```
%this program is intended to introduce the student to Matlab
%Initially it is a good idea to clear all variables starting a program

clear

%Matlab can be used as a calculator or as a programming language
%in the matlab window you can type in commands just like a sophisticated
%calculator.  But it is usually better to write a program that can execute a
%series of commands.

%Let's save what we've typed so far as an "m-file" so we can execute it in the
%matlab command window.  This is a critical step.
%You have to make sure that Matlab knows where to search for your m-file.
%While in the command window type "pwd<e>"  note: <e> = Enter key
%You can save your m-file in the directory that is displayed, for example: C:\MATLAB\bin
%Or, you can you use the path browser in the command window to add your particular
%directory to the Matlab path.

%Save the program you created using a .m extension, for example: "EML4220hw1.m"
%Note Matlab is case sensitive
%Now if you want to run your program go into the Matlab command window
%and type "EML4220hw1<e>"
%the program you created will now execute all the commands in file EML4220hw1.m

%Now lets create some code that will plot the response of a SDOF SMD system given an
%inital
%displacement and velocity

%Input the given constant or quantities
m=40;                   %mass
k=1000;                 %stiffness
zeta=0.1;               %damping ratio
xo=.5;                  %initial displacement
vo=0;                   %initial velocity

%note the semicolon at the end prevents those variables from being printed each time you
%run
%your m-file.  If you omit the semicolon it will print those variables in the command
%window.

wn=sqrt(k/m);           %calculate the natural freq.
cc=2*m*wn;              %calculate the critical damping
c=2*m*wn*zeta;          %calculate the vicsous damping constant
wd=wn*sqrt(1-zeta^2);   %calculate the damped natural frequency

%The solution has the form e^(-zeta*wn*t) * {X1*cos(wd*t)+X2*sin(wd*t)}

%Calculate X1 and X2
X1=xo;
X2=(vo+zeta*wn*xo)/wd;

%Define the time vector for this response
t=0:.01:3;  %Define the time from 0 to 3 seconds in steps of 0.01 seconds
%t is now a vector and will look like [0 .01 .02 .03 .... 2.98 2.99 3.0]

%I'm going to solve this problem 2 different ways.  The first way using
%a loop, making calculations at each frequency.  The second
```

```
%way is to calculate all the values at once using advanced Matlab features.

%Method 1: create a loop that calculates the response for each value of t
%Note the loop will repeat starting at integer 1, stepped by 1.  The number
%of times the loop repeats  is equal to the length of the vector t

for j=1:1:length(t),
   x(j) = exp(-zeta*wn*t(j))*( X1*cos(wd*t(j)) + X2*sin(wd*t(j)) );
end %loop j

%now I have two vectors x and t that both have the same # of elements
%lets plot the results

%now let's plot the results
figure(1)
plot(t,x),grid
title('SMD Response Method 1'),grid
ylabel('Displacement [m]')
xlabel('Time [s]')

%Method 2: Matlab knows that t is a vector and so we don't need to use a loop if we
%don't want to.  Let's calculate the reponse a different way and call it x2
%for multiplication we have to use ".*" instead of "*"
x2 = exp(-zeta.*wn.*t).*( X1.*cos(wd.*t) + X2.*sin(wd.*t) );

%now let's plot the results for both methods

figure(2)
subplot(2,1,1)
plot(t,x),grid
title('SMD Response Method 1'),grid
ylabel('Displacement [m]')

subplot(2,1,2)
plot(t,x2),grid
title('SMD Response Method 2'),grid
ylabel('Displacement [m]')
xlabel('Time [s]')

%Now let's do a new example:  Let's plot  y1=t, y2=t^2 and y3=t*sin(100t) on the
%same graph
%Let's redefine t from 0 to 0.5 seconds in steps of 0.001
t=0:.001:.5;
y1=t;
y2=t.^2;
y3=t.*sin(100.*t);

%Now plot the results
figure(3)
plot(t,y1,'r',t,y2,'b',t,y3,'g')
title('Three functions y1, y2, & y3')
xlabel('time [sec.]')
ylabel('y1, y2, and y3')
legend('y1=t','y2=t^2','y3=t*sin(100*t)',0)

%Two commands that are very important are "who" and "whos"
%type those in at the command window to see your variables displayed
```

```
%To clear particular variables type "clear " followed by the variable you want
%to be cleared.  For example:
%clear y1 y2 y3

%Matlab is also excellent when dealing with complex #'s
%Let's say we had a transfer function given by
%XF(s)=1/(m*s^2 + c*s +k) where s=iw (Laplace variable)
%Let's plot the transfer function

%Let's create a frequency range over which we will calculate the XF from 1 r/s to
%1000 r/s

w=1:.5:1000;
%now let's calculate the XF fro each frequency using a loop
for j=1:1:length(w),
   s=i*w(j);
   num=1;
   den=m*s^2 + c*s + k;
   XF(j)=num/den;
end

%Note that XF is a complex vector!  Plotting a complex function vs. w doesn't
%make sense.  We need to plot the real, imaginary, magnitude, or phase of XF

%Now let's plot the magnitude and phase of XF
MAGXF=abs(XF);
PhaseXFradians=angle(XF);
PhaseXFdeg=(180/pi)*PhaseXFradians; %convert to degrees

%This time it makes sense to plot in log scale
figure(4)
subplot(2,1,1)
loglog(w,MAGXF),grid
ylabel('Magnitude of XF')
xlabel('Frequency [r/s]')

subplot(2,1,2)
semilogx(w,PhaseXFdeg),grid
ylabel('Phase of XF'),grid
xlabel('Frequency [r/s]')

%More examples: Now let's enter a matrix and perform some manipulation on it
A=[1 3 2 9;33 0 2 5.6;12 1 1 1;9 14 17 99]   %this is 4x4 matrix
inverseA=inv(A)      %let's get the inverse
detA=det(A)    %let's get the determinant
transA=A'      %let's get the transpose
eigenA=eig(A)     %let's get the eigen values

%for the polynomial 3x^4 + 5x^3 + 1.7x^2 + 7.9 = 0
%let's find the roots
ourpoly=[3 5 1.7 0 7.9];
rootsourpoly=roots(ourpoly)
```