1. **Problem Definition:** FE modeling of 2-D Stenosed, Compliant Vessel Wall to obtain its deformation under external pressure load. The larger motivation is to set up this model in response to stresses applied by the fluid (external force, here) flowing around it on the solid, compliant wall.

2. **FEM Formulation:**

Quasi-static Equation of Motion:

$$\frac{\partial \sigma_{ij}}{\partial X_j} + f_i = 0 \tag{1}$$

Constitutive Relationship for Viscoelasticity (Kelvin Voigt Model):

$$\sigma_{ij} = C_{ijkl}\epsilon_{kl} + \mu_{ijkl}\frac{d\epsilon_{kl}}{dt} \tag{2}$$

Deriving Weak-formualtion with η_i as the test functions under the assumption of **small strain**:

$$\int_R \frac{\partial \sigma_{ij}}{\partial X_j}\eta_i dV + \int_R f_i \eta_i dV = 0$$

Using Chain-rule on the first term, and Gauss-Divergence Theorem on the second term,

$$\int_R \sigma_{ij}\frac{\partial \eta_i}{\partial X_j} dV + \int_S f_i \eta_i dS = 0 \tag{3}$$

Substituting (2) in (3), we get

$$\int_R \left(C_{ijkl}\epsilon_{kl} + \mu_{ijkl}\frac{d\epsilon_{kl}}{dt}\right)\frac{\partial \eta_i}{\partial X_j} dV + \int_{S^*} f_i \eta_i dS = 0$$

where S^* is the inner wall of the vessel.

Using the isoparametric shape-functions for both u_k and η_i,

$$u_k = N^a u_k^a$$
$$\eta_i = N^b \eta_i^b$$

$$\left(\int_R C_{ijkl}\Big(\frac{\partial N^a}{\partial X_l}\Big)\Big(\frac{\partial N^b}{\partial X_j}\Big)dV\right)u_k^a + \int_R \mu_{ijkl}\frac{d(u_k^a\frac{\partial N^a}{\partial X_l})}{dt}\frac{\partial N^b}{\partial X_j}dV = \int_{S^*} f_i N^b dS$$

$$\left(\int_R C_{ijkl}\Big(\frac{\partial N^a}{\partial X_l}\Big)\Big(\frac{\partial N^b}{\partial X_j}\Big)dV\right)u_k^a + \left(\int_R \mu_{ijkl}\frac{\partial N^a}{\partial X_l}\frac{\partial N^b}{\partial X_j}dV\right)\left(\frac{du_k^a}{dt}\right) = \int_{S*} f_i N^b dS$$

$$[K^e][u_k^a] + [K^v]\left(\frac{d[u_k^a]}{dt}\right) = [F] \tag{4}$$

$$[K^e] = \int_R C_{ijkl}\Big(\frac{\partial N^a}{\partial X_l}\Big)\Big(\frac{\partial N^b}{\partial X_j}\Big)dV = \int_R [B]^T[D^e][B]dV =$$

$$[K^v] = \int_R \mu_{ijkl}\frac{\partial N^a}{\partial X_l}\frac{\partial N^b}{\partial X_j}dV = \int_R [B]^T[D^v][B]dV$$

$$[D^e] = \frac{E}{(1+\nu)(1-2\nu)}\begin{pmatrix}(1-\nu) & \nu & 0\\ \nu & (1-\nu) & 0\\ 0 & 0 & 0.5(1-2\nu)\end{pmatrix}$$

$$[D^v] = \frac{\eta}{(1+\nu)(1-2\nu)}\begin{pmatrix}(1-\nu) & \nu & 0\\ \nu & (1-\nu) & 0\\ 0 & 0 & 0.5(1-2\nu)\end{pmatrix}$$

Using Forward-Euler Scheme to approximate $\left(\frac{d[u_k^a]}{dt}\right)$,

$$[K^e][u_k^a]^N + [K^v]\left(\frac{[u_k^a]^{N+1} - [u_k^a]^N}{\Delta t}\right) = [F]$$

$$[K^v][u_k^a]^{N+1} = \Delta t[F] + \left([K^v] - \Delta t[K^e]\right)[u_k^a]^N \tag{5}$$

Boundary Condition:
Nodal Displacement on inner curve of the wall.

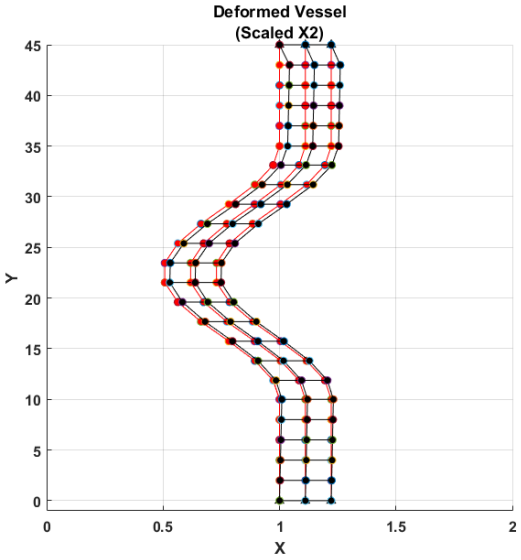
$$u_1^a = 0.005X_1^a + 0.00002 \qquad \forall a \in \text{Inner curve}$$

(As a representative for declining fluid pressure across the length of vessel)

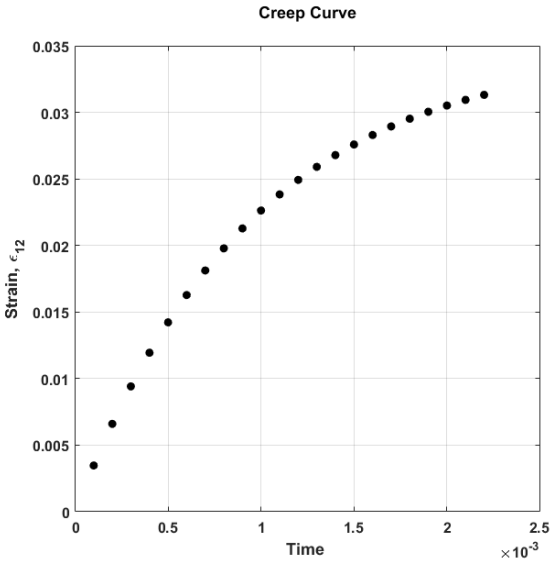
3. **Simulations & Result**

Table 1: Data Table

Property	*Value*
E	1e6
η	1e3
ν	0.3
Vessel wall inner radius, Ri	1
Vessel thickness, h	0.33
Percentage stenosis, f	0.5



(a)



(b)

A Appendix I

```
% ENGN 2340: Project

clc
clear all
close all


factor=2;

% Material Properties

E  = 10^6;                  % Young's modulus
eta = 10^3;
nu = 0.3;                   % Poisson's ratio

% Vessel Properties

f  = 0.5;                  % Percentage stenosis
Ri = 1;                    % Inner Radius
h  = Ri/3;                 % Vessel wall thickness
ls = 25;                   % Stenosis length
le = 10;                    % Length of pre- and post- stenosis regions


% Mesh properties

nx_elem  = 2;              % No. of elements in x-direction
ny_elem  = 23;             % No. of elements in y-direction

n_elems = nx_elem*ny_elem;

% PRE-PROCESSING

% Part 1: [D] matrix assembly, Plain strain

coeffe = E/((1+nu)*(1-2*nu));
coeffv =eta/((1+nu)*(1-2*nu)) ;

De = coeffe*[(1-nu) nu 0;nu (1-nu) 0; 0 0 0.5*(1-2*nu)];
Dv = coeffv*[1-nu nu 0;nu 1-nu 0;0 0 0.5*(1-2*nu)];

% Part 2: Mesh Generation

nx_nodes = nx_elem+1;
```

```
ny_nodes = ny_elem+1;

n_nodes = nx_nodes*ny_nodes;

disp(['No. of elements:',num2str(n_elems)]);
disp(['No. of nodes:',num2str(n_nodes)]);

nodes = getnodes(Ri,f,h,ls,le,nx_nodes,ny_nodes);

connectivity = [1 2 nx_nodes+2 nx_nodes+1];
add_x=1;
add_y=nx_nodes;

element = getelems(connectivity, nx_elem,ny_elem,add_x,add_y);

% Part 3: Defining Boundaries

topleft = nx_nodes*(ny_nodes-1)+1;
topright = nx_nodes*ny_nodes;
bottomleft = 1;
bottomright = nx_nodes;

outeredge=[bottomright:nx_nodes:topright-1;bottomright+nx_nodes:nx_nodes:topright]'
inneredge=[topleft:-nx_nodes:bottomleft+1;topleft-nx_nodes:-nx_nodes:1]';

bottomedge=[bottomleft:1:bottomright-1;bottomleft+1:1:bottomright]';
topedge=[topright:-1:topleft+1;topright-1:-1:topleft]';

% Part 4: Displacement Boundary conditions

fixedNodeX=[bottomleft:1:bottomright,topleft:1:topright]';      % Top & Bottom Edge
fixedNodeY=[bottomleft:1:bottomright,topleft:1:topright]';      % Top & Bottom Edge


plotmesh(nodes,element,'r');
hold on;

plot(nodes(fixedNodeX,1),nodes(fixedNodeX,2),'k^','MarkerFaceColor','black');
hold on;
plot(nodes(fixedNodeY,1),nodes(fixedNodeY,2),'k^','MarkerFaceColor','black');
hold on;

axis([0 2 -1 ls+2*le]);

uFixed=zeros(size(fixedNodeX));
vFixed=zeros(size(fixedNodeY));

prescNodeX=(inneredge(2:end,1));
```

```
prescNodeY=(inneredge(1:end-1,2));

yf=nodes(inneredge(2:end,1),2);
pdisp=0.0005*yf+0.00002;
uPres=pdisp.*ones(size(prescNodeX));
vPres=0.0*ones(size(prescNodeY));

uBC=[uFixed;uPres];
vBC=[vFixed;vPres];

% FEM

% Initializing matrices: U, Ke, f, B

Un=zeros(2*n_nodes,1);

Un(2*fixedNodeX-1,1)=uFixed;
Un(2*fixedNodeY,1)=vFixed;
Un(2*prescNodeX-1,1)=uPres;
Un(2*prescNodeY,1)=vPres;


Un1=zeros(2*n_nodes,1);
F=zeros(2*n_nodes,1);
Ke=zeros(2*n_nodes,2*n_nodes);
Kv=zeros(2*n_nodes,2*n_nodes);

tol=10^-3;
calcTol=1;

dt=10^-4;
iter=1;
t=0;

while calcTol>tol

    disp(['Iter.: ',num2str(iter),' calcTol: ',num2str(calcTol)]);

    for e=1:n_elems

        local=element(e,:);

        xilist=ip_loc(2,4);                    % 2X4
        w=integweight(2,4);

        localK(1:2:8,1)=2*local-1;
        localK(2:2:8,1)=2*local;
```

```
    % Calculating Kel

    for ip=1:4

     xi=xilist(:,ip);


     N=shapefunction(2,4,xi);
     dNdxi=Nderiv(2,4,xi);                % 4X2

     dxdxi=nodes(local,:)'*dNdxi;         % 2X2

     dxidx=inv(dxdxi);                    % 2X2

     dNdx=dNdxi*dxidx;                    % 4X2

     detJ=det(dxdxi);

     B = zeros(3,8);

     B(1,1:2:8)=dNdx(:,1);
     B(2,2:2:8)=dNdx(:,2);
     B(3,1:2:8)=dNdx(:,2);
     B(3,2:2:8)=dNdx(:,1);

     Ke(localK,localK)=Ke(localK,localK)+B'*De*B*w(ip)*detJ;
     Kv(localK,localK)=Kv(localK,localK)+B'*Dv*B*w(ip)*detJ;

    end
end

% Boundary displacement condition

bwt=mean(diag(Kv));

udofs=[2*fixedNodeX-1;2*prescNodeX-1];
vdofs=[2*fixedNodeY;2*prescNodeY];

F(udofs)=Ke(udofs,:)*Un;
F(vdofs)=Ke(vdofs,:)*Un;

Kv(udofs,:)=0;
Kv(vdofs,:)=0;

Kv(udofs,udofs)=bwt*speye(length(udofs));
Kv(vdofs,vdofs)=bwt*speye(length(vdofs));

Un1=Kv\(dt*F+(Kv-dt*Ke)*Un);
```

```
    calcTol=sqrt(sum((Un1-Un).^2));

    stressnew=zeros(n_elems,size(element,2),3);

    stressPoints=[-1 -1;1 -1;1 1;-1 1];

    for es=1:n_elems

        local_s=element(e,:);
        local=[2*local_s-1 2*local_s];

        for ips=1:length(local_s)
            xi_s=stressPoints(ips,:);
            N_s=shapefunction(2,4,xi_s);
            dNdxi_s=Nderiv(2,4,xi_s);

            dxdxi_s=nodes(local_s,:)'*dNdxi_s;
            dxidx_s=inv(dxdxi_s);
            dNdx_s=dNdxi_s*dxidx_s;

            strainold=B*Un(local);
            strainnew=B*Un1(local);

            stressnew(es,ips,:)= De*strainnew + Dv*(strainnew-strainold)/dt;

            end
    end

    strain(:,iter)=strainnew;

    ssx(iter)=stressnew(es,2,1);
    ssy(iter)=stressnew(es,2,2);
    ssz(iter)=stressnew(es,2,3);

    Un=Un1;
    iter=iter+1;
    t=t+dt;
end

% POST-PROCESSING

% Plotting Deformed Wall

newnodes=nodes+factor*reshape(Un,2,n_nodes)';
plotmesh(newnodes,element,'k');

                    % ---------- %
```

```
% FUNCTION 1

function xi = ip_loc(ncoord,nelnodes)

xi = zeros(ncoord,nelnodes);
npoints=nelnodes;        % For the case of 4-noded quad elem

    if ncoord==1
        if (npoints==1)
           xi(1,1) = 0.;
         elseif (npoints == 2)
           xi(1,1) = -0.5773502692;
           xi(1,2) = -xi(1,1);
         elseif (npoints == 3)
           xi(1,1) = -0.7745966692;
           xi(1,2) = 0.0;
           xi(1,3) = -xi(1,1);
        end

    else
    % Only for 4-noded quadrilateral elements

        xi(1,1) = -0.5773502692;
        xi(2,1) = xi(1,1);
        xi(1,2) = -xi(1,1);
        xi(2,2) = xi(1,1);
        xi(1,3) = xi(1,1);
        xi(2,3) = -xi(1,1);
        xi(1,4) = -xi(1,1);
        xi(2,4) = -xi(1,1);
    end

end


% FUNCTION 2:

function nodes = getnodes(R,f,h,ls,le,nnx,nny)

if (mod(nny,3)~=0)
    disp('Enter number of nodes along y-dir in multiples of 3');
    return
end

dy=(ls+2*le)/nny;
dh=h/nnx;

n=nnx*nny;
```

```
ne=ceil(le*nny/(ls+2*le));
ns=(nny-2*ne);

yo=le+0.5*ls;

length=ls+2*le;

nodes_tempx=zeros(nny,nnx);
nodes_tempy=zeros(nny,nnx);

for i=1:nnx

ye=linspace(0,le,ne);
xe=dh*(i-1)+R*(1+0.*ye);

ys=linspace(le+dy,le+ls-dy,ns);
xs=dh*(i-1)+R*(1-0.5*f*(1+cos(2*pi*((ys-yo)./ls))));

yen=linspace(le+ls,length,ne);
xen=dh*(i-1)+R*(1+0.*yen);

nodes_tempx(:,i)=[xe,xs,xen];
nodes_tempy(:,i)=[ye,ys,yen];

end

A=nodes_tempx';
B=nodes_tempy';

nodes=[A(:), B(:)];

plot(nodes(:,1),nodes(:,2),'o');
%axis square;

end

% FUNCTION 3:

function elements = getelems(connectivity, nx_e,ny_e,addx,addy)

nelem=nx_e*ny_e;
incx=addx*ones(1,4);
incy=addy*ones(1,4);


elements(1,:)=connectivity;

        for j=2:nelem
```

```
            if (mod(j,nx_e)==1)
                elements(j,:)=elements(j-nx_e,:)+incy;
            else
            elements(j,:)=elements(j-1,:)+incx;
            j=j+addx;
            end
        end


end

% FUNCTION 4:

function plotmesh(nodes,elements,c)

nelem=size(elements,1);
nelnodes=size(nodes,2);

        for lmn = 1:nelem
            for i = 1:4
                x(i,1:2) = nodes(elements(lmn,i),1:2);
            end

            scatter(x(:,1),x(:,2),'MarkerFaceColor',c);
            patch('Vertices',x,'Faces',[1 2 3 4],'FaceColor','none','EdgeColor',c);
            hold on

        end

end

% FUNCTION 5:

function w = integweight(ncoord,npoints)

    w=zeros(npoints,1);

    if (ncoord == 1)
     if (npoints == 1)
       w(1) = 2.;
     elseif (npoints == 2)
       w = [1.,1.];
     elseif (npoints == 3)
       w = [0.555555555,0.888888888,0.555555555];
     end

    else
```

```
    w = [1.,1.,1.,1.];

    end
end

% FUNCTION 6:

function dNdxi = Nderiv(ncoord,nelnodes,xi)


if (ncoord == 1)
    if (nelnodes==2)
      dNdxi(1,1) = 0.5;
      dNdxi(2,1) = -0.5;
    elseif (nelnodes == 3)
      dNdxi(1,1) = -0.5+xi(1);
      dNdxi(2,1) =  0.5+xi(1);
      dNdxi(3,1) = -2.*xi(1);
    end

else
       dNdxi(1,1) = -0.25*(1.-xi(2));
       dNdxi(1,2) = -0.25*(1.-xi(1));
       dNdxi(2,1) = 0.25*(1.-xi(2));
       dNdxi(2,2) = -0.25*(1.+xi(1));
       dNdxi(3,1) = 0.25*(1.+xi(2));
       dNdxi(3,2) = 0.25*(1.+xi(1));
       dNdxi(4,1) = -0.25*(1.+xi(2));
       dNdxi(4,2) = 0.25*(1.-xi(1));
end

end

% FUNCTION 7:

function N = shapefunction(ncoord,nelnodes,xi)

if (ncoord == 1)
    if (nelnodes==2)
      N(1) = 0.5*(1.+xi(1));
      N(2) = 0.5*(1.-xi(1));
    elseif (nelnodes == 3)
      N(1) = -0.5*xi(1)*(1.-xi(1));
      N(2) =  0.5*xi(1)*(1.+xi(1));
      N(3) = (1.-xi(1))*(1.+xi(1));
    end
```

```
else

        N(1) = 0.25*(1.-xi(1))*(1.-xi(2));
        N(2) = 0.25*(1.+xi(1))*(1.-xi(2));
        N(3) = 0.25*(1.+xi(1))*(1.+xi(2));
        N(4) = 0.25*(1.-xi(1))*(1.+xi(2));
end

end
```