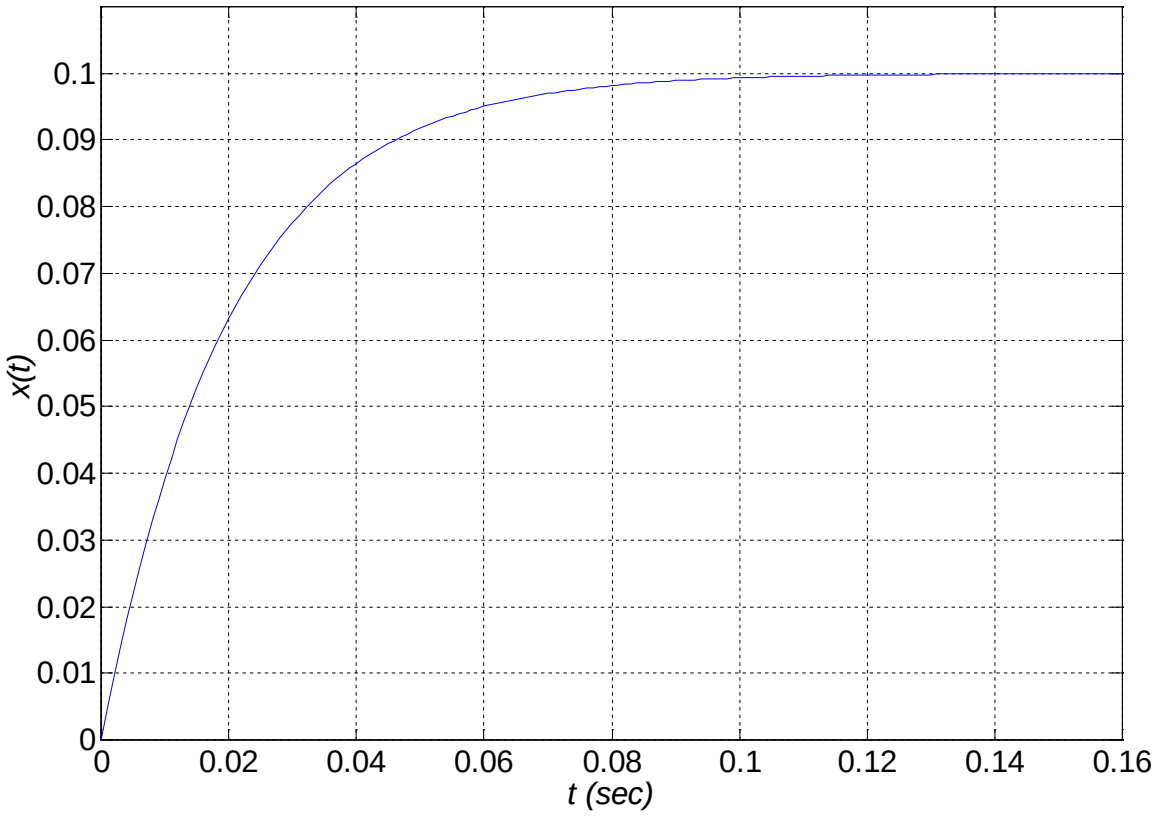
The $x(t)$ plot for this example is given below.

First-order System Example 1: *ck* system

Model: $c\dot{x}(t)+kx(t)=u(t)$ $\dot{x}(t)+50x(t)=u(t)$

$x(0) = 0$ and $u(t)$ = step input of magnitude 5

Solution: $x(t)=0.1\left(1-e^{-50t}\right)$



The displacement $x(t)$ starts at zero, as specified by the initial condition. The solution transient approaches zero by t = 0.15 sec. The steady-state displacement is x_{SS} = 0.10 *m* (a constant since the input force is constant); this result agrees with Hooke's Law: $u = kx_{SS}$; $x_{SS} = 5/50 = 0.10$.

Verify initial and steady-state values using the Initial- and Final-Value theorems:

Initial Value Theorem

$$\lim_{t\to 0} x(t)=\lim_{s\to\infty} sX(s)=\lim_{s\to\infty} s\left(\frac{5}{s(s+50)}\right)=0$$

Final Value Theorem

$$\lim_{t\to\infty} x(t)=\lim_{s\to 0} sX(s)=\lim_{s\to 0} s\left(\frac{5}{s(s+50)}\right)=0.1$$

Time constant

τ: $Ae^{-t/\tau}$

So in this example τ=1/50 *sec*. A system with a smaller time constant rises faster than one with a larger time constant. After a time equal to three time constants has elapsed, the step response of a first-order system is within 5% of its final steady-state value. In this example, three time constants is 3/50 or 0.06 *sec*. At t=0.06 *sec*, the $x(t)$ value is 0.095 *m*, 95% of the final value x_{SS} = 0.100.

Three common first-order system time constants for the massless spring/damper, springless rotational, and series voltage-driven *RL* circuit systems, respectively, are:

$$\tau = \frac{c}{k} \qquad \tau = \frac{J}{c_R} \qquad \tau = \frac{L}{R}$$

τ units for this example? $\frac{Nsm}{mN} = s$. This is general, i.e. all system time constants have *sec* units.

General Solution Form to Scalar first-order ODEs

(in the form of the matrix differential equations)

The first part is the ***homogeneous solution***: the transient response to the initial condition $x(0)$.

The second part is the ***particular solution***: the steady-state response to the forcing function $u(t)$.

Recall exponential function can be represented by an infinite series:

Generalize Scalar Solution Form to Matrix of first-order ODEs

The first part is the ***homogeneous solution***: the transient response to the initial conditions $X(0)$.

The second part is the ***particular solution***: the steady-state response to the forcing functions $U(t)$.

More properly, the first part is known as the ***zero-input response*** and the second part is known as the ***zero-state response*** (meaning zero initial state). That is, it doesn't split so neatly as stated above, but there is a transient component due to the forcing function also.

Φ(*t*) State Transition Matrix

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

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

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

- $\Phi(t) = L^{-1}\left\{(sI - A)^{-1}\right\}$ (see general solution form above)

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

Properties of the State Transition Matrix Φ(*t*)

1) $\Phi(0) = I$

2) $\Phi^{-1}(t) = \Phi(-t)$

3) $\Phi(t_2 - t_1)\Phi(t_1 - t_0) = \Phi(t_2 - t_0)$

4) $\left[\Phi(t)\right]^k = \Phi(kt)$ For positive integers *k*

Characteristic Polynomial and System Poles

Classical Control

Transfer function:

Characteristic polynomial:

System poles (assuming a fully-reduced $G(s)$):

Modern Control

State-space description:

Characteristic polynomial:

System poles (assuming a minimal state-space realization):

What are these? Eigenvalues of A! B, C, and D do not affect the system poles.

```
CharPoly = poly(A);
poles    = eig(A);
```

Poles and Transient Response

The system poles determine the nature of the transient response. For example, solve:

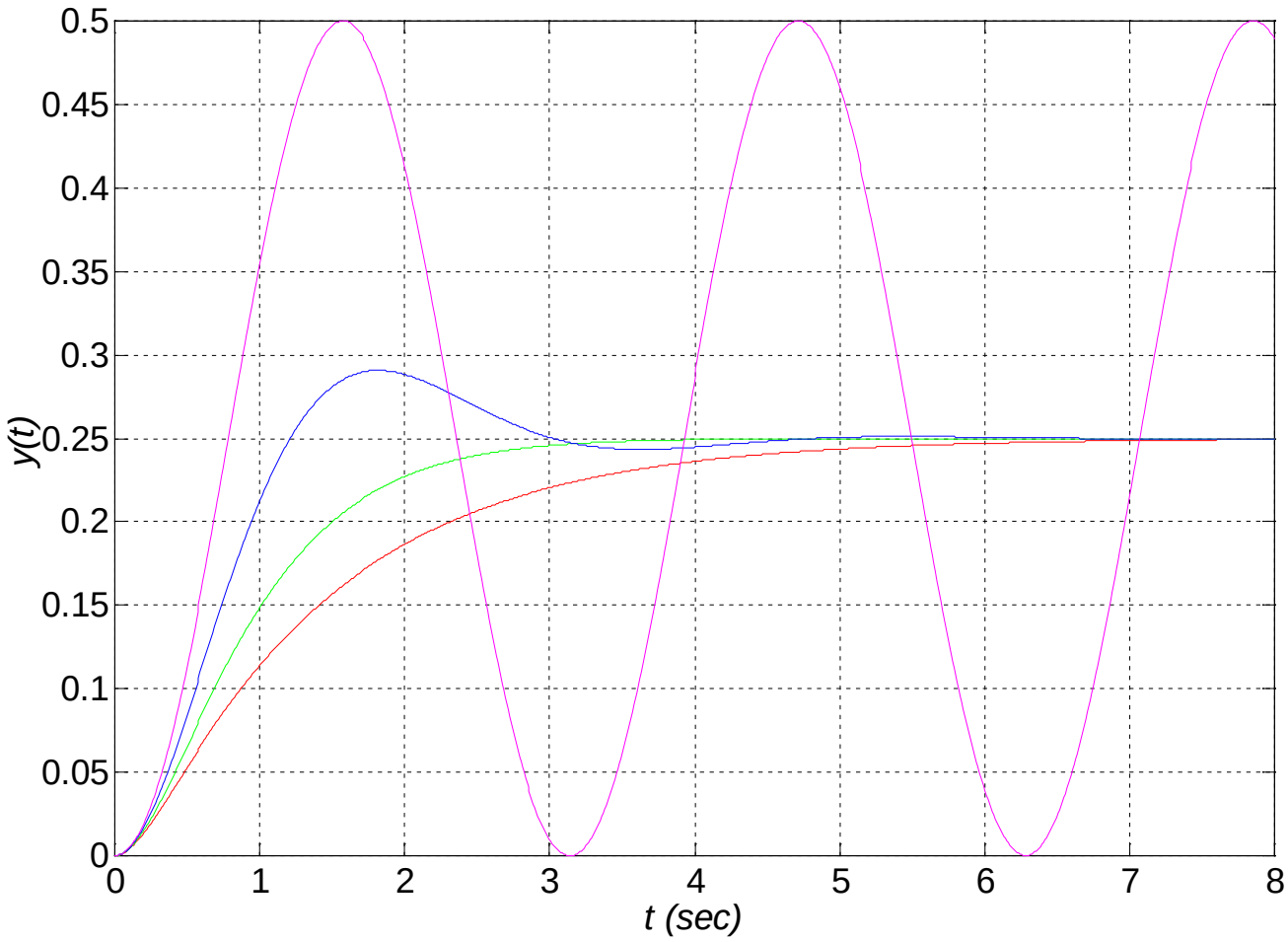
$$\begin{aligned} \ddot{y}(t)+6\dot{y}(t)+4y(t)&=1\\ \ddot{y}(t)+4\dot{y}(t)+4y(t)&=1\\ \ddot{y}(t)+2\dot{y}(t)+4y(t)&=1\\ \ddot{y}(t)+4y(t)&=1 \end{aligned} \quad \text{for } y(t)$$

All subject to initial conditions $\begin{aligned} y(0)&=0\\ \dot{y}(0)&=0 \end{aligned}$ and a unit step input.

MATLAB program:

```
%-----------------------------------------------------------------------
%  Over-, Critical, Under-, and Undamped cases using step function
%    Dr. Bob, ME 601
%-----------------------------------------------------------------------

clear; clc;

num = [1];
denOVER = [1 6 4];          % overdamped
denCRIT = [1 4 4];          % critically-damped
denUNDR = [1 2 4];          % underdamped
denUN   = [1 0 4];          % undamped

polesOVER = roots(denOVER); % poles for each case
polesCRIT = roots(denCRIT);
polesUNDR = roots(denUNDR);
polesUN   = roots(denUN);

t = [0:0.01:8];
[yOVER,xOVER] = step(num,denOVER,t);  % unit step responses
[yCRIT,xCRIT] = step(num,denCRIT,t);
[yUNDR,xUNDR] = step(num,denUNDR,t);
[yUN,xUN]     = step(num,denUN,t);

figure;
plot(t,yOVER,'r',t,yCRIT,'g',t,yUNDR,'b',t,yUN,'m'); % plot responses
set(gca,'FontSize',18);
grid; ylabel('\ity(t)'); xlabel('\itt (\itsec)');
```

Results plot:



Key:

overdamped
critically-damped
underdamped
undamped

Poles:

$s_1 = -0.764,\ s_2 = -5.236$
$s_1 = s_2 = -2$
$s_{1,2} = -1 \pm \sqrt{3}\mathrm{i}$
$s_{1,2} = \pm 2\mathrm{i}$

The final value is ¼ (0.25).

State-Space Simulation Example

Solve the given linear SISO second-order system for $y(t)$ subject to $u(t)$ (step input of magnitude 3), and initial conditions $y(0), \dot{y}(0)$:

$$\ddot{y}(t)+7\dot{y}(t)+12y(t)=u(t) \qquad \begin{matrix} y(0)=0.10 \\ \dot{y}(0)=0.05 \end{matrix}$$

Characteristic polynomial and poles:

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

There are distinct, negative real roots so this is an overdamped system. By either the slow ME way, using Laplace transforms, or MATLAB **`dsolve`**, 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:

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

Then the system dynamics differential equations and initial state vector are (the input $u(t)$ is a scalar step input of magnitude 3):

$$\begin{Bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{Bmatrix}=\begin{bmatrix} 0 & 1 \\ -12 & -7 \end{bmatrix}\begin{Bmatrix} x_1(t) \\ x_2(t) \end{Bmatrix}+\begin{bmatrix} 0 \\ 1 \end{bmatrix}\{u(t)\} \qquad X(0)=\begin{Bmatrix} 0.10 \\ 0.05 \end{Bmatrix}$$

Solve in the Laplace frequency domain:

$$X(s)=(sI-A)^{-1}X(0)+(sI-A)^{-1}BU(s)$$

$$\text{then} \qquad \{X(t)\}=L^{-1}\{X(s)\}$$

$$(sI-A)=\begin{bmatrix} s & -1 \\ 12 & s+7 \end{bmatrix} \qquad (sI-A)^{-1}=\frac{1}{\Delta}\begin{bmatrix} s+7 & 1 \\ -12 & s \end{bmatrix}$$

$\Delta=s^2+7s+12=(s+3)(s+4)$ is $|sI-A|$, the characteristic polynomial.

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

where the Laplace transform of the unit step function is $\frac{1}{s}$. Simplifying:

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

$$x_1(s) = \frac{0.10s + 0.75 + \dfrac{3}{s}}{(s+3)(s+4)}$$

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

(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 for the residues C_i; the result is:

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

Substitute the residues into the partial fraction expansion and take inverse Laplace transform to find *y*(*t*):

$$y(t) = x_1(t) = L^{-1}\{x_1(s)\}$$

$$= L^{-1}\left\{\frac{0.25}{s} - \frac{0.55}{(s+3)} + \frac{0.40}{(s+4)}\right\}$$

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

Agrees with the stated solution! Now find the solution for the 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:

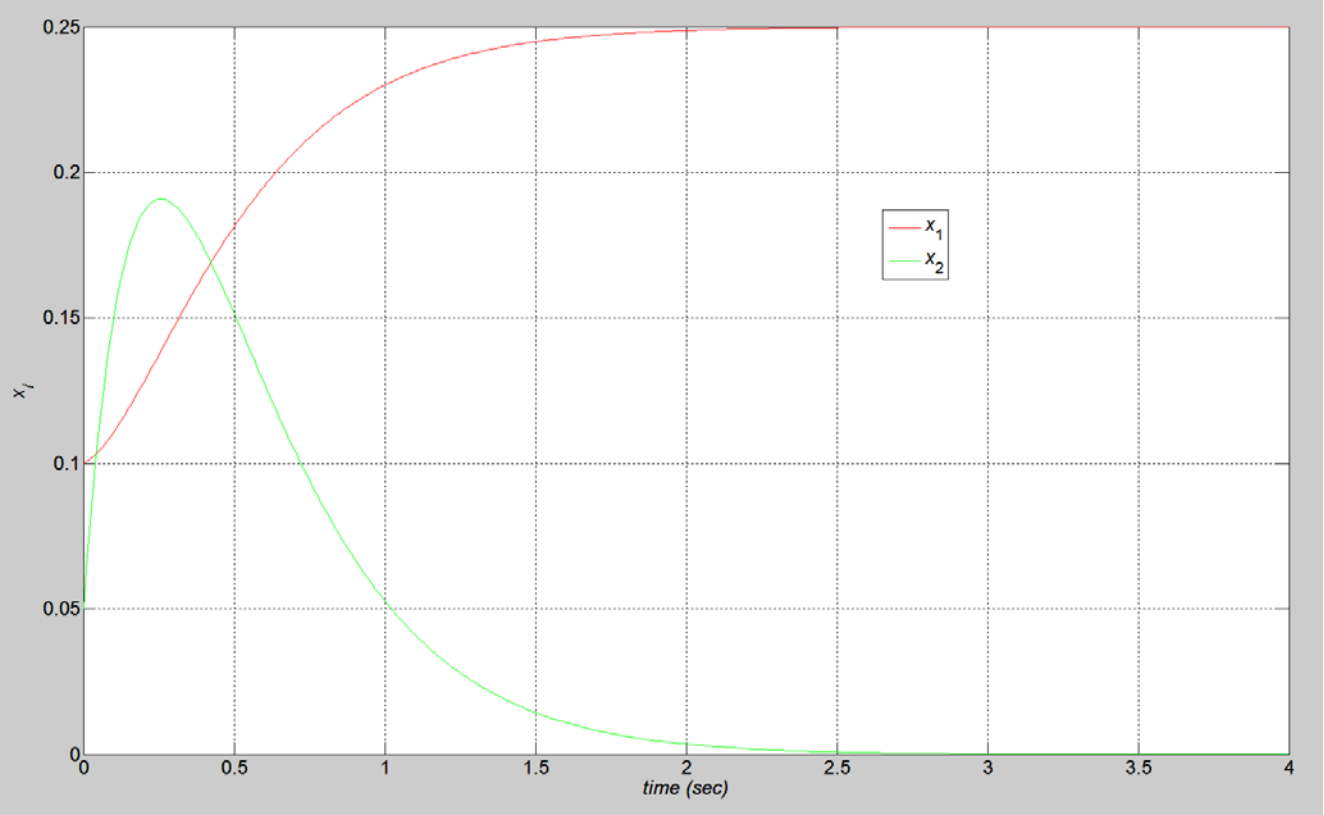
$$C_1 = 1.65$$
$$C_2 = -1.60$$

$$x_2(t) = L^{-1}\{x_2(s)\}$$

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

Check: $x_2(t) = \dot{x}_1(t)$? $\dot{x}_1(t) = -(-3)0.55e^{-3t} + (-4)0.40e^{-4t} = 1.65e^{-3t} - 1.60e^{-4t}$ Yes, it agrees!

The plot below gives the graphical results for this state-space simulation example, plotting the state components vs. time. The output $y(t)$, not shown, is identical to the first state $x_1(t)$.



Note that both $x_1(t)$ and its derivative $x_2(t)$ both start from their required initial conditions, 0.1 and 0.05, respectively. The slope of $x_1(t)$ at $t = 0$ is 0.05.

Though the problem did not specify, let us assume $x_1(t)$ is a displacement with m units and $x_2(t)$ is its velocity with m/sec units.

The displacement $x_1(t)$ plot is overdamped since the poles are negative, real, and distinct ($s_{1,2} = -3, -4$). The final value is 0.25, that is the force 3 N divided by the spring 12 N/m. Later in the course we will review rise time and settling time to quantify the time-rise nature of second-order systems such as this one (here t_R = 0.99 sec and t_S = 1.72 sec).

The velocity $x_2(t)$ plot is also overdamped. The final value is 0, since the velocity eventually goes to zero.

2.3 Simulation of State-Space Systems

By simulation we mean solving and then plotting $\dot{X}(t) = AX(t) + BU(t)$ for the states $X(t)$, given matrices A and B, input vector $U(t)$, and initial states X_0. In general this is a coupled set of n first-order linear ODEs. Once $X(t)$ is known, we can easily find the output with $Y(t) = CX(t) + DU(t)$. The previous section discussed the analytical solution; this section presents MATLAB numerical solution for this problem, with examples.

MATLAB Function **`lsim`**: **`l`**inear **`sim`**ulation

```
SysName = ss(a,b,c,d);
[y,t,x] = lsim(SysName,u,t,x0);
```

where:

- function `ss` creates a MATLAB system `SysName` from state-space matrices `a,b,c,d`.
- `a,b,c,d` are the numerical state-space matrices for your system.
- `t` is the evenly spaced time vector: `t = [t0:dt:tf];` As shown in the MATLAB syntax above, `t` is both input and output (I didn't write it).
- `u` is the matrix of inputs; `u` must have as many columns as inputs (m). Each row of `u` corresponds to a new time point, so `u` must have `length(t)` rows.
- `y` is the matrix of outputs; `y` has as many columns as there are outputs (p) and `length(t)` rows.
- `x` is the matrix of states; `x` has as many columns as there are states (n) and `length(t)` rows.
- `x0` is the n x 1 vector of initial states (`x0` is assumed by MATLAB to be zero if it is omitted).

Note: the above **`lsim`** code will not plot anything – the user must then plot the desired variables vs. time to see the responses:

```
figure;                                                    % plot outputs
plot(t,y(:,1),'r',t,y(:,2),'g',t,y(:,3),'b', . . .);
figure;                                                    % plot states
plot(t,x(:,1),'r',t,x(:,2),'g',t,x(:,3),'b', . . .);
```

MATLAB **`lsim`** is general – any imaginable inputs `u` can be specified. There are two useful MATLAB functions to determine system response for two specific inputs (unit impulse and unit step):

```
[y,t,x] = impulse(SysName,t);      % unit impulse response

[y,t,x] =    step(SysName,t);      % unit step response
```

Example 1: Solve second-order ODE

Solve $\ddot{y}(t)+0.2\dot{y}(t)+10y(t)=5$ for $y(t)$ given u (step of 5) and subject to initial conditions:

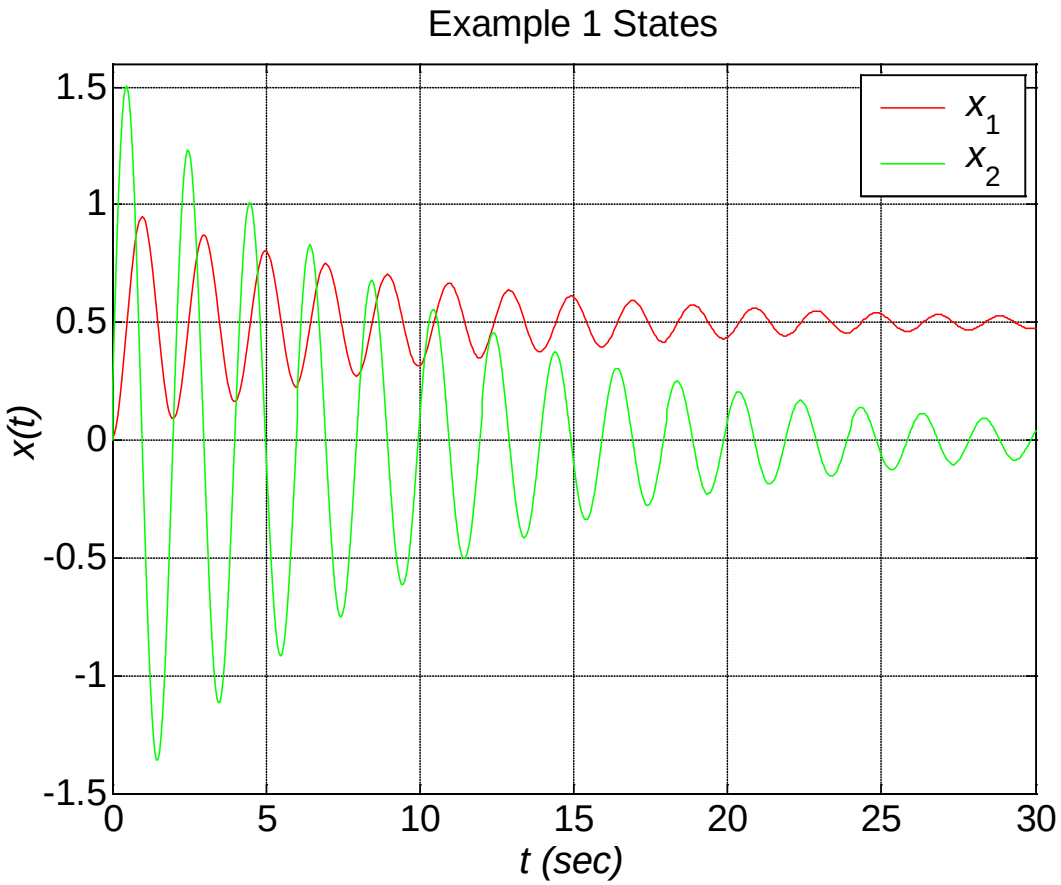
$$y(0)=0$$
$$\dot{y}(0)=0$$

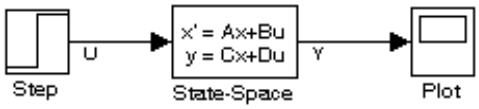
For the following state definitions, The state-space matrices are:

$$x_1(t)=y(t)$$
$$x_2(t)=\dot{y}(t)=\dot{x}_1(t)$$

$$A=\begin{bmatrix}0 & 1\\ -10 & -0.2\end{bmatrix} \qquad B=\begin{bmatrix}0\\ 1\end{bmatrix} \qquad C=\begin{bmatrix}1 & 0\end{bmatrix} \qquad D=[0]$$

The state responses are plotted vs. time on the next page. This is a stable, underdamped system (the poles are $s_{1,2}=-0.10\pm 3.16i$). We see it is still vibrating after 30 *sec*. The steady-state 'position' state x_1 is 5/10 = 0.5 and the steady-state 'velocity' state x_2 is zero. For these definitions, the output $y(t)$ (not shown) is identical to the plot of $x_1(t)$.

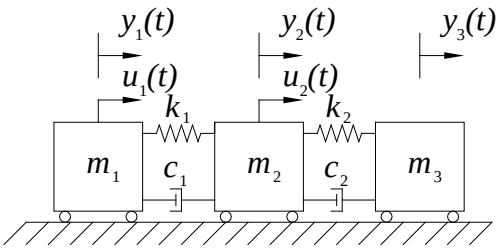




Example 1 Simulink Diagram

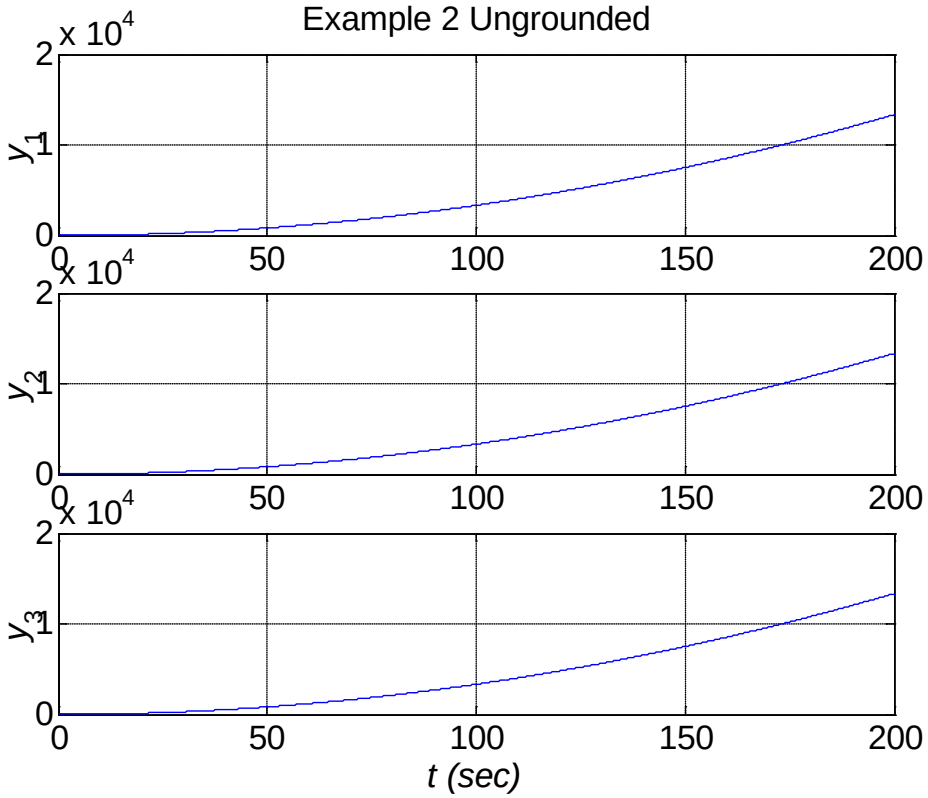
MATLAB's GUI Simulink is very powerful to simulate state-space systems. The Simulink diagram for solving this example is also shown on the next page. In this case, double-click on the input step to assign a step of 5 (the default is 1) and double-click on the State-Space block to enter specific arrays for the state-space matrices *A*, *B*, *C*, *D*. The result for $y(t)$ is identical to $x_1(t)$ above.

Example 2: Three mass, two input, ungrounded system



Given zero initial conditions and u_1 = step of magnitude 2, u_2=0, calculate the response y_1, y_2, y_3. Given $m_1 = m_2 = m_3 = 1$ *kg*, $c_1 = c_2 = 0.1$ *Ns/m*, and $k_1 = k_2 = 10$ *N/m*. Can we predict what will happen? The state-space matrices are below and the system poles are $s_{1,2} = -0.15 \pm 5.48i$, $s_{3,4} = -0.05 \pm 3.16i$, $s_5 = s_6 = 0$:

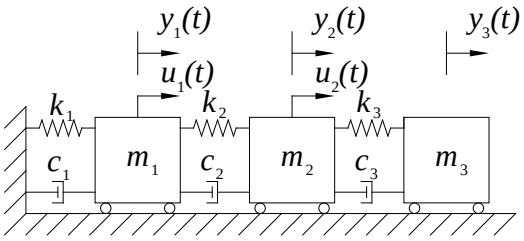
$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -10 & -0.1 & 10 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 10 & 0.1 & -20 & -0.2 & 10 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 10 & 0.1 & -10 & -0.1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$



With no spring tying the system to ground, with no friction to slow it down, and with a constant force u_1, the three-mass system will accelerate forever to the right! The zero eigenvalues (system poles, characteristic equation roots) tell us that there is a rigid body mode, where all three masses move as one to the right. (Note: mass 1 leads 2, which leads 3.)

Example 3: Three mass, two input, grounded system

This is the same as Example 2, except we will now tie the first mass m_1 to the ground with another c, k:

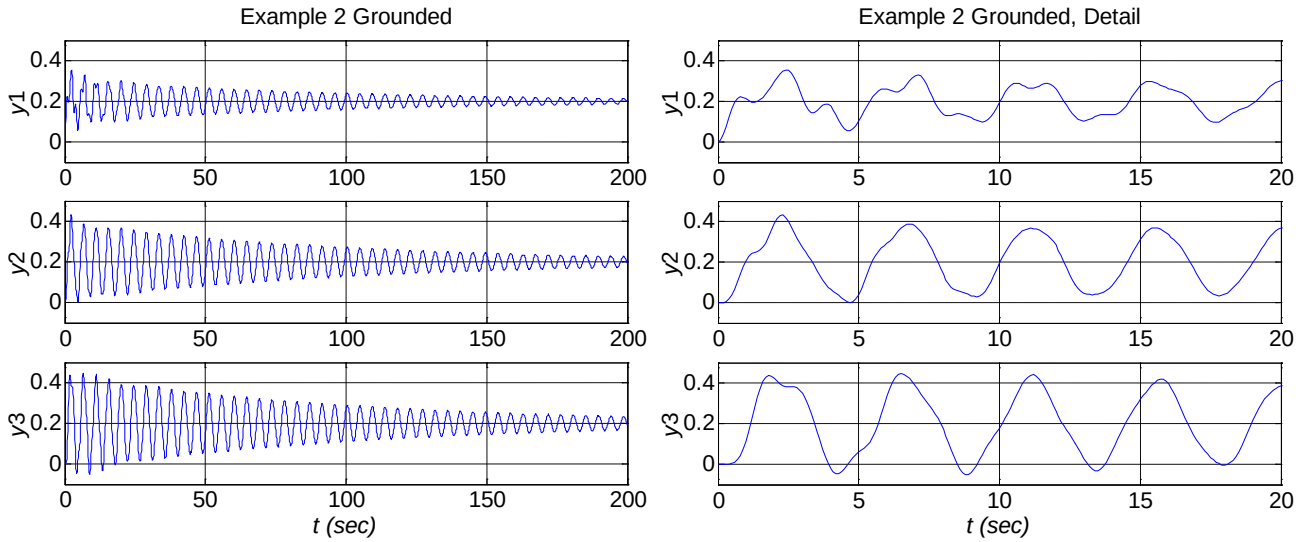


We use three identical mass, spring, damper coefficients (from Example 2). Given zero initial conditions and u_1 = step of magnitude 2, $u_2 = 0$, calculate the response y_1, y_2, y_3. Only the A matrix has changed (only two elements, 21 and 22 were doubled) and thus the poles have changed:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -20 & -0.2 & 10 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 10 & 0.1 & -20 & -0.2 & 10 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 10 & 0.1 & -10 & -0.1 \end{bmatrix}$$

$$s_{1,2} = -0.16 \pm 5.70i \qquad s_{3,4} = -0.08 \pm 3.94i \qquad s_{5,6} = -0.01 \pm 1.41i$$

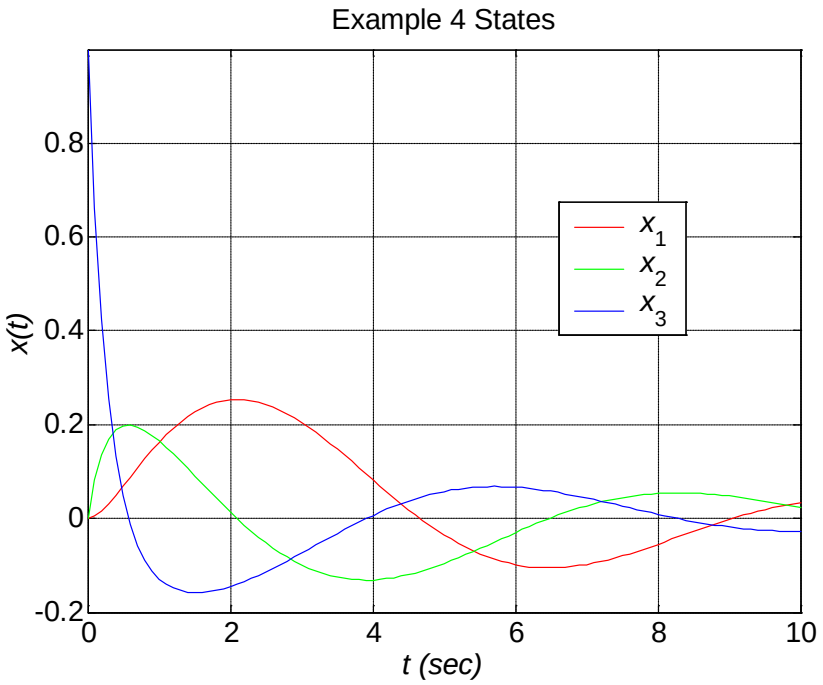
The output plots for this case are given below, for two different time scales. These are bounded, cyclical, vibrating outputs. (Note: mass 1 still leads 2, which leads 3.)



Example 4: Transient response to initial conditions

Given the following system, determine the response to initial conditions (the homogeneous solution, also called the zero input solution). The poles are $s_{1,2} = -0.20 \pm 0.72i$ and $s_3 = -3.60$.

$$[A] = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -2 & -2 & -4 \end{bmatrix} \quad [B] = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad [C] = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \quad [D] = [0] \quad X(0) = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix} \quad U(t) = 0$$



At t = 10 *sec*, the final state values are (from looking at the MATLAB **`lsim`** data):

$$X(10) = \begin{Bmatrix} 0.034 \\ 0.023 \\ -0.028 \end{Bmatrix}$$

Compare this to calculating final state using the state transition matrix $\Phi(t)$ in MATLAB:

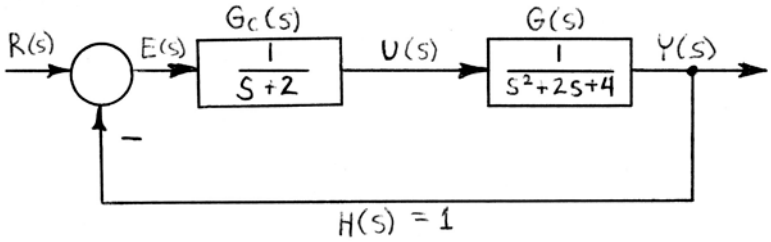
```
t   = 10;
Phi = expm(A*t);
x0  = [0;0;1];
x10 = Phi * x0;
```

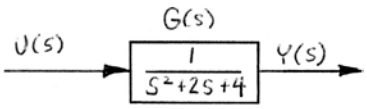
$$\Phi(10) = \begin{bmatrix} 0.132 & 0.160 & 0.034 \\ -0.069 & 0.064 & 0.023 \\ -0.046 & -0.115 & -0.028 \end{bmatrix}$$

The last column of $\Phi(10)$ is the response of x_1, x_2, x_3 to an initial condition of 1 on the third state x_3, which agrees with the above result $X(10)$.

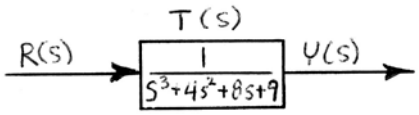
Example 5: Impulse response of open- and closed-loop systems

Block diagrams of open- (left) and closed-loop (right) systems in transfer function form:





$s_{1,2} = -1 \pm 1.73i$



$s_{1,2,3} = -0.88 \pm 1.80i, -2.22$

Open-loop system:

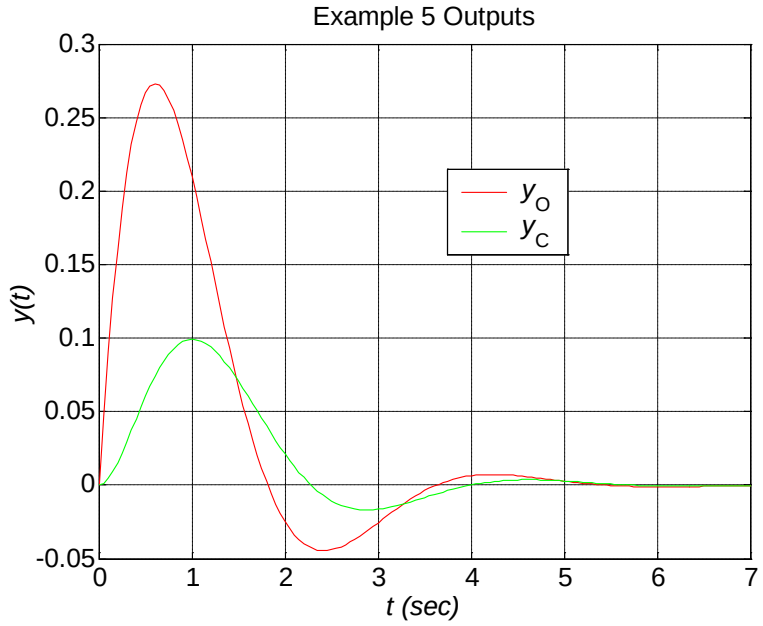
$$[A_o] = \begin{bmatrix} 0 & 1 \\ -4 & -2 \end{bmatrix} \qquad [B_o] = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad [C_o] = [1 \quad 0] \qquad [D_o] = [0]$$

Closed-loop system:

$$[A_c] = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -9 & -8 & -4 \end{bmatrix} \qquad [B_c] = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad [C_c] = [1 \quad 0 \quad 0] \qquad [D_c] = [0]$$

MATLAB code:

```
t = [0:0.1:7];
[yo,t,xo] = impulse(ao,bo,co,do,t);
[yc,t,xc] = impulse(ac,bc,cc,dc,t);
figure;  plot(t,yo,'r',t,yc,'g');
```



The closed-loop response y_C is preferable to the open-loop response y_O in terms of lower amplitude; it is similar to the open-loop in terms of time response (the closed-loop response lags).

Example 6: State-Space representation is not unique

Given two third-order SISO systems:

$$A_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -5 & -8 \end{bmatrix} \qquad B_1 = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} \qquad C_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad D_1 = [0]$$

$$A_2 = \begin{bmatrix} 0.5 & 0.5 & 0.707 \\ -0.5 & -0.5 & 0.707 \\ -6.364 & -0.707 & -8.0 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} \qquad C_2 = \begin{bmatrix} 0.707 & 0.707 & 0 \end{bmatrix} \qquad D_2 = [0]$$

Determine the transfer function representations (using MATLAB function **`ss2tf`**):

$$G_1(s) = \frac{4}{s^3 + 8s^2 + 5s + 4} \qquad G_2(s) = \frac{4}{s^3 + 8s^2 + 5s + 4}$$

These transfer functions are identical, so both state-space realizations are valid (this example demonstrates that state-space representation is not unique).

The poles are identical (**`roots(den)`**): $-7.397, -0.301 \pm 0.671i$

The poles could also be found from the eigenvalues of either A matrix – eigenvalues are invariant and equal to poles of the system (for minimal realizations). They are the roots of the characteristic equation.

However, the eigenvectors of A_1 and A_2 are different.

2.4 Controllability and Observability

Conditions of ***controllability*** and ***observability*** govern the existence of a complete solution to the control system design problem in state space. Introduced by Kalman (of the Kalman filter) – he is the father of modern linear state-space methods. He developed these concepts as the first step in complete control system design. Kalman flunked his prelims at MIT; apparently his ideas were too modern at the time (early 1960's). He then 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)
- **Observable** if all initial states x_i can be determined from at least one output y_j (sensors)

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

2.4.1 Controllability

The continuous-time linear system:

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

- Controllability is a 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, 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 controlled.

<u>Controllability Criterion</u>

Controllability Matrix P:

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

recall:

	m	number of inputs
	p	number of outputs
	n	number of states

$B, AB, A^2B, \cdots$ each have dimension:

so the dimension of P is:

SISO Case: *b* is a column matrix

P is an (*n* x *n*) square matrix.

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

Controllability Examples

1) $G(s) = \dfrac{1}{s^3 + a_2 s^2 + a_1 s + a_0}$

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

Clearly, $|P| = 0$ so the system is NOT completely state controllable. Why?

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

The states do not depend on *u*, so the system is uncontrollable.

2.4.2 Observability

The continuous-time linear system:

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

- Observability is a 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. If a system is completely observable, we can design an observer to support the state-space controller.

Observability Criterion

Observability Matrix Q:

If $rank(Q) = n$, the system is completely observable.

recall:

	m	number of inputs
	p	number of outputs
	n	number of states

$C, CA, CA^2, \cdots$ each have dimension:

so the dimension of Q is:

SISO Case: c is a row matrix

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

Observability Examples

1) $G(s) = \dfrac{1}{s^3 + a_2 s^2 + a_1 s + a_0}$

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

Clearly, $|Q| = 0$ so the system is NOT completely observable. Why?

$$y = x_1 + x_2$$

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 and observer with closed-loop poles as we specify.

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

2.4.3 MATLAB for Controllability and Observability

1) $A=\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -8 & -7 & -2 \end{bmatrix}$ $b=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ $c=\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$

`ctrb(A,b)` calculates: $P=\begin{bmatrix} b & Ab & A^2b \end{bmatrix}=\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & -2 \\ 1 & -2 & -3 \end{bmatrix}$

`rank(ctrb(A,b)) = 3` or **`det(ctrb(A,b)) = -1`**

The rank of P is full, or $|P| \neq 0$, so this system is completely state controllable.

`obsv(A,c)` calculates: $Q=\begin{bmatrix} c \\ cA \\ cA^2 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

`rank(obsv(A,c)) = 3` or **`det(obsv(A,c)) = 1`**

The rank of Q is full, or $|Q| \neq 0$, so this system is completely observable.

2) $A=\begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix}$ $b=\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ $c=\begin{bmatrix} 1 & 1 \end{bmatrix}$

`ctrb(A,b)` calculates: $P=\begin{bmatrix} b & Ab \end{bmatrix}=\begin{bmatrix} 1 & 2 \\ -1 & -2 \end{bmatrix}$

`rank(ctrb(A,b)) = 1` or **`det(ctrb(A,b)) = 0`**

The rank of P is not full, that is $|P| = 0$, so this system is not completely state controllable.

`obsv(A,c)` calculates: $Q=\begin{bmatrix} c \\ cA \end{bmatrix}=\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

`rank(obsv(A,c)) = 1` or **`det(obsv(A,c)) = 0`**

The rank of Q is not full, that is $|Q| = 0$, so this system is not completely observable.

2.5 Similarity Transformations and Canonical Realizations

Realization: one possible state-space description for a given system, out of infinite valid possibilities.
Canonical Realization: various simple, structured, standard forms for state-space description.

Textbook locations for Similarity Transformations and Canonical Realizations

Our coverage of Similarity Transformations and Canonical Forms (Controller Canonical Form, Observer Canonical Form, Diagonal Canonical Form) is contiguous in the notes, but is all over the place (text, examples, MATLAB, and homework problems, 3 chapters) in the textbook:

Topic	**Textbook Section**
Similarity Transformations	2.5
Diagonal Canonical Form (DCF)	2.5
MATLAB for DCF	2.6
Continuing Examples (DCF)	2.7
Controller Canonical Form (CCF)	3.3
MATLAB for CCF	3.5
Continuing Examples (CCF)	3.6
Observer Canonical Form (OCF)	4.4
MATLAB for OCF	4.6
Continuing Examples (OCF)	4.7

2.5.1 Similarity Transformations

Linear state vector coordinate transformation:

Where	X	original state vector	nx1
	Z	new state vector	nx1
	T	non-singular transformation matrix	nxn

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

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:

Same characteristic polynomial and eigenvalues.

2.5.2 Controllable Canonical Form (CCF)

CCF is our familiar form so far. Let:

$$\Delta(s) = s^n + a_{n-1}s^{n-1} + \cdots + a_2 s^2 + a_1 s + a_0$$

represent the characteristic polynomial. a_n must be 1, i.e. the polynomial must be normalized. Any state-space realization *A, B, C, D* can be transformed to **CCF** by:

$P = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$ is the controllability matrix (*n* x *nm*)

M =

So this only works for single input *m*=1; then *T* is (*n* x *n*)

CCF:

$\overline{C} = CT$ and $\overline{D} = D$ have no particular form.

CCF Example

Given the following system, calculate the *Controllable Canonical Form* (**CCF**).

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

CCF transformation matrix T:

Result:

$$\left[\overline{A}\right]=T^{-1}AT=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -3 & -1 & -3\end{bmatrix} \quad \left[\overline{B}\right]=T^{-1}B=\begin{bmatrix}0\\0\\1\end{bmatrix} \quad \left[\overline{C}\right]=CT=\begin{bmatrix}3 & 1 & 2\end{bmatrix} \quad \left[\overline{D}\right]=[D]=[0]$$

$$eig(A)=eig\left(\overline{A}\right)=roots\left(\begin{bmatrix}1 & 3 & 1 & 3\end{bmatrix}\right)=-3,\pm i$$

If you start with **CCF**, $T=PM=I$.

2.5.3 Observable Canonical Form (OCF)

OCF is the companion form to **CCF**. Let $\Delta(s)=s^n+a_{n-1}s^{n-1}+\cdots+a_2s^2+a_1s+a_0$ again represent the characteristic polynomial. Any state-space realization *A*, *B*, *C*, *D* can be transformed to **OCF** by:

$$Q=\begin{bmatrix}C\\ CA\\ CA^2\\ \vdots\\ CA^{n-1}\end{bmatrix}$$ is the observability matrix (*np* x *n*)

$$M=\begin{bmatrix}a_1 & a_2 & \cdots & a_{n-1} & 1\\ a_2 & a_3 & \cdots & 1 & 0\\ \vdots & \vdots & \iddots & \vdots & \vdots\\ a_{n-1} & 1 & \cdots & 0 & 0\\ 1 & 0 & \cdots & 0 & 0\end{bmatrix}$$ Same as in **CCF** case (*n* x *n*)

So this only works for single output $p=1$; then T is (n x n)

OCF:

$\overline{B} = T^{-1}B$ and $\overline{D} = D$ have no particular form.

CCF and **OCF** are companions, or duals:

OCF Example

Given the **CCF** example, calculate the *Observable Canonical Form* (**OCF**).

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

$|sI - A| = s^3 + 3s^2 + s + 3 = 0$ $\quad a_0$=3 $\quad a_1$=1 $\quad a_2$=3

$$[M] = \begin{bmatrix} 1 & 3 & 1 \\ 3 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \qquad [Q] = \qquad T = [MQ]^{-1} =$$

Result (check duality with **CCF**):

$$[\overline{A}] = T^{-1}AT = \begin{bmatrix} 0 & 0 & -3 \\ 1 & 0 & -1 \\ 0 & 1 & -3 \end{bmatrix} \quad [\overline{B}] = T^{-1}B = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \qquad [\overline{C}] = CT = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \qquad [\overline{D}] = [D] = [0]$$

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

If you start with **OCF**, $T = [MQ]^{-1} = I$.

2.5.4 Diagonal Canonical Form (DCF)

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

where v_i is the eigenvector of A associated with eigenvalue λ_i.

DCF:

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

- solve *n* first-order ODEs independently
- In **DCF**, if all *B* elements are non-zero, the system is completely controllable
- In **DCF**, if all *C* elements are non-zero, the system is completely observable

DCF Example

Given the same **CCF** and **OCF** examples, calculate *Diagonal Canonical Form* (**DCF**).

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

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

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

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

$\overline{B}$ and $\overline{C}$ are fully populated (not 0). Therefore, this system is completely controllable and observable.

$$eig(A) = eig\left(\overline{A}\right) = roots\left(\begin{bmatrix} 1 & 3 & 1 & 3 \end{bmatrix}\right) = -3, \pm i$$

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

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

Same example, start with **CCF**:

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

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

$$\left[\overline{A}\right] = T^{-1}AT = \begin{bmatrix} i & 0 & 0 \\ 0 & -i & 0 \\ 0 & 0 & -3 \end{bmatrix} \quad \left[\overline{B}\right] = T^{-1}B = \begin{bmatrix} -0.05 - 0.15i \\ -0.05 + 0.15i \\ 0.1 \end{bmatrix} \quad \left[\overline{C}\right] = CT = \begin{bmatrix} 1+i & 1-i & 18 \end{bmatrix} \quad \left[\overline{D}\right] = \left[D\right] = \left[0\right]$$

Imaginary terms in results! MATLAB follows another algorithm for canonical diagonalization (**DCF**), allowing only real numbers.

2.5.5 MATLAB for Canonical Realizations

1. Observer Canonical Form (OCF)

```
[ao,bo,co,do,to] = canon(a,b,c,d,'companion');
```

- **ao,bo,co,do** are the resulting state-space matrices in OCF corresponding to the original **a,b,c,d** system given in any form.
- **'companion'** is the text switch for **OCF**.
- **to** is the **OCF** transformation matrix.

2. Diagonal Canonical Form (DCF)

```
[ad,bd,cd,dd,td] = canon(a,b,c,d,'modal');
```

- **ad,bd,cd,dd** are the resulting state-space matrices in **DCF** corresponding to the original **a,b,c,d** system given in any form.
- **'modal'** is the text switch for **DCF**.
- **td** is the **DCF** transformation matrix.

3. Controller Canonical Form (CCF)

There is no **CCF** text switch for the **canon** function. Instead, one can use `'companion'` to find **OCF** and then apply the duality rules to determine **CCF**.

MATLAB `canon` Examples

DCF
Previous example, using **`canon`** with **`'modal'`**:

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

```
[ad,bd,cd,dd,td] = canon(a,b,c,d,'modal');
```

The **`'modal'`** option calculates the **DCF** canonical realization where the first-order differential equations are decoupled.

$$\text{ad}=\begin{bmatrix}-3 & 0 & 0\\ 0 & 0 & 1\\ 0 & -1 & 0\end{bmatrix} \qquad \text{bd}=\begin{bmatrix}2.12\\-0.49\\-1.16\end{bmatrix} \qquad \text{cd}=\begin{bmatrix}0.85 & -0.35 & -0.02\end{bmatrix} \qquad \text{dd}=0$$

OCF
Previous example, using **`canon`** with **`'companion'`**:

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

```
[ao,bo,co,do,to] = canon(a,b,c,d,'companion');
```

The **`'companion'`** option calculates the **OCF** realization, which is the dual realization for **CCF**.

$$\text{ao}=\begin{bmatrix}0 & 0 & -3\\ 1 & 0 & -1\\ 0 & 1 & -3\end{bmatrix} \qquad \text{bo}=\begin{bmatrix}1\\0\\0\end{bmatrix} \qquad \text{co}=\begin{bmatrix}2 & -5 & 16\end{bmatrix} \qquad \text{do}=0$$

$$T=\begin{bmatrix}1/3 & 4/3 & 2/3\\ 1 & 1/3 & -1\\ 1/3 & 0 & -1/3\end{bmatrix}$$

This **OCF** violates the duality rule state earlier since C_{OCF} should equal B_{CCF}^{T}; however, both are correct realizations since they both yield the same transfer function.

Also see the MATLAB **`ss2ss`** function.

canon function: return to controllability & observability examples

1) $a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -8 & -4 & -2 \end{bmatrix}$ $b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ $c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ $d = [0]$

```
[ad,bd,cd,dd,td] = canon(a,b,c,d,'modal');
```

The **'modal'** option calculates the **DCF** canonical realization where the first-order differential equations are decoupled.

$\text{ad} = \begin{bmatrix} -2 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & -2 & 0 \end{bmatrix}$ $\text{bd} = \begin{bmatrix} -0.57 \\ -0.25 \\ -0.77 \end{bmatrix}$ $\text{cd} = \begin{bmatrix} -0.22 & -0.10 & 0.19 \end{bmatrix}$ $\text{dd} = 0$

There are no zeros in **bd**, so this system is completely state controllable.
There are no zeros in **cd**, so this system is completely observable.

eig(a) are distinct so **ad** is fully decoupled (note MATLAB disallows imaginary terms on the diagonal and in the transformation matrix):

$$\text{td} = \begin{bmatrix} -2.29 & 0 & -0.57 \\ -3.08 & -2.04 & -0.25 \\ 1 & -1.04 & -0.77 \end{bmatrix}$$

transformation matrix, for the state transformation $\{Z\} = [\text{td}]\{X\}$.

2) $a = \begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ $c = \begin{bmatrix} 1 & 1 \end{bmatrix}$ $d = [0]$

```
[ad,bd,cd,dd,td] = canon(a,b,c,d,'modal');
```

The **'modal'** option calculates the **DCF** canonical realization where the first-order differential equations are decoupled.

$\text{ad} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ $\text{bd} = \begin{bmatrix} 0 \\ 1.414 \end{bmatrix}$ $\text{cd} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ $\text{dd} = 0$

There is a zero in **bd**, so this system is not completely state controllable.
There is a zero in **cd**, so this system is not completely observable.

1) $a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -8 & -4 & -2 \end{bmatrix}$ $b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ $c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ $d = [0]$

```
[ao,bo,co,do,to] = canon(a,b,c,d,'companion');
```

The **'companion'** option calculates the **OCF** realization, which is the dual realization for **CCF**.

$$\text{ao} = \begin{bmatrix} 0 & 0 & -8 \\ 1 & 0 & -4 \\ 0 & 1 & -2 \end{bmatrix} \quad \text{bo} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{co} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad \text{do} = 0$$

$$\text{to} = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

2) $a = \begin{bmatrix} 2 & 0 \\ -1 & 1 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ $c = \begin{bmatrix} 1 & 1 \end{bmatrix}$ $d = [0]$

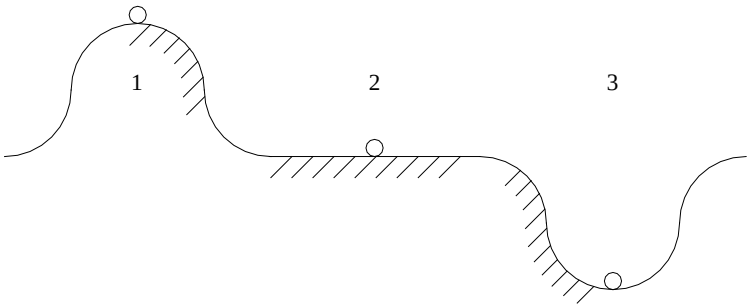
```
[ao,bo,co,do,to] = canon(a,b,c,d,'companion');
```

Error: system must be controllable from first input.

The OCF transformation fails since the system is not fully controllable (the transformation matrix **to** is singular).

2.6 Stability

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 1. unstable equilibrium, 2. neutral equilibrium, or 3. stable equilibrium:



2.6.1 Eigenvalue Test for System Stability

For a strictly stable system, the real part of all poles must be negative. This is because the e^{at} component of the solution will blow up for positive a (the real part of the poles). Recall that poles are the eigenvalues of the system dynamics matrix A.

The characteristic equation and poles (eigenvalues of A) are given below (this assumes a SISO second-order system – the\is stability test also applies to higher order systems):

$$\left|sI - A\right| = 0 \qquad s_{1,2} = a \pm ib$$

The form of the transient solution is:

$$y(t) = Ce^{at}\cos(bt + \phi)$$

If all	$a < 0$	stable	
If any	$a = 0$	marginally stable	(assuming all other $a \le 0$)
If any	$a > 0$	unstable	(regardless of other a)

Re-Im pole map:

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.

2.6.2 Stability Analysis Based on Energy and Phase Plots

Stability analysis based on system energy applies to linear/non-linear and constant/time varying systems. We can determine stability without solving $\dot{X} = AX + BU$ (or non-linear system equations). E is the total system energy and an equilibrium point is $X=\{0\}$:

If	then $X \to \{0\}$ and the system is *stable*.
If	then the system is *marginally stable*.
If	then the system is *unstable* because something is continuously adding energy.

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

If the system is stable.

This is always true for positive damping.

Stability analysis via phase plots

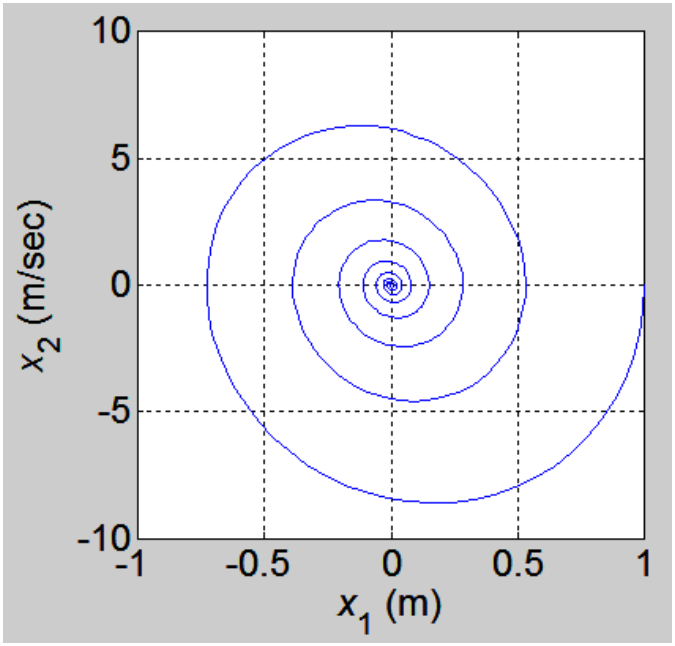
Plot velocity x_2 vs. displacement x_1:

- Increasing, unchanging, or decreasing energy
- Stable if orbit converges to a point (constant x_1, $x_2=0$)
- Show examples: +, 0, – damping
- Initial conditions

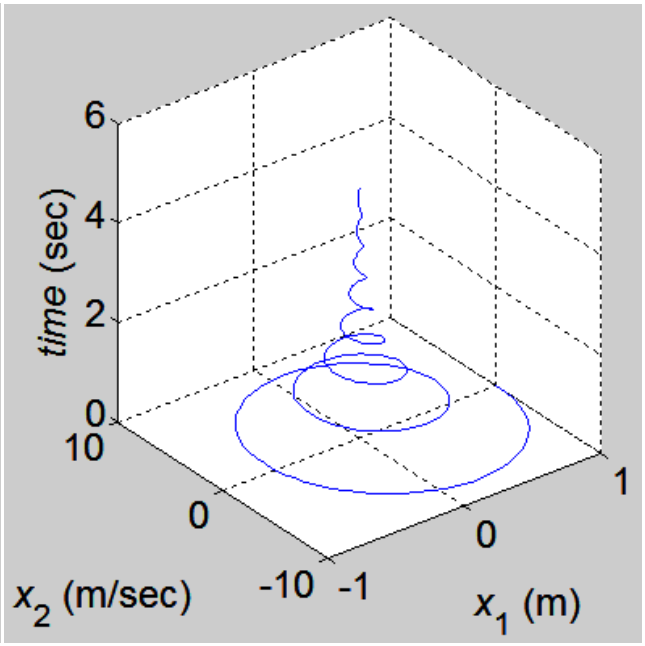
1. Positive damping system $s_{1,2} = -1 \pm 9.95i$ $G(s) = \dfrac{1}{s^2 + 2s + 100}$

$$[A] = \begin{bmatrix} 0 & 1 \\ -100 & -2 \end{bmatrix} \qquad [B] = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad [C] = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad [D] = [0]$$

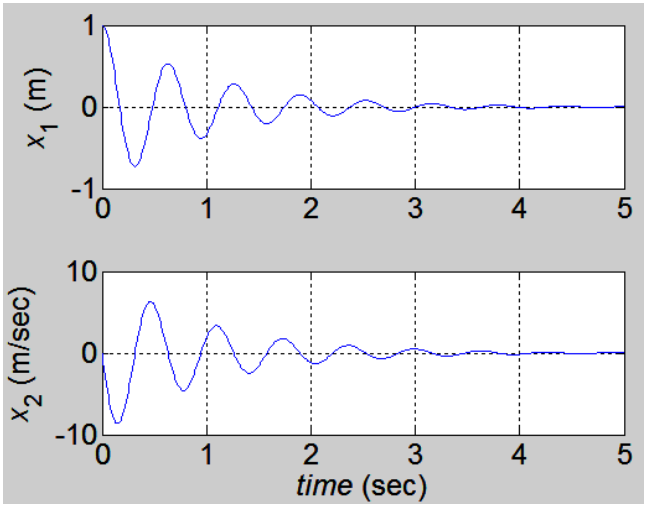
Zero input response with initial conditions $\{X_0\} = \{1 \quad 0\}^T$



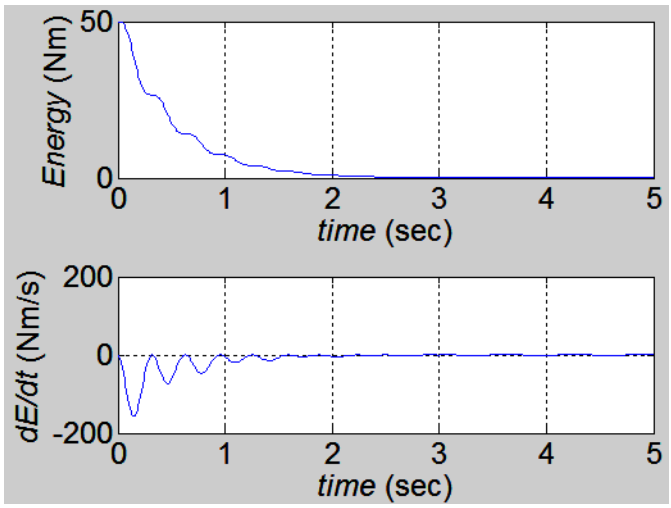
Phase Plot, + damping



3D Phase Plot, + damping



Zero Input State Responses

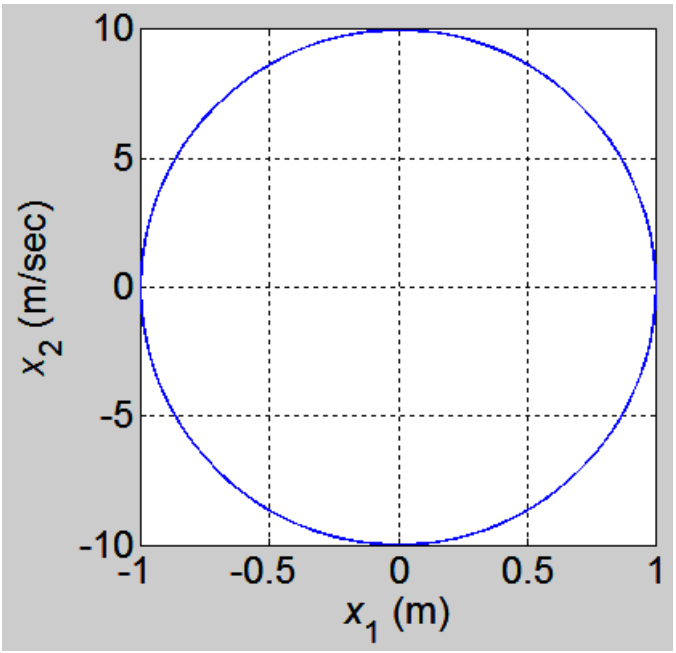


System Energy and Derivative

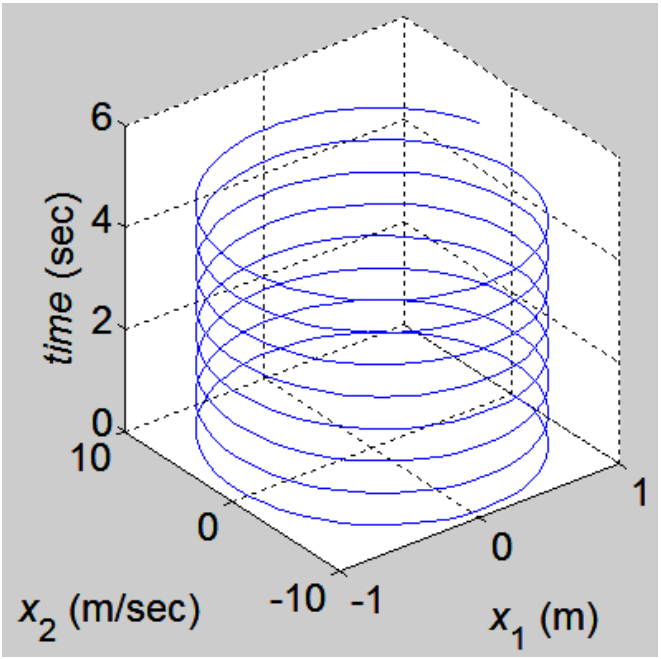
2. Zero damping system $s_{1,2} = 0 \pm 10i$ $G(s) = \dfrac{1}{s^2 + 100}$

$$[A] = \begin{bmatrix} 0 & 1 \\ -100 & 0 \end{bmatrix} \quad [B] = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad [C] = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad [D] = [0]$$

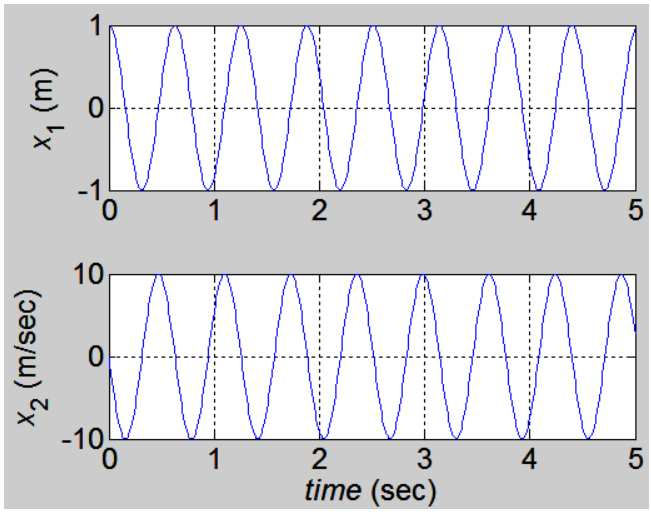
Zero input response with initial conditions $\{X_0\} = \{1 \quad 0\}^T$



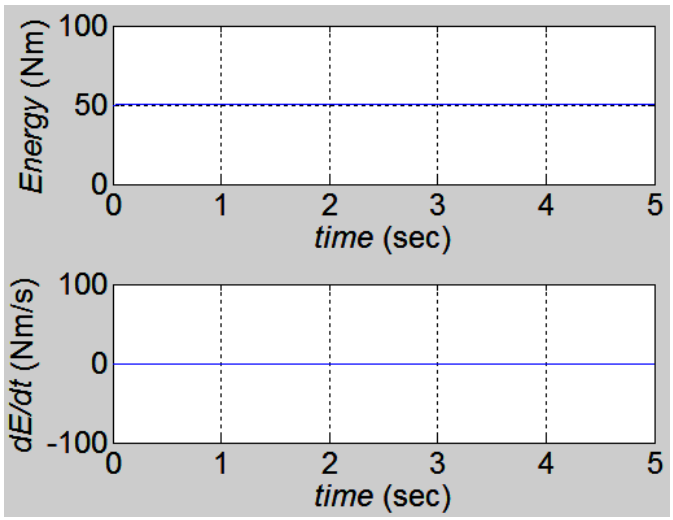
Phase Plot, 0 damping



3D Phase Plot, 0 damping



Zero Input State Responses



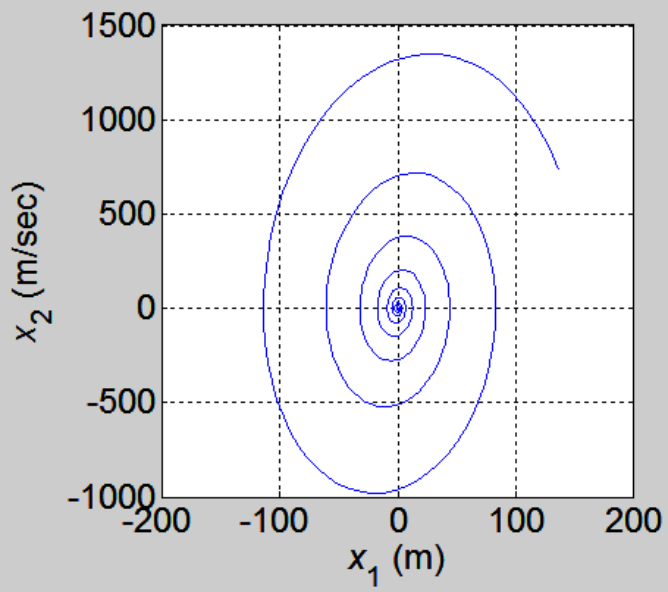
System Energy and Derivative

Note: $\omega_n = 10 = 2\pi f = \dfrac{2\pi}{T}$ or $T = \dfrac{\pi}{5} = 0.63$; this time period agrees with the plotted responses.

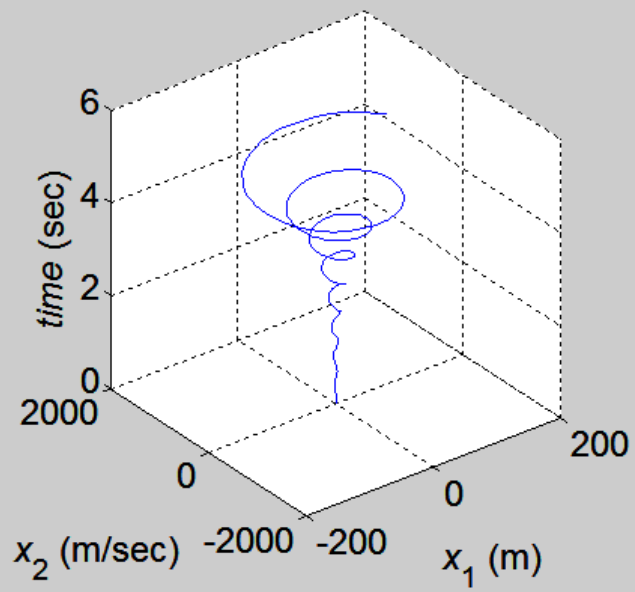
3. Negative damping system $s_{1,2} = +1 \pm 9.95i$ $G(s) = \dfrac{1}{s^2 - 2s + 100}$

$$[A] = \begin{bmatrix} 0 & 1 \\ -100 & 2 \end{bmatrix} \quad [B] = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad [C] = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad [D] = [0]$$

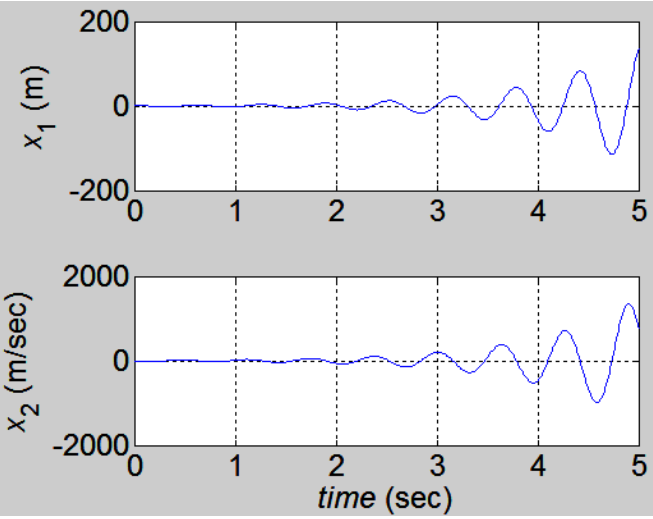
Zero input response with initial conditions $\{X_0\} = \{1 \quad 0\}^T$



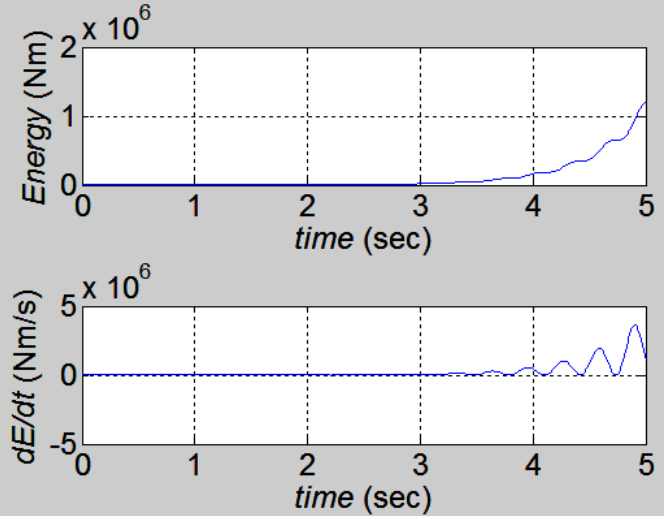
Phase Plot, – damping



3D Phase Plot, – damping



Zero Input State Responses



System Energy and Derivative

2.6.3 Lyapunov Stability Analysis

Definitions:

- Any time varying nonlinear system can be represented as:

$$\dot{X} = f(X,t)$$

- A state is an equilibrium state X_e if $f(X_e,t)=0$ for all t.

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

- We can always shift an equilibrium state X_e to zero by coordinate shifts: $f(0,t)=0$ for all t.

- Hyperspherical region of radius k about an equilibrium state X_e: $\|X - X_e\| \le k$

  using the Euclidean norm:

$$\|X - X_e\| = \sqrt{(x_1 - x_{1e})^2 + (x_2 - x_{2e})^2 + \cdots + (x_n - x_{ne})^2}$$

- Define two such spherical regions: $\|X - X_e\| \le \delta$ and $\|X - X_e\| \le \varepsilon$, with $\delta < \varepsilon$.

  Graphical representation for Lyapunov regions:

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

2. An equilibrium state X_e is said to be <u>*asymptotically stable*</u> if trajectories starting within δ converge to X_e without leaving the ε region as t increases indefinitely. This case is preferable to *stability I.S.L.*

3. An equilibrium state X_e is said to be <u>*asymptotically stable in the large*</u> if asymptotic stability holds for all possible initial states X_0. There must be only one equilibrium state in the whole state space.

4. An equilibrium state X_e is said to be <u>*unstable*</u> 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*)

Lyapunov Stability Method

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

$$X_e = 0 \qquad \text{or} \qquad AX_e = 0$$

Second Method of Lyapunov (1892, Russian):

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

$V(X)$ Lyapunov function: generalized energy function, not unique

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

Quadratic form $X^T PX$ scalar function, P is real and symmetric. This form is positive-definite if P is positive-definite. **Note**: P is NOT the controllability matrix. Quadratic form for $n = 2$:

$$X^T PX = \begin{Bmatrix} x_1 & x_2 \end{Bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} x_1 & x_2 \end{Bmatrix} \begin{Bmatrix} p_{11}x_1 + p_{12}x_2 \\ p_{12}x_1 + p_{22}x_2 \end{Bmatrix} = p_{11}x_1^2 + 2p_{12}x_1x_2 + p_{22}x_2^2$$

Positive-definite matrix: <u>*Sylvester's criterion*</u>:

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.

Lyapunov's Direct Method

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

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

$$V(X) = X^T PX$$

If A is constant, P is constant:

$$\dot{P} = 0 \text{ and using } \dot{X} = AX :$$

Lyapunov Equation

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

must be negative-definite. That is:

for some positive-definite *Q*. **Note**: *Q* is NOT the observability matrix. 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*. This is unfortunate since it is easy to find *Q* given *P*, but it may not work this way. Therefore we must solve:

Necessary and sufficient condition:

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 $X_e = 0$ is *asymptotically stable in the large*.

Lyapunov Stability Analysis Example

Determine the stability condition for:

$$A = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix}$$

Solve the following equation for symmetric *P* and determine its definiteness. Let *Q* = *I* (a simple positive-definite matrix).

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

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

due to symmetry, use either 2,1 or 1,2 (they are the same equation).

Actually, these linear equations are decoupled:

Check Sylvester's criterion for positive definiteness:

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

Note:

`eig(A)` = [–1, –2]. All real parts of the poles are strictly negative so this simple pole analysis agrees with the above Lyapunov result (the system is strictly stable).

MATLAB Solution of Lyapunov Equation

`X = lyap(A,C)` solves $AX + XA^T = -C$.

Then check resulting matrix X for positive definiteness. Note the MATLAB convention is different:

$A^T P + PA = -Q$	standard convention
$AX + XA^T = -C$	MATLAB convention

- X is not the state vector, it's the Lyapunov P matrix!!
- C is not the output matrix, it's the Lyapunov positive definite matrix Q!!
- Note we have to put in A^T (**`A'`** in MATLAB parlance).

Example 1

$$A=\begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} \qquad Q=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \texttt{P = lyap(A',Q)}$$

$$P=\begin{bmatrix} 1.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix}$$

P is positive definite (both principal minors are positive), so the system is *asymptotically stable in the large*, which agrees with our earlier results for this example.

There is no MATLAB function for Sylvester's criterion so we must test it ourselves using MATLAB.

Example 2

$$A=\begin{bmatrix} 0 & 1 \\ -75 & -0.3 \end{bmatrix} \qquad Q=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \texttt{P = lyap(A',Q)}$$

$$P=\begin{bmatrix} 126.7 & 0.007 \\ 0.007 & 1.69 \end{bmatrix}$$

P is positive definite (both principal minors are positive), so the system is Asymptotically Stable.

Example 3

$$A=\begin{bmatrix} 0 & 1 \\ -75 & 0 \end{bmatrix} \qquad Q=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \texttt{P = lyap(A',Q)}$$

Error: `Solution is not unique` – MATLAB cannot solve for *P*. This case is *marginally stable* according to simple eigenvalue analysis.

Example 4

$$A=\begin{bmatrix} 0 & 1 \\ -75 & 0.3 \end{bmatrix} \qquad Q=\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \texttt{P = lyap(A',Q)}$$

$$P=\begin{bmatrix} -126.7 & 0.007 \\ 0.007 & -1.69 \end{bmatrix}$$

P is not positive definite (obviously the first principal minor is negative), therefore the system is *unstable*.

3. Full-State Feedback Controller and Observer Design

3.1 Shaping Dynamic Response

Force system output to perform as desired using feedback control. Design of controller: place poles of closed-loop system in order to:

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

Generic Second-Order System Details Why?

- many real systems modeled with second-order ODEs
- design controller so higher-order system mimics desired second-order system

Linear 1-dof *m-c-k* mechanical translational system with *y*(*t*) output (displacement) and *f*(*t*) input (force)

where: (*rad*/*s*) is the *natural frequency*

is the *dimensionless damping ratio*

Generic second-order system transfer function:

Characteristic polynomial:

Roots of the denominator (poles):

Five damping cases based on ξ (the first four cases we studied earlier: see the poles and transient response example under the Solution of State-Space Equations section):

Damping	Unit step response
$\xi > 1$	**overdamped**: real distinct negative roots s_1, s_2, sluggish response
$\xi = 1$	**critically damped**: real repeated negative roots s_1, s_1, fastest response without overshoot
$0 < \xi < 1$	**underdamped**: complex conjugate roots with negative real part $s_{1,2} = a \pm ib$, overshoot and oscillation, damps out
$\xi = 0$	**undamped**: complex conjugate roots with zero real part $s_{1,2} = \pm ib$, simple harmonic motion oscillation, theoretically never damps out
$\xi < 0$	**unstable**: positive real part, exponential blows up

Second-Order System Performance Specifications

underdamped case:

where (*rad*/*s*) is the *damped natural frequency*

To obtain a steady-state response of 1, $u(t)$ must be a unit step function when using the generic second-order transfer function – *final value theorem* to calculate y_{SS}:

Unit step response solution for the generic second-order system:

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

Steady-state error:

$$e_{SS} = \frac{y_{SSactual} - y_{SSdesired}}{y_{SSdesired}} \times 100\%$$

So, theoretically, there is 0% steady-state error in this case (see plot later). Plot of response with rise time, peak time, % overshoot, and settling time:

Without derivation (see Dorf and Bishop):

1) Rise Time (10-90%) $t_R \cong \frac{2.16\xi + 0.60}{\omega_n} \; (0.3 \le \xi \le 0.8)$

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

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

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

Control swiftness of response: change rise and peak time
Control error of response: change % overshoot, settling time
These are competing requirements; see examples below.

Second-Order System Performance Specifications Example 1

Standard translational mechanical system (m = 1 kg, c = 0.5 Ns/m, k = 10 N/m).

natural frequency $\omega_n = \sqrt{10} = 3.16$

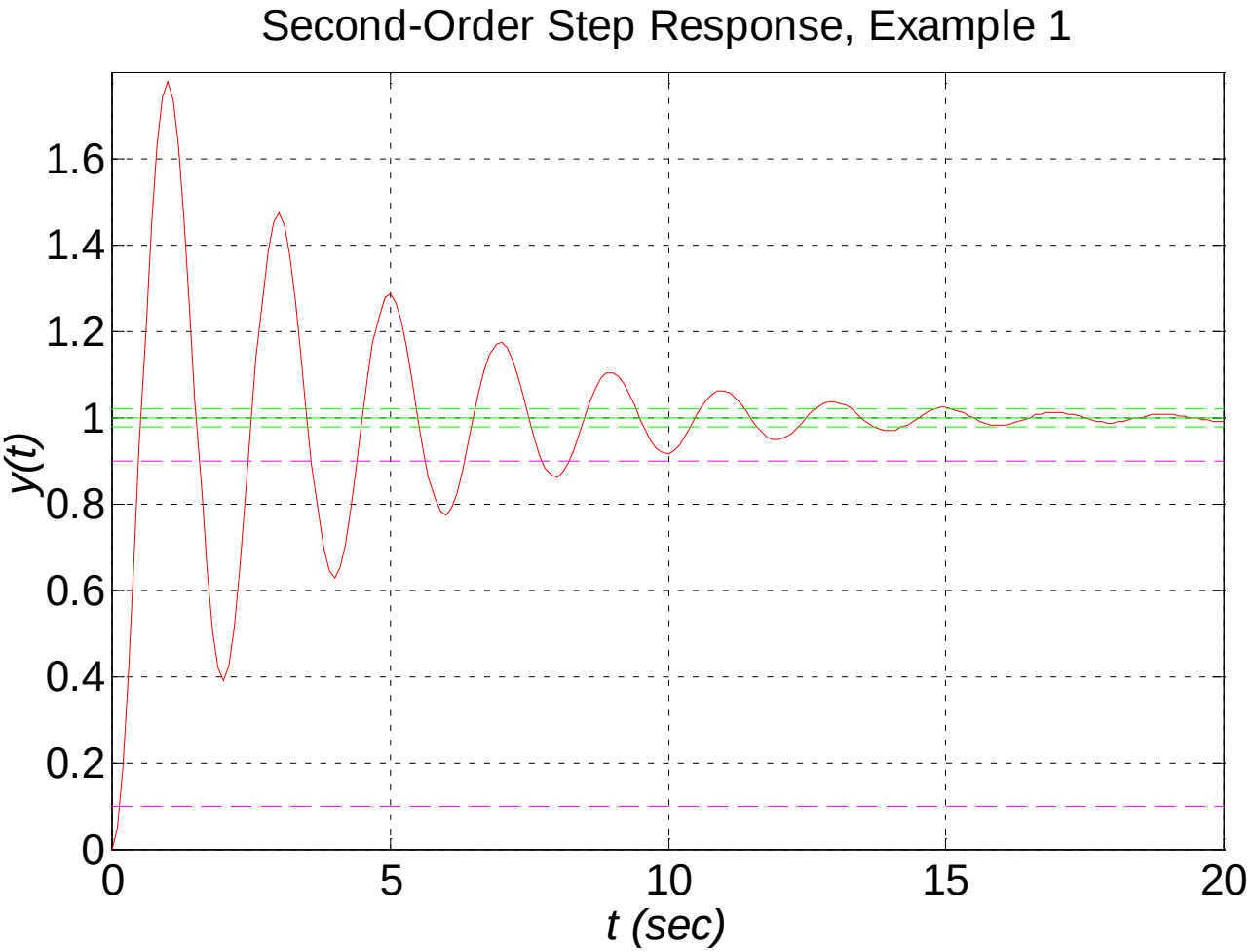
damping ratio $\xi = 0.08$

Generic second-order system transfer function: $G(s) = \dfrac{10}{s^2 + 0.5s + 10}$

Characteristic equation: $s^2 + 0.5s + 10 = 0$

Roots (poles): $s_{1,2} = -0.25 \pm 3.15i$

Unit step response solution: $y(t) = 1 - 1.003e^{-0.25t}\sin(3.15t + 85.5^\circ)$



Example 1 Performance Specifications

t_R = 0.25 s (bad approximation $\xi < 0.3$ – from data t_R = 0.34 s)

t_P = 1.00 s

PO = 77.90%

t_S = 16.00 s

Second-Order System Performance Specifications Example 2

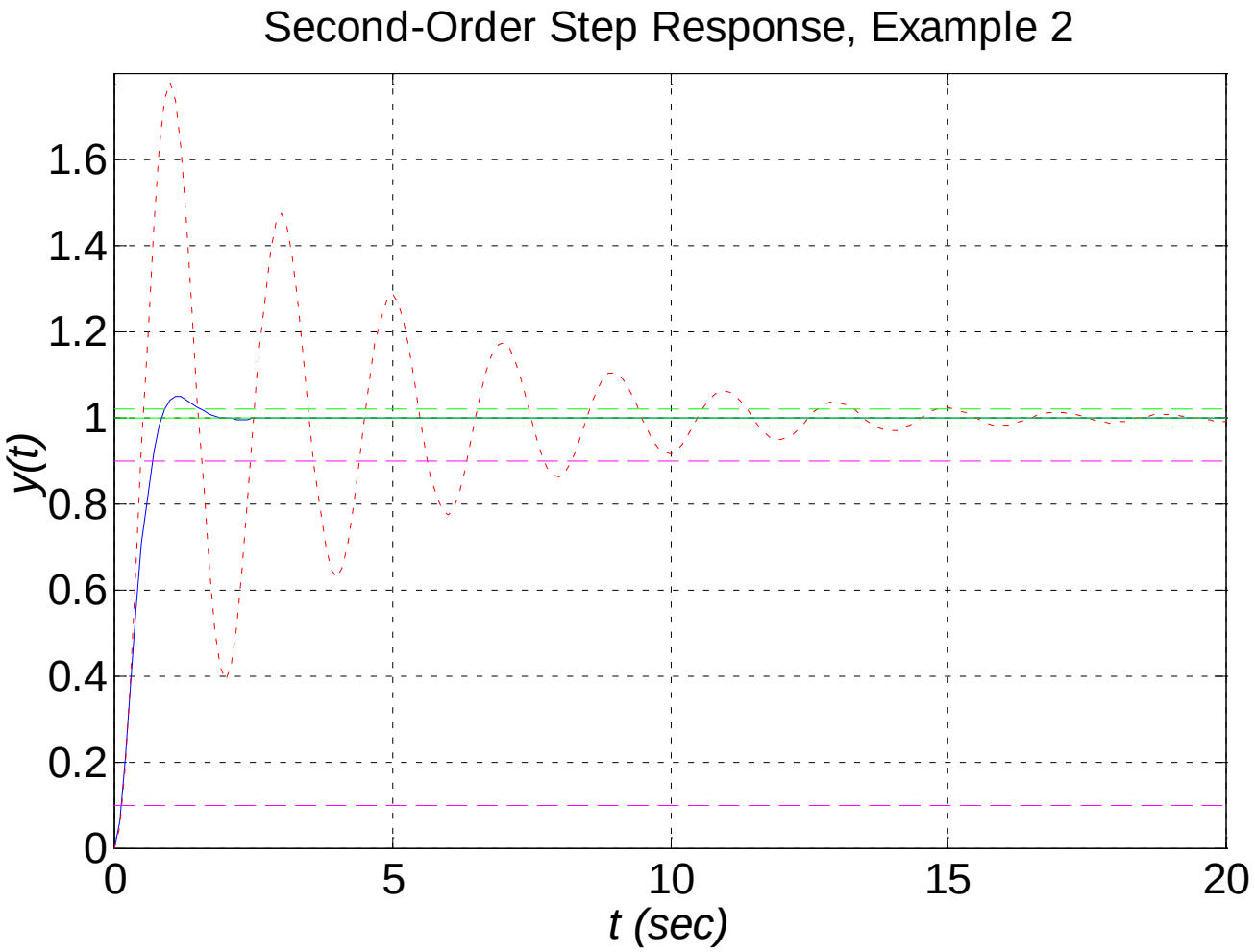
The second-order system in Example 1 is highly *underdamped*; ξ is close to 0, the percent overshoot is large, and the settling time is relatively high. This is typical of real-world systems such as flexible space robots (both joint motion and Cartesian motion, on all degrees-of-freedom). Now let us create another example, to specify more desirable behavior for such a system. Let us specify t_S = 1.5 *sec* and *PO* = 5%; the result will still be *underdamped* but not severely so.

From $PO = 5 = 100e^{\left(\frac{-\xi\pi}{\sqrt{1-\xi^2}}\right)}$, calculate ξ: $L_{PO} = \ln\left(\frac{PO}{100}\right) = \frac{-\xi\pi}{\sqrt{1-\xi^2}}$ $\xi = \sqrt{\frac{L_{PO}^2}{\pi^2 + L_{PO}^2}} = 0.69$

Then from $t_S = 1.5 \cong \frac{4}{\xi\omega_n}$, calculate the approximate ω_n: $\omega_n \cong \frac{4}{\xi t_S} = 3.86$

Second-Order System Example 2

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} = \frac{14.93}{s^2 + 5.33s + 14.93} \qquad \omega_n = 3.86 \qquad \xi = 0.69 \qquad s_{1,2} = -2.67 \pm 2.80i$$



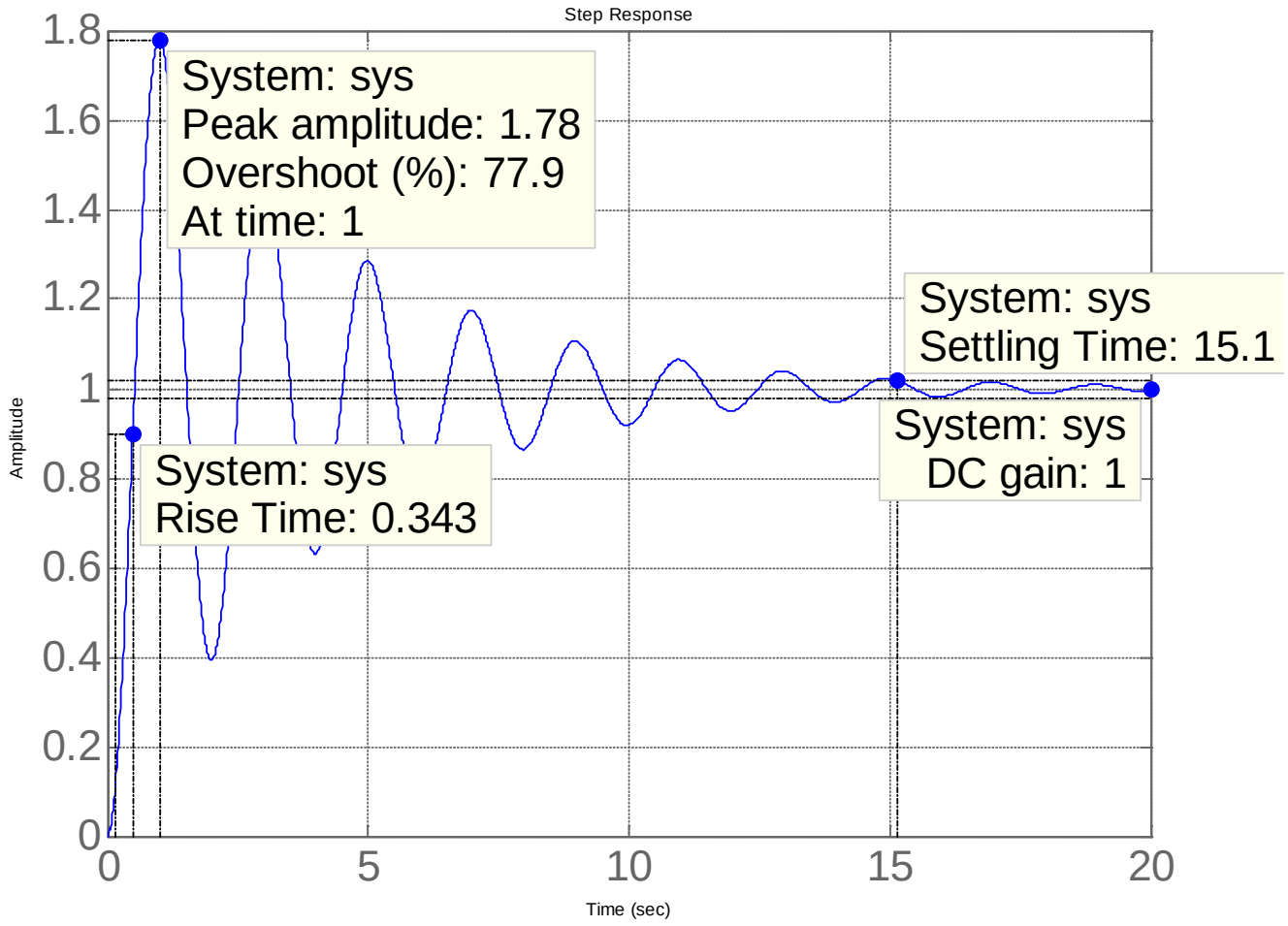
Example 2		Example 1	
t_R	= 0.54 *s*	t_R	= 0.34 *s*
t_P	= 1.12 *s*	t_P	= 1.00 *s*
PO	= 5.00%	*PO*	= 77.90%
t_S	= 1.50 *s*	t_S	= 16.00 *s*

Second-Order System Performance Specifications: MATLAB

```
A = [0 1;-10 -0.5];   B = [0; 1];   C = [1 0];   D = 0;
t = [0:0.01:20];
BobSys = ss(A,B,C,D)
step(BobSys,t);  set(gca,'FontSize',18);  grid;
```

Try this m-code (for Example 1); then right-click in figure window to add the performance specifications (Characteristics) automatically; then compare with the computed results:

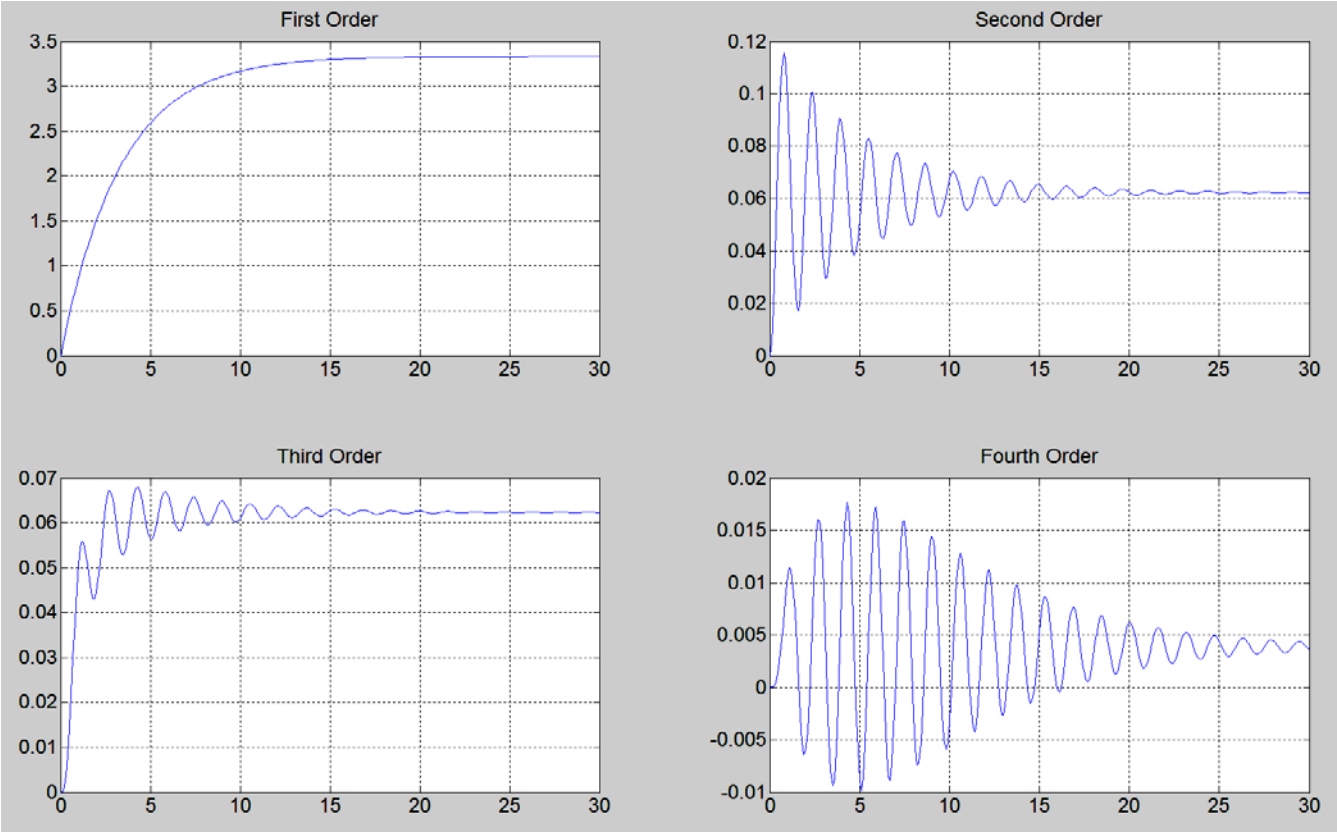
Characteristics ▷ Peak Response
Settling Time
Rise Time
Steady State



Computed Performance Specifications from before:

t_R = 0.25 *s* (bad approximation $\xi < 0.3$ – from data t_R = 0.34 *s*)
t_P = 1.00 *s*
PO = 77.90%
t_S = 16.00 *s*

First-, Second-, Third-, and Fourth-Order Systems: Step Responses



Poles:

first $s_1 = -0.3$	second $s_{1,2} = -0.2 \pm 4i$
third $s_1 = -1$, $s_{2,3} = -0.2 \pm 4i$	fourth $s_{1,2} = -0.2 \pm 4i$, $s_{3,4} = -0.22 \pm 4i$

Now, we have focused on looking at the generic second-order system unit step response, plus performance specifications, in order to specify good behavior for our controller to mimic. This good behavior is expressed in terms of some new desired system poles (different than the original open-loop poles). So we have two dominant poles from $s^2 + 2\xi\omega_n s + \omega_n^2 = 0$, for the underdamped case, $s_{1,2} = -\xi\omega_n \pm i\omega_d$.

What if your original open-loop system is n^{th}-order, and not second-order? Then you need to include n–2 additional poles, in addition to the original, dominant second-order poles, according to the following rule of thumb.

Additional poles: choose about 10x higher than the real part of the dominant second-order poles, keep them negative for stability and real-only. The 10x higher poles will cause these modes to occur much faster than the dominant poles, so their effect is hardly noticed in the overall responses, so the dominant second-order poles still dominate.

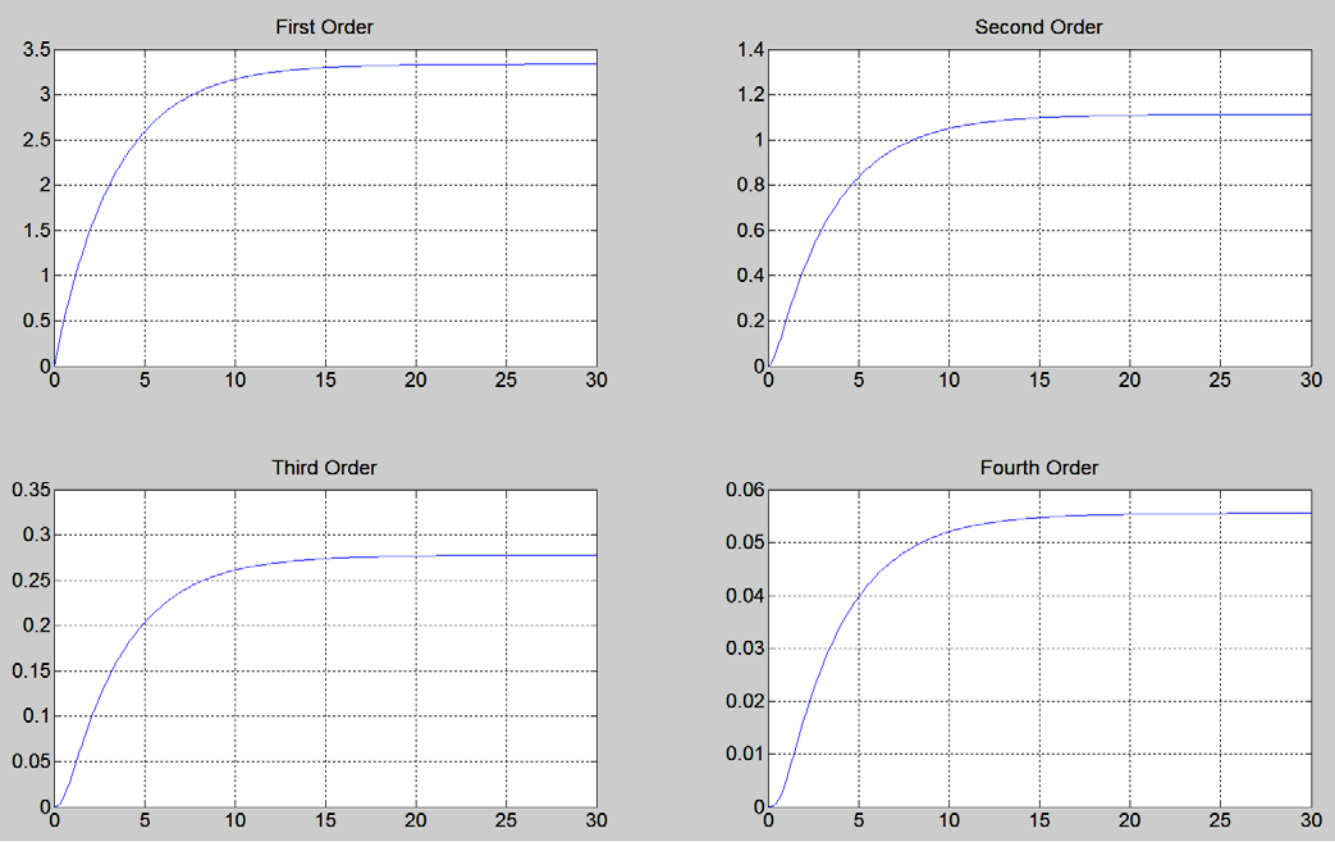
Example: from Example 2 earlier, $s_{1,2} = -2.67 \pm 2.80i$. If your system is sixth-order, include 6–2=4 additional, non-dominant poles:

$$s_{3,4,5,6} = -27, -28, -29, -30$$

These are the six poles to specify to the controller design process, to basically achieve the desired dominant second-order behavior. When you need the new, desired sixth-order polynomial, you must multiply all the poles out (see MATLAB function **conv**):

$$\Delta_6(s) = \left(s^2 + 2\xi\omega_n s + \omega_n^2\right)(s+s_3)(s+s_4)(s+s_5)(s+s_6)$$

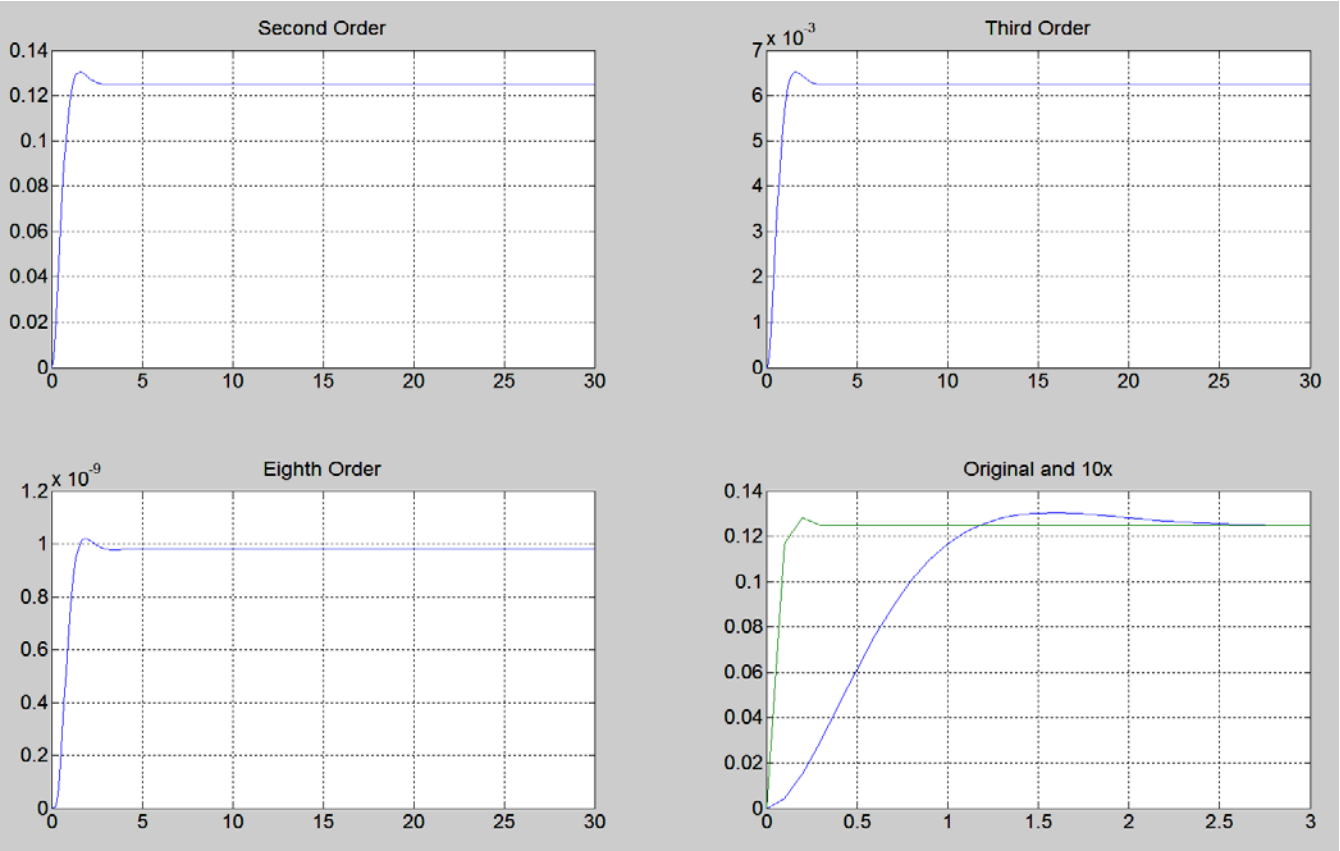
Mimic First-Order System with Higher Order Systems: Step Responses



Poles:

first $s_1 = -0.3$	second $s_{1,2} = -0.3, -3$
third $s_{1,2,3} = -0.3, -3, -4$	fourth $s_{1,2,3,4} = -0.3, -3, -4, -5$

Mimic Second-Order System with Higher Order Systems: Step Responses



Poles:

second $s_{1,2} = -2 \pm 2i$	third $s_{1,2,3} = -2 \pm 2i, -20$
eighth $s_{1,2,3,4} = -2 \pm 2i, -20, -21$ $s_{5,6,7,8} = -22, -23, -24, -25$	original $s_{1,2} = -2 \pm 2i$ 10x $s_{1,2} = -20 \pm 20i$

***ITAE* Performance Index** Alternate pole shaping method

Control swiftness of response: change rise and peak time
Control error of response: change % overshoot, settling time

These are competing requirements. How to choose desired poles for closed-loop system under control? Try performance indices – many cases have been solved to tell you the optimal poles given various performance measures (indices). Here we will only consider one: the *integral of time multiplied by the absolute error* (*ITAE*).

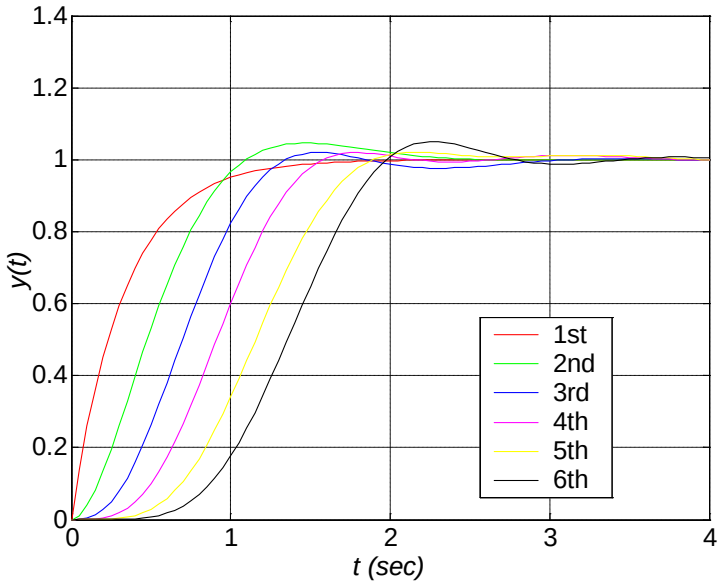
$$ITAE = \int_0^\infty t\left|e(t)\right|dt$$

Minimize *ITAE* to simultaneously optimize competing requirements. Minimum *ITAE* means short time and small error at once. For first- through sixth-order systems, the following characteristic polynomials minimize *ITAE* (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*.

n	**Optimal Characteristic Polynomials**
1	$s+\omega_n$
2	$s^2+1.4\omega_n s+\omega_n^2$
3	$s^3+1.75\omega_n s^2+2.15\omega_n^2 s+\omega_n^3$
4	$s^4+2.1\omega_n s^3+3.4\omega_n^2 s^2+2.7\omega_n^3 s+\omega_n^4$
5	$s^5+2.8\omega_n s^4+5.0\omega_n^2 s^3+5.5\omega_n^3 s^2+3.4\omega_n^4 s+\omega_n^5$
6	$s^6+3.25\omega_n s^5+6.6\omega_n^2 s^4+8.6\omega_n^3 s^3+7.45\omega_n^4 s^2+3.95\omega_n^5 s+\omega_n^6$

The engineer must set the value for ω_n to suit the system at hand.

Note for all cases (except the first order system) some overshoot is required to optimize *ITAE*. Simulate with MATLAB to see this (using ω_n = 3 rad/s for all plots below):



ITAE Desired Unit Step Responses

3.2 Linear Full State-Feedback Controller Design

3.2.1 Background

Original open-loop system:

Feedback Control Law:

Draw open-loop diagram first. Assume the output $Y(t)$ is not performing as desired (underdamped, unstable, etc.). Add full state $X(t)$ feedback.

Diagram:

$r(t)$	reference input	same units and dimensions (m x 1) as $U(t)$
K	constant state feedback gain matrix	dimensions (m x n)

Feedback control law for SISO:

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

Derive Closed-Loop State Dynamics Equation:

If the original open-loop system represented by *A*, *B* is *completely state-controllable*, a matrix *K* exists that can give arbitrary eigenvalues in the closed-loop system *A* – *BK*. That is, we can place the roots (system poles) of the characteristic equation shown below anywhere.

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

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

3.2.2 Decoupled CCF Solution

Controllable Canonical Form (**CCF**) SISO:

$A - BK =$

Eigenvalues of *A*-*BK* are the roots of the closed-loop characteristic equation:

Equate the desired closed-loop characteristic polynomial $\alpha(s)$ (known numbers, determined by the engineer in the dynamic shaping process) with $\alpha(s,k_i)$, the closed-loop characteristic polynomial that is a function of controller gain unknowns $k_i,\ i = 1, 2, \cdots, n$:

With **CCF**, solution for *K* is decoupled:

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

Example: Controller pole placement via feedback gain matrix *K*

Given the following third-order open-loop system:

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

The first step in controller design is to understand the characteristics of the as-given open-loop system:

Open-loop characteristic polynomial: $s^3+2s^2+15s+18=0$

Poles: **`eig(A) = roots([1 2 15 18]) =`** $-1.28, -0.36 \pm 3.73i$

Typical third-order lightly damped response:

This open-loop system is already stable. Let's design a closed-loop state feedback controller to change the poles to improve transient performance. In this example let us choose a desirable dominant second-order system. We want the resulting closed-loop system to mimic a generic second-order system with **6% overshoot** and **3 sec settling time**. The associated dimensionless damping ratio and natural frequency are calculated:

$$\xi = 0.67 \qquad \omega_n = 2$$

The associated closed-loop dominant desired poles are:

$$s_{1,2} = -\xi\omega_n \pm i\omega_d = -1.33 \pm 1.49i$$

The original open-loop system is third-order, so we need to choose a third desired pole. Make it negative, real, and 10 times higher than the real part of the dominant second-order poles:

$$s_3 = -13.33$$

So the desired characteristic polynomial is (use MATLAB **`conv`**):

$$\alpha(s) = \left(s^2+2.68s+4\right)\left(s+13.33\right) = s^3+16s^2+39.55s+53.26$$

This will lead to better transient performance:

Controller design (determine unknowns $k_i,\ i = 1, 2, \cdots, n$)

$A - BK =$

$A - BK =$

$\left| sI - \left(A - BK \right) \right| = \alpha(s, k_i) =$

The top characteristic polynomial $\alpha(s,k_i)$ is a function of the unknown controller gain matrix K, the bottom is the numerical desired closed-loop characteristic polynomial $\alpha(s)$. Equate these two polynomials to solve for K (with **CCF**, the solution for K is decoupled):

Units?

Units of k, c, m in a translational mechanical system.

Again, for a given open-loop SISO system in **CCF**, there is a simple decoupled solution for the unknown full-state feedback gain matrix K:

$$k_i = \alpha_{i-1} - a_{i-1}$$

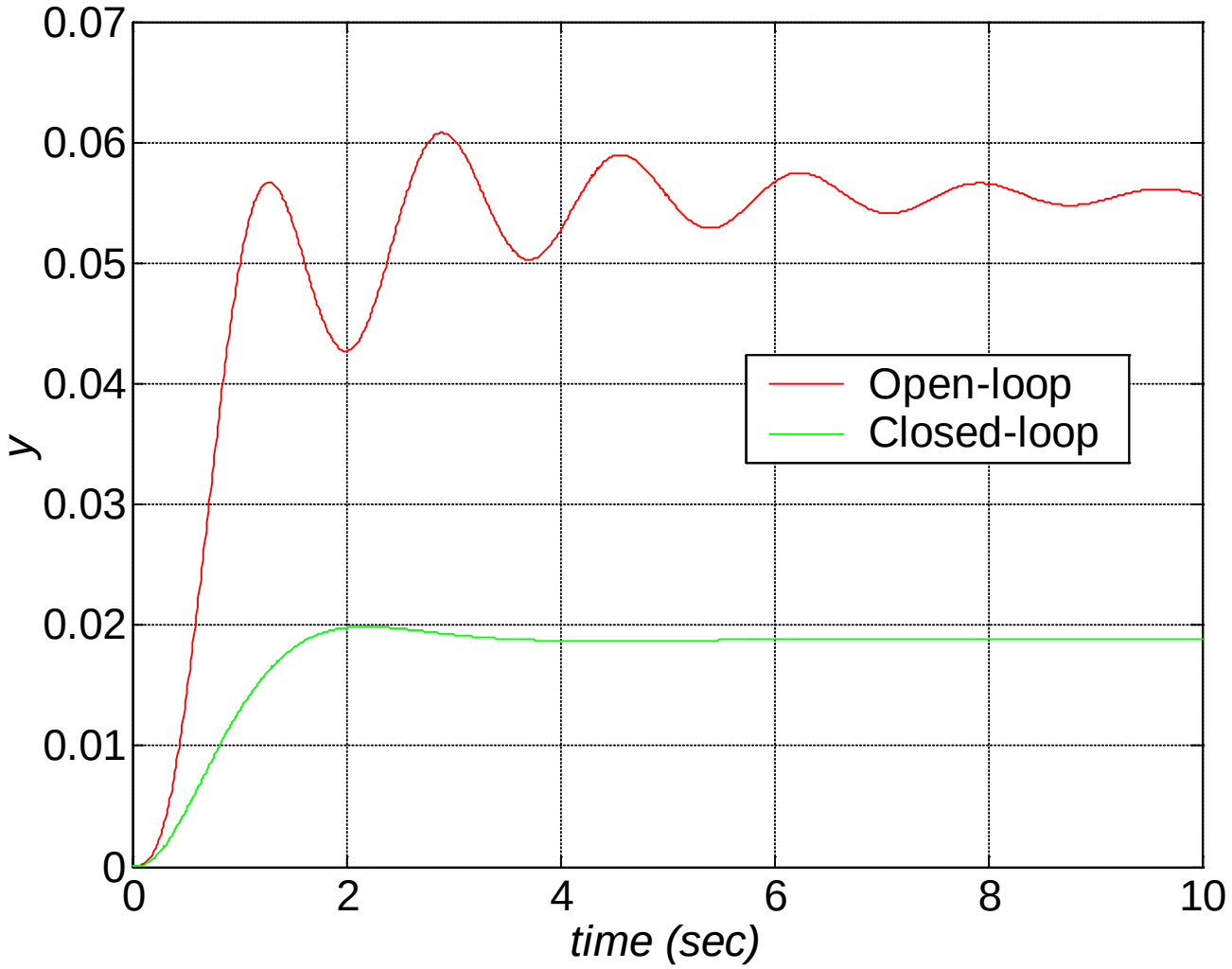
where:

α_{i-1} are the desired characteristic polynomial coefficients

a_{i-1} are the open-loop characteristic polynomial coefficients

Controller Example Simulation

Perhaps the most important step in controller design is simulation of the obtained closed-loop vs. original open-loop results to demonstrate if there is any improvement (did you achieve your controller design goals?). For our controller design example, using our existing simulation techniques and *A–BK* for the closed loop dynamics matrix, with a unit step input:



The transient response looks better (designed for **6% overshoot** and **3 sec settling time**), but what happened to the steady-state output magnitude? This is called **output attenuation** – the closed-loop system has a stiffer virtual spring than the open-loop system.

3.2.3 Ackerman's Formula

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

Ackerman's formula is:

where:

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

and $\alpha(A)$ is the desired closed-loop system controller characteristic polynomial evaluated with the system dynamics matrix A instead of the scalar Laplace variable s:

Ackerman's Formula Example

Same open-loop system and desired behavior as before.

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

$P = \begin{bmatrix} B & AB & A^2B \end{bmatrix} =$

$P^{-1} =$

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

$$\alpha(A) = A^3 + 16A^2 + 39.55A + 53.26I_3 = \begin{bmatrix} 35.26 & 24.55 & 14 \\ -252 & -174.74 & -3.45 \\ 62.10 & -200.25 & -167.84 \end{bmatrix}$$

$K =$

MATLAB functions for controller design:

```
K = place(A,B,des)
```

- **A,B** are the open-loop systems dynamics and input matrices
- **des** is an array containing the *n* numerical desired poles

```
K = acker(A,B,des)
```

- same inputs as for **place**
- for single input only – since the system is linear this will yield the same **K** results as a single input **place**

3.2.4 Input Correction

As seen in the above controller design MATLAB simulation, 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 perform input correction on reference input $r(t)$ so the closed-loop controller yields the original open-loop steady-state value.

One-dof translational mechanical system:

For constant *f*: ($\ddot{y} = \dot{y} = 0$ at steady-state)

State-space description

At steady-state:

so

Input Correction Example

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

Open-loop steady-state value:

Check:

also checks with MATLAB simulation results. Now, the steady-state value with closed-loop state feedback is:

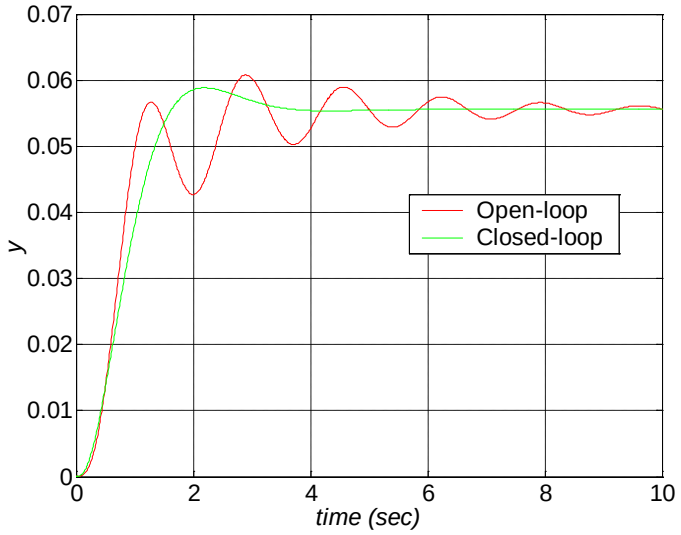
So for the closed-loop system with state feedback controller, we must modify the reference input *r*:

where *u* is the open-loop input magnitude and *corr* is the correction factor.

For SISO systems, the correction factor is a ratio, found 2 ways (see equations below). This is the ratio of the effective closed-loop system stiffness with controller to the actual open-loop system stiffness. For the example:

$$corr=\frac{A_{CL}(3,1)}{A(3,1)}=\frac{53.26}{18}=\frac{y_{ssOL}}{y_{ssCL}}=\frac{0.056}{0.019}=2.96$$

This is the reference input used to produce the corrected closed-loop controller results with the desired steady-state value, shown below. The closed-loop transient response was designed for **6% overshoot** and **3 sec settling time** (did this succeed?).



For input correction in MIMO systems, output attenuation correction is not so simple:

1) For the open-loop system, also use:

$$\{X_{ss}\} = -[A]^{-1}\{B\}\{U\}$$

to determine the desired steady-state values (same as the open-loop system).

2) Then:

$$\begin{aligned} \dot{X}_{ss} &= 0 = A_c X_{ss} + B_c r \\ B_c r &= -A_c X_{ss} \\ r &= -\text{pinv}(B_c) A_c X_{ss} \end{aligned}$$

where **`pinv`** is the MATLAB Moore-Penrose pseudoinverse of non-square matrix B_c.

3.3 Linear Full State-Feedback Observer Design

3.3.1 Background

Recall the state-space control law is $U(t) = r(t) - KX(t)$. Controller design is straightforward and the same pole-placement procedure is used for any controllable system. You can use the special **CCF** decoupled solution or Ackerman's formula for SISO systems to obtain the unique controller solution. Use MATLAB function **`place`** for MIMO (or SISO) systems.

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 physical quantities. So we must design an observer.

Observer – estimate full state vector for use in feedback controller based on known inputs $U(t)$ and measured outputs $Y(t)$. We will integrate the observer with the current state-feedback controller system.

Observer Diagram (high-level):

X true current state vector

$\hat{X}$ estimate for current state vector, from Observer

Modified control law:

We want to drive the estimation error to zero:

This error must converge to zero faster than the transient response dynamics of the linear system with controller. Therefore, we choose observer eigenvalues at least ten times higher than the controller poles.

Original open-loop system:

Form for Observer:

Choose the same form as the open-loop plant dynamics. Use a state vector estimate, add a zero term, and assume D = [0]. L is the observer gain matrix, with dimension (n x p).

Observer Diagram (details):

Derivation of error dynamics equation:

If all $\mathrm{Re}(\mathrm{eig}(\hat{A})) < 0$ then this is an asymptotically stable error equation and:

If the original open-loop system represented by *A*, *C* is *completely observable*, then we can arbitrarily place eigenvalues of *A* – *LC* by proper selection of matrix *L*. We must ensure the observer is *stable* and we can control the rate and transient nature of convergence $\hat{X} \to X$.

3.3.2 Observer Design

We must place the eigenvalues of *A* – *LC* to achieve observer estimation convergence faster than the closed-loop controller transient response. Assume we have already found controller gain matrix *K*. Compare to the controller design problem where we placed eigenvalues of *A* – *BK* to achieve stability and the desired transient response design specifications. The order is reversed:

BK
LC

A matrix and its transpose have the same characteristic equation and the same eigenvalues. Therefore, let us select *L* to control the eigenvalues of:

and then the eigenvalues of *A* – *LC* will be the same. Match format from controller design:

The algorithms for controller design (**CCF**, Ackerman, **`place`**) apply to observer design if:

A, *C* must be completely state *observable*.

The controllability matrix $P = \begin{bmatrix} B & AB & \cdots & A^{n-1}B \end{bmatrix}$ with A^T and C^T is:

Full-state observer design is ***dual*** to the pole placement problem for design of full-state-feedback controllers. Partial state observers are also possible (this would be more efficient: just estimate those states you can't measure via sensors).

Example: Observer pole placement via observer gain matrix *L*

This is the same given open-loop system as the controller example; we use the controller $K = \begin{bmatrix} 35.26 & 24.55 & 14.00 \end{bmatrix}$ designed previously. The form of the unknown *L* is given also.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -18 & -15 & -2 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad D = [0] \qquad L = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix}$$

In the controller design algorithm, replace:

Choose observer eigenvalues ten times higher than controller eigenvalues $s_{1,2,3} = -1.33 \pm 1.49i, -13.33$, which were designed for **6% overshoot** and **3 sec settling time**.

Observer eigenvalues $s_{1,2,3} = -13.33 \pm 14.90i, -133.33$

So the desired observer characteristic equation is:

$$\alpha_{OBS}(s) = s^3 + 160s^2 + 3955s + 53260$$

Desired observer coefficients are multiples of the controller coefficients, due to the 10x factor:

$$\alpha(s) = s^3 + 16s^2 + 39.55s + 53.26$$

Match like powers of *s* between the function of *L* and desired characteristic equation:

This coefficient matrix looks like a skew-transpose of P^{-1} from controller design (more evidence of ***duality***):

$$P^{-1} = M = \begin{bmatrix} 15 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Actually we don't need a matrix because the equations are partially decoupled (they would be fully decoupled if you started with **OCF**). Caution – you must use only one realization *A, B, C, D* to design the controller and observer, i.e. you cannot choose A_{CCF} to obtain a decoupled controller solution and then use A_{OCF} to obtain a decoupled observer solution. Those two K_{CCF} and L_{OCF} would not work together.

Units?

3.3.3 Ackerman's Formula

Ackerman's formula for observer design is:

1) Ackerman's observer formula: $L^T = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} (Q^T)^{-1} \alpha_{OBS}(A^T)$

where the observability matrix *Q* is: $Q = \begin{bmatrix} C^T & A^T C^T & \cdots & (A^{n-1})^T C^T \end{bmatrix}^T$

and $\alpha_{OBS}(A^T)$ is the desired observer characteristic polynomial evaluated with A^T. For more evidence of controller/observer duality, remember Ackerman's formula for controller design is:

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

2) alternate Ackerman's observer formula (transpose the formula of 1):

$$L = \alpha_{OBS}(A)(Q)^{-1} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

3) MATLAB: **`LT = place(A',C',obs);`**

where **`obs`** are the *n* desired observer eigenvalues, ten times higher than the closed-loop controller poles **`des`**, and we must use the transposes of *A* and *C*. The result, **`LT`**, must in turn be transposed to get *L*.

Controller & Observer Example Summary

Original third-order open-loop LTI system, expressed in **CCF**:

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

State-feedback controller K:

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

$$A_c = A - BK = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -53.26 & -39.55 & -16 \end{bmatrix} \qquad B_c = B \qquad C_c = C \qquad D_c = D = [0]$$

State-feedback observer L:

$$L = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} 158 \\ 3624 \\ 43624 \end{bmatrix} \qquad \hat{A} = A - LC = \begin{bmatrix} -158 & 1 & 0 \\ -3624 & 0 & 1 \\ -43642 & -15 & -2 \end{bmatrix}$$

Example Redux: Same observer design example, use OCF

The solution for *L* was not fully decoupled when we started with the system in **CCF**. It will be a fully decoupled solution for *L* when starting with **OCF**.

$$A=\begin{bmatrix}0 & 0 & -18\\ 1 & 0 & -15\\ 0 & 1 & -2\end{bmatrix} \qquad B=\begin{bmatrix}1\\0\\0\end{bmatrix} \qquad C=\begin{bmatrix}0 & 0 & 1\end{bmatrix} \qquad L=\begin{bmatrix}L_1\\L_2\\L_3\end{bmatrix}$$

$$A^T-C^TL^T=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18 & -15 & -2\end{bmatrix}-\begin{bmatrix}0\\0\\1\end{bmatrix}\begin{bmatrix}L_1 & L_2 & L_3\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ -18-L_1 & -15-L_2 & -2-L_3\end{bmatrix}$$

$$\left|sI_3-\left(A^T-C^TL^T\right)\right|=\alpha_{OBS}(s,L_i)=s^3+\left(L_3+2\right)s^2+\left(L_2+15\right)s+\left(L_1+18\right)$$

Use the same desired observer characteristic equation:

$$\alpha_{OBS}(s)=s^3+160s^2+3955s+53260$$

Matching like powers of *s* between the function of L_i and desired observer characteristic polynomial yields a fully decoupled solution:

$$\begin{aligned}L_3+2&=160\\ L_2+15&=3955\\ L_1+18&=53260\end{aligned} \qquad L_{OCF}=\begin{bmatrix}L_1\\L_2\\L_3\end{bmatrix}=\begin{bmatrix}53242\\3940\\158\end{bmatrix}$$

Note that this L_{OCF} result is different than the original **CCF** example for observer design. This makes sense since we started with different *A, C* matrices. Check these new L_{OCF} results to ensure the proper observer eigenvalues were created in the observer system matrix:

$$\hat{A}=A-L_{OCF}C=\begin{bmatrix}0 & 0 & -53260\\ 1 & 0 & -3955\\ 0 & 1 & -160\end{bmatrix}$$

The eigenvalues of $\hat{A}=A-L_{OCF}C$ are $-13.33\pm14.89i,-133.33$, identical to those specified (ten times the desired controller poles).

Also note that we cannot use the previously-calculated full state feedback gain matrix *K* for the controller based on the **CCF** system, but we must calculate the new one based on the **OCF** system (this solution, not shown, is NOT fully decoupled):

$$K_{OCF} = \begin{bmatrix}14.00 & -3.45 & -167.84\end{bmatrix}$$

$$A_c=A-BK_{OCF}=\begin{bmatrix}-14.00 & 3.45 & 149.84\\ 1 & 0 & -15\\ 0 & 1 & -2\end{bmatrix} \qquad B_c=B \qquad C_c=C \qquad D_c=D=[0]$$

3.4 Combined Closed-Loop Controller and Observer

We now combine the actual and estimated state and observer error dynamics to derive the combined controller/observer dynamics. The original open-loop system, the modified feedback control law using the observer estimate for states, and the observer error dynamics equation are:

Define a new state vector *W* containing the original state and the observer estimation error:

Derive the super system including the full-state feedback and observer gain matrices *K* and *L*:

The system size has doubled to 2*n*. To simulate the closed-loop system dynamics with controller and observer, use our existing MATLAB methods, with A_r, B_r, C_r, D_r, W, r in place of A, B, C, D, X, U. For the example:

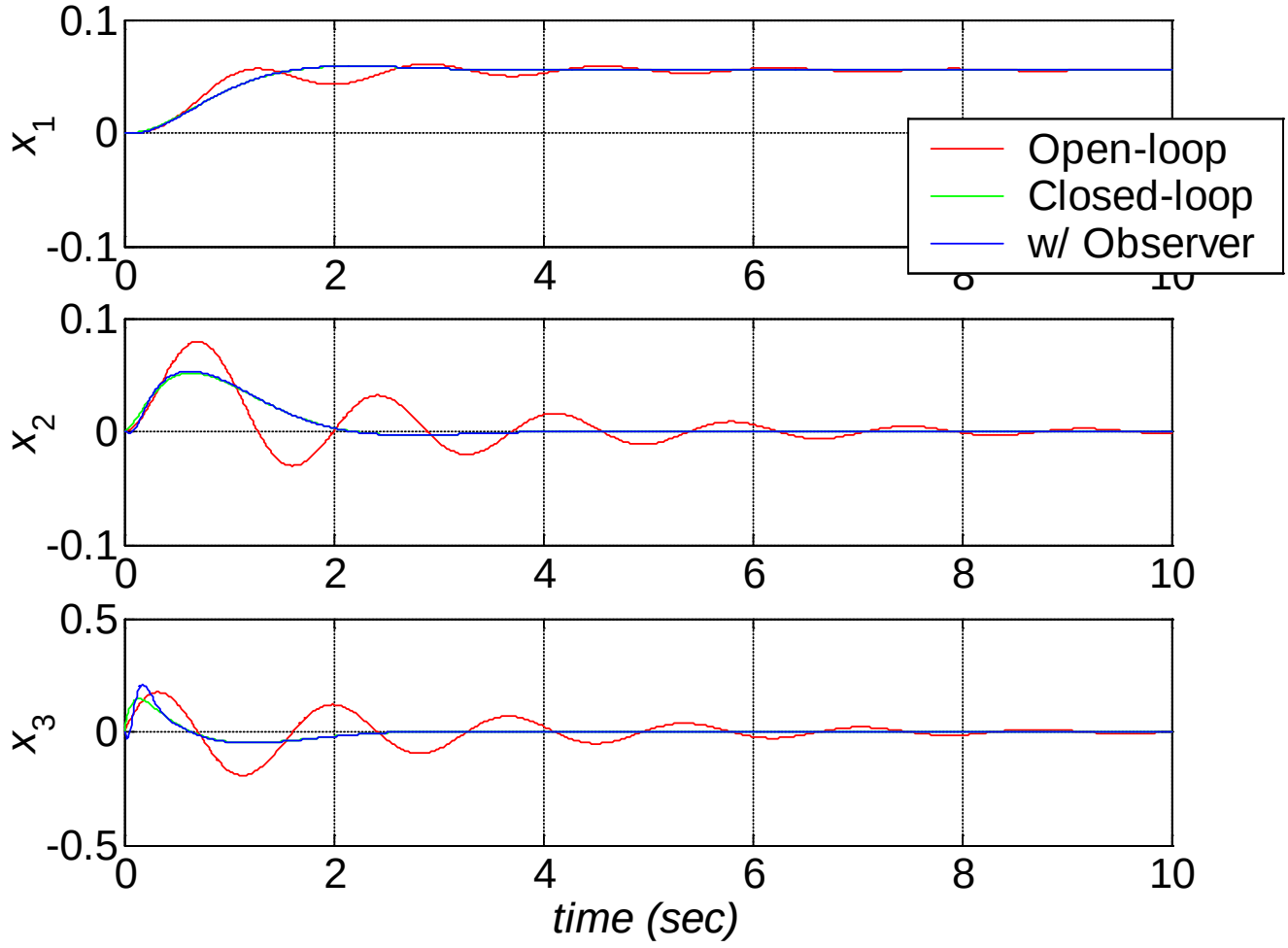
$$A_r = \begin{bmatrix} A - BK & BK \\ 0 & A - LC \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -53.26 & -39.55 & -16 & 35.26 & 24.55 & 14 \\ 0 & 0 & 0 & -158 & 1 & 0 \\ 0 & 0 & 0 & -3624 & 0 & 1 \\ 0 & 0 & 0 & -43642 & -15 & -2 \end{bmatrix}$$

$$B_r = \begin{bmatrix} B \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad C_r = \begin{bmatrix} C & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad D_r = D = [0]$$

One must include some artificial observer error via initial conditions on the observer error *e*, otherwise the simulation will be perfect and no observer estimation convergence effect can be seen. This is agrees with the real-world situation since we cannot know the initial error in general.

The super-system simulation results for the Example are plotted below using **`lsim`** with A_r, B_r, C_r, D_r, and W.



Note the observer error converges faster than the closed-loop controller dynamics, as desired.

Observer Error

$$\dot{e} = (A - LC)e$$

The Laplace transform of the above equation is:

$$sE(s) = (A - LC)E(s)$$

$$(sI - (A - LC))E(s) = 0$$

If $|sI - (A - LC)| = 0$ (as it must be for Observer L design), there are infinite solutions $E(s)$

If $|sI - (A - LC)| \neq 0$ there is a unique solution $E(s)$

3.5 Closed-Loop System Input Effort

We must plot the required input effort in the simulation of closed loop systems (controller, controller/observer) in order to ascertain if the required actuator commands are feasible:

- The actuator commands cannot change too fast to be realizable in the real world
- The actuator commands cannot exceed the limits of the physical actuator. If the limits are violated this is called actuator saturation and the simulated control is not possible.

Closed-loop system control law:

Closed-loop system control law with observer:

Calculate and plot $U(t)$ using MATLAB:

Even better, using Simulink:

Include real-world discussion on your closed-loop controller/observer simulation results regarding actuator feasibility.

3.6 Disturbances Evaluation after Controller/Observer Design

What are disturbances?

Disturbances are unknown, unmodeled, unwanted actions external to the control system that interfere with the expected behavior. Examples:

Disturbance sources:

- Friction
- Wind
- Bumps
- Unmodeled dynamics
- Non-linearities
- Changing load inertia
- System changes over product life
- Contact with workspace
- Gravity

We can model disturbances as input(s) affecting (subtracting from or adding to) the actuator input effort (the same units as your open-loop system input). The disturbance is felt by the actuator as something interfering with normal operation.

Compare transient response and steady-state error associated with open- and closed-loop systems, with and without the disturbances included. Generally you will find that the closed-loop system with controller and observer will perform much better than the open-loop system given unwanted disturbances.

General disturbance diagrams (open- and closed-loop):

4. Optimal Control

Linear state-feedback controllers can be designed to stabilize a controllable system and provide desired transient response.

Can also optimize (usually minimize) objective functions

- time
- error
- energy
- combinations
- other

Performance Index (Objective Function):

If $r(t) = 0$, the controller is a *regulator* (reject initial conditions and/or disturbances) whose job is to maintain the equilibrium state $X_e = 0$. Any deviation X is the error.

4.1 X^TX Optimal Controller

Let us choose to minimize the error X for our objective function:

(scalar)

To obtain minimum J, we assume the existence of an exact differential such that:

where Q is a constant symmetric matrix. Derivation details:

which satisfies the exact differential when:

This is the same form as the Lyapunov stability analysis equation.

For the SISO case, solve for:

$$Q = Q\left(k_1, k_2, \cdots, k_n\right)$$

Then:

Assuming stability (this must be the case), $X\left(\infty\right) = 0$, so:

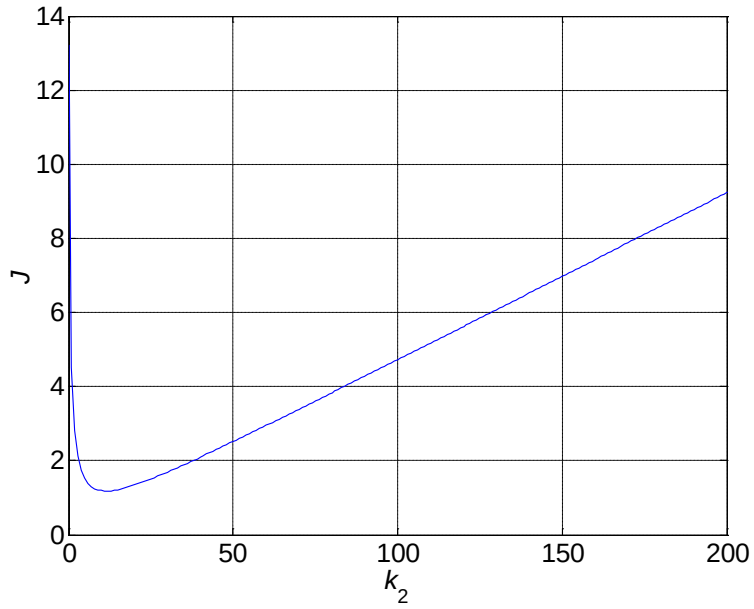
Choose $k_1, k_2, \cdots, k_n$ to minimize *J*.

Optimal Control Example

$$\text{Minimize} \quad J = \int_0^\infty \left(X^T X\right) dt$$

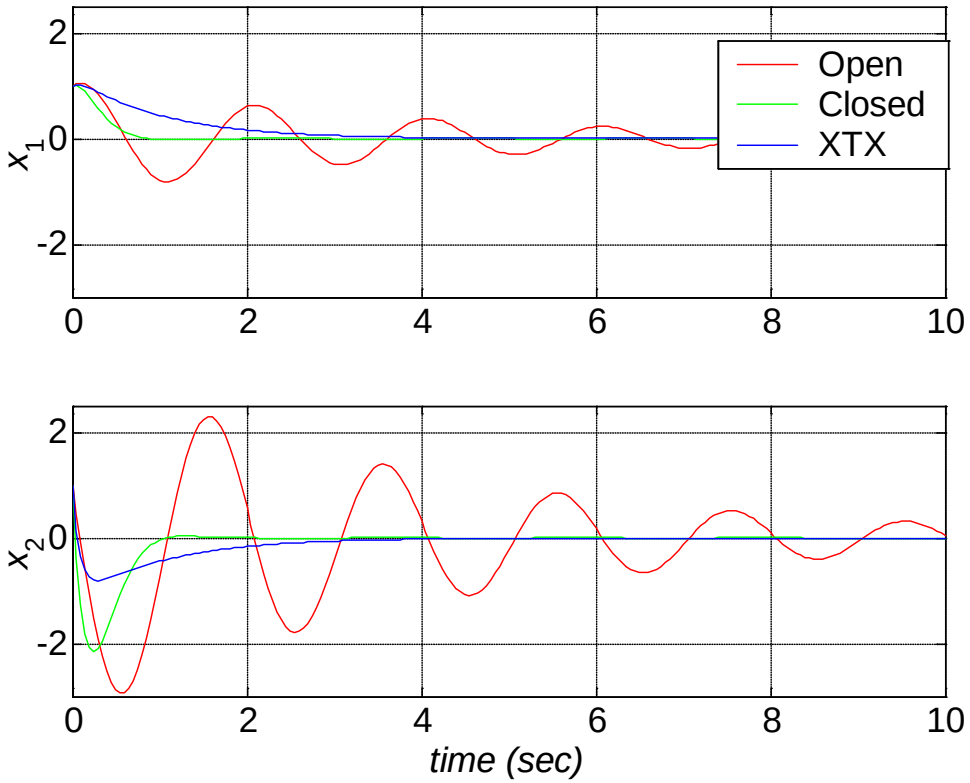
$$A = \begin{bmatrix} 0 & 1 \\ -10 & -0.5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad u = -KX \qquad K = \begin{bmatrix} k_1 & k_2 \end{bmatrix} \qquad X\left(0\right) = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

Minimize *J*:



For $k_1 = 1$ and $X(0) = \{1 \quad 1\}^T$, the optimal value is $k_2 = 11.5$. The second k_2 value (–12.5) causes the closed-loop system to be unstable.

We plot the state step responses below and compare with results from a standard controller design. With this J, we minimize error w/o regard to time (slower response but less error).



Standard, "tight" controller:	$\xi = 0.8$	and	$\omega_n = 5$ rad/sec
Optimal X^TX regulator:	$\xi = 1.81$	and	$\omega_n = 3.32$ rad/sec

4.2 Linear Quadratic Regulator (LQR) Optimal Controller

We now minimize a new objective function to consider input effort as well as the previous state error:

where $Q \in n \times n$ and $R \in m \times m$ are weighting matrices. This objective function minimizes control effort U in addition to error X.

The LQR derivation requires Hamiltonian theory from advanced dynamics (see Williams and Lawrence). The highlights are given here. Minimize J subject to:

Solve the Matrix-Ricatti Equation for a new gain matrix K_{LQR} ($K_{LQR} \in n \times n$):

The new LQR optimal control law is:

The new closed-loop system dynamics matrix A_{CLQR} is:

LQR Example

This is the same example as the open-loop system in the X^TX controller:

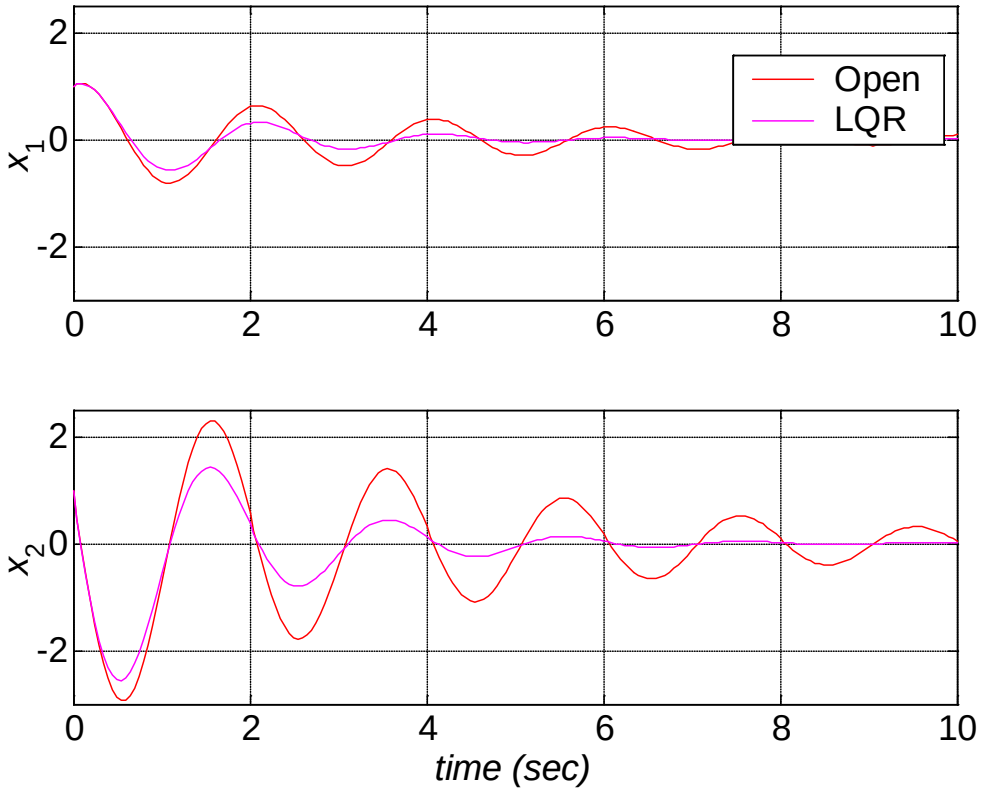
$$A = \begin{bmatrix} 0 & 1 \\ -10 & -0.5 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad Q = I_2 \qquad R = 1 \qquad X(0) = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

Solve via MATLAB:

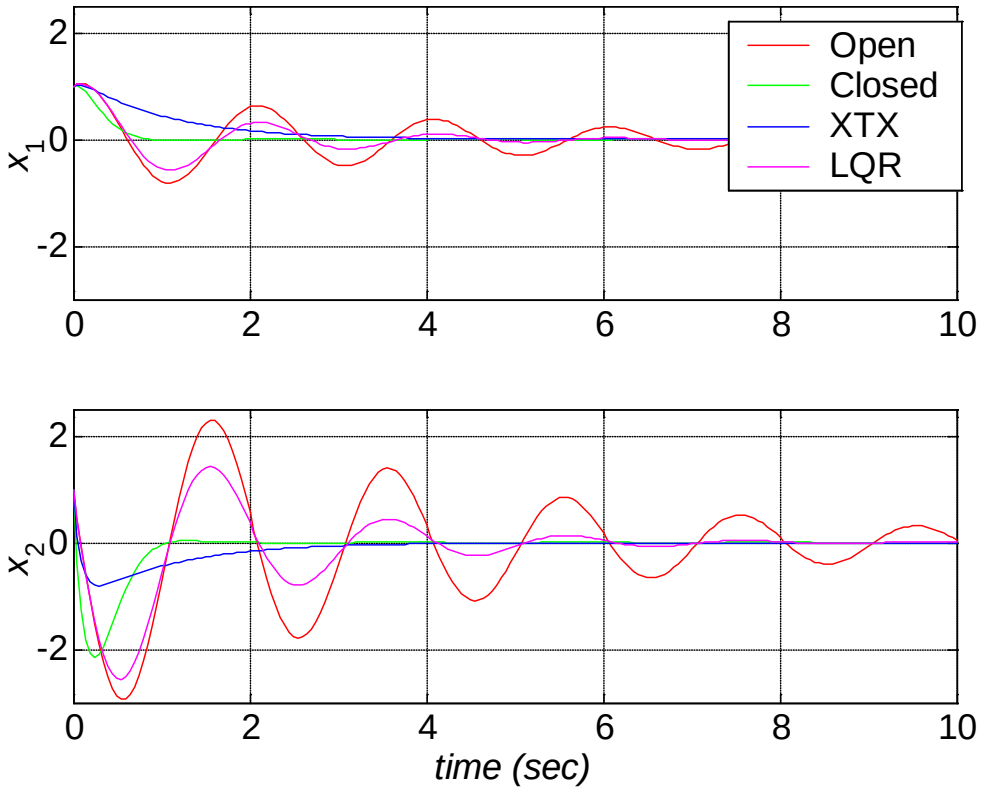
```
K_LQR = are(A,BB,Q);      % Solve algebraic Ricatti equation
```

Where BB = $BR^{-1}B^T = BB^T$ (for $R = 1$). Result:

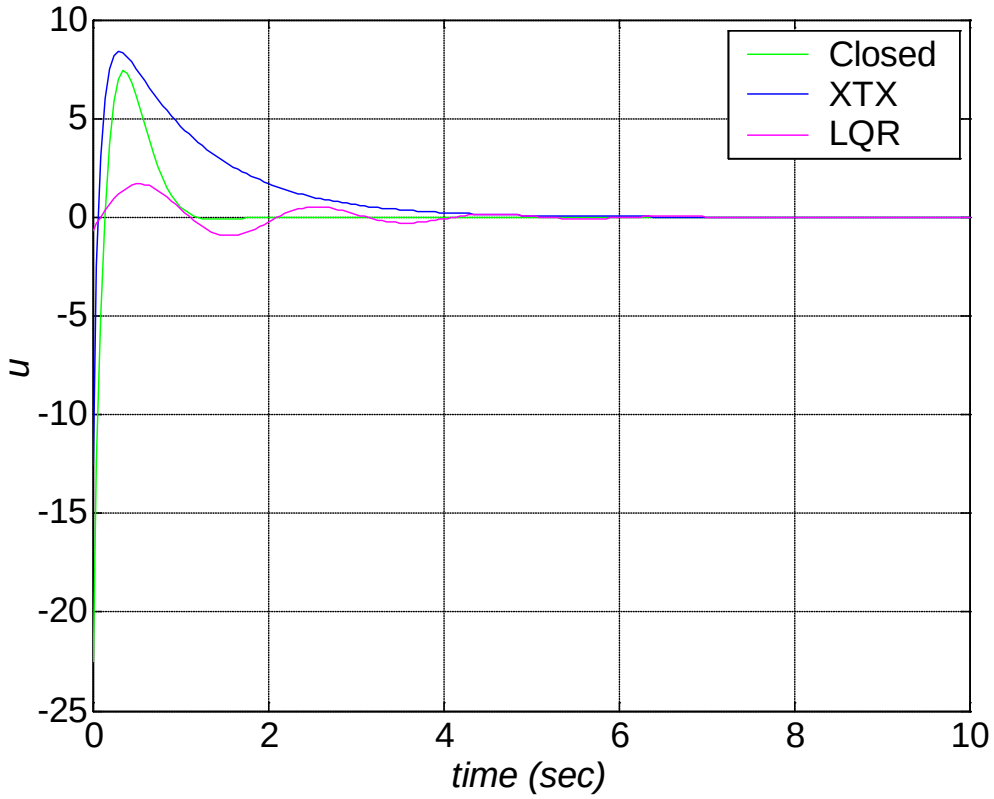
The simulation results (step responses to the given initial conditions) for both states are plotted below.



The LQR controller yielded $\xi = 0.18$ and $\omega_n = 3.17$ rad/sec. This is 'looser' transient response than we have been doing all quarter. What went wrong? Let us also compare this with the standard and X^TX controllers:



Let us plot the single input effort $U(t)$ required for each controller:



Nothing went wrong! The LQR is the first controller that tried to minimize $U(t)$ as well as the error $X(t)$. The input energy requirement is much lower for the LQR than the other controllers! Also, though the states look like they follow the open-loop case, they do damp out much faster, so the error is not nearly as bad as the open-loop case. There is a tradeoff between error and input effort that LQR balances, while the other methods ignore the input effort.

Comparison of all controllers in the example

System	$s_{1,2}$	ξ	ω_n
Open-loop	$-0.25 \pm 3.15i$	0.079	3.16
Standard state-space controller	$-4 \pm 3i$	0.80	5
X^TX optimal controller (overdamped)	$-1, -11$	1.81	3.32
LQR optimal controller	$-0.58 \pm 3.12i$	0.18	3.17