Simulation with Matlab

Professor Nhut Tan Ho

ME584

Agenda

- Representations of dynamic systems
- Simulation of
  - Linear systems
  - Non-linear systems
- Active learning activities: pair-share exercises

Representation of Dynamic Systems

Mathematical Representation

System Equations
Classical differential equation $$\ddot{x}+2\dot{x}+3x=f$$
Transfer function $$X(s)s^2+2sX(s)+3X(s)=F(s)$$ $$\frac{X(s)}{F(s)}=\frac{1}{s^2+2s+3}$$
State space $$\begin{bmatrix}\dot{x}\\ \dot{z}\end{bmatrix}=\begin{bmatrix}0 & 1\\ -3 & -2\end{bmatrix}\begin{bmatrix}x\\ z\end{bmatrix}+\begin{bmatrix}0\\ 1\end{bmatrix}f$$

Finding Roots of Polynomial

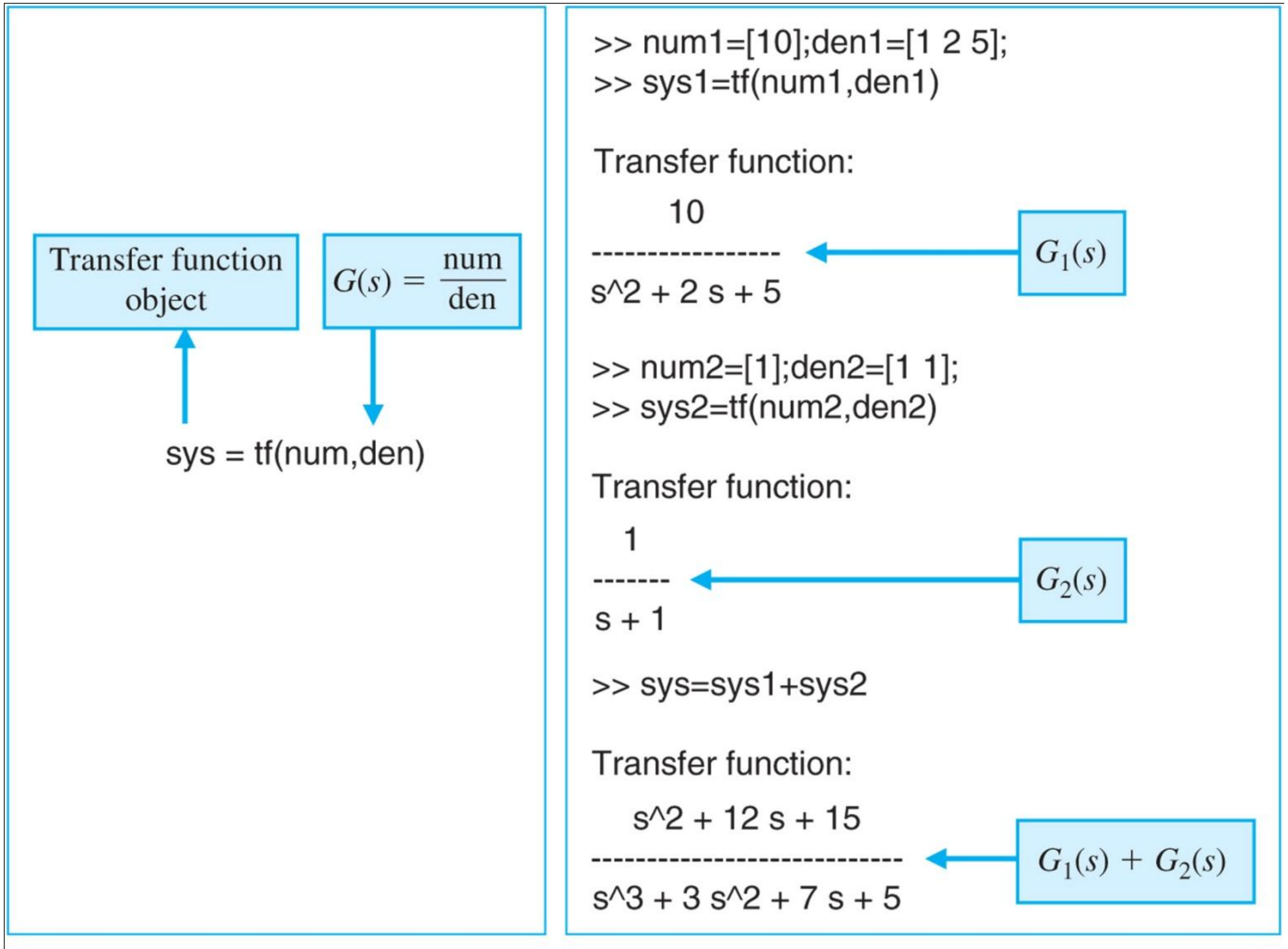
```
>>p=[1 3 0 4];
>>r=roots(p)

r =
 -3.3553
  0.1777+ 1.0773i
  0.1777- 1.0773i
>>p=poly(r)
p =


  1.0000     3.0000     0.0000     4.0000
```

$p(s) = s^3 + 3s^2 + 4$

Calculate roots of $p(s) = 0$.

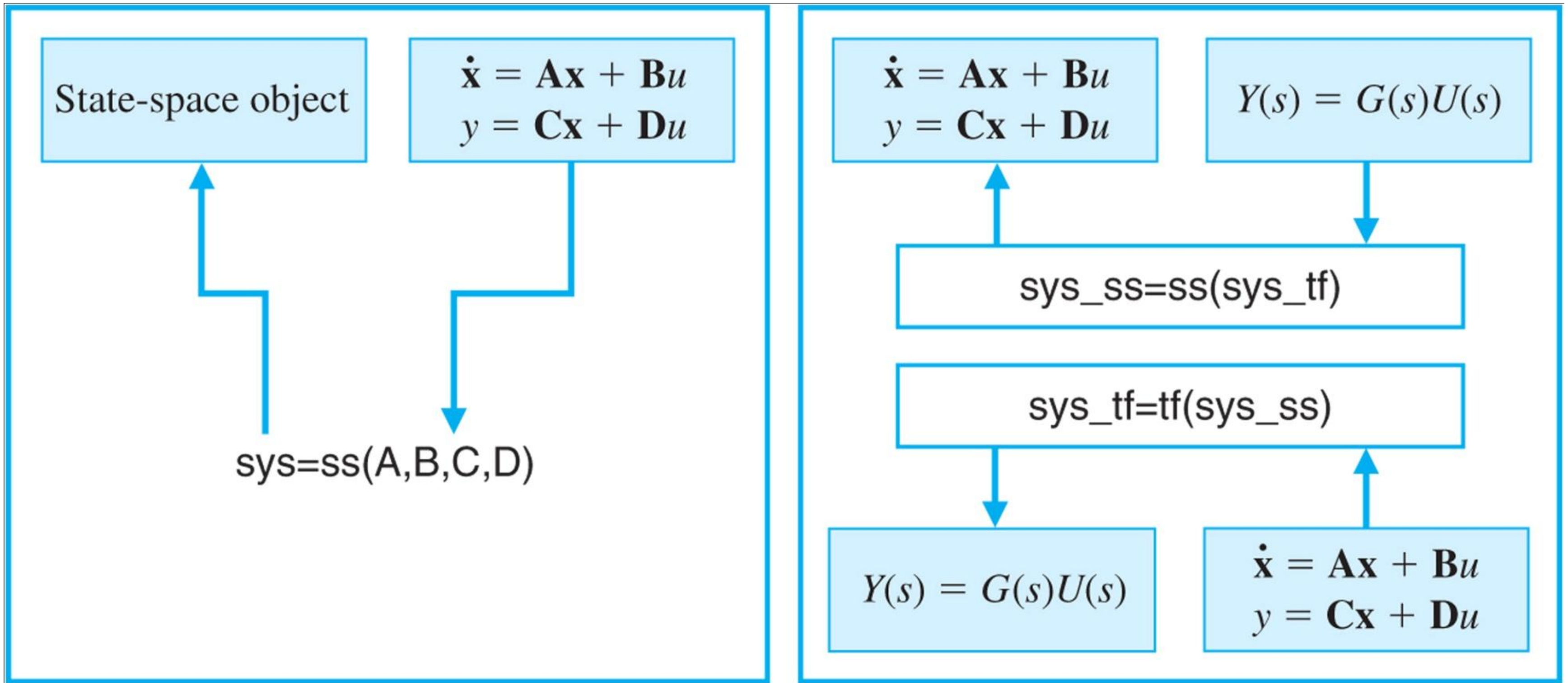
Reassemble polynomial from roots.

Creating Transfer Function



(a) (b)

ss function and model conversion



(a) (b)

Conversion Example

convert.m

```
% Convert G(s) = (2s^2+8s+6)/(s^3+8s^2+16s+6)
% to a state-space representation
%
num=[2 8 6]; den=[1 8 16 6]; sys_tf=tf(num,den);
sys_ss=ss(sys_tf);
```

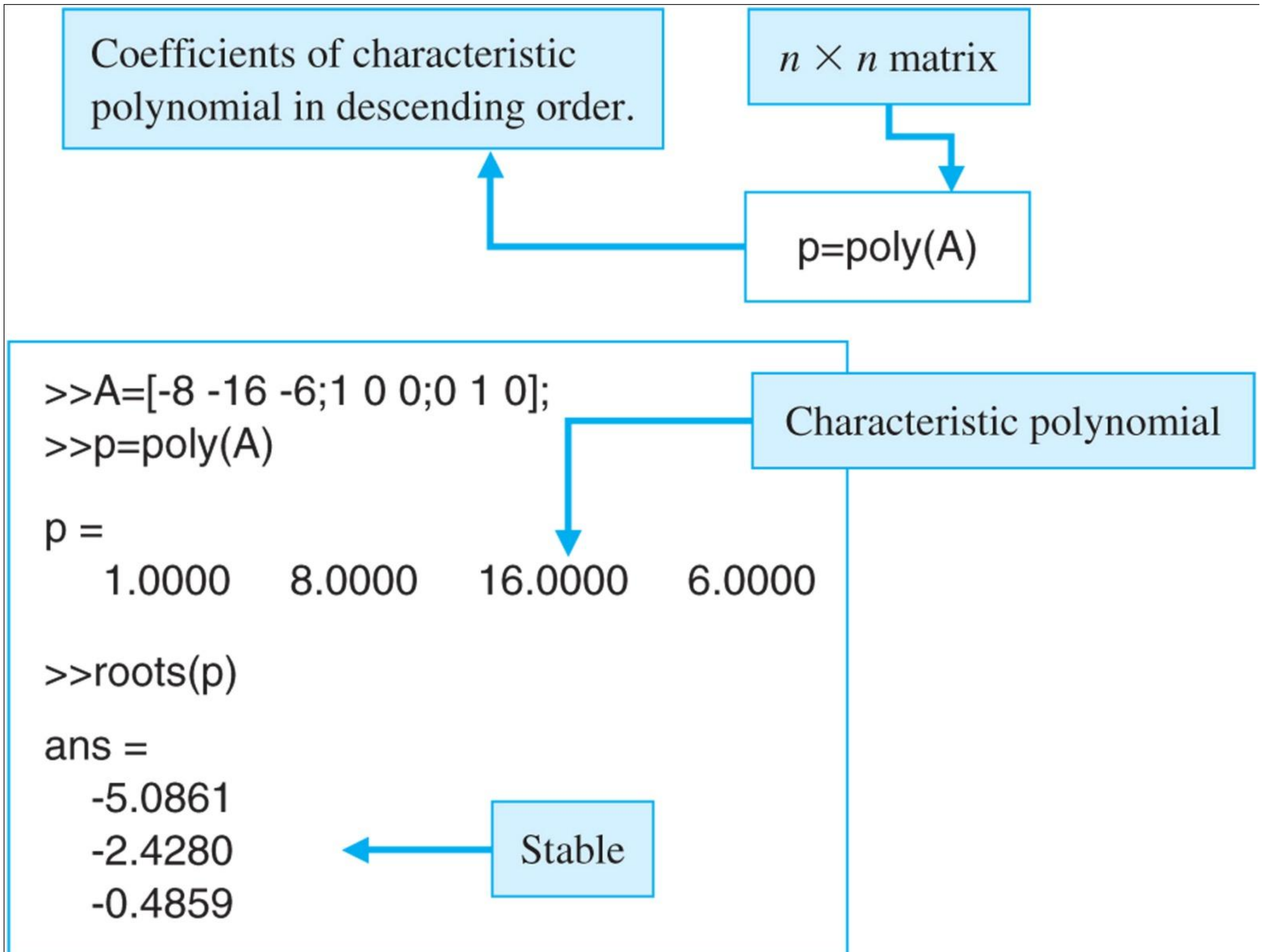
(a)

```
>>convert
a =
          x1      x2      x3
  x1      -8      -4    -1.5
  x2       4       0       0
  x3       0       1       0
b =
          u1
  x1       2
  x2       0
  x3       0
c =
          x1      x2      x3
  y1       1       1    0.75
d =
          u1
  y1       0
```

(b)

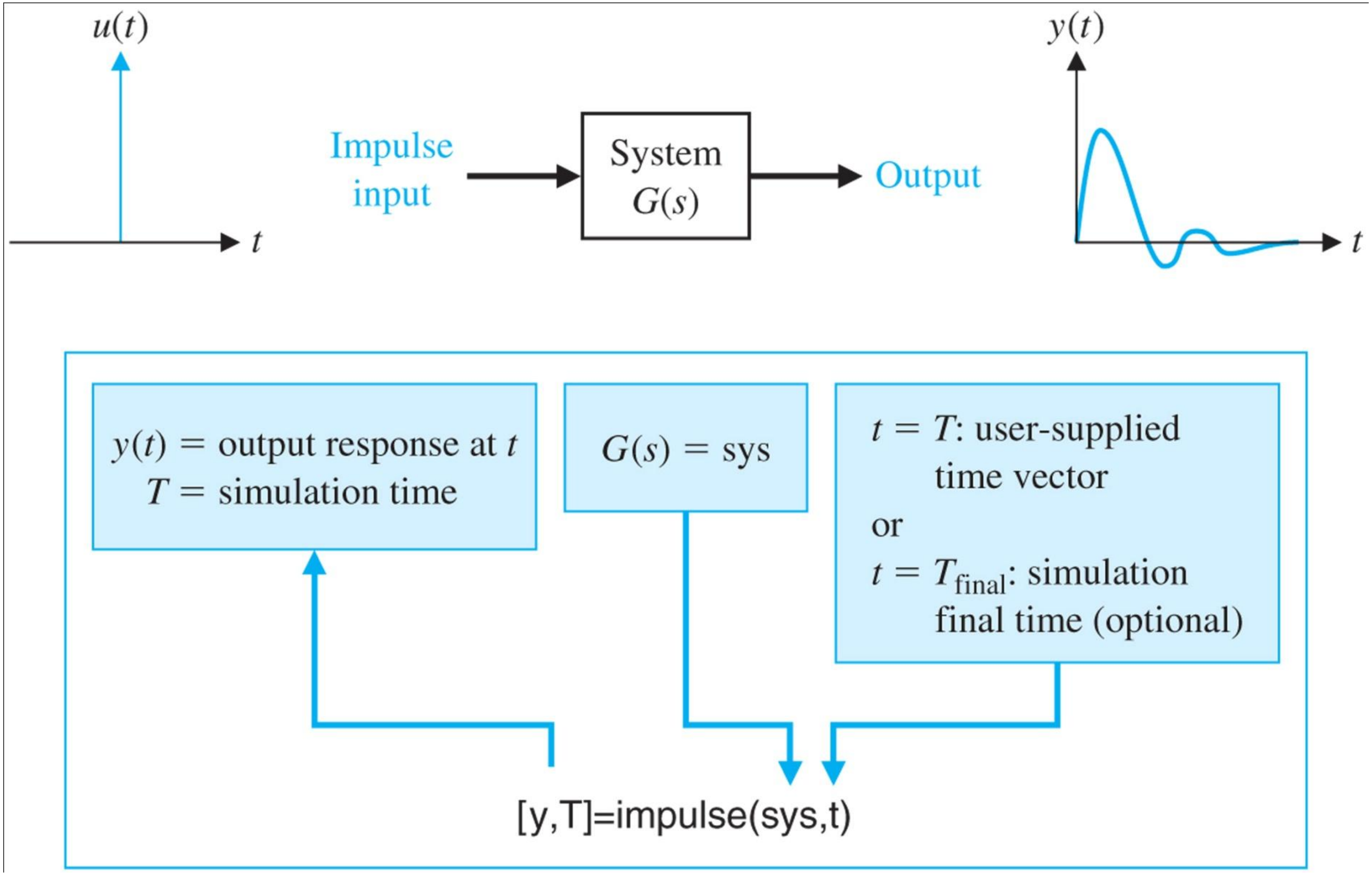
Roots of Characteristic Polynomial

If $\mathbf{A}$ is an $n \times n$ matrix, poly($\mathbf{A}$) is an $n + 1$ element row vector whose elements are the coefficients of the characteristic equation $\det(s\mathbf{I} - \mathbf{A}) = 0$.



Simulation of Linear Systems

Impulse Response

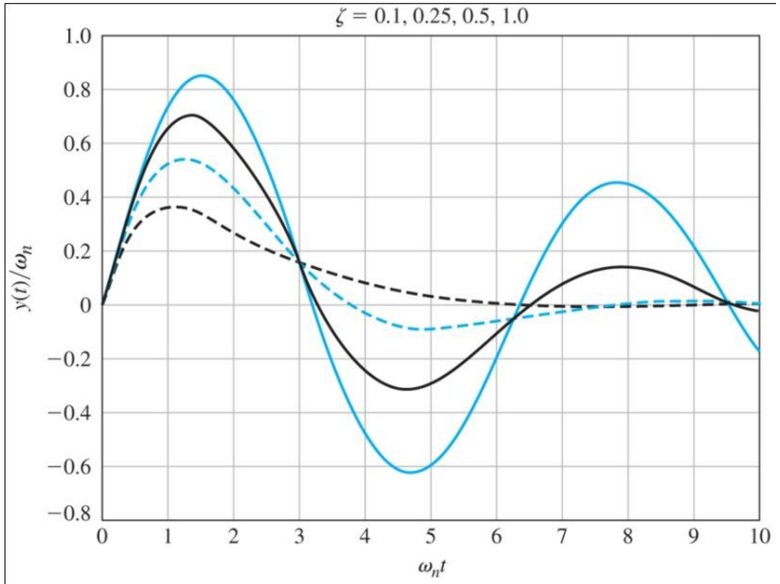


Impulse Example

Second order system

$$Y(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} R(s)$$

ω_n=1



(a)

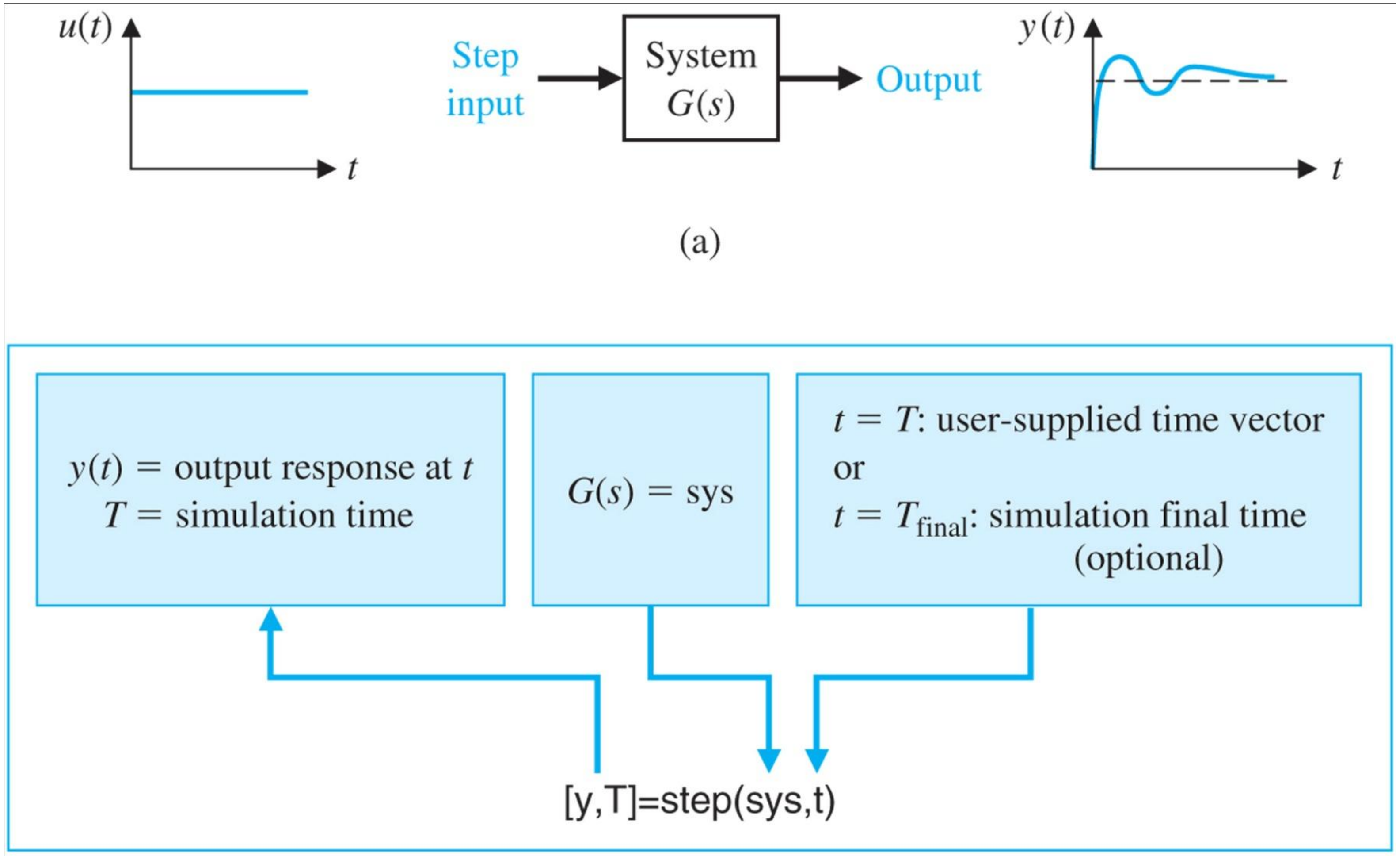
```
%Compute impulse response for a second-order system
%Duplicate Figure 5.6
%
t=[0:0.1:10]; num=[1];
zeta1=0.1; den1=[1 2*zeta1 1]; sys1=tf(num,den1);
zeta2=0.25; den2=[1 2*zeta2 1]; sys2=tf(num,den2);
zeta3=0.5; den3=[1 2*zeta3 1]; sys3=tf(num,den3);
zeta4=1.0; den4=[1 2*zeta4 1]; sys4=tf(num,den4);
%
[y1,T1]=impulse(sys1,t);
[y2,T2]=impulse(sys2,t);
[y3,T3]=impulse(sys3,t);
[y4,T4]=impulse(sys4,t);
%
plot(t,y1,t,y2,t,y3,t,y4)
xlabel(' \omega _nt'), ylabel('y(t)/\omega_n')
title('\zeta = 0.1, 0.25, 0.5, 1.0'), grid
```

Compute impulse response.

Generate plot and labels.

(b)

Step Function

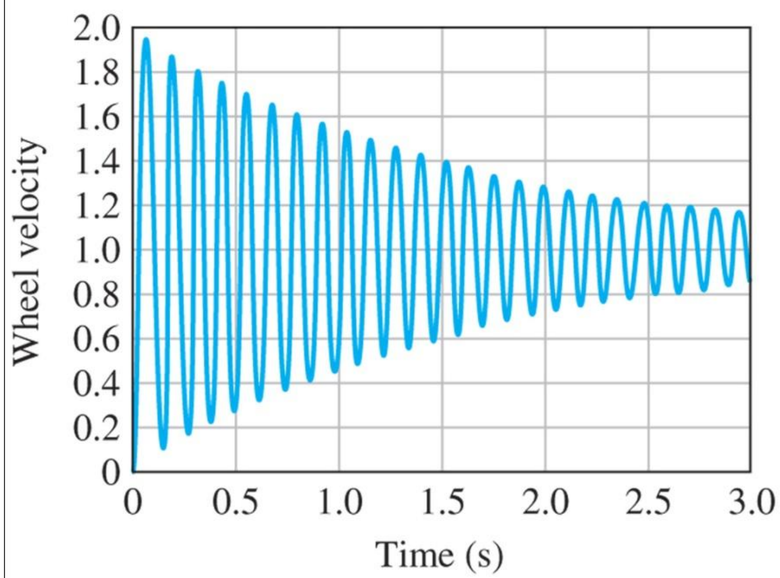


(b)

Step Function Example

Transfer function:

$$\frac{5400}{2\ s\^{}2 + 2.5\ s + 5402}$$

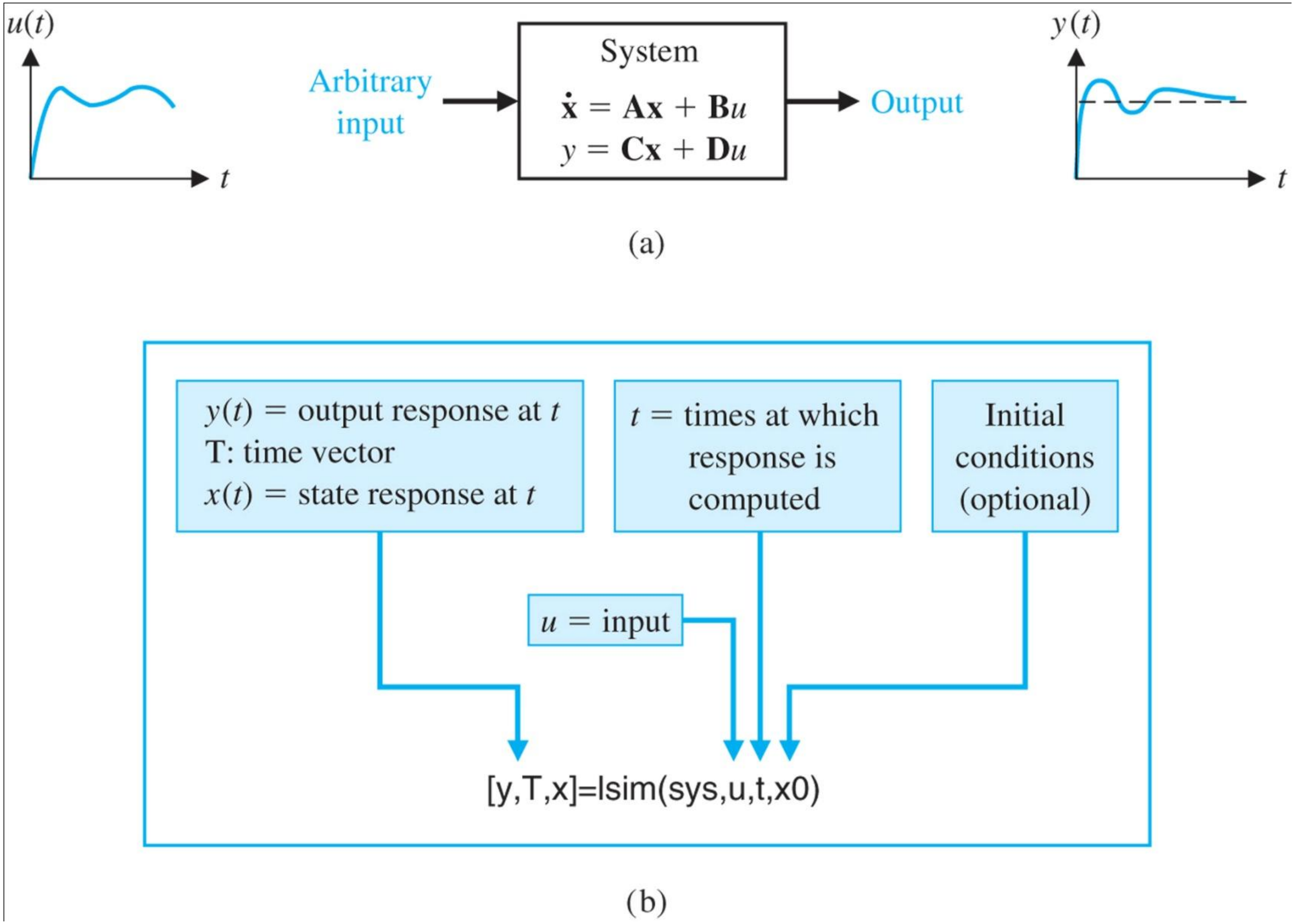


(a)

```
% This script computes the step
% response of the traction motor
% wheel velocity
%
num=[5400]; den=[2 2.5 5402]; sys=tf(num,den);
t=[0:0.005:3];
[y,t]=step(sys,t);
plot(t,y),grid
xlabel('Time (s)')
ylabel('Wheel velocity')
```

(b)

lsim function



lsim example

$$\mathbf{A} = \begin{bmatrix} -8 & -2 & -0.75 \\ 8 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0.125 \\ 0 \\ 0 \end{bmatrix},$$

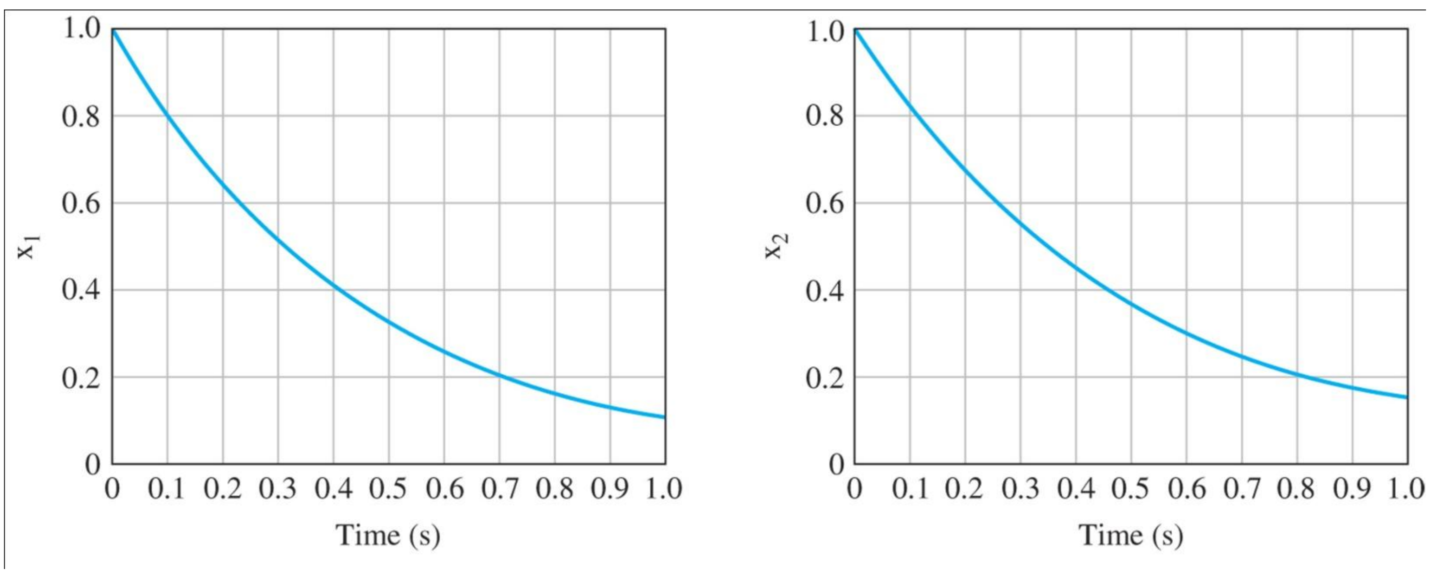
$$\mathbf{C} = [16 \quad 8 \quad 6], \mathbf{D} = [0]$$

```
A=[0 -2;1 -3]; B=[2;0]; C=[1 0]; D=[0];
sys=ss(A,B,C,D);
x0=[1 1];
t=[0:0.01:1];
u=0*t;
[y,T,x]=lsim(sys,u,t,x0);
subplot(121), plot(T,x(:,1))
xlabel('Time (s)'), ylabel('x_1')
subplot(122), plot(T,x(:,2))
xlabel('Time (s)'), ylabel('x_2')
```

State-space model

Initial conditions

Zero input

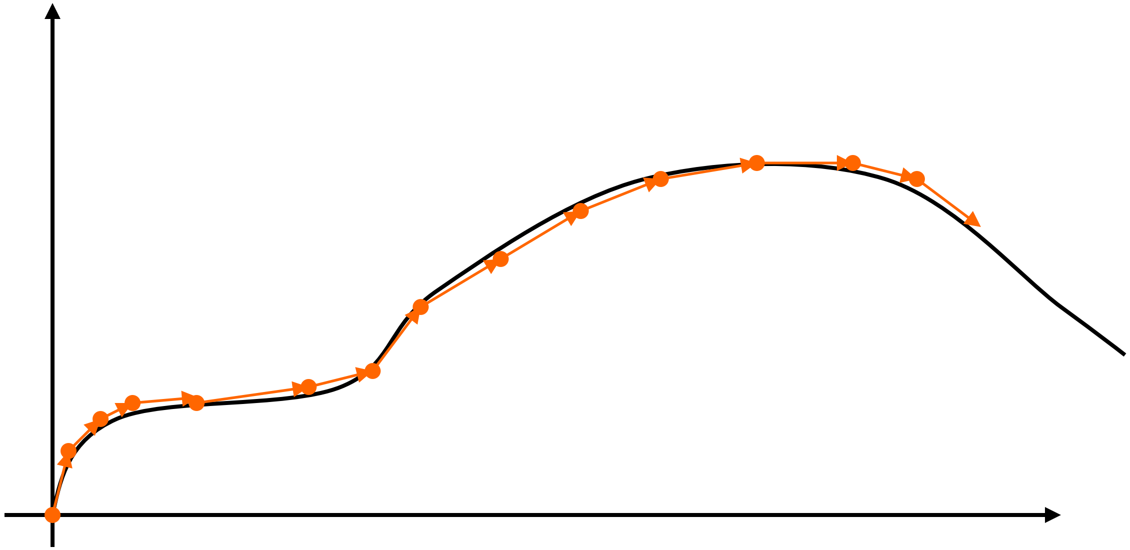


Simulation of Nonlinear Systems

ODE Solvers

ODE Solvers: Method

- Given a differential equation, the solution can be found by integration:



- Evaluate the derivative at a point and approximate by straight line
- Errors accumulate!
- Variable timestep can decrease the number of iterations

ODE Solvers: Matlab

- Matlab contains implementations of common ODE solvers

- Using the correct ODE solver can save you lots of time and give more accurate results
  - ode23
    - Low-order solver. Use when integrating over small intervals or when accuracy is less important than speed
  - ode45
    - High order (Runge-Kutta) solver. High accuracy and reasonable speed. Most commonly used.
  - ode15s
    - Stiff ODE solver (Gear's algorithm), use when the diff eq's have time constants that vary by orders of magnitude

ODE Solvers: Standard Syntax

- To use standard options and variable time step
  - [t,y]=ode45('myODE',[0,10],[1;0])

ODE integrator:
23, 45, 15s

ODE function

Time range

Initial conditions

- Inputs:
  - ODE function name (or anonymous function). This function takes inputs (t,y), and returns dy/dt
  - Time interval: 2-element vector specifying initial and final time
  - Initial conditions: column vector with an initial condition for each ODE. This is the first input to the ODE function
- Outputs:
  - t contains the time points
  - y contains the corresponding values of the integrated variables.

ODE Function

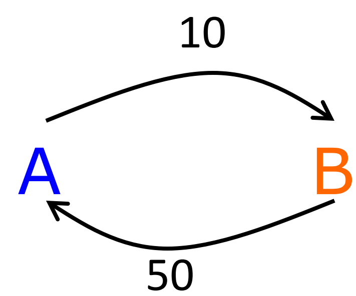
- The ODE function must return the value of the derivative at a given time and function value

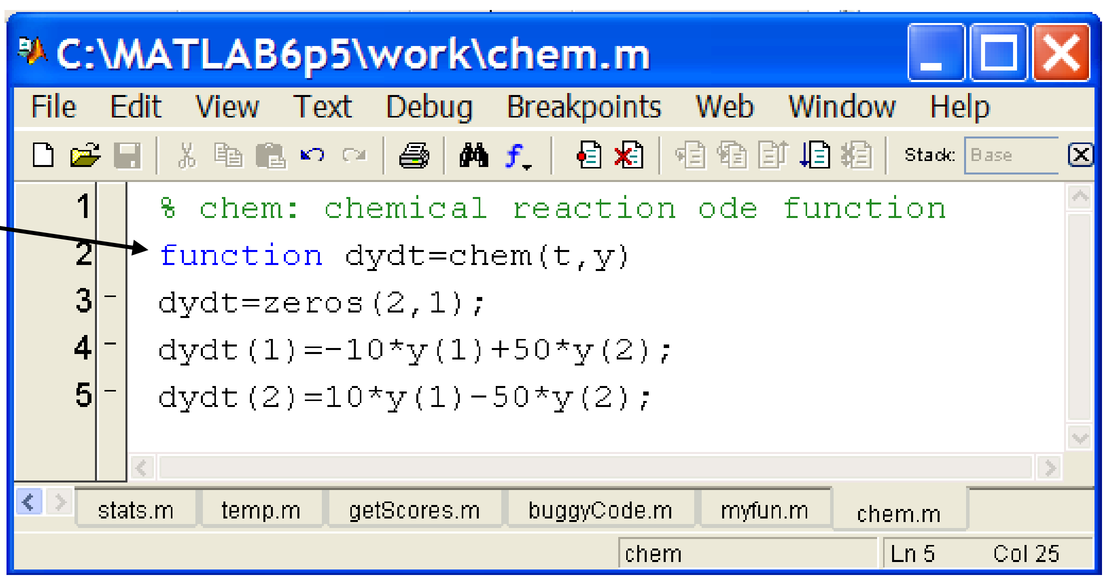
- Example: chemical reaction
  - Two equations

$$\frac{dA}{dt} = -10A + 50B$$

$$\frac{dB}{dt} = 10A - 50B$$

  - ODE file:
    – y has [A;B]
    – dydt has [dA/dt;dB/dt]



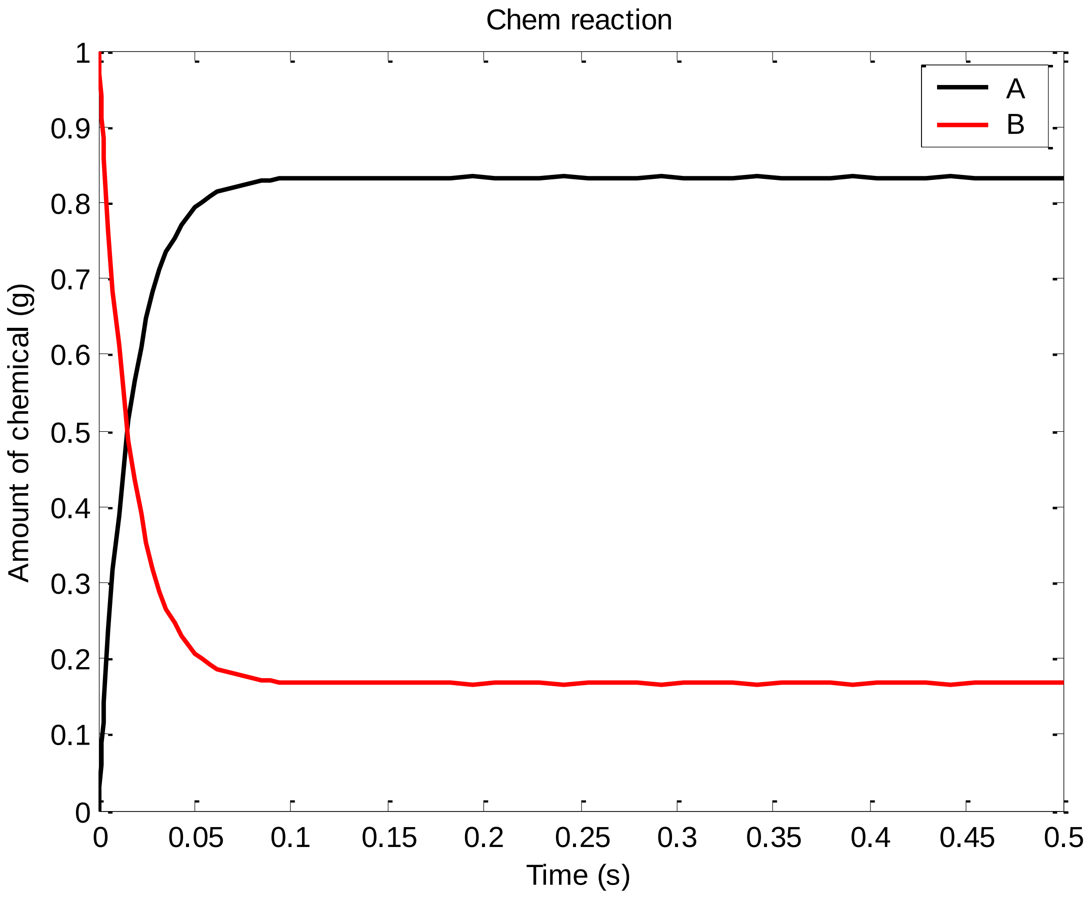


ODE Function: viewing results

- To solve and plot the ODEs on the previous slide:
  - [t,y]=ode45('chem',[0 0.5],[0 1]);
    - assumes that only chemical B exists initially
  - plot(t,y(:,1),'k','LineWidth',1.5);
  - hold on;
  - plot(t,y(:,2),'r','LineWidth',1.5);
  - legend('A','B');
  - xlabel('Time (s)');
  - ylabel('Amount of chemical (g)');
  - title('Chem reaction');

ODE Function: viewing results

- The code on the previous slide produces this figure



Higher Order Equations

- Must make into a system of first-order equations to use ODE solvers
- Nonlinear is OK!
- Pendulum example:

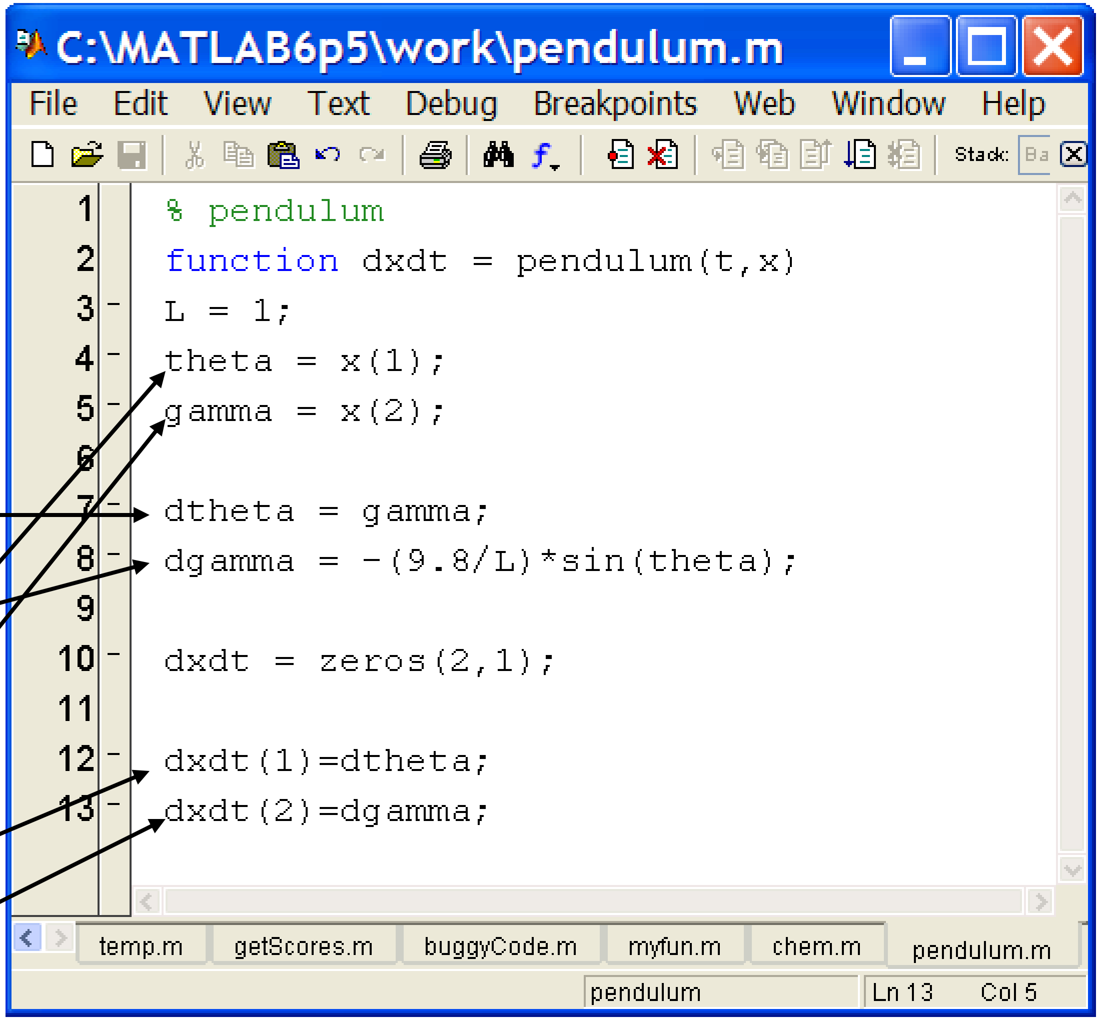
$$\ddot{\theta} + \frac{g}{L}\sin(\theta) = 0$$

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta)$$

$$let \quad \dot{\theta} = \gamma$$

$$\dot{\gamma} = -\frac{g}{L}\sin(\theta)$$

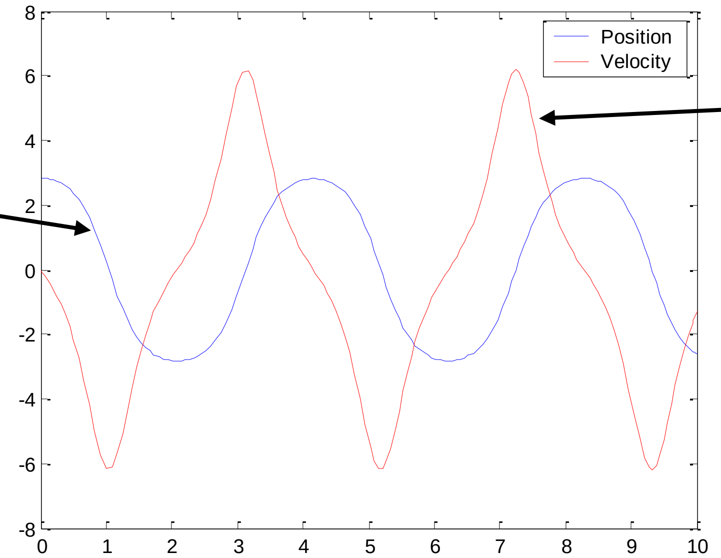
$$\boxed{\begin{aligned} \vec{x} &= \begin{bmatrix} \theta \\ \gamma \end{bmatrix} \\ \frac{d\vec{x}}{dt} &= \begin{bmatrix} \dot{\theta} \\ \dot{\gamma} \end{bmatrix} \end{aligned}}$$



Plotting the Output

- We can solve for the position and velocity of the pendulum:
  - [t,x]=ode45('pendulum',[0 10],[0.9*pi 0]);
    - assume pendulum is almost horizontal
  - plot(t,x(:,1));
  - hold on;
  - plot(t,x(:,2),'r');
  - legend('Position','Velocity');

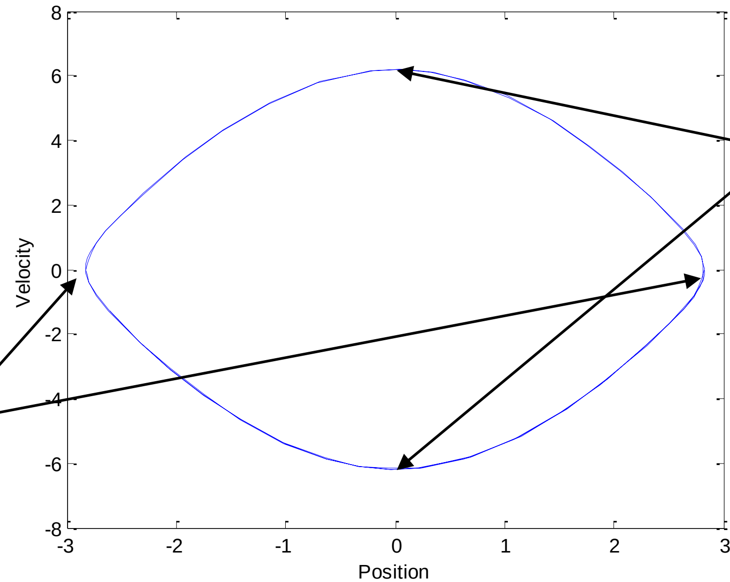
Position in terms of angle (rad)



Velocity (m/s)

Plotting the Output

- Or we can plot in the phase plane:
  - plot(x(:,1),x(:,2));
  - xlabel('Position');
  - yLabel('Velocity');
- The phase plane is just a plot of one variable versus the other:



Velocity is greatest when theta=0

Velocity=0 when theta is the greatest

ODE Solvers: Custom Options

- Matlab's ODE solvers use a variable timestep
- Sometimes a fixed timestep is desirable
  - [t,y]=ode45('chem',[0:0.001:0.5],[0 1]);
    - Specify the timestep by giving a vector of times
    - The function value will be returned at the specified points
    - Fixed timestep is usually slower because function values are interpolated to give values at the desired timepoints

- You can customize the error tolerances using odeset
  - options=odeset('RelTol',1e-6,'AbsTol',1e-10);
  - [t,y]=ode45('chem',[0 0.5],[0 1],options);
    - This guarantees that the error at each step is less than **`RelTol`** times the value at that step, and less than **`AbsTol`**
    - Decreasing error tolerance can considerably slow the solver
    - See doc odeset for a list of options you can customize

Pair-Share Exercise: ODE

- Use **`ode45`** to solve for $y(t)$ on the range t=[0 10], with initial condition $y(0)=10$ and $dy/dt=-t\,y/10$
- Plot the result.

Pair-Share Exercise: ODE

- Use **`ode45`** to solve for $y(t)$ on the range t=[0 10], with initial condition $y(0)=10$ and $dy/dt=-t\,y/10$
- Plot the result.

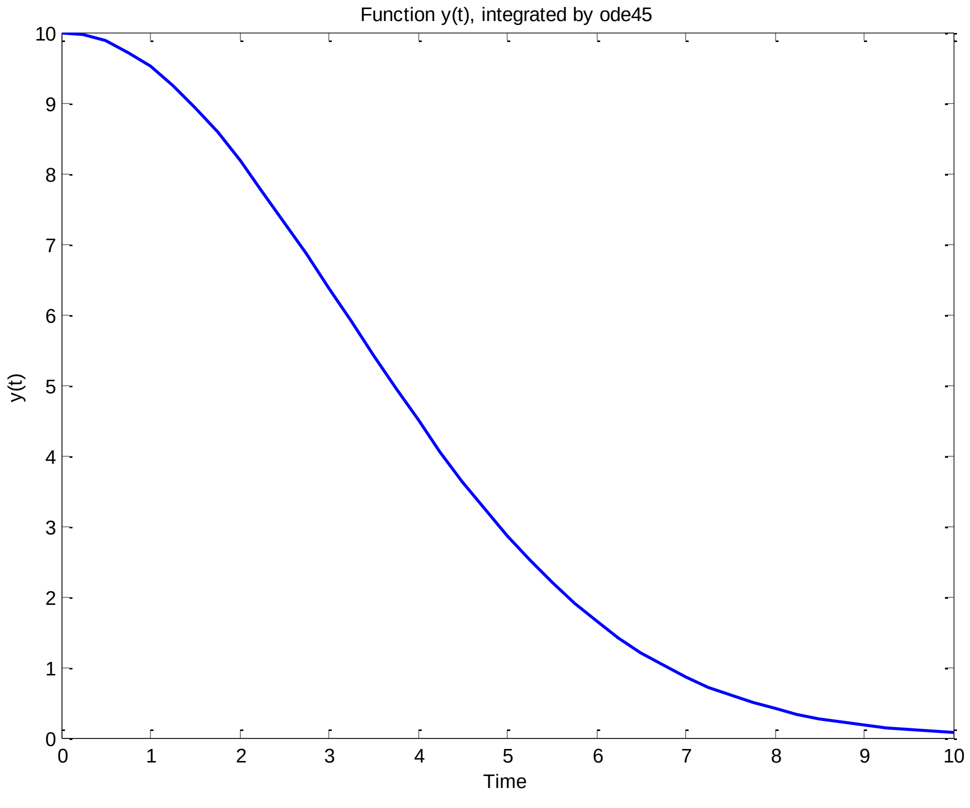
- Make the following function
  - function dydt=odefun(t,y)
  - dydt=-t*y/10;
- Integrate the ODE function and plot the result
  - [t,y]=ode45('odefun',[0 10],10);

- Alternatively, use an anonymous function
  - [t,y]=ode45(@(t,y) –t*y/10,[0 10],10);

- Plot the result
  - plot(t,y);xlabel('Time');ylabel('y(t)');

Exercise: ODE

- The integrated function looks like this:



ODE on Mathworks.com

- From Matlab website: http://www.mathworks.com/support/tech-notes/1500/1510.html#time

Enter this into MATLAB in the following format:

```
function dy = secondode(x,y)
% function to be integrated
dy = zeros(4,1);


dy(1) = y(2);
dy(2) = -3*y(1) -exp(x)*y(4) + exp(2*x);
dy(3) = y(4);
dy(4) = -y(1) -cos(x)*y(2) + sin(x);
```

Note the change of variable from `x` to `t` (it is simply the independent variable).

Now solve the system using ODE45 and the initial conditions `u(0) = 1`, `u'(0) = 2`, `v(0) = 3`, `v'(0) = 4` over the interval from `x = 0` to `x = 3`. The commands you will need to use are:

```
xspan = [0 3];
y0 = [1; 2; 3; 4];
[x, y] = ode45(@secondode, xspan, y0);
```

The values in the first column of y correspond to the values of u for the x values in x. The values in the second column of y correspond to the values of u', and so on.

- More on ode45 commands: http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ode45.html

Case Study Simulation Model

$$Simulate\ the\ following\ system\ with$$

$$tspan = [0\ 3];\ [x_o\ \dot{x}_o\ \phi_o\ \dot{\phi}_o]' = [0\ 0\ 10 * pi/180\ 0]';$$

$$You\ can\ choose\ different\ values\ for\ T\ and\ f$$

$$I_p\ddot{\phi} - mL\ddot{x}\cos\phi = T + mgl\sin\phi$$

$$(m + M)\ddot{x} - mL\cos\phi\ddot{\phi} + mL\sin\phi\dot{\phi}^2 = f$$

Case Study Simulation Model

$$I_p\ddot{\phi} - mL\ddot{x}\cos\phi = T + mgl\sin\phi$$

$$or$$

$$a\ddot{\phi} + b\ddot{x} = c$$

$$where$$

$$a = I_p;\ b = -mL\cos\phi; c = T + mgl\sin\phi$$

$$(m+M)\ddot{x} - mL\cos\phi\ddot{\phi} + mL\sin\phi\dot{\phi}^2 = f$$

$$or$$

$$d\ddot{x} + e\ddot{\phi} = g$$

$$where\, d = (m+M);\ e = -mL\cos\phi;\ \ g = -mL\sin\phi\dot{\phi}^2 + f$$

$$Solve\ \ for\ \ \ddot{x}\ \ and\ \ddot{\phi}$$

$$\ddot{x} = \frac{ag - ec}{ad - eb} = \frac{I_p\left(-mL\sin\phi\dot{\phi}^2 + f\right) + mL\cos\phi\left(T + mgl\sin\phi\right)}{I_p\left(m+M\right) + mL\cos\phi\left(-mL\cos\phi\right)}$$

$$\ddot{\phi} = \frac{c}{a} - \frac{b}{a}\left[\frac{ag - ec}{ad - eb}\right] = \frac{T + mgl\sin\phi}{I_p} + \frac{mL\cos\phi}{I_p}\left[\frac{I_p\left(-mL\sin\phi\dot{\phi}^2 + f\right) + mL\cos\phi\left(T + mgl\sin\phi\right)}{I_p\left(m+M\right) + mL\cos\phi\left(-mL\cos\phi\right)}\right]$$

Matlab Simulation


```
tspan = [0 3]; %simulation time vector

%State vector = y = [x ẋ φ φ̇]'
y0 = [0 0 10*pi/180 0]';
[t, y] = ode45(@secondode, tspan, y0);

x = y(:,1); xdot = y(:,2); phi = y(:,3); phidot = y(:,4);
%Plot state vector against time
figure
subplot(221)
plot(t, x)
ylabel('x,[m]')
xlabel('t,[s]')

subplot(222)
plot(t, xdot)
ylabel('xdot,[m/s]')
xlabel('t,[s]')

subplot(223)
plot(t, phi)
ylabel('phi,[m]')
xlabel('t,[s]')

subplot(224)
plot(t, phidot)
ylabel('phidot,[rad/s]')
```

```
% function to be integrated
function dy = secondode(t, y)
dy = zeros(4,1);

% Define parameters
I_p = 47.5; %kg-m^2
L = 7/8; %m
m = 40; %kg
M = 100; %kg
g = 9.81; %m/s^2
T = 0;
f = 0;
a = I_p;
b = -m*L*cos(y(3));
c = T + m*g*l*sin(y(3));
d = (m + M);
e = -m*L*cos(y(3));
g = -m*L*sin(y(3))*y(4)*y(4) + f;

%State vector = y = [x ẋ φ φ̇]' = [y(1) y(2) y(3) y(4)]'
dy(1) = y(2);
dy(2) = (a*g - e*c)/(a*d - e*b);
dy(3) = y(4);
dy(4) = c/a - (b/a)*(a*g - e*c)/(a*d - e*b);
```

References

- Woods, R. L., and Lawrence, K., Modeling and Simulation of Dynamic Systems, Prentice Hall, 1997.
- www.mathworks.com
- Dorf, R., Bishop, R., Modern Control System, Eleventh Edition, Prentice Hall
- Lecture 3 : Solving Equations and Curve Fitting, 6.094 - Introduction to programming in MATLAB, by Danilo Šćepanović, IAP 2010 Course, MIT