Experiment 7

Blood Flow in Arteries

Aim:

To determine and visualize the velocity of blood **in arteries** with certain assumptions using Womersley Equation, Womersley Numbers and Bessel Function

Mathematical Background:

How fast does the blood flow through the arteries with each beat? Is there any relation between the pulse and the flow of blood? These are questions addressed by Womersley's in his research paper. The more we can understand the nature of how the blood flow and perform accurate calculations of arterial blood flow, the more we can detect heart disease and defects.

Womersely's model makes use of Navier-Stokes equations. After accounting for simplification of Navier Stokes Equation and introducing the blood flow velocity(u), we have the DE as:

$$\frac{\partial^2 w}{\partial r^2} + \frac{1}{r}\frac{\partial w}{\partial r} - \frac{1}{v}\frac{\partial w}{\partial t} = -\frac{a_n e^{ifnt}}{\mu}$$

The above equation can be compared with Bessel's Equation of order zero, and hence, the solution (when compared with solution of Bessel's order zero Equation) comes out to be:

$$u_n(r) = c_1 J_0(ri^{\frac{3}{2}}\sqrt{\frac{fn}{v}}) + c_2 Y_0(ri^{\frac{3}{2}}\sqrt{\frac{fn}{v}})$$

The Womersley number (α) is a dimensionless number in biofluid mechanics. It is named after John R. Womersley (1907–1958) for his work with blood flow in arteries. It is expressed as:

$$\alpha = L\left(\frac{\omega\rho}{\mu}\right)^{\frac{1}{2}}$$

where:

L -> Length of cross section

omega-> Angular Velocity

nu -> Viscocity

rho-> Density

It is 2.21 for a normal human artery

Code:

```
clear vars;close all;clc
syms t p y m A R r C %omega mu
for j=1:4
    alpha = input('Input the womersley number: ');
    w = input('Input angular velocity: ');
    u = (A*R^2/1i*alpha^2)*(1-besselj(0, y*(1i^3/2)*alpha)/besselj(0, (1i^3/2)*alpha));
    u_sub = subs(u,{A,R},[1,1]);
    ww = real(u_sub*exp(1i*w*t));
    t_vec = 0:pi/6:2*pi;
    ww_sub = subs(ww,t,t_vec);
    subplot(2,2,j)
    for i=1:numel(ww_sub)
        if j<3
            ezplot(ww_sub(i),[-1 1 -1.5 1.5]);
            hold on;
        else
            ezplot(ww_sub(i),[-1 1 -0.1 0.1]);
            hold on;
        end
    end
    title(['alpha: ',num2str(alpha),'  Angular Frequency: ',num2str(w)])
end
```

Question 1: Major Assumptions used in Mathematical Formulation

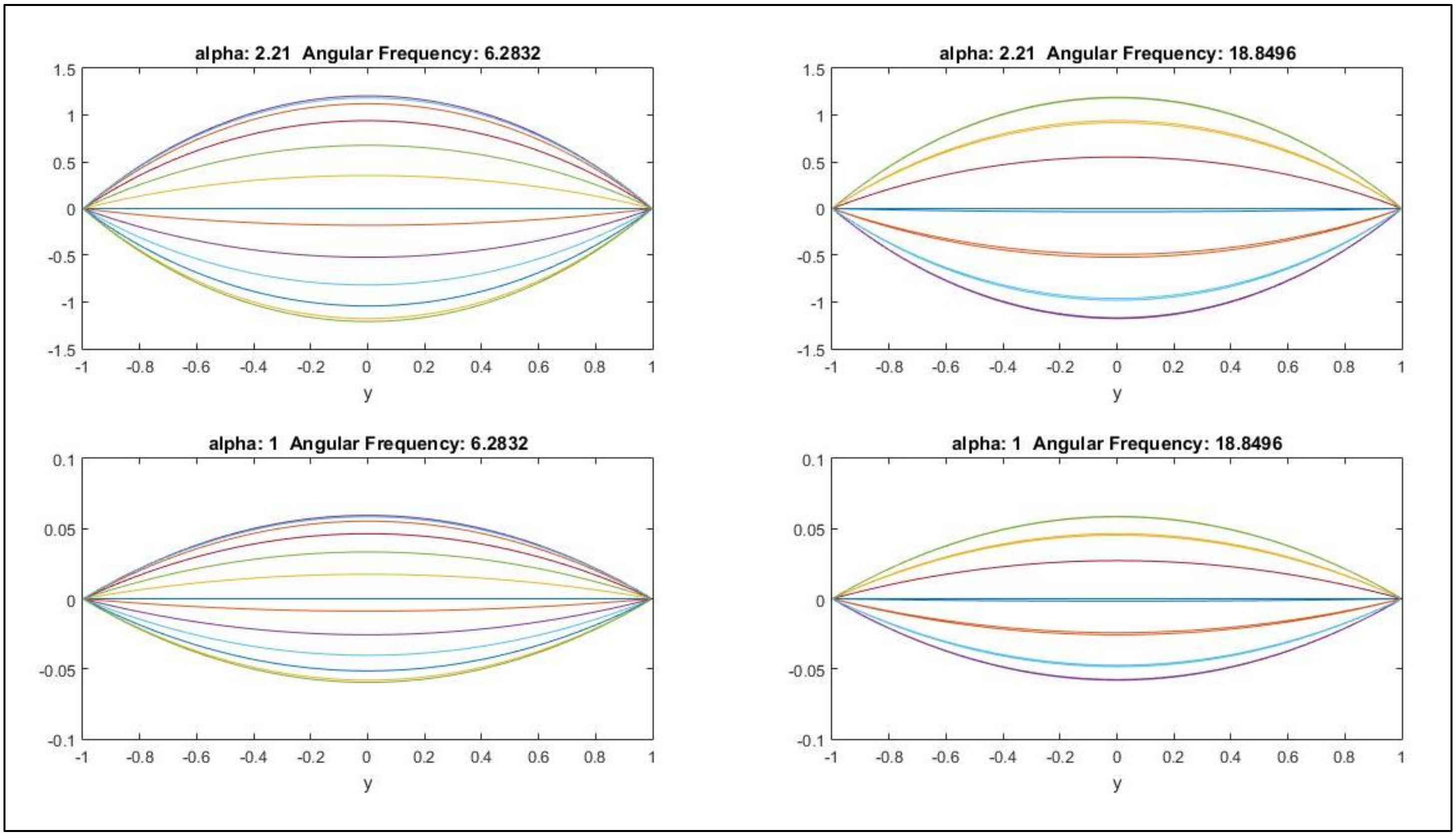
- Velocity at ends is almost Zero (No slip condition).
- Velocity at middle is finite.
- Blood is considered as a Newtonian Fluid (no RBC Plasma etc.).
- Artery is considered as a perfect cylinder (with both ends open).
- Blood is propagating due to Pressure Gradient

Question 2: Plotting Blood Flow Velocity at two different Womersley Numbers (alpha) at two different angular velocities (small omega) (total - 4)

Code: As written in the code section above

I/O:

```
Command Window
Input the womersley number: 2.21
Input angular velocity: 2*pi
Input the womersley number: 2.21
Input angular velocity: 6*pi
Input the womersley number: 1
Input angular velocity: 2*pi
Input the womersley number: 1
Input angular velocity: 6*pi
fx >>
```


alpha: 2.21 Angular Frequency: 6.2832
alpha: 2.21 Angular Frequency: 18.8496
alpha: 1 Angular Frequency: 6.2832
alpha: 1 Angular Frequency: 18.8496
y
y
y
y

Experiment 8

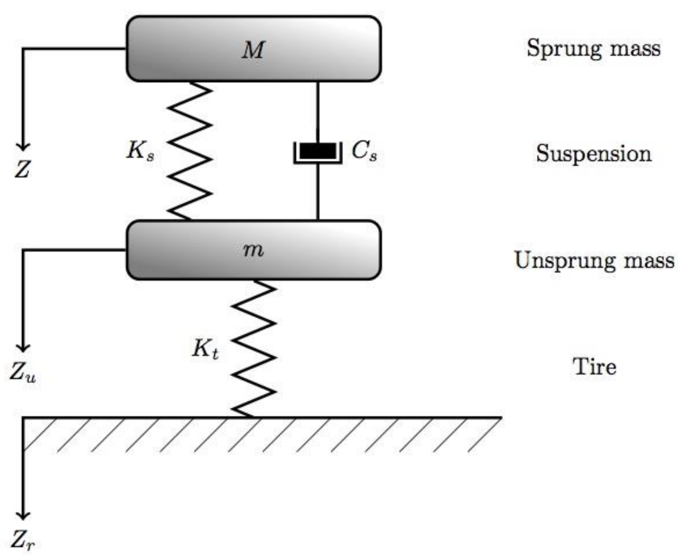
Sprung Mass Displacement in a Quarter Car Model

Aim:

- To solve a system of DEs given in the matrix form X'' + AX (Using Diagonalization)
- Solving a coupled system of masses (coupled system of ODEs). Hence, determining sprung mass displacement in a car suspension of one wheel quarter car model.

Mathematical Background:

The suspension of the car plays an important role in providing comfort and stability to the passengers on both even and uneven road. This design of suspension requires accurate calculation and careful design.



➔ Basic Quarter Car Model

$m_1x_1'' + (c_1+c_2)x_1' - c_2x_2' + (k_1+k_2)x_1 - k_2x_2 = c_1y' + k_1y$ **-> Ignore y (irregularities of road)**

$m_2x_2'' - c_2x_1' + c_2x_2' - k_2x_1 + k_2x_2 = 0$ **-> This equation and above form a coupled system.**

After applying Newton's Law and Hooke's Law to the above model, we get a coupled system which cannot be solved normally. But, we can reduce to an uncoupled system of form X'' + AX = 0 using Y=PX Transformation where P is your modal matrix.

Question 1: Solving the given coupled system of two masses and two springs

Code:

```
%Double Mass System Solving - 16BCE0783
syms y(t)
m1 = input('Input m1: ');
m2 = input('Input m2: ');
k1 = input('Input k1: ');
k2 = input('Input k2: ');
A = inv([m1 0;0 m2])*[-(k1+k2) k2;k1 -k2];
%or A = [m1 0;0 m2]/[-(k1+k2) k2;k1 -k2]
%V2 is diagonal matrix
dim = size(A);
[V1,V2] = eig(sym(A));
sols = [];
for i=1:dim
    temp = dsolve(diff(y,2) == V2(i,i)*y);
    sols = [sols temp];
end
vec_X = V1*transpose(sols);
```

Sample Input and Output:

```
>> apr_22

Input m1: 1

Input m2: 4

Input k1: 2

Input k2: 3

>> vec_X


vec_X =

 (385^(1/2)/4 - 17/4)*(C6*cos((2^(1/2)*t*(23  - 385^(1/2))^(1/2))/4) + C7*sin((2^(1/2)*t*(23  - 385^(1/2))^(1/2))/4)) - (C3*cos((2^(1/2)*t*(385^(1/2)  + 23)^(1/2))/4) + C4*sin((2^(1/2)*t*(385^(1/2) + 23)^(1/2))/4))*(385^(1/2)/4 + 17/4)

                                          C3*cos((2^(1/2)*t*(385^(1/2)  + 23)^(1/2))/4) + C4*sin((2^(1/2)*t*(385^(1/2) + 23)^(1/2))/4) + C6*cos((2^(1/2)*t*(23  - 385^(1/2))^(1/2))/4) + C7*sin((2^(1/2)*t*(23  - 385^(1/2))^(1/2))/4)
```

Question 2: Finding displacement and velocity as a function of time for the given masses, spring constants and initial values and also to plot them

Code:

```
%Double Mass System Solving (x, v, Initial Values and plot) - 16BCE0783
clear vars;close all;clc;syms y(t)
m1 = input('Input m1: ');
m2 = input('Input m2: ');
k1 = input('Input k1: ');
k2 = input('Input k2: ');
x0 = input('Input x1(0): ');
x10 = input('Input diff(x1)(0): ');
x2 = input('Input x2(0): ');
x20 = input('Input diff(x2)(0): ');
A = inv([m1 0;0 m2])*[-(k1+k2) k2;k1 -k2];
dim = size(A);
[V1,V2] = eig(sym(A));
sols = [];
for i=1:dim
    temp = dsolve(diff(y,2) == V2(i,i)*y);
    sols = [sols temp];
end
vec_X = V1*transpose(sols);
matlab_consts = setdiff(symvar(vec_X),[t]);
comp_sols =
solve(subs(vec_X(1),t,0)==x0,subs(diff(vec_X(1),t),t,0)==x10,subs(vec_X(2),t,0)==x2,subs(
diff(vec_X(1),t),t,0)==x20,matlab_consts);
final_vec_X =
subs(vec_X,matlab_consts,[comp_sols.C3,comp_sols.C4,comp_sols.C6,comp_sols.C7]);

subplot(2,2,1)
ezplot(final_vec_X(1),[0 4*pi])
title('16BCE0783')
ylabel('X1')
disp(['The displacement of mass m1:',char(10)])
pretty(vpa(final_vec_X(1)))

subplot(2,2,2)
ezplot(diff(final_vec_X(1)),[0 4*pi])
title('For identification, refer y-axis')
ylabel('V1')
disp(['The velocity of mass m1:',char(10)])
pretty(vpa(diff(final_vec_X(1))))

subplot(2,2,3)
ezplot(final_vec_X(2),[0 4*pi])
title(' ')
ylabel('X2')
disp(['The displacement of mass m2:',char(10)])
pretty(vpa(final_vec_X(2)))

subplot(2,2,4)
ezplot(diff(final_vec_X(2)),[0 4*pi])
title(' ')
ylabel('V2')
disp(['The velocity of mass m2:',char(10)])
pretty(vpa(diff(final_vec_X(2))))
```

I/O

Input m1: 1

Input m2: 4

Input k1: 2

Input k2: 3

Input x1(0): 1

Input diff(x1)(0): 0

Input x2(0): 1

Input diff(x2)(0): 0

The displacement of mass m1:

cos(2.3081761433637539729205652570581 t) 0.32162348299683105937703632918926 +
cos(0.64986374818605404836309888872273 t) 0.67837651700316894062296367081074

The velocity of mass m1:

- sin(2.3081761433637539729205652570581 t) 0.74236365059884341558013886264 119 -
sin(0.64986374818605404836309888872273 t) 0.44085230602107979293332679567113

The displacement of mass m2:

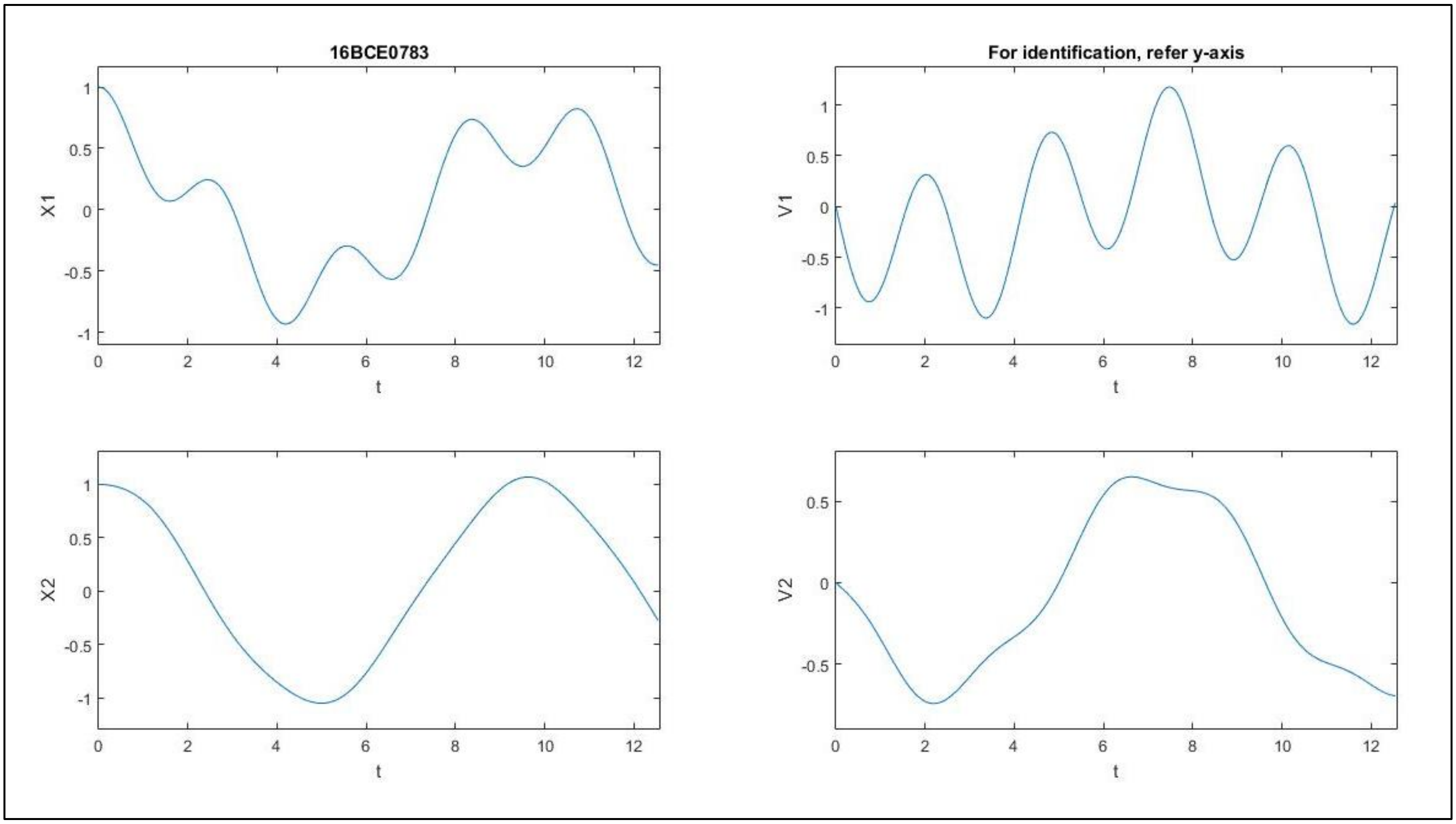
cos(0.64986374818605404836309888872273 t) 1.0351295510095068218688910124322 -
cos(2.3081761433637539729205652570581 t) 0.0351295510095068218688 9101243223

The velocity of mass m2:

sin(2.3081761433637539729205652570581 t) 0.081085191567223726180148669134497 -
sin(0.64986374818605404836309888872273 t) 0.67269316987718533043688149829066

Command Window

```
Input m1: 1
Input m2: 4
Input k1: 2
Input k2: 3
Input x1(0): 1
Input diff(x1)(0): 0
Input x2(0): 1
Input diff(x2)(0): 0
The displacement of mass m1:

cos(2.3081761433637539729205652570581 t) 0.32162348299683105937703632918926 + cos(0.649863748

The velocity of mass m1:

- sin(2.3081761433637539729205652570581 t) 0.74236365059884341558013886264119 - sin(0.6498637

The displacement of mass m2:

cos(0.64986374818605404836309888872273 t) 1.0351295510095068218688910124322 - cos(2.308176143

The velocity of mass m2:

sin(2.3081761433637539729205652570581 t) 0.081085191567223726180148669134497 - sin(0.64986374

>>
```


16BCE0783
For identification, refer y-axis
X1
V1
X2
V2
t

Experiment 9

Vertical Deflection in Swimming Pool Diving Board

Aim:

To finding the Vertical Deflection in Swimming Pool Diving Board (which acts like a cantilever beam) and also Visualize it.

Mathematical Background:

The above aim can be achieved once we have the DE of the Cantilever Beam which can be solved to get the deflection and to plot it. The DE comes out to be

$$\frac{d^4y}{dx^4} = \frac{w(x)}{EI},$$

where:

y is differentiated w.r.t x, w(x) is your weight function, E is the modulus of elasticity and I is the moment of inertia across the cross section.

Now, to solve to this DE, we will use LaPlace Transform which is defined for the function f(t):

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^{+\infty} e^{-st} f(t)dt$$

We take the LaPlace Transform of the given DE (on both sides), then we separate F(s) as a function of 's' and then we take inverse LaPlace Transform of F(s) which ultimately gives us f(t).

$$\mathcal{L}^{-1}\{F(s)\} = f(t)$$

While taking LaPlace Transform, we come across a point where LaPlace of Derivative is required. For that we use the below formulae:

$$\mathfrak{L}\{y'\} = sY(s) - y(0)$$

$$\mathfrak{L}\{y''\} = s^2Y(s) - sy(0) - y'(0)$$

If we are given the values of y(0), Dy(0), D2y(0),…….., then we can solve for f(t) completely without any constants.

Code:

```
%Solving Second Order DE using Laplace Transform
%Use '' to enter string as an argument in function
%sym2str - Copyright (c) 2008, Martin Lawson. All rights reserved.
%Ignore the dots(.) in Output. - 16BCE0783
clear vars;close all;clc
syms y(t) s Y
a = input('Input the coefficient of D2y(t): ');
b = input('Input the coefficient of D1y(t): ');
c = input('Input the coefficient of D0y(t) or y(t): ');
y0 = input('Input the value of y(0): ');
y10 = input('Input the value of Dy(0): ');
type = {'Color','LineStyle'};
value = {'g','k','r',':','--','-'};
sol_store = [];
for j=1:3
    fprintf('Input f1(t) for iteration number %d:\t',j)
    inhomo1 = input('');
    fprintf('Input f2(t) for iteration number %d:\t',j)
    inhomo2 = input('');
    inhomo = inhomo1 + inhomo2;
    f = a*diff(y(t),2)+b*diff(y(t))+c*y(t)-inhomo;
    F = laplace(f);
    F = subs(F,'laplace(y(t),t,s)',Y);
    eq = subs(F,{'y(0)' 'D(y)(0)'},[y0 y10]);
    sol = solve(eq,Y);
    func = ilaplace(sol);
    sol_store = [sol_store func];
    fprintf('The solution for iteration number %d is:\n',j)
    disp(sym2str(func))
    fprintf('\n')
    H = ezplot(func,[0 5*pi]);
    set(H,type(1),value(j),type(2),value(j+3));
    legend(H,sym2str(inhomo))
    hold on;
    title('16BCE0783')
end
legend(sym2str(sol_store(1)),sym2str(sol_store(2)),sym2str(sol_store(3)));
```

Question 1: Solving a second order DE with const. coeff. using LaPlace Transform and given initial conditions with 3 given cases for f1(t) and f2(t)

Code:

As written in the code section above

I/O

Input the coefficient of D2y(t): 1

Input the coefficient of D1y(t): 0

Input the coefficient of D0y(t) or y(t): 1

Input the value of y(0): 1

Input the value of Dy(0): 0

Input f1(t) for iteration number 1: 0

Input f2(t) for iteration number 1: 0

The solution for iteration number 1 is:

cos(t)

Input f1(t) for iteration number 2: 2*dirac(t-2*pi)

Input f2(t) for iteration number 2: 0

The solution for iteration number 2 is:

cos(t) + 2.*sin(t).*heaviside(t - 2.*pi)

Input f1(t) for iteration number 3: 2*dirac(t-2*pi)

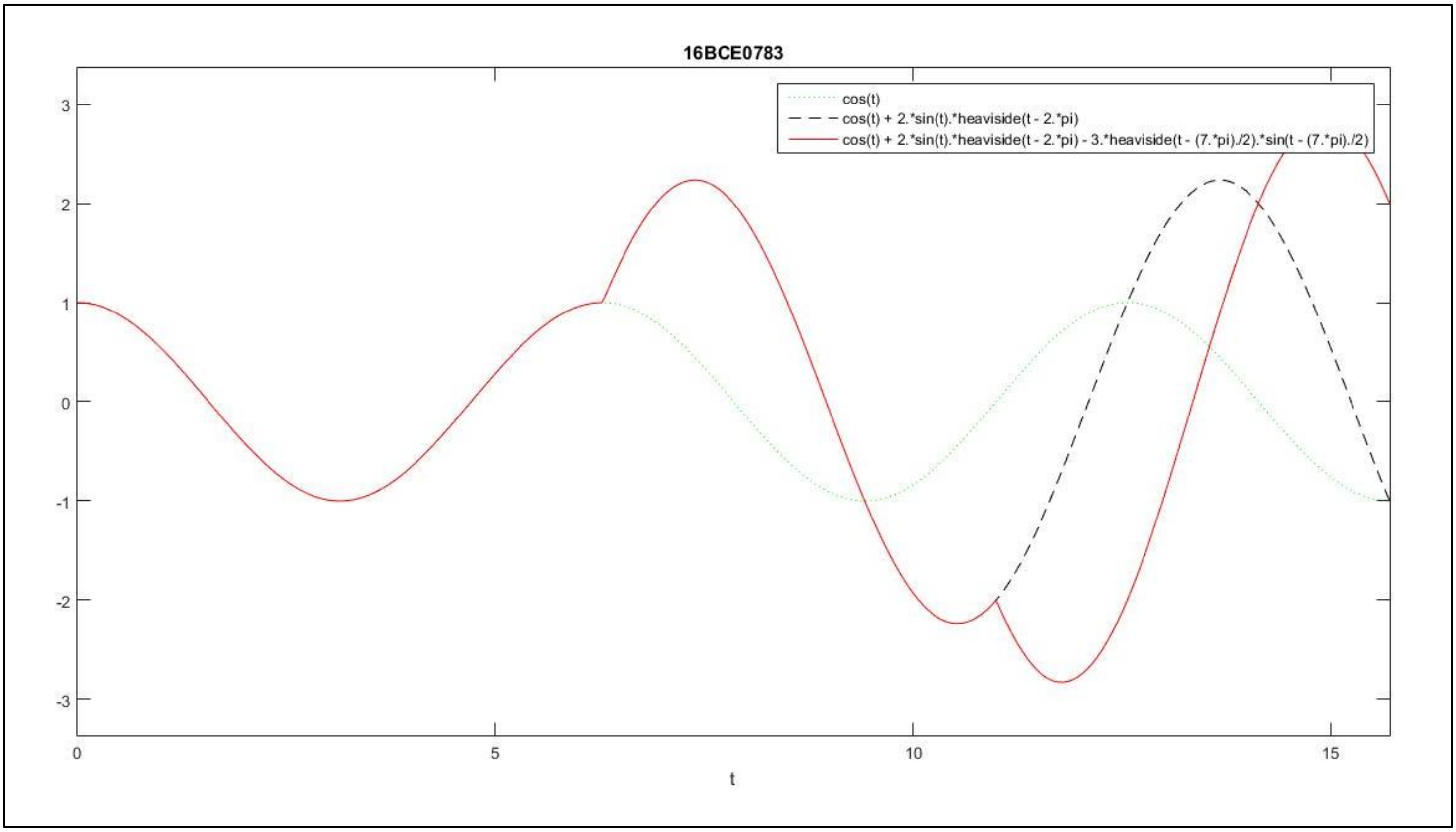
Input f2(t) for iteration number 3: -3*dirac(t-7*pi/2)

The solution for iteration number 3 is:

cos(t) + 2.*sin(t).*heaviside(t - 2.*pi) - 3.*heaviside(t - (7.*pi)./2).*sin(t - (7.*pi)./2)

Command Window

```
Input the coefficient of D2y(t): 1
Input the coefficient of D1y(t): 0
Input the coefficient of D0y(t) or y(t): 1
Input the value of y(0): 1
Input the value of Dy(0): 0
Input f1(t) for iteration number 1: 0
Input f2(t) for iteration number 1: 0
The solution for iteration number 1 is:
cos(t)

Input f1(t) for iteration number 2: 2*dirac(t-2*pi)
Input f2(t) for iteration number 2: 0
The solution for iteration number 2 is:
cos(t) + 2.*sin(t).*heaviside(t - 2.*pi)

Input f1(t) for iteration number 3: 2*dirac(t-2*pi)
Input f2(t) for iteration number 3: -3*dirac(t-7*pi/2)
The solution for iteration number 3 is:
cos(t) + 2.*sin(t).*heaviside(t - 2.*pi) - 3.*heaviside(t - (7.*pi)./2).*sin(t - (7.*pi)./2)

>>
```



Question 2: Deflection in the Cantilever Beam provided with given properties and inputs

Code:

```
%Deflection in Cantilever Beam - 16BCE0783
%sym2str - Copyright (c) 2008, Martin Lawson. All rights reserved.
%Ignore the dots(.) in Output.
clear vars;close all;clc
syms y(x) s Y x a b
E = input('Input the Elasticity(E): ');
I = input('Input the Moment of Inertia(I): ');
L = input('Input the length of Beam(L): ');
wx = input('Input the Weight Function(w(x)): ');
y0 = input('Input the value of y(0): ');
y10 = input('Input the value of Dy(0): ');
y2L = input('Input the value of D2y(L): ');
y3L = input('Input the value of D3y(L): ');
f = diff(y(x),4)-wx/(E*I);
F1 = laplace(f,x,s);
F = subs(F1,'laplace(y(x),x,s)',Y);
eq = subs(F,{'y(0)' 'D(y)(0)' 'D(D(y))(0)' 'D(D(D(y)))(0)'},[y0 y10 a b]);
sol = solve(eq,Y);
func_temp = ilaplace(sol,s,x);
temp_1 = subs(diff(func_temp,x,2),x,L) - y2L;
temp_2 = subs(diff(func_temp,x,3),x,L) - y3L;
sol_ab = solve(temp_1,temp_2,a,b);
func = subs(func_temp,{a,b},{sol_ab.a sol_ab.b});
disp(['The Deflection in the Cantilever Beam is: ',char(10),sym2str(func)])
ezplot(-func,[0 L]);
xlabel('Length of the Beam - 16BCE0783')
ylabel('Deflection in the Beam')
```

I/O:

Input the Elasticity(E): 2.1*10^11

Input the Moment of Inertia(I): 4.5*10^-11

Input the length of Beam(L): 4

Input the Weight Function(w(x)): x

Input the value of y(0): 0

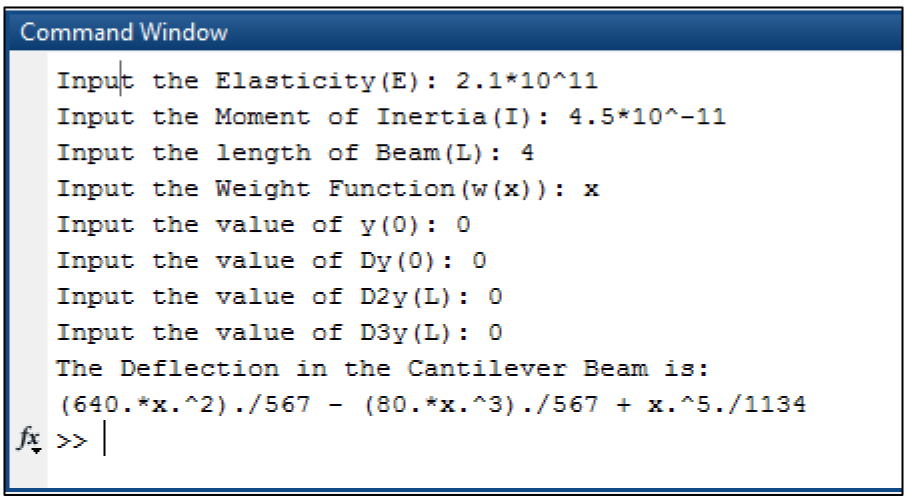
Input the value of Dy(0): 0

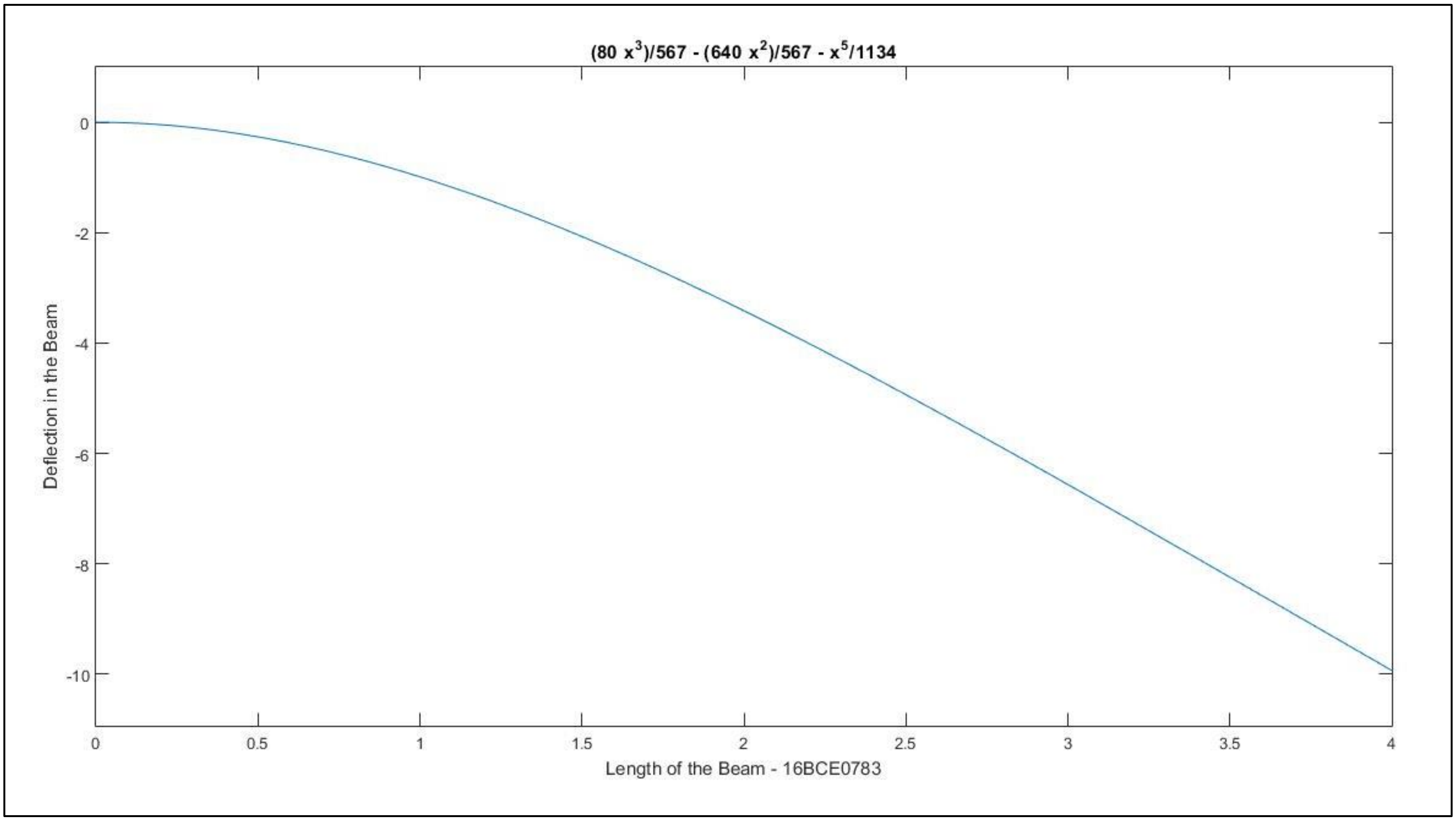
Input the value of D2y(L): 0

Input the value of D3y(L): 0

The Deflection in the Cantilever Beam is:

(640.*x.^2)./567 - (80.*x.^3)./567 + x.^5./1134





(80 x3)/567 - (640 x2)/567 - x5/1134
Deflection in the Beam
Length of the Beam - 16BCE0783
0
-2
-4
-6
-8
-10
0
0.5
1
1.5
2
2.5
3
3.5
4

Experiment 10

Stability of Motor Driven Gear Train using Z-transform

Aim:

- To find, understand and visualize poles and zeroes of a transfer function.
- To determine the stability of the Motor Driven Gear Train using Z-Transform and to plot Impulse and step function for the transfer function of the same.

Mathematical Background:

The Z-Transform of a sequence x(n) is given by:

$$Zx_n = X(z) = \sum_{n=0}^{\infty} x_n z^{-n}$$

And the inverse z-transform is given by:

$$x_n = Z^{-1}\left\{X(z)\right\}$$

When the transfer function is properly factored, the roots of the numerator are called zeroes and the roots of the denominator are called poles. A system is said to be stable only if all the poles lie inside of a unit circle on the z-plane.

A Gear train system is a mechanical system formed by mounting gears on a frame so that the teeth of the gears engage. Gear train system occurs in many of the automobile drivetrains. Gearing is employed in the transmission, which contains a number of different sets of gears that can be changed to allow a wide range of vehicle speeds. The transfer function of the gear train is given by:

$$H(z) = \frac{bz}{z^2 - (1+a)z + a}$$

Where a and b are:

$$a = \frac{R_2LC}{(R_1+L)(R_2C+1)+R_2} \text{ and } b = \frac{R_2}{(R_1+L)(R_2C+1)+R_2}$$

Code:

```
%Stability of System using Z Transform - 16BCE0783
%cross tells poles and bubbles tell zeroes
%dimpulse(num,den,number of impulses)
%dstep(num,den,number of step)
clear all;close all;clc
syms z
r1 = input('Input the value of R1: ');
r2 = input('Input the value of R2: ');
l = input('Input the value of L: ');
c = input('Input the value of C: ');
a = (r2*l*c)/((r1+l)*(r2*c+1)+r2);
b = (r2)/((r1+l)*(r2*c+1)+r2);
h = (b*z)/(z^2-(1+a)*z+a);
[z,p,k] = tf2zpk([b 0],[1 -(1+a) a]);
subplot(2,2,1)
zplane(z,p)
title('16BCE0783')
m1 = double(vpa(max(p)));
m2 = double(vpa(min(p)));
fprintf('\nThe largest of poles is %.2f and the smallest of poles is %.2f1\n',m1,m2);
if ((m1<1.0) && (m2>-1.0))
    disp('The system is stable.')
else
    disp('The system is unstable.')
end
subplot(2,2,2)
dstep([b 0],[1 -(1+a) a],10);
subplot(2,2,3)
dimpulse([b 0],[1 -(1+a) a],10);
```

Question: Determine the poles, zeros and stability of the transfer function for the motor driven gear train using R1 = 1ohms, R2 = 1ohms, L = 2Henry and C = 2 farad. Also plot the impulse and unit step responses.

Code: As written in the code section above

I/O:

Input the value of R1: 1

Input the value of R2: 1

Input the value of L: 2

Input the value of C: 2

The largest of poles is 1.00 and the smallest of poles is 0.401

The system is unstable.

Command Window

```
Input the value of R1: 1
Input the value of R2: 1
Input the value of L: 2
Input the value of C: 2

The largest of poles is 1.00 and the smallest of poles is 0.401
The system is unstable.
>>
```