Solving Problems in Dynamics and Vibrations Using MATLAB

Parasuram Harihara

And

Dara W. Childs

Dept of Mechanical Engineering
Texas A & M University
College Station.

Contents

An Introduction to MATLAB

Purpose of the Handout

This handout was developed to help you understand the basic features of MATLAB and also to help you understand the use of some basic functions. This is not a comprehensive tutorial for MATLAB. To learn more about a certain function, you should use the online help. For example, if you want to know more about the function ‘solve’, then type the following command in the command window at the prompt:

help solve

Introduction

MATLAB is a high performance language for technical computing. The name MATLAB stands for *matrix laboratory*. Some of the typical uses of MATLAB are given below:

- Math and Computation
- Algorithm Development
- Modeling, Simulation and Prototyping

M-Files

Files that contain code in MATLAB language are called M-Files. You create a M-File using a text editor and then use them as you would any other MATLAB function or command. But before using the user defined functions always make sure that the ‘path’ is set to the current directory. There are two kinds of M-Files:

- Scripts, which do not accept input arguments or return output arguments.
- Functions, which can accept input and output arguments.

When you invoke a *script* MATLAB simply executes the commands found in the file. For example, create a file called ‘SumOfNumbers’ that contains these commands:

```
% To find the sum of first 10 natural numbers

n = 0;
i = 1;
for i=1:10;
     n = n + i;
     i = i + 1;
end
n
```

Go to the command window and set the path to the current directory. Then typing

SumOfNumbers

at the command prompt causes MATLAB to execute the commands in the M-File and print out the value of the sum of the first 10 natural numbers.

Functions are dealt in detail later in the handout.

Matrices

Suppose you have to enter a 2x2 identity matrix in MATLAB. Type the following command in the command window:

```
A=[1 0; 0 1]
```

MATLAB displays the matrix you just entered

```
A=
         1        0
         0        1
```

The basic conventions you need to follow are as follows:

- Separate the elements in the row with blanks or commas
- Use a semicolon, (;) to indicate the end of each row
- Surround the entire list of elements with square brackets, [].

There are many other ways of representing matrices, but the above-mentioned method is one of the popularly used methods. To know more about the different types of representation, go through the online help or the MATLAB user guide.

Suppose, you want to separate out the first column in the above matrix and use it for some other calculation, then type the following in the MATLAB command window.

```
Y=A(:,1)
```

Here Y contains the elements of the first column. Y is a 2x1 matrix. Note that the colon in the above expression indicates that MATLAB will consider all rows and '1' indicates the first column. Similarly if you want to separate the second row then type the following command

```
T=A(2,:)
```

Solving Linear Equations

Suppose for example, you have to solve the following linear equations for 'x' and 'y'.

$$x + 2y = 6$$
$$x - y = 0$$

There are two methods to solve the above-mentioned linear simultaneous equations.

The first method is to use matrix algebra and the second one is to use the MATLAB command '*solve*'.

Matrix Algebra

Representing the above two equations in the matrix form, we get

$$\begin{bmatrix} 1 & 2 \\ 1 & -1 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} 6 \\ 0 \end{Bmatrix}$$

The above equation is in the form of

$$AX = B$$

where A is known as the coefficient matrix, X is called the variable matrix and B, the constant matrix.

To solve for X, we find the inverse of the matrix A (provided the inverse exits) and then pre-multiply the inverse to the matrix B i.e.,

$$X = A^{-1}B$$

The MATLAB code for the above-mentioned operations is as shown below. Open a new M-File and type in the following commands in the file.

```
% To solve two simultaneous linear equations.

A = [1 2;1 -1];
B = [6;0];
X = inv (A)*B
```

Before executing the program, always remember to set the path to the current directory. Note that the first statement is preceded by a '%' symbol. This symbol indicates that the statement following the symbol is to be regarded as a comment statement. Also note that if you put a semicolon ';' after each statement, then the value of that statement is not printed in the command window. Since we require MATLAB to print the value of 'X', the semicolon does not follow the last statement.

Solve Command

The ‘*solve*’ command is a predefined function in MATLAB. The code for solving the above equations using the ‘*solve*’ command is as shown. Open a new M-File and type the following code.

```
% To solve the linear equations using the solve command

p = 'x + 2*y = 6';
q = 'x - y = 0';
[x,y] = solve(p,q)
```

Subs Command

This command is explained by means of the following example. Suppose you want to solve the following linear equations:

$$x + 2y = a + 6$$
$$x - y = a$$

Note that there are three unknown and only two equations. So we are required to solve for ‘x’ and ‘y’ in terms of ‘a’. To calculate the numerical values of ‘x’ and ‘y’ for different values of ‘a’, we use the subs command.

The following MATLAB code is used to demonstrate the ‘subs’ command.

```
% To solve the linear equations using the subs command

p = 'x + 2*y = a + 6'
q = 'x - y = a'
[x,y] = solve(p,q)

a = 0;
[x] = subs(x)
[y] = subs(y)
```

Here the ‘*solve*’ command solves for the values of ‘x’ and ‘y’ in terms of ‘a’. Then ‘a’ is assigned the value of ‘0’. Then we substitute the assigned value of ‘a’ in ‘x’ and ‘y’ to get the numerical value.

Suppose a varies from 1 to 10 and we need to find the value of ‘x’ and ‘y’ for each value of ‘a’ then use the following code:

```
% To solve the linear equations using the subs command
```

```
p = 'x + 2*y = a + 6'
q = 'x - y = a'
[x,y] = solve(p,q)

i = 0;
for a = 1:10
      t(i) = subs(x);
      p(i) = subs(y);
      i = i + 1     ;
end
```

In the above code t(i) and p(i) contains the values of 'x' and 'y' for different values of 'a' respectively.

Functions

Functions are M-Files that can accept input arguments and return output arguments. The name of the M-File and the function should be the same. Functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command prompt. The following example explains the use of functions in more detail.

Define a new function called hyperbola in a new M-File.

```
Function y = twice(x)
y = 2*x;
```

Now in a new M-file plot 'y' with respect to 'x' for different values of 'x'. The code is as shown below

```
x = 1:10
y = twice(x)
plot(x,y)
```

Basically the function 'twice' takes the values of x, multiplies it by 2 and then stores each value in y. The 'plot' function plots the values of 'y' with respect to 'x'.

To learn more about the use of functions, go through the user guide.

Hold Command

HOLD ON holds the current plot and all axis properties so that subsequent graphing commands add to the existing graph. HOLD OFF returns to the default mode whereby PLOT commands erase the previous plots and reset all axis properties before drawing new plots.

One can also just use the '*Hold*' command, which toggles the hold state. To use this command, one should use a counter in connection with an *'if'* loop; which first toggles the hold on state and then the counter must be incremented such that, the flow of command does not enter the '*if*' loop.

```
% Example to show the use of the hold command

counter = 0;

if (counter ==0)
   hold;
end

counter = counter + 1;
```

Here the '*hold*' command is executed only if the value of the counter is equal to zero. In this way we can make sure that the plots are held. The drawback of this procedure is that we cannot toggle the '*hold*' to off.

Plotting Techniques

The plot function has different forms depending on the input arguments. For example, if y is a vector, plot (y) produces a linear graph of the elements of y versus the index of the elements of y. If you specify two vectors as arguments, plot (x,y) produces a graph of y versus x.

Whenever a plot is drawn, title's and a label's for the x axis and y axis are required. This is achieved by using the following command.

To include a title for a plot use the following command

title ('Title')

For the x and y labels, use

xlabel(' label for the x-axis')
ylabel('label for the y-axis')

Always remember to plot the data with the grid lines. You can include the grid lines to your current axes by including the following command in your code.

grid on

For more information about various other plotting techniques, type

help plot

in the command window.

Suppose one wants to change the color of the curve in the plot to magenta, then use the following command

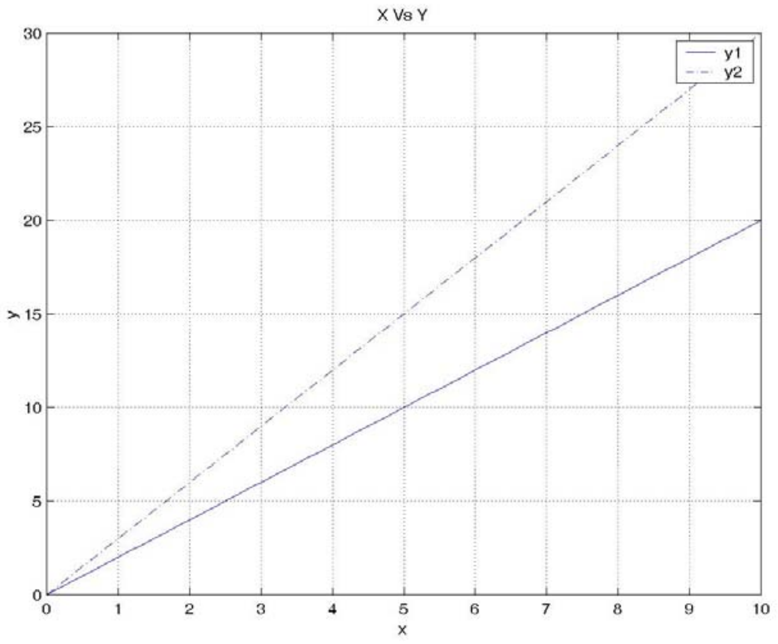
plot (x,y,'m')

'm' is the color code for magenta. There are various other color codes like 'r' for red, 'y' for yellow, 'b' for blue etc.

The following example makes use of the above-mentioned commands to plot two different data in the same plot. The MATLAB code is given below.

```
x=0:0.1:10;
y1 = x.*2;
y2 = x.*3;
plot(x,y1)
hold on
plot(x,y2,'-.')
grid on
xlabel('x')
ylabel('y')
title('X Vs Y')
legend('y1','y2')
```

When you run the above code, you will get the following plot.



In order to save the plot as a JPEG file, click the file icon in the figure window and then click the export command. In the 'save as type' option select '*.jpg'. In order to import the file in MS word, go to 'Insert' icon and then select 'Picture from file' command.

Figure Command

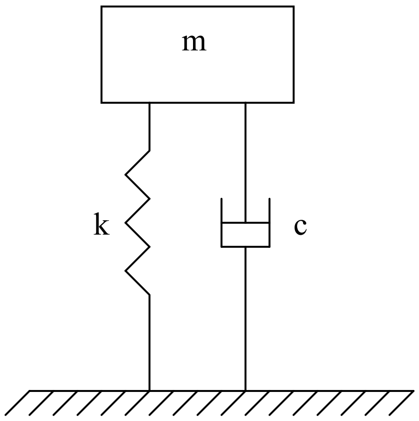
Suppose you want to plot two different vectors in two different windows, use the “figure” command. The example given below illustrates the use of the “figure” command.

```
% to illustrate the figure command

x=[1,2,3,4,5,6,7,8,9,10];
y=[12 23 4 5 65 67 89];

figure(1)
plot(x)
grid
title('plot of X')

figure(2)
plot(y)
grid
title('plot of Y')
```

The above code will plot the values of x and y in two different figure windows. If you want to know more about the command type the following at the command prompt.

help figure

1. Spring Mass Damper System – Unforced Response



Example

Solve for five cycles, the response of an unforced system given by the equation

$$m\ddot{x} + c\dot{x} + kx = 0 \tag{1}$$

For ξ = 0.1; m = 1 kg; k = 100 N/m; x (0) = 0.02 m; $\dot{x}(0) = 0$;

Solution

The above equation is a second order constant-coefficient differential equation. To solve this equation we have to reduce it into two first order differential equations. This step is taken because MATLAB uses a Runge-Kutta method to solve differential equations, which is valid only for first order equations.

Let

$$\dot{x} = v \tag{2}$$

From the above expression we see that

$$m\dot{v} + cv + kx = 0$$

so the equation (1) reduces to

$$\dot{v} = [(\frac{-c}{m})v - (\frac{k}{m})x] \tag{3}$$

We can see that the second order differential equation (1) has been reduced to two first order differential equations (2) and (3).

For our convenience, put

$x = y(1);$

$\dot{x} = v = y(2);$

Equations (2) and (3) reduce to

$\dot{y}(1) = y(2);$ (4)

$\dot{y}(2) = [(-c/m)*y(2) - (k/m)*y(1)];$ (5)

To calculate the value of 'c', compare equation (1) with the following generalized equation.

$$\ddot{x} + 2\xi\omega_n \dot{x} + \omega_n^2 = 0$$

Equating the coefficients of the similar terms we have

$$\frac{c}{m} = 2\xi\omega_n \qquad (6)$$

$$\omega_n^2 = \frac{k}{m} \qquad (7)$$

Using the values of 'm' and 'k', calculate the different values of 'c' corresponding to each value of ξ. Once the values of 'c' are known, equations (4) and (5) can be solved using MATLAB.

The problem should be solved for five cycles. In order to find the time interval, we first need to determine the damped period of the system.

Natural frequency $\omega_n = \sqrt{(k/m)} = 10$ rad/sec.

For $\xi = 0.1$

Damped natural frequency $\omega_d = \omega_n\sqrt{1-\xi^2} = 9.95$ rad/sec.

Damped time period $T_d = 2\pi/\omega_d = 0.63$ sec.

Therefore for five time cycles the interval should be 5 times the damped time period, i.e., 3.16 sec.

MATLAB Code

In order to apply the ODE45 or any other numerical integration procedure, a separate function file must be generated to define equations (4) and (5). Actually, the right hand side of equations (4) and (5) are what is stored in the file. The equations are written in the form of a vector.

The MATLAB code is given below.

```
function yp = unforced1(t,y)
yp = [y(2);(-((c/m)*y(2))-((k/m)*y(1)))];                    (8)
```

Open a new M-file and write down the above two lines. The first line of the function file must start with the word "function" and the file must be saved corresponding to the function call; i.e., in this case, the file is saved as unforced1.m. The derivatives are stored in the form of a vector.

This example problem has been solved for $\xi = 0.1$. We need to find the value of 'c/m' and 'k/m' so that the values can be substituted in equation (8). Substituting the values of ξ and ω_n in equations (6) and (7) the values of 'c/m' and 'k/m' can be found out. After finding the values, substitute them into equation (8).

Now we need to write a code, which calls the above function and solves the differential equation and plots the required result. First open another M-file and type the following code.

```
tspan=[0 4];
y0=[0.02;0];
[t,y]=ode45('unforced1',tspan,y0);
plot(t,y(:,1));
grid on
xlabel('time')
ylabel('Displacement')
title('Displacement Vs Time')
hold on;
```

The ode45 command in the main body calls the function unforced1, which defines the systems first order derivatives. The response is then plotted using the plot command. 'tspan' represents the time interval and 'y0' represents the initial conditions for y(1) and y(2) which in turn represent the displacement 'x' and the first derivative of 'x'. In this example, the initial conditions are taken as 0.02 m for 'x' and 0 m/sec for the first derivative of 'x'.

MATLAB uses a default step value for the vector 'tspan'. In order to change the step size use the following code

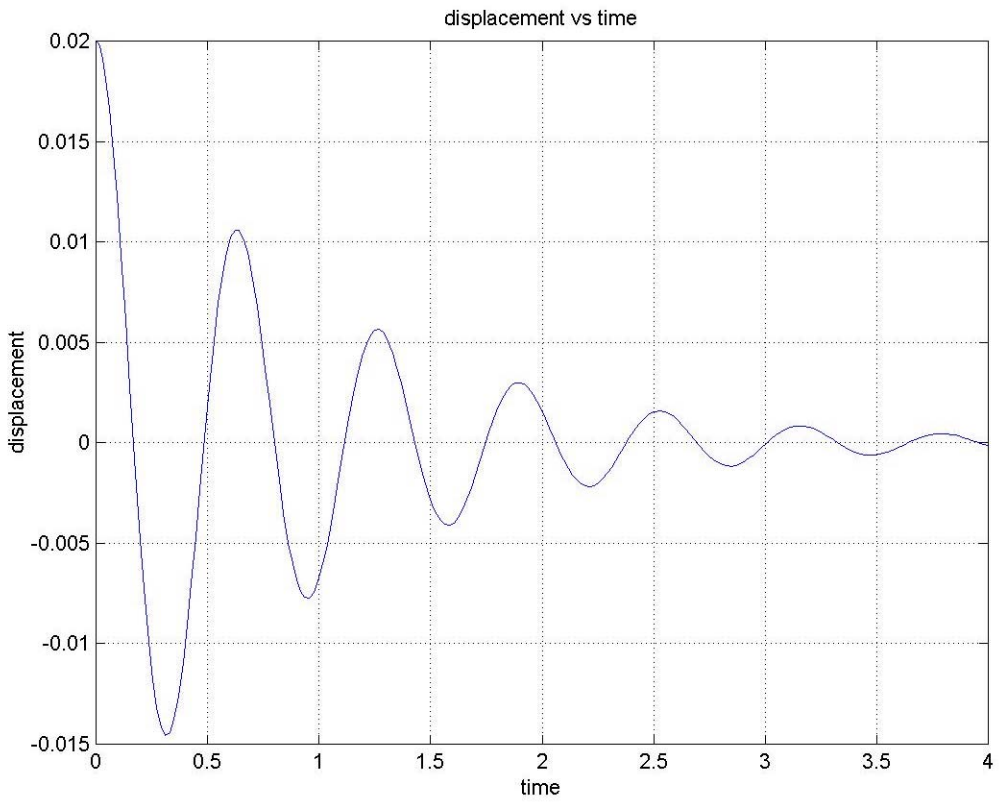
```
tspan=0:0.001:4;
```

This tells MATLAB that the vector 'tspan' is from 0 to 4 with a step size of 0.001. This example uses some arbitrary step size. If the step size is too small, the plot obtained will not be a smooth curve. So it is always better to use relatively larger step size. But it also takes longer to simulate programs with larger step size. So you have to decide which is the best step size you can use for a given problem.

In order to solve for different values of ξ, calculate the values of 'c/m' for each value of ξ. Substitute each value of ξ in the function file, which has the derivatives, save the file and then run the main program to view the result. This process might be too tedious. The code can be made more efficient by incorporating a 'For loop' for the various values of zeta.

In the above code 'y(:,1) represents the displacement 'x'. To plot the velocity, change the variable in the plot command line to 'y(:,2)'.

The plot is attached below



Assignment

Solve for six cycles the response of an unforced system given by

$$m\ddot{x} + c\dot{x} + kx = 0$$

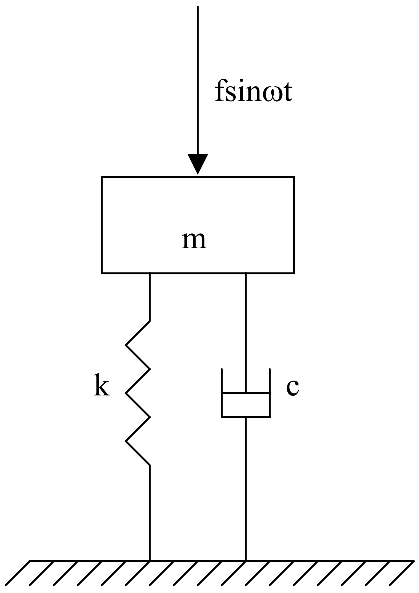
For ξ = {0, 0.1 0.25, 0.5, 0.75, 1.0}.

Take m = 5 kg; k = 1000 N/m; x(0) = 5 cms; $\dot{x}(0) = 0$;

Develop a plot for the solutions corresponding to the seven ξ values and comment on the plots obtained.

Note: When you save the M-File, make sure that the name of the file does not coincide with any of the variables used in the MATLAB code.

Spring Mass Damper System – Forced Response



Example

Plot the response of a forced system given by the equation

$$m\ddot{x} + c\dot{x} + kx = f\sin\omega t \qquad (1)$$

For ξ = 0.1; m = 1 kg; k = 100 N/m; f = 100 N; ω = 2ω_n; x(0) = 2 cms; $\dot{x}(0) = 0$.

Solution

The above equation is similar to the unforced system except that it has a forcing function. To solve this equation we have to reduce it into two first order differential equations. Again, this step is taken because MATLAB uses a Runge-Kutta method to solve differential equations, which is valid only for first order equations.

Let

$$\dot{x} = v \qquad (2)$$

so the above equation reduces to

$$\dot{v} = [(\frac{f}{m})\sin\omega\ t - (\frac{c}{m})v - (\frac{k}{m})x] \qquad (3)$$

We can see that the second order differential equation has been reduced to two first order differential equations.

For our convenience, put

x = y (1);

$\dot{x} = v = y(2);$

Equations (2) and (3) then reduce to

$\dot{y}(1) = y(2);$

$\dot{y}(2) = [(f/m)*\sin(\omega * t) + (-c/m)*y(2) - (k/m)*y(1)];$

Again, to calculate the value of 'c', compare equation (1) with the following generalized equation

$$\ddot{x} + 2\xi\omega_n \dot{x} + \omega_n{}^2 = f \sin \omega \ t$$

Equating the coefficients of the similar terms we have

$$\frac{c}{m} = 2\xi\omega_n$$

$$\omega_n{}^2 = \frac{k}{m}$$

Using the values of 'm' and 'k', calculate the different values of 'c' corresponding to each value of ξ.

To find the time interval the simulation should run, we first need to find the damped time period.

Natural frequency $\omega_n = \sqrt{(k/m)} = 10$ rad/sec.

For $\xi = 0.1$;

Damped natural frequency $\omega_d = \omega_n \sqrt{1-\xi^2} = 9.95$ rad/sec.

Damped time period $T_d = 2\pi/\omega_d = 0.63$ sec.

Therefore, for five time cycles the interval should be 5 times the damped time period, i.e., 3.16 sec. Since the plots should indicate both the transient and the steady state response, the time interval will be increased.

MATLAB Code

The MATLAB code is similar to that written for the unforced response system, except that there is an extra term in the derivative vector, which represents the force applied to the system.

The MATLAB code is given below.

```
function yp = forced(t,y)
yp = [y(2);(((f/m)*sin(ωn*t))-((c/m)*y(2))-((k/m)*y(1)))];
```

Again the problem is to be solved for ξ = 0.1. So, calculate the value of 'c/m', 'k/m' and 'f/m' by following the procedure mentioned in the earlier example and then substitute these values into the above expression. Save the file as 'forced.m'.

The following code represents the main code, which calls the function and solves the differential equations and plots the required result.

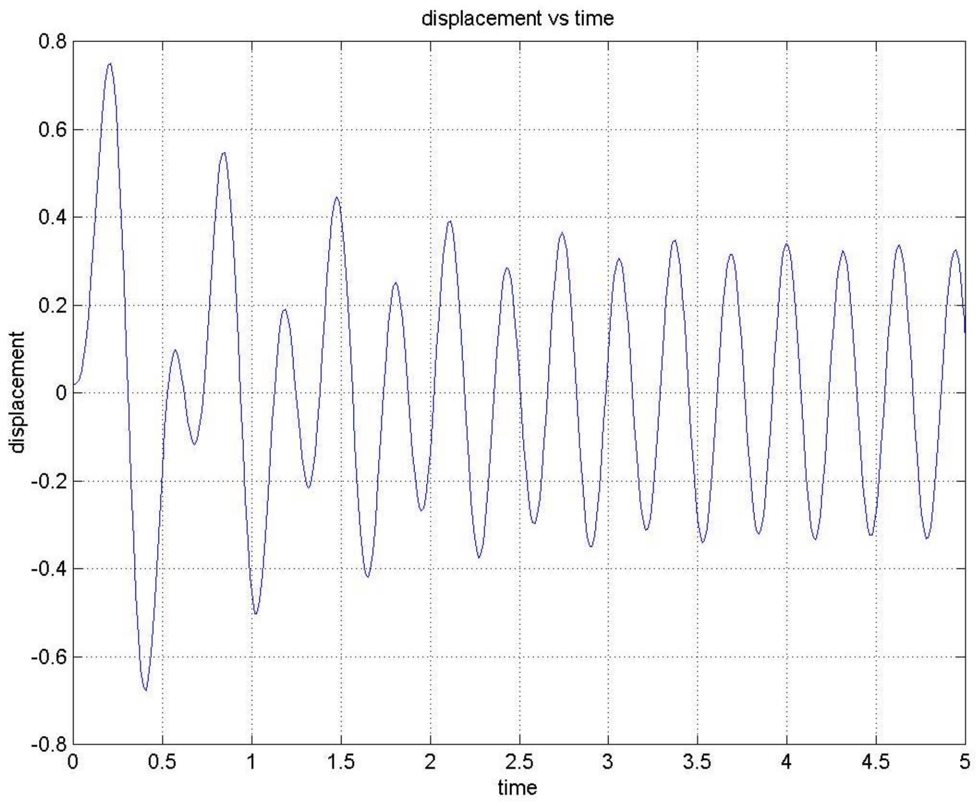
```
tspan=[0 5];
y0=[0.02;0];
[t,y]=ode45('forced',tspan,y0);
plot(t,y(:,1));
grid on
xlabel('time')
ylabel('Displacement')
title('Displacement Vs Time')
hold on;
```

Again, 'tspan' represents the time interval and 'y0' represents the initial conditions for y(1) and y(2) which in turn represent the displacement 'x' and the first derivative of 'x'. In this example the initial conditions are taken as 0.02 m for 'x' and 0 cm/sec for the first derivative of 'x'. Again the default step size of the vector 'tspan' can be changed accordingly as explained in the previous example.

To solve for different values of ξ, calculate the values of 'c/m' for each value of ξ. Substitute each value of ξ in the function file, which has the derivatives, save the file and then run the main program to view the result.

In the above code 'y(:,1) represents the displacement 'x'. To plot the velocity, change the variable in the plot command line to 'y(:,2)'.

The plot is attached below.


displacement vs time
displacement
time
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Assignment

Plot the response of a forced system given by the equation

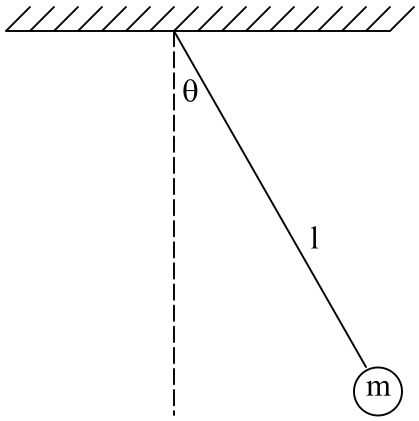
$$m\ddot{x} + c\dot{x} + kx = f \sin \omega t$$

For $\xi = \{0, 0.1\ 0.25, 0.5, 0.75, 1.0\}$.

Take m = 5 kg; k = 1000 N/m; f = 50 N; $\omega = 4\omega_n$, x(0) = 5 cms; $\dot{x}(0) = 0$.

Develop a plot for the solutions corresponding to the seven ξ values and comment on the plots obtained.

2. Simple Pendulum



Example
Compute and plot the linear response of a simple pendulum having a mass of 10 grams and a length of 5 cms. The initial conditions are $\theta(0) = 90°$ and $\dot{\theta}(0) = 0$
Also compare the generated plot with the nonlinear plot.

Solution

The differential equation of motion for the simple pendulum without any damping is given by

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

If we consider the system to be linear, i.e., for small angles,

$$sin\theta = \theta$$

So the linearized version of the above non-linear differential equation reduces to

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

The above equation is a second order, constant-coefficient differential equation. In order to use MATLAB to solve it, we have to reduce it to two first order differential equations as MATLAB uses a Runge-kutta method to solve differential equations, which is applicable only for first order differential equations.

Let

$$\theta = y(1)$$

$$\dot{\theta} = y(2)$$

When the values of 'theta' and the first derivative of 'theta' are substituted in the second order linearized differential equation, we get the following two first order differential equation:

$$\dot{y}(1) = y(2)$$

For a nonlinear system,

$$\sin\theta \neq \theta$$

The second order non-linear differential equation reduces to the following, two first order differential equation:

$$\dot{y}(1) = y(2)$$

$$\dot{y}(2) = (-g/l)\sin(y(1))$$

MATLAB Code

The MATLAB code is written following the procedure adopted to solve the spring-mass-damper system. For the linear case the function file is saved as 'linear.m'.

```
function yp = linear(t,y)
yp = [y(2);((-g/l) *(y(1)))];
```

In this example, the value of 'l' is chosen to be 5 cms. So the value of 'g/l' turns out to be –196.2 per square second. Note that since the mass and the length are expressed in grams and centimeters respectively, the value of the acceleration due to gravity 'g' is taken to be 981 cms per square second. Substitute this value in the above MATLAB code.

For the nonlinear case, open a new M-file to write a separate function file for the nonlinear differential equation. In this case the function file is saved as 'nonlinear.m'.

```
function yp = nonlinear(t,y)
yp = [y(2);((-g/l) *sin(y(1)))];
```

The main code should be written in a separate M-file, which will actually call the above function files, solve the derivatives stored in them and plot the required result.

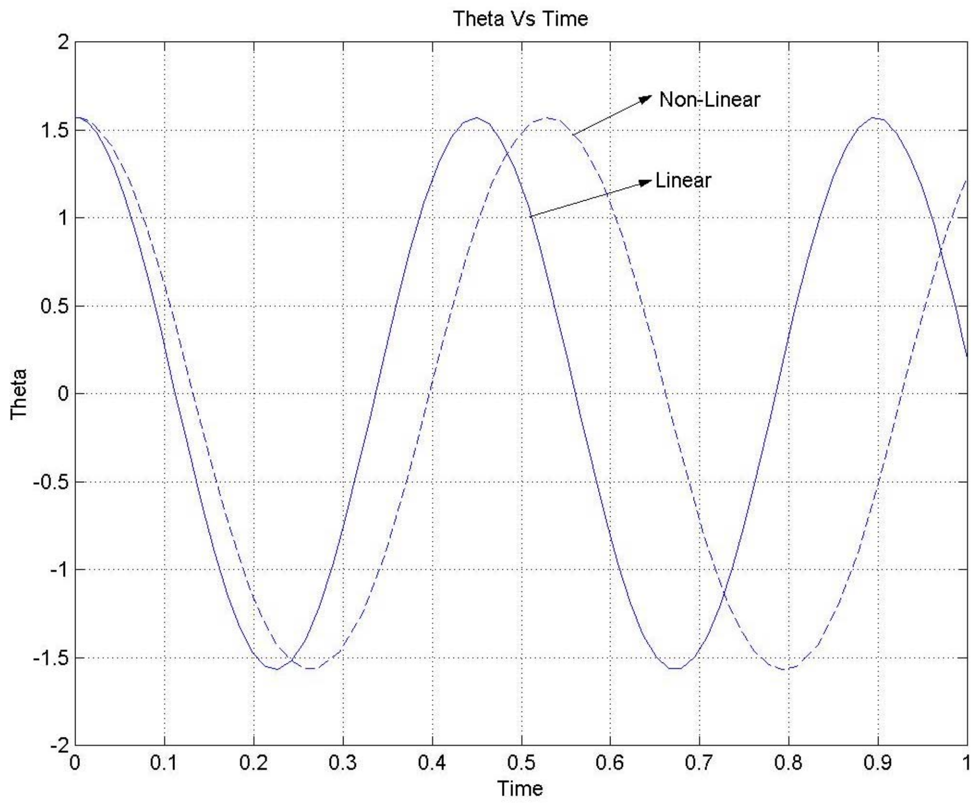
```
tspan = [0 5];
y0 = [1.57;0];
[t,y] = ode45('linear',tspan,y0)
plot(t,y(:,1))
grid on;
xlabel('Time')
ylabel('Theta')
title('Theta Vs Time')
hold on;
```

‘tspan’ represents the time interval and ‘y0’ represents the initial condition. Note that the value of θ in the initial conditions must be expressed in radians.

Now run the code for the linearized version of the simple pendulum. A plot of ‘theta’ versus ‘time’ will be obtained. The ‘hold’ command in the code freezes the plot obtained. This command is used so that the nonlinear version can also be plotted in the same graph. After running the program for the linear version, change the name of the function in the third line of the main code from ‘linear’ to ‘nonlinear’. By doing this, the Ode45 command will now solve the nonlinear differential equations and plot the result in the same graph. To differentiate between the linear and the non-linear curve, use a different line style.

The plots of the responses are attached. From the plots we can see that the time period for the non linear equation is greater than that of the linear equation.

To plot the angular velocity with respect to ‘time’ change the variable in the plot command line in the main code from ‘y(:,1)’ to ‘y(:,2)’. This is done because initially we assigned y(1) to ‘theta’ and y(2) to the first derivative of ‘theta’.


Theta Vs Time
Non-Linear
Linear
Theta
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Assignment

a) Compute and plot the response of a simple pendulum with viscous damping whose equation of motion is given by

$$ml^2\ddot{\theta} + c\dot{\theta} + mgl\theta = 0$$

Compute the value of the damping coefficients for $\xi = 0.1$.The initial conditions are

$\theta(0) = \{15°, 30°, 45°, 60°, 90°\}$ and $\dot{\theta}(0) = 0$. M = 10 g ; l = 5 cms.

b) Compare the generated plot with Non-linear plot for each value of $\theta(0)$.
What do you infer from the plots obtained?

c) Plot the natural frequency of the nonlinear solution versus initial condition $\theta(0)$.

(Note: Plot the linear and the nonlinear curve in the same plot. Remember to convert the initial values of theta given in degrees, to radians.)

Also make sure that the difference between the linear and the non-linear responses can be clearly seen from the plot. If necessary use the zoom feature in order to show the difference.

3. Coulomb Friction

Example

Solve the following differential equation for $\dot{\theta} \geq 0$.

$$\ddot{\theta}(1+\mu_d \sin\theta)+\mu_d \cos\theta\,\dot{\theta}^2 = \frac{g}{l}[\cos\theta(1-\mu_d^2)-\mu_d \sin\theta]$$

$\mu_d = \{0, 0.1, 0.2\}$; $l = 0.25m$. $\theta(0) = 0; \dot{\theta}(0) = 0$

Solution

As explained earlier, to solve the above differential equation by MATLAB, reduce it into two first order differential equations.

Let $\theta = y(1)$

$\dot{\theta} = y(2)$

The above equation reduces to

$$\dot{y}(1) = y(2)$$
$$\dot{y}(2) = \frac{g}{l(1+\mu_d \sin(y(1)))}\{\cos(y(1))[1-\mu_d^2]-\mu_d \sin(y(1))\} - \frac{1}{l(1+\mu_d \sin(y(1)))}\{\mu_d \cos(y(1))(y(2)\hat{}2)\}$$

MATLAB Code

A M-file must be written to store the derivatives. In this case the file is saved as 'coulomb.m'.

```
function yp =coulomb(t,y)
g=9.81;
mu=0;
l=0.25;
k1=1+mu*sin(y(1));
k2=mu*cos(y(1))*(y(2)^2);
k3=cos(y(1))*(1-mu^2);
k4=mu*sin(y(1));
yp=[y(2);((g/l)*((k3-k4)/k1)-(k2/k1))]
```

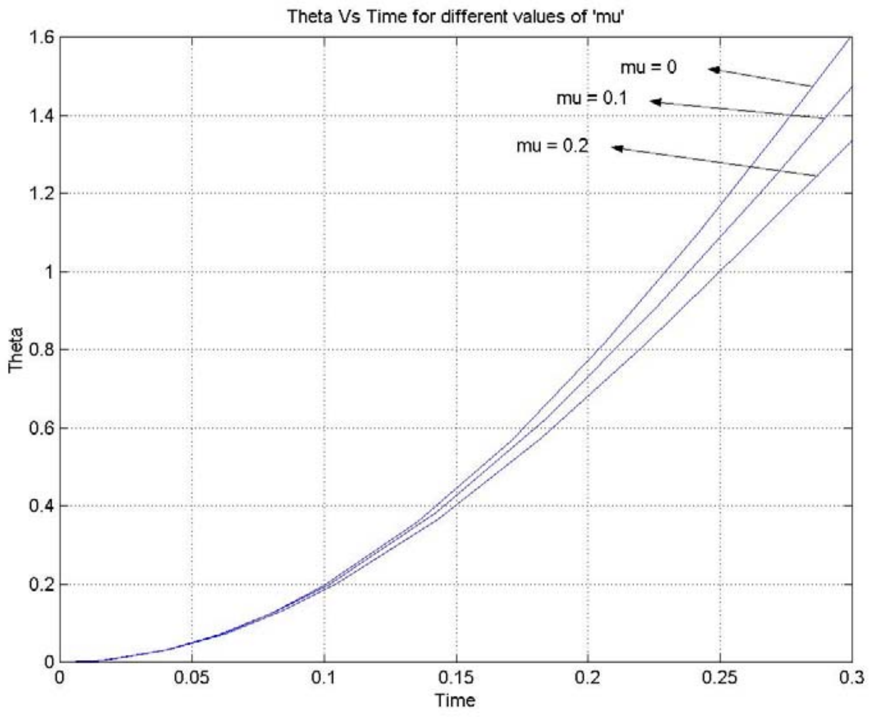
Type the following code in a new M-file.

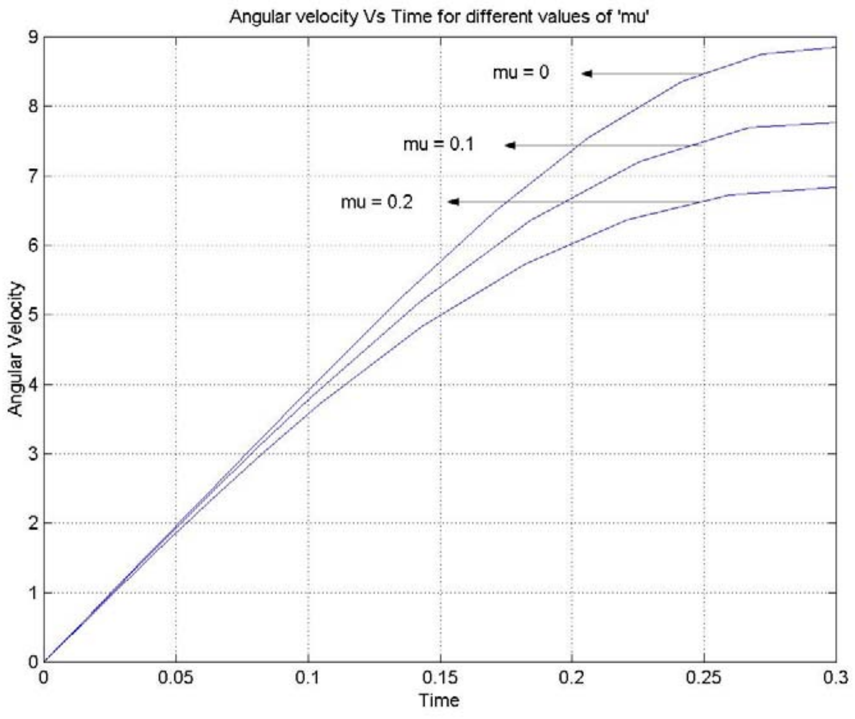
```
tspan=[0 2]
y0=[0;0]
[t,y]=ode45('coulomb',tspan,y0)
plot(t,y(:,1))
grid
xlabel('Time')
ylabel('Theta')
title('Theta Vs Time')
hold on;
```

This main file calls the function 'coulomb.m ' to solve the differential equations. 'Tspan' represents the time interval and 'y0' represents the initial condition vector. To plot the results for different values of μ, change the value of μ in the function file, save it and then run the main code.

The vector 'y(:,1)' represents the values of theta at the various time instants and the vector 'y(:,2)' gives the values of the angular velocities. To plot the angular velocity, change the variable in the plot command line in the main code from 'y (:,1)' to 'y(:,2)'. Also plot the angular velocity with respect to theta.

The plots are attached. From the plots it can be seen that by increasing the value of the coulomb damping constant the system takes more time to reach $\theta = \frac{\pi}{2}$ and the maximum value of the angular velocity also decreases.


Theta Vs Time for different values of 'mu'
mu = 0
mu = 0.1
mu = 0.2
Theta
Time
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
0
0.05
0.1
0.15
0.2
0.25
0.3


Angular velocity Vs Time for different values of 'mu'
mu = 0
mu = 0.1
mu = 0.2
Angular Velocity
Time
0
1
2
3
4
5
6
7
8
9
0
0.05
0.1
0.15
0.2
0.25
0.3

Assignment

Solve the following differential equation for $\dot{\theta} \geq 0$.

$$\ddot{\theta}(1+\mu_d \sin\theta)+\mu_d \cos\theta\,\dot{\theta}^2 = \frac{g}{l}[\cos\theta(1-\mu_d^2)-\mu_d \sin\theta]$$

$\mu_d=\{0, 0.05,0.1,0.15,0.2\}$; $l = 0.5m$. $\theta(0)=\pi/2; \dot{\theta}(0)=0$

a) Plot the maximum value of angular velocity and maximum value of theta with respect to μ

b) Is the differential equation valid for $\dot{\theta} < 0$. If not, what is the correct equation?

c) Analytically confirm the angular velocity predictions for $\mu = 0.0$ and $\mu = 0.2$.

d) Comment on the results obtained.

4. Trajectory Motion with Aerodynamic Drag

Example

Solve the following differential equations for c_d ={0.1, 0.3,0.5} N/m.

$$m\ddot{x} + c_d\,\dot{x}(\dot{x}^2 + \dot{y}^2)^{(1/2)} = 0$$

$$m\ddot{y} + c_d\,\dot{y}(\dot{x}^2 + \dot{y}^2)^{(1/2)} = -w$$

m = 10 kg. $\dot{x}(0) = 100$ m/s; $\dot{y}(0) = 10$ m/s; x(0) = 0; y(0) = 0.

Solution

As explained earlier, to solve the above two differential equations, we have to reduce each of them into two first order differential equations.

So Let

$x = y(1)$

$\dot{x} = y(2)$

$y = y(3)$

$\dot{y} = y(4)$

The above equations reduce to the following first order differential equations.

$$\dot{y}(1) = y(2)$$

$$\dot{y}(2) = (-\frac{c_d}{m}) * y(2) * [y(2)^2 + y(4)^2]^{\frac{1}{2}}$$

$$\dot{y}(3) = y(4)$$

$$\dot{y}(4) = (\frac{-w}{m}) - (\frac{c_d}{m}) * y(4) * [y(2)^2 + y(4)^2]^{\frac{1}{2}}$$

MATLAB Code

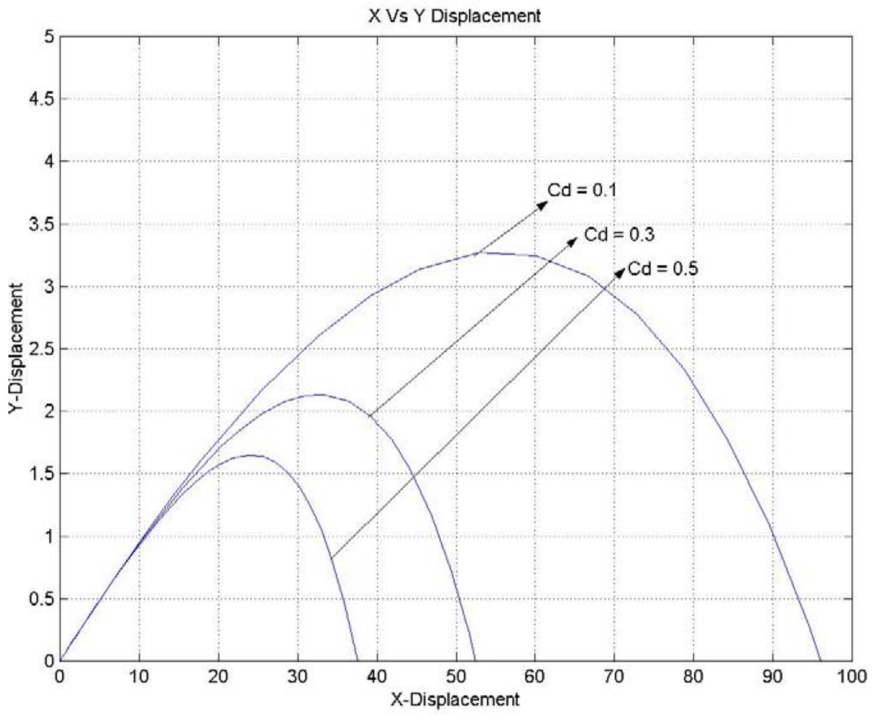
Type the following code in a separate m-file. As mentioned earlier, the system derivatives must be written in a separate m-file. Since the name of the function is ‘airdrag’, save the m-file as ‘airdrag.m’ in the current directory.

```
function yp = airdrag(t,y)
m = 10;
cd = 0.2;
g = 9.81;
w = m*g;
yp = zeros(4,1);
yp(1) = y(2);
yp(2) = ((-cd/m)*y(2)*(y(2)^2+y(4)^2)^(0.5));
yp(3) = y(4);
yp(4) = ((-w/m)-(cd/m)*y(4)*(y(2)^2+y(4)^2)^(0.5));
```

Open a new m-file and type the following code for the main program. This program calls the function ‘airdrag.m’ in order to solve the differential equations. Again ‘tspan’ represents the time interval and ‘y0’ represents the initial condition vector. The plots have to be drawn between the displacement in the x direction and the displacement in the y direction. Refer to the handout ‘Introduction to MATLAB’ for the explanation about using the ‘hold’ command with the help of a ‘counter’ variable.

```
tspan=[0 5];
y0=[0;100;0;10]
[t,y]=ode45('airdrag',tspan,y0);
plot(y(:,1),y(:,3))
grid
xlabel(‘X-Displacement’)
ylabel(‘Y-Displacement’)
title(‘X vs Y Displacement’)
hold on;
```

It can be seen from the plots that as the coefficient of the aerodynamic drag is increased the maximum value of the displacement in the y direction decreases. The value of the displacement in the x direction, for which the displacement in the y direction is maximum, also decreases. Here the plots have been zoomed to the current version so that the difference in the curves could be seen clearly.


X Vs Y Displacement
Cd = 0.1
Cd = 0.3
Cd = 0.5
Y-Displacement
X-Displacement
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
10
20
30
40
50
60
70
80
90
100

Assignment

Solve the following differential equations for $c_d = \{0.0, 0.1, 0.2, 0.3\}$ N/m.

$$m\ddot{x} + c_d\,\dot{x}(\dot{x}^2 + \dot{y}^2)^{(1/2)} = 0$$

$$m\ddot{y} + c_d\,\dot{y}(\dot{x}^2 + \dot{y}^2)^{(1/2)} = -w$$

m = 5 kg. $\dot{x}(0) = 10$ m/s; $\dot{y}(0) = 200$ m/s; x(0) = 0; y(0) = 0.

a) Plot separately the maximum range and the maximum elevation with respect the damping coefficient, $C_{d.}$

b) Analytically confirm the results for $C_d = 0$.

c) Comment on the results obtained.

5. Pendulum with Aerodynamic and Viscous Damping

Example

Solve the following non-linear differential equation with aerodynamic damping

$$\ddot{\theta} + \frac{g}{l}\sin\theta + \frac{C_a}{ml}\dot{\theta}^2 \operatorname{sgn}(\dot{\theta}) = 0 \qquad (1)$$

and compare it with the following linearized differential equation with viscous damping

$$\ddot{\theta} + \frac{C_d}{m}\dot{\theta} + \frac{g}{l}\theta = 0 \qquad (2)$$

m = 10 g; l = 5 cms; $C_a = C_d = 14$; $\theta(0) = 1.57; \dot{\theta}(0) = 0$.
(Note: The initial value of theta is given in radians.)

Solution

As mentioned earlier, to solve the above differential equations, convert them into two first order differential equations.

Let

$\theta = y(1)$

$\dot{\theta} = y(2)$

Equation (2) becomes

$$\dot{y}(1) = y(2) \qquad (3)$$

$$\dot{y}(2) = (-\frac{g}{l}) * (y(1)) - (\frac{C_d}{m}) * (y(2)) \qquad (4)$$

Similarly equation (1) becomes

$$\dot{y}(1) = y(2) \qquad (5)$$

$$\dot{y}(2) = (-\frac{g}{l}) * \sin(y(1)) - (\frac{C_d}{ml}) * (y(2)\char`^2) \operatorname{sgn}(y(2)) \qquad (6)$$

MATLAB Code

In this example, we are required to solve two separate problems, i.e., a pendulum with aerodynamic drag and a pendulum with viscous damping. So, we have to write two separate M-files, 'pendulum1.m', which contains the derivatives of the problem with aerodynamic drag and 'pendulum2.m', which contains the derivatives of the problem with viscous damping.

Open a new m-file and type the following code which contains equations (5) and (6). Note that the derivatives are represented in the form of a vector. Also note that the signum function, ‘sgn’ has to be written as ‘sign’ in MATLAB.

```
function yp = pendulum1(t,y)
m=10;
g=981;
l=5;
Ca=14;
yp=[y(2);((-g/l)*sin(y(1))-Ca/(m*l)*(y(2)^2)*sign(y(2)))];
```

In a new m-file type in the following main program, which calls the function, named ‘pendulum1.m’ and solves the non-linear differential equation and plots it.

```
tspan=[0 10];
y0=[1.57;0]
[t,y]=ode45('pendulum1',tspan,y0);
plot(t,y(:,1));grid
xlabel(‘Time’)
ylabel(‘Theta’)
title(‘Theta Vs Time’)
hold on;
```

Open a new m-file type the following code. This file contains equations (3) and (4).

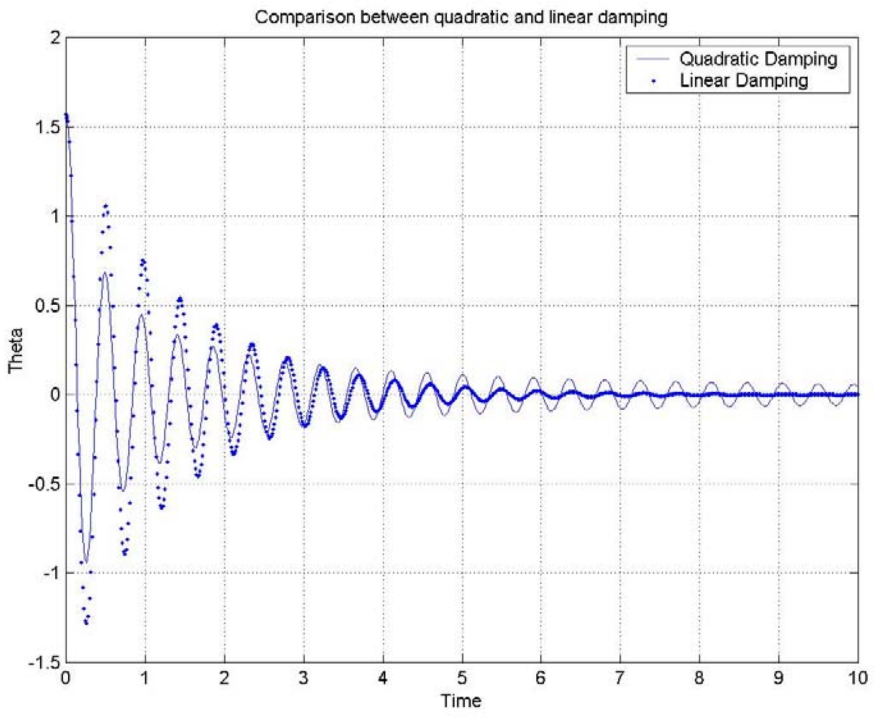
```
function yp = pendulum2(t,y)
m=10;
g=981;
l=5;
Ca=14;
yp=[y(2);((-g/l)*(y(1))-(Ca/m)*(y(2)))];
```

In a new m-file type the following main program. This code calls the function and solves the linearized differential equation and plots it.

```
tspan=[0 10];
y0=[1.57;0]
[t,y]=ode45('pendulum2',tspan,y0);
plot(t,y(:,1));grid
xlabel(‘Time’)
ylabel(‘Theta’)
title(‘Theta Vs Time’)
hold on;
```

As mentioned in the earlier examples, ‘tspan’ represents the time interval and ‘y0’ represents the initial condition vector. The vector ‘y(:,1)’ gives the values of theta at the various time instants and the vector ‘y(:,2)’ gives the values of the angular velocity. To plot the angular velocity, change the variable ‘y(:,1)’ in the plot command line to ‘y(:,2)’.

From the plots it can be seen that, quadratic damping causes a rapid initial amplitude reduction but is less effective as time increases and amplitude decreases. Whereas, linear damping is less effective initially but is more effective as time increases.



Assignment

a) Solve the following non-linear differential equation with aerodynamic damping

$$\ddot{\theta} + \frac{g}{l}\sin\theta + \frac{C_a}{ml}\dot{\theta}^2 \operatorname{sgn}(\dot{\theta}) = 0 \qquad (1)$$

and compare it with the following linearized differential equation with viscous damping

$$\ddot{\theta} + \frac{C_d}{m}\dot{\theta} + \frac{g}{l}\theta = 0 \qquad (2)$$

m = 5 g; l = 10 cms; $C_a = C_d = 140$; $\theta(0) = 0.5233$; $\dot{\theta}(0) = 0$.

Show the results in the same plot.

b) Comment on the results obtained. Assuming a harmonic motion, i.e., $\theta = A\cos(\omega_n t)$, for what amplitude 'A', are the linear and the non-linear damping moments equal? Also confirm the result from the plot obtained.

6. A Half Cylinder rolling on a Horizontal Plane

Example

The general governing differential equation of motion of the above problem is

$$(\frac{3mr^2}{2} - 2mer\sin\beta)\ddot{\beta} - mer\cos\beta\,\dot{\beta}^2 = we\cos\beta\ . \qquad (1)$$

Solve the above differential equation for m = 5 Kg; $e = \frac{4r}{3\pi}$ m; r = 0.1 m.

Solution

To solve the differential equation, convert the equation into two first order differential equation.

With

$\beta = y(1);$

$\dot{\beta} = y(2);$

Equation (1) reduces to the following equations

$$\dot{y}(1) = y(2);$$
$$\dot{y}(2) = \frac{[we\cos(y(1)) + mer\cos(y(1))(y(2)\hat{}2)]}{[1.5mr\hat{}2 - 2mer\sin(y(1))]}$$

MATLAB Code

The function file 'tofro.m' contains the systems derivatives in the form of a vector.

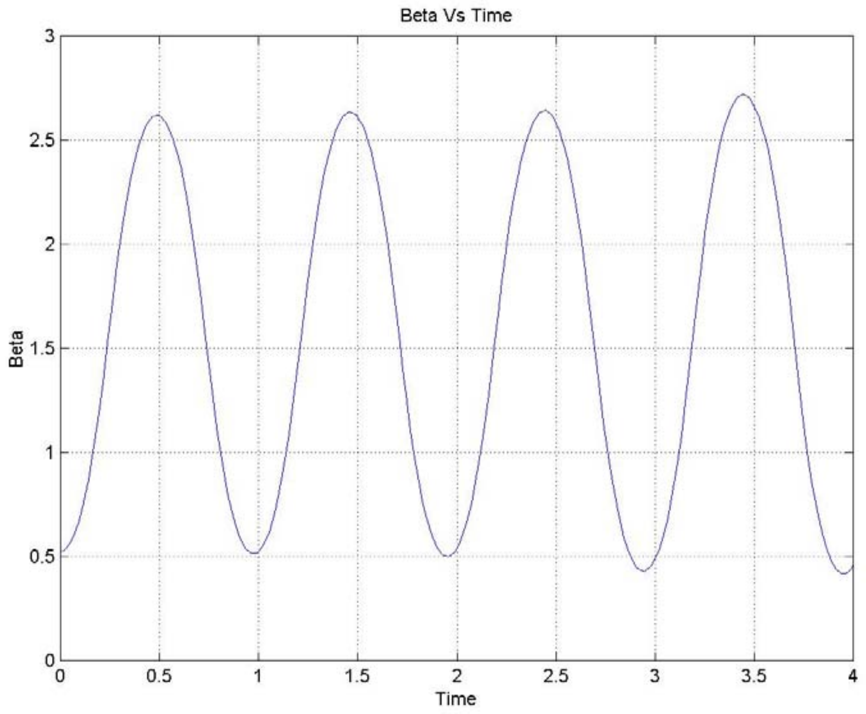
```
function yp = tofro(t,y)
m=5;
r=0.1;
e=(4*r)/(3*pi);
g=9.81;
yp=[y(2);((m*g)*e*cos(y(1))+m*e*r*cos(y(1))*(y(2)^2))/(1.5*m*r*r
-2*m*e*r*sin(y(1)))]
```

The main code, which is given below calls the function 'tofro.m', which contains the system differential equation and plots the displacement with respect to time. Since there is no damping present, the plot resembles a sinusoid of constant amplitude.

```
tspan=[0 4];
```

```
y0=[0.5233;0];
[t,y]=ode45('tofro',tspan,y0);
plot(t,y(:,1));grid
xlabel('Time')
ylabel('Beta')
title('Beta Vs Time')
```

The plot is attached below.



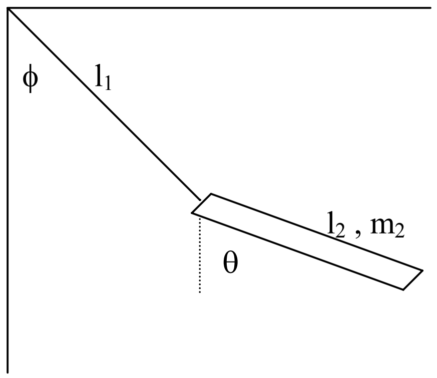
Assignment

a) Solve the differential equation given in the example for m = 10 Kg; $e = \frac{4r}{3\pi}$ m; r = 0.2 m. Plot 'β' with respect to time. Compare the result with that obtained in the example and comment on the result.

b) Determine the equilibrium value of 'β'.
c) Linearize the differential equation for small motion about the equilibrium value and determine the natural frequency for the small motion. Also plot the value of the 'β' with respect to time. Note that for plotting the value of 'β', you have to solve the linearized differential equation.
d) Compare the natural frequency of the linearized model to that produced by the non- linear model.

7. A bar supported by a cord

Derive the governing differential equation of motion of a swinging bar supported at its ends by a cord. Calculate the eigen values and the eigen vectors. Solve the derived differential equation and plot the initial response of the system for the following initial conditions:
a) Arbitrary initial condition
b) Initial condition vector proportional to the eigen vector.
Choose $l_1 = 1$m; $l_2 = 1$m; $m_2 = 5$Kg



The differential equations of motion for the above system when represented in a matrix form is

$$\begin{bmatrix} m_2 l_1 & \dfrac{m_2 l_2 \cos(\theta - \phi)}{2} \\ \dfrac{m_2 l_1 l_2 \cos(\theta - \phi)}{2} & \dfrac{m_2 l_2^{\,2}}{3} \end{bmatrix} \begin{Bmatrix} \ddot{\phi} \\ \ddot{\theta} \end{Bmatrix} = \begin{Bmatrix} -w_2 \sin\phi + \dfrac{m_2 l_2 \dot{\theta}^2 \sin(\theta - \phi)}{2} \\ \dfrac{-w_2 l_2 \sin\theta + m_2 l_1 l_2 \dot{\phi}^2 \sin(\theta - \phi)}{2} \end{Bmatrix}$$

Eliminating the second order terms in the above equation gives the linear equation

$$\begin{bmatrix} m_2 l_1^{\,2} & \dfrac{m_2 l_2 l_1}{2} \\ \dfrac{m_2 l_1 l_2}{2} & \dfrac{m_2 l_2^{\,2}}{3} \end{bmatrix} \begin{Bmatrix} \ddot{\phi} \\ \ddot{\vartheta} \end{Bmatrix} + \begin{bmatrix} l_1 w_2 & 0 \\ 0 & \dfrac{w_2 l_2}{2} \end{bmatrix} \begin{Bmatrix} \phi \\ \theta \end{Bmatrix} = 0$$

Note that the inertia matrix has been made symmetric via a multiplication of the first row by l_1.

MATLAB Code

To find the eigen values and the eigen vectors

To find the eigen values and the eigen vectors, the linearized version of the model is used. The mass of the bar was considered to be 5 Kg and the length of the cord and the bar to be 1m. The MATLAB code is given below.

```
% to find the eigen values and the eigen vectors.
l1=1;
l2=1;
m2=5;
g=9.81;
w2=m2*g;
M=[m2*l1^2 (m2*l1*l2)/2;(m2*l1*l2)/2 (m2*l1^2)/3];
K=[l1*w2 0;0 (w2*l2)/2];

[v,d]=eig(K,M)
```

The columns ‘v’ represent the eigen vectors corresponding to each eigen value ‘d’.
‘M’ represents the inertia matrix and ‘K’ represents the stiffness matrix. The square root of the eigen values gives the natural frequencies of the system.

The eigen values were found to be 6.2892 and 91.8108 and the corresponding eigen vectors were $\begin{Bmatrix} 0.6661 \\ 0.7458 \end{Bmatrix}$ and $\begin{Bmatrix} -0.4885 \\ 0.8726 \end{Bmatrix}$ respectively.

Initial response

To plot the initial response of the system the original nonlinear differential equation has to be used. Moreover the second order differential equation has to be converted into a vector of first

$\phi = y(1);$

$\dot{\phi} = y(2);$

$\theta = y(3);$

$\dot{\theta} = y(4);$

order differential equation. This step is done by following the procedure given below.
Substituting the above relations in the original nonlinear differential equation, we get the following nonlinear first order differential equation, which when represented in matrix form is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_2 l_1 & 0 & \dfrac{m_2 l_2 \cos(y(3)-y(1))}{2} \\ 0 & 0 & 1 & 0 \\ 0 & \dfrac{m_2 l_1 l_2 \cos(y(3)-y(1))}{2} & 0 & \dfrac{m_2 l_2{}^2}{3} \end{bmatrix} \begin{Bmatrix} \dot{y}(1) \\ \dot{y}(2) \\ \dot{y}(3) \\ \dot{y}(4) \end{Bmatrix} = \begin{Bmatrix} y(2) \\ -w_2 \sin\phi + \dfrac{m_2 l_2 \theta^2 \sin(y(3)-y(1))}{2} \\ y(4) \\ \dfrac{-w_2 l_2 \sin\theta + m_2 l_1 l_2 \dot{\phi}^2 \sin(\theta-\phi)}{2} \end{Bmatrix}$$

MATLAB Code

In this type of a problem where the inertia matrix (mass matrix) is a function of the states or the variables, a separate M-file has to be written which incorporates a switch/case programming with a flag case of ‘mass’.

For example if the differential equation is of the form,

M (t, y) *y’ (t)=F (t, y),

then the right hand side of the above equation has to be stored in a separate m-file called ‘F.m’. Similarly the inertia matrix (mass matrix) should also be stored in a separate m-file named ‘M.m’. So, when the flag is set to the default, the function ‘F.m’ is called and later when the flag is set to ‘mass’ the function ‘M.m’ is called.

The code with the switch/case is given below. Note that it is a function file and should be saved as ‘indmot_ode.m’ in the current directory.

```
function varargout=indmot_ode(t,y,flag)

switch flag
case ''                           %no input flag
   varargout{1}=FF(t,y);
case 'mass'                        %flag of mass calls mass.m
   varargout{1}=MM(t,y);
otherwise
   error(['unknown flag ''' flag '''.']);
end
```

To store the right hand side of the original matrix form of differential equation, a separate function file must be written as shown below. Note that the name of the function is ‘FF’, so this file must be saved as ‘FF.m’.

```
%the following function contains the right hand side of the
%differential equation of the form
%M(t,y)*y'=F(t,y)
%i.e. it contains F(t,y).it is also stored in a separate file
named, FF.m.

function yp=FF(t,y)
l1=1;
l2=1;
m2=5;
g=9.81;
w2=m2*g;
yp=zeros(4,1);
```

```
yp(1)=y(2);
yp(2)=-w2*sin(y(1))+ (m2*l2/2)*(y(4)^2)*sin(y(3)-y(1));
yp(3)=y(4);
yp(4)=(-w2*l2*sin(y(3)))/2+(m2*l1*l2/2)*(y(2)^2)*sin(y(3)-y(1));
```

Similarly, to store the mass matrix a separate function file is written which is stored as ‘MM.m’.

```
% the following function contains the mass matrix.
%it is separately stored in a file named, MM.m

function n = MM(t,y)
l1=1;
l2=1;
m2=5;
g=9.81;
w2=m2*g;
n1=[1 0 0 0];
n2=[0 m2*l1 0 (m2*l2/2)*cos(y(3)-y(1))];
n3=[0 0 1 0];
n4=[0 (m2*l1*l2/2)*cos(y(3)-y(1)) 0 m2*l2*l2/3];
n=[n1;n2;n3;n4];
```

To plot the response, the main file should call the function ‘indmot_ode.m’, which has the switch/case programming which in turn calls the corresponding functions depending on the value of the flag. For the main file to recognize the inertia matrix, the MATLAB command ODESET is used to set the mass to ‘M (t, y)’.

This MATLAB code for the main file should be written in the same M-file, following the code to solve for the eigen values and eigen vectors.

```
tspan=[0 30]

y0=[0.5233;0;1.0467;0]       %   Arbitrary   Initial   condition
x01=[v(1,1);0;v(2,1);0]     %  Low Frequency mode initial
                               Condition
x02=[v(1,2);0;v(2,2);0]     %  High frequency mode initial
                               Condition

options=odeset('mass','M(t,y)')
[t,y]=ode113('indmot_ode',tspan,y0,options)
subplot(2,1,1)
plot(t,y(:,1))
grid
xlabel('Time')
ylabel('phi')

subplot(2,1,2)
plot(t,y(:,3))
grid
xlabel('Time')
ylabel('Theta')
```

The above code plots the values of ‘theta’ and ‘phi’ with respect to time for the arbitrary initial condition case. To plot for the initial condition case where the initial conditions are proportional to the eigen vectors, change the variable ‘y0’ in line 6 of the code to first ‘x01’ and then to ‘x02’.

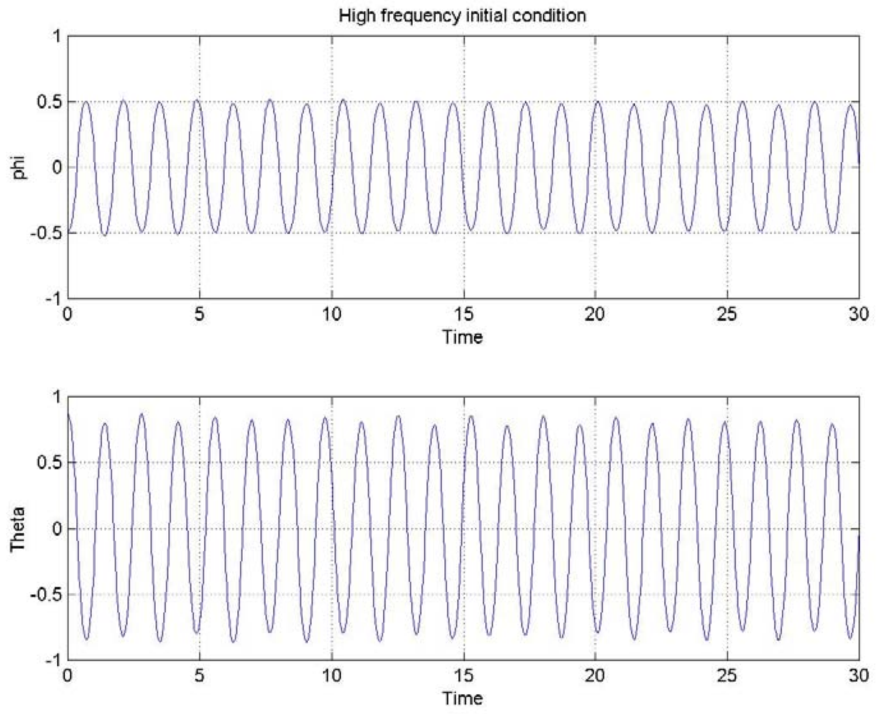
Notice the command “ subplot”.

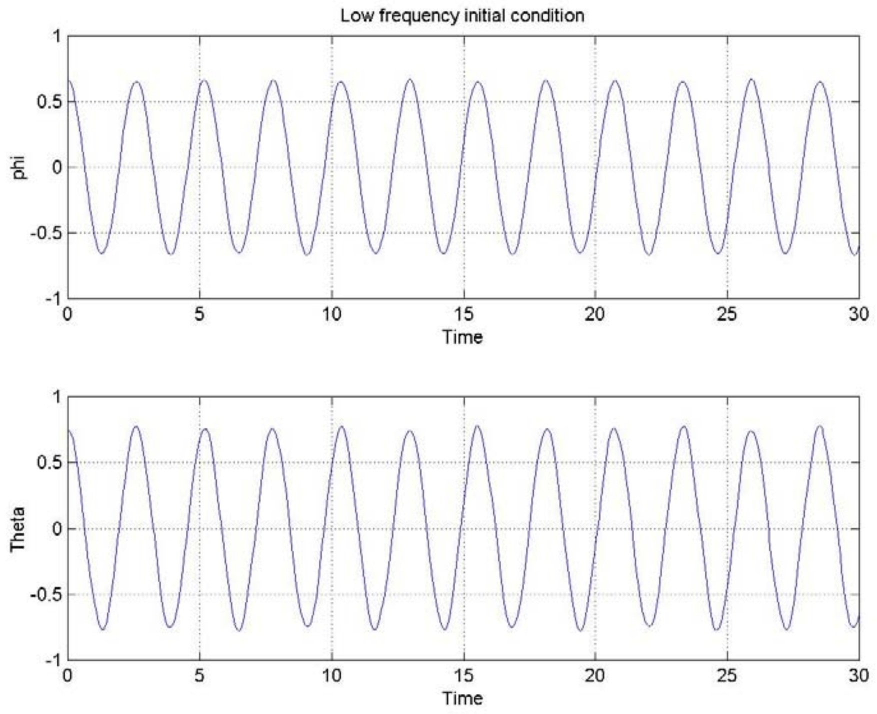
subplot(m,n,p), breaks the Figure window into an m-by-n matrix of small axes and selects the p-th axes for the current plot. The axes are counted along the top row of the Figure window, then thesecond row, etc. For example,

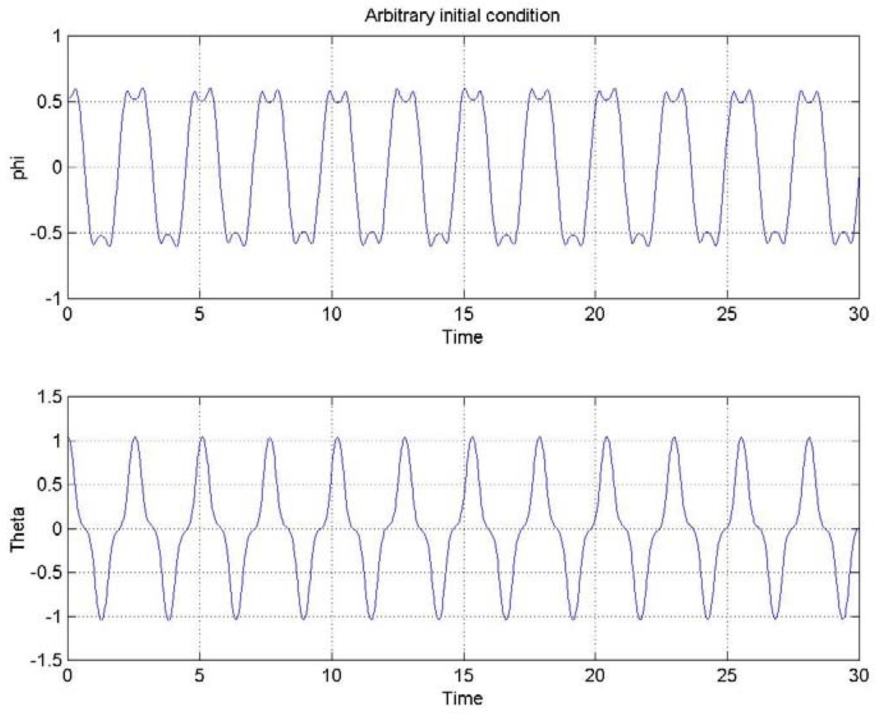
    subplot(2,1,1), plot(income)
    subplot(2,1,2), plot(outgo)

plots “income” on the top half of the window and “outgo” on the bottom half.

From the plots it can be observed that: a) For initial condition vector that is proportional to the eigen vectors, the response amplitudes are harmonic at their corresponding natural frequency. b) For an arbitrary initial condition, we get a coupled response displaying response at both the natural frequencies. The plots are attached below


High frequency initial condition
phi
Time
Theta
Time
1
0.5
0
-0.5
-1
0
5
10
15
20
25
30


Low frequency initial condition
phi
Time
Theta
Time
1
0.5
0
-0.5
-1
0
5
10
15
20
25
30


Arbitrary initial condition
phi
Time
Theta
Time
1
0.5
0
-0.5
-1
1.5
-1.5
0
5
10
15
20
25
30

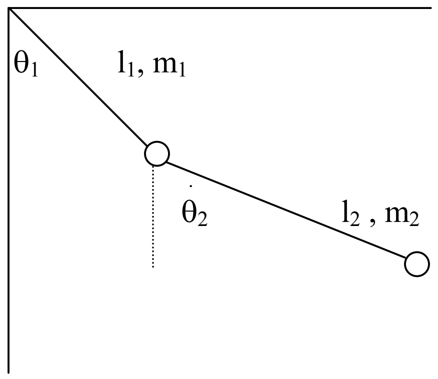
Assignment

Solve the above-derived differential equation and plot the initial response of the system for the following initial conditions:
a) Arbitrary initial condition
b) Initial condition vector proportional to the eigen vector.
Choose l1=2m; l2=4m; m2=10Kg.

Compare the results obtained from that of the example and comment on the plots obtained.

8. Double Pendulum

Derive the governing differential equation of motion of a double pendulum. Solve the derived differential equation and plot the values of θ₁ and θ₂ with respect to time.
Choose l_1 = 1m; l_2 = 1m; $m_1 = m_2$ = 5Kg. The initial conditions for θ_1 and θ_2 are 0.5233 and 0.5233 radians respectively.



Solution

When represented in a matrix form, the differential equations of motion for the above system is

$$\begin{bmatrix} (m_2+m_1)l_1 & m_2l_2\cos(\theta_2-\theta_1) \\ m_2l_1\cos(\theta_2-\theta_1) & m_2l_2 \end{bmatrix} \begin{Bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{Bmatrix} = \begin{Bmatrix} -(w_1+w_2)\sin\theta_1 + m_2l_2\dot{\theta}_2^{\,2}\sin(\theta_2-\theta_{1)} \\ -w_2\sin\theta_2 - m_1l_1\dot{\theta}_1^{\,2}\sin(\theta_2-\theta_{1)} \end{Bmatrix}$$

MATLAB Code

These coupled second order differential equations must be converted into a vector of first order differential equation. This step is done by the following substitutions.

$\theta_1 = y(1);$

$\dot{\theta}_1 = y(2);$

$\theta_2 = y(3);$

$\dot{\theta}_2 = y(4);$

Substituting the above relations in the original nonlinear differential equation, we get the following first order differential equations,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & (m_2+m_1)l_1 & 0 & m_2l_2\cos(\theta_2-\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & m_2l_1\cos(\theta_2-\theta_1) & 0 & m_2l_2 \end{bmatrix} \begin{Bmatrix} \dot{y}(1) \\ \dot{y}(2) \\ \dot{y}(3) \\ \dot{y}(4) \end{Bmatrix} = \begin{Bmatrix} y(2) \\ -(w_1+w_2)\sin\theta_1 + m_2l_2\dot{\theta}_2^{\,2}\sin(\theta_2-\theta_1) \\ y(4) \\ -w_2\sin\theta_2 - m_1l_1\dot{\theta}_1^{\,2}\sin(\theta_2-\theta_1) \end{Bmatrix}$$

In this type of a problem where the coefficient matrix is a function of the states or the variables, a separate M-file must be written which incorporates a switch/case programming with a flag case of ‘mass’.

For example if the differential equation is of the form,

M (t, y) *y’ (t) = F (t, y),

then the right hand side of the above equation has to be stored in a separate m-file called ‘F.m’. Similarly the coefficient matrix should be stored in a separate m-file named ‘M.m’. So, when the flag is set to the default, the function ‘F.m’ is called and later when the flag is set to ‘mass’ the function ‘M.m’ is called.

The code with the switch/case is given below. Note that it is a function file and should be saved as ‘indmot_ode.m’ in the current directory.

```
function varargout=indmot_ode(t,y,flag)

switch flag
case ''                      %no input flag
   varargout{1}=pend(t,y);
case 'mass'                   %flag of mass calls mass.m
   varargout{1}=mass(t,y);
otherwise
   error(['unknown flag ''' flag '''.']);
end
```

To store the right hand side of the state variable matrix form of the model, a separate function file must be written as shown below. Note that the name of the function is ‘FF’, so this file must be saved as ‘FF.m’.

```
%the following function contains the right hand side of the
%differential equation of the form
%M(t,y)*y'=F(t,y)
%i.e. it contains F(t,y).it is also stored in a separate file
named, pend.m.

function yp= pend(t,y)
M1=5;
M2=5;
g=9.81;
l1=1;
l2=1;
w2=M2*9.81;
w1=M1*9.81;
yp=zeros(4,1);
```

```
yp(1)=y(2);
yp(2)=-(w1+w2)*sin(y(1))+M2*l2*(y(4)^2)*sin(y(3)-y(1));
yp(3)=y(4);
yp(4)=-w2*sin(y(3))-M2*l1*(y(2)^2)*sin(y(3)-y(1));
```

Similarly, to store the coefficient matrix a separate function file is written which is stored as 'MM.m'.

```
% the following function contains the mass matrix.
%it is separately stored in a file named, mass.m

function m = mass(t,y)
M1=5;
M2=5;
g=9.81;
l1=1;
l2=1;
m1=[1 0 0 0];
m2=[0 (M1+M2)*l1 0 M2*l2*cos(y(3)-y(1))];
m3=[0 0 1 0];
m4=[0 M2*l1*cos(y(3)-y(1)) 0 M2*l2];
m=[m1;m2;m3;m4];
```

To plot the response, the main file should call the function 'indmot_ode.m', which has the switch/case programming which in turn calls the corresponding functions depending on the value of the flag. For the main file to recognize the coefficient matrix, the MATLAB command ODESET is used to set the mass to 'M (t, y)'.

This MATLAB code for the main file should be written in the same M-file, following the code to solve for the eigen values and eigen vectors.

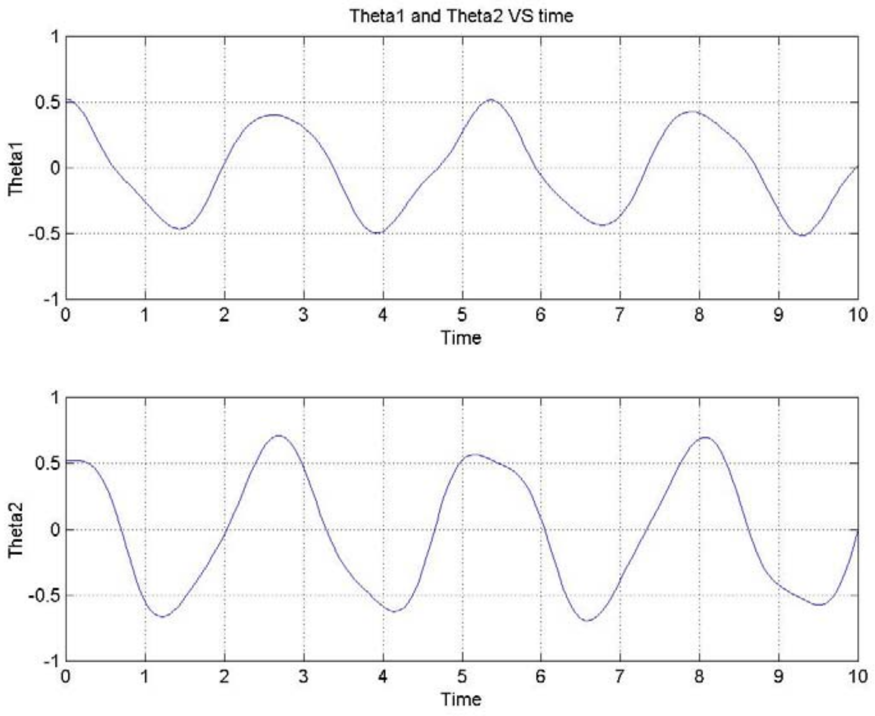
```
% this is the main file, which calls the equations and solves
using ode113.
%it then plots the first variable.

tspan=[0 10]
y0=[0.5233;0;0.5233;0]
options=odeset('mass','M(t,y)')
[t,y]=ode113('indmot_ode',tspan,y0,options)
subplot(2,1,1)
plot(t,y(:,1))
grid
xlabel('Time')
ylabel('Theta1')
```

```
subplot(2,1,2)
plot(t,y(:,3))
grid
xlabel('Time')
ylabel('Theta2')
```

For help regarding ‘subplot’ refer to the previous example or you can look in the online help in MATLAB by typing the following statement at the command prompt.

help subplot

The plot has been attached.


Theta1 and Theta2 VS time
Theta1
Time
Theta2
Time
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10

Assignment

1.) For the double pendulum problem, calculate the eigen values and eigen vectors.

2.) Solve the above-derived differential equation and plot the initial response of the system for the following initial conditions:

a) Arbitrary initial condition

b) Initial condition vector proportional to both eigen vectors.

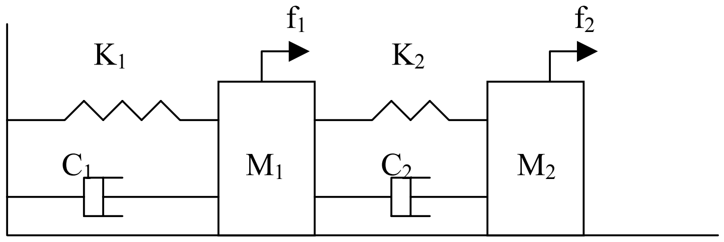
Choose l_1 = 2m; l_2 = 4m; $m_1 = m_2$ = 10Kg.

What do you infer from the plots?

9. Frequency response of systems having more than one degree of freedom

Example

Consider the following system as shown in the figure below. Derive the governing differential equation of motion and plot the amplitude and the phase of the masses m_1 and m_2 with respect to the frequency. Choose m_1 = 1Kg; m_2 = 2Kg; $k_1 = k_2$ = 1N/m; $c_1 = c_2$ = 0.001 N sec/m. The amplitude of the excitation force vector is 1 N.



Solution

The governing differential equation of motion for the above system when written in matrix form is given by

$$\begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}\begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix} + \begin{bmatrix} (c_1+c_2) & -c_2 \\ -c_2 & c_2 \end{bmatrix}\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{Bmatrix} + \begin{bmatrix} (k_1+k_2) & -k_2 \\ -k_2 & k_2 \end{bmatrix}\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = \begin{Bmatrix} f_1(t) \\ f_2(t) \end{Bmatrix}$$

The above equation can be written in a more simplified manner,

$$[M]\left(\ddot{x}_i\right) + [C]\left(\dot{x}_i\right) + [K](x_i) = (f_{io})\sin\omega\ t$$

where [M] is the inertia matrix, [C] is the damping matrix and [K] is the stiffness matrix.

Assuming the steady state solution of the differential equation to be

$$x_{iss} = x_{is}\sin\omega\ t + x_{ic}\cos\omega\ t$$

where i = 1,2.

Substituting the values of x_1 and x_2 and their corresponding first and second derivatives in the original matrix differential equation results in a single matrix equation given by

$$\begin{bmatrix} -\omega^2[M]+[K] & \omega[C] \\ -\omega[C] & -\omega^2[M]+[K] \end{bmatrix} \begin{Bmatrix} x_{ic} \\ x_{is} \end{Bmatrix} = \begin{Bmatrix} 0 \\ f_{io} \end{Bmatrix}$$

The solution of the above given matrix equation is normally given in terms of the amplitude and the phase as defined by

$$X_i = (x_{ic} + x_{is})^{\frac{1}{2}}$$

$$\phi_i = \tan^{-1}(\frac{x_{ic}}{x_{is}})$$

The mass matrix, the damping matrix and the stiffness matrix are given as below

$$[M] = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \text{ Kg}$$

$$[K] = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \text{ N/m}$$

$$[C] = \begin{bmatrix} 0.002 & -0.001 \\ -0.001 & 0.001 \end{bmatrix} \text{ N sec/m}$$

The amplitude of the excitation force vector is 1 N.

MATLAB Code

The objective of the problem is to find the amplitude and phase and plot them with respect to the excitation frequency.

The MATLAB code is attached below.

```
%**********************************************************%
%Defining the frequency vector and the mass matrix, %damping
matrix, the stiffness matrix and the amplitude of %the
excitation force.
%**********************************************************%

f=linspace(0,0.7,50);
m=[1 0;0 2];
k=[2 -1;-1 1];
c=[0.002 -0.001;-0.001 0.001];
fi=[1;1];
```

```
%*********************************************************%
%Calculating the amplitude and the phase for each frequency
%defined by the frequency vector.
%*********************************************************%

for i = 1:50
   omega(i)=2*pi*f(i);        %omega in terms of frequency
   omega2(i)=omega(i)*omega(i); % squaring omega
   a11=-omega2(i)*m+k;      % representing the left hand…
   a12=omega(i)*c;          %  matrix of the single matrix…
   a21=-omega(i)*c;         %  equation
   a22=-omega2(i)*m+k;
   a=[a11 a12;a21 a22];
   b=inv(a);
   c1=[0;0;fi];
   d(1,i)=b(1,:)*c1;
   d(2,i)=b(2,:)*c1;
   d(3,i)=b(3,:)*c1;
   d(4,i)=b(4,:)*c1;
   x(1,i)=sqrt(abs(d(1,i))^2+abs(d(3,i))^2);
   x(2,i)=sqrt(abs(d(2,i))^2+abs(d(4,i))^2);
   p(1,i)=atan(d(1,i)/d(3,i))*180/pi;

   if p(1,i)<0           % to check whether the angle is
                           negative or not.
      p(1,i)=180+p(1,i);
  else

      p(1,i)=p(1,i);
   end

   p(2,i)=atan(d(2,i)/d(4,i))*180/pi;

    if p(2,i)<0
      if d(4,i)<0
         p(2,i) = -180 + p(2,i)
      else
         p(2,i)=180+p(2,i);
      end

   else
      p(2,i)=p(2,i);
   end


end
```

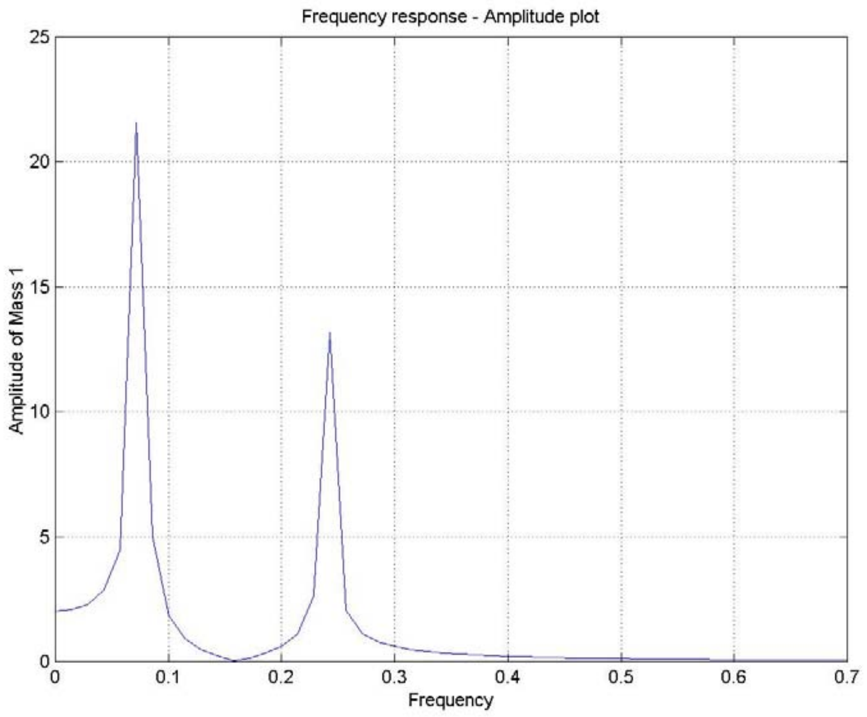
```
figure(1)
plot(f,x(1,:));grid
xlabel('Frequency')
ylabel('Amplitude of Mass 1')


figure(2)
plot(f,x(2,:));grid
xlabel('Frequency')
ylabel('Amplitude of Mass 2')

figure(3)
plot(f,p(1,:));grid
xlabel('Frequency')
ylabel('Phase of Mass 1')

figure(4)
plot(f,p(2,:));grid
xlabel('Frequency')
ylabel('Phase of Mass 2')
```

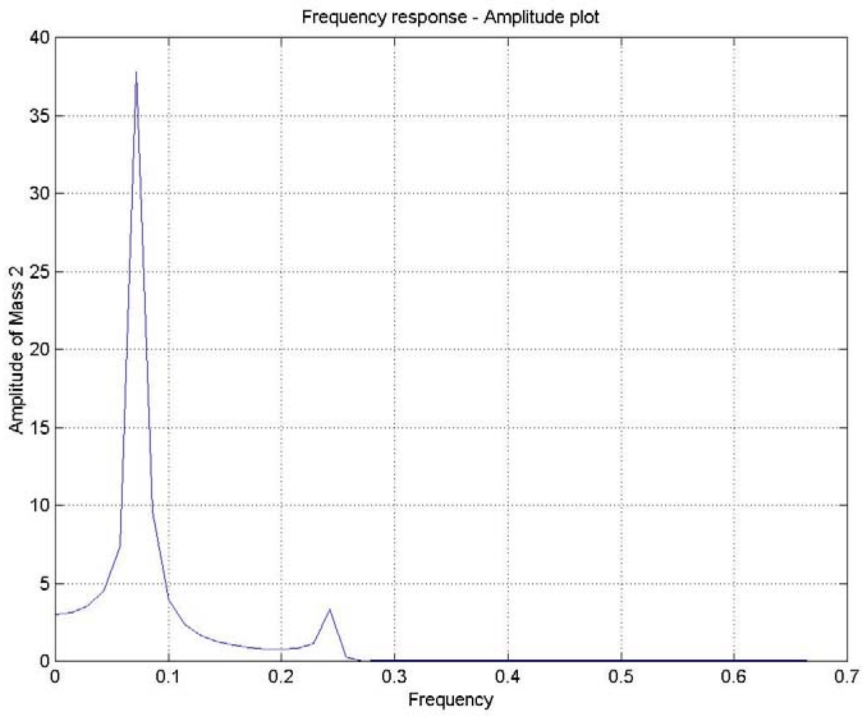
In the above code, a command 'linspace ' is used. linspace(x1, x2, N) generates a row vector of N points between x1 and x2.

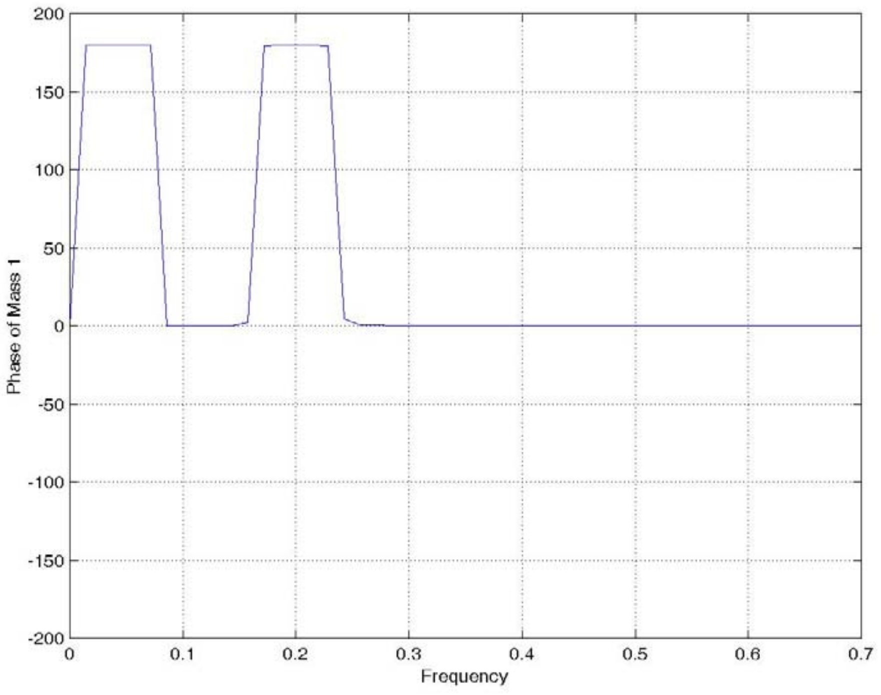
The above code calculates the amplitude and phase of each of the masses and plots them with respect to the excitation frequency.

where x(1,:) and x(2,:) are the amplitudes of the two masses and p(1,:) and p(2,:) are the respective phase difference.

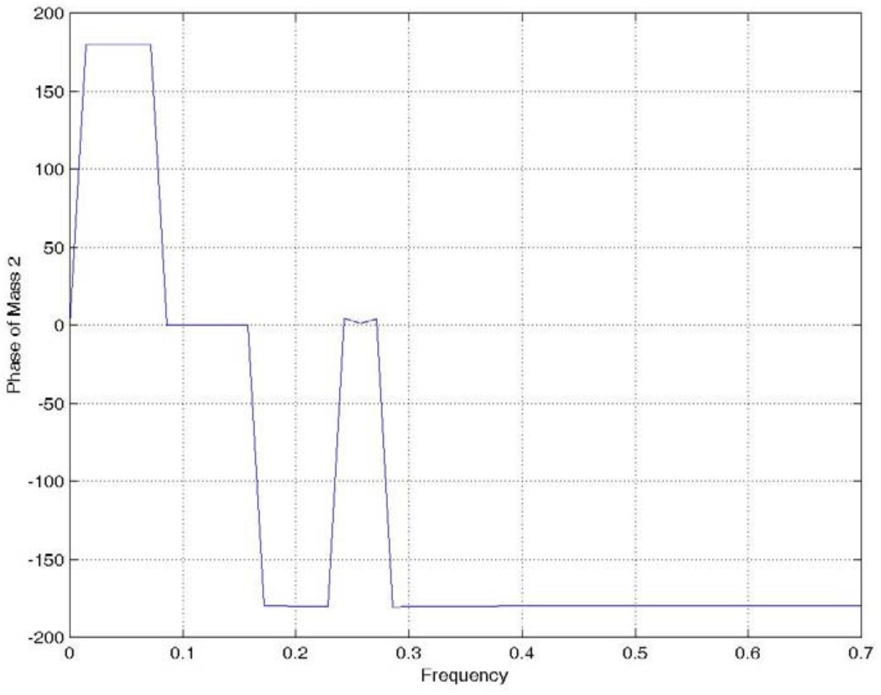
The plots are attached .


Frequency response - Amplitude plot
Amplitude of Mass 1
Frequency
0
5
10
15
20
25
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


Frequency response - Amplitude plot
Amplitude of Mass 2
Frequency
0
5
10
15
20
25
30
35
40
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7



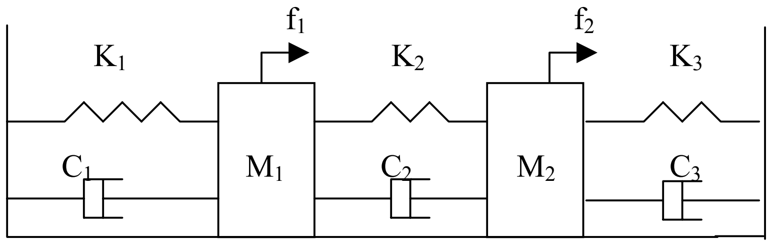
Phase of Mass 1
Frequency
200
150
100
50
0
-50
-100
-150
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7



Phase of Mass 2
Frequency
200
150
100
50
0
-50
-100
-150
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

Assignment

Consider the system shown below.



Derive the governing differential equation of motion and solve for the eigen values and the eigen vectors.

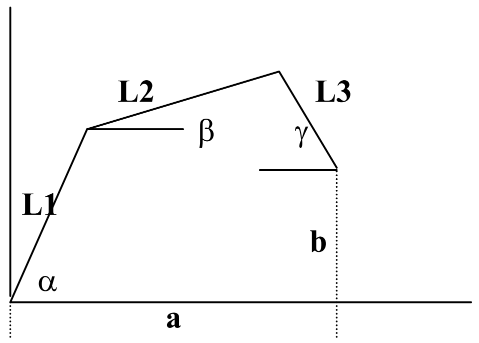
Plot the amplitude and the phase of the masses m_1 and m_2 with respect to the frequency for the system shown above. Choose m_1 = 1Kg; m_2 = 2Kg; $k_1 = k_2 = k_3$ = 1N/m; $c_1 = c_2 = c_3$ = 0.001 N sec/m. The amplitude of the excitation force vector is 2 N.

What do you infer from the plots?

10. A Three Bar Linkage Problem

Example

Consider the three-bar-linkage mechanism shown below. For a constant rotation rate ω of link L1, determine and plot the angular velocities and accelerations of links L2 and L3 for one cycle of rotation of L1. Choose L1, L2 and L3 as 0.35m, 1m and 1m respectively. Also choose 'a' and 'b' as 0.6m and 0.4m respectively. The angular velocity, ω of link L1 is chosen to be 3 rad/sec.



Solution

From the above figure, the geometric relations can be derived as

$$L1\cos\alpha + L2\cos\beta + L3\cos\gamma = a$$
$$L1\sin\alpha + L2\sin\beta - L3\sin\gamma = b$$

First we have to solve for 'β' and 'γ' for values of 'α' ranging from 0 to 2π given the values of 'a' and 'b'. The value of alpha should be given in radians.

The MATLAB code for solving the above equation is given below.

```
p='cos(b)+cos(c)=0.6-0.35*cos(a)';
q='sin(b)-sin(c)=0.4-0.35*sin(a)';
[b,c]=solve(p,q)
i=1;
for a=0:0.1:6.3
   beta(:,i)=subs(b)
   gamma(:,i)=subs(c)
   i=i+1;
end
alpha=0:0.1:6.3
figure(1)
plot(alpha,beta(1,:));grid
xlabel('alpha')
ylabel('beta')
```

```
title('beta VS alpha')

figure(2)
plot(alpha,gamma(1,:));grid
xlabel('alpha')
ylabel('beta')
title('gamma VS alpha')
```

To know, how to use the 'figure command', refer to the "Introduction to MATLAB" chapter. The positive values of both 'beat' and 'gamma' have to be plotted. This is the reason why only the first row is used to plot the results.

To find out the angular velocities, differentiate the above two equations (geometric relations) with respect to time and put them into matrix form. We get the following equation.

$$\begin{bmatrix} -\sin\beta & -\sin\gamma \\ \cos\beta & -\cos\gamma \end{bmatrix} \begin{Bmatrix} L2\dot{\beta} \\ L3\dot{\gamma} \end{Bmatrix} = L1\omega \begin{Bmatrix} \sin\alpha \\ -\cos\alpha \end{Bmatrix}$$

The MATLAB code to solve this equation to get the angular velocities and then plot them with respect to 'α' is given below.

```
L1 = 0.35;
L2 = 1;
L3 = 1;
omega = 3;

for i=1:size(alpha,2)
  A=[-L2*sin(beta(1,i))  -L3*sin(gamma(1,i));  L2*cos(beta(1,i))
  -L3*cos(gamma(1,i))];
  B=[omega*L1*sin(alpha(i));-omega*L1*cos(alpha(i))];
  C=inv(A)*B;
  betadot(1,i)= C(1);
  gammadot(1,i)= C(2);
end

figure(3)
plot(alpha,betadot(1,:));grid
grid
xlabel('alpha')
ylabel('beta dot')
title('betadot VS aplha')

figure(4)
plot(alpha,gammadot(1,:));grid
grid
```

```
xlabel('alpha')
ylabel('gamma dot')
title('gammadot VS aplha')
```

In the above code 'A' contains the coefficient matrix and 'B' contains the RHS matrix.

To calculate the angular accelerations, the geometric relations given by the second equation set must be differentiated to obtain the following relation,

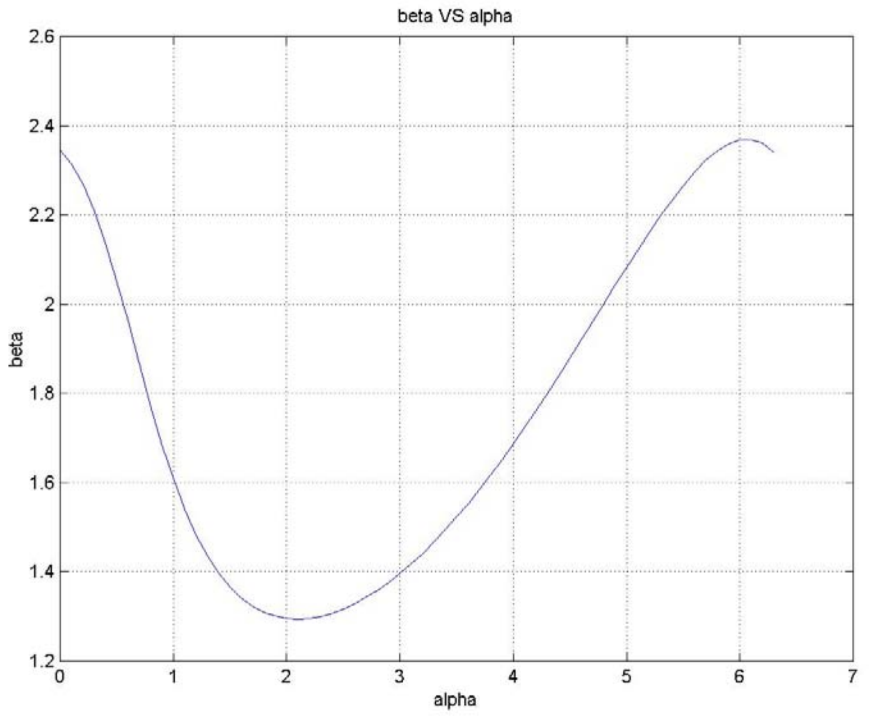
$$\begin{bmatrix} -\sin\beta & -\sin\gamma \\ \cos\beta & -\cos\gamma \end{bmatrix} \begin{Bmatrix} L2\ddot{\beta} \\ L3\ddot{\gamma} \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix}$$

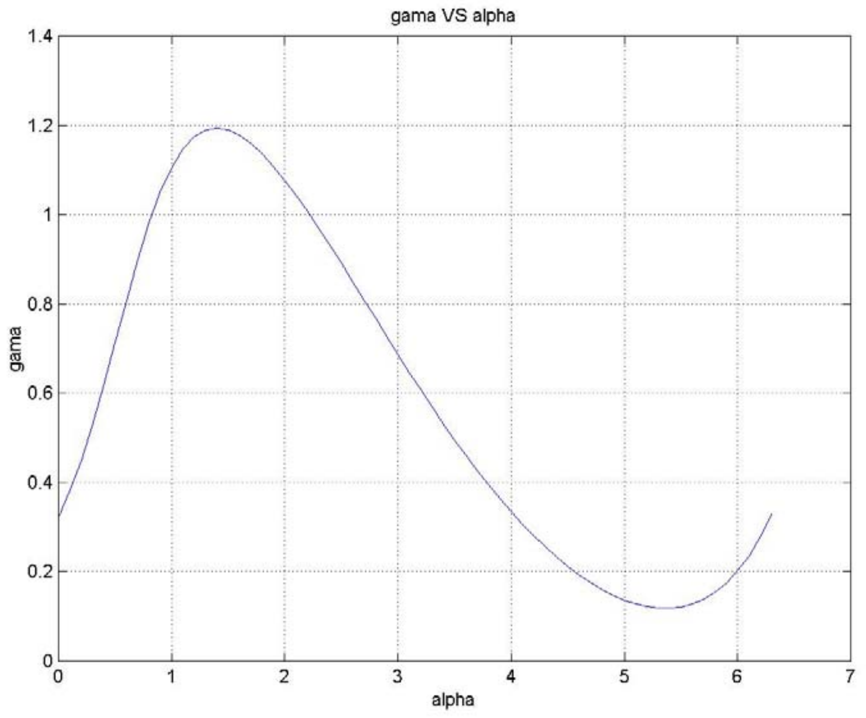
$$\begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = \begin{Bmatrix} L1\omega^2\cos\alpha + L2\dot{\beta}^2\cos\beta + L3\dot{\gamma}^2\cos\gamma \\ L1\omega^2\sin\alpha + L2\dot{\beta}^2\sin\beta - L3\dot{\gamma}^2\sin\gamma \end{Bmatrix}$$

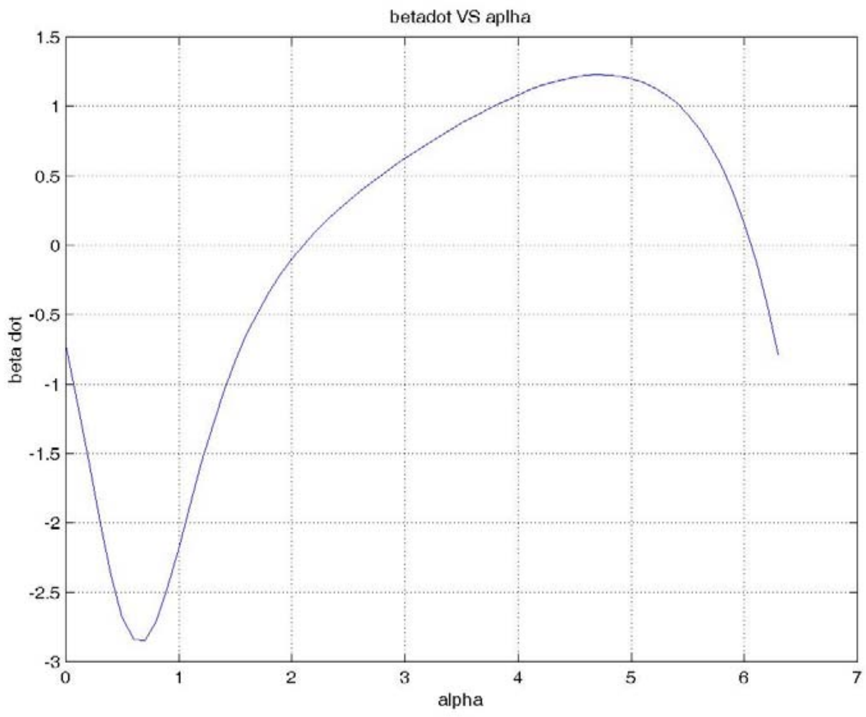
The MATLAB code for solving this equation is given below.

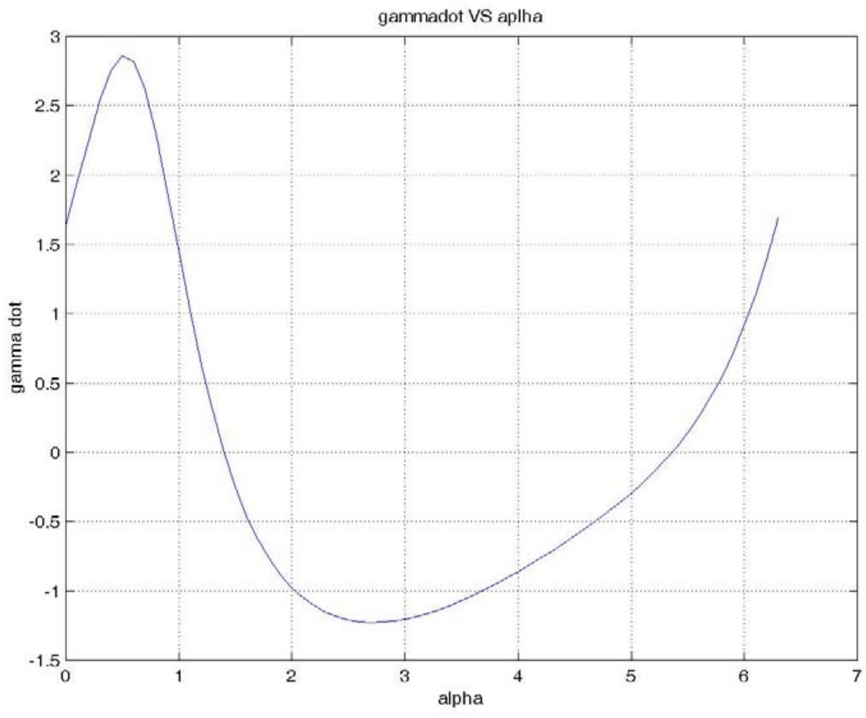
```
for i=1:size(alpha,2)
A1=[-L2*sin(beta(1,i)) -L3*sin(gamma(1,i)); L2*cos(beta(1,i)) -
L3*cos(gamma(1,i))];
B1=[omega*omega*L1*cos(alpha(i))+L2*(betadot(1,i)^2)*cos(beta(1,
i))+L3*(gammadot(1,i)^2)*cos(gamma(1,i));...

omega*omega*L1*sin(alpha(i))+(betadot(1,i)^2)*L2*sin(beta(1,i))-
(gammadot(1,i)^2)*L3*sin(gamma(1,i))];
C2=inv(A1)*B1;
beta_accl(1,i)=C2(1);
gamma_accl(1,i)=C2(2);
end

figure(5);plot(alpha,beta_accl(1,:));grid
xlabel('alpha')
ylabel('beta acceleration')
title('beta accl VS alpha')

figure(6);plot(alpha,gamma_accl(1,:));grid
xlabel('alpha')
ylabel('gamma_accl')
title('gamma_accl VS aplha')
```

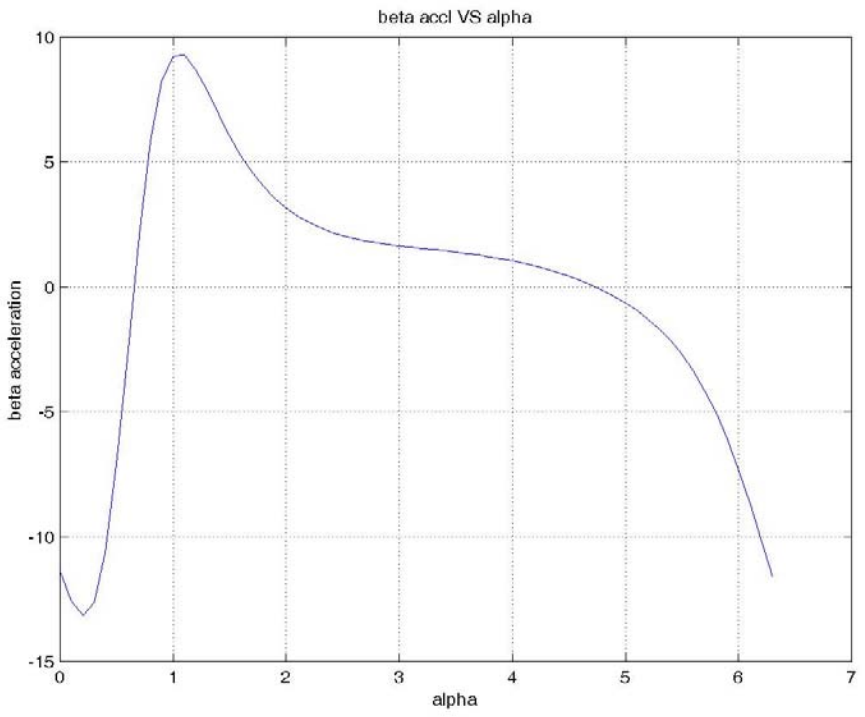
In the above code 'A1' contains the coefficient matrix and 'B1' contains the RHS matrix.

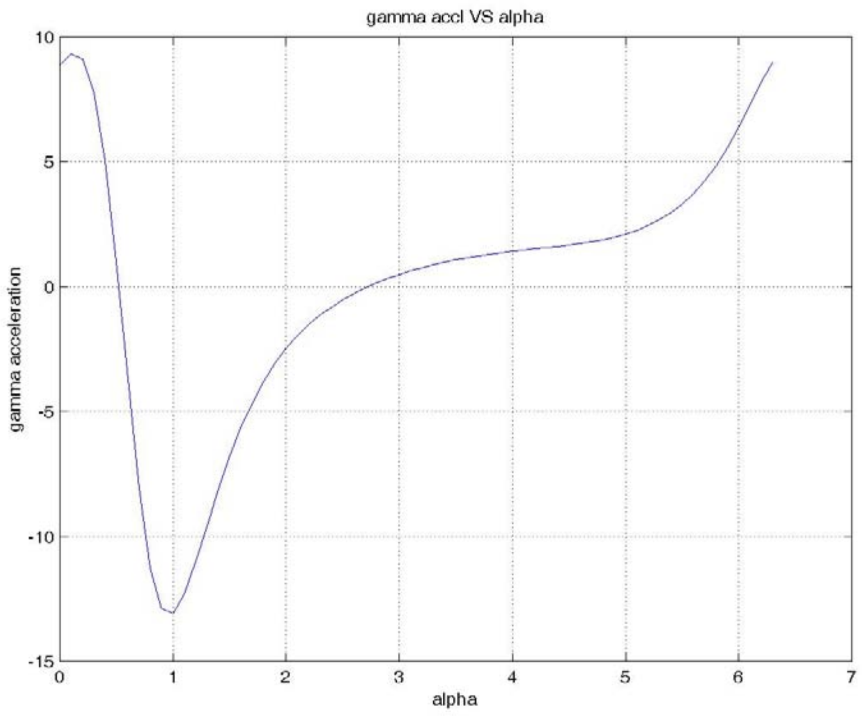

beta VS alpha
beta
alpha
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
0
1
2
3
4
5
6
7


gama VS alpha
gama
alpha
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7


betadot VS aplha
beta dot
alpha
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
0
1
2
3
4
5
6
7


gammadot VS aplha
gamma dot
alpha
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6
7


beta accl VS alpha
beta acceleration
alpha
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7


gamma accl VS alpha
gamma acceleration
alpha
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7

Assignment

For a constant rotation rate ω = 4 rad/sec of link L1, determine the angular velocities and accelerations of links L2 and L3 for one cycle of rotation of L1 for the system shown in the above example. Choose L1, L2 and L3 as 0.5m, 1.2m and 1m respectively. Also choose ‘a’ and ‘b’ as 0.7m and 0.45m respectively.

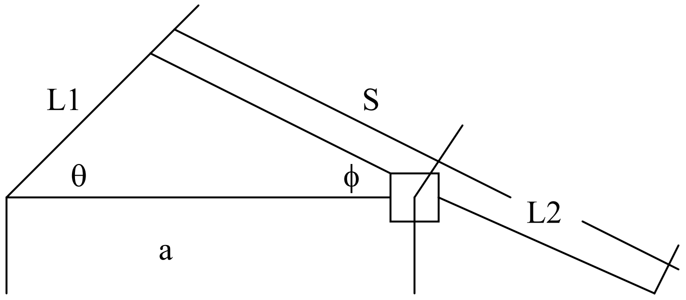
Also plot the angular velocities and the accelerations with respect to ‘alpha’.

What do you infer from the obtained?

11. Slider-Crank Mechanism

Example

Consider the slider crank mechanism shown below. For constant angular velocity, ‘ω’ equal to 3 rad/sec of link L1, determine ϕ and S and their first and second derivatives for one cycle of θ. Choose L1 as 0.5m and ‘a’ as 1m.



Solution

From the figure, the following geometric relation can be derived,

$$S\cos\phi = a - L1\cos\theta$$
$$S\sin\phi = L1\sin\theta.$$

These equations are nonlinear but can be readily solved for ϕ and S in terms of θ.

The MATLAB code for solving the above equation is given below. The values of S and ϕ are plotted with respect to θ.

```
p='s*cos(b)=1-0.5*cos(a)'
q='s*sin(b)=0.5*sin(a)'
[b,s]=solve(p,q)

i=1;
for a=0:0.1:6.3
   phi(:,i)=subs(b)
   S(:,i)=subs(s)
   i=i+1;
end

theta=0:0.1:6.3
figure(1)
plot(theta,S(1,:))
```

```
grid
xlabel('Theta')
ylabel('S')
title('S Vs Theta')

figure(2)
plot(theta,phi(1,:))
grid
xlabel('Theta')
ylabel('Phi')
title('Phi Vs Theta')
```

Here notice that, a new command 'figure' is used to plot different variables. Refer to the "Introduction to MATLAB" chapter to know more about the command and how to use it.

To find the angular velocity, $\dot{\phi}$ and the linear velocity, $\dot{S}$ differentiate the above equations once with respect to time. When written in matrix format the result is shown below,

$$\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{Bmatrix} \dot{S} \\ S\dot{\phi} \end{Bmatrix} = L1\omega \begin{Bmatrix} \sin\theta \\ \cos\theta \end{Bmatrix}$$

The MATLAB code is as shown below.

```
L1 = 0.5;
omega = 3;
for i=1:size(theta,2)
   A=[cos(phi(1,i)) -S(1,i)*sin(phi(1,i));...
         sin(phi(1,i)) S(1,i)*cos(phi(1,i))];
   B=[L1*omega*sin(theta(i));L1*omega*cos(theta(i))];
   C=inv(A)*B;
   linvel(1,i) = C(1);
   angvel(1,i) = C(2)
end
figure(3)
plot(theta,linvel(1,:));grid
xlabel('Theta');
ylabel('linvel');
title('Linear velocity vs Theta');

figure(4)
plot(theta,angvel(1,:));grid
xlabel('Theta')
ylabel('Angvel')
title('Angular velocity Vs Theta')
```

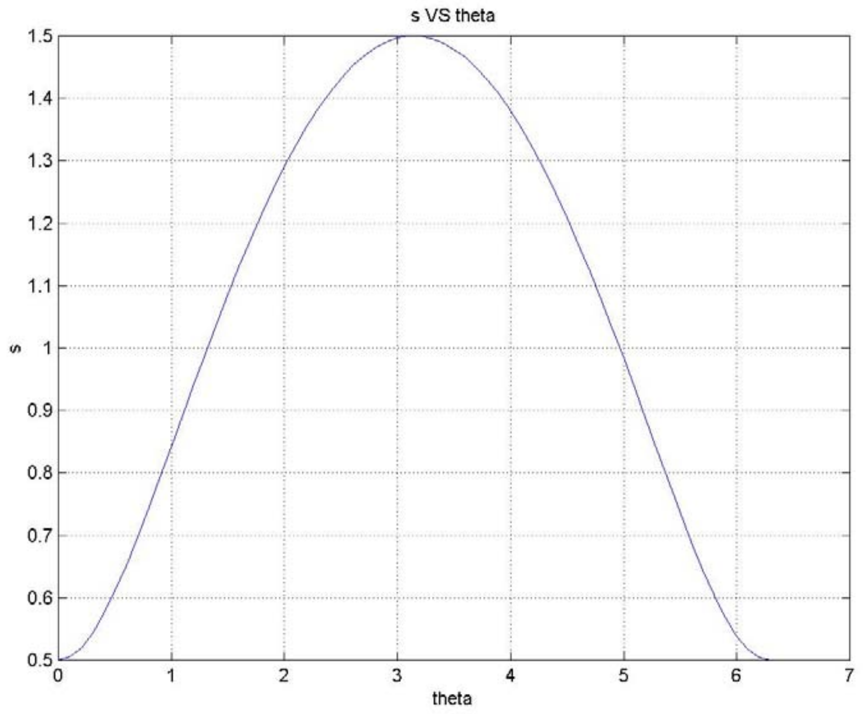
To calculate the angular acceleration, $\ddot{\phi}$ and linear acceleration, $\ddot{S}$, differentiate the last equation with respect to time. Representing the result in matrix format gives,

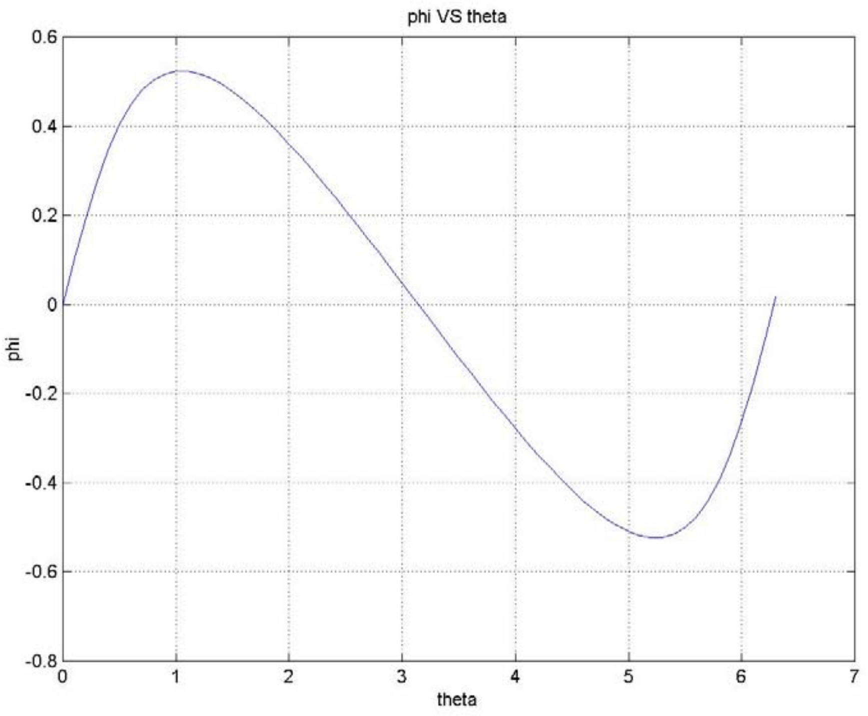
$$\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{Bmatrix} \ddot{S} \\ S\ddot{\phi} \end{Bmatrix} = \begin{Bmatrix} L1\cos\theta\,\dot{\theta}^2 + 2\dot{S}\dot{\phi}\sin\phi + S\cos\phi\,\dot{\phi}^2 \\ -L1\sin\theta\,\dot{\theta}^2 - 2\dot{S}\dot{\phi}\cos\phi + S\sin\phi\,\dot{\phi}^2 \end{Bmatrix}$$

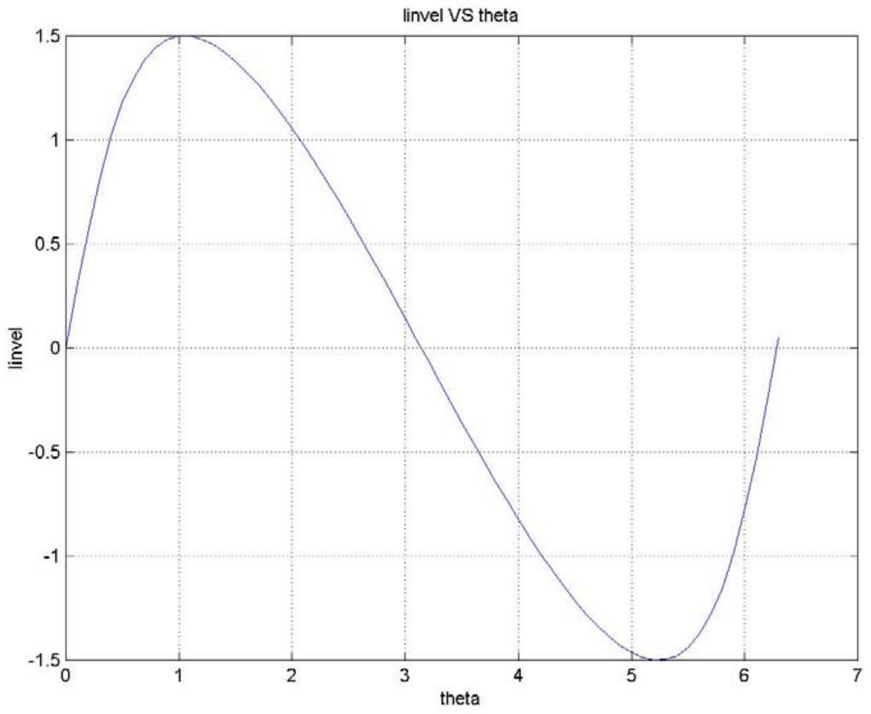
The MATLAB code for solving the above equation is given below.

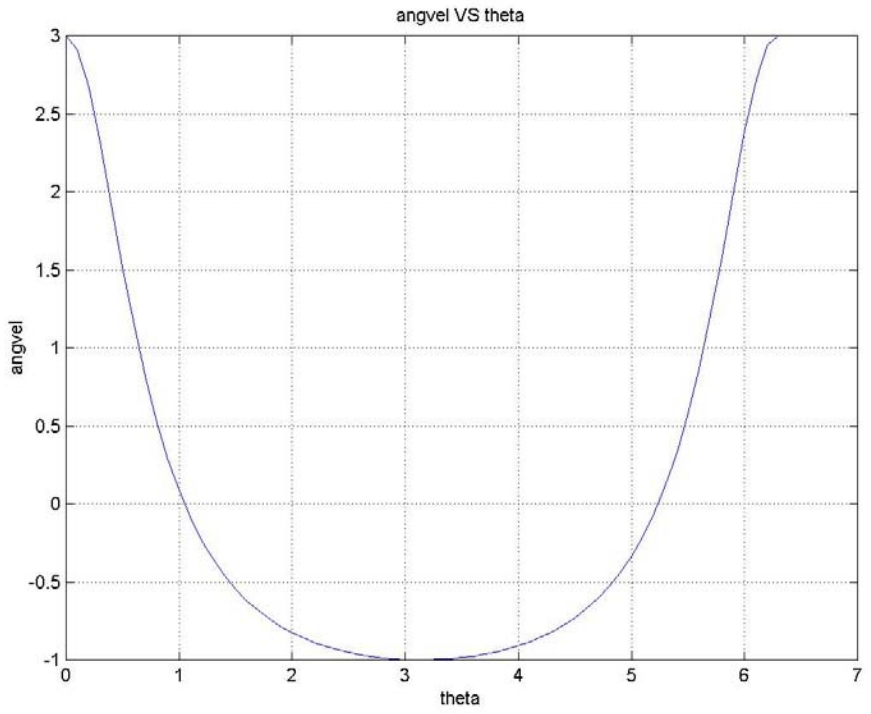
```
L1 = 0.5;
omega = 3;

for i=1:size(theta,2)
   A1=[cos(phi(1,i)) -S(1,i)*sin(phi(1,i));...
         sin(phi(1,i)) S(1,i)*cos(phi(1,i))];

  B1=[L1*(omega^2)*cos(theta(i))+2*linvel(1,i)*angvel(1,i)*sin(p
  hi(1,i))+S(1,i)*(angvel(1,i)^2)*cos(phi(1,i));...
  -L1*(omega^2)*sin(theta(i))-
  2*linvel(1,i)*angvel(1,i)*cos(phi(1,i))+S(1,i)*(angvel(1,i)^2)
  *sin(phi(1,i))];

   C2=inv(A1)*B1;
   Linaccl(1,i) = C2(1);
   Angaccl(1,i) = C2(2);
end

figure(5)
plot(theta,linaccl(1,:));grid
xlabel('Theta');
ylabel('linaccl');
title('Linear acceleration vs Theta');

figure(6)
plot(theta,angaccl(1,:));grid
xlabel('Theta')
ylabel('angaccl')
title('Angular acceleration Vs Theta')
```

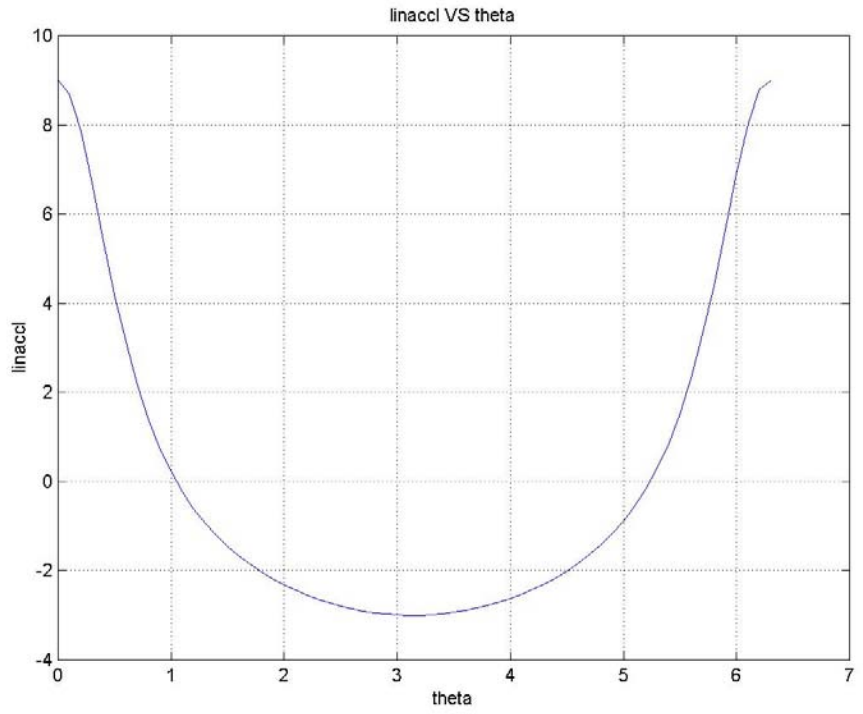
The plots are attached below.


s VS theta
s
theta
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
0.5
0
1
2
3
4
5
6
7

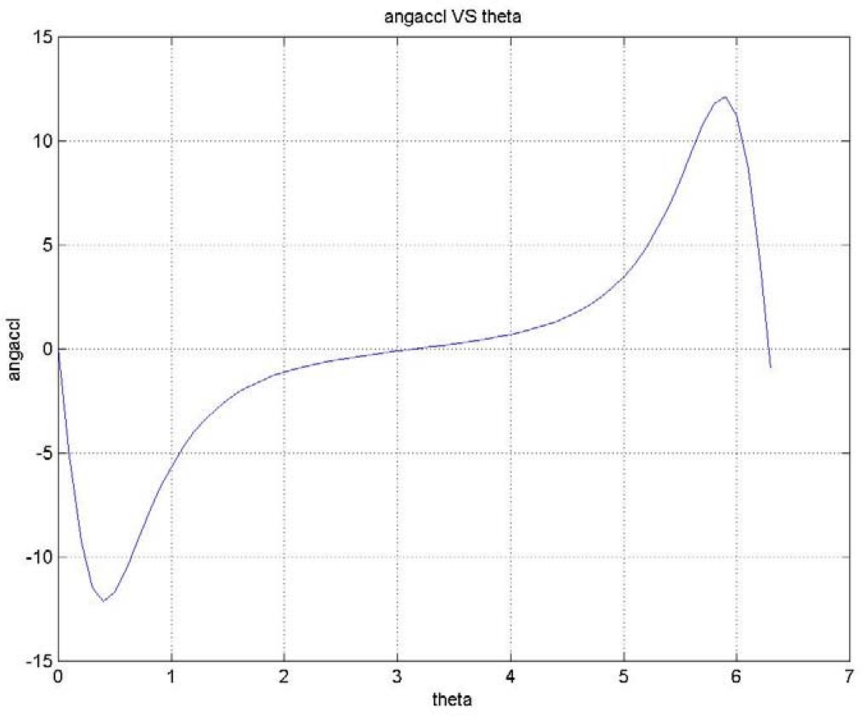

phi VS theta
phi
theta
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1
2
3
4
5
6
7


linvel VS theta
linvel
theta
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6
7


angvel VS theta
angvel
theta
3
2.5
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7



linaccl VS theta
linaccl
theta
10
8
6
4
2
0
-2
-4
0
1
2
3
4
5
6
7



angaccl VS theta
angaccl
theta
15
10
5
0
-5
-10
-15
0
1
2
3
4
5
6
7

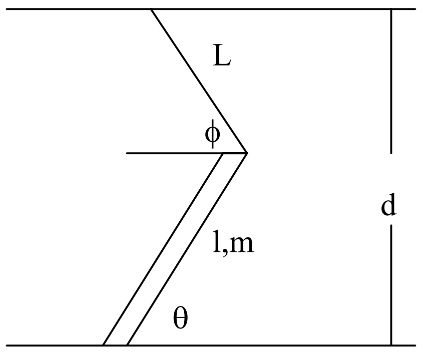
Assignment

Consider the Geneva-wheel mechanism shown in figure 5/194 in Merriam, page 412. Derive the governing differential equations of motion and plot the angular velocity ω_2.

12. A Bar supported by a wire and a horizontal plane

Example

Consider the system shown below. Derive the governing model of motion and plot the values of ‘φ’ and ‘θ’ with respect to time and also their first derivatives with respect to time. Choose ‘l’, ‘L’, ‘m’ and ‘d’ as 1m, 1m, 5 Kg and 1.1m respectively. The initial value of ‘θ’ is chosen to be 0.5233 radians, whereas the initial value of ‘φ’ is calculated in the main code of MATLAB using the constraint equation.`



Solution

The model for the above system is

$$\begin{bmatrix} \frac{ml^2}{12} & 0 & \frac{-l}{2}\sin(\theta+\phi) & \frac{l}{2}\cos\theta \\ \frac{ml}{2}\cos\theta & 0 & -\sin\phi & -1 \\ \frac{ml}{2}\sin\theta & -mL\sin\phi & \cos\phi & 0 \\ l\cos\theta & L\cos\phi & 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\theta} \\ \ddot{\phi} \\ T_c \\ N \end{Bmatrix} = \begin{Bmatrix} 0 \\ -w+\frac{ml}{2}\sin\theta\,\dot{\theta}^2 \\ mL\cos\phi\,\dot{\phi}^2-\frac{ml}{2}\cos\theta\,\dot{\theta}^2 \\ L\sin\phi\,\dot{\phi}^2+l\sin\theta\,\dot{\theta}^2 \end{Bmatrix}$$

First we solve for $\dot{\phi}$ and ϕ. Then we substitute them into the following constraint equations to obtain the values of $\dot{\theta}$ and θ. The constraint equations are the geometrical relations, which can be derived from the above figure to be

$$L\cos\phi\,\dot{\phi}+l\cos\theta\,\dot{\theta}=0$$
$$d=L\sin\phi+l\sin\theta$$

The above model involves second derivatives of the variables and must be converted to an equation involving first derivatives so that MATLAB can be used to solve the differential

equation. The method employed is similar to the one followed to solve the problem in which a bar was connected to a cord.
So with

$\theta = y(1);$

$\dot{\theta} = y(2);$

$\phi = y(3);$

$\dot{\phi} = y(4);$

The model becomes

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{ml^2}{12} & 0 & 0 & \frac{-l}{2}\sin(\theta+\phi) & \frac{l}{2}\cos\theta \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{ml}{2}\cos\theta & 0 & 0 & -\sin\phi & -1 \\ 0 & \frac{ml}{2}\sin\theta & 0 & -mL\sin\phi & \cos\phi & 0 \\ 0 & l\cos\theta & 0 & L\cos\phi & 0 & 0 \end{bmatrix} \begin{Bmatrix} \dot{y}(1) \\ \dot{y}(2) \\ \dot{y}(3) \\ \dot{y}(4) \\ T_c \\ N \end{Bmatrix} = \begin{Bmatrix} y(2) \\ 0 \\ y(4) \\ -w + \frac{ml}{2}\sin\theta\,\dot{\theta}^2 \\ mL\cos\phi\,\dot{\phi}^2 - \frac{ml}{2}\cos\theta\,\dot{\theta}^2 \\ L\sin\phi\,\dot{\phi}^2 + l\sin\theta\,\dot{\theta}^2 \end{Bmatrix}$$

As mentioned earlier, since the coefficient matrix is dependent on the states or the variables, a separate M-file has to be written which incorporates a switch/case programming, and a separate M-file for the mass matrix must also be written.

MATLAB Code

The code with the switch/case programming is given below.

```
% In order to solve this type of problem, the odefile needs % to
have switch/case programming with a flag case of            %
'mass'. when the flag is set to default the function     % FF1.m
is called. Later in the time step, the function      % with the
flag of 'mass' is called. As you can see in the % function
indmot_ode1, when the flag is 'mass', the        % function MM1.m
is called.
function varargout=indmot_ode1(t,y,flag)

switch flag
case ''                        %no input flag
   varargout{1}=FF1(t,y);
case 'mass'                     %flag of mass calls MM1.m
   varargout{1}=MM1(t,y);
otherwise
   error(['unknown flag ''' flag '''.']);
```

```
end
```

To store the right hand side vector of the original model, a separate function file must be written as shown below. Note that the name of the function is ‘FF1’, so this file must be saved as ‘FF1.m’.

```
%the following function contains the right hand side of the
%differential equation of the form
%M(t,y)*y'=F(t,y)
%i.e. it contains F(t,y).it is also stored in a separate %file
named, FF1.m.

function yp=FF1(t,y)
l=1;
L=1;
m=5;
g=9.81;
w=m*g;
yp=zeros(6,1);
yp(1)=y(2);
yp(2)=0;
yp(3)=y(4);
yp(4)=-w+(m*l/2)*sin(y(1))*(y(2)^2);
yp(5)=m*L*cos(y(3))*(y(4)^2)-(m*l/2)*cos(y(1))*(y(2)^2);
yp(6)=L*sin(y(3))*(y(4)^2)+l*sin(y(1))*(y(2)^2);
```

Similarly, to store the coefficient matrix, a separate function file is written which is stored as ‘MM1.m’.

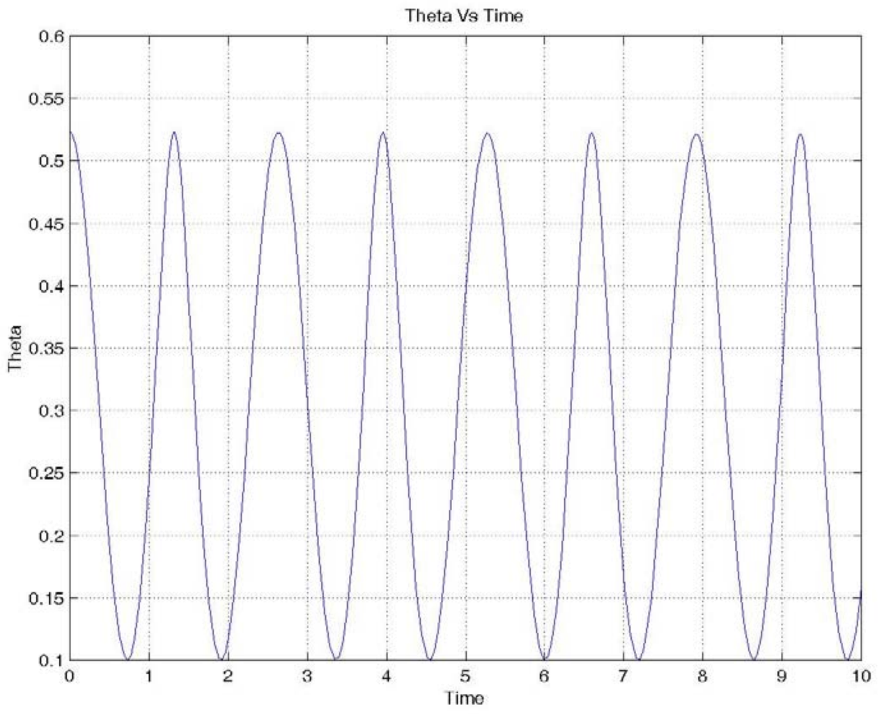
```
% the following function contains the mass matrix.
%it is separately stored in a file named, MM1.m

function n = MM1(t,y)
l=1;
L=1;
m=5;
g=9.81;
w=m*g;
n1=[1 0 0 0 0 0];
n2=[0 (m*l^2)/12 0 0 (-l/2)*sin(y(1)+y(3)) (l/2)*cos(y(1))];
n3=[0 0 1 0 0 0];
n4=[0 (m*l/2)*cos(y(1)) 0 0 -sin(y(3)) -1];
n5=[0 (m*l/2)*sin(y(1)) 0 -m*L*sin(y(3)) cos(y(3)) 0];
n6=[0 l*cos(y(1)) 0 L*cos(y(3)) 0 0];
n=[n1;n2;n3;n4;n5;n6];
```

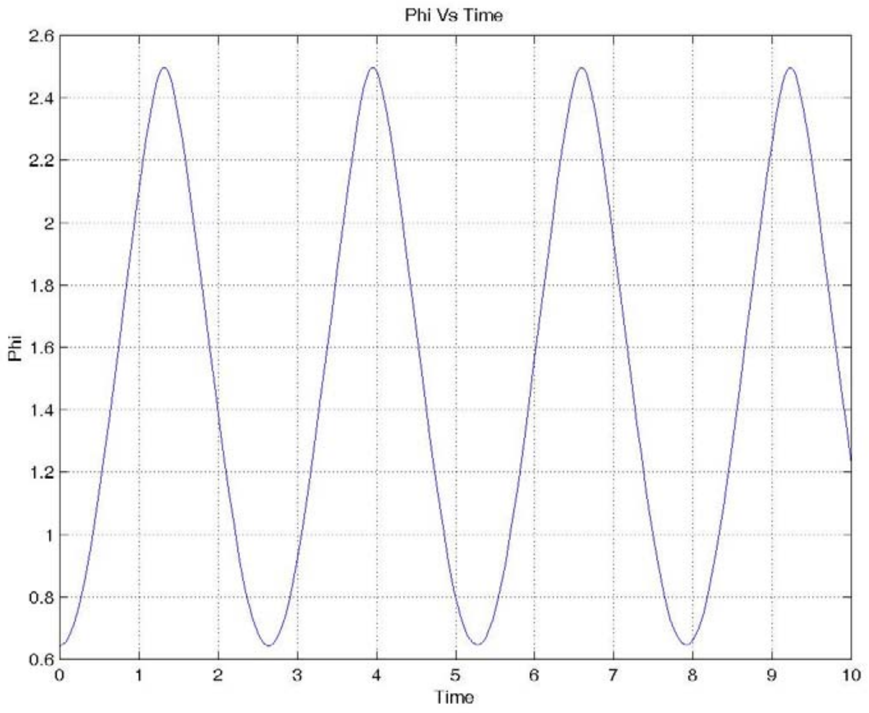
To plot the response, the main file should call the function ‘indmot_ode1.m’, which has the switch/case programming which in turn calls the corresponding functions depending on the value of the flag. For the main file to recognize the coefficient matrix, the MATLAB command ODESET is used to set the mass to ‘M (t, y)’.

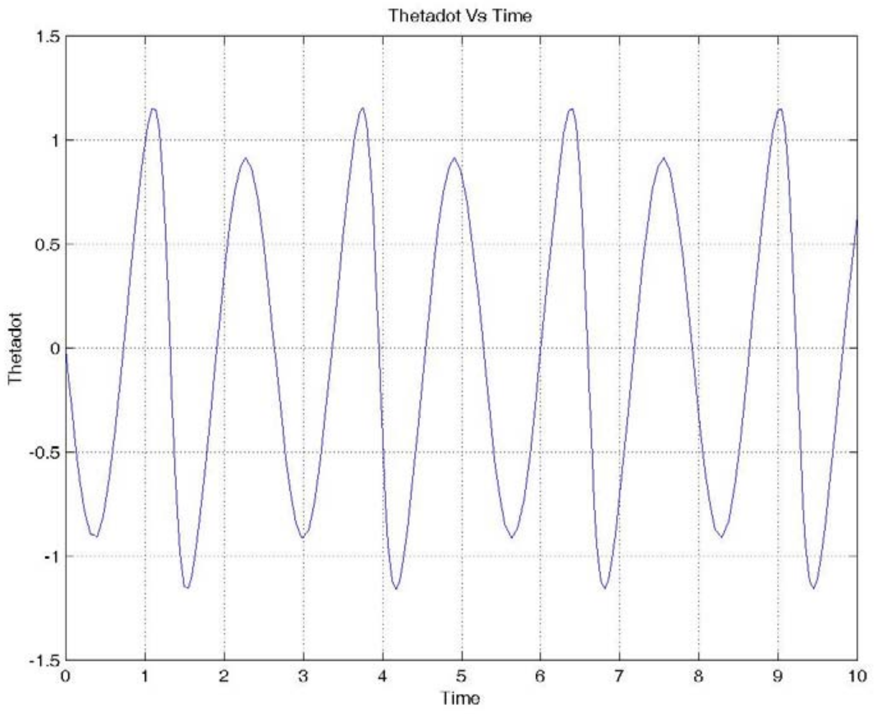
```
l=1;
L=1;
m=5;
g=9.81;
w=m*g;
d=1.1;
tspan=[0 10];
options=odeset('mass','M(t,y)');

% to calculate the initial conditions.
% Theta = 0.5233. For this value of theta
% using the constraint equation, the value
% phi has to be calculated.
theta_init = 0.5233;
phi_init = asin((d-l*sin(theta_init))/L);

y0=[theta_init;0;phi_init;0;0;0];
[t,y]=ode23('indmot_ode1',tspan,y0,options);
figure(1)
plot(t,y(:,3));grid
xlabel('Time')
ylabel('Phi')
title('Phi Vs Time')

figure(2)
plot(t,y(:,4));grid
xlabel('Time')
ylabel('Phidot')
title('Phidot Vs Time')


phi=y(:,3);
phidot=y(:,4);

% solving the constraint equation to get the value of phi.

for i = 1:size(phi,1)
   theta(i)=asin((d-L*sin(phi(i)))/l);
end

% solving the constraint equation to get the value of phidot.
for i = 1:size(phi,1)
thetadot(i)=(L*(-phidot(i))*cos(phi(i)))/(l*cos(theta(i)));
end
```

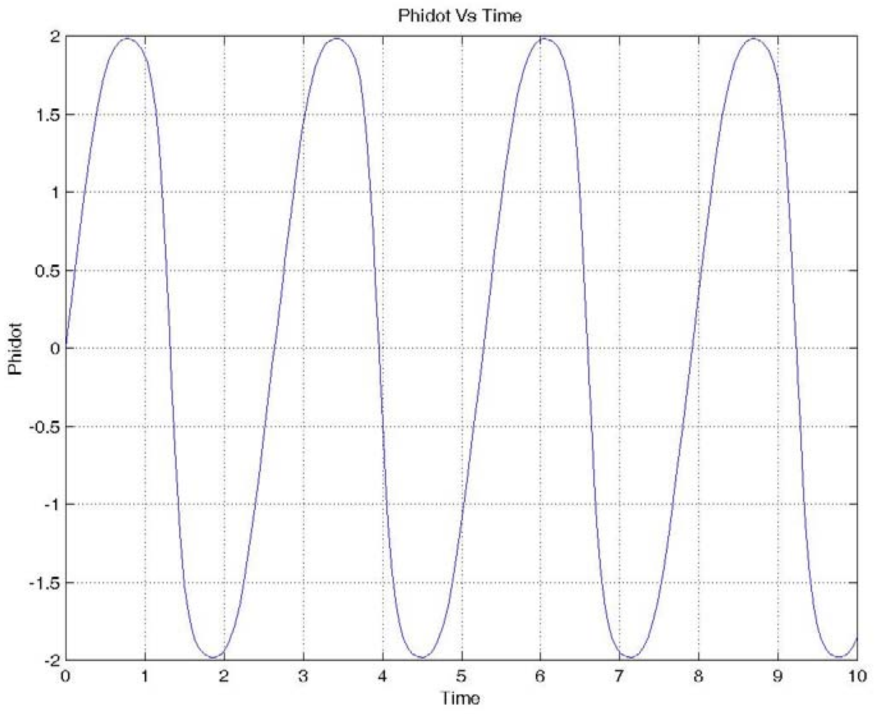
```
t1=t';
figure(3)
plot(t1,theta);grid
xlabel('Time')
ylabel('Theta')
title('Theta Vs Time')

figure(4)
plot(t1,thetadot);grid
xlabel('Time')
ylabel('Thetadot')
title('Thetadot Vs Time')
```

Once the values of ‘θ’ and $\dot{\theta}$ are known, they are substituted in the constraint equation to get the values of ‘ϕ’ and $\dot{\phi}$. To know more about the ‘figure’ command, either refer to the “Introduction to MATLAB” chapter or go through the online help as suggested in the previous example. The plots are attached below.


Theta Vs Time
Theta
Time
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
1
2
3
4
5
6
7
8
9
10


Phi Vs Time
Phi
Time
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0
1
2
3
4
5
6
7
8
9
10

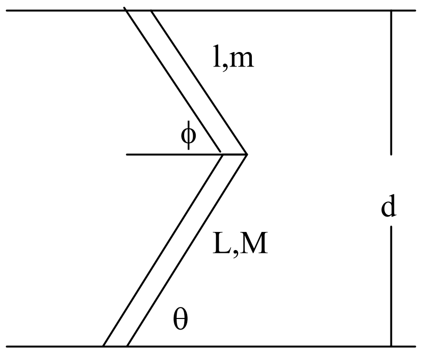

Thetadot Vs Time
Thetadot
Time
1.5
1
0.5
0
-0.5
-1
-1.5
0
1
2
3
4
5
6
7
8
9
10


Phidot Vs Time
Phidot
Time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8
9
10

13. Two bar linkage assembly supported by a pivot joint and a horizontal plane

Example

Consider the system shown below. Derive the governing model and plot the values of 'ϕ' and 'θ' with respect to time and also their first derivatives with respect to time. Choose 'l', 'L', 'm', 'M' and 'd' as 1m, 1.5m, 5 Kg, 5Kg and 1.2m respectively. The initial value of 'ϕ' is chosen to be 0.5233 radians, whereas the initial value of 'θ' is calculated in the main code of MATLAB using the constraint equation.



Solution

The model for the above system is

$$\begin{bmatrix} \frac{ml^2}{12} & 0 & \frac{l}{2}\sin\theta & \frac{-l}{2}\cos\theta & \frac{l}{2}\cos\theta \\ 0 & \frac{ML^2}{3} & -L\sin\phi & -L\cos\phi & 0 \\ \frac{ml}{2}\cos\theta & 0 & 0 & -1 & -1 \\ \frac{ml}{2}\sin\theta & -mL\sin\phi & -1 & 0 & 0 \\ l\cos\theta & L\cos\phi & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\theta} \\ \ddot{\phi} \\ B_x \\ B_y \\ N \end{Bmatrix} = \begin{Bmatrix} 0 \\ W\frac{L}{2}\cos\phi \\ -w+\frac{ml}{2}\sin\theta\,\dot{\theta}^2 \\ mL\cos\phi\,\dot{\phi}^2 - \frac{ml}{2}\cos\theta\,\dot{\theta}^2 \\ L\sin\phi\,\dot{\phi}^2 + l\sin\theta\,\dot{\theta}^2 \end{Bmatrix}$$

First we solve for $\dot{\phi}$ and ϕ. Then we substitute them into the following constraint equations to obtain the values of $\dot{\theta}$ and θ. The constraint equations are the geometrical relations, which can be derived from the above figure to be

$$L\cos\theta\,\dot{\theta} + l\cos\phi\,\dot{\phi} = 0$$
$$d = L\sin\theta + l\sin\phi$$

The above model involves second derivatives of the variables and must be converted to an equation involving first derivatives so that MATLAB can be used to solve the differential equation. The method employed is similar to the one followed to solve the problem in which a bar was connected to a cord.

So with

$\theta = y(1);$

$\dot{\theta} = y(2);$

$\phi = y(3);$

$\dot{\phi} = y(4);$

The above model becomes

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{ml^2}{12} & 0 & 0 & \frac{l}{2}\sin\theta & \frac{-l}{2}\cos\theta & \frac{l}{2}\cos\theta \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{ML^2}{3} & -L\sin\phi & -L\cos\phi & 0 \\ 0 & \frac{ml}{2}\cos\theta & 0 & 0 & 0 & -1 & -1 \\ 0 & \frac{ml}{2}\sin\theta & 0 & -mL\sin\phi & -1 & 0 & 0 \\ 0 & l\cos\theta & 0 & L\cos\phi & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \dot{y}(1) \\ \dot{y}(2) \\ \dot{y}(3) \\ \dot{y}(4) \\ B_x \\ B_y \\ N \end{Bmatrix} = \begin{Bmatrix} y(2) \\ 0 \\ y(4) \\ W\frac{L}{2}\cos\phi \\ -w + \frac{ml}{2}\sin\theta\,\dot{\theta}^2 \\ mL\cos\phi\,\dot{\phi}^2 - \frac{ml}{2}\cos\theta\,\dot{\theta}^2 \\ L\sin\phi\,\dot{\phi}^2 + l\sin\theta\,\dot{\theta}^2 \end{Bmatrix}$$

As mentioned earlier, since the coefficient matrix is dependent on the states or the variables, a separate M-file has to be written which incorporates a switch/case programming, and a separate M-file for the mass matrix must also be written.

MATLAB Code

The code with the switch/case programming is given below.

```
% In order to solve this type of problem, the odefile needs % to
have switch/case programming with a flag case of       % 'mass'.
When the flag is set to default the function      % FF2bar1.m is
called. Later in the time step, the function % with the flag of
'mass'  is  called.  As  you  can  see  in  the  %  function
'indmot_ode2bar1.m', when the flag is 'mass',
% the function MM2bar1.m is called.


function varargout=indmot_ode2bar1(t,y,flag)
```

```
switch flag
case ''                    %no input flag
   varargout{1}=FF2bar1(t,y);
case 'mass'                 %flag of mass calls mass.m
   varargout{1}=MM2bar1(t,y);
otherwise
   error(['unknown flag ''' flag '''.']);
end
```

To store the right hand side vector of the original model, a separate function file must be written as shown below. Note that the name of the function is ‘FF2bar1’, so this file must be saved as ‘FF2bar1.m’.

```
%the following function contains the right hand side of the
%differential equation of the form
%M(t,y)*y'=F(t,y)
%i.e. it contains F(t,y).it is also stored in a separate file
named, FF2bar1.m.

function yp=FF2bar1(t,y)
l=1;
L=1.5;
m=5;
M=5;
g=9.81;
w=m*g;
W=M*g;
yp=zeros(7,1);
yp(1)=y(2);
yp(2)=0;
yp(3)=y(4);
yp(4)=(W*L*cos(y(3)))/2;
yp(5)=-w+(m*(l/2))*sin(y(1))*(y(2)^2);
yp(6)=m*L*cos(y(3))*(y(4)^2)-(m*(l/2))*cos(y(1))*(y(2)^2);
yp(7)=L*sin(y(3))*(y(4)^2)+l*sin(y(1))*(y(2)^2);
```

Similarly, to store the coefficient matrix a separate function file is written which is stored as ‘MM2bar1.m’.

```
% the following function contains the mass matrix.
%it is separately stored in a file named, MM2bar1.m


function n = MM2bar1(t,y)
```

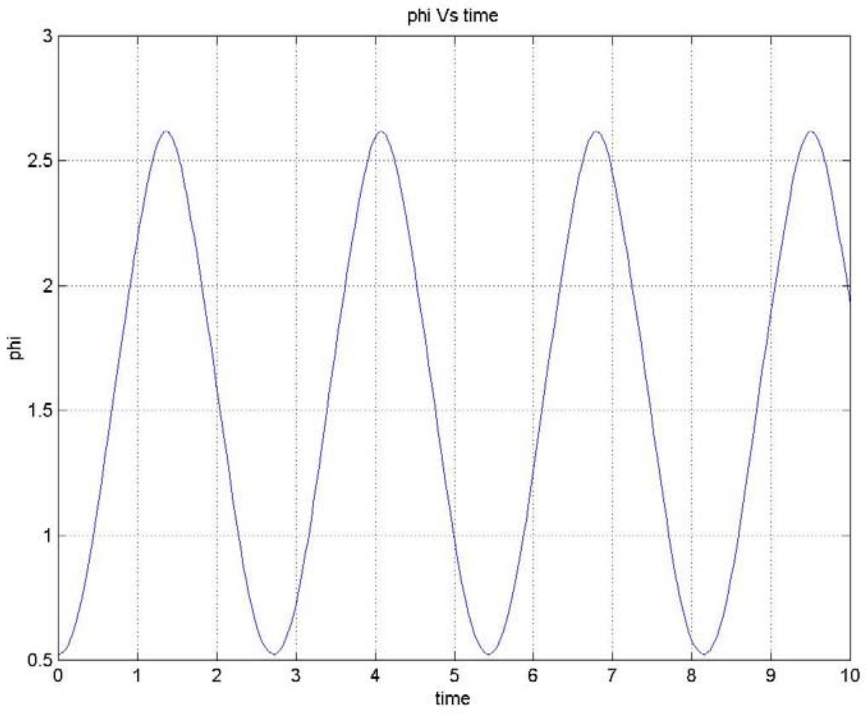
```
l=1;
L=1.5;
m=5;
M=5;
g=9.81;
w=m*g;
n1=[1 0 0 0 0 0 0];
n2=[0 (m*(l^2))/12 0 0 (l/2)*sin(y(1)) -(l/2)*cos(y(1))
(l/2)*cos(y(1))];
n3=[0 0 1 0 0 0 0];
n4=[0 0 0 (M*(L^2))/3 -L*sin(y(3)) -L*cos(y(3)) 0];
n5=[0 (m*(l/2))*cos(y(1)) 0 0 0 -1 -1];
n6=[0 (m*(l/2))*sin(y(1)) 0 -m*L*sin(y(3)) -1 0 0];
n7=[0 l*cos(y(1)) 0 L*cos(y(3)) 0 0 0];
n=[n1;n2;n3;n4;n5;n6;n7];
```

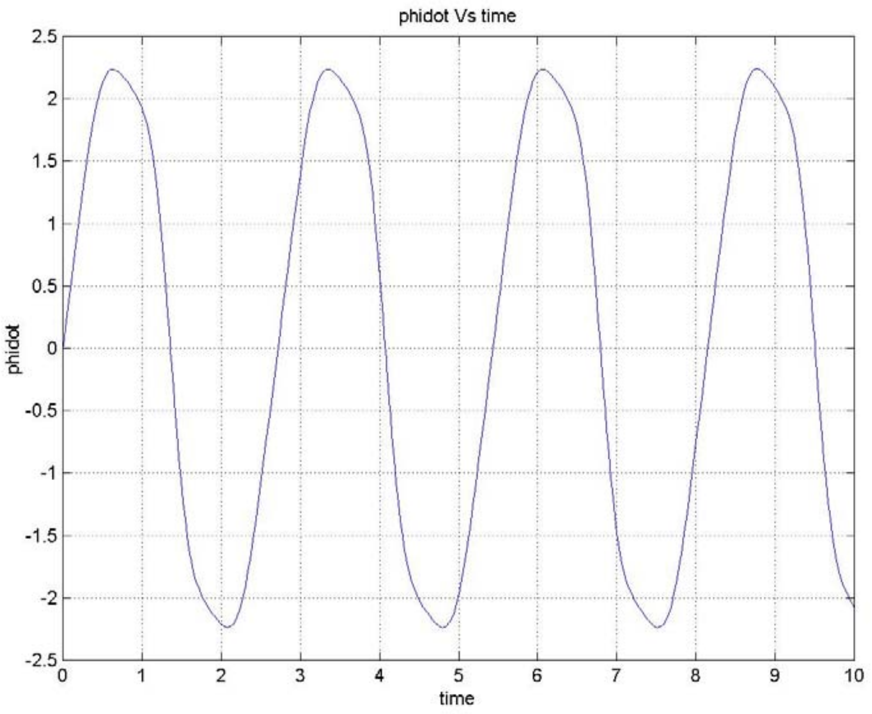
To plot the response, the main file should call the function ‘indmot_ode1.m’, which has the switch/case programming which in turn calls the corresponding functions depending on the value of the flag. For the main file to recognize the coefficient matrix, the MATLAB command ODESET is used to set the mass to ‘M (t, y)’.

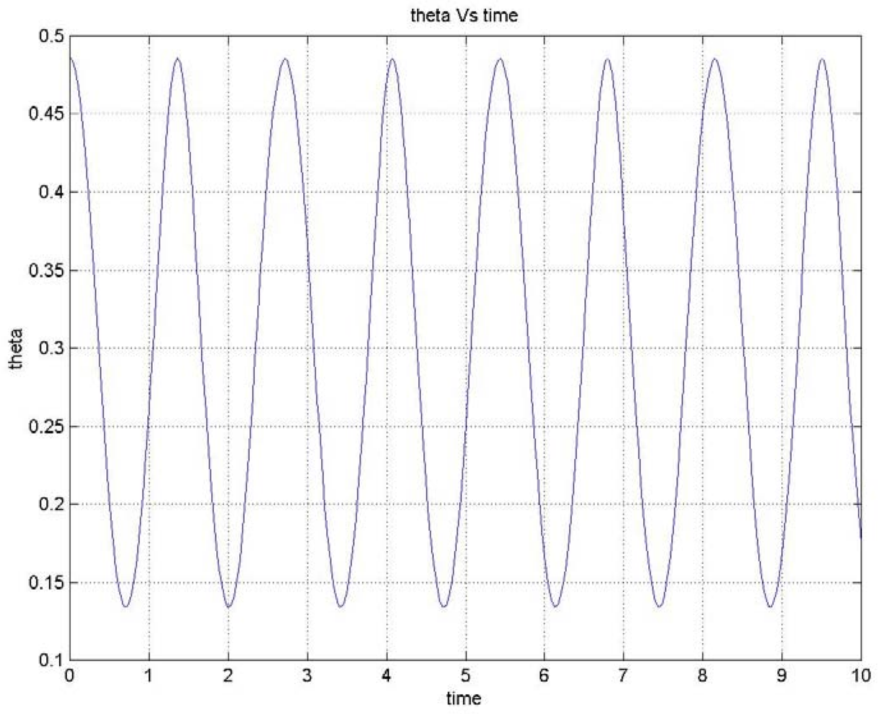
```
l=1;
L=1.5;
m=5;
M=5;
g=9.81;
w=m*g;
W=M*g;
d=1.2;

tspan=[0 10];
options=odeset('mass','M(t,y)');

% to calculate the initial conditions.
% Theta = 0.5233. For this value of theta
% using the constraint equation, the value
% phi has to be calculated.
phi_init = 0.5233;
theta_init = asin((d-l*sin(phi_init))/L);

y0=[theta_init;0;phi_init;0;0;0;0]
[t,y]=ode23tb('indmot_ode2bar1',tspan,y0,options);
figure(1)
plot(t,y(:,3));grid
xlabel('time')
```

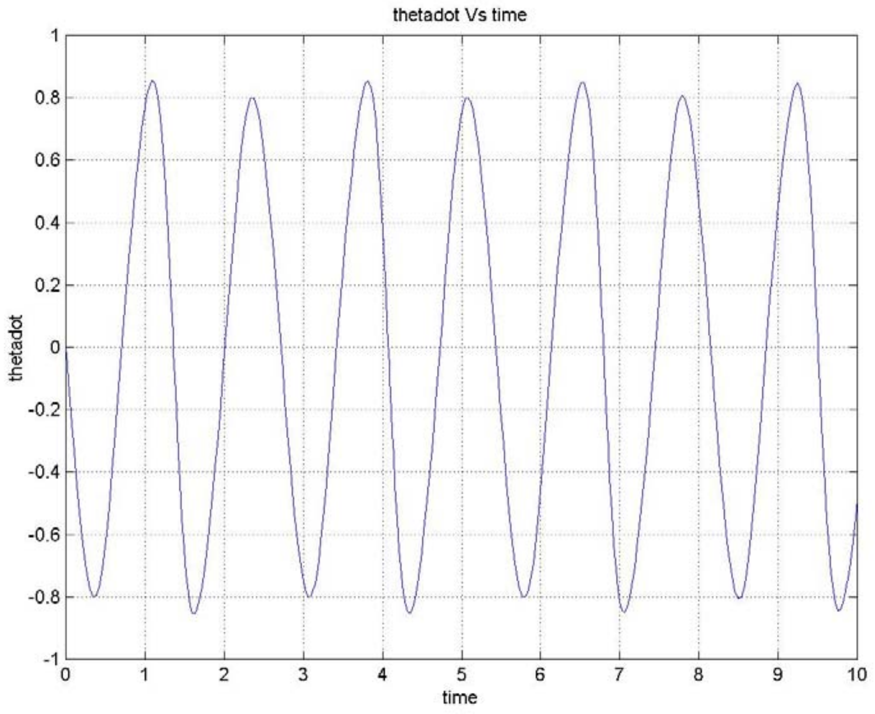
```
ylabel('phi')
title('phi Vs time')

figure(2)
plot(t,y(:,4));grid
xlabel('time')
ylabel('phidot')
title('phidot Vs time')

phi=y(:,3);
phidot=y(:,4);

% solving the constraint equation to get the value of phi.
for i = 1:size(phi,1)
   theta(i)=asin((d-l*sin(phi(i)))/L);
end

% solving the constraint equation to get the value of      %
phidot.
for i = 1:size(phi,1)
   thetadot(i)=(l*(-phidot(i))*cos(phi(i)))/(L*cos(theta(i)));
end

t1=t';
figure(3)
plot(t1,theta);grid
xlabel('time')
ylabel('theta')
title('theta Vs time')

figure(4)
plot(t1,thetadot);grid
xlabel('time')
ylabel('thetadot')
title('thetadot Vs time')
```

Once the values of ‘ϕ’ and $\dot{\phi}$ are known, they are substituted in the constraint equation to get the values of ‘θ’ and $\dot{\theta}$. The plots are attached below.


phi Vs time
phi
time
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
9
10

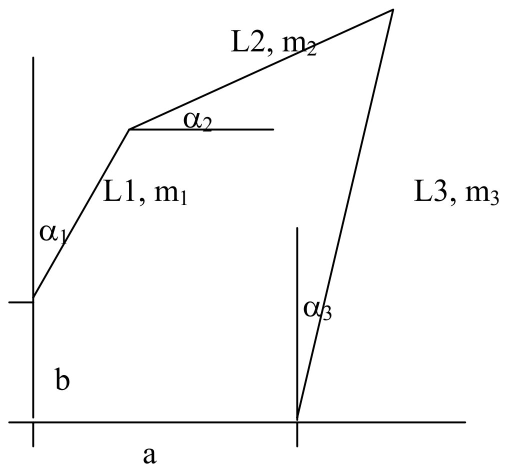

phidot Vs time
phidot
time
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
1
2
3
4
5
6
7
8
9
10


theta Vs time
theta
time
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0
1
2
3
4
5
6
7
8
9
10


thetadot Vs time
thetadot
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9
10

Assignment: Three bar linkage assembly

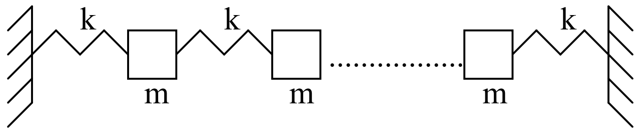
Consider the system shown below. Derive the governing differential equations of motion and plot the values of 'α_1', 'α_2' and 'α_3' with respect to time and also their first derivatives with respect to time. Choose 'L1', 'L2', 'L3', 'm_1', 'm_2', 'm_3', 'a' and 'b' as 0.35m, 1m, 1.2m, 5Kg, 5Kg, 6Kg, 0.6m and 0.4m respectively. The initial values of 'α_1'is chosen to be 0.1744 radians. The initial values of 'α_2' and 'α_3' are calculated from the constraint equations.



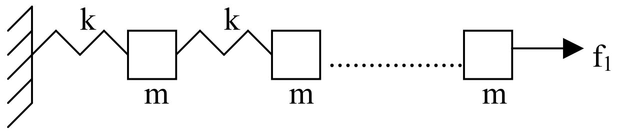
14. Valve Spring Model

Consider the following system, which has 10 blocks of equal weight that are interconnected by springs. The objective of this exercise is to calculate the eigen values and eigen vectors for the system defined for three different boundary conditions. The boundary conditions are as follows.

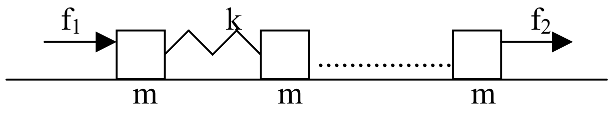
1. Fixed at both ends



2. Fixed at one end and free at the other



3. Free at both ends



In all the above cases, each lumped mass of the model has mass, 0.002 Kg and the stiffness constants of the springs are chosen as 175000 N/m.

Solution

Case 1: Prescribed motion for both ends

Let x_2, x_3,...,x_{11} be the displacement of each of the blocks with respect to a reference frame and x_1 and x_{12} be the displacements of the end supports. The eigen values and eigen vectors are defined for no excitation, i.e, zero motion for x_1 and x_{12}. So this implies that both x_1 and x_{12} are equal to zero. The equation of motion for individual masses are

$$m\ddot{x}_2 + k(x_2 - x_3) + kx_2 = kx_1$$

$$m\ddot{x}_3 + k(x_3 - x_2) + k(x_3 - x_4) = 0$$

.

$$m\ddot{x}_{11} + k(x_{11} - x_{10}) + kx_{11} = kx_{12}$$

If the above equations are written in the matrix format, the system equations of motion reduce to

$$M1\ddot{X}+K1X=F1$$

where

$$M1=\begin{bmatrix} m&0&0&0&0&0&0&0&0&0\\ 0&m&0&0&0&0&0&0&0&0\\ 0&0&m&0&0&0&0&0&0&0\\ 0&0&0&m&0&0&0&0&0&0\\ 0&0&0&0&m&0&0&0&0&0\\ 0&0&0&0&0&m&0&0&0&0\\ 0&0&0&0&0&0&m&0&0&0\\ 0&0&0&0&0&0&0&m&0&0\\ 0&0&0&0&0&0&0&0&m&0\\ 0&0&0&0&0&0&0&0&0&m \end{bmatrix}$$

$$K1=\begin{bmatrix} 2k&-k&0&0&0&0&0&0&0&0\\ -k&2k&-k&0&0&0&0&0&0&0\\ 0&-k&2k&-k&0&0&0&0&0&0\\ 0&0&-k&2k&-k&0&0&0&0&0\\ 0&0&0&-k&2k&-k&0&0&0&0\\ 0&0&0&0&-k&2k&-k&0&0&0\\ 0&0&0&0&0&-k&2k&-k&0&0\\ 0&0&0&0&0&0&-k&2k&-k&0\\ 0&0&0&0&0&0&0&-k&2k&-k\\ 0&0&0&0&0&0&0&0&-k&2k \end{bmatrix}$$

$$F1=\begin{bmatrix} kx_1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ kx_{12} \end{bmatrix}$$

To calculate the eigen values and eigen vectors, the right hand matrix is ignored and only the mass and stiffness matrices are used. Hence the boundary conditions for the eigen vectors are $x_1 = x_{12} = 0$. The ‘eig’ command in MATLAB is used for this purpose. To know more about the command, go through the online help by typing the following at the command prompt.

help eig

The square root of the eigen values gives the natural frequencies of the system.

Case 2: Prescribed motion at one end and prescribed force at the other end.

In this case, since there is support only at one end, the equation of motion for the last block changes. The equation of motion for the 10th mass is

$$m\ddot{x}_{11} + k(x_{11} - x_{10}) = f_1(t)$$

This is very much similar to the equation in case 1, except that since there is a prescribed force at the right hand instead of a prescribed motion, x_{12} does not appear in the equation. Here again we assume that the end support at one end is stationary, which implies that x_1 is equal to zero. The effect of this change in the equation of motion can be seen in the stiffness matrix. Notice that the element formed due to the intersection of the 10th row and 10th column of the stiffness matrix is half of that in the stiffness matrix of the previous case.

The right hand side matrix for this case will be similar to that of the previous case except that x_{12} will be replaced with zero. Here again the right hand matrix is ignored to calculate the eigen values and eigen vectors. The mass matrix, the stiffness matrix and the right hand side matrix for the case wherein one end is fixed and the other is free is given by

$$M2 = M1$$

$$K2 = \begin{bmatrix} 2k & -k & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -k & 2k & -k & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -k & 2k & -k & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -k & 2k & -k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k & 2k & -k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -k & 2k & -k & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k & 2k & -k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -k & 2k & -k & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k & 2k & -k \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k & k \end{bmatrix}$$

$$F2 = \begin{bmatrix} kx_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ f_1 \end{bmatrix}$$

The same procedure is adopted to obtain the eigen values and eigen vectors for the system

Case 3: Applied forces at both ends

In this case, the displacements of each block with respect to a reference frame are chosen slightly different from the previous two cases. Let x_1, x_2,..., x_{10} be the displacements of the 10 blocks with respect to a reference frame. Since there is no support at both ends and a prescribed force is applied at both ends, the equation of motion for the first and the last block changes, which is given as follows.

$$m\ddot{x}_1 + k(x_1 - x_2) = f_1(t)$$

$$m\ddot{x}_{10} + k(x_{10} - x_9) = f_2(t)$$

This is very much similar to the equation in case 1, except that since there are no supports at the ends, the effect of this change in the equation of motion can be seen in the stiffness matrix. Notice that the element formed due to the intersection of the 10th row and 10th column of the stiffness matrix and the element formed due to the intersection of the first row and first column are half of that in the stiffness matrix of case 1.

The right had side matrix for this case will be a zero vector of size 10 x 1. The mass, the stiffness and the right hand side matrices for the case wherein both ends are free are given by

$M3 = M1$

$$K3 = \begin{bmatrix} k & -k & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -k & 2k & -k & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -k & 2k & -k & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -k & 2k & -k & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -k & 2k & -k & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -k & 2k & -k & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k & 2k & -k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -k & 2k & -k & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k & 2k & -k \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -k & k \end{bmatrix}$$

$$F3 = \begin{bmatrix} f_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ f_2 \end{bmatrix}$$

The same procedure is adopted to obtain the eigen values and eigen vectors for the system.

The plots are attached after the MATLAB code.

MATLAB Code

The MATLAB code is used to calculate the eigen values, eigen vectors and also the natural frequencies of the system for different boundary conditions. But only the first three mode shapes are plotted.

```
% The first code is for a system where both the ends are    %
fixed. In the following codes 'M' is the mass matrix, 'K' % is
the stiffness matrix. 'm' is the mass of each block    % in Kgs
and 'k' is the value of the stiffness constant in % N/m. 'v' and
'd' give the eigen vectors and eigen values % respectively.
```

```
m = 0.002;
k = 175000;

M1 = m*eye(10);
K1=k*[2 -1 0 0 0 0 0 0 0 0;
   -1 2 -1 0 0 0 0 0 0 0;
   0 -1 2 -1 0 0 0 0 0 0;
   0 0 -1 2 -1 0 0 0 0 0;
   0 0 0 -1 2 -1 0 0 0 0;
   0 0 0 0 -1 2 -1 0 0 0;
   0 0 0 0 0 -1 2 -1 0 0;
   0 0 0 0 0 0 -1 2 -1 0;
   0 0 0 0 0 0 0 -1 2 -1;
   0 0 0 0 0 0 0 0 -1 2;];


[v1,d1] = eig(K1,M1);

% This code is written to separate the natural frequencies of
the system. w_nat1 contains the natural frequencies.

cnt =1;
for i = 1:10
   for j=1:10
      if(i==j)
         w_nat1(cnt)=sqrt(d1(i,j));
         cnt = cnt +1;
      end
   end
end


% To plot the first three mode shapes, we need to find the %
first three eigen values.
```

% The 'EIGS' command has been used to find the required % eigen values and the corresponding eigen vectors. 'SM' in % the code denotes that the command should return the first % three smallest magnitudes of the eigen values and the % corresponding eigen vectors.

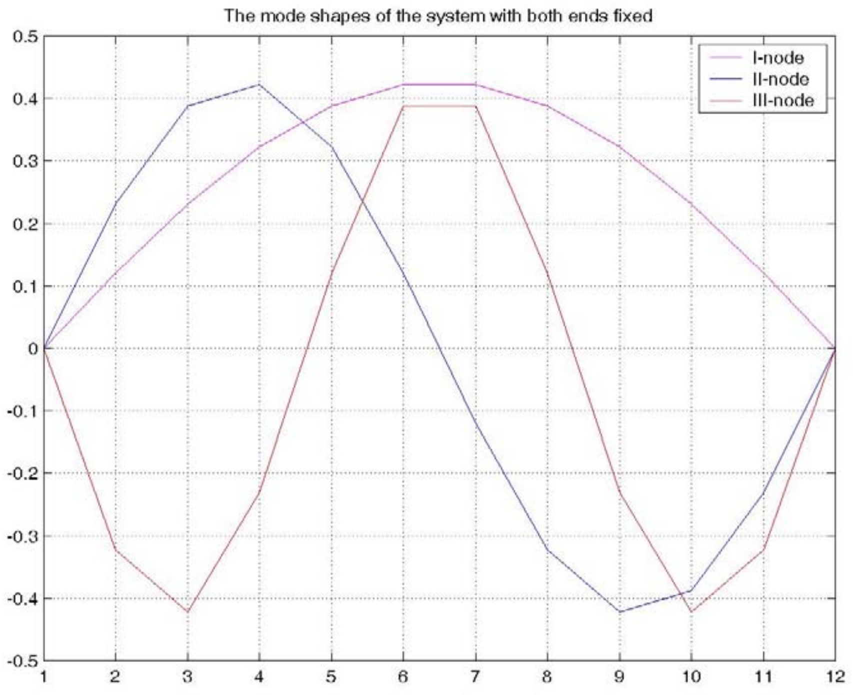
```
number = 3;

[vect1,lamda1]=eigs(K1,M1,number,'SM');

% Since this is for a fixed-fixed boundary condition, the   %
mode shapes originate from the origin. Note that zeros   % have
to be augmented on both sides of the vector.
```

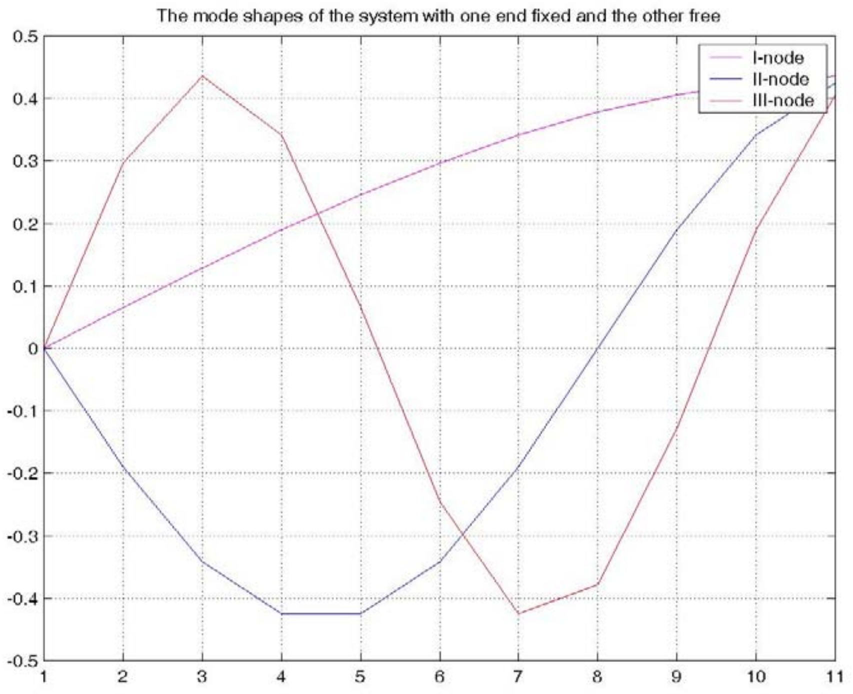
```
vect_aug1 = [0 0 0;vect1;0 0 0];

% wn1 is the vector containing the first three natural        %
frequencies.

wn1 = sqrt(lamda1);

c = ['m','b','r'];

figure(1)
for i=1:size(vect_aug1,2)
   plot(vect_aug1(:,i),c(i))
   hold on;
end

title('The mode shapes of the system with both ends fixed')
grid on
legend('I-node', 'II-node','III-node')
axis([1 12 -0.5 0.5])

%**********************************************************%
% This is the second code for a system where one end is       %
fixed and the other free.
%**********************************************************%

M2 = m*eye(10);
K2 = k*[2 -1 0 0 0 0 0 0 0 0;
   -1 2 -1 0 0 0 0 0 0 0;
   0 -1 2 -1 0 0 0 0 0 0;
   0 0 -1 2 -1 0 0 0 0 0;
   0 0 0 -1 2 -1 0 0 0 0;
   0 0 0 0 -1 2 -1 0 0 0;
   0 0 0 0 0 -1 2 -1 0 0;
   0 0 0 0 0 0 -1 2 -1 0;
   0 0 0 0 0 0 0 -1 2 -1;
   0 0 0 0 0 0 0 0 -1 1;];

[v2,d2] = eig(K2,M2)

% This code is written to separate the natural frequencies % of
the system. w_nat2 contains the natural frequencies.

cnt =1;
for i = 1:10
   for j=1:10
      if(i==j)
         w_nat2(cnt)=sqrt(d2(i,j));
```

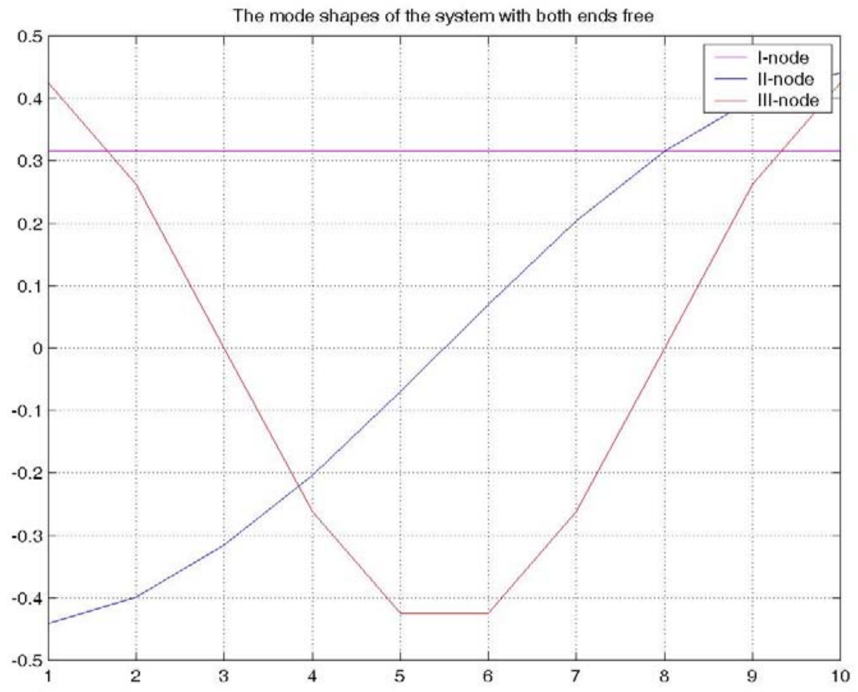
```
            cnt = cnt +1;
        end
    end
end

% To plot the first three mode shapes, we need to find the %
first three eigen values. The 'EIGS' command has been     % used
to find the required eigen values and the                  %
corresponding eigen vectors. 'SM' in the code denotes      % that
the command should return the first three smallest  % magnitudes
of the eigen values and the corresponding      % eigen vectors.

number = 3;

[vect2,lamda2]=eigs(K2,M2,number,'SM');

% wn1 is the vector containing the first three natural       %
frequencies.
wn2 = sqrt(lamda2);

% Since this is for a fixed-free boundary condition, the     %
mode shapes originate from the origin. Note that zeros    % have
to be augmented only on one side of the vector.

vect_aug2 = [0 0 0;vect2];


c = ['m','b','r'];

figure(2)
for i=1:size(vect_aug2,2)
   plot(vect_aug2(:,i),c(i))
   hold on;
end

title('The mode shapes of the system with one end fixed and the
other free')
grid on
legend('I-node', 'II-node','III-node')

%***********************************************************%
% The following code is for a system where both the ends    % is
free.
%***********************************************************%

M3 = m*eye(10);
K3 = k*[1 -1 0 0 0 0 0 0 0 0;
```

```
   -1 2 -1 0 0 0 0 0 0 0;
   0 -1 2 -1 0 0 0 0 0 0;
   0 0 -1 2 -1 0 0 0 0 0;
   0 0 0 -1 2 -1 0 0 0 0;
   0 0 0 0 -1 2 -1 0 0 0;
   0 0 0 0 0 -1 2 -1 0 0;
   0 0 0 0 0 0 -1 2 -1 0;
   0 0 0 0 0 0 0 -1 2 -1;
   0 0 0 0 0 0 0 0 -1 1;];

[v3,d3] = eig(K3,M3)

% This code is written to separate the natural frequencies % of
the system. w_nat3 contains the natural frequencies.

cnt =1;
for i = 1:10
   for j=1:10
      if(i==j)
         w_nat3(cnt)=sqrt(d3(i,j));
         cnt = cnt +1;
      end
   end
end

% To plot the first three mode shapes, we need to find the %
first three eigen values. The 'EIGS' command has been     % used
to find the required eigen values and the                 %
corresponding eigen vectors. 'SM' in the code denotes     % that
the command should return the first three smallest  % magnitudes
of the eigen values and the corresponding     % eigen vectors.
number = 3;

[vect3,lamda3]=eigs(K3,M3,number,'SM');

% wn1 is the vector containing the first three natural       %
frequencies.
wn3 = sqrt(lamda3);

% Since this is for a free-free boundary condition, the      %
mode shapes do not originate from the origin. Note that  % here
zeros should not to be augmented to the vector.

c = ['m','b','r'];

figure(3)
for i=1:size(vect3,2)
```

```
   plot(vect3(:,i),c(i))
   hold on;
end

title('The mode shapes of the system with both ends free')
grid on
legend('I-node', 'II-node','III-node')
```


The mode shapes of the system with both ends fixed
I-node
II-node
III-node
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
1
2
3
4
5
6
7
8
9
10
11
12


The mode shapes of the system with one end fixed and the other free
I-node
II-node
III-node
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
1
2
3
4
5
6
7
8
9
10
11


The mode shapes of the system with both ends free
I-node
II-node
III-node
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
1
2
3
4
5
6
7
8
9
10

Assignment

For the system shown in case (1), suppose the support at one of the end were to oscillate harmonically, whose equation is given by

$$x_1 = F_0 \sin\omega \quad t$$

Calculate the amplitude and phase of each of the masses and plot the amplitude and the phase with respect to the excitation frequency.