

---

# CHAPTER 2

---

## MATHEMATICAL MODELS OF SYSTEMS

### 2.1 Differential Equations of Physical Systems

The dynamic performance of physical systems is obtained by utilizing the physical laws of mechanical, electrical, fluid and thermodynamic systems. We generally model physical systems with linear differential equations with constant coefficients when possible. Other models can be derived from more general differential equations.

### 2.2 Numerical Solution

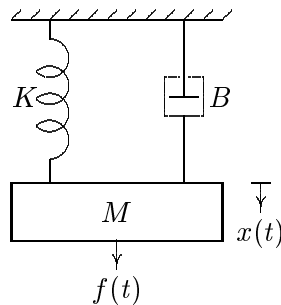
Analytical solutions of linear time-invariant equations are obtained through the Laplace transform and its inversion. There are other techniques which use the state transition matrix  $\phi(t)$  to provide a solution. These analytical methods are normally restricted to linear differential equations with constant coefficients. Numerical techniques solve differential equations directly in the time domain; they apply not only to linear time-invariant but also to nonlinear and time varying differential equations. The value of the function obtained at any step is an approximation of the value which would have been obtained analytically, whereas the analytical solution is exact. However, an analytical solution may be difficult, time consuming or even impossible to find.

*MATLAB* provides two functions for numerical solutions of differential equations employing the Runge-Kutta method. These are **ode23** and **ode45**, based on the

Fehlberg second and third order pair of formulas for medium accuracy and fourth and fifth order pair for high accuracy. The  $n$ th-order differential equation must be transformed into  $n$  first order differential equations and must be placed in an M-file that returns the state derivatives of the equations. The following examples demonstrate the use of these functions.

### Example 2.1

Consider the simple mechanical system of Figure 2.1. Three forces influence the motion of the mass, namely, the applied force, the frictional force, and the spring force.



**FIGURE 2.1**  
Mechanical translational system.

Applying Newton's law of motion, the force equation of the system is

$$M \frac{d^2 x}{dt^2} + B \frac{dx}{dt} + Kx = f(t)$$

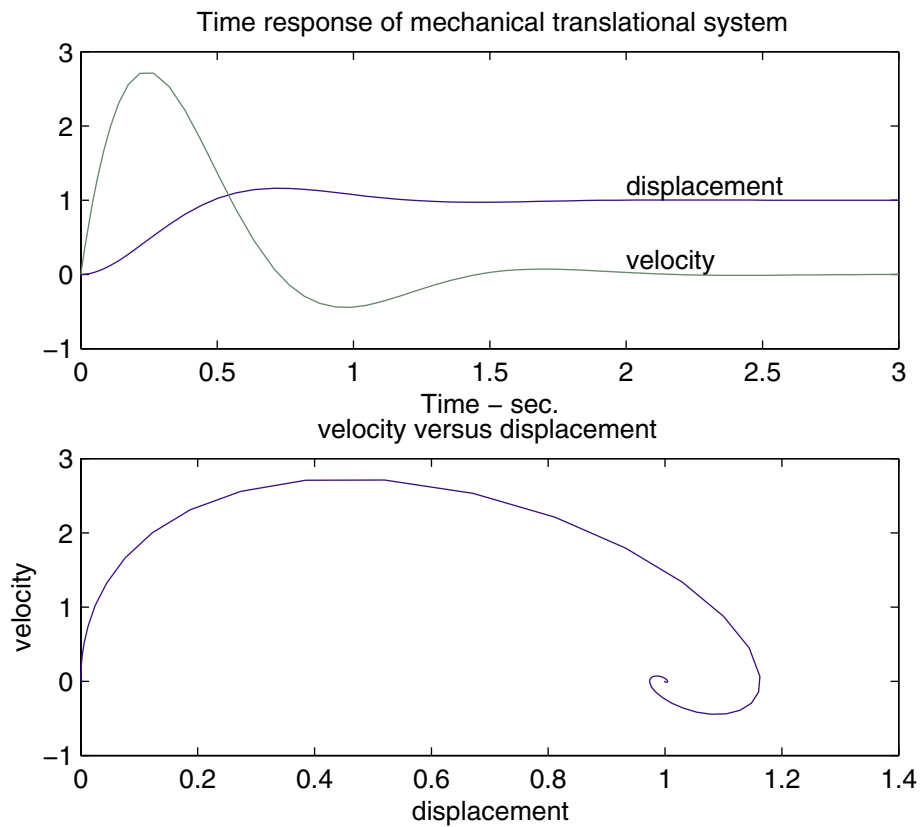
Let  $x_1 = x$  and  $x_2 = \frac{dx}{dt}$ , then

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= \frac{1}{M}[f(t) - Bx_2 - Kx_1] \end{aligned}$$

With the system initially at rest, a force of 25 Newton is applied at time  $t = 0$ . Assume that the mass  $M = 1$  Kg, frictional coefficient  $B = 5$  N/m/sec., and the spring constant  $K = 25$  N/m. The above equations are defined in an M-file **mechsys.m** as follows:

```
function xdot = mechsys(t,x);% returns the state derivatives
F = 25;                      % Step input
M = 1; B = 5; K = 25;
xdot = [x(2) ; 1/M*( F - B*x(2) - K*x(1) ) ];
```

The following M-file, **ch2ex01.m** uses **ode23** to simulate the system over an interval of 0 to 3 sec., with zero initial conditions.



**FIGURE 2.2**  
Response of the mechanical system of Example 2.1.

```

tspan = [0, 3]      ;           % time interval
x0 = [0, 0];       % initial conditions
[t,x] = ode23('mechsys', tspan, x0);
subplot(2,1,1),plot(t,x)
title('Time response of mechanical translational system')
xlabel('Time - sec.')
text(2,1.2,'displacement')
text(2,.2,'velocity')

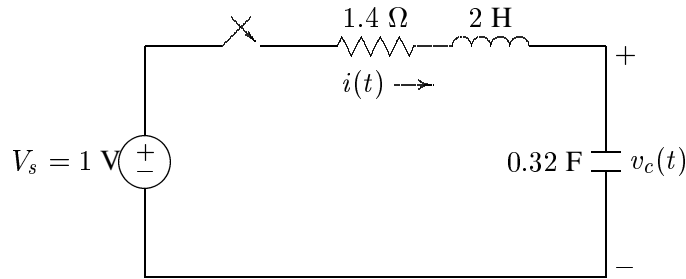
d= x(:,1);  v = x(:,2);
subplot(2,1,2), plot(d, v)
title('velocity versus displacement ')
xlabel('displacement')
ylabel('velocity')
subplot(111)

```

Results of the simulation are shown in Figure 2.2.

**Example 2.2**

The circuit elements in Figure 2.3 are  $R = 1.4\Omega$ ,  $L = 2\text{H}$ , and  $C = 0.32\text{F}$ , the initial inductor current is zero, and the initial capacitor voltage is .5 volts. A step voltage of 1 volt is applied at time  $t = 0$ . Determine  $i(t)$  and  $v(t)$  over the range  $0 < t < 15$  sec. Also, obtain a plot of current versus capacitor voltage.



**FIGURE 2.3**  
RLC circuit for time-domain solution example.

Applying KVL

$$Ri + L \frac{di}{dt} + v_c = V_s$$

and

$$i = C \frac{dv_c}{dt}$$

Let

$$x_1 = v_c$$

and

$$x_2 = i$$

then

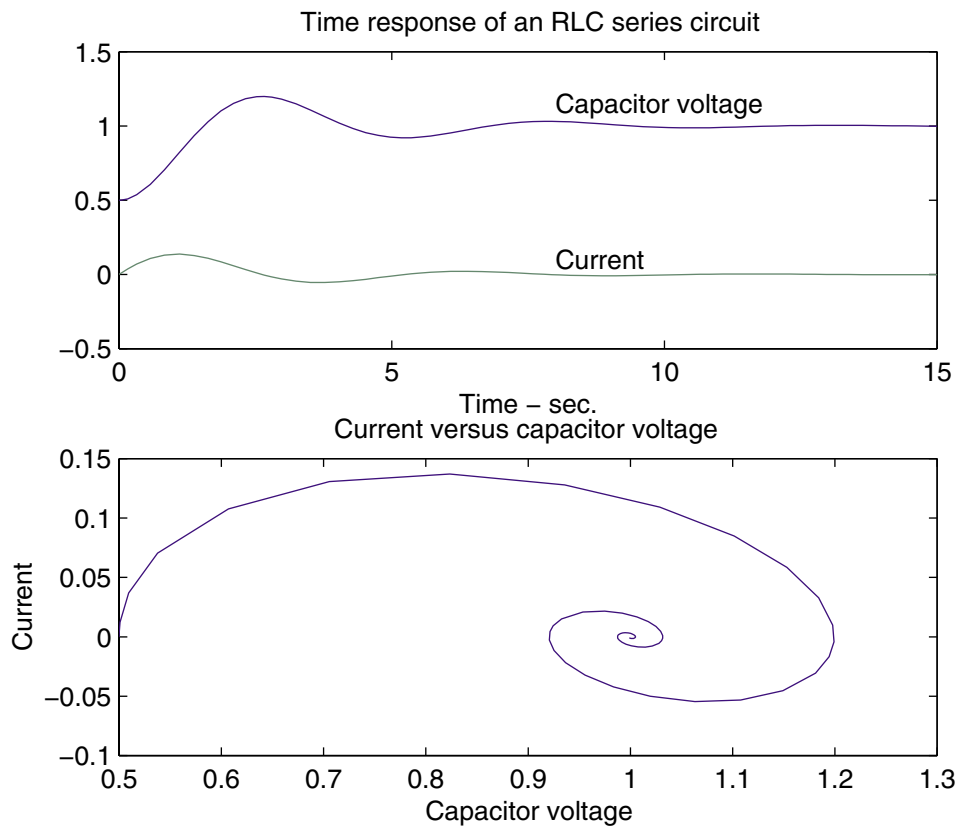
$$\dot{x}_1 = \frac{1}{C}x_2$$

and

$$\dot{x}_2 = \frac{1}{L}(V_s - x_1 - Rx_2)$$

The above equations are defined in an M-file **electsys.m** as follows:

```
function xdot = electsys(t,x);
                                % returns the state derivatives
                                % Step input
V = 1;
R = 1.4; L = 2; C = 0.32;
xdot = [x(2)/C ; 1/L*( V - x(1) - R*x(2) ) ];
```



**FIGURE 2.4**  
Response of the series RLC circuit of Example 2.2.

The following M-file, **ch2ex02.m**, uses **ode23** to simulate the system over an interval of 0 to 15 sec.

```
x0 = [0.5, 0];          % initial conditions
tspan=[0, 15];        % time interval
[t,x] = ode23('electsys',tspan, x0);
subplot(2, 1, 1),plot(t,x)
title('Time response of an RLC series circuit')
xlabel('Time - sec.')
text(8,1.15, 'Capacitor voltage')
text(8, .1, 'Current')
vc= x(:,1);   i = x(:,2);
subplot(2, 1, 2),plot(vc, i)
title('Current versus capacitor voltage ')
xlabel('Capacitor voltage')
ylabel('Current'), subplot(111)
```

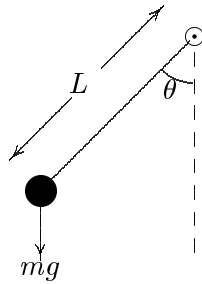
Results of the simulation are shown in Figure 2.4.

### 2.3 Nonlinear Systems

A great majority of physical systems are linear within some range of the variables. However, all systems ultimately become nonlinear as the ranges are increased without limit. For the nonlinear systems, the principle of superposition does not apply. **ode23** and **ode45** simplify the task of solving a set of nonlinear differential equations as demonstrated in Example 2.3.

#### Example 2.3

Consider the simple pendulum illustrated in Figure 2.5 where a weight of  $W = mg$  kg is hung from a support by a weightless rod of length  $L$  meters. While usually approximated by a linear differential equation, the system really is nonlinear and includes viscous damping with a damping coefficient of  $B$  kg/m/sec.



**FIGURE 2.5**  
Pendulum oscillator.

If  $\theta$  in radians is the angle of deflection of the rod, the velocity of the weight at the end will be  $L\dot{\theta}$  and the tangential force acting to increase the angle  $\theta$  can be written:

$$F_T = -W \sin \theta - BL\dot{\theta}$$

From Newton's law

$$F_T = mL\ddot{\theta}$$

Combining the two equations for the force, we get:

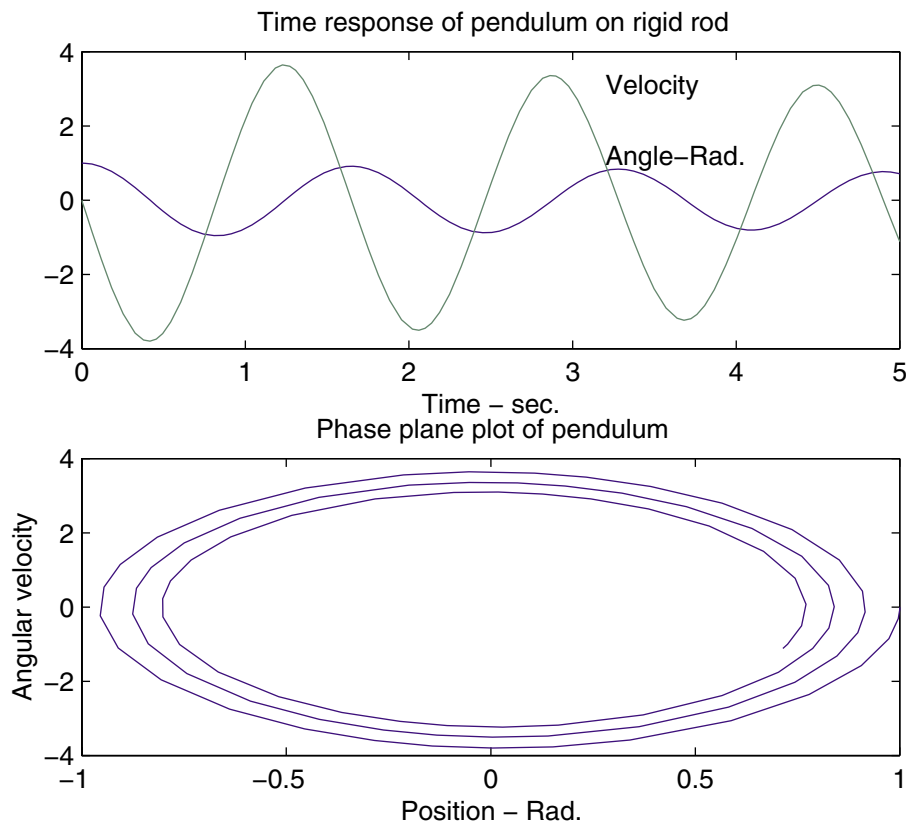
$$mL\ddot{\theta} + BL\dot{\theta} + W \sin \theta = 0$$

Let  $x_1 = \theta$  and  $x_2 = \dot{\theta}$  (angular velocity), then

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

$$\dot{x}_2 = -\frac{B}{m}x_2 - \frac{W}{mL} \sin x_1$$

The above equations are defined in an M-file **pendulum.m** as follows:



**FIGURE 2.6**  
Response of the pendulum described in Example 2.3.

```
function xdot = pendulum(t,x);%returns the state derivatives
W = 2; L = .6; B = 0.02; g = 9.81; m = W/g;
xdot = [x(2) ; -B/m*x(2)-W/(m*L)*sin(x(1)) ];
```

The following M-file, **ch2ex03.m**, uses **ode23** to simulate the system over an interval of 0 to 5 sec.

```
tspan = [0, 5]; % time interval
x0 = [1, 0]; % initial conditions
[t,x] = ode23('pendulum', tspan, x0);
subplot(2,1,1),plot(t,x)
title('Time response of pendulum on rigid rod')
xlabel('Time - sec.')
text(3.2,3.1,'Velocity'), text(3.2,1.2,'Angle-Rad.')
th= x(:,1); w = x(:,2);
subplot(2,1,2),plot(th, w)
title('Phase plane plot of pendulum')
xlabel('Position - Rad. '), ylabel('Angular velocity')
```

Results of the simulation are shown in Figure 2.6.

## 2.4 Linearization

Nonlinear systems are often linearized assuming small signal conditions. The nonlinear differential equation describing the motion of the pendulum in Example 2.3 can be linearized if the initial angle of deflection is small. When  $\theta = \theta_0 + \Delta\theta$ , the pendulum equation can be written as

$$mL(\ddot{\theta} + \Delta\ddot{\theta}) + BL(\dot{\theta} + \Delta\dot{\theta}) + W \sin(\theta + \Delta\theta) = 0 \quad (2.1)$$

For small  $\Delta\theta$  assuming  $\sin \Delta\theta \simeq 0$ ,  $\cos \Delta\theta \simeq 1$  and expanding the sine term yields the following linear differential equation.

$$mL\Delta\ddot{\theta} + BL\Delta\dot{\theta} + W\Delta\theta = 0 \quad (2.2)$$

It is left as an exercise to show that the above linearized equation will yield approximately the same response as long as  $\Delta\theta$  is small.

## 2.5 Transfer Function

The transfer function of a linear, time-invariant, differential equation system is defined as the ratio of the Laplace transform of the output variable to the Laplace transform of the input variable, with all initial conditions assumed to be zero. Although the transfer function can be used only for linear systems, it yields more intuitive information than the differential equation. The characteristic equation is obtained by setting the denominator polynomials of the transfer function to zero. The roots of the denominator are the system poles, and the roots of the numerator are the system zeros. The system transfer function can then be specified to within a constant by specifying the system poles and zeros. The constant, usually denoted by  $K$ , is the system gain factor. The transfer function model enables us to change system parameters and rapidly sense the effect of these changes on the system response. The transfer function is also useful in modeling the interconnection of subsystems by forming a block diagram representation. The time response of a system is obtained by the inverse transform of the  $s$ -domain response. This usually requires expansion of the rational function using partial fractions.

In this section, several examples are presented to demonstrate the use of *MATLAB* in finding the roots of the characteristic equation, poles and zeros of a transfer function, partial fraction expansion, and transformation of poles and zeros to transfer function.

The Control System Toolbox function `sys = tf(num, den)` creates a continuous-time transfer function. The output `sys` is a `tf` object. For SISO models, `num` and `den` are row vectors listing the numerator and denominator coefficients in descending powers of  $s$ . For example, the commands

```
num=[1 4]; den=[1 2 10];
sys=tf(num, den)
```

results in

```

Transfer function:
      s + 4
-----
s^2 + 2 s + 10

```

### 2.5.1 Polynomial Roots and Characteristic Polynomial

If  $p$  is a row vector containing the coefficients of a polynomial, **roots(p)** returns a column vector whose elements are the roots of the polynomial. If  $r$  is a column vector containing the roots of a polynomial, **poly(r)** returns a row vector whose elements are the coefficients of the polynomial.

#### Example 2.3

Find the roots of the following polynomial.

$$s^6 + 9s^5 + 31.25s^4 + 61.25s^3 + 67.75s^2 + 14.75s + 15$$

The polynomial coefficients are entered in a row vector in descending powers. The roots are found using **roots**.

```

p = [ 1  9  31.25  61.25  67.75  14.75  15 ]
r = roots(p)

```

The polynomial roots are obtained in column vector

```

r =
-4.0000
-3.0000
-1.0000 + 2.0000i
-1.0000 - 2.0000i
 0.0000 + 0.5000i
 0.0000 - 0.5000i

```

#### Example 2.4

The roots of a polynomial are  $-1$ ,  $-2$ ,  $-3 \pm j4$ . Determine the polynomial equation.

In order to enter a complex number, it is first necessary to generate a complex unit. The roots are then entered in a column vector. The polynomial equation is obtained using **poly** as follows:

```

i = sqrt(-1)
r = [-1  -2  -3+4*i  -3-4*i ]
p = poly(r)

```

The coefficients of the polynomial equation are obtained in a row vector.

```

p =
 1  9  45  87  50

```

Therefore, the polynomial equation is

$$s^4 + 9s^3 + 45s^2 + 87s + 50 = 0$$

**Example 2.5**

Determine the roots of the characteristic equation of the following matrix.

$$A = \begin{bmatrix} 0 & 1 & -1 \\ -6 & -11 & 6 \\ -6 & -11 & 5 \end{bmatrix}$$

The characteristic equation of the matrix is found by **poly**, and the roots of this equation are found by **roots**.

```
A = [ 0 1 -1; -6 -11 6; -6 -11 5];
p = poly(A)
r = roots(p)
```

The result is as follows:

```
p =
    1.0000    6.0000   11.0000    6.0000
r =
   -3.0000
   -2.0000
   -1.0000
```

**2.5.2 Poles and Zeros of a Transfer Function**

**tf2zp** finds the zeros, poles and gains of a transfer function.

**Example 2.6**

Find the poles and zeros of the following transfer function:

$$H(s) = \frac{s^3 + 11s^2 + 30s}{s^4 + 9s^3 + 45s^2 + 87s + 50}$$

```
num = [ 1 11 30 0];
den = [ 1 9 45 87 50];
[z,p,k] = tf2zp(num,den)
```

The zeros, poles and gains are:

```
z =
   -6.0000
   -5.0000
    0.0000
    inf
p =
   -3.0000    +4.0000i
   -3.0000    -4.0000i
   -2.0000
   -1.0000
k =
    1
```

Therefore

$$H(s) = \frac{s(s+5)(s+6)}{(s+1)(s+2)(s+3+j4)(s+3-j4)}$$

**zp2tf** forms transfer function polynomials from the zeros, poles and gains of systems.

### Example 2.7

A system has zeros at  $-6$ ,  $-5$ ,  $0$ , poles at  $-3 \pm j4$ ,  $-2$ ,  $-1$ , and a gain of 1. Determine the system transfer function.

```
z = [-6; -5; 0]; k=1;
i = sqrt(-1);
p = [-3+4*i; -3-4*i; -2; -1];
[num, den] = zp2tf(z,p,k)
```

The above program results in

```
num =
    1   11   30   0
den =
    1    9   45   87   50
```

which yields the following transfer function

$$H(s) = \frac{s^3 + 11s^2 + 30s}{s^4 + 9s^3 + 45s^2 + 87s + 50}$$

### 2.5.3 Partial-Fraction Expansion

**[r,p,k] = residue[b,a]** finds the residues, poles, and direct terms of a partial fraction expansion of the ratio of two polynomials

$$\frac{P(s)}{Q(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} \quad (2.3)$$

Vectors **b** and **a** specify the coefficients of the polynomials in descending powers of  $s$ . The residues are returned in column vector **r**, the pole locations in column vector **p**, and the direct terms in row vector **k**.

### Example 2.8

Determine the partial fraction expansion for

$$F(s) = \frac{2s^3 + 9s + 1}{s^3 + s^2 + 4s + 4}$$

```

num = [ 2  0  9  1];
den = [ 1  1  4  4];
[res, poles ,k] = residue(num, den)

```

The result is as follows

```

res =
    0.0000    -0.2500i
    0.0000    +0.2500i
   -2.0000
poles =
    0.0000    +2.0000i
    0.0000    -2.0000i
   -1.0000
K =
    2.0000

```

Therefore the partial fraction expansion is

$$2 + \frac{-2}{s+1} + \frac{j0.25}{s+j2} + \frac{-j0.25}{s-j2} = 2 + \frac{-2}{s+1} + \frac{1}{s^2+4}$$

**[num, den] = residue(res, poles, K)** converts the partial fraction expansion back to the polynomial  $P(s)/Q(s)$ .