

Chapter 2. MATLAB® Fundamentals

E2.1: The motion of a piston in an internal combustion engine is shown in Figures 2.10.

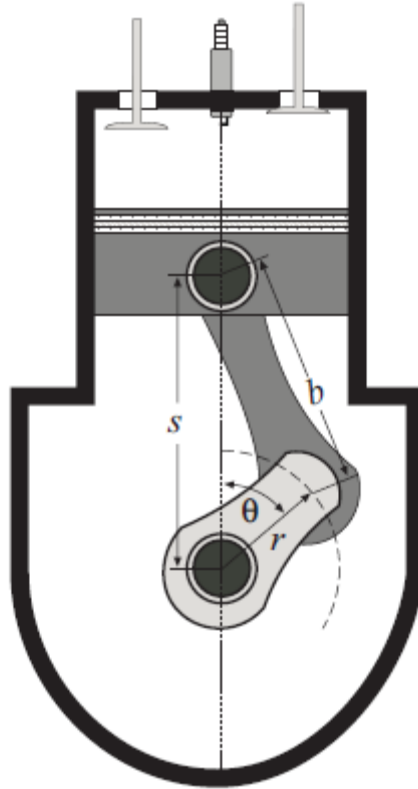


Fig. 2.10.

The piston's position, s , as seen from the crank shaft center is determined to be

$$s(t) = r \cos(2\pi \omega t) + \sqrt{b^2 - r^2 \sin^2(2\pi \omega t)} \quad (2.6)$$

where

b = the length of the piston rod.

r = the radius of the crankshaft.

ω = rotational speed of the crankshaft in revolutions per second.

Develop a MATLAB program that determines s vs. t for $0 \leq t \leq 0.01$ seconds. Use 20 subdivisions on the t domain. Take $r = 9 \text{ cm}$, $\omega = 100$ revolutions per second and $b = 14 \text{ cm}$. Create a table of s vs. t and print the results to both the screen and to a file.

The program follows:

```
% E2_1.m

% This project involves determining the position, s vs. time, t
% of a piston in an engine and tabulates the results for
% 0 <= t <= 0.01 s. 20 sub-divisions on the t domain are to be used.
% Problem parameters are: r=9 cm, omega = 100 rev per second
% and b=14 cm.

clear; clc;

r=9; omega=100; b=14;

tt=0:0.0005:0.01;

fo=fopen('output.txt','w');

fprintf(fo,'  t(s)          s(cm)    \n');

fprintf(fo,'----- \n');

fprintf('  t(s)          s(cm)    \n');

fprintf('----- \n');

for i=1:length(tt)

    t=tt(i);

    arg1=2*pi*omega*t;

    arg2=b^2-r^2*(sin(arg1))^2;

    s(i)=r*cos(arg1)+sqrt(arg2);

    fprintf(fo,'%7.4f    %10.4f  \n',t,s(i));

    fprintf('%7.4f    %10.4f  \n',t,s(i));

end
```

Program results

t (s)	s (cm)
0.0000	23.0000
0.0005	22.2805
0.0010	20.2432
0.0015	17.2477
0.0020	13.8597
0.0025	10.7238
0.0030	8.2974
0.0035	6.6676
0.0040	5.6809
0.0045	5.1615
0.0050	5.0000
0.0055	5.1615
0.0060	5.6809
0.0065	6.6676
0.0070	8.2974
0.0075	10.7238
0.0080	13.8597
0.0085	17.2477
0.0090	20.2432
0.0095	22.2805
0.0100	23.0000

E2.2: The position, y , of a mass in a mass-spring-dashpot system (for a derivation of Equation (2.7) see Project P2.5) is given by

$$y = \exp\left(-\frac{c}{2m}t\right) \left\{ A \cos\left(\sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} t\right) + B \sin\left(\sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} t\right) \right\} \quad (2.7)$$

Take

$$m = 25.0 \text{ kg.}$$

$$c = \text{damping factor} = 5.0 \text{ N}\cdot\text{s}/\text{m.}$$

$$k = \text{spring constant} = 200.0 \text{ N}/\text{m.}$$

$$A = 5.0 \text{ m.}$$

$$B = 0.25 \text{ m.}$$

- a. Determine $y(t)$ for $0 \leq t \leq 10$ seconds in steps of 0.1 seconds.

- b. Create a Table of y vs. t every 1 second and print the results both to the screen and to a file.

The program follows:

```
% E2_2.m

% This program determines the position, y, of a mass connected to a
% mass-spring-dashpot system. The parameters of the system are:
% m = 25.0 kg, c = 5.0 N-s/m, k = 200.0 N/m, A = 5.0 m, B = 0.25 m.
% The governing equations is  $y = \exp(-ct/2m)(A\cos(\arg*t) + B\sin(\arg*t))$ .

clear; clc;

m=25.0; c=5.0; k=200.0; A=5.0; B=0.25;

arg=sqrt(k/m-(c/2/m)^2);

tt=0:0.1:10;

fo=fopen('output.txt','w');

fprintf(fo,' t(s)          y(m)  \n');

fprintf(' t(s)          y(m)  \n');

fprintf(fo,'----- \n');

fprintf('----- \n');

for i=1:length(tt)

    t=tt(i);

    y(i)=exp(-c*t/2/m)*(A*cos(arg*t)+B*sin(arg*t));

end

for i=1:10:length(tt)

    fprintf('%6.2f      %10.4f  \n',tt(i),y(i));

    fprintf(fo,'%6.2f      %10.4f  \n',tt(i),y(i));

end

-----
```

Program Results:

$t(s)$	$y(m)$
0.00	5.0000
1.00	-4.2316
2.00	3.1875
3.00	-2.0202
4.00	0.8663
5.00	0.1634
6.00	-0.9904
7.00	1.5703
8.00	-1.8912
9.00	1.9684
10.00	-1.8386

E2.3. A basketball player shoots the ball when he is 6 m from the center of the hoop as shown Figure 2.11.

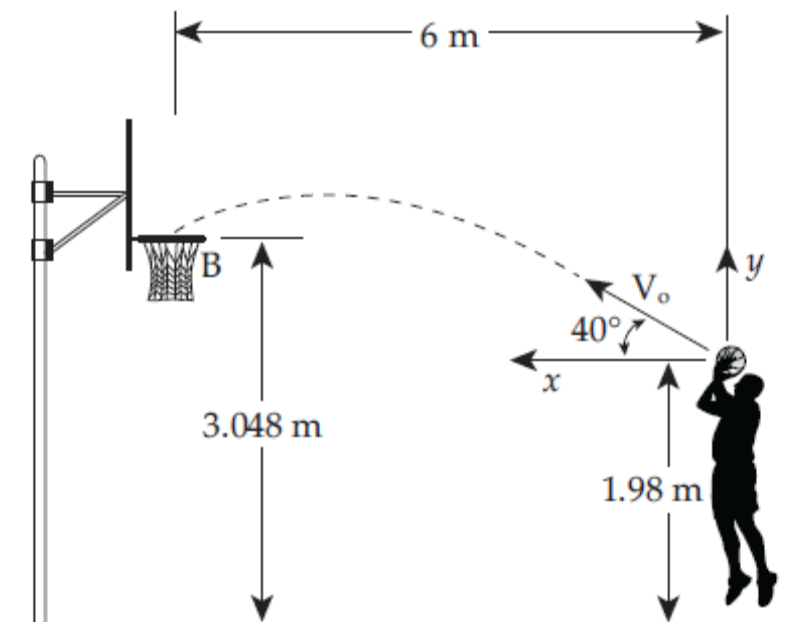


Figure 2.11.

The ball is released at a velocity, V_o , and makes angle of 40° with the horizontal. We will neglect the drag on the basketball in this analysis.

Using Newton's Second Law, which is a vector equation, is

$$\sum \vec{F}_i = m \vec{a} \quad (2.8)$$

The x component of Equation (2.8) is

$$\sum F_{x,i} = 0 = m \frac{dV_x}{dt} \rightarrow V_x = \text{constant} = V_o \cos(\vartheta)$$

$$\frac{dx}{dt} = V_o \cos(\vartheta) \rightarrow x = V_o \cos(\vartheta)t + c_1, \quad x(0) = 0 \rightarrow c_1 = 0$$

Thus,

$$x = V_o \cos(\vartheta)t \quad (2.9)$$

The y component of Equation (2.8) is

$$\sum F_{y,i} = -W = \frac{W}{g} \frac{dV_y}{dt} \rightarrow V_y = -gt + c_2 \rightarrow V_y(0) = V_o \sin(\vartheta) = c_2$$

Thus,

$$V_y = \frac{dy}{dt} = -gt + V_o \sin(\vartheta) \rightarrow y = -g \frac{t^2}{2} + V_o \sin(\vartheta)t + c_3$$

$$y(0) = 0 \rightarrow c_3 = 0$$

$$y = V_o \sin(\vartheta)t - \frac{g}{2}t^2 \quad (2.10)$$

Use the following parameters: $x_f = 6.0 \text{ m}$, $y_f = 3.048 - 1.98 \text{ m}$, and $\vartheta = 40^\circ$.

- Determine the time, t_f , that it takes for the ball to reach the center of the hoop. Time, t equals zero when the ball leaves the player's hands.
- Determine the velocity, V_o , that will result in the ball reaching the center of the hoop.
- Create a table consisting of t, x, y for $0 \leq t \leq t_f$ in steps of 0.01 s.

Solution:

a. We first need to determine t_f . Rewrite Equations (2.9) and (2.10) in the following form:

$$x = V_o \cos(\vartheta) t, \quad (\text{E2.3a})$$

$$y + \frac{g t^2}{2} = V_o \sin(\vartheta) t \quad (\text{E2.3b})$$

We know the final position of $(x, y) = (x_f, y_f)$ at time t_f , so solve Equation (E2.3a) for V_o

and substitute V_o into Equation (E2.3b), replace $\sin(\theta) / \cos(\vartheta)$ with $\tan(\vartheta)$ and

rearrange terms giving

$$t_f^2 = \frac{2}{g} (x_f \tan(\vartheta) - y_f) \quad (\text{E2.3c})$$

b. Having t_f and x_f , we can determine V_o

c. & d. Now sub-divide the t domain and determine $x(t)$ and $y(t)$ to create the desired table.

The program follows:

```
% E2_3.m

% This program involves a basketball player shooting a basketball
% towards the hoop. As the ball is released, it makes an angle
% of 40 degrees with the horizontal. Drag is neglected.

% The problem is to determine the time, tf, it takes the ball to reach
% the center of the hoop and the release velocity, Vo, of the ball.

% A table of ball positions (x,y) as a function of time, t, is created.

% A plot of (x,y) as a function of time is also created.

% The position of the center of the hoop (xf,yf) is known

% The governing equations are xf=Vo*cos(theta)*tf (Eq 1) and
% yf=Vo*sin(theta)*tf-0.5*g*tf^2 (Eq 2)

% Solve (Eq 1) for Vo and substitute Vo into (Eq 2) giving
% yf=xf*tan(theta)-g/2*tf^2 (Eq 3)
```

```

% xf=6-0.28, yf=3.048-1.98

% Substituting xc and yc into (Eq 3) leaves tf as the only unknown.

clear; clc;

theta=40/180*pi;

g=9.81;

xf=6.0;

yf=3.048-1.98;

tfsq=2/g*(xf*tan(theta)-yf);

tf=sqrt(tfsq);

Vo=xf/(cos(theta)*tf);

fo=fopen('output.txt','w');

fprintf(fo,'Exercise E2.3 \n');

fprintf(fo,'Basketball problem \n');

fprintf(fo,'tf=%7.4f s   Vo=%7.4f m/s \n\n',tf,Vo);

fprintf(fo,'   t(s)           x(m)           y(m)   \n');

fprintf(fo,'----- \n');

t=0:0.01:tf;

ne=length(t);

for n=1:ne

    x(n)=Vo*cos(theta)*t(n);

    y(n)=Vo*sin(theta)*t(n)-g/2*t(n)^2;

    fprintf(fo,'%6.2f   %7.4f   %7.4f   \n',t(n),x(n),y(n));

end

t(ne+1)=tf;

x(ne+1)=Vo*cos(theta)*tf;

y(ne+1)=Vo*sin(theta)*tf-g/2*tf^2;

fprintf(fo,'%6.4f   %10.4f   %10.4f   \n',tf,x(ne+1),y(ne+1));

-----

```


Program results.

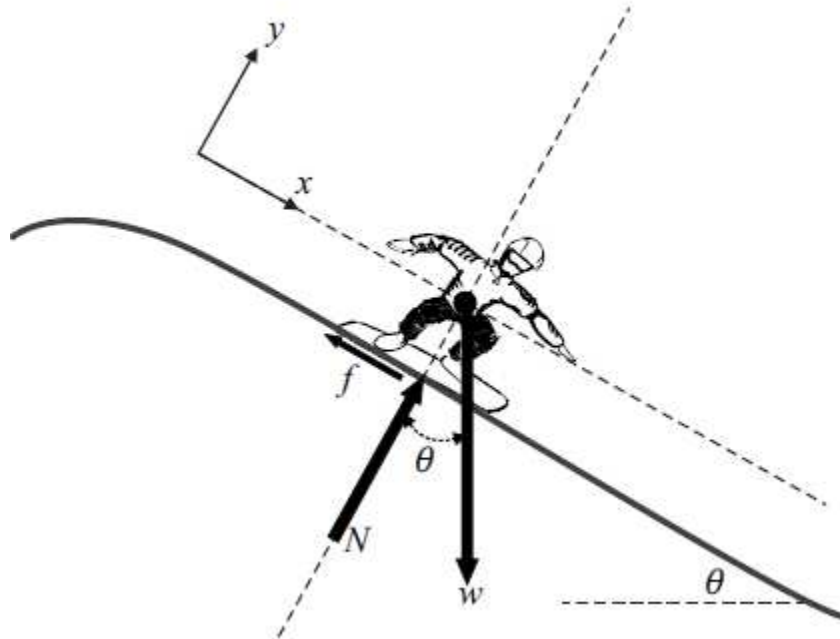
Exercise E2.3

Basketball problem

tf= 0.8993 s Vo= 8.7098 m/s

t(s)	x(m)	y(m)
0.00	0.0000	0.0000
0.01	0.0667	0.0555
0.02	0.1334	0.1100
0.03	0.2002	0.1635
0.04	0.2669	0.2161
0.05	0.3336	0.2677
0.06	0.4003	0.3183
0.07	0.4670	0.3679
0.08	0.5338	0.4165
0.09	0.6005	0.4641
0.10	0.6672	0.5108
.	.	.
.	.	.
0.80	5.3377	1.3396
0.81	5.4044	1.3167
0.82	5.4711	1.2927
0.83	5.5378	1.2677
0.84	5.6046	1.2418
0.85	5.6713	1.2149
0.86	5.7380	1.1870
0.87	5.8047	1.1581
0.88	5.8714	1.1283
0.89	5.9382	1.0975
0.8993	6.0000	1.0680

E2.4. A boy on a snowboard, initially at rest, slides down a smooth hill which makes an angle ϑ with the horizontal (see Figure 2.12).



The boy weighs 650 N and the friction coefficient, μ , between the snowboard and the snow is 0.05. Neglect the drag on the boy and his snowboard. Using Newton's Second Law

$$\sum \vec{F}_i = m \vec{a}$$

The x and y components of the above equation are

$$\sum F_{x,i} = m a_x = m \frac{dV_x}{dt} \quad \text{and} \quad \sum F_{y,i} = N - W \cos \theta = 0$$

$$\sum F_{x,i} = W \sin \theta - f$$

The friction force, $f = \mu N$ and $W = m g$

Thus,

$$\sum F_{x,i} = W \sin \theta - \mu W \cos \theta = \frac{W}{g} \frac{dV_x}{dt} \quad (2.11)$$

Determine an expression for $V_x(t)$, taking $V_x(0) = 0$

$$\frac{dx}{dt} = V_x \quad (2.12)$$

Determine an expression for $x(t)$, taking $x(0) = 0$.

Develop a MATLAB program that calculates position, x , and velocity, V_x as a function of time, t , for $0 \leq t \leq 10$ s in steps of 2 s. Print the results both to the screen and to a file. Take

$\theta = 10^\circ$, $g = 9.81 \text{ m/s}^2$. Do the results look reasonable? If not, what has been neglected?

Solution:

From Equation (2.11),

$$dV_x = g(\sin(\theta) - \mu \cos \theta) dt \rightarrow V_x = g(\sin \theta - \mu \cos \theta)t + C$$

Initial condition: $V_x(0)=0 \rightarrow C = 0$

From Equation (2.12),

$$dx = V_x dt = g(\sin \theta - \mu \cos \theta)t dt$$

$$x = g(\sin \theta - \mu \cos \theta) \frac{t^2}{2} + C$$

Initial condition: $x(0) = 0$ gives $C = 0$

The program follows:

```
% E2_4.m

% This program determines the position, x, and velocity, v, of a
% boy on a snowboard as he slides down a hill.
% x=g(sin(theta)-mu*cos(theta))*t^2/2.
% Vx=g(sin(theta)-mu*cos(theta))*t

clear; clc;

tt=0:0.5:10;

g=9.81; theta=1/18*pi; mu=0.05;

fo=fopen('output.txt','w');
```

```

fprintf(fo,'This program determines the velocity and \n');
fprintf(fo,'position of a snowboarder as he slides \n);
fprintf(fo,'down a hill \n');
fprintf('This program determines the velocity and \n');
fprintf('position of a snowboarder as he slides \n');
fprintf('down a hill \n');
% Table headings
fprintf(fo,'  t(s)      x(m)      velocity(m/s)  \n');
fprintf(fo,'----- \n');
fprintf('  t(s)      x(m)      velocity(m/s)  \n');
fprintf('----- \n');
for i=1:length(tt)
    t=tt(i);
    x(i)=g/2*(sin(theta)-mu*cos(theta))*t^2;
    V(i)=g*(sin(theta)-mu*cos(theta))*t;
    fprintf(fo,'%5.1f    %10.2f    %10.2f  \n',t,x(i),V(i));
    fprintf('%5.1f    %10.2f    %10.2f  \n',t,x(i),V(i));
end

```

Program results:

This program determines the velocity and position of a snowboarder as he slides down a hill

t(s)	x(m)	velocity(m/s)
0.0	0.00	0.00
0.5	0.15	0.61
1.0	0.61	1.22
1.5	1.37	1.83
2.0	2.44	2.44
2.5	3.81	3.05
3.0	5.49	3.66
3.5	7.48	4.27
4.0	9.76	4.88
4.5	12.36	5.49
5.0	15.26	6.10

5.5	18.46	6.71
6.0	21.97	7.32
6.5	25.78	7.93
7.0	29.90	8.54
7.5	34.32	9.15
8.0	39.05	9.76
8.5	44.09	10.37
9.0	49.43	10.98
9.5	55.07	11.59
10.0	61.02	12.20

Running the program for 1 minute, still does not result in the snowboarder reaching a terminal velocity. That is because we neglected drag. This is covered in a later chapter.

E2.5. A small sphere moving through a fluid at a slow velocity will have a drag force acting on it which is described by Stokes' Law,. The sphere could be a dust particle or a raindrop moving in air, or a ball bearing moving in oil. The drag force described by Stokes' Law is

$$D = 6\pi R \mu V \quad (2.13)$$

where

D = drag

R = radius of the sphere.

μ = viscosity of the fluid.

V = the velocity of sphere.

Let's consider a ball bearing dropped in oil (see Figure 2.13) with an initial velocity of zero. We want to determine the velocity and position as a function of time. The forces acting on the ball bearing are the gravitational force, W , buoyancy force, B , and the drag force, D . The buoyancy force is equal the weight of the fluid displaced.

Applying Newton's Second Law to the sphere gives

$$W - B - D = \frac{W}{g} \frac{dV}{dt} \quad (2.14)$$

where

$$W = \text{weight of sphere} = \rho \forall g$$

$$B = \text{buoyancy} = \text{weight of fluid displaced}$$

$$\rho = \text{mass density}$$

$$g = \text{gravitational constant} = 9.81 \text{ m/s}^2$$

$$\forall = \text{volume of sphere} = \frac{4}{3} \pi R^3$$

Terminal velocity (no change in the velocity of the moving sphere) occurs when

$$W - B - D = 0$$

$$W - B - 6 \pi R \mu V_T = 0 \rightarrow W - B = 6 \pi R \mu V_T \quad (2.15)$$

$$W = \rho_{\text{steel}} g \times \frac{4}{3} \pi R^3, \quad B = \rho_{\text{oil}} g \times \frac{4}{3} \pi R^3 \quad (2.16)$$

$$W - B = g (\rho_{\text{steel}} - \rho_{\text{oil}}) \times \frac{4}{3} \pi R^3 \quad (2.17)$$

Substituting Equation 2.15 into Equation 2.14 gives

$$6 \pi R \mu (V_T - V) = \frac{W}{g} \frac{dV}{dt} \quad (2.18)$$

Separating variables and integrating gives

$$\int_0^V \frac{dV'}{(V' - V_T)} = - \int_0^t \frac{6 \pi R \mu g}{W} dt \quad (2.19)$$

Integrating Equation 2.19 gives

$$\left[\ln(V' - V_T) \right]_0^V = \ln(V - V_T) - \ln(-V_T) = \ln\left(\frac{V - V_T}{-V_T}\right) = \ln\left(\frac{V_T - V}{V_T}\right) = -\frac{6\pi R \mu g}{W} t \quad (2.20)$$

Then,

$$1 - \frac{V}{V_T} = e^{-\frac{6\pi R \mu g}{W} t} \quad (2.21)$$

or

$$V = V_T \left(1 - e^{-\frac{6\pi R \mu g}{W} t} \right) \quad (2.22)$$

Take $\mu = 3.85 \text{ (N-s)/m}^2$, $R = 0.01 \text{ m}$, $\rho_{steel} = 7910 \text{ kg/m}^3$, $\rho_{oil} = 899 \text{ kg/m}^3$,

$g = 9.81 \text{ m/s}^2$

Create a MATLAB program that will:

- determine the buoyancy force, B .
- determine the weight of the ball bearing, W .
- determine the terminal velocity, V_T .
- use a while loop to determine V as a function of time, for $0 \leq t \leq 0.05 \text{ s}$ in steps of 0.001 s .
- create and print to a file a table containing t and V . Also print values for W , B , and V_T to the same file.

The program follows:

```
% E2_5.m

% This program determines the terminal velocity of a ball bearing
% dropped in a vat of oil. The program then determines the velocity of
% the ball bearing as a function of time. The forces acting on the
% ball bearing are weight, buoyancy and drag, which is governed by
% Stokes Law.
```

```

clear; clc;

mu = 3.85; R = 0.01; rho_steel = 7910; rho_oil=899; g = 9.81;

B=4/3*pi*R^3*rho_oil*g;
W=4/3*pi*R^3*rho_steel*g;
Vt=(W-B)/(6*pi*R*mu);

fo=fopen('output.txt','w');

fprintf(fo,'This program determines the terminal velocity of \n');
fprintf(fo,'a ball bearing dropped in a vat of oil \n');
fprintf(fo,'The program also determines the ball bearing \n');
fprintf(fo,'velocity as a function of time \n');
fprintf(fo,'\nB=%7.4f N   W=%7.4f N       Vt=%6.4f   N/s \n',B,W,Vt);
fprintf(fo,'\n\n');

fprintf(fo,'   t(s)           V(m/s)       \n');
fprintf(fo,'----- \n');

t=0:0.001:0.05;

arg=6*pi*R*mu*g/W;

i=1;

while i<=length(t)

    V(i)=Vt*(1-exp(-arg*t(i)));

    fprintf(fo,'%10.4f      %10.4f   \n',t(i),V(i));

    i=i+1;

end

```

Program results:

This program determines the terminal velocity of
 a ball bearing dropped in a vat of oil
 The program also determines the ball bearing
 velocity as a function of time

B= 0.0369 N W= 0.3250 N Vt=0.3970 N/s

$t(s)$	$V(m/s)$
0.0000	0.0000
0.0050	0.0412
0.0100	0.0781
0.0150	0.1112
0.0200	0.1408
0.0250	0.1674
0.0300	0.1912
0.0350	0.2125
0.0400	0.2317
0.0450	0.2488
0.0500	0.2642

E2.6. The voltage in a parallel resistance, inductor, capacitor (RLC) circuit (for a derivation of Equation (2.23) see Project P2.7)

is given

$$v = \exp\left(-\frac{1}{2RC}t\right) \left\{ A \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2} t\right) + B \sin\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2} t\right) \right\} \quad (2.23)$$

Take

$$R = 100 \, \Omega$$

$$L = 1.0\text{e-}3 \, \text{H}$$

$$C = 1.0\text{e-}6 \, \text{F}$$

$$A = 6.0 \, \text{V}$$

$$B = -8.9 \, \text{V}$$

- Determine $v(t)$ for $0 \leq t \leq 5.0 \times 10^{-4}$ seconds. Use 100 sub-divisions on the time domain.
- Print out a Table of v vs. t every fourth sub-division.

The program follows:

```

% E2_6.m

% This program determines the voltage, v, in a parallel (RCL) circuit ,

% The parameters of the system are:

% R=100 ohm, L = 1.0e-3 henry, C = 1.0e-6 farad, A = 6.0 V, B = -8.9V.

% The governing equation is  $v = \exp(-t/2RC)(A \cos(\arg*t) + B \sin(\arg*t))$ .

clear; clc;

R=100; L=1.0e-3; C=1.0e-6; A=6.0; B=-8.9; dt=5.0e-6;

arg=sqrt(1/L/c - 1/(2*R*C)^2);

fo=fopen('output.txt','w');

fprintf(fo,' t(s)          v(m)  \n');

fprintf(' t(s)          y(m)  \n');

fprintf(fo,'----- \n');

fprintf('----- \n');

for i=1:101

    t(i)=(i-1)*dt;

    v(i)=exp(-t/(2*R*C))*(A*cos(arg*t(i))+B*sin(arg*t(i)));

end

for i=1:4:101

    fprintf('%6.2f      %10.4f  \n',t(i),v(i));

end

```

Program Results:

t(s)	v(V)
0.00e+000	6.00
2.00e-005	-0.30
4.00e-005	-5.36
6.00e-005	-7.62
8.00e-005	-6.80
1.00e-004	-3.74
1.20e-004	0.07
.	.
.	.
4.00e-004	0.90
4.20e-004	0.06
4.40e-004	-0.65

```

4.60e-004      -1.00
4.80e-004      -0.94
5.00e-004      -0.56
>>

```

E2.7. In North America, residential electrical service is usually delivered as two 340 V_{peak-to-peak} sinusoids at 60 Hz with no DC component, each 180° out of phase with each other (see Figure 2.14), referred to as *one-phase* service. For high power appliances (e.g. air conditioners and stoves), the current is drawn from both “legs” of the service.

1. For phase A, use MATLAB’s `sin()` function to create a table of v vs. t , for $0 \leq t \leq 100$ ms in steps of 4 ms.
2. Compute and print to the screen the root-mean-square (RMS) voltage for this waveform by performing the following operations in MATLAB:
 - a. Squaring it.
 - b. Computing the mean value of the squared waveform by averaging it over the 100 ms time interval.
 - c. Taking the square root of the average.

The program follows:

```

% E2_7.m

% This program determines the RMS of a sinusoidal waveform of AC
% voltage. The program also creates a table of the voltage for
% 0<=t<=100 millisec in steps of 4 millisec.

clear; clc;

f=60;

omega=2*pi*f;

ampl=170;

dt=0.001;

```

```

fprintf('  t(s)          vA(V)      \n')

fprintf('----- \n');

for i=1:101

    t(i)=(i-1)*dt;

    vA(i)=ampl*sin(omega*t(i));

    vAsq(i)=vA(i)^2;

end

for i=1:4:101

    fprintf('%6.4f      %10.3f   \n',t(i),vA(i));

end

vAavg=mean(vAsq);

RMS=sqrt(vAavg);

fprintf('\nRMS of vA=%10.3f volts \n',RMS);

```

Program Results:

t(s)	vA(V)
0.0000	0.000
0.0040	169.665
0.0080	21.307
0.0120	-166.989
0.0160	-42.277
0.0200	161.680
0.0240	62.581
0.0280	-153.821
.	.
.	.
0.0800	-161.680
0.0840	42.277
0.0880	166.989
0.0920	-21.307
0.0960	-169.665
0.1000	-0.000
RMS of vA= 119.612 volts	
>>	

E2.8. In the following exercises, carry out the prescribed modifications:

- a. Modify Example 2.5 program to include creating 3 separate plots of

T vs. z , p vs. z and ρ vs. z for $0 \leq z \leq 11000$ m.

The program follows:

```
% E2_8a.m

% This program determines atmospheric properties of temperature, T,
% pressure, p, and density, rho every 1000 m of altitude and prints
% these values to a file in table format.

% The governing equations are  $T = T_0 - \lambda z$ ,  $p = p_0 (1 - \lambda z / T_0)^{ex}$ ,
% where  $ex = g / (\lambda R)$ , and  $\rho = p / (R T)$ .

clear; clc;

To = 288.15; po = 1.01325e5; R = 287.0; g = 9.81; lamda = 0.0065;
z=0:1000:11000;

ex=g/(lamda*R);

fo=fopen('output.txt','w');

fprintf(fo,'Atmospheric Properties \n');

% Table headings
fprintf(fo,' z      T      p      rho \n');
fprintf(fo,' (m)    (K)      (Pa)      (kg/m^3) \n');
fprintf(fo,'----- \n');

% Need to make T, p and rho as vectors
for i=1:length(z)
    T(i)=To-lamda*z(i);
    p(i)=po*(1 -lamda*z(i)/To)^ex;
    rho(i)=p(i)/(R*T(i));

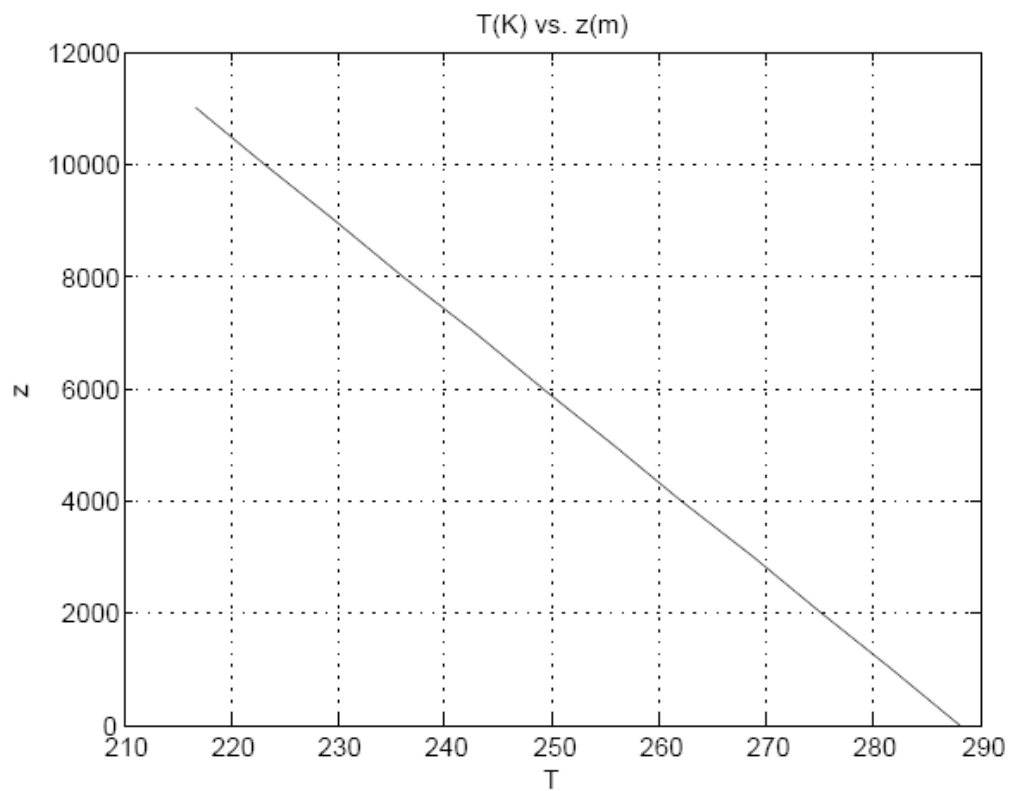
    fprintf(fo,'%6.0f  %8.2f  %10.4e  %10.4f \n',...
        z(i),T(i),p(i),rho(i));
end
```

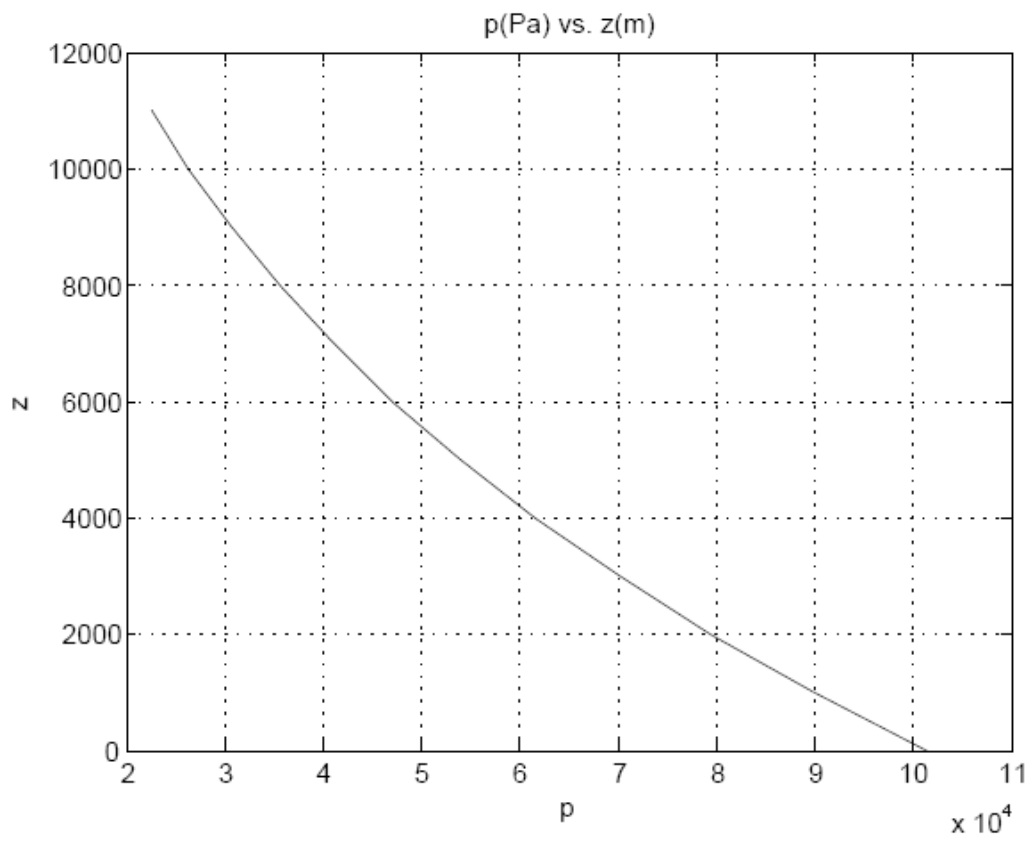
```

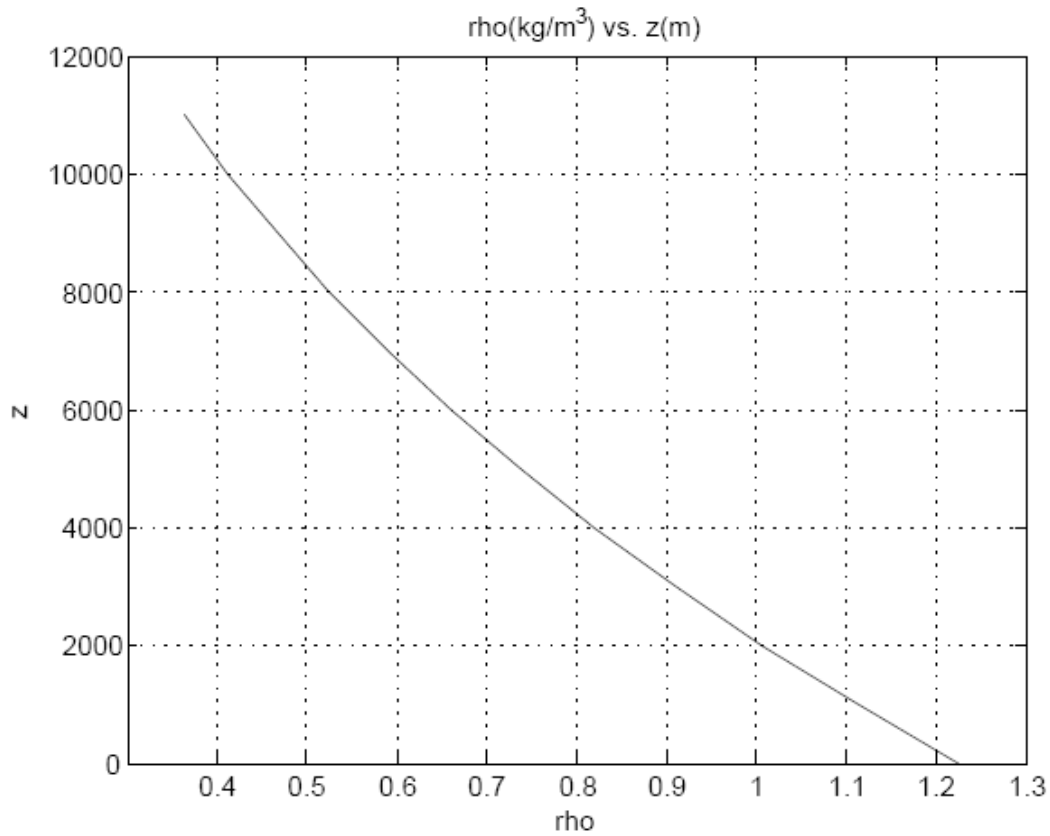
plot(T,z), xlabel('T'), ylabel('z'), title('T(K) vs. z(m)'), grid;
figure;
plot(p,z), xlabel('p'), ylabel('z'), title('p(Pa) vs. z(m)'), grid;
figure;
plot(rho,z), xlabel('rho'), ylabel('z'), title('rho(kg/m^3) vs. z(m)'),
grid;

```

Program results:







The modification programs for Exercises E2.8b through E2.8h are with the original solution programs for Exercises E2.1-E2.7.

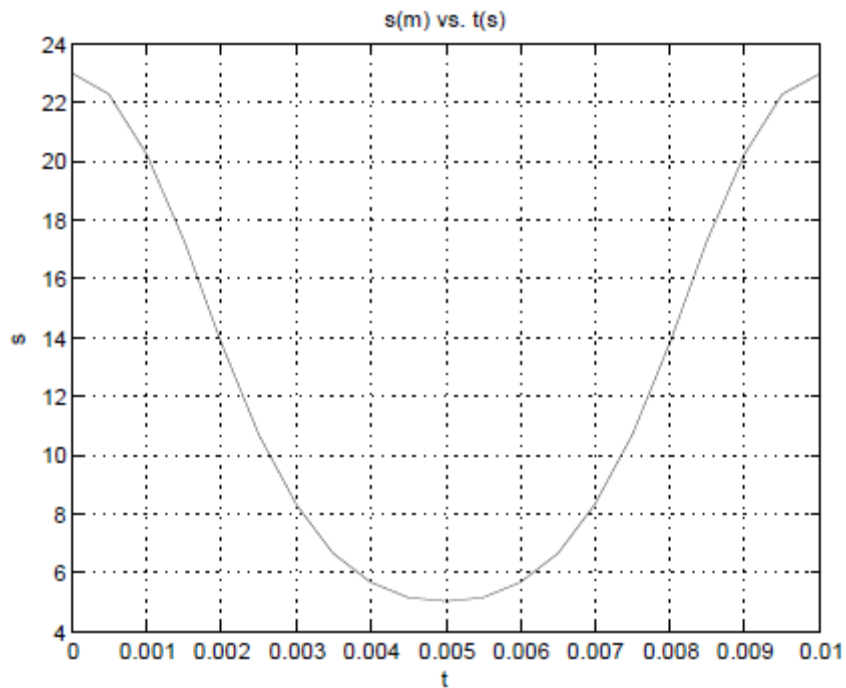
- b. Modify Exercise E2.1 to include a plot of s vs. t for $0 \leq t \leq 0.01 s$.
- c. Modify Exercise E2.2 include a plot of y vs. t for $0 \leq t \leq 10 s$.
- d. Modify Exercise E2.3 to include a plot of y vs. x at $0 \leq t \leq t_f$ in steps of $0.01 s$,
use small circles to define the basketball path.
- e. Modify Exercise E2.4 to include a plot of x vs. t and V vs. t on 2 separate graphs.

- f. Modify Exercise E2.5 to include a plot of V vs. t . Also, print to the screen values of W , B and V_T .
- g. Modify Exercise E2.6 to include a plot of v vs. t .
- h. Modify Exercise E2.7 to include a plot of V vs. t .

E2.8b.

Adding the following line to Exercise E2_1.m program produces the following plot.

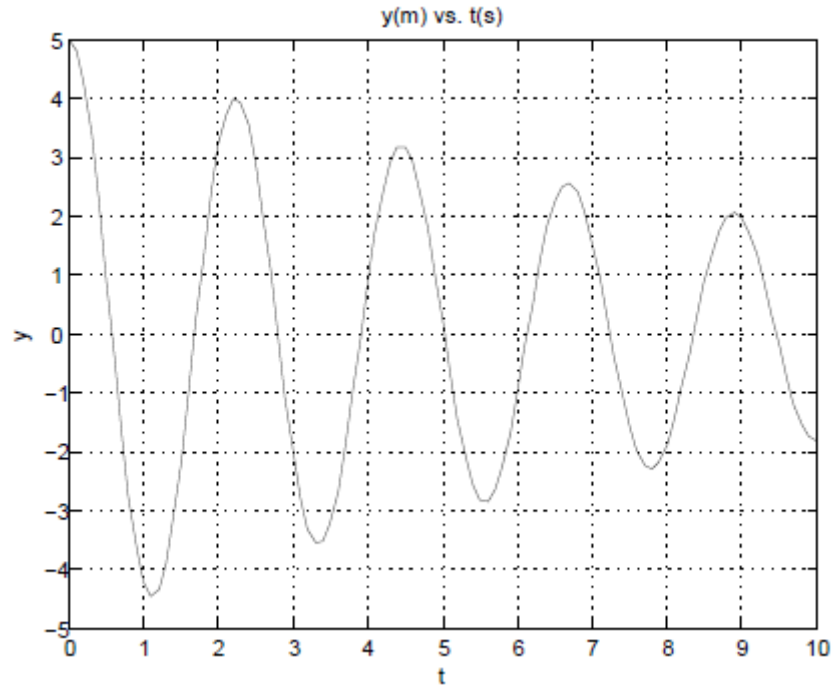
```
% E2.8b
plot(tt,s), xlabel('t'), ylabel('s'), title('s(m) vs. t(s)'), grid;
```



E2.8c

Adding the following line to Exercise E2_2.m program produces the following plot.

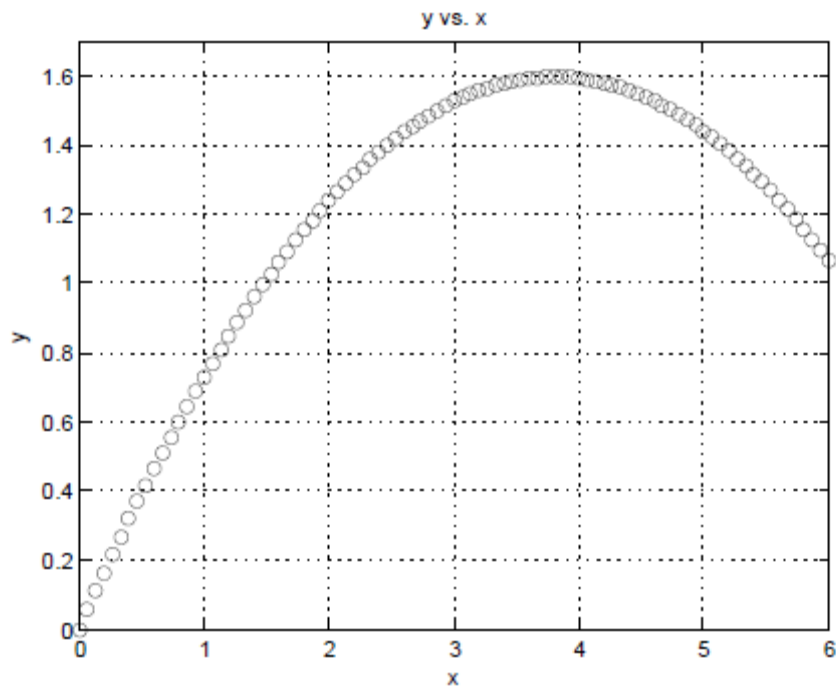
```
plot(tt,y), xlabel('t'), ylabel('y'), title('y(m) vs. t(s)'),grid;
```



E2.8d.

Adding the following line to Exercise E2_3.m program produces the following plot.

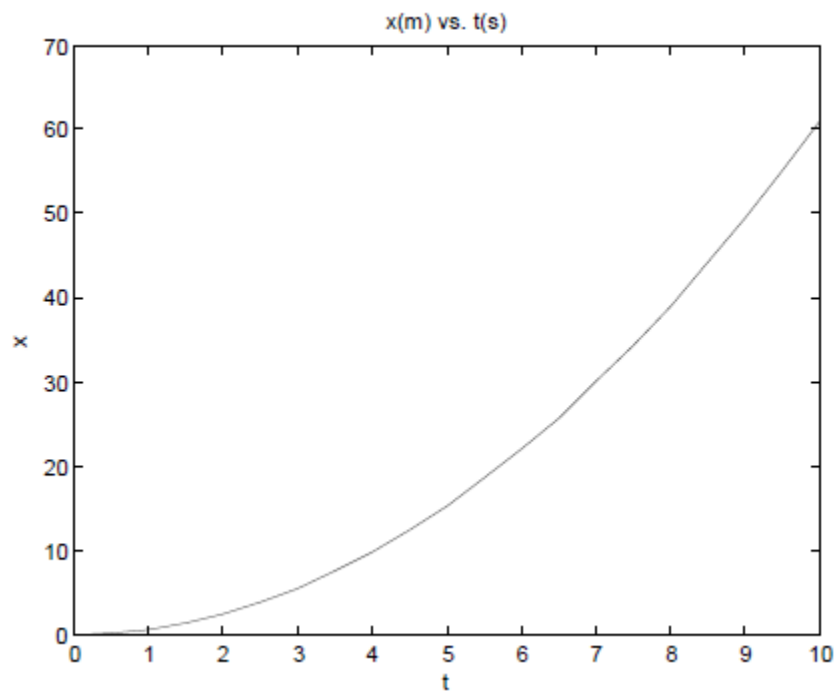
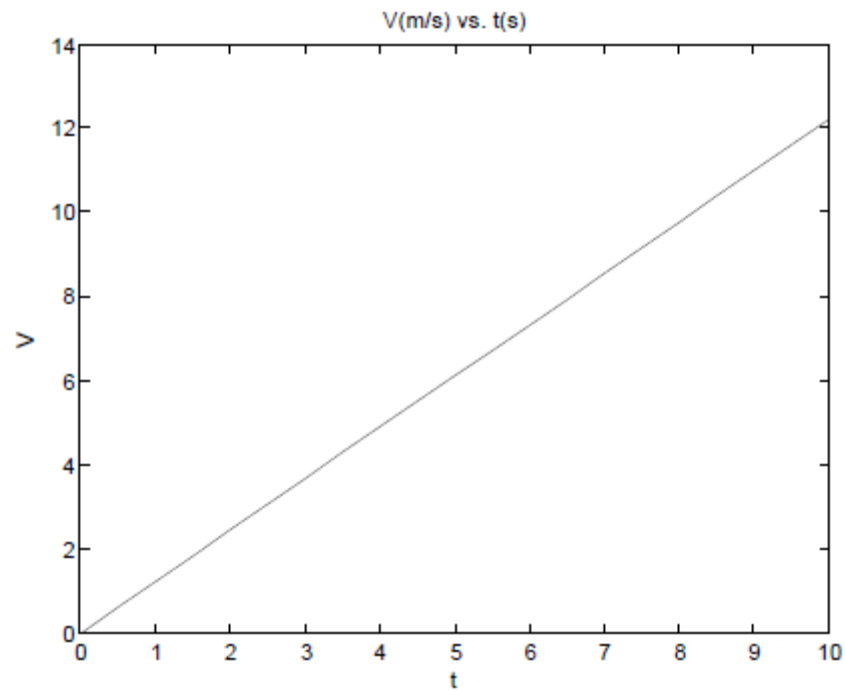
```
plot(x,y, 'o'), xlabel('x'), ylabel('y'), title('y vs. x'),  
axis([0,6,0,1.7]),grid;
```



E2.8e.

Adding the following lines to Exercise E2_4.m program produces the following plot.

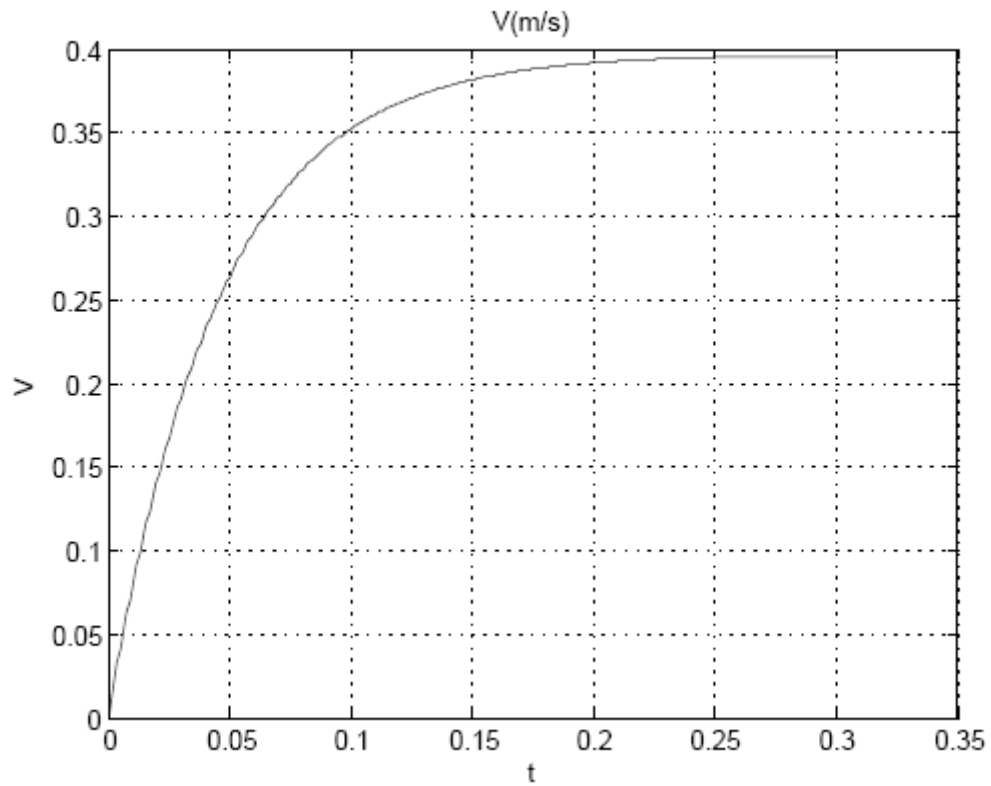
```
plot(t,V), xlabel('t'), ylabel('V'), title('V(m/s) vs. t(s)'), grid;
figure;
plot(t,x), xlabel('t'), ylabel('x'), title('x(m) vs. t(s)'), grid;
```



E2.8f

Adding the following line to Exercise E2_5.m program produces the following plot.

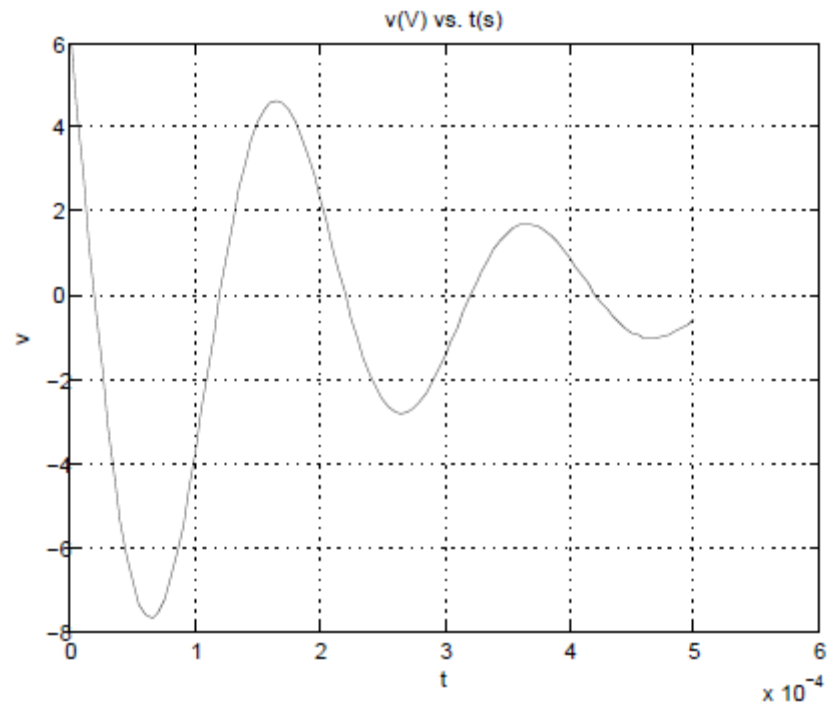
```
plot(t,V), xlabel('t'), ylabel('V'), title('V(m/s)'), grid;
```



E2.8g

Adding the following line to Exercise E2_6.m program produces the following plot.

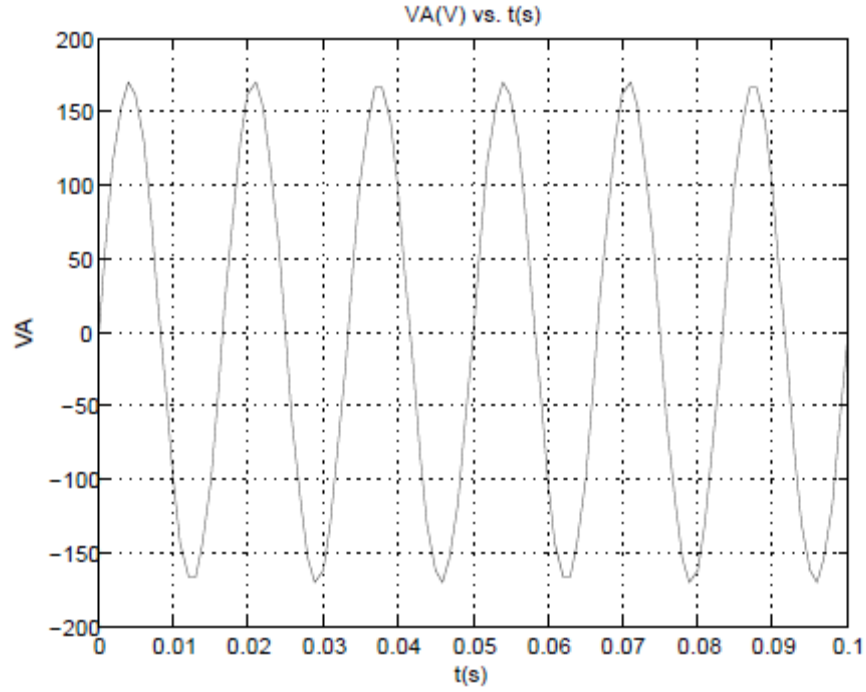
```
plot(t,v), xlabel('t'), ylabel('v'), title('v(V) vs. t(s)'),grid;
```



E2.8h

Adding the following lines to Exercise E2_7.m program produces the following plot.

```
plot(t,VA), xlabel('t(s)'), ylabel('VA'),
title('VA(V) vs. t(s)'), grid;
```



E2.9. In this exercise, we consider two cars on a collision course:

car1: $x_1=500$ m, $y_1=100$ m, $|\vec{V}_1|=40$ m/s. car1 moves in a straight line which

makes an angle of 60° with the x axis.

car2: $x_2=2000$ m, $y_2=200$ m. car2 moves in a straight line and makes an angle

of 45° with the $(-x)$ axis.

The collision coordinates are (x_c, y_c) . See Figure 2.20

$$\frac{y_c - y_1}{x_c - x_1} = \tan(60^\circ), \quad \frac{y_c - y_2}{x_2 - x_c} = \tan(45^\circ) \quad (2.24)$$

$$y_c = y_1 + (x_c - x_1) \times \tan(60^\circ) \quad (2.25)$$

$$y_c = y_2 + (x_2 - x_c) \times \tan(45^\circ) \quad (2.26)$$

$$y_1 + (x_c - x_1) \times \tan(60^\circ) = y_2 + (x_2 - x_c) \times \tan(45^\circ) \quad (2.27)$$

$$x_c = \frac{y_2 - y_1 + x_1 \tan(60^\circ) + x_2 \tan(45^\circ)}{\tan(60^\circ) + \tan(45^\circ)} \quad (2.28)$$

$$d_1 = \sqrt{(x_c - x_1)^2 + (y_c - y_1)^2} = |\vec{V}_1| t_c \quad (2.29)$$

$$d_2 = \sqrt{(x_2 - x_c)^2 + (y_c - y_2)^2} = |\vec{V}_2| t_c \quad (2.30)$$

$$\text{where } t_c = \text{time of collision} = \frac{d_1}{|\vec{V}_1|} \quad (2.31)$$

$$\frac{d_1}{|\vec{V}_1|} = \frac{d_2}{|\vec{V}_2|} \rightarrow |\vec{V}_2| = |\vec{V}_1| \times \frac{d_2}{d_1} \quad (2.32)$$

On line 1:

$$x(t) = x_1 + |\vec{V}_1| \cos(60^\circ) t, \quad y(t) = y_1 + |\vec{V}_1| \sin(60^\circ) t \quad (2.33)$$

On line 2:

$$x(t) = x_2 - |\vec{V}_2| \cos(45^\circ) t, \quad y(t) = y_2 + |\vec{V}_2| \sin(45^\circ) t \quad (2.34)$$

Create a MATLAB program that will do the following.

- a. Create a plot of the intersecting lines of lengths d_1 and d_2 .

Note: you only need to specify two points on the line to plot the line.

- b. Determine $|\vec{V}_2|$ that will cause the collision to take place.
- c. Take $t = 0 : t_c/5 : t_c$
- d. Plot the two lines and the two cars positions at t_i , shown as small circles, all on

the same graph. Your second plot should look like Figure 2.21.

The program follows:

```
% E_2_9.m
% This program plots the paths of two cars on a collision course.
```



```

% The paths are straight lines. The program also plots the
% positions of the two cars as a function of time.

% The speed of one car is 40 m/s

% The program determines the speed of the second car that
% would result in a collision.

clear; clc;

x1=500.0; y1=100.0; x2=2000.0; y2=200.0; va=40;

num=y2-y1+x1*tand(60)+x2*tand(45);

den=tand(60)+tand(45);

xc=num/den;

yc=y1+(xc-x1)*tand(60);

yc2=y2+(x2-xc)*tand(45);

fprintf('xc=%10.4f   yc=%10.4f   yc2=%10.4f \n',xc,yc,yc2);

arg1=(xc-x1)^2+(yc-y1)^2;

d1=sqrt(arg1);

arg2=(x2-xc)^2+(yc-y2)^2;

d2=sqrt(arg2);

fprintf('d1=%10.4f   d2=%10.4f   \n',d1,d2);

vb=va*d2/d1;

tc=d1/va;

xcaf=x1+va*cosd(60)*tc; ycaf=y1+va*sind(60)*tc;

xcbf=x2-vb*cosd(45)*tc; ycbf=y2+vb*sind(45)*tc;

fprintf('xcaf=%10.4f   ycaf=%10.4f \n',xcaf,ycaf);

fprintf('xcbf=%10.4f   ycbf=%10.4f \n',xcbf,ycbf);

% Line LA

xa(1)=x1; ya(1)=y1;

xa(2)=xc; ya(2)=yc;

%line LB

xb(1)=x2; yb(1)=y2;

```

```

xb(2)=xc; yb(2)=yc;

plot(xa,ya,xb,yb), xlabel('x'), ylabel('y'),

title('plot of lines LA and LB'),

figure;

t=0.0:tc/5:tc;

for i=1:length(t)

    xca(i)=x1+va*cosd(60)*t(i); yca(i)=y1+va*sind(60)*t(i);

    xcb(i)=x2-vb*cosd(45)*t(i); ycb(i)=y2+vb*sind(45)*t(i);

end

plot(xa,ya,xb,yb,xca,yca,'o',xcb,ycb,'o'), xlabel('x'), ylabel('y'),

title('vehicle positions at times t');

```

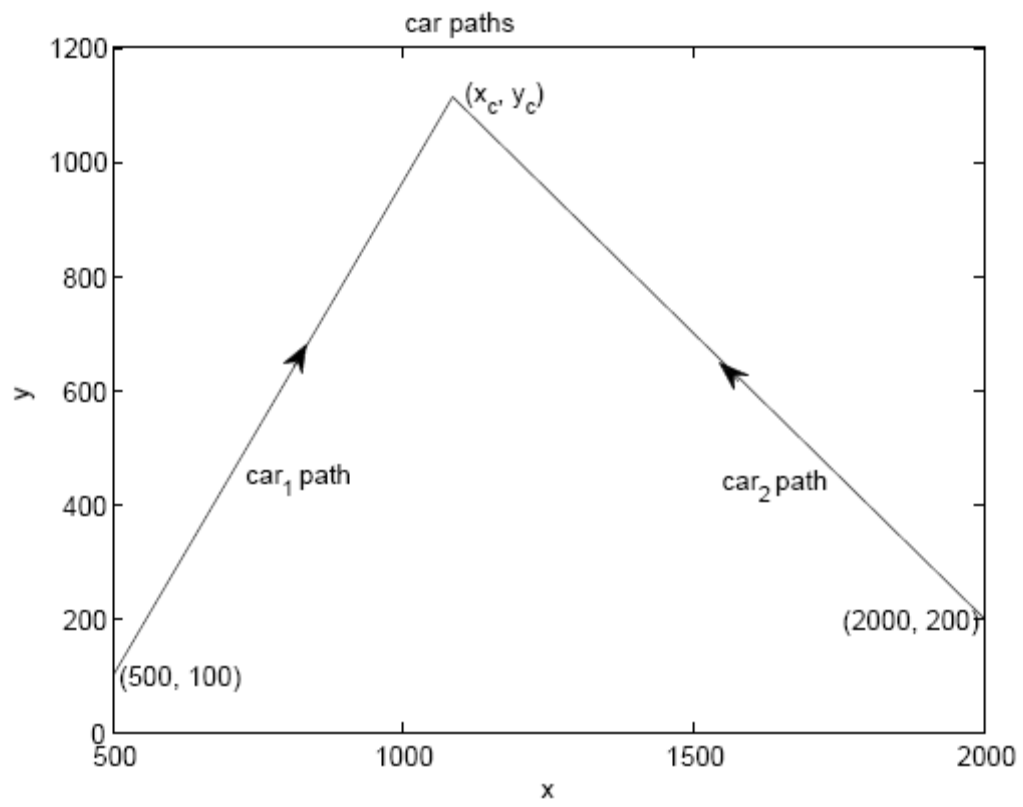
Program results:

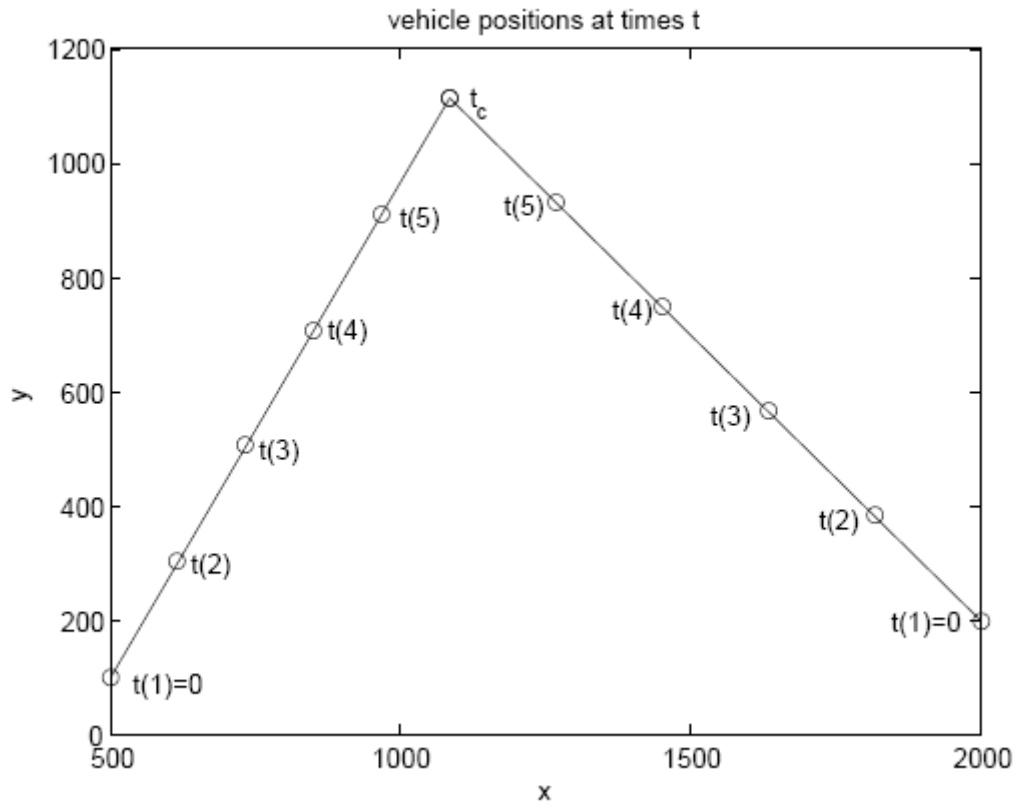
```

xc= 1085.6406   yc= 1114.3594   yc2= 1114.3594
d1= 1171.2813   d2= 1293.0994
xcaf= 1085.6406   ycaf= 1114.3594
xcbf= 1085.6406   ycbf= 1114.3594
>>

```

Insert textbox and arrow on the figure to produce the figure below.





E2.10. A formula describing the fluid level, $h(t)$, in a tank as the fluid discharges through a small orifice (see Figure 2.22) is:

$$\sqrt{h} = \sqrt{h_o} - \frac{C_d A_o}{2 A_T} \sqrt{2g} t \quad (2.35)$$

where

C_d = the discharge coefficient.

h_o = the fluid level in the tank at time, $t = 0$.

A_o = the circular area of the orifice.

A_T = the circular cross-sectional area of the tank.

Create a MATLAB program that will

- determine vectors h vs. t , for $0 \leq t \leq 80$ s.
- create a table of containing 20 values of t and h (every fourth time step) and print the table to a file.
- create a plot of h vs. t and print it .

Use the following parameters:

$h_o = 0.3$ m, the tank diameter, $D = 0.8$ m and the orifice diameter,

$d = 0.05$ m, $g = 9.81$ m/s² and $C_d = 0.7$.

The Program follows:

```
% E2_10.m

% This program determines the height of the free surface of a tank
% discharging through an orifice.

% The formula for the height, h, is:

% sqrt(h)=sqrt(ho)-Cd*Ao/2/At*sqrt(2*g)*t

% ho=0.3 m; D=0.8 m; d=0.05 m; Cd=0.7; g=9.81 m/s^2;

clear; clc;

ho=0.3; D=0.8; d=0.05; Cd=0.7; g=9.81;

At=pi/4*D^2; Ao=pi/4*d^2;

t=0:80;

for i=1:length(t)

    sqh=sqrt(ho)-Cd/2*Ao/At*sqrt(2*g)*t(i);

    h(i)=sqh^2;

end

fo=fopen('output.txt','w');

fprintf(fo,' t(s)      h(m)  \n');

fprintf(fo,'----- \n');

for i=1:4:length(t)

    fprintf(fo,'%3i      %10.3f \n',t(i),h(i));
```

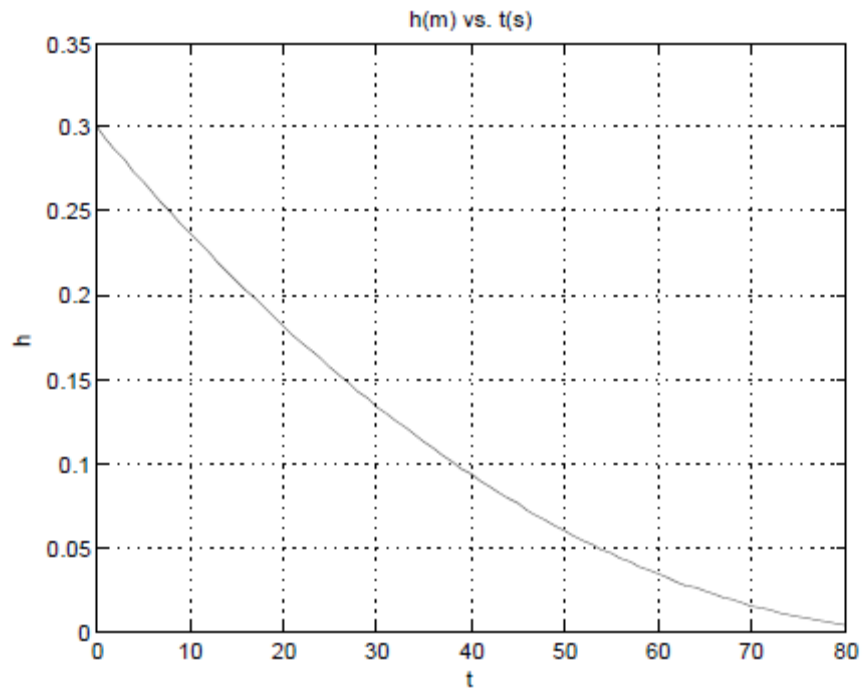
```
end
```

```
plot(t,h), xlabel('t'), ylabel('h'), title('h(m) vs. t(s)'), grid;
```

Program results:

t(s)	h(m)

0	0.300
4	0.274
8	0.249
12	0.226
16	0.203
20	0.182
24	0.162
28	0.143
32	0.125
36	0.109
40	0.093
44	0.079
48	0.066
52	0.054
56	0.044
60	0.034
64	0.026
68	0.018
72	0.012
76	0.008
80	0.004



E2.11. Figure 2.23 shows a differential amplifier using bipolar junction transistors (BJTs) which takes an input V_{in} and generates an output V_{out} .

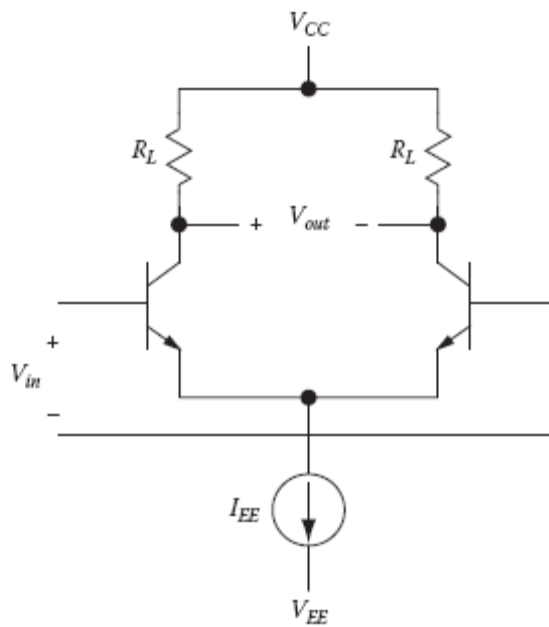


Figure 2.23

The relationship between the output and input voltage for this amplifier corresponds to the equation

$$V_{out} = \alpha I_{EE} R_L \tanh\left(-\frac{q}{2kT} V_{in}\right) \quad (2.38)$$

where q is the electron charge, k is Boltzmann's constant, T is absolute temperature, and α is the collector-emitter current ratio (and is dependent on process used to fabricate the transistors). V_{in} , V_{out} , R_L , and I_{EE} are defined in Figure 2.23. The gain, A , of the amplifier which is defined as the slope of the curve of V_{out} vs. V_{in} , which we can approximate as

$$A \approx \frac{\Delta V_{out}}{\Delta V_{in}} \quad (2.39)$$

However, we can also obtain the slope by taking the derivative of Equation (2.23), i.e.,

$$\frac{dV_{out}}{dV_{in}} = \frac{d}{dV_{in}} \left(\alpha I_{EE} R_L \tanh\left(-\frac{qV_{in}}{2kT}\right) \right) \quad (2.40)$$

Note:

$$\frac{d \tanh x}{dx} = \text{sech}^2 x = \frac{1}{\cosh^2 x} \quad (2.41)$$

Create a MATLAB program that

- a. Determines V_{out} and A for $-200 < V_{in} < 200$ millivolts in steps of 2 millivolts.

Assume $\alpha = 0.98$, $I_{EE} = 2 \text{ mA}$, $R_L = 10 \text{ k}\Omega$, and $T = 300 \text{ K}$.

- b. Plots V_{out} vs. V_{in} and A vs. V_{in} , both by Equations (2.37) and (2.38).

The Program follows:

```
% E2_11.m

% This program determines the gain,A, of a bipolar junction
% transistor. The relationship between the voltage output, V_out,
% and the voltage input, V_in, is given by:
%  $V_{out} = \alpha I_{ee} R_L \tanh(-q V_{in} / (2 k T))$ 
% approx  $A = d(V_{out}) / d(V_{in})$ 
% analytically,  $A = d(V_{out}) / d(V_{in})$ 
% -0.2 < V_in < 0.2 millivolts in steps of 2 millivolts.
%  $\alpha = 0.98$ ,  $I_{ee} = 2\text{mA}$ ,  $R_L = 100\text{ kohm}$ ,  $T = 300\text{K}$ 

clear; clc;

alpha=0.98; Iee=2e-3; RL=100e+3; T=300; q=1.6e-19; k=1.38e-23;
arg=-q/(2*k*T);
V_in=-0.2:0.002:0.2;

% compute output voltage (an arithmetic expression of a vector
% produces a vector)
for i=1:length(V_in)
    V_out(i)=alpha*Iee*RL*tanh(arg*V_in(i));
end

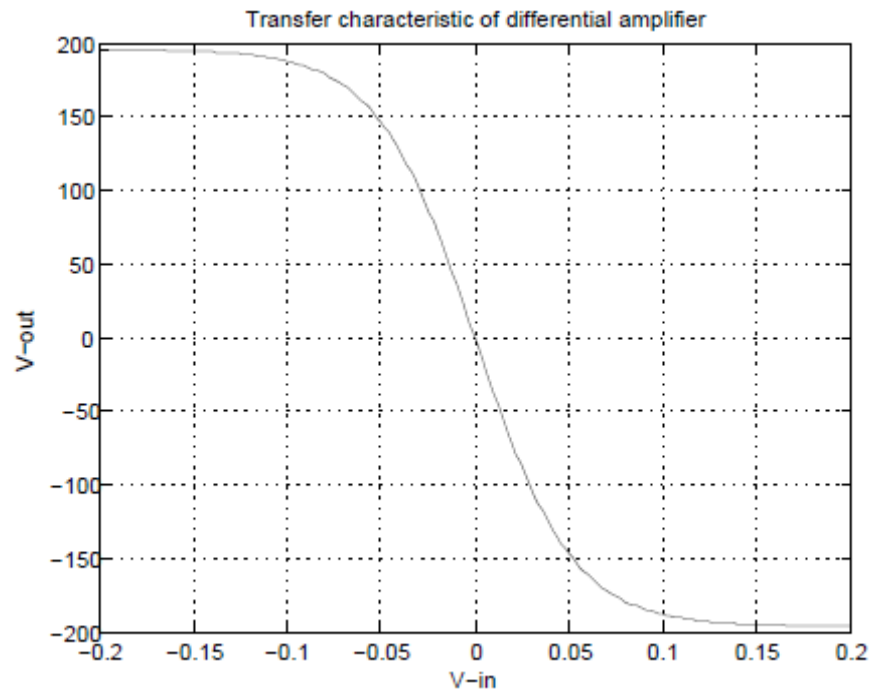
plot(V_in,V_out), xlabel('V-in'), ylabel('V-out'); grid;
title('Transfer characteristic of differential amplifier');
figure;

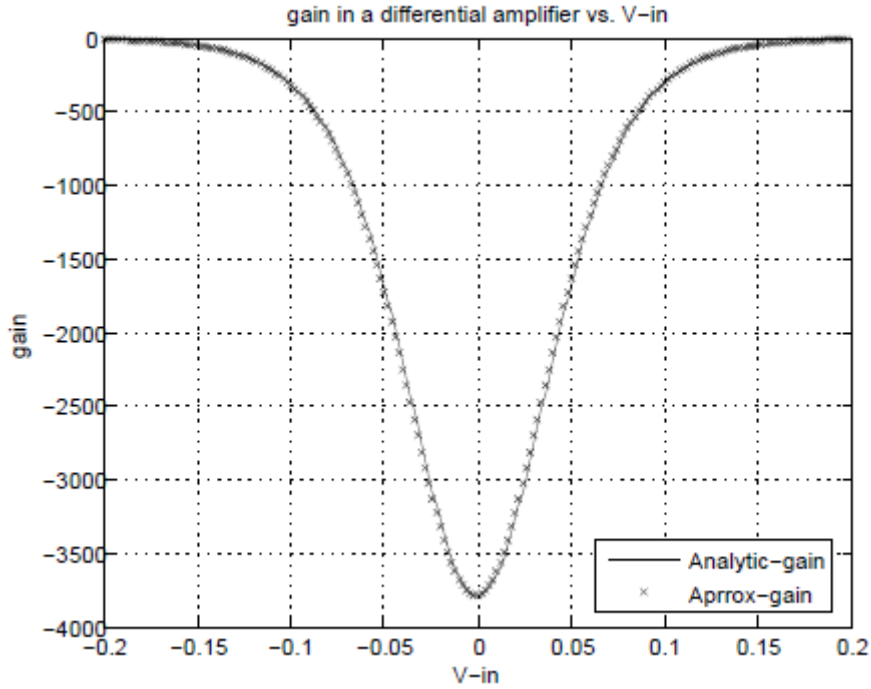
for i=1:length(V_in)-1
    d_V_out = V_out(i+1)-V_out(i);
    d_V_in=V_in(i+1)-V_in(i);
    Approx_gain(i)=d_V_out/d_V_in;
    Analytic_gain(i)= alpha*Iee*RL*(-q/(2*k*T))/(cosh(arg*V_in(i)))^2;
    V_in2(i)=V_in(i);
```

```
end
```

```
plot(V_in2,Analytic_gain,V_in2,Approx_gain,'x'), xlabel('V-in'),  
ylabel('gain'), title('gain in a differential amplifier vs. V-in'),  
grid; legend('Analytic-gain','Aprrox-gain');
```

Program results:





E2.12. When a fluid flows through a pipe there is a pressure drop that is proportional to the pipes length (see Figure 2.24).



Figure 2.24

For a pipe having a circular cross-section, the pressure drop, $p_1 - p_2$ is given by

$$p_1 - p_2 = \frac{\rho V^2}{2} \frac{L}{D} f \quad (2.42)$$

where

ρ = fluid density.

V = Average fluid velocity in the pipe.

D = the pipe diameter.

L = pipe length between points 1 and 2.

f = friction factor.

The friction factor has been determined by experiment. For smooth pipes a formula that approximates the experimental data is [2].

$$f = (1.82 \log_{10} \text{Re} - 1.64)^{-2} \quad (2.43)$$

where

$$\text{Re} = \frac{\rho V D}{\mu} \text{ (Reynolds number).} \quad (2.44)$$

μ = absolute fluid viscosity.

Develop a MATLAB program that will calculate f vs. Re . Take:

```
Re = [5e3 7.5e3 1e4 2.5e4 5e4 7.5e4 1e5 2.5e5 5.0e5 5e5 7.5e5 1e6 ...  
      2.5e6 5e6 7.5e6 1e7 2.5e7 5e7 7.5e7 1e8]
```

Plot $\log(\text{Re})$ on the x axis and f on the y axis (semi-log plot).

The Program follows:

```
% E2_12.m  
  
% This program calculates the friction factor, f, vs. the Reynolds  
% Number, Re. It then creates a plot of f vs. Re.  
  
clear; clc;  
  
Re = [5.0e3 7.5e3 1.0e4 2.5e4 5.0e4 7.5e4 1.0e5 2.5e5 5.0e5 ...  
      7.5e5 1.0e6 2.5e6 5.0e6 7.5e6 1.0e7 2.5e7 5.0e7 7.5e7 1.0e8];  
  
for i=1:length(Re)
```

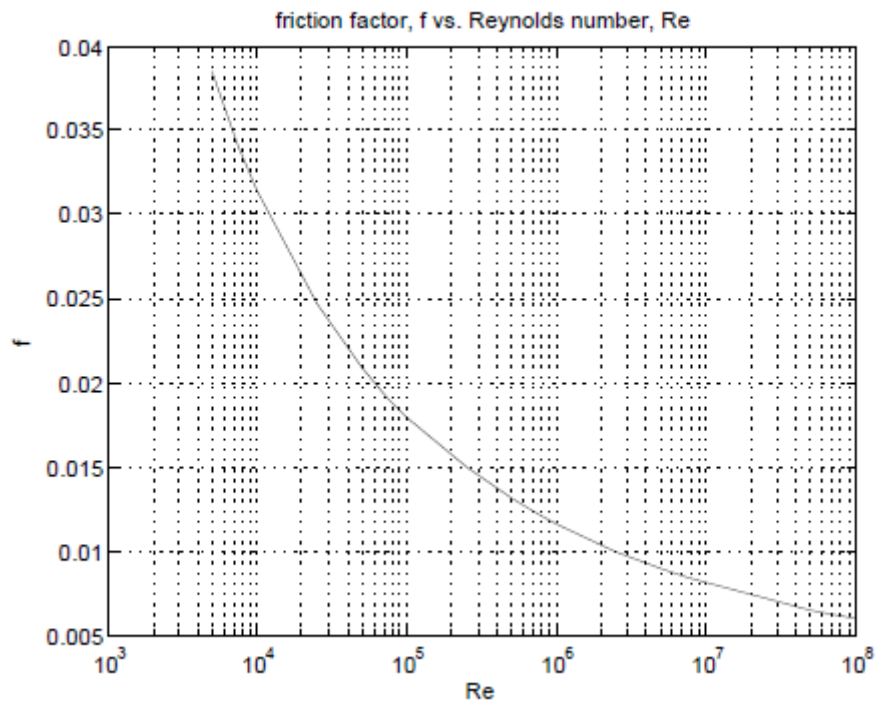
```

    f(i)=1/(1.82*log10(Re(i))-1.64)^2;
end

semilogx(Re,f), xlabel('Re'), ylabel('f'), grid,
title('friction factor, f vs. Reynolds number, Re');

```

Program results:



E2.13. This exercise is a modification of Exercise E2.12. When the interior of a pipe is not smooth, a friction factor, $\frac{e}{D}$ is introduced, where e is a roughness factor for the pipe interior, which is determined by experiment.

The Swamee-Jain formula has been used to approximate the experimental data. The Swamee-Jain formula [3] is

$$f = \frac{1.325}{\left[\log \left(\frac{e}{3.7 * D} + \frac{5.74}{\text{Re}^{0.9}} \right) \right]^2} \quad (2.45)$$

Develop a MATLAB program that will calculate f as a function of Re and $\frac{e}{D}$. Take:

$\text{Re} = [5.0\text{e}3 \ 7.5\text{e}3 \ 1.0\text{e}4 \ 2.5\text{e}4 \ 5.0\text{e}4 \ 7.5\text{e}4 \ 1.0\text{e}5 \ 2.5\text{e}5 \ 5.0\text{e}5 \dots$
 $7.5\text{e}5 \ 1.0\text{e}6 \ 2.5\text{e}6 \ 5.0\text{e}6 \ 7.5\text{e}6 \ 1.0\text{e}7 \ 2.5\text{e}7 \ 5.0\text{e}7 \ 7.5\text{e}7 \ 1.0\text{e}8]$

$\frac{e}{D} = [0.0 \ 0.0001 \ 0.001 \ 0.005];$

Create a semi-log plot of f vs. $\log(\text{Re})$ for different $\frac{e}{D}$ values. Make $\log(\text{Re})$ on the x axis and f on the y axis.

The Program follows:

```
% E2_13.m

% This program calculates the friction factor, f, vs. the Reynolds
% Number, Re for several different values of e/D.
% It then plots the results.

clear; clc;

Re = [5.0e3 7.5e3 1.0e4 2.5e4 5.0e4 7.5e4 1.0e5 2.5e5 5.0e5 ...
      7.5e5 1.0e6 2.5e6 5.0e6 7.5e6 1.0e7 2.5e7 5.0e7 7.5e7 1.0e8];

eod=[0.0 0.0001 0.001 0.005];

for j=1:length(eod);
    for i=1:length(Re)
        arg=log(eod(j)/3.7+5.74/Re(i)^0.9);
        if j==1
            f1(i)=1.325/arg^2;
        elseif j==2
            f2(i)=1.325/arg^2;
        elseif j==3
```

```

        f3(i)=1.325/arg^2;
    else
        f4(i)=1.325/arg^2;
    end

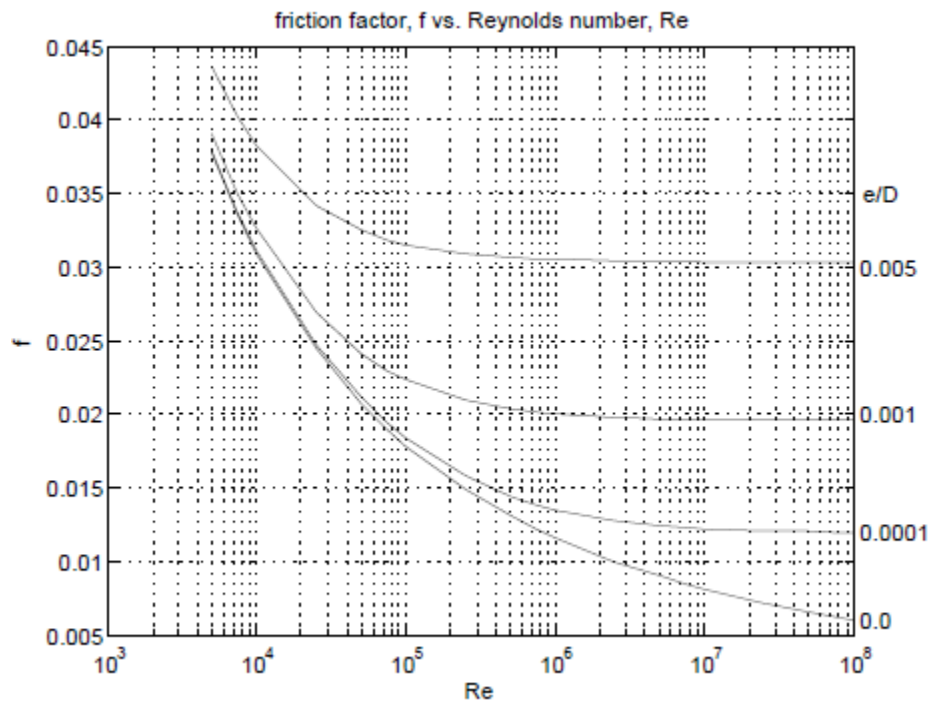
end

end

semilogx(Re,f1,Re,f2,Re,f3,Re,f4), xlabel('Re'), ylabel('f'), grid,
title('friction factor, f vs. Reynolds number, Re'),
text(1.15e8,0.035,'e/D'),
text(1.1e8,0.006,'0.0'),text(1.1e8,0.012,'0.0001'),
text(1.1e8,0.02,'0.001'), text(1.1e8,0.03,'0.005');

```

Program results:



E2.14. Carbon Dioxide properties of temperature, T , specific volume, v , and pressure, p , based on the Redlich-Kwong equation are tabulated in Table 2.1. We wish to determine by interpolation v and p at a temperature between table values. The general interpolation formula in terms of y and x was given in Equation (2.43), which is:

$$y = y_1 + \frac{(y_2 - y_1) \times (x - x_1)}{x_2 - x_1}$$

where x_1 and x_2 are the values of x that enclose x , and y_1 and y_2 are the values of y at x_1 and x_2 respectively. Develop a MATLAB program that does the following:

1. Using Table 2.1, create vectors T , v and p .
2. Asks the user to enter a temperature, T , from the keyboard that is not in the table.
3. Using the if-elseif ladder determines the interval enclosing the entered T value.
4. Interpolate for v and p .
5. Print to the screen the interpolated values of v and p .

Table 2.1. Carbon Dioxide Properties.

T (K)	v ($m^3/kmol$)	p (bars)
350	0.28	7.650
400	0.32	8.574
450	0.36	9.159
500	0.40	9.547
550	0.44	9.813
600	0.48	10.001
650	0.52	10.136
700	0.56	10.236

750	0.60	10.310
-----	------	--------

The Program follows:

```
% E2_14.m

% This program interpolates for CO2 properties at a specified
% temperature entered from the key board by a user of the program.
% The program continues as long as the user enters 'Y' when asked
% whether he/she wishes to enter another temperature.
% The temperature range is from 350K to 750K. The data used in the
% interpolation is in 3 vectors: Tt,vt,pt
% Temperature, T, is in K, specific volume, v, is in m^3/kmol
% and pressure, p, is in bar.
% The output goes to the screen.
% The use of the if-elseif ladder is not the most efficient way to do
% the problem. The prescribed method is only used as an exercise in
% using the if-elseif ladder.
% The program should be started from the command window by entering
% E2_14.

clear; clc;

Tt=350:50:750;
vt=0.28:0.04:0.60;
pt=[7.650 8.574 9.159 9.547 9.813 10.001 10.136 10.236 10.310];
char1='Y'
while char1=='Y';

    fprintf('350 < T < 750. Enter the temperature, T , \n');

    T=input('at which the properties are to be determined \n');

    if T < 350 || T > 750
```

```

        fprintf('You entered a temperature out of range, please \n');

        fprintf('restart program \n');

        break;

elseif T >= Tt(1) && T < Tt(2)

    T1=Tt(1); T2=Tt(2); v1=vt(1); v2=vt(2);

    p1=pt(1); p2=pt(2);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K)  v=%5.3f(m^3/kmol)  p=%8.4f(bar) \n\n',...

        T,v,p);

elseif T >= Tt(2) && T < Tt(3)

    T1=Tt(2); T2=Tt(3); v1=vt(2); v2=vt(3);

    p1=pt(2); p2=pt(3);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K)  v=%5.3f(m^3/kmol)  p=%8.4f(bar) \n\n',...

        T,v,p);

elseif T >= Tt(3) && T < Tt(4)

    T1=Tt(3); T2=Tt(4); v1=vt(3); v2=vt(4);

    p1=pt(3); p2=pt(4);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K)  v=%5.3f(m^3/kmol)  p=%8.4f(bar)\n\n',...

        T,v,p);

elseif T >= Tt(4) && T < Tt(5)

    T1=Tt(4); T2=Tt(5); v1=vt(4); v2=vt(5);

    p1=pt(4); p2=pt(5);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

```

```

        fprintf('T=%5.1f(K) v=%5.3f(m^3/kmol) p=%8.4f(bar) \n\n',...
        T,v,p);

elseif T >= Tt(5) && T < Tt(6)

    T1=Tt(5); T2=Tt(6); v1=vt(5); v2=vt(6);

    p1=pt(5); p2=pt(6);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K) v=%5.3f(m^3/kmol) p=%8.4f(bar) \n\n',...
    T,v,p);

elseif T >= Tt(6) && T < Tt(7)

    T1=Tt(6); T2=Tt(7); v1=vt(6); v2=vt(7);

    p1=pt(6); p2=pt(7);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K) v=%5.3f(m^3/kmol) p=%8.4f(bar) \n\n',...
    T,v,p);

elseif T >= Tt(7) && T < Tt(8)

    T1=Tt(7); T2=Tt(8); v1=vt(7); v2=vt(8);

    p1=pt(7); p2=pt(8);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K) v=%5.3f(m^3/kmol) p=%8.4f(bar) \n\n',...
    T,v,p);

elseif T >= Tt(8) && T <= Tt(9)

    T1=Tt(8); T2=Tt(9); v1=vt(8); v2=vt(9);

    p1=pt(8); p2=pt(9);

    v=v1+(v2-v1)*(T-T1)/(T2-T1);

    p=p1+(p2-p1)*(T-T1)/(T2-T1);

    fprintf('T=%5.1f(K) v=%5.3f(m^3/kmol) p=%8.4f(bar) \n\n',...

```

```

        T,v,p);
    end

    fprintf('Do you wish to enter another Temperature \n');

    char1=input('enter Y for yes or N for no \n','s');

end

```

Start the program from the command window by entering E2_14

Program results:

```

The temperature from 350K to 750K. Enter the temperature
at which the properties are to be determined.
380
T=380.0(K)  v = 0.304(m^3/kmol)  p = 8.2044(bar)

Do you wish to enter another Temperature
enter Y for yes or N for no
Y
The temperature from 350K to 750K. Enter the temperature.
at which the properties are to be determined
720
T=720.0(K)  v = 0.576(m^3/kmol)  p = 10.2656(bar)

Do you wish to enter another Temperature
enter Y for yes or N for no
N
>>

```

E2.15. Repeat Exercise E2.14, but this time use a for loop and a single if statement to determine the interval that encloses the temperature entered from the key board.

The Program follows:

```

% E2_15.m

% This program is a modification of the program for Exercise 2.14.

% This program uses a 'for' loop and a single 'if' statement to

% determine the interval containing the temperature entered from the

% key board. This is a more efficient way of interpolating the CO2

% properties than the method used in Exercise E2.14, which utilized the

% if-elseif ladder. The program continues as long as the user enters

```

```

% 'Y' when asked whether he/she wishes to enter another temperature.
% The temperature range is from 350K to 750K. The data used in the
% interpolation is in the 3 vectors: Tt,vt,pt
% Temperature,T, is in K, specific volume, v, is in m^3/kmol
% and pressure, p, is in bar.
% The output is to the screen. The program should be started from the
% command window by entering E2_15.

clear; clc;

Tt=350:50:750;

vt=0.28:0.04:0.60;

pt=[7.650 8.574 9.159 9.547 9.813 10.001 10.136 10.236 10.310];

char1='Y';

while char1=='Y';

    fprintf('The temperature range is from 350K to 750K. \n');

    fprintf('Enter the temperature at which the');

    T=input('properties are to be determined \n\n');

    if T < 350 || T > 750

        fprintf('You entered a temperature out of range, please \n');

        fprintf('restart program \n');

        break;

    end

    for i=1:length(Tt)-1

        if T >= Tt(i) && T < Tt(i+1)

            T1=Tt(i); T2=Tt(i+1); v1=vt(i); v2=vt(i+1);

            p1=pt(i); p2=pt(i+1);

            v=v1+(v2-v1)*(T-T1)/(T2-T1);

            p=p1+(p2-p1)*(T-T1)/(T2-T1);

            fprintf('T=%5.1f(K)   v=%5.3f(m^3/kmol)   p=%8.4f(bar) \n\n',...

                T,v,p);

```

```

        break;

    end

end

fprintf('Do you wish to enter another Temperatue \n');

char1=input('enter Y for yes or N for no \n','s');

end

```

Program results:

The temperature range is from 350K to 750K.
Enter the temperature at which the properties are to be determined

380
T=380.0(K) v = 0.304(m³/kmol) p = 8.2044(bar)

Do you wish to enter another Temperatue
enter Y for yes or N for no
Y
The temperature range is from 350K to 750K.
Enter the temperature at which the properties are to be determined

720
T=720.0(K) v =0.576(m³/kmol) p= 10.2656(bar)

Do you wish to enter another Temperatue
enter Y for yes or N for no
Y
The temperature range is from 350K to 750K.
Enter the temperature at which the properties are to be determined

520
T=520.0(K) v = 0.416(m³/kmol) p = 9.6534(bar)

Do you wish to enter another Temperatue
enter Y for yes or N for no
N
>>

E2.16. Many gasses under certain conditions behave as an ideal gas. The governing equation for an ideal gas is:

$$p v = R T \quad (2.48)$$

where

R = gas constant (R is different for different gases).

p = gas pressure.

v = specific volume.

T = absolute temperature.

Create a MATLAB program that will plot v vs. T for the following 4 gasses:

air, hydrogen, oxygen and carbon dioxide. Although air is a mixture of different gasses, it is frequently treated as a pure substance. Use the switch group to determine the proper gas constant and to create the requested plots. Take, $p = 2 \text{ atm}$ and $300 \leq T \leq 600 \text{ K}$ in steps of 50 K .

Note: $1 \text{ atm} = 1.01325 \times 10^5 \frac{\text{N}}{\text{m}^2}$

The gas constant, R , for each of the above listed gasses is given in Table 2.2.

Table 2.2. Gas constant for several gasses.

Gas	R (N-m/kg-K)
air	287.0
hydrogen	4124.0
oxygen	259.8
carbon dioxide	188.9

The Program follows:

% E2_16.m

```

% This program uses the ideal gas law to determine the specific volume,
% v, of one of four gasses. The temperature, T, varies from 300K to
% 600K in steps of 50K, while the pressure is held constant at 2 atm.
% A switch statement is used to determine the proper gas constant, R.
clear; clc;

Rt=[287.0  4124  259.8 188.9];

Tt=300:50:600;

p=2*1.01325e5;

gast=['air' 'hydrogen' 'oxygen' 'carbon dioxide'];

for i=1:4

    var=i;

    switch(var)

        case 1

            R=Rt(1);

        case 2

            R=Rt(2);

        case 3

            R=Rt(3);

        case 4

            R=Rt(4);

    end

    for j=1:length(Tt)

        v(j)=R*Tt(j)/p;

    end

    plot(Tt,v), xlabel('T'), ylabel('v'), grid,

    title('specic volume, v(m^3/kg) vs. temperature, T(K)'),

    if i==1

```



```

        text(555,0.42,'air');

        figure;

    elseif i==2

        text(555,6.2,'hydrogen');

        figure;

    elseif i==3

        text(555,0.37,'oxygen');

        figure;

    else

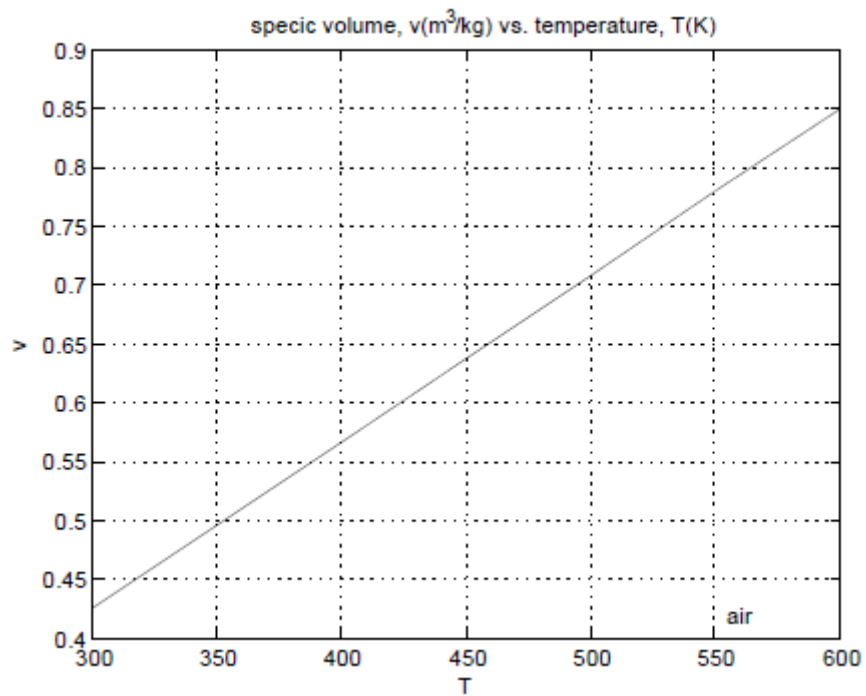
        text(540,0.27,'carbon dioxide');

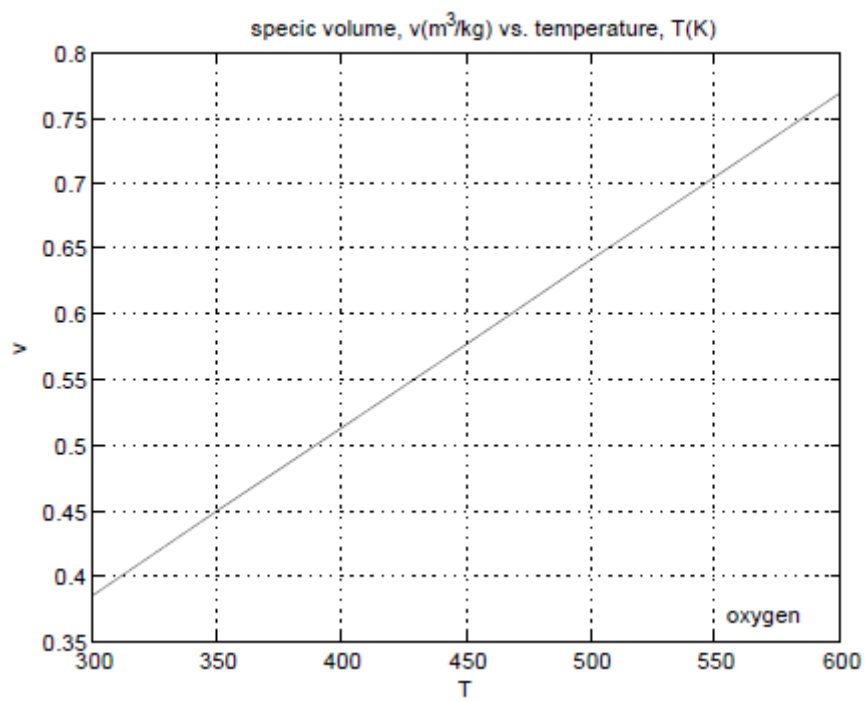
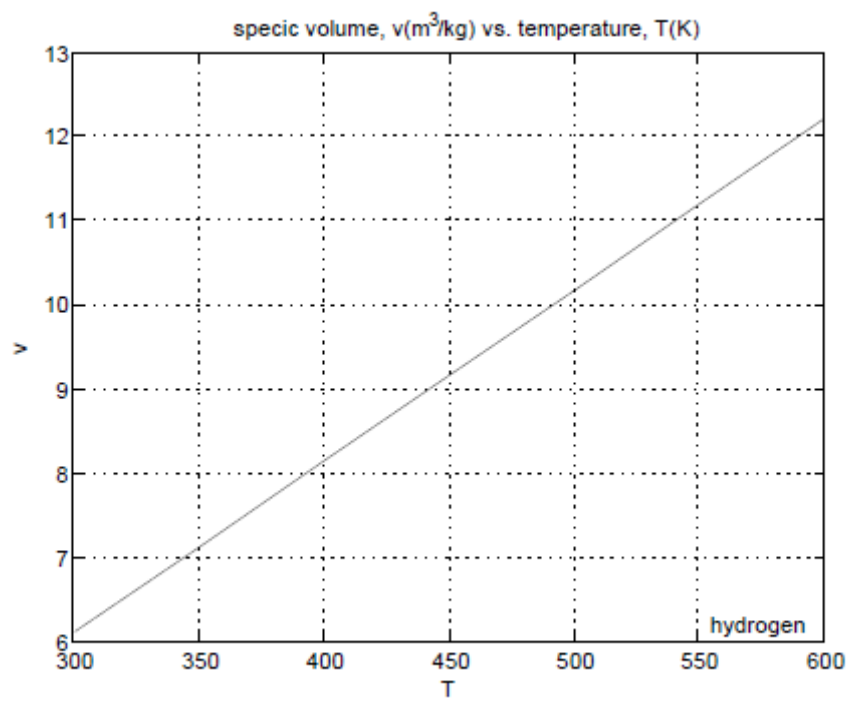
    end

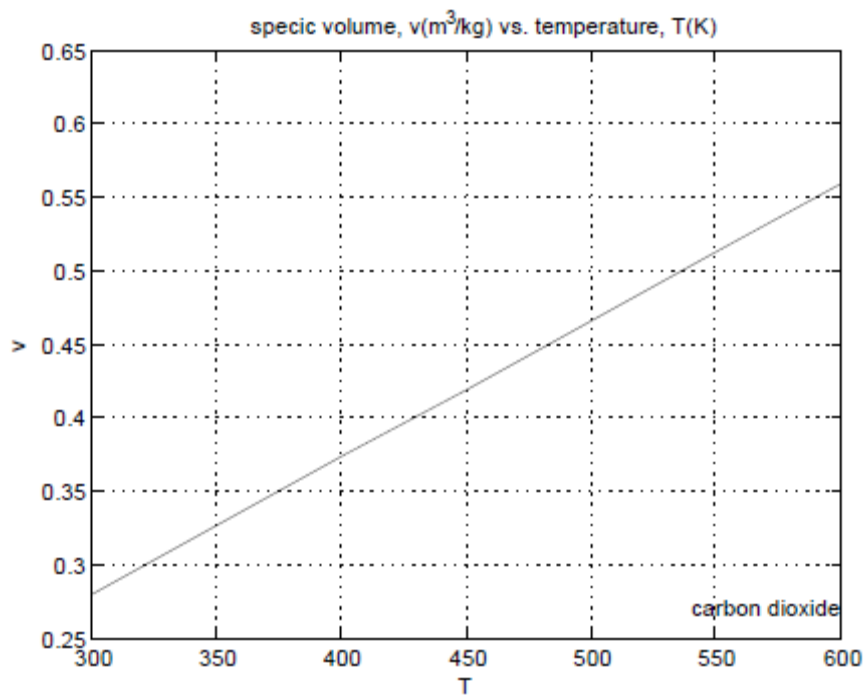
end
end

```

Program Results:







E2.17. Repeat Exercise E2.16, but this time use the if-elseif ladder to determine the proper gas constant and to create the four requested plots, all on the same page.

The Program follows:

```
% E2_17.m

% This program uses the ideal gas law to determine the specific volume,
% v, of one of four gasses. The temperature, T, varies from 300K to
% 600K in steps of 50K, while the pressure is held constant at 2 atm.
% A switch statement is used to determine the proper gas constant, R.

clear; clc;

Rt=[287.0  4124  259.8 188.9];

Tt=300:50:600;

p=2*1.01325e5;
```

```

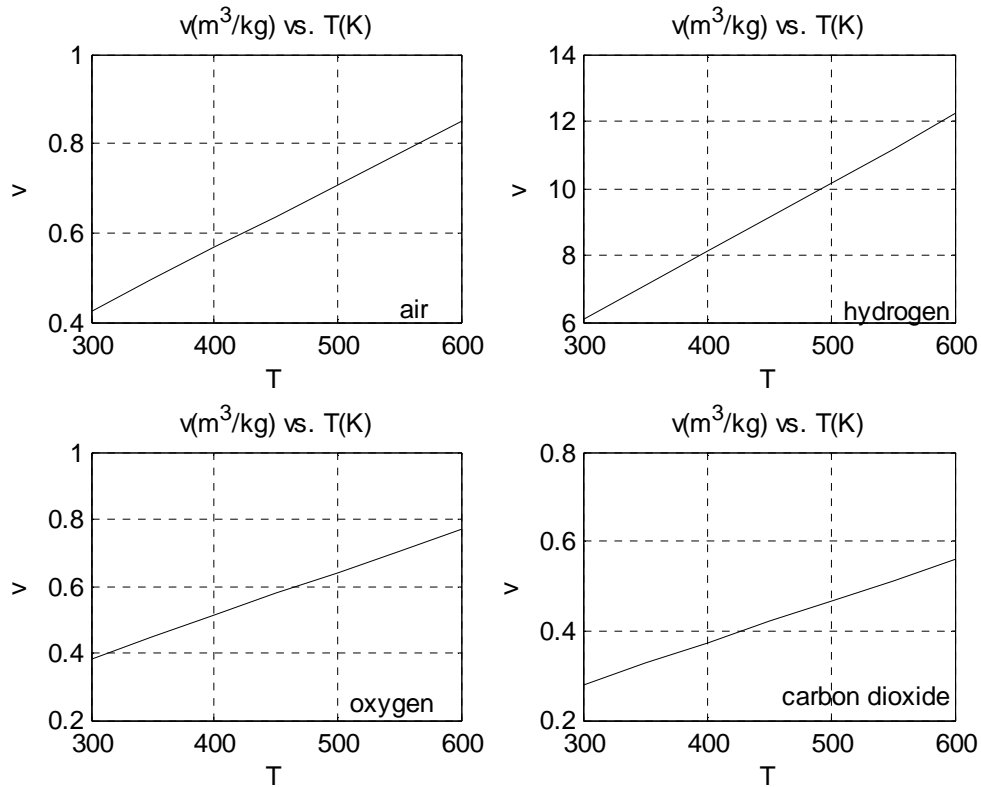
for i=1:4
    if i==1
        R=Rt(1);
    elseif i==2
        R=Rt(2);
    elseif i==3
        R=Rt(3);
    else
        R=Rt(4)
    end
    for j=1:length(Tt)
        v(j)=R*Tt(j)/p;
    end
    subplot(2,2,i);
    if I ==1
        plot(Tt,v), xlabel('T'), ylabel('v'), grid,
        title('v(m^3/kg) vs. T(K)'),
        text(550,0.43,'air');
    elseif i==2
        plot(Tt,v), xlabel('T'), ylabel('v'), grid,
        title('v(m^3/kg) vs. T(K)'),
        text(510,6.3,'hydrogen');
    elseif i==3
        plot(Tt,v), xlabel('T'), ylabel('v'), grid,
        title('v(m^3/kg) vs. T(K)'),
        text(510,0.25,'oxygen');
    elseif i==4
        plot(Tt,v), xlabel('T'), ylabel('v'), grid,

```

```

        title('v(m^3/kg) vs. T(K)'), text(460,0.25,'carbon dioxide');
    end
end

```



P2.1. Though atmospheric conditions vary from day to day, it is convenient for design purposes, to have a model for atmospheric properties with altitude. The US Standard Atmosphere, modified in 1976, is such a model. The model consists of two types of regions, one in which the temperature varies linearly with altitude, and the other is a region where the temperature is a constant (see Figure P2.1).

The temperature and approximate pressure relations are as follows:

(a) For a region where the temperature varies linearly

$$p = p_i \left(1 - \frac{\lambda_i (z - z_i)}{T_i} \right)^{\frac{g_i}{\lambda_i R}} \quad (\text{P2.1a})$$

$$T = T_i - \lambda_i (z - z_i) \quad (\text{P2.3b})$$

$$\rho = \frac{p}{RT} \quad (\text{P2.1c})$$

where

z = the altitude, z_i = the altitude at the beginning of the region of interest.

(p_i, T_i) = the pressure and temperature at the beginning of the region of interest.

λ_i = the lapse rate in the region.

R = the air gas constant = $287.0 \frac{N - m}{kg - K}$.

ρ = air density

g_i = the gravitational constant varies slightly with altitude. The above expression for p assumes that within the region of interest, g_i is a constant; otherwise the expression for p would be a lot more complicated than the one shown above.

(b) For a region where the temperature is constant

$$p = p_i \exp \left(- \frac{g (z - z_i)}{RT_i} \right) \quad (\text{P2.1d})$$

$$T = T_i$$

The following table gives the values of pressure, temperature and the gravitational constant at the beginning of each region, as well as the lapse rate of the region.

Table P2.1. US Standard Atmosphere property table.

Regional Properties of US Standard Atmosphere					
Region	z_i (km)	T_i (K)	p_i (Pa)	λ_i (K/m)	g_i (m/s ²)

	0	288.15	101325		9.810
1				0.0065	
	11.0	216.65	22632.05		9.776
2				0.0000	
	20.0	216.65	5474.88		9.749
3				-0.001	
	32.0	228.65	868.02		9.712
4				-0.0028	
	47.0	270.65	110.91		9.666
5				0.0000	
	51.0	270.65	66.94		9.654
6				0.0028	
	71.0	214.65	3.956		9.594
7				0.0020	
	84.9	186.95	0.373		9.553

- (a) Create a MATLAB program using the if-elseif ladder that will determine the property values of (T, p, ρ) for every 1000 m from $z = 0$ (sea level) to $z = 80000 m$. Use one for loop in establishing z .
- (b) In your program, create vectors for the table values of z, T, p, λ and g .
- (c) Assign a character to each region which can be used to determine which equations to be used in determining T, p .
- (d) Save the results as a data file to be used in project P2.2. Note: the data file is to only contain numbers.
- (e) Print the results to a file in table format for every 1000 m .

(f) Plot T vs. z , p vs. z and ρ vs. z as three separate plots, but all on the same page.

The program follows:

```
% P2_1.m

% THIS PROGRAM CALCULATES & TABULATES PRESSURE,TEMPERATURE & DENSITY
% VS. ALTITUDE. The PROGRAM ALLOWS FOR TWO TYPES OF REGIONS:
% NON-ZERO LAPSE RATE & ZERO LAPSE RATE. THE TABLE DATA CONTAINS THE
% NUMBER OF REGIONS AS WELL AS THE ATMOSPHERIC PROPERTIES AT THE
% BEGINNING OF EACH REGION. THE PROGRAM PRINTS OUT ATMOSPHERIC
% PROPERTIES EVERY 1000m UP TO 80000m,
% z=altitude, p= pressure, T=temperature, rho=density, lamda=lapse rate
% reg_type=type of region (zero or non-zero lapse rate)

clear; clc;

zt=[0.0 11.0 20.0 32.0 47.0 51.0 71.0 84.9]*1.0e+3;
Tt=[288.15 216.65 216.65 228.65 270.65 270.65 214.65 186.95];
pt=[101325 22632.05 5474.88 868.02 110.91 66.88 3.956 0.373];
gt=[9.810 9.776 9.749 9.712 9.666 9.654 9.594 9.553];
lamdat=[0.0065 0.0 -0.001 -0.0028 0.0 0.0028 0.0020];
reg_typed= ['a' 'b' 'a' 'a' 'b' 'a' 'a'];

z=zeros(81,1); T=zeros(81,1); p=zeros(81,1); rho=zeros(81,1);
Tc=zeros(81,1);

dz=1000.0; R=287.0;

fo=fopen('output.txt','w');
fo2=fopen('atm.txt','w');

fprintf(fo,'ALTITUDE      PRESSURE      TEMP      DENSITY \n');
fprintf(fo,' (m)          (Pa)          (C)          (kg/m^3) \n');
fprintf(fo,'-----\n');

z(1)=zt(1); T(1)=Tt(1); p(1)=pt(1); rho(1)=p(1)/(R*T(1));
```



```

Tc(1)=T(1)-273.15;

for j=1:81

    z(j)=(j-1)*dz;

    if z(j)>=zt(1) && z(j)< zt(2)

        zi=zt(1);pi=pt(1);Ti=Tt(1);gi=gt(1);lamda=lamdat(1);

        reg_type='a';

    elseif z(j)>=zt(2) && z(j)<zt(3)

        zi=zt(2);pi=pt(2);Ti=Tt(2);gi=gt(2);lamda=lamdat(2);

        reg_type='b';

    elseif z(j)>=zt(3) && z(j)<zt(4)

        zi=zt(3);pi=pt(3);Ti=Tt(3);gi=gt(3);lamda=lamdat(3);

        reg_type='a';

    elseif z(j)>=zt(4) && z(j)<zt(5)

        zi=zt(4);pi=pt(4);Ti=Tt(4);gi=gt(4);lamda=lamdat(4);

        reg_type='a';

    elseif z(j)>=zt(5) && z(j)<zt(6);

        zi=zt(5);pi=pt(5);Ti=Tt(5);gi=gt(5);lamda=lamdat(5);

        reg_type='b';

    elseif z(j)>=zt(6) && z(j)<zt(7)

        zi=zt(6);pi=pt(6);Ti=Tt(6);gi=gt(6);lamda=lamdat(6);

        reg_type='a';

    else

        zi=zt(7);pi=pt(7);Ti=Tt(7);gi=gt(7);lamda=lamdat(7);

        reg_type='a';

    end

    if reg_type == 'a'

        ex=gi/(lamda*R);

        T(j)=Ti-lamda*(z(j)-zi);

        p(j)=pi*(1-lamda*(z(j)-zi)/Ti)^ex;

```

```

end

if reg_type == 'b'

    T(j)=Ti;

    ex=gi*(z(j)-zi)/(R*Ti);

    p(j)=pi*exp(-ex);

end

rho(j)=p(j)/(R*T(j));

Tc(j)=T(j)-273.15;

fprintf(fo,'%6.0f      %12.6e      %10.2f      %12.5e \n',...

        z(j),p(j),Tc(j),rho(j));

%fprintf(fo,'-----\n');

fprintf(fo2,'%6.0f      %12.6e      %10.2f      %12.5e \n',...

        z(j),p(j),Tc(j),rho(j));

end

plot(Tc,z), xlabel('T'), ylabel('z'), title('z vs. T'), grid;

figure;

for i=1:3

    subplot(2,2,i);

    if i==1

        plot(Tc,z), xlabel('T'), ylabel('z'), title('z vs. T'), grid;

    end

    if i==2

        plot(p,z), xlabel('p'), ylabel('z'), title('z vs. p'), grid;

    end

    if i==3

        plot(rho,z), xlabel('rho'), ylabel('z'), title('z vs. rho'),

        grid;

    end

end

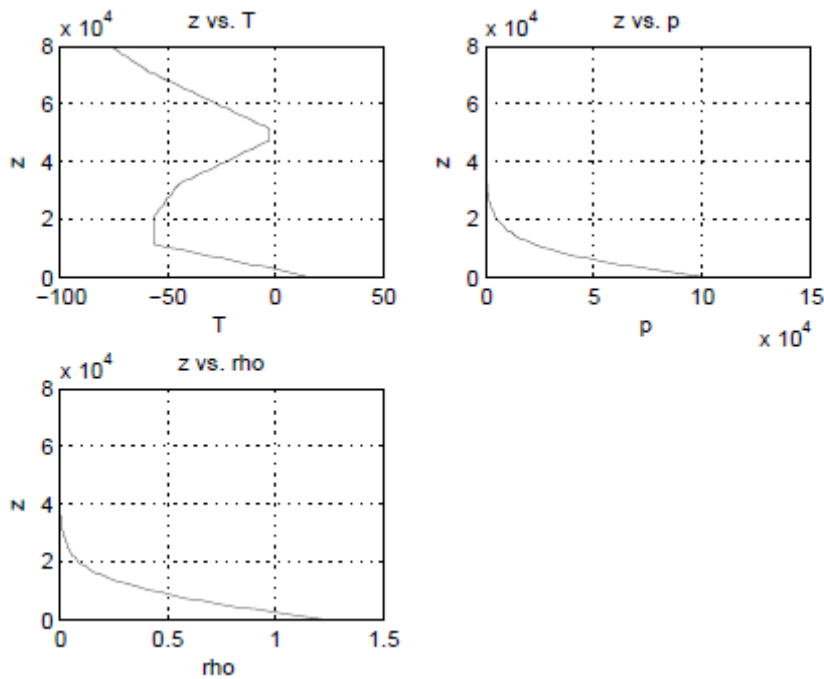
end

```

```
fclose(fo);
```

Program results:

ALTITUDE (m)	PRESSURE (Pa)	TEMP (C)	DENSITY (kg/m^3)
0	1.0133e+05	15.00	1.2252e+00
1000	8.9869e+04	8.50	1.1118e+00
2000	7.9485e+04	2.00	1.0065e+00
3000	7.0095e+04	-4.50	9.0911e-01
4000	6.1624e+04	-11.00	8.1907e-01
5000	5.4002e+04	-17.50	7.3601e-01
6000	4.7162e+04	-24.00	6.5955e-01
7000	4.1041e+04	-30.50	5.8933e-01
8000	3.5580e+04	-37.00	5.2498e-01
9000	3.0723e+04	-43.50	4.6614e-01
10000	2.6418e+04	-50.00	4.1249e-01
11000	2.2632e+04	-56.50	3.6398e-01
12000	1.9339e+04	-56.50	3.1103e-01
13000	1.6526e+04	-56.50	2.6578e-01
14000	1.4121e+04	-56.50	2.2711e-01
15000	1.2067e+04	-56.50	1.9407e-01
.	.	.	.
.	.	.	.
65000	1.0209e+01	-41.70	1.5369e-04
66000	8.8200e+00	-44.50	1.3441e-04
67000	7.6065e+00	-47.30	1.1735e-04
68000	6.5478e+00	-50.10	1.0229e-04
69000	5.6259e+00	-52.90	8.9000e-05
70000	4.8243e+00	-55.70	7.7303e-05
71000	3.9560e+00	-58.50	6.4216e-05
72000	3.3830e+00	-60.50	5.5432e-05
73000	2.8888e+00	-62.50	4.7782e-05
74000	2.4630e+00	-64.50	4.1130e-05
75000	2.0967e+00	-66.50	3.5353e-05
76000	1.7822e+00	-68.50	3.0343e-05
77000	1.5124e+00	-70.50	2.6004e-05
78000	1.2813e+00	-72.50	2.2251e-05
79000	1.0838e+00	-74.50	1.9010e-05
80000	9.1516e-01	-76.50	1.6215e-05



P2.2. Project P2.1 provided a table of temperature, T , pressure, p and density, ρ every 1000 meters of altitude. Atmospheric property values at altitudes between table values can be determined by linear interpolation. Construct a MATLAB program that will load the data file created in Project P2.1 giving (T, p, ρ) every 1000 m . The program is to determine by linear interpolation (Equation 2.43 and the method described in Example 2.13) the properties of (T, p, ρ) at altitudes not in the loaded table. Make the program interactive; that is, the program is to ask the user if he/she wishes to know the temperature, pressure and density at a specific altitude. If the answer is 'Y', the program is to ask the user to enter an altitude from the key board. The program then determines (T, p, ρ) at the entered altitude and prints the results to the screen. The program then asks the user if he/she wishes to enter another altitude. The program is to continue as long as the response to the question is 'Y'.

The Program follows:

```
% P2_2.m

% This program interpolates for atmospheric properties T,p and rho at
% an altitudes entered from the keyboard.

% Atmospheric table data is loaded into the program from the data file
% atm.txt which is created in Project P2.1. The data file contains
% atmospheric properties every 1000 m from z=0 to z=80000 m.

% Properties of temperature, pressure and density at the specified
% altitudes are determined by interpolation and printed to the screen.

% Run the program from the command window by typing in P2_2

clear; clc;

load 'atm.txt'

zt=atm(:,1);

pt=atm(:,2);

Tt=atm(:,3);

rhot=atm(:,4);

char1='Y';

while char1=='Y';

    fprintf('The altitude range is from z=0 m to z=80000 m. \n');

    fprintf('Enter the altitude at which ');

    z=input('the properties are to be determined \n');

    if z < 0 || z > 80000

        fprintf('You entered an altitude out of range, please \n');

        fprintf('restart program \n');

        break;

    end

    for i=1:length(zt)-1
```

```

    if z >= zt(i) && z < zt(i+1)

        T1=Tt(i); T2=Tt(i+1);

        z1=zt(i); z2=zt(i+1);

        T=T1+(z-z1)*(T2-T1)/(z2-z1);

        p1=pt(i); p2=pt(i+1);

        p=p1+(z-z1)*(p2-p1)/(z2-z1);

        rho1=rhot(i); rho2=rhot(i+1);

        rho=rho1+(z-z1)*(rho2-rho1)/(z2-z1);

        fprintf('ALTITUDE      PRESSURE      TEMP      DENSITY \n');
        fprintf('    (m)              (Pa)              (C)      (kg/m^3)\n');
        fprintf('----- \n');
        fprintf('%6i    %12.5e    %10.2f    %12.5e \n',z,p,T,rho);

        break;

    end

end

fprintf('Do you wish to enter another altitude \n');

char1=input('enter Y for yes or N for no \n','s');
end

```

Program results:

The altitude range is from z=0 m to z=80000 m.

Enter the altitude at which the properties are to be determined

3278

ALTITUDE	PRESSURE	TEMP	DENSITY
(m)	(Pa)	(C)	(kg/m^3)

3278	6.77401e+04	-6.31	8.84079e-01

Do you wish to enter another altitude

enter Y for yes or N for no

Y

The altitude range is from $z=0$ m to $z=80000$ m.

Enter the altitude at which the properties are to be determined

57940

ALTITUDE	PRESSURE	TEMP	DENSITY
(m)	(Pa)	(C)	(kg/m ³)

57940	2.73388e+01	-21.93	3.79153e-04

Do you wish to enter another altitude

enter Y for yes or N for no

Y

The altitude range is from $z=0$ m to $z=80000$ m.

Enter the altitude at which the properties are to be determined

76480

ALTITUDE	PRESSURE	TEMP	DENSITY
(m)	(Pa)	(C)	(kg/m ³)

76480	1.65270e+00	-69.46	2.82603e-05

Do you wish to enter another altitude

enter Y for yes or N for no

N

>>

P2.3. The properties of specific volume, v , and pressure, p , as a function of temperature, T , for Carbon Dioxide based on the Redlich-Kwong Equation of state are given in the table below:

Table P2.3. Carbon Dioxide Properties

<i>T</i> (K)	<i>v</i> (m³/kmol)	<i>p</i> (bar)
350	0.28	7.65
400	0.32	8.57
450	0.36	9.16
500	0.40	9.55
550	0.44	9.81
600	0.48	10.00
650	0.52	10.14
700	0.56	10.24
750	0.60	10.31

Note 1 bar = 10^5 N/m²

Determine v and p at temperatures $T2$ where

$$T2 = [367 \ 634 \ 420 \ 587 \ 742]$$

Use interpolation formula of Equation (2.46) and the method described in Example 2.13.

Write a MATLAB program that will do the following:

- Construct 3 separate vectors containing the Carbon Dioxide properties of T , v and p shown in Table P2.3
- Determine v and p at temperatures $T2$ using (Equation 2.43 and the interpolation method described in Example 2.13). The program should not be interactive.

- c. Print to the screen in a table format (with table headings) values of v and p at temperatures T_2 .

The program follows:

```
% P2_3.m

% This program interpolates for specific volume, v, and pressure, p,
% of carbon dioxide at specified temperatures, T2
% The temperature range is from 350K to 750K. The data used in the
% interpolation is in the 3 vectors: Tt,vt,pt
% Temperature,T, is in K, specific volume, v, is in m^3/kmol
% and pressure, p, is in Pa.
% The output is to the screen.

clear; clc;

Tt=350:50:750;

vt=0.28:0.04:0.60;

pt=[7.650 8.574 9.159 9.547 9.813 10.001 10.136 10.236 10.310];

TT = [367 634 420 587 742];

fprintf(' T(K)          v(m^3/kg)          p(bar) \n');

fprintf('----- \n');

for j=1:length(TT)

    for i=1:length(Tt)-1

        if TT(j) >= Tt(i) && TT(j) < Tt(i+1)

            T1=Tt(i); T2=Tt(i+1); v1=vt(i); v2=vt(i+1);

            p1=pt(i); p2=pt(i+1);

            v=v1+(v2-v1)*(TT(j)-T1)/(T2-T1);

            p=p1+(p2-p1)*(TT(j)-T1)/(T2-T1);

            fprintf('%5.1f          %5.3f          %8.4f \n',TT(j),v,p);

            break;

        end

    end

end
```

```

end
end

```

Program results:

```

T(K)      v(m^3/kg)      p(bar)
-----
367.0      0.294          7.9642
634.0      0.507          10.0928
420.0      0.336          8.8080
587.0      0.470          9.9521
742.0      0.594          10.2982
>>

```

P2.4. The positioning of a piston in an internal combustion engine is shown in Figures P2.4(a) and P2.4(b). The piston's position, s , as seen from the crank shaft center can be determined by the Law of cosines; i.e.,

$$b^2 = s^2 + r^2 - 2sr \cos \theta$$

or

$$s^2 - (2r \cos \theta)s + (r^2 - b^2) = 0 \quad (\text{P2.4a})$$

where

b = the length of the piston rod.

r = the radius of the crankshaft.

Equation (P2.4a) is a quadratic equation in s and therefore

$$s = \frac{1}{2} \left(2r \cos \theta + \sqrt{4r^2 \cos^2 \theta - 4(r^2 - b^2)} \right) = r \cos \theta + \sqrt{r^2 (\cos^2 \theta - 1) + b^2}$$

or

$$s = r \cos \theta + \sqrt{b^2 - r^2 \sin^2 \theta} \quad (\text{P2.4b})$$

The piston is constrained to move in the vertical direction and its position, s , varies as the crankshaft rotates. The angle, θ , varies with time, t , and can be expressed in terms of the rotational speed, ω , of the crankshaft. The angle θ is thus given by

$$\theta = 2\pi\omega t \quad (\text{P2.4c})$$

where ω is in revolutions per second. Substituting Equation (P2.4c) into Equation (P2.4b)

gives

$$s(t) = r \cos(2\pi\omega t) + \sqrt{b^2 - r^2 \sin^2(2\pi\omega t)} \quad (\text{P2.4d})$$

The piston velocity, v , can be obtained by taking the derivative of Equation. (P2.4d) with respect to time giving

$$v(t) = -2\pi\omega r \sin(2\pi\omega t) - \frac{2\pi\omega r^2 \sin(2\pi\omega t) \cos(2\pi\omega t)}{\sqrt{b^2 - r^2 \sin^2(2\pi\omega t)}} \quad (\text{P2.4e})$$

The piston acceleration, a , can be obtained by taking the derivative of Equation (P2.4e) with respect to time giving

$$a(t) = -4\pi^2\omega^2 r \cos(2\pi\omega t) - \frac{4\pi^2\omega^2 r^4 \sin^2(2\pi\omega t) \cos^2(2\pi\omega t)}{[b^2 - r^2 \sin^2(2\pi\omega t)]^{3/2}} - \frac{4\pi^2\omega^2 r^2 \cos^2(2\pi\omega t)}{\sqrt{b^2 - r^2 \sin^2(2\pi\omega t)}} + \frac{4\pi^2\omega^2 r^2 \sin^2(2\pi\omega t)}{\sqrt{b^2 - r^2 \sin^2(2\pi\omega t)}} \quad (\text{P2.4f})$$

(a). In MATLAB, create a matrix consisting of s vs. t , v vs. t , and a vs. t for

$0 \leq t \leq 0.01$ seconds. Use 50 subdivisions on the t domain. Take $r = 9 \text{ cm}$,

$\omega = 100$ revolutions per second and $b = 14 \text{ cm}$. Plot s vs. t , v vs. t and a vs. t on three

separate plots, all on the same page.

(b). Using MATLAB's `max` function and the matrix obtained in part (a), determine the approximate maximum velocity and maximum acceleration, and print out those values to the screen..

(c). Plot on a single graph s vs. t for $\omega = [50\ 100\ 150\ 200]$ revolutions per second.

The Program follows:

```
% P2_4.m

% This project involves determining the position, velocity and
% acceleration of a piston in an engine as a function of time, t, where
% 0 <= t <= 0.1 s. 50 sub-divisions on the t domain are to be used.
% Problem parameters are: r=9 cm, omega = 100 rev per second
% and b=14 cm.

clear; clc;

r=9; omega=100; b=14; dt=0.0002;

for i=1:51

    t(i)=(i-1)*dt;

    arg1=2*pi*omega*t(i);

    arg2=b^2-r^2*(sin(arg1))^2;

    s(i)=r*cos(arg1)+sqrt(arg2);

    v(i)=-2*pi*omega*r*sin(arg1)- ...

        (2*pi*omega*r^2*sin(arg1)*cos(arg1))/sqrt(arg2);

    % Decompose a into 4 parts, a1, a2, a3, a4

    a1=-4*pi^2*omega^2*r*cos(arg1);

    a2num=-4*pi^2*omega^2*r^4*(sin(arg1))^2*(cos(arg1))^2;

    a2den=arg2^1.5;

    a2=a2num/a2den;

    a3=-4*pi^2*omega^2*r^2*(cos(arg1))^2/sqrt(arg2);
```

```

    a4=4*pi^2*omega^2*r^2*(sin(arg1))^2/sqrt(arg2);

    a(i)=a1+a2+a3+a4;

    end

plot(t,s), xlabel('t(s)'), ylabel('s(cm)'), title('s vs. t'), grid;
figure;
plot(t,v), xlabel('t(s)'), ylabel('v(cm/s)'), title('v vs. t'), grid;
figure;
plot(t,a), xlabel('t(s)'), ylabel('a(cm/s^2)'), title('a vs. t'), grid;

% Using MATLAB's max function to obtain the maximum velocity and
% maximum acceleration

vmax=max(v); amax=max(a);

fprintf('The maximum velocity =%8.1f(cm/s) \n',vmax);
fprintf('The maximum acceleration=%12.3e(cm/s^2) \n',amax);

% Program for part (c) follows:

% This project involves determining the position, velocity and
% acceleration of a piston in an engine as a function of time, t, where
% 0 <= t <= 0.1 s.

% we are to use 50 sub-divisions on the t domain.

% Problem parameters are: r=9 cm and omega = 100 rev. per second. b=14
cm.

clear; clc;

r=9; omega=100; b=14; dt=0.0002;

for j=1:4

    omegat = [50 100 150 200];

    omega=omegat(j);

    for i=1:51

        t(i)=(i-1)*dt;

        arg1=2*pi*omega*t(i);

        arg2=b^2-r^2*(sin(arg1))^2;

```

```

        s(i)=r*cos(arg1)+sqrt(arg2);
    end

    if j==1
        s1=s;
    elseif j==2
        s2=s;
    elseif j==3
        s3=s;
    elseif j==4
        s4=s;
    end

end

plot(t,s1,'y',t,s2,'r',t,s3,'b',t,s4,'g'), xlabel('t(s)'),
ylabel('s(cm)'),
title('s vs. t'), grid;
text(0.0028,19,'s1'),text(0.0027,11,'s2'),text(0.007,19,'s3'),
text(0.0012,7,'s4');

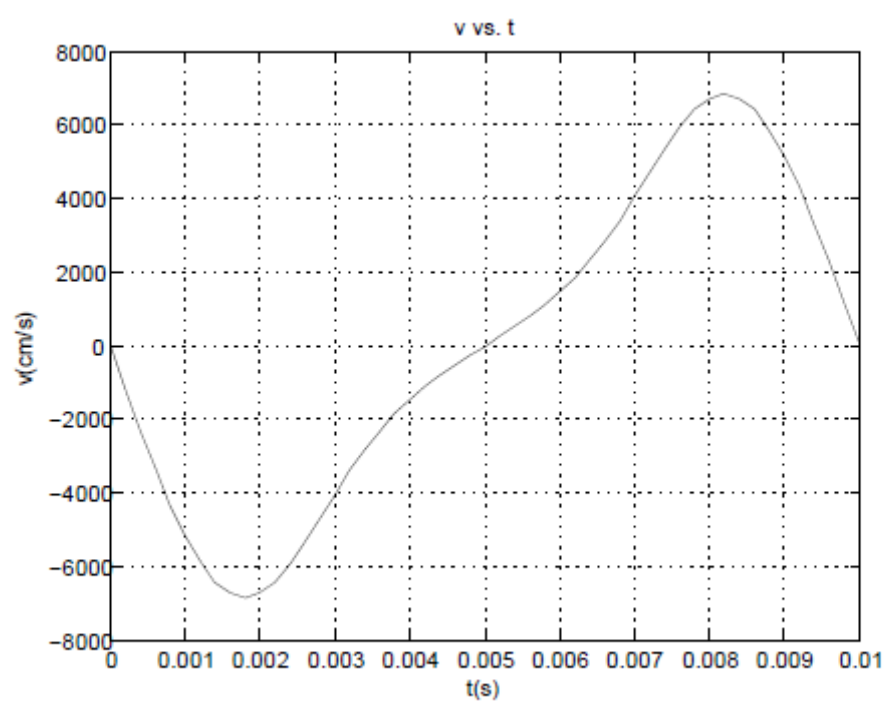
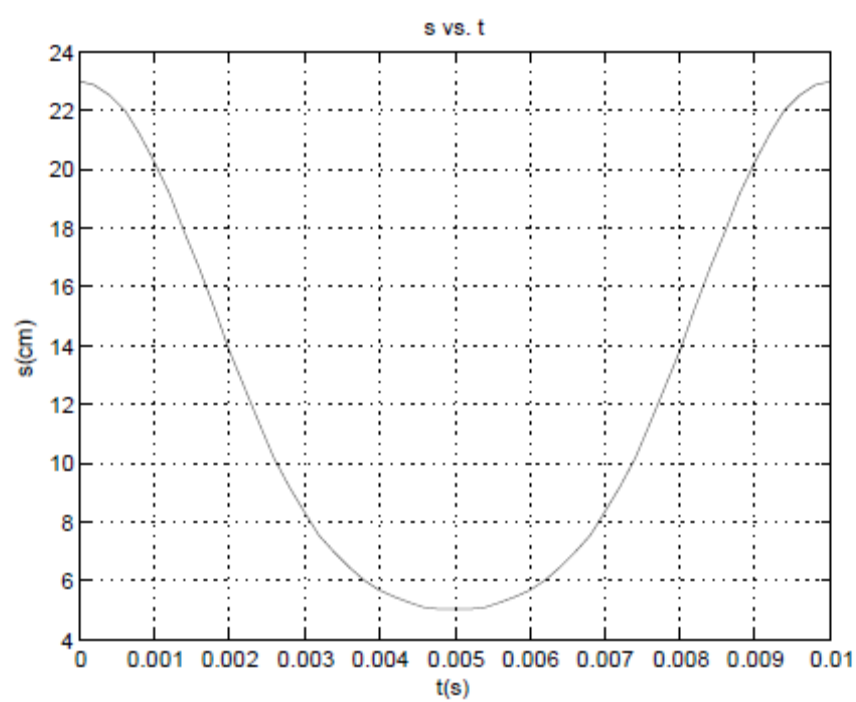
```

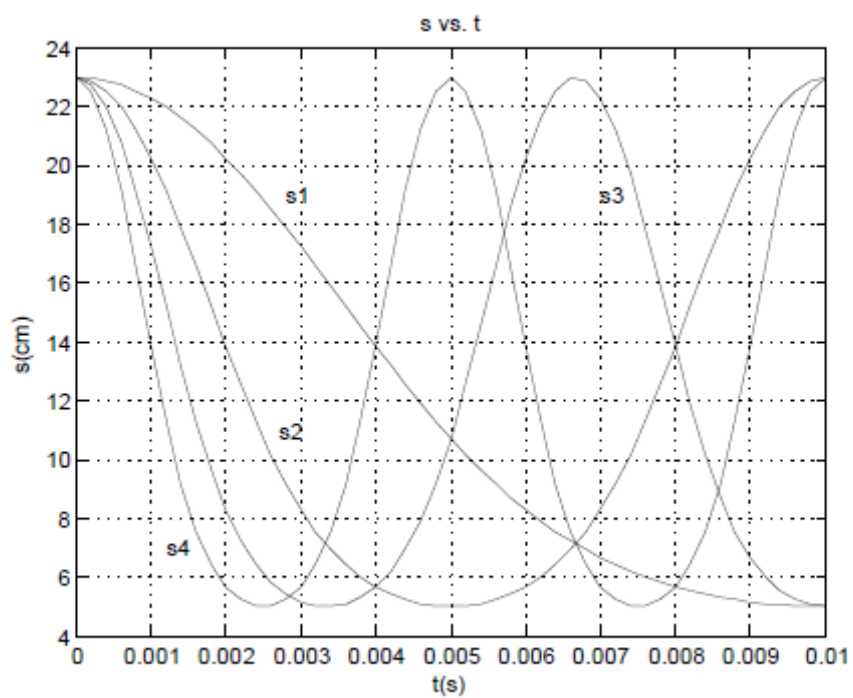
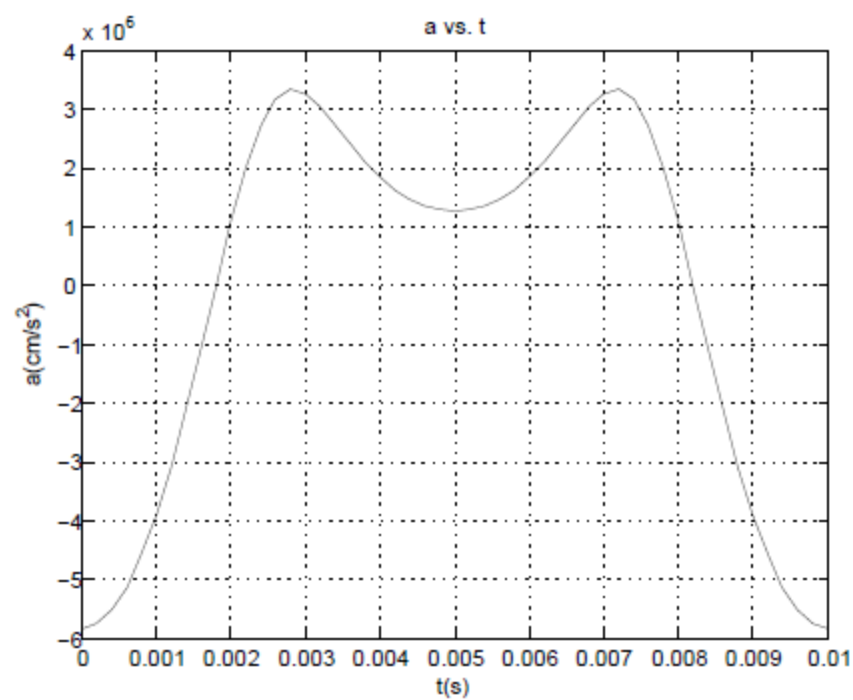
Program results:

```

The maximum velocity = 6838.4(cm/s)
The maximum acceleration= 3.336e+06(cm/s^2)
>>

```





P2.5. (a). The project is to plot the several motions that can occur with a mass-spring-dashpot system. The following parameters were specified:

$$m = 75 \text{ kg}, \quad k = 87.5 \frac{\text{N}}{\text{m}}, \quad c = 875 \frac{\text{N-s}}{\text{m}}$$

$$\left(\frac{c}{2m} \right)^2 = \left(\frac{875}{2(75)} \right)^2 = 34.03, \quad \frac{k}{m} = \frac{87.5}{75} = 1.167$$

The system is *over-damped*.

The governing equation for the *over-damped* system is:

$$y = \exp\left(-\frac{c}{2m}t\right) \left\{ A \exp\left(\sqrt{\left(\frac{c}{2m}\right)^2 - \frac{k}{m}}t\right) + B \exp\left(-\sqrt{\left(\frac{c}{2m}\right)^2 - \frac{k}{m}}t\right) \right\}$$

Three initial conditions are considered:

1. $y(0) = 0.5m, \quad y'(0) = 1.0 \frac{m}{s}.$
2. $y(0) = 0.5m, \quad y'(0) = -1.0 \frac{m}{s}.$
3. $y(0) = 0.5m, \quad y'(0) = 0 \frac{m}{s}.$

For each case the project is to:

- c. determine $y(t)$ for $0 \leq t \leq 10$ seconds in steps of 0.1 seconds.
- d. print out a Table of y vs. t every 1 second.
- e. plot y vs. t for all three cases on the same graph. Each curve is to be labeled with the value of y' .

The Program follows:

```
% P2_5.m
% This program solves project P2.5 in the textbook
```

```

% Need to determine the general solution constants for
% the specified initial conditions.

%  $y(0) = A+B$ 

%  $\arg = (c/2/m)^2 - k/m$ 

%  $y'(0) = -c/(2m)*(A+B) + \sqrt{\arg}*(A-B)$ 

clear; clc;

% case (a)

% case 1.  $y(0)=0.5$  m,  $y'(0)=0.25$  m/s
% case 2.  $y(0)=0.5$  m,  $y'(0)=-1.0$  m/s
% case 3.  $y(0)=0.5$  m,  $y'(0)=0$  m/s

% m is in kg, k is in N/m and c is in N-s/m
m=75.0; k=87.5; c=875.0;

arg=(c/2/m)^2-k/m;

sq=sqrt(arg);

dt=0.1;

d=zeros(2,1);

for i=1:3

    if i==1

        y0=0.5; yprime0=1.0;

    elseif i==2

        y0=0.5; yprime0=-1.0;

    elseif i==3

        y0=0.5; yprime0=0;

    end

    a(1,1)=1.0; a(1,2)=1.0;

    a(2,1)=sqrt(arg)-c/2/m;

    a(2,2)=-(sqrt(arg)+c/2/m);

    % Solution of A(i) by the method of determinants

    num=a(2,2)*y0-a(1,2)*yprime0;

```

```

den=a(1,1)*a(2,2)-a(1,2)*a(2,1);

A(i)=num/den;

% Solution of B(i)by the method of determinants

num=a(1,1)*yprime0-a(2,1)*y0;

B(i)=num/den;

fprintf('case= %2i   A=%10.4f   B=%10.4f \n',i,A(i),B(i));

for j=1:101

    t(j)=(j-1)*dt;

    t1=t(j);

    if i==1

        y1(j)=exp(-c/2*t1/m)*(A(i)*exp(sq*t1)+B(i)*exp(-sq*t1));

    elseif i==2

        y2(j)=exp(-c/2*t1/m)*(A(i)*exp(sq*t1)+B(i)*exp(-sq*t1));

    elseif i==3

        y3(j)=exp(-c/2*t1/m)*(A(i)*exp(sq*t1)+B(i)*exp(-sq*t1));

    end

end

end

end

for i=1:3

    fprintf(' t           y \n');

    fprintf('-----\n');

    fprintf('i=%3i \n',i);

    for j=1:10:101

        if i==1

            fprintf('%5.1f   %10.4f \n',t(j),y1(j));

        elseif i==2

            fprintf('%5.1f   %10.4f \n',t(j),y2(j));

        elseif i==3


```

```

        fprintf('%5.1f      %10.4f  \n',t(j),y3(j));
    end

end

fprintf('\n\n');

end

plot(t,y1,t,y2,'--',t,y3,'-.'), xlabel('t'),ylabel('y'),grid,

title('y vs. t')

text(6.3,0.325,'y ' ' = 1.0'),
text(2.5,0.275,'y ' ' = -1.0'),
text(0.6,0.425,'y ' ' = 0');

```

Program Results:

```

case=  1      A=  0.5916      B= -0.0916
case=  2      A=  0.4172      B=  0.0828
case=  3      A=  0.5044      B= -0.0044

```

t	y

i= 1	
0.0	0.5000
1.0	0.5349
2.0	0.4835
3.0	0.4371
4.0	0.3952
5.0	0.3573
6.0	0.3230
7.0	0.2920
8.0	0.2640
9.0	0.2387
10.0	0.2158

t	y

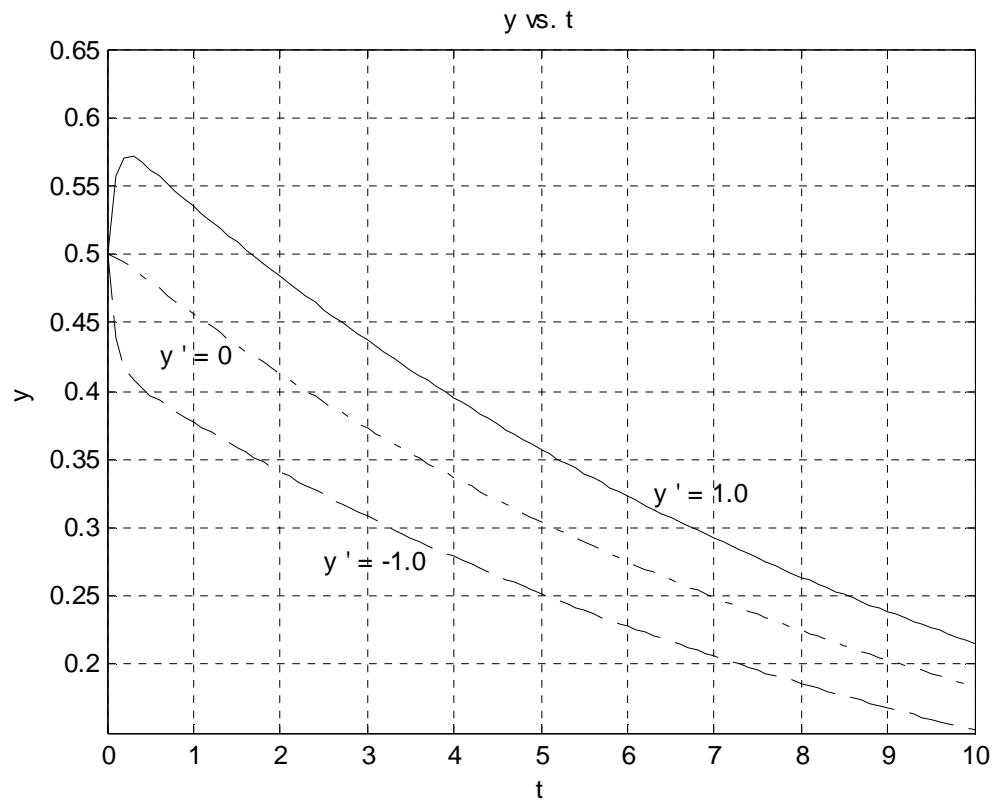
i= 2	
0.0	0.5000
1.0	0.3771
2.0	0.3410
3.0	0.3082
4.0	0.2787
5.0	0.2519
6.0	0.2278
7.0	0.2059
8.0	0.1861

9.0	0.1683
10.0	0.1521

t	y
---	---

i= 3

0.0	0.5000
1.0	0.4560
2.0	0.4122
3.0	0.3727
4.0	0.3369
5.0	0.3046
6.0	0.2754
7.0	0.2490
8.0	0.2251
9.0	0.2035
10.0	0.1839



P2.5 (b). The project is to plot the motion of the mass in a mass-spring-dashpot system when the system is *under-damped*. The plot is also to include the envelope of the motion.

The parameters of the system are:

$$m = 25 \text{ kg}, \quad k = 200 \frac{\text{N}}{\text{m}}, \quad c = 5 \frac{\text{N-s}}{\text{m}}$$

$$y(0) = 5 \text{ m}, \quad y'(0) = 0 \frac{\text{m}}{\text{s}} \quad \text{and} \quad 0 \leq t \leq 40.$$

The y position of the mass is given by

$$y = \exp\left(-\frac{c}{2m}t\right) \left\{ A \cos\left(\sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} t\right) + B \sin\left(\sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2} t\right) \right\}$$

and the equation of the envelope is:

$$y = \pm A \exp\left(-\frac{c}{2m}t\right)$$

For the specified initial conditions, $A = 5 \text{ m}$ and $B = \frac{c}{2m} \times \frac{A}{\sqrt{\frac{k}{m} - \left(\frac{c}{2m}\right)^2}}$

The Program follows:

```
% P2_5b.m

% This program solves for the position, y, of a mass attached to
% a spring-dashpot system. First we need to determine whether the
system
% is over-damped or under-damped. The arbitrary constants in the
general
% solution is determined by the specified initial conditions.
% The constants for the system are: m=25kg, k=200 N/m, c=5 N-s/m.
% k/m - c/2m)^2= 8, thus the system is under-damped.
% arg=sqrt(k/m-(c/2/m)^2)
% The general solution is
% y=exp(-c/2*t/m)*(A*cos(arg*t)+B*sin(-arg*t));
```

```

% y(0)=5.0 m, y'(0)=0 m/s

% A=5, B=c/(2*m)*A/arg

% m is in kg, k is in N/m and c is in N-s/m

clear; clc;

m=25.0; k=200.0; c=5.0;

arg=sqrt(k/m-(c/2/m)^2); arg2=c/(2*m)/arg; arg3=c/(2*m);

A=5.0; B=A*arg2;

dt=0.1

for j=1:401

    t(j)=(j-1)*dt;

    t1=t(j);

    y1(j)=exp(-c/(2*m)*t1)*(A*cos(arg*t1)+B*sin(arg*t1));

    y2(j)=A*exp(-arg3*t1);

    y3(j)=-A*exp(-arg3*t1);

end

fprintf(' t          y \n');

fprintf('-----\n')

for j=1:10:401

    fprintf('%5.1f    %10.4e \n',t(j),y1(j));

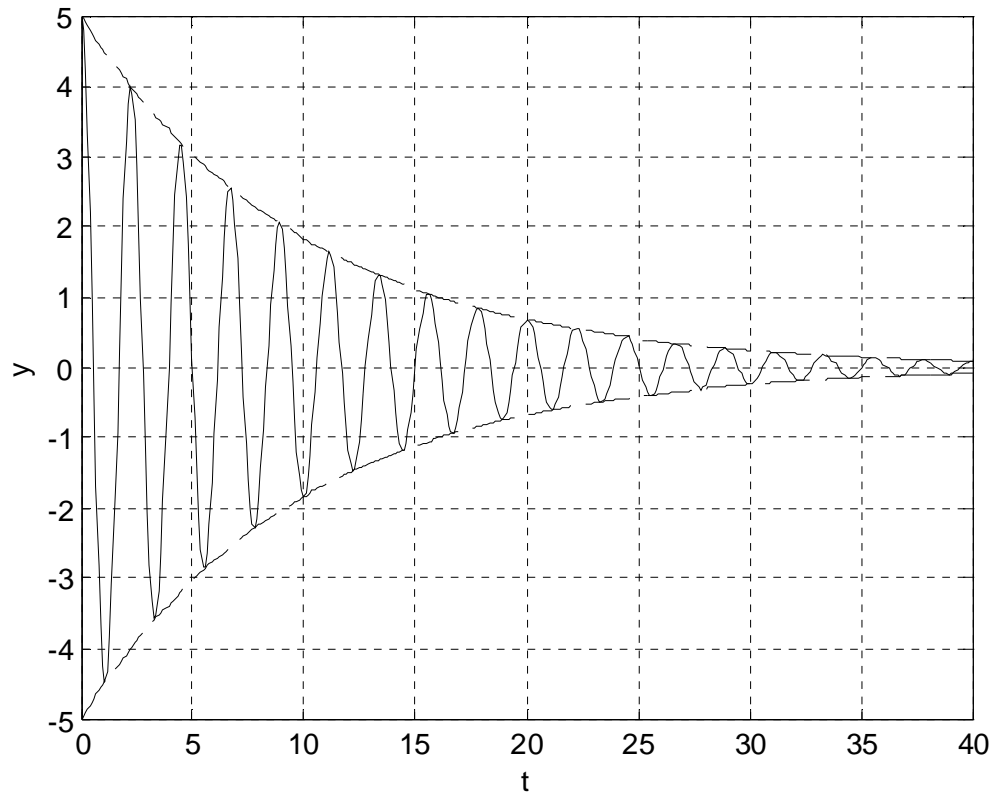
end

plot(t,y1,t,y2,'--',t,y3,'--'), xlabel('t'),ylabel('y'),grid,

-----

```

Program Results:



P2.6. In this project we use a mass-spring-dashpot system subjected to an oscillatory driving force to demonstrate the concept of resonance. Resonance is a phenomenon in which the response amplitude to the driving force of a particular frequency is much larger than at other frequencies. The governing equation for this system is:

$$y'' + \frac{c}{m} y' + \frac{k}{m} y = \frac{F_0}{m} \sin \omega t$$

The particular solution to this equation is

$$y_p = \frac{F_0}{k} \frac{1}{\sqrt{\left(1 - \frac{\omega}{\omega_n}\right)^2 + \left(2\zeta \frac{\omega}{\omega_n}\right)^2}} \sin(\omega t - \phi)$$

where

$$\omega_n = \sqrt{\frac{k}{m}} \quad , \quad \zeta = \frac{c}{c_c}, \quad c_c = 2m\omega_n$$

The term $\frac{F_0}{k} \frac{1}{\sqrt{\left(1 - \frac{\omega}{\omega_n}\right)^2 + \left(2\zeta \frac{\omega}{\omega_n}\right)^2}}$ is the amplitude of the steady state oscillation.

The larger the term $\frac{1}{\sqrt{\left(1 - \frac{\omega}{\omega_n}\right)^2 + \left(2\zeta \frac{\omega}{\omega_n}\right)^2}}$, the larger is the amplitude.

$$\text{Let } \text{ampl} = \frac{1}{\sqrt{\left(1 - \frac{\omega}{\omega_n}\right)^2 + \left(2\zeta \frac{\omega}{\omega_n}\right)^2}}.$$

Construct a MATLAB program to create a plot of ampl vs. ω/ω_n for values

of $\zeta = 1.0, 0.5, 0.25, 0.10, 0.05$, and $0 < \omega/\omega_n < 2$ in steps of 0.01, where

The program follows:

```
% P2_6.m

% This program creates plots of ampl vs. w/wn for several values of c.

% ampl=1/sqrt(arg), where arg=(1-w/wn)^2+(2*c*w/wn)^2

clear; clc;

cm=[1.0 0.5 0.25 0.10 0.05];

dw=0.01;

ampl=zeros(201,5);

wr=zeros(201,1);

for i=1:5

    c=cm(i);

    for j=1:201
```

```

        wr(j)=(j-1)*dw;

        arg=(1.0-wr(j))^2+(2*c*wr(j))^2;

        ampl(j,i)=1/sqrt(arg);

    end

    if i==1

        curve1=ampl(:,i);

    elseif i==2

        curve2=ampl(:,i);

    elseif i==3

        curve3=ampl(:,i);

    elseif i==4

        curve4=ampl(:,i);

    elseif i==5

        curve5=ampl(:,i);

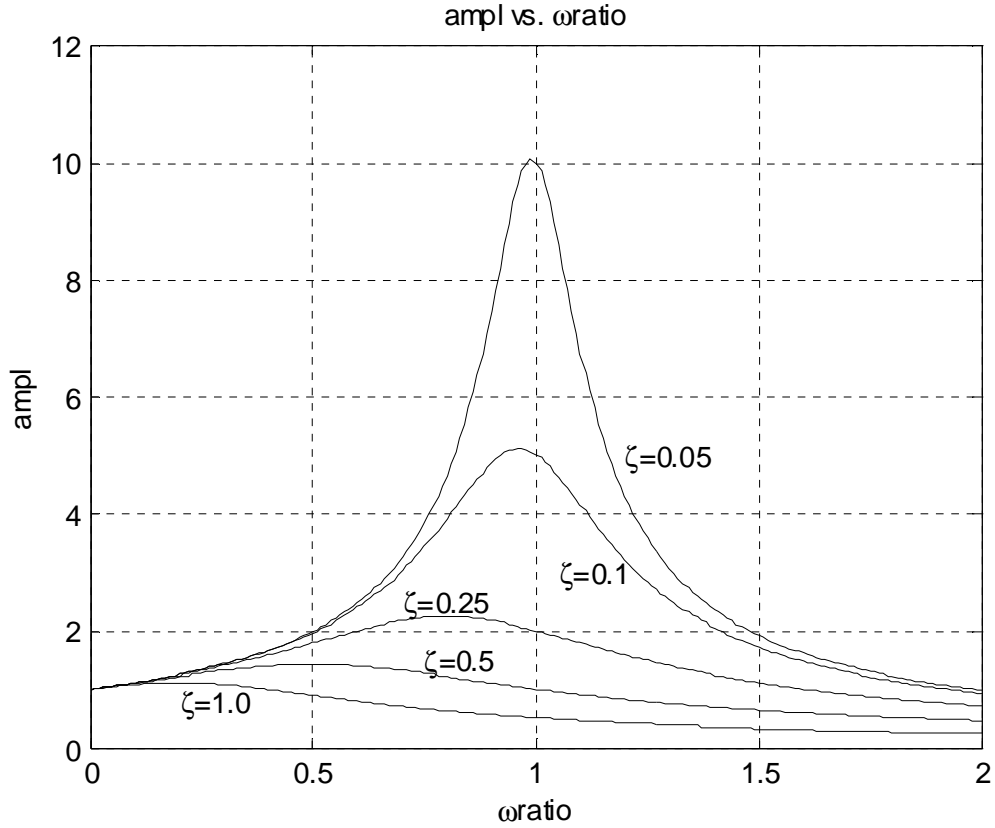
    end

end

plot(wr,curve1,wr,curve2, wr,curve3,wr,curve4,wr,curve5), grid,
xlabel('\omegaratio'), ylabel('ampl'),title('ampl vs. \omegaratio'),
text(0.2,0.8,'\zeta=1.0'),text(0.75,1.5,'\zeta=0.5'),text(0.7,2.5,'\zet
a=0.25'),
text(1.05,3.0,'\zeta=0.1'),text(1.2,5.0,'\zeta=0.05');

```

Program Results:



P2.7. The RLC circuit is a fundamental problem in electrical engineering circuit [4]. In this project we consider the parallel RLC circuit shown in Figure P2.7 and derive the governing equation for the voltage across each circuit component when at $t = 0$, the switch is opened.

We start the analysis with the constituent voltage-current relations for resistors, inductors, and capacitors:

$$\text{Resistor (Ohm's Law):} \quad v_R = R i_R \quad (\text{P2.7a})$$

$$\text{Inductor:} \quad v_L = L \frac{d i_L}{dt} \quad (\text{P2.7b})$$

Capacitor:
$$i_C = C \frac{dv_C}{dt} \quad (\text{P2.7c})$$

where R , L , and C are the component values for the resistor (in ohms), inductor (in henries) and capacitor (in farads), and the voltages v and currents i are as defined in the Figure P2.7. In addition, we assume that the stateful circuit elements L and C have known initial conditions $i_L(0)$ and $v_C(0)$ at the moment that the switch is opened.

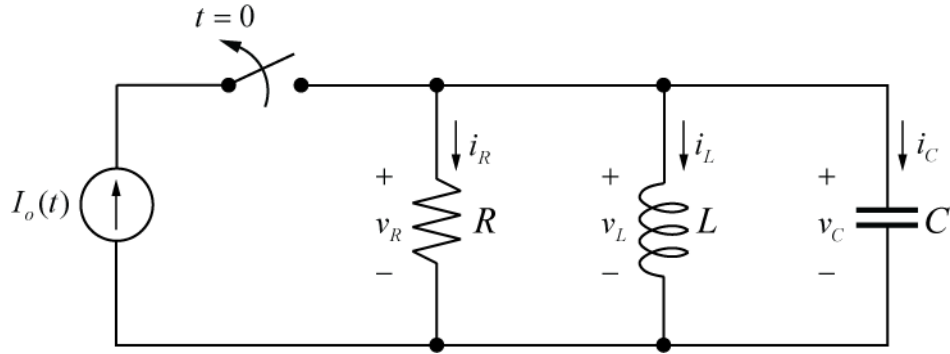


Figure P2.7. Parallel RLC circuit

Kirchhoff's Current Law (KCL) states that the sum of currents at any circuit node is zero. Applying KCL at the top node in the circuit gives

$$i_R + i_L + i_C = 0 \quad (\text{P2.7d})$$

Also, the parallel topology of the circuit gives

$$v_R = v_L = v_C \equiv v \quad (\text{P2.7e})$$

Differentiating Equation (P2.7d) gives

$$\frac{di_R}{dt} + \frac{di_L}{dt} + \frac{di_C}{dt} = 0 \quad (\text{P2.7f})$$

Substituting Equation (P2.7e) into Equation (P2.7a) and differentiating gives

$$\frac{di_R}{dt} = -\frac{1}{R} \frac{dv}{dt} \quad (\text{P2.7g})$$

Substituting Equation (P2.7e) into Equation (P2.7b) gives

$$\frac{di_L}{dt} = \frac{1}{L} v \quad (\text{P2.7h})$$

Substituting Equation (P2.7e) into Equation (P2.7c) gives

$$\frac{dv}{dt} = \frac{i_c}{C} \quad (\text{P2.7i})$$

Differentiating Equation (P2.7i) gives and rearranging

$$\frac{di_c}{dt} = C \frac{d^2 v}{dt^2} \quad (\text{P2.7j})$$

Substituting Equations (P2.7g), (P2.7h) and (P2.7j) into Equation P2.7f) gives

$$\frac{d^2 v}{dt^2} + \frac{1}{RC} \frac{dv}{dt} + \frac{1}{LC} v = 0 \quad (\text{P2.7k})$$

To solve this homogeneous differential equation, we seek a function $v(t)$ such that the

derivatives $\frac{dv}{dt}(t)$ and $\frac{d^2 v}{dt^2}(t)$ reproduce the form of $v(t)$. A function that satisfies this

condition is $Ae^{\alpha t}$, where A and α are arbitrary constants. If we assume that $v = Ae^{\alpha t}$,

then

$$\frac{dv}{dt}(t) = \alpha Ae^{\alpha t} \quad \text{and} \quad \frac{d^2 v}{dt^2}(t) = \alpha^2 Ae^{\alpha t}$$

Substituting these terms into the Differential Equation (P2.7k) gives:

$$\left(\alpha^2 + \frac{1}{RC} \alpha + \frac{1}{LC} \right) A e^{\alpha t} = 0 \quad (\text{P2.7L})$$

Since $e^{\alpha t} \neq 0$, then

$$\left(\alpha^2 + \frac{1}{RC} \alpha + \frac{1}{LC} \right) = 0 \quad (\text{P2.7m})$$

The solutions are:

$$\alpha = -\frac{1}{2RC} \pm \sqrt{\left(\frac{1}{2RC}\right)^2 - \frac{1}{LC}} \quad (\text{P2.7n})$$

Thus, there are two solutions for α , and there are two solution that satisfy the differential equation (P2.7k). The general solution to Equation (P2.7k) is the sum of all known solutions:

$$v = A \exp\left(-\frac{1}{2RC}t + \sqrt{\left(\frac{1}{2RC}\right)^2 - \frac{1}{LC}} t\right) + B \exp\left(-\frac{1}{2RC}t - \sqrt{\left(\frac{1}{2RC}\right)^2 - \frac{1}{LC}} t\right)$$

or

$$v = \exp\left(-\frac{1}{2RC}t\right) \left\{ A \exp\left(\sqrt{\left(\frac{1}{2RC}\right)^2 - \frac{1}{LC}} t\right) + B \exp\left(-\sqrt{\left(\frac{1}{2RC}\right)^2 - \frac{1}{LC}} t\right) \right\} \quad (\text{P2.7o})$$

where $\exp(x) = e^x$ and A and B are constants which depend on the initial conditions of the circuit. Note that this solution has three regions of interest:

- a) *Over-damped*: if $\left(\frac{1}{2RC}\right)^2 > \frac{1}{LC}$, then the solutions are decaying exponentials over time.
- b) *Under-damped*: if $\left(\frac{1}{2RC}\right)^2 < \frac{1}{LC}$, then the solutions are decaying sinusoids over time.

For the under-damped case, we can show the sinusoidal behavior by applying the

identities $e^{jx} = \cos x + j \sin x$ and $e^{-jx} = \cos x - j \sin x$ to Equation (P2.7o) giving:

$$v = \exp\left(-\frac{1}{2RC}t\right) \left\{ A \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2} t\right) + B \sin\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2} t\right) \right\} \quad (\text{P2.7p})$$

where the coefficients A and B are to be determined by initial conditions.

(c) If $\left(\frac{1}{2RC}\right)^2 = \frac{1}{LC}$, then the square root term is zero and the system is said to be

critically damped. For this case, the solution is

$$v = (A + Bt) \exp\left(-\frac{1}{2RC}t\right) \quad (\text{P2.7q})$$

We now consider the under-damped case, Equation (P2.7p) where A and B are constants to be determined by initial conditions and have the units of volts. For the component values

$R = 100 \Omega$, $L = 1 \text{ mH}$, $C = 1 \mu\text{F}$ and initial conditions $i_L(0) = 0$ and $v_C(0) = 6 \text{ V}$:

- Solve for the coefficients A and B .
- Create a MATLAB program that will calculate $v(t)$ for $0 \leq t \leq 500 \mu\text{s}$ in steps of $5 \mu\text{s}$.
- Plot $v(t)$ vs. t for $0 \leq t \leq 500 \mu\text{s}$ in steps of $5 \mu\text{s}$.

Solution:

To solve for A and B , we must apply the initial conditions to Equation P2.7p. Evaluating Equation P2.7p at $t = 0$ gives $A = v(0)$. To apply the second initial condition, we know from Equation P2.7d that $i_L = -i_R - i_C$, and then applying Equations P2.7a and

P2.7c, we know $i_L = -\frac{v}{R} - C \frac{dv}{dt}$. Differentiating Equation P2.7p gives

$$\frac{dv}{dt} = e^{K_1 t} \left((K_1 B - K_2 A) \sin K_2 t + (K_1 A + K_2 B) \cos K_2 t \right)$$

where we define constants $K_1 = -\frac{1}{2RC}$ and $K_2 = \sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2}$.

Thus,

$$i_L = -\frac{v}{R} - Ce^{K_1 t} \left((K_1 B - K_2 A) \sin K_2 t + (K_1 A + K_2 B \cos K_2 t) \right)$$

At $t = 0$, $i_L(0) = -\frac{v(0)}{R} - C(K_1 A + K_2 B)$ and thus $B = -\frac{\frac{i_L(0)}{C} + \frac{v(0)}{RC} + K_1 A}{K_2}$.

The program follows:

```
% Project_2_7.m

% This script determines the values of the coefficients A and B
% based on the initial conditions.

% R=100 ohms; L=1e-3 henry; C=1e-6 Farads;

% v(0)=vc(0)=6 V

% iL(0)=0 A

clear; clc;

R=100; L=1e-3; C=1e-6; v0=6; iL0=0;

% PART A: Calculate coefficients

K1 = -1/(2*R*C);

K2 = sqrt( 1/(L*C) - (1/(2*R*C))^2 );

A=v0;

B= -(iL0/C + v0/(R*C) + K1*A) / K2;

fprintf('Coefficients: A=%.4f, B=%.4f\n\n', A,B);

% PART B: Calculate v(t) for t=[0,500 microsec] in steps of 5 microsec

fprintf('    t(microsec)        v(volts)    \n');

fprintf('-----\n');

dt=5.0e-6;

for i=1:101

    t(i)=(i-1)*dt;

    v(i)=exp(K1*t(i))*(A*cos(K2*t(i))+B*sin(K2*t(i)));

    fprintf('%10.2f        %10.4f\n', t(i)*1e6,v(i));

end
```



```
% PART C: Plot v vs t

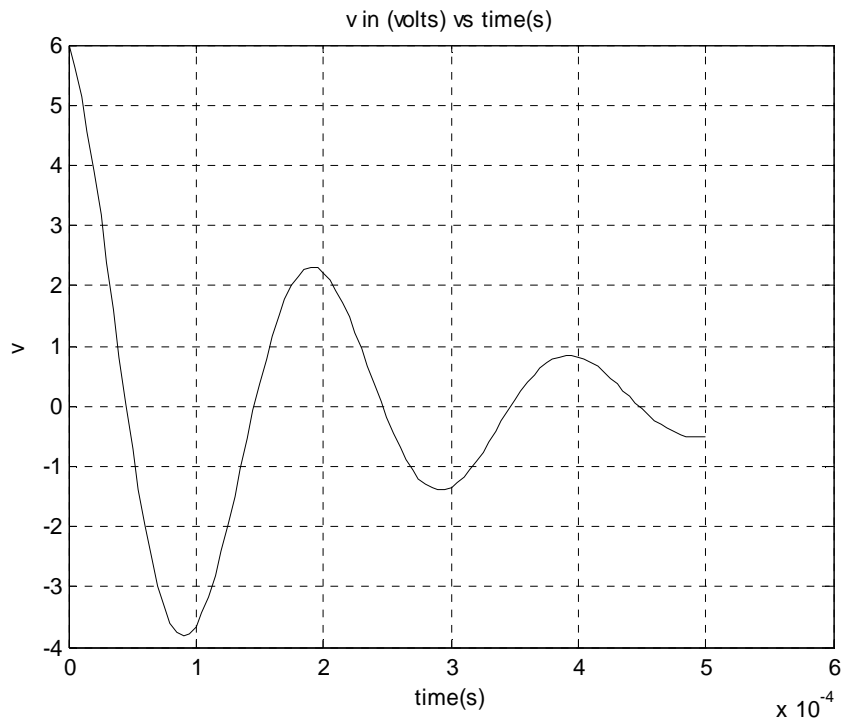
plot(t,v), xlabel('time(s)'), ylabel('v'), grid,

title('v in (volts) vs time(s)');
```

Program results:

Coefficients: A=6.0000, B=-0.9608

t(microsec)	v(volts)
<hr/>	
0.00	6.0000
5.00	5.6350
10.00	5.1506
15.00	4.5646
20.00	3.8960
25.00	3.1653
⋮	⋮
475.00	-0.4259
480.00	-0.4667
485.00	-0.4941
490.00	-0.5082
495.00	-0.5092
500.00	-0.4978



P2.8. Using Equations P2.7a through P2.7e, show that the governing differential equation for the inductor current, i_L , in the parallel RLC circuit of Figure P2.7 is given by

$$\frac{d^2 i_L}{dt^2} + \frac{1}{RC} \frac{di_L}{dt} + \frac{1}{LC} i_L = 0 \quad (\text{P2.8a})$$

Comparing Equations P2.8a and P2.7k, we see that the equations are alike, with i_L replacing v , and therefore, they have similar solutions. For the under-damped case,

$$i_L = \exp\left(-\frac{1}{2RC}t\right) \left\{ \alpha \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2} t\right) + \beta \sin\left(\sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2} t\right) \right\} \quad (\text{P2.8b})$$

For the component values $R = 100 \Omega$, $L = 1 \text{ mH}$, $C = 1 \mu\text{F}$ and initial conditions $i_L(0) = 0$ and $v_C(0) = 6 \text{ V}$:

- Solve for the coefficients α and β .
- Plot $i_L(t)$ for $0 \leq t \leq 500 \mu\text{sec}$ in steps of $5 \mu\text{sec}$
- Plot $i_L(t)$ and $v_C(t)$ (solved in Project 2.7) on the same axes. Note that when v_C is at a maximum, i_L is zero and vice versa. This indicates how the energy in the circuit oscillates back and forth between the capacitor and inductor.

Solution:

Differentiating both sides of Equation P2.7b gives

$$\frac{dv}{dt} = L \frac{d^2 i}{dt^2}$$

Substituting into Equation P2.7c gives

$$i_C = LC \frac{d^2 i}{dt^2}$$

Also, substituting Equation P2.7b into P2.7a gives

$$i_R = \frac{L}{R} \frac{di_L}{dt}$$

Substituting the above two equations into Equation P2.7d gives

$$\frac{L}{R} \frac{di_L}{dt} + i_L + LC \frac{d^2 i_L}{dt^2} = 0$$

Rearranging gives

$$\frac{d^2 i_L}{dt^2} + \frac{1}{RC} \frac{di_L}{dt} + \frac{1}{LC} i_L = 0$$

Evaluating Equation P2.8b at $t=0$ gives $\alpha = i_L(0)$. To apply the second initial condition

for $v(0)$, apply Equation P2.7b:

$$v = L \frac{di_L}{dt} = L e^{K_1 t} ((K_1 \beta - K_2 \alpha) \sin K_2 t + (K_1 \alpha + K_2 \beta) \cos K_2 t)$$

where we define $K_1 = -\frac{1}{2RC}$ and $K_2 = \sqrt{\frac{1}{LC} - \left(\frac{1}{2RC}\right)^2}$. At $t=0$, we get

$$v(0) = L(K_1 \alpha + K_2 \beta), \text{ and thus } \beta = \frac{\frac{v(0)}{L} - K_1 \alpha}{K_2}.$$

The program follows.

```
% P2_8.m

% This script determines the values of the arbitrary coefficients,
% alpha and beta for Project 2.8. It also calculates A and B from
% Project 2.7. It then plots i and v on the same axes.

% R=100 ohms; L=1e-3 henry; C=1e-6 Farads;

% The system is under damped [1/LC > 1/(2*R*C)^2 ];

% v(0)=6 V

% iL(0)=0.0 A

clear; clc;

R=100; L=1e-3; C=1e-6; v0=6; iL0=0.0;
```

```

% PART A: solve for alpha, beta

K1 = -1/(2*R*C);

K2 = sqrt( 1/(L*C) - (1/(2*R*C))^2 );

alpha=iL0;

beta= (v0/L - K1*alpha)/K2;

fprintf('Coefficients: alpha=%.4f, beta=%.4f\n\n', alpha,beta);

% From Project 2.7:

A=v0;

B= -(iL0/C + v0/(R*C) + K1*A) / K2;

% PART B: calculate iL(t) and print:

fprintf('  t(microsec)  iL(amp)    v(volt)    \n');

fprintf('-----\n');

dt=5e-6;

for j=1:101

    t(j)=(j-1)*dt;

    v(j)=exp(K1*t(j))*(A*cos(K2*t(j))+B*sin(K2*t(j)));

    i(j)=exp(K1*t(j))*(alpha*cos(K2*t(j))+beta*sin(K2*t(j)));

    fprintf(' %10.2f  %8.4f  %8.4f\n',t(j)*1e6,i(j),v(j));

end

% PART C: Plot iL, v vs. t

% Note: we scale the voltage plot by .01 to make the scales of i and v

% similar and thus show a more meaningful plot.

plot(t,i,t,v/100,'--'),xlabel('time(s)'), ylabel('i, v/100'), grid,

title('i(amps) and v/100(volts) vs time(sec)'),

legend('i','v/100');

-----

```

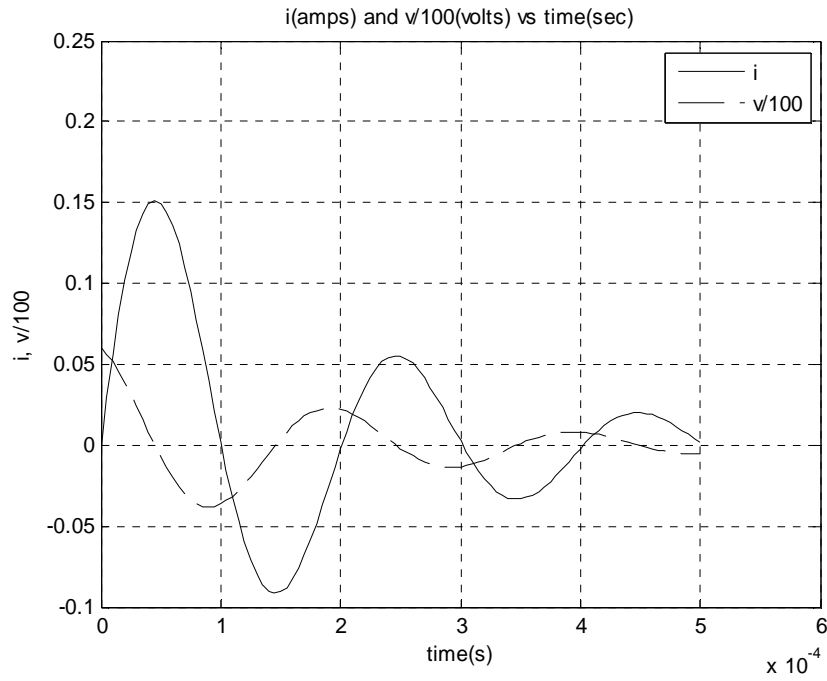
Program results:

```

Coefficients: alpha=0.0000, beta=0.1922
  t(microsec)  iL(amp)    v(volt)
-----

```

0.00	0.0000	6.0000
5.00	0.0291	5.6350
10.00	0.0562	5.1506
15.00	0.0805	4.5646
20.00	0.1017	3.8960
25.00	0.1193	3.1653
⋮	⋮	⋮
475.00	0.0137	-0.4259
480.00	0.0115	-0.4667
485.00	0.0091	-0.4941
490.00	0.0066	-0.5082
495.00	0.0040	-0.5092
500.00	0.0015	-0.4978



P2.9. The classical form for a second-order differential equation is

$$\frac{d^2 y}{dt^2} + 2\zeta\omega_n \frac{dy}{dt} + \omega_n^2 y = 0 \quad (\text{P2.9a})$$

where ζ is the *damping factor* and ω_n is the *natural frequency*. We can match the terms of Equation (P2.9a) with Equation (P2.8a) to find the damping factor and natural frequency for the parallel RLC circuit. Thus,

$$\omega_n = \frac{1}{\sqrt{LC}} \quad \text{and} \quad \text{Error! Objects cannot be created from editing field codes.}$$

(P2.9b)

Note that for $\zeta < 1$ the circuit is underdamped, for $\zeta > 1$ the circuit is overdamped, and for $\zeta = 1$, it is critically damped.

For the component values $L=1$ mH, $C=1\mu\text{F}$ and initial conditions $i_L(0) = 0.25$ amp and $v_C(0) = 6$ volt:

- Determine the resistor value R_{crit} which makes the circuit critically damped.
- Plot the inductor current i_L vs. time for the two values of R : $R = R_{crit}$ and $R = 5R_{crit}$ for $0 \leq t \leq 500 \mu\text{s}$ in steps of $1/2 \mu\text{s}$. Plot all of the waveforms on one graph

Solution:

- Setting $\zeta = 1$ in Equation (P2.9b) and solving for R gives R_{crit} , thus

$$R_{crit} = \frac{\sqrt{LC}}{2C} = \frac{\sqrt{10^{-3} \times 10^{-6}}}{2 \times 10^{-6}} = 15.8114 \text{ ohm}$$

For the critically-damped case, the solution is of the form given in Equation P2.7q:

$$i_L = (A + Bt)e^{-\zeta\omega_n t}$$

Setting $t=0$ and applying the initial condition gives $A = i_L(0)$. To apply the second initial condition, we know from Equation P2.7b that

$$v = L \frac{di_L}{dt} = Le^{-\zeta\omega_n t} (-A\zeta\omega_n + B(1 - \zeta\omega_n t))$$

At $t = 0$, $v(0) = L(-A\zeta\omega_n + B)$ and thus $B = \frac{v(0)}{L} + A\zeta\omega_n$.

For the underdamped case where $R = 5R_{crit}$, we use the same solution as in Project 2.7.

$$i_L = e^{-\zeta \omega_n t} \left\{ \alpha \cos \left(\sqrt{1 - \zeta^2} \omega_n t \right) + \beta \sin \left(\sqrt{1 - \zeta^2} \omega_n t \right) \right\}$$

where $\alpha = i_L(0)$, $\beta = \frac{\frac{v(0)}{L} + \zeta \omega_n \alpha}{\sqrt{1 - \zeta^2} \omega_n}$.

The program follows.

```
% P2_9.m

% This project determines iL(t) of an RLC circuit for 2 cases:
% R=Rcrit and R=5Rcrit.
% Component values: L=1mH and C=1uF
% Initial conditions: v0=6V and iL0=0.25A

clear; clc;

L=0.001; C=1.0e-6; iL0=0.25; v0=6;

Rcrit = sqrt(L*C)/(2*C);

omega = 1/sqrt(L*C);

% Calculate coefficients for critically damped case
zeta_crit = 1/(2*Rcrit*C*omega);

A = iL0;

B = v0/L - A*zeta_crit*omega;

% Calculate coefficients for underdamped case
zeta_underdamped = 1/(2*(5*Rcrit)*C*omega);

alpha = iL0;

beta = (v0/L + zeta_underdamped*omega*alpha)/ ...
    (sqrt(1-zeta_underdamped^2)*omega);

fprintf('Rcrit=%.4f ohm \n',Rcrit);

dt = 0.5e-6;

for n=1:1001

    t(n)=(n-1)*dt;

    iL_crit(n) = exp(-zeta_crit*omega*t(n))*(A+B*t(n));
```

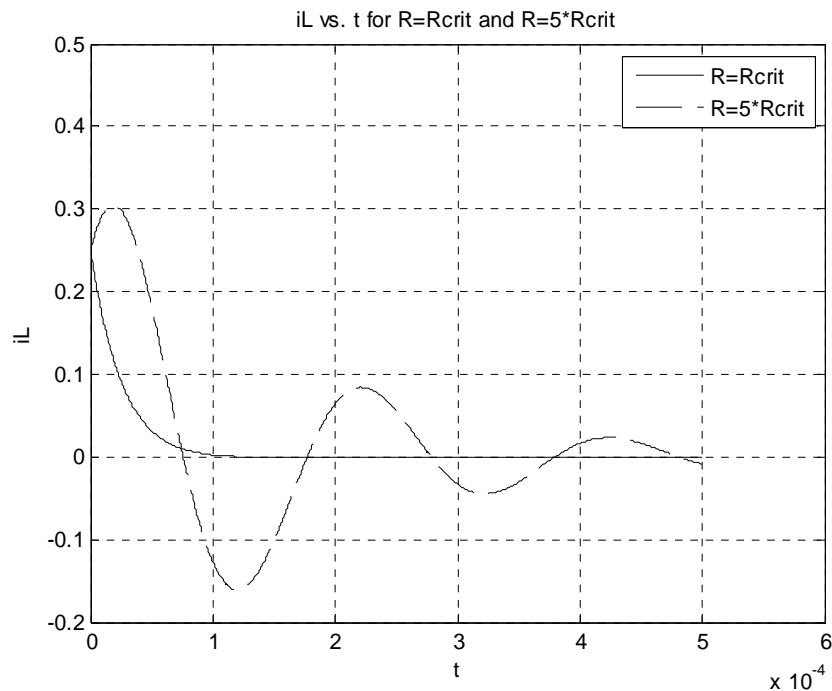
```

iL_underdamped(n) = exp(-zeta_underdamped*omega*t(n)) * ...
    ( alpha * cos(sqrt(1-zeta_underdamped^2)*omega*t(n)) + ...
    beta * sin(sqrt(1-zeta_underdamped^2)*omega*t(n)) );
end
plot(t,iL_crit,t,iL_underdamped,'--'), xlabel('t'), ylabel('iL'),
title('iL vs. t for R=Rcrit and R=5*Rcrit'), grid,
legend('R=Rcrit','R=5*Rcrit');

```

Program results:

Rcrit=15.8114 ohm



P2.10. Using Equations (P2.7c), (P2.7d), (P2.8b) and (P2.7n), we can show that for under-damped case in the parallel RLC circuit that

$$\begin{aligned}
i_R(t) = & -\exp\left(-\frac{1}{2RC}t\right)\left\{\alpha\cos\left(\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}t\right)+\beta\sin\left(\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}t\right)\right\} \\
& +\frac{1}{2R}\exp\left(-\frac{1}{2RC}t\right)\left\{A\cos\left(\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}t\right)+B\sin\left(\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}t\right)\right\} \\
& -C\exp\left(-\frac{1}{2RC}t\right)\left\{-A\sin\left(\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}t\right)+B\cos\left(\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}t\right)\right\} \\
& \times\sqrt{\frac{1}{LC}-\left(\frac{1}{2RC}\right)^2}
\end{aligned}
\tag{P2.10a}$$

Take the following component values to be: $R = 100\ \Omega$, $L=1\text{ mH}$, $C=1\mu\text{F}$ and initial conditions $i_L(0) = 0$ and $v(0) = 6\text{ V}$:

- If you have completed Project P2.7, use the values for A and B obtained in that program in Equation P2.10a, otherwise do part (a) of Project 2.7.
- If you have completed Project P2.8 use the values for α and β obtained in that program in Equation P2.10a, otherwise do part (a) of Project 2.8
- Construct a MATLAB program to calculate $i_R(t)$ for $0 \leq t \leq 500\ \mu\text{s}$ in steps of $5\ \mu\text{s}$.

Plot $i_R(t)$ vs. t for $0 \leq t \leq 500\ \mu\text{s}$ in steps of $5\ \mu\text{s}$.

Solution:

The program follows.

```

% P2_10.m:
% Solve for iR in the parallel RLC circuit by using iR = -iL - iC
clear; clc;

```

```

R=100; L=1e-3; C=1e-6; v0=6; iL0=0;

% Coefficients from Project 2.7:

K1 = -1/(2*R*C);

K2 = sqrt( 1/(L*C) - (1/(2*R*C))^2 );

A=v0;

B= -(iL0/C + v0/(R*C) + K1*A) / K2;

% Coefficients from Project 2.8:

alpha=iL0;

beta= (v0/L - K1*alpha)/K2;

fprintf('Coefficients from Proj. 2.7: A=%.4f, B=%.4f\n', A,B);

fprintf('Coefficients from Proj. 2.8: alpha=%.4f, beta=%.4f\n\n',
alpha,beta);

% Calculate iR(t) for t=[0,500 microsec] in steps of 5 microsec:

fprintf('    t(microsec)      iR(amps)    \n');

fprintf('-----\n');

dt=5.0e-6;

for i=1:101

    t(i)=(i-1)*dt;

    iR(i)=exp(K1*t(i))* ...

        ( -( alpha*cos(K2*t(i))+beta*sin(K2*t(i)) ) ...

          + 1/(2*R)*(A*cos(K2*t(i))+B*sin(K2*t(i))) ...

          - C*K2*(-A*sin(K2*t(i))+B*cos(K2*t(i))) );

    fprintf('%10.2f      %10.4f\n', t(i)*1e6,iR(i));

end

% Plot:

plot(t,iR), xlabel('time(s)'), ylabel('i_R'), grid,

title('i_R in (amps) vs time(s)');

-----

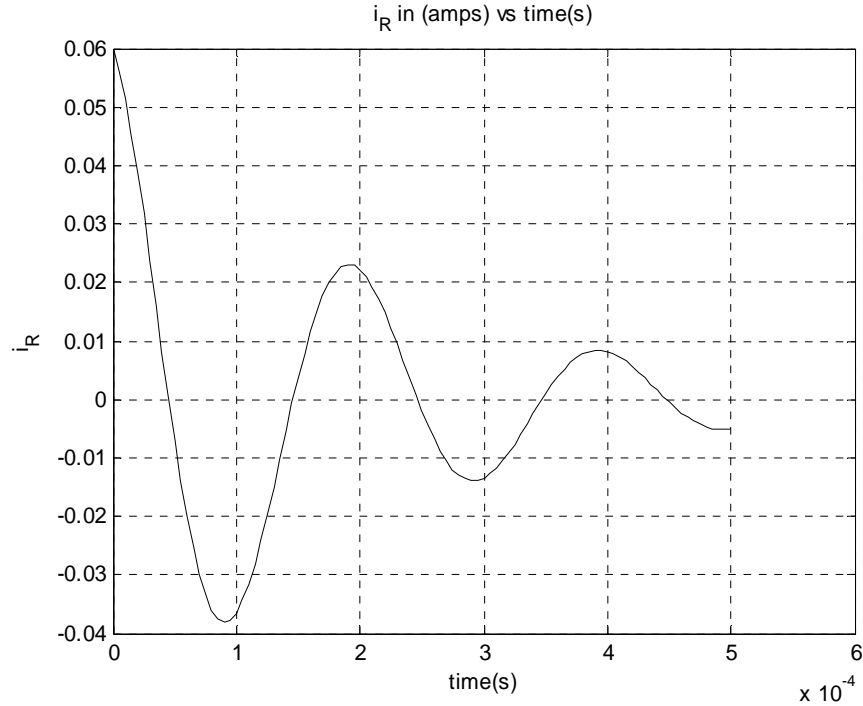
```

Program results:

Coefficients from Proj. 2.7: A=6.0000, B=-0.9608

Coefficients from Proj. 2.8: alpha=0.0000, beta=0.1922

t(microsec)	i _R (amps)
0.00	0.0600
5.00	0.0563
10.00	0.0515
15.00	0.0456
20.00	0.0390
25.00	0.0317
⋮	⋮
475.00	-0.0043
480.00	-0.0047
485.00	-0.0049
490.00	-0.0051
495.00	-0.0051
500.00	-0.0050



P2.11. This project involves a series RLC circuit as shown in Figure P2.11.

The resistor, inductor, and capacitor voltage-current relations using their respective constituent relations are:

$$v_R = i_R R \quad (\text{P2.11a})$$

$$v_L = L \frac{di_L}{dt} \quad (\text{P2.11b})$$

$$i_C = C \frac{dv_C}{dt} \quad (\text{P2.11c})$$

where v_R , v_L and v_C are the voltages across the resistor, inductor, and capacitor respectively, i_R , i_L and i_C are their respective currents, and R , L and C are their respective component values in ohms, farads, and henries.

Applying Kirchhoff's voltage law around the RLC loop gives

$$V_{in} - v_R - v_L - v_C = 0 \quad (\text{P2.11d})$$

In the series case, all of the components have the identical current and thus we define

$$i = i_R = i_L = i_C. \quad (\text{P2.11e})$$

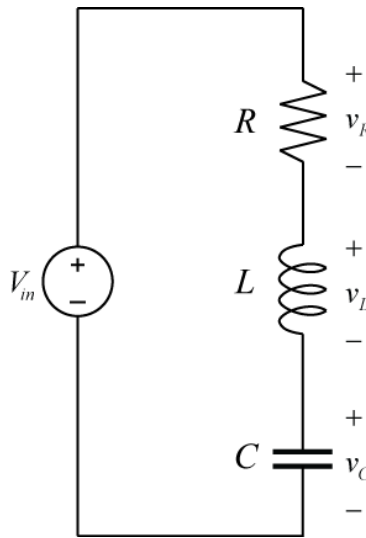


Figure P2.11. Series RLC circuit

a) Using Equations (P2.11a) – (P2.11e) derive the governing differential equation for the capacitor voltage v_C .

b) Using Project P2.7 as a guide, derive the three possible solutions to the governing differential equation for the voltage v_C . Assume that the voltage V_{in} becomes zero for $t > 0$.

c) The obtained solutions in part (b) should involve two arbitrary coefficients A and B .

Determine the values for A and B for the following two cases:

1. $R = 100 \Omega$, $L = 1 \text{ mH}$, $C = 1 \mu\text{F}$ and initial conditions $i(0) = 0$ and $V_{in}(0) = 6 \text{ V}$.

2. $R = 100 \Omega$, $L = 1 \text{ mH}$, $C = 0.1 \mu\text{F}$ and initial conditions $i(0) = 0$ and $V_{in}(0) = 6 \text{ V}$.

d) For the two cases of part (c), plot i vs. t , for $0 \leq t \leq 10^{-4} \text{ s}$ in steps of 10^{-6} s on the same page.

Solution:

a) Substituting Equations P2.11a and P2.11b into P2.11d and rearranging gives

$$iR + L \frac{di}{dt} + v_C = V_{in}$$

Now substituting Equation P2.11c into the above equation gives

$$RC \frac{dv_C}{dt} + LC \frac{d^2 v_C}{dt^2} + v_C = V_{in}$$

$$\Rightarrow \frac{d^2 v_C}{dt^2} + \frac{R}{L} \frac{dv_C}{dt} + \frac{1}{LC} v_C = \frac{V_{in}}{LC}$$

b) From Project P2.7:

To solve this homogeneous differential equation, we seek a function $v_C(t)$ such that the

derivatives $\frac{dv_C}{dt}(t)$ and $\frac{d^2 v_C}{dt^2}(t)$ reproduce the form of $v_C(t)$. A function that satisfies

this condition is $Ae^{\alpha t}$, where A and α are arbitrary constants. If we assume that

$v_C = Ae^{\alpha t}$, then

$$\frac{dv_c}{dt}(t) = \alpha A e^{\alpha t} \quad \text{and} \quad \frac{d^2 v_c}{dt^2}(t) = \alpha^2 A e^{\alpha t}$$

Substituting these terms into the differential equation (assuming $V_{in} = 0$) gives:

$$\left(\alpha^2 + \frac{R}{L} \alpha + \frac{1}{LC} \right) A e^{\alpha t} = 0$$

Since $e^{\alpha t} \neq 0$, then

$$\left(\alpha^2 + \frac{R}{L} \alpha + \frac{1}{LC} \right) = 0$$

The solutions are:

$$\alpha = -\frac{R}{2L} \pm \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}}$$

Thus, there are two solutions for α , and there are two solution that satisfy the differential equation. The general solution is the sum of all known solutions:

$$v_c = A \exp\left(-\frac{R}{2L}t + \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} t\right) + B \exp\left(-\frac{R}{2L}t - \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} t\right)$$

or

$$v_c = \exp\left(-\frac{R}{2L}t\right) \left\{ A \exp\left(\sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} t\right) + B \exp\left(-\sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} t\right) \right\}$$

where $\exp(x) = e^x$ and A and B are constants which depend on the initial conditions of the circuit.

Note that this solution has three regions of interest:

- I. *Over-damped*: if $\left(\frac{R}{2L}\right)^2 > \frac{1}{LC}$, then the solutions are decaying exponentials over time.

II. *Under-damped*: if $\left(\frac{R}{2L}\right)^2 < \frac{1}{LC}$, then the solutions are decaying sinusoids over time.

For the under-damped case, we can show the sinusoidal behavior by applying the

identities $e^{jx} = \cos x + j \sin x$ and $e^{-jx} = \cos x - j \sin x$ giving:

$$v_C = \exp\left(-\frac{R}{2L}t\right) \left\{ A \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t\right) + B \sin\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t\right) \right\}$$

where the coefficients A and B are to be determined by initial conditions.

III. If $\left(\frac{R}{2L}\right)^2 = \frac{1}{LC}$, then the square root term is zero and the system is said to be

critically damped. For this case, the solution is

$$v_C = (A + Bt) \exp\left(-\frac{R}{2L}t\right)$$

c) For $R = 100 \Omega$, $L = 1 \text{ mH}$, $C = 1 \mu\text{F}$, $\left(\frac{R}{2L}\right)^2 > \frac{1}{LC}$ and thus the system is over-damped.

For $R = 100 \Omega$, $L = 1 \text{ mH}$, $C = 0.1 \mu\text{F}$, $\left(\frac{R}{2L}\right)^2 < \frac{1}{LC}$ and thus the system is underdamped.

For the over-damped case,

$$v_C = e^{K_1 t} (A e^{K_2 t} + B e^{-K_2 t}) \quad (\text{S2.11a})$$

where we define the constants $K_1 = -\frac{R}{2L}$ and $K_2 = \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}}$. In order to solve for

A and B , we apply the initial conditions. Because $i(0) = 0$, we assume that the circuit is in

the steady state and thus $v_C(0) = V_{in}(0)$ (because in the steady state, the voltage drops

across the resistor and inductor will be zero). Thus, evaluating Equation S2.11a at $t=0$

gives

$$v_c(0) = A + B \quad (\text{S2.11b})$$

We now need to find an expression to apply the initial condition for i . From Equation

P2.11c, we know $i = C \frac{dv_c}{dt}$ and thus

$$i = C e^{K_1 t} \left(A(K_1 + K_2) e^{K_2 t} + B(K_1 - K_2) e^{-K_2 t} \right) \quad (\text{S2.11c})$$

At $t=0$,

$$i(0) = C \left(A(K_1 + K_2) + B(K_1 - K_2) \right) \quad (\text{S2.11d})$$

Solving Equations S2.11b and S2.11d for A and B gives

$$A = \frac{v_c(0)(K_1 - K_2) - \frac{i(0)}{C}}{-2K_2}$$

$$B = \frac{\frac{i(0)}{C} - v_c(0)(K_1 + K_2)}{-2K_2}$$

For the under-damped case, we assume a solution of the form

$$v_c = e^{K_1 t} (\alpha \cos K_3 t + \beta \sin K_3 t) \quad (\text{S2.11e})$$

where $K_3 = \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}$ and α and β are determined by the initial conditions.

Evaluating Equation S2.11e at $t=0$ gives $v_c(0) = \alpha$. To apply the initial condition for i ,

we know $i = C \frac{dv_c}{dt}$ and thus

$$i = C \frac{dv_c}{dt} = C e^{K_1 t} \left((K_1 \alpha + K_3 \beta) \cos K_3 t + (K_1 \beta - K_3 \alpha) \sin K_3 t \right)$$

Evaluating at $t=0$ gives $i(0) = C(K_1 \alpha + K_3 \beta)$ and thus $\beta = \left(\frac{i(0)}{C} - K_1 v_c(0) \right) / K_3$.

The program follows.

```
% P2_11.m
```

```
% Solve series RLC circuit for over-damped and
```



```

% underdamped cases.

clear; clc;

% Circuit parameters:
R=100; L=1e-3;

C_overdamped=1e-6; C_underdamped=0.1e-6;

% Initial conditions:
Vin0=6; i0=0.0;

% Solve for coefficients
K1 = -R/(2*L);

K2 = sqrt( (R/(2*L))^2 - 1/(L*C_overdamped) );

K3 = sqrt( 1/(L*C_underdamped) - (R/(2*L))^2 );

A = (Vin0*(K1-K2)-i0)/(-2*K2);

B = (i0-Vin0*(K1+K2))/(-2*K2);

alpha = Vin0;

beta = ( i0/C_underdamped - K1*Vin0 )/ K3;

fprintf('Coefficients: A=%.4f, B=%.4f\n', A,B);

fprintf('Coefficients: alpha=%.4f, beta=%.4f\n\n', alpha,beta);

% Calculate waveforms:

fprintf('  t(microsec)  i(amp)   vC(volt)   \n');

fprintf('-----\n');

dt=1e-6;

for j=1:101

    t(j)=(j-1)*dt;

    vC_overdamped(j) = exp(K1*t(j)) * (A*exp(K2*t(j)) + B*exp(-
K2*t(j)));

    i_overdamped(j)= C_overdamped*exp(K1*t(j)) * ...

        (A*(K1+K2)*exp(K2*t(j)) + B*(K1-K2)*exp(-K2*t(j)));

    vC_underdamped(j) = exp(K1*t(j)) * ...

        (alpha*cos(K3*t(j)) + beta*sin(K3*t(j)));

```

```

i_underdamped(j) = C_underdamped*exp(K1*t(j)) * ...
    ( (K1*alpha+K3*beta)*cos(K3*t(j)) + ...
    (K1*beta-K2*alpha)*sin(K3*t(j)));

fprintf(' %10.2f  %8.4f  %8.4f  %8.4f  %8.4f \n',t(j)*1e6,...
i_overdamped(j),vC_overdamped(j),i_underdamped(j),...
vC_underdamped(j));

end

% Plot:

subplot(2,1,1);

plot(t,i_overdamped),ylabel('i (A)'), grid,
title('Project 2.11, overdamped case'),
subplot(2,1,2);

plot(t,vC_overdamped),ylabel('v_C (V)'),grid,
xlabel('time (s)');

figure;

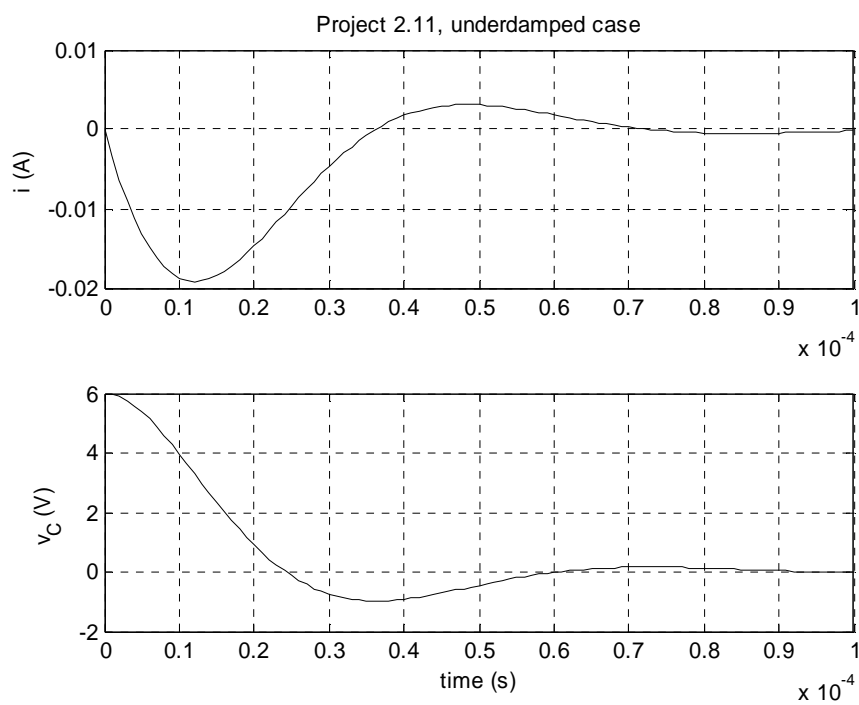
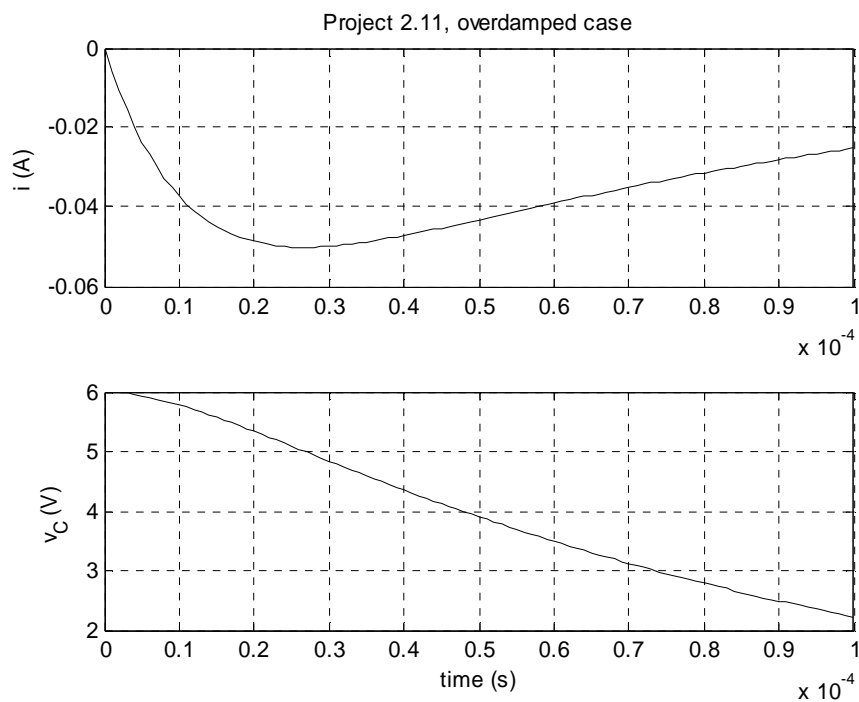
subplot(2,1,1);

plot(t,i_underdamped),ylabel('i (A)'), grid,
title('Project 2.11, underdamped case'),
subplot(2,1,2);

plot(t,vC_underdamped),ylabel('v_C (V)'),grid,
xlabel('time (s)');

```

Program results:



P2.12. This project is a variation of Project P2.11. Instead of solving for the voltage v_C in the series RLC circuit of Figure P2.11, we can also solve for the current i through the circuit elements.

- a) Use Equations P2.11a through P2.11e show that for the case when $V_{in} = 0$, the governing differential equation for v_C is:

$$\frac{d^2 i}{dt^2} + \frac{R}{L} \frac{di}{dt} + \frac{i}{LC} = 0 \quad (\text{P2.12a})$$

- b) Show that the general solution to Equation (P2.12a) is:

$$i = \exp\left(-\frac{R}{2L}t\right) \left\{ \alpha \cos\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t\right) + \beta \sin\left(\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t\right) \right\} \quad (\text{P2.12b})$$

- c) For the following component and initial conditions, determine α and β .

$R = 10 \Omega$, $L = 1 \text{ mH}$, $C = 2 \mu\text{F}$, and initial conditions $v_C(0) = 10$ and

$$\frac{dv_C}{dt}(0) = \frac{6}{\sqrt{LC}}.$$

- d) Plot i vs. t for $0 \leq t \leq 400 \mu\text{s}$.

Solution:

- a) We start by differentiating both sides of Equation P2.11d and setting $V_{in} = 0$:

$$\frac{dv_R}{dt} + \frac{dv_L}{dt} + \frac{dv_C}{dt} = 0 \quad (\text{S2.12a})$$

By differentiating both sides of Equation P2.11b, we know that $\frac{dv_L}{dt} = L \frac{d^2 i}{dt^2}$. By

differentiating both sides of Equation P2.11a, we know that $\frac{dv_R}{dt} = R \frac{di}{dt}$. Finally, from

Equation P2.11c, we know that $\frac{dv_C}{dt} = \frac{i}{C}$. Substituting these three results into Equation S2.12a gives

$$R \frac{di}{dt} + L \frac{d^2 i}{dt^2} + \frac{i}{C} = 0$$

Rearranging gives

$$\frac{d^2 i}{dt^2} + \frac{R}{L} \frac{di}{dt} + \frac{i}{LC} = 0$$

b) Assume an exponential solution with constants A and K :

$$\begin{aligned} i(t) &= Ae^{Kt} \\ \frac{di}{dt} &= KAe^{Kt} \\ \frac{d^2i}{dt^2} &= K^2 Ae^{Kt} \end{aligned}$$

Substituting into the differential equation gives

$$K^2 Ae^{Kt} + \frac{R}{L} KAe^{Kt} + \frac{1}{LC} Ae^{Kt} = 0$$

Dividing both sides by Ae^{Kt} gives

$$K^2 + \frac{R}{L} K + \frac{1}{LC} = 0$$

Solving for K gives

$$K = -\frac{R}{2L} \pm \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}}$$

In underdamped cases where $\frac{1}{LC} > \left(\frac{R}{2L}\right)^2$, then the square root term is complex and

$K = -\frac{R}{2L} \pm j\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}$. Thus, the solution is a sum of the two solutions for K :

$$i(t) = e^{-\frac{R}{2L}t} \left(Ae^{j\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}t} + Be^{-j\sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}t} \right) \quad (\text{S2.12b})$$

where A and B are constants. Applying the identity $e^{jx} = \cos x + j\sin x$ to Equation

S2.12b gives

$$\begin{aligned}
i(t) &= e^{-\frac{R}{2L}t} \left(A \cos \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t + jA \sin \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t + B \cos \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t - jB \sin \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t \right) \\
&= e^{-\frac{R}{2L}t} \left((A+B) \cos \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t + j(A-B) \sin \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t \right)
\end{aligned}$$

We now define the new constants α and β in terms of the previous constants A and B

such that $\alpha = A+B$ and $\beta = j(A-B)$ and thus

$$i(t) = \exp\left(-\frac{R}{2L}t\right) \left(\alpha \cos \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t + \beta \sin \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2} t \right) \quad (\text{S2.12c})$$

c) The initial conditions are $v_C(0) = 10$ and $\frac{dv_C}{dt}(0) = \frac{6}{\sqrt{LC}}$. We know that $i = C \frac{dv_C}{dt}$

and thus

$\frac{dv_C}{dt}(0) = \frac{6}{\sqrt{LC}} = \frac{i(0)}{C}$ and thus $i(0) = 6\sqrt{\frac{C}{L}}$. Evaluating Equation S2.12c for $t=0$ gives

$\alpha = i(0)$. To apply the second initial condition, rearrange Equation P2.11d to get

$v_C = -v_R - v_L$. We know from Equation P2.11a that $v_R = iR$ and from Equation P2.11b

that $v_L = L \frac{di}{dt}$, and thus

$$\begin{aligned}
v_C &= -iR - L \frac{di}{dt} \\
&= -iR - L \left(e^{K_1 t} ((\alpha K_1 + \beta K_2) \cos K_2 t + (\beta K_1 - \alpha K_2) \sin K_2 t) \right)
\end{aligned} \quad (\text{S2.12d})$$

where we define the constants $K_1 = -\frac{R}{2L}$ and $K_2 = \sqrt{\frac{1}{LC} - \left(\frac{R}{2L}\right)^2}$. Evaluating Equation

S2.12d at $t=0$ gives $v_C(0) = -i(0)R - L(\alpha K_1 + \beta K_2)$ and thus $\beta = -\frac{v_C(0) + i(0)R/2}{LK_2}$.

d) **The program follows:**

```
% P2_12.m
```

```
% Solve series RLC circuit using current as state
```

```

% variable

clear; clc;

% Circuit parameters:

R=10; L=1e-3; C=2e-6;

fprintf('Circuit parameters: R=%d, L=%.0e, C=%.0e\n',R,L,C);

% Initial conditions:

vC0=10;

i0=6*sqrt(C/L);

fprintf('Initial conditions: vC0=%.4f, i0=%.4f\n',vC0,i0);

K1 = -R/(2*L);

K2 = sqrt( 1/(L*C) - (R/(2*L))^2 );

alpha=i0;

beta= -(vC0+i0*R/2)/(L*K2);

fprintf('Coefficients: alpha=%.4f, beta=%.4f\n\n', alpha,beta);

fprintf('  t(microsec)    i(amp)    \n');

fprintf('-----\n');

% Calculate i for 0<=t<=400 usec

dt=1e-6;

for j=1:401

    t(j)=(j-1)*dt;

    i(j)=exp(K1*t(j))*(alpha*cos(K2*t(j))+beta*sin(K2*t(j)));

    fprintf(' %10.2f  %8.4f \n',t(j)*1e6,i(j));

end

% Plot

plot(t,i),xlabel('time(s)'), ylabel('i'), grid,

title('i(amps) vs time(sec)');

-----

```

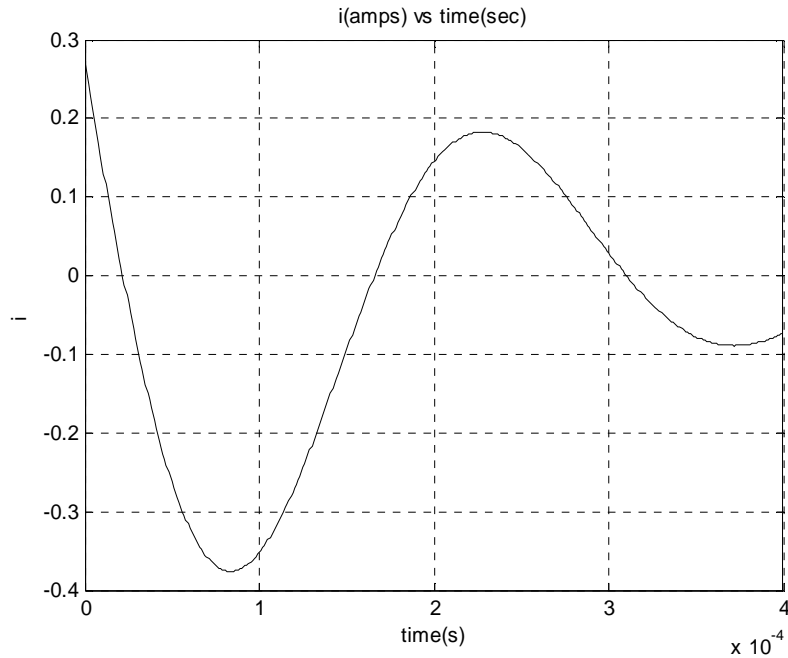
Program results:

Circuit parameters: $R=10$, $L=1e-03$, $C=2e-06$

Initial conditions: $vC0=10.0000$, $i0=0.2683$

Coefficients: $\alpha=0.2683$, $\beta=-0.5204$

t(microsec)	i(amp)
0.00	0.2683
1.00	0.2556
2.00	0.2430
3.00	0.2303
4.00	0.2176
5.00	0.2050
⋮	⋮
395.00	-0.0781
396.00	-0.0772
397.00	-0.0763
398.00	-0.0753
399.00	-0.0744
400.00	-0.0734



P2.13. A common problem in serial data communication is that digital bitstreams often contain long sequences of ones or zeros which can cause the receiver to become unsynchronized. One solution is to use Manchester coding in order to ensure that the bitstream contains many transitions so that it is easier to recover the clock.

10BaseT Ethernet uses Manchester coding as depicted in Figure P2.13a where a logical “one” is encoded as a low-to-high transition and a logical “zero” is encoded as a high-to-low transition. By ensuring that each bit contains a transition, the receiver can extract (or “recover”) a clock signal from the transmitted waveform by using a phase-locked loop circuit.

a) Using the `if-else` ladder in MATLAB, generate the waveform of Figure P2.13b for the bit sequence “1101” and plot. Assume the following parameters:

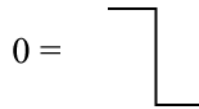
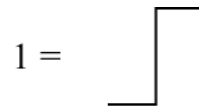
“High” voltage: +2.5V

“Low” voltage: -2.5V

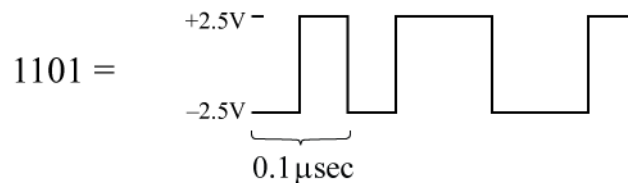
Bit rate: 10 Mbps (i.e. each bit is 0.1 μ s in duration)

When modeling your waveform in MATLAB, generate values from zero to 0.4 μ s in steps of 0.01 μ s. Plot using MATLAB’s `plot` command.

b) What is one disadvantage of Manchester coding?



(a)



(b)

Figure P2.13. (a) Manchester coding. (b) Bit sequence “1101.”

Solution:

The main disadvantage of Manchester code is that it requires twice the bandwidth as other encoding schemes.

The program follows:

```
% P2_13.m
% plot Manchester-coded bit sequence using if-elseif
step = 0.01e-6;
interval = 0.4e-6;
high=2.5;
low=-2.5;
t = 0:step:interval;
for i=1:length(t)
    if t(i) < 0.05e-6
        v(i)=low;
    elseif t(i)< 0.10e-6
        v(i)=high;
    elseif t(i)< 0.15e-6
        v(i)=low;
    elseif t(i)< 0.25e-6
        v(i)=high;
    elseif t(i)< 0.35e-6
        v(i)=low;
    else
        v(i)=high;
    end
end
plot(t,v);
```

```
axis([0,interval,-3,3]);  
  
xlabel('t (sec)'),ylabel('output (volts)'),  
  
title('Manchester code output for ''1101'');
```

Program results:

