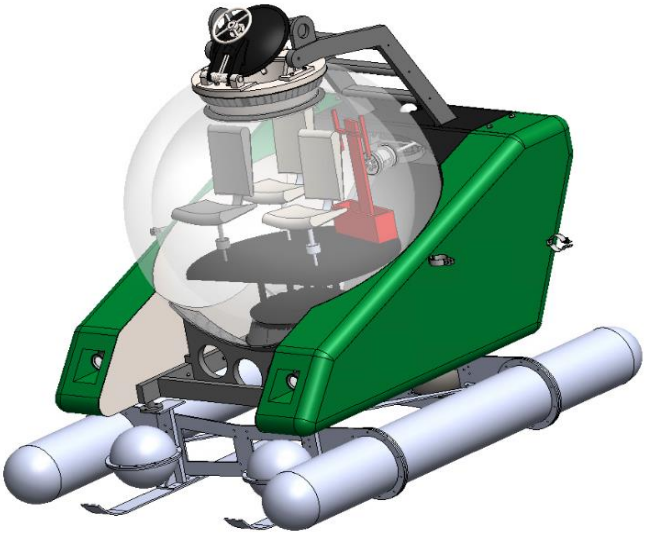
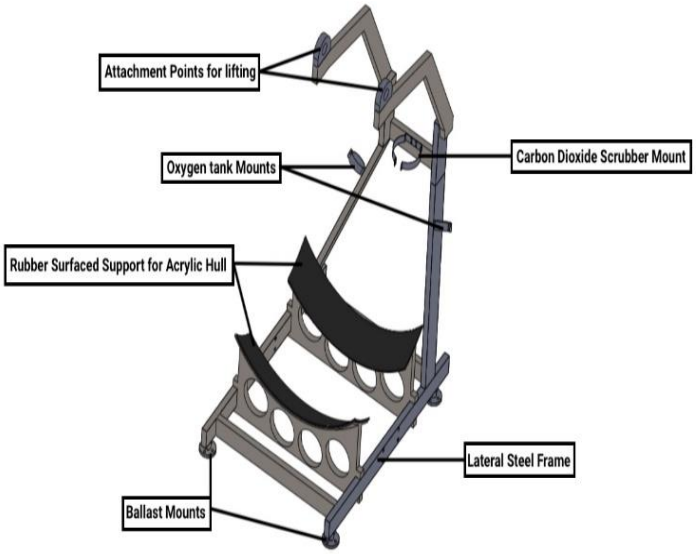
MECH 514

Static Analysis of the Frame of a Manned Submersible





Name: **Jonas Chianu**

Student #: **31298391**

Abstract:

Submarine are used in many different industries such as tourism, recreation, and military, and so it is very important to understand how to analyse its structure for safety. Having an idea of how much stresses are felt by the frame is a very critical analysis to undertake. In this project, static analysis is performed on a 3d deformable solid using Abaqus and the results are validated using a 3d frame analysis performed in Matlab and also by using literature, the maximum stress in the structure can be compared to the yield strength of the material. The results show that the maximum stress felt by the structure is 67.47 MPa and the maximum displacement is around 5mm. The displacement is comparable to the displacement in Matlab of 8mm. The different in displacement is due to the limited number of nodes available in the student version of Abaqus. Also, the maximum stress is less than the yield strength of the material, therefore the Abaqus results are validated. These results are significant because it shows that the analysis is reliable, and the submarine can be safely lifted out of the water with an attachment point on its frame.

1. Introduction and Method:

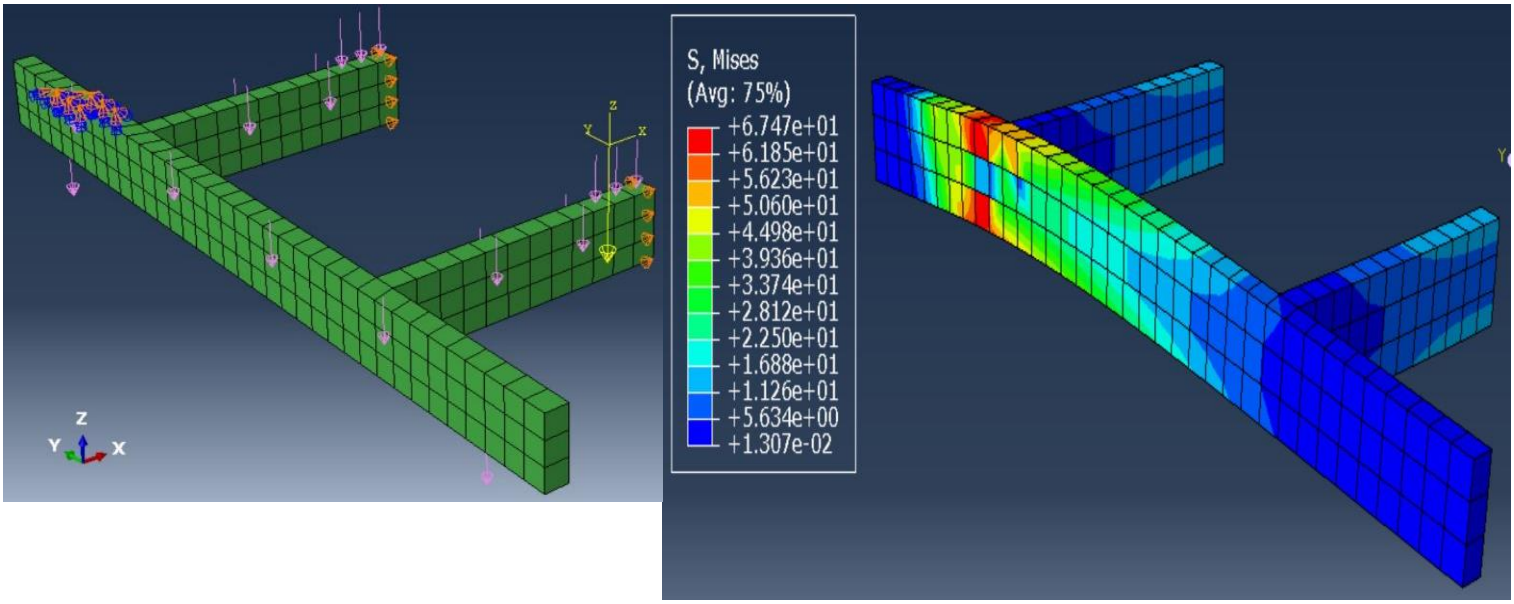
When designing a submersible, there are 2 different structural analysis that must be done to ensure the safety of the crew and the integrity of the structure. The first structural analysis is on the spherical acrylic hull to make sure that it can withstand the pressures at high depth. The second important structural analysis is on the frame when it is subjected to the highest load. I focus on the latter because it is simpler to validate using Matlab. A frame of a manned submersible is most at risk of failure when it is being lifted out of the water, through its attachment point as seen in the above image, to be brought back to land. This is because at the lifting stage, the frame must support the entire weight of the hull, ballast, body (fairing) and the other additional components in addition to its own weight. To make the analysis simpler, only the bottom part of the frame is analyzed since that is where most of the other parts of the submarine are attached to. The frame structure is also simplified to have a constant cross section everywhere. The goal of this project is to analyse the stresses within the bottom part of the frame and to validate that the frame structure will not fail due to the total weight of the submersible. (For more information on manned submersible see "Manned submersible by Frank Busby"(1)).

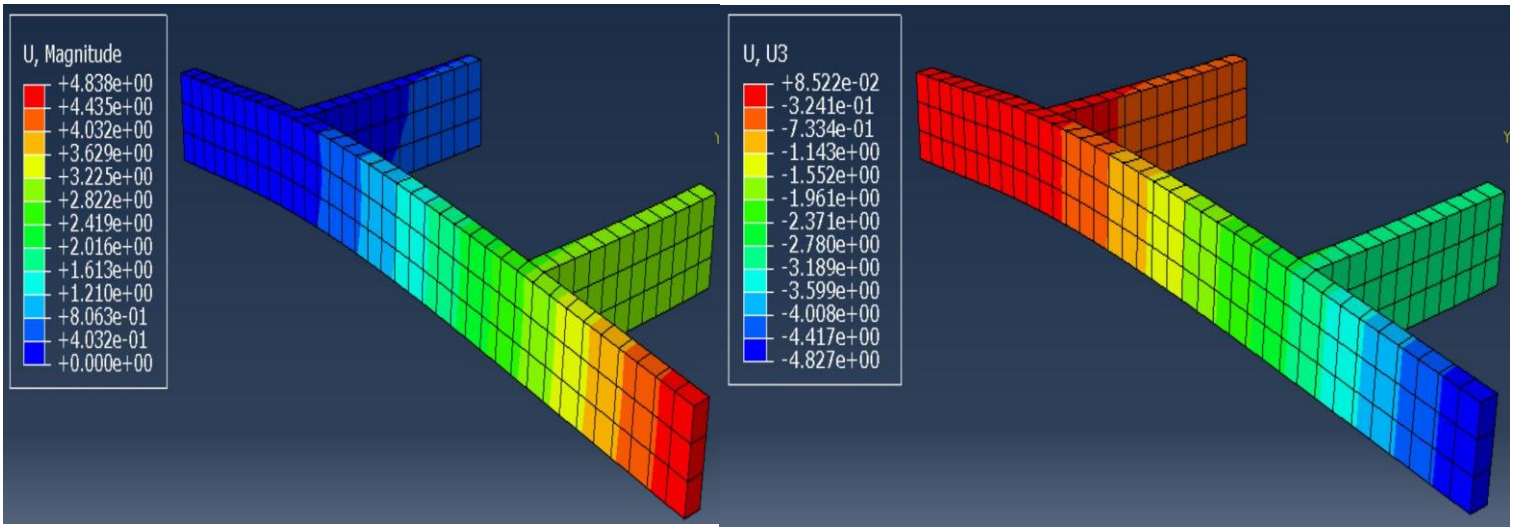
This is accomplished by doing a static analysis on the frame structure with all the weight applied at their respective locations. The location of all the forces on the frame are shown in the image in the appendix along with a table describing the relevant forces. Only the bottom frame is analysed because most of the weight act in this area and the cross section is assumed to be constant to simplify to problem. Due to symmetry, only half of the bottom frame needs to be analyzed, and so the nodes at the plane of symmetry are fixed in the direction normal to the plane. The frame has 150mmX50mm cross section and is made of stainless steel 416 (2).

Within Abaqus, the frame is modeled as a 3d deformable solid. The unloaded structure is meshed and the load is applied to the meshed INP file. This gets rid of the problem of having to create multiple partitions to apply forces. Finally, a general static analysis in conducted on the structure to see the stresses and displacement (magnitude and u_z) of the structure. Within Matlab, a frame structure analysis is conducted to see the displacement of the structure and calculate the various loads in the structure.

2. Results: [units: SI (mm)]

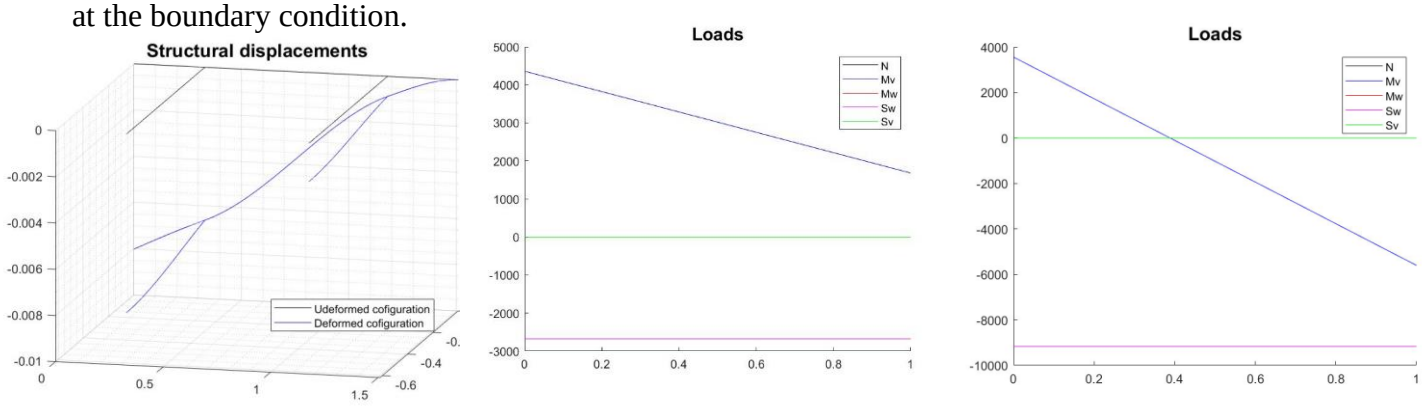
All the Abaqus results are shown below. The first image shows the continuum structure in Abaqus with the forces added at their respective locations. There is an encastre boundary condition where the top part of the frame would have been. The second image shows the stresses on the structure. The max stress is 67.47 MPA and it occurs are regions close to the boundary condition. The third and fourth image show the displacement of the structure at different location, with the third image showing the overall displacement magnitude while the fourth image shows the displacement downwards. In both cases the maximum displacement occurs at the tip of the frame and is 4.8 mm.





3. Validation and Limitations:

The Matlab results are shown below. The first image shows he displacement plot of Matlab and this is used to validates the displacement plot from Abaqus. The maximum downward displacement in Abaqus was around 5mm, while in Matlab it is 8mm, which is fairly similar. Another important plot are the plots of the loads. The two load plots are the plots of locations with maximum displacement which fall near the boundary condition, and although the plots don't validate the stresses in Abaqus, it confirms that the point of maximum stress is in fact at the boundary condition.



A quick validation through literature is to compare the maximum stress felt by the structure in Abaqus to the yield strength of stainless steel 416. The maximum stress must be less than the yield strength to prevent permanent deformation. The yield strength of stainless steel 416 is 275 MPA (2) and the maximum stress in Abaqus is 67.47 MPA. This means the structure doesn't undergo any plastic deformation. Since my result are validated with Matlab and literature, I can safely assume that my results are valid overall, and the structure won't fail.

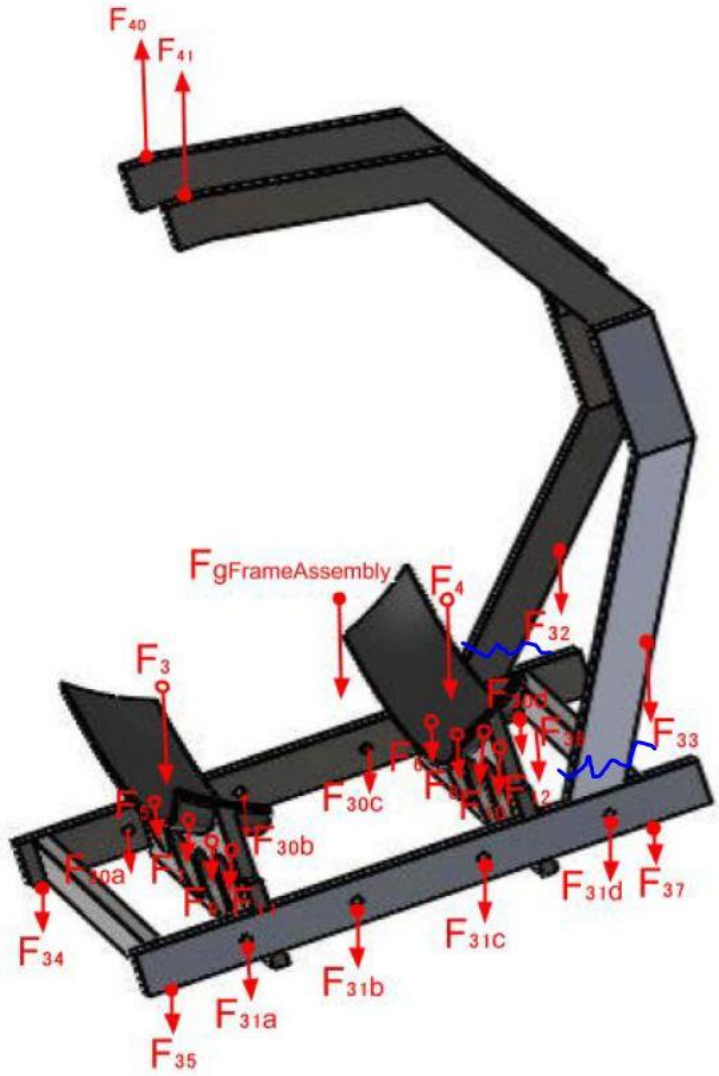
There are some limitations in my results. The student version of Abaqus has a limited number of nodes that can be analyze and so the accuracy of my result aren't as accurate. Also, the Matlab results are for a 3d frame structure not a 3d continuum structure so it is not as accurate. A last limitation is the fact that, my structure is a novel design so there is not a lot out there to compare with.

4. References

1. R. Frank Busby, MANNED SUBMERSIBLES BY the OFFICE OF THE OCEANOGRAPHER OF THE NAVY 1976
2. F. D. O. 23 2001, "Stainless Steel - Grade 416 (UNS S41600)," AZoM.com, 25-Apr-2014. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=971.

5. Appendix:

Below is the frame structure with the load locations and a table showing a description of important forces:



F3, F4	Weight of hull
F5, F6, F7, F8, F9, F10	Weight of Compressed air tanks
F30 (a,b,c,d), F31 (a,b,c,d)	Weight of Body (fairing) and battery on frame
F34, F35, F36, F37	Weight of Ballast and drop weights
FgFrameAssembly	Overall weight of frame
F40, F41	Force at attachment point.

(End of report. The Matlab code for frame structure analysis is attached below)

```
%%% a3D frame structure %%%
clear;
close all;
clc;
% Nodal coordinates
%        xi yi zi
nodes = [ 0  0  0;  % node 1
          0.255  0  0;
          0.33  0  0;  % node 2 we assume a nominal distance
          0.33 -0.22 0;
          0.33 -0.42 0;
          0.33 -0.62 0;
          0.605 0 0;
          1.005 0 0;
          1.18 0 0;
          1.18 -0.22 0;
          1.18 -0.42 0;
          1.18 -0.62 0;
          1.405 0 0;
          1.51 0 0]; % node 4
n = size(nodes,1); % number of nodes
L_total=nodes(14,1)-nodes(1,1)+abs(sum(nodes(:,2)))
% Elements
E = 200e9; % Modulus of Elasticity [pa] of Stainless Steel 416
G = 83e9; % Shear Modulus [pa] of Stainless Steel 416
b=0.05; h=0.15; % width and height of beam cross section
A = b*h; % area of beam cross section
Iy= b*(h^3)/12; Iz= h*(b^3)/12; It=Iy+Iz;
E_A=E*A;
E_Iy= E*Iy; E_Iz= E*Iz; G_It=G*It;
%       i j EA EIy EIz GIt theta_z
elem = [1 2  E_A  E_Iy   E_Iz   G_It       0;  % element 1
        2 3  E_A  E_Iy   E_Iz   G_It       0;  % element 2
        3 4  E_A  E_Iy   E_Iz   G_It       -pi/2;
        4 5  E_A  E_Iy   E_Iz   G_It       -pi/2;
        5 6  E_A  E_Iy   E_Iz   G_It       -pi/2;
        3 7  E_A  E_Iy   E_Iz   G_It       0;
        7 8  E_A  E_Iy   E_Iz   G_It       0;
        8 9  E_A  E_Iy   E_Iz   G_It       0;
        9 10  E_A  E_Iy   E_Iz   G_It       -pi/2;
        10 11  E_A  E_Iy   E_Iz   G_It       -pi/2;
        11 12  E_A  E_Iy   E_Iz   G_It       -pi/2;
        9 13  E_A  E_Iy   E_Iz   G_It       0;
        13 14  E_A  E_Iy   E_Iz   G_It       0]; % element 3
m = size(elem,1); % number of elements
% Stiffness matrix
K = zeros(6*n,6*n); % now a node has 6 dofs, 3 displacement directions and 3 rotation↙
angles
% vector of stiffnesses, angles of inclination, and stiffness matrix;
for q = 1:m
```

```
    % extracting the coordinates of the nodes of pertinence
    i = elem(q,1); xi = nodes(i,1); yi = nodes(i,2); zi = nodes(i,3);
    j = elem(q,2); xj = nodes(j,1); yj = nodes(j,2); zj = nodes(j,3);
    L = norm([xj;yj;zj]-[xi;yi;zi]); Lv(q) = L;
    EA = elem(q,3);
    EIy = elem(q,4);
    EIz = elem(q,5);
    GIt = elem(q,6);
    kh = EA*L^(-1);
    kv = 12*EIy*L^(-3);
    kv_ = 12*EIz*L^(-3);
    kvb = 6*EIy*L^(-2);
    kvb_ = 6*EIz*L^(-2);
    km = 4*EIy*L^(-1);
    km_ = 4*EIz*L^(-1);
    kt = GIt*L^(-1);
    Ke_ = [ kh,   0,     0,   0,     0,    0, -kh,    0,    0,   0,     0,    0;
              0,  kv,    0,   0,     0,  kvb,   0,  -kv,    0,   0,     0,  kvb;
              0,   0,   kv_,   0, -kvb_,    0,   0,    0, -kv_,   0, -kvb_,    0;
    % ** ** **
              0,   0,    0,  kt,     0,    0,   0,    0,    0, -kt,     0,    0;
              0,   0, -kvb_,   0,   km_,    0,   0,    0, kvb_,   0, km_/2,    0;
              0, kvb,    0,   0,     0,   km,   0, -kvb,    0,   0,     0, km/2;
    % ** ** **
             -kh,   0,    0,   0,     0,    0,  kh,    0,    0,   0,     0,    0;
              0, -kv,    0,   0,     0, -kvb,   0,   kv,    0,   0,     0, -kvb;
              0,   0,  -kv_,   0,   kvb_,   0,   0,    0,  kv_,   0,   kvb_,   0;
    % ** ** **
              0,   0,    0, -kt,     0,    0,   0,    0,    0,  kt,     0,    0;
              0,   0, -kvb_,   0, km_/2,    0,   0,    0, kvb_,   0,    km_,   0;
              0, kvb,    0,   0,     0, km/2,   0, -kvb,    0,   0,     0,   km];
    theta = elem(q,7); c = cos(theta); s = sin (theta);
    R = [ c s 0;
         -s c 0;
          0 0 1];
    Re = [         R zeros(3,3);
          zeros(3,3)          R];
    Re_ = [        Re zeros(6,6);
           zeros(6,6)         Re];
    Ke = Re_'*Ke_*Re_;
    edofs = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,6*j-3,6*j-2,6*j-1,6*j];
    K(edofs,edofs) = K(edofs,edofs)+Ke;
end
% Boundary conditions
fix_nod = [6 12 14];
fix_dofs = [];
for nod = 1:size(fix_nod,2)
    ii = fix_nod(nod);
    if ii==6 | ii == 12
        fix_dofs = [fix_dofs 6*ii-5 6*ii-4 6*ii-2 6*ii-1 6*ii];
```

```
    else
        fix_dofs = [fix_dofs 6*ii-5 6*ii-4 6*ii-3 6*ii-2 6*ii-1 6*ii];
    end
end
free_dofs = setdiff([1:6*n],fix_dofs); % the total number of dofs is 2*n (2 per node);
this function curtails the dofs that are not constrained
% Loads
g = 9.81;
W_hull = (2057.06+103.49+475) * 9.81 % weight of hull + hatch + inner components
W_b_d = (800+814)*9.81 %weight of ballast + dropweight
W_comp = 136.0777*9.81 %weight of compressed air tanks
W_body = (778.59+220)*9.81 % weight of fairing + battery
W_frame = 180.1875*9.81
f = zeros(6*n,1);
f(6*1-3) = -W_b_d/4;
f(6*2-3) = -W_body/8;
f(6*4-3) = -W_comp/2;
f(6*5-3) = -W_comp/2;
f(6*6-3) = -W_hull/4;
f(6*7-3) = -W_body/8;
f(6*8-3) = -W_body/8;
f(6*10-3) = -W_comp/2;
f(6*11-3) = -W_comp/2;
f(6*12-3) = -W_hull/4;
f(6*13-3) = -W_body/8;
f(6*14-3) = -W_b_d/4;
for i=1:14
    if i == 1
        f(6*i-3)=f(6*i-3)-(W_frame/2)*(nodes(i+1,1)-nodes(i)+abs(nodes(i,2)))
/2/L_total;
    elseif i==14
        f(6*i-3)=f(6*i-3)-(W_frame/2)*(nodes(i,1)-nodes(i-1)+abs(nodes(i,2)))
/2/L_total;
    else
        f(6*i-3)=f(6*i-3)-(W_frame/2)*(nodes(i+1,1)-nodes(i-1)+abs(nodes(i,2)))
/2/L_total;
    end
end
% Solution
u = zeros(6*n,1);
u(free_dofs) = K(free_dofs,free_dofs)\f(free_dofs);
% Plot results
figure(1)
hold on
grid on
grid minor
view(45,45)
for q =1:m
    i = elem(q,1); j = elem(q,2);
    xi = nodes(i,1); yi = nodes(i,2); zi = nodes(i,3); % undeformed configuration
```

```
    xj = nodes(j,1); yj = nodes(j,2); zj = nodes(j,3);
    L = Lv(q);
    edofs = [6*i-5,6*i-4,6*i-3,6*i-2,6*i-1,6*i,6*j-5,6*j-4,6*j-3,6*j-2,6*j-1,6*j];
    % xi vector along the axis of the frame
    xiv = [0:0.001:1]';
%     if q==1
%         xiv = [0:0.01:0.255]';
%     end
%     elseif q==2 | q==4
%         xiv = [0:0.01:0.62]';
%     elseif q==3
%         xiv = [0:0.01:0.7]';
%     elseif q==5
%         xiv = [0:0.01:0.225]';
%     else
%         xiv = [0:0.01:0.105]';
%     end
    np = size(xiv,1); % number of components of the xi vector
    % shape functions
    xs1 = 1-xiv; xs2 = xiv;
    ys1 = 1-3*xiv.^2+2*xiv.^3; ys2 = L*(xiv-2*xiv.^2+xiv.^3);
    ys3 = 3*xiv.^2-2*xiv.^3; ys4 = L*(-xiv.^2+xiv.^3);
    zerv = zeros(np,1);
    unv = ones(np,1);
    %%% undeformed config
    % matrix of shape functions for mapping the undeformed config
    fu = [ xs1 zerv zerv  xs2 zerv zerv;
           zerv  xs1 zerv zerv  xs2 zerv;
           zerv zerv  xs1 zerv zerv  xs2];
    xuv = fu*[xi;yi;zi;xj;yj;zj]; % vector of points of undeformed configuration
    xu = xuv(1:np);
    yu = xuv(np+1:2*np);
    zu = xuv(2*np+1:3*np);
    figure(1)
    plot3(xu,yu,zu,'k-')
    %%% deformed config
    ueq = u(edofs);
    theta = elem(q,7); c = cos(theta); s = sin (theta);
    R = [ c s 0;
         -s c 0;
          0 0 1];
    Re = [         R zeros(3,3);
           zeros(3,3)          R];
    Re_= [        Re zeros(6,6);
           zeros(6,6)         Re];
    ueq_ = Re_*ueq;
    uv_ = [xs1 zerv zerv zerv zerv zerv xs2 zerv zerv zerv zerv zerv]*ueq_;
    vv_ = [zerv ys1 zerv zerv zerv ys2 zerv ys3 zerv zerv zerv ys4 ]*ueq_;
    wv_ = [zerv zerv ys1 zerv -ys2 zerv zerv zerv ys3 zerv -ys4 zerv]*ueq_;
    um = R'*[uv_';vv_';wv_'];
```

```
    uv = um(1,:)';
    vv = um(2,:)';
    wv = um(3,:)';
    xd = xu+uv;
    yd = yu+vv;
    zd = zu+wv;
    figure(1)
    plot3(xd,yd,zd,'b-')
    %%% loads
    EA = elem(q,3);
    EIy = elem(q,4);
    EIz = elem(q,5);
    N = EA/L*[-unv zerv zerv zerv zerv zerv unv zerv zerv zerv zerv zerv]*ueq_;
    Mw = EIz*[zerv 6*(2*xiv-unv) zerv zerv zerv 2*L*(3*xiv-2*unv) zerv 6*(unv-2*xiv) ...
    zerv zerv zerv 2*L*(3*xiv-unv)]*ueq_;
    Mv = EIy*[zerv zerv 6*(2*xiv-unv) zerv -2*L*(3*xiv-2*unv) zerv zerv zerv 6*(unv-2*xiv) ...
    zerv -2*L*(3*xiv-unv) zerv]*ueq_;
    Sv = EIz*[zerv 12*unv zerv zerv zerv 6*L*unv zerv -12*unv zerv zerv zerv 6*L*unv]*ueq_;
    Sw = EIy*[zerv zerv 12*unv zerv -6*L*unv zerv zerv zerv -12*unv zerv -6*L*unv zerv]*ueq_;
    figure(1+q)
    hold on
    if q==1000
        plot(xiv(1:255),N(1:255),'k',xiv(1:255),Mv(1:255),'b',xiv(1:255),Mw(1:255),'r',xiv(1:255),Sw(1:255),'m',xiv(1:255),Sv(1:255),'g')
    else
        plot(xiv,N,'k',xiv,Mv,'b',xiv,Mw,'r',xiv,Sw,'m',xiv,Sv,'g')
    end
end
figure(1)
title('Structural displacements','fontsize',15)
legend({'Udeformed cofiguration','Deformed cofiguration'},'Location','southeast')
for q = 1:m
    figure(1+q)
    title('Loads','fontsize',15)
    legend({'N','Mv','Mw','Sw','Sv'})
end
```