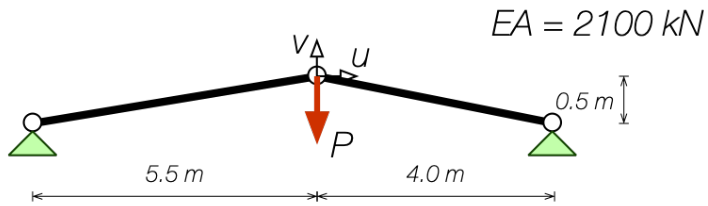
With the last assignment, you explored two simple nonlinear systems. You solved Problem 1-1 using displacement control (in its most primitive), and Problem 1-2 using incremental load control (prescribing the load level).

This week, you'll be exploring path following techniques for tracking equilibrium paths beyond a snap-through[1] or a snap-back[2] point.

Problem 2-1: Displacement control for a two-degrees-of-freedom (2 DOF) problem



This problem expands Problem 1-2 from Assignment #1 such that you shall track the entire equilibrium path from $(u = 0, v = 0)$ and $\lambda = 0$ at least through $\lambda \geq 2.0$.

1. Using Henky strain, a linear relation $\sigma = E\varepsilon$, $A = const.$, and equilibrium on the deformed system, derive the relationship between displacements and forces on the free node. Adjust it for path following as

$$\mathbf{R}(\gamma, \mathbf{u}) = \gamma\bar{\mathbf{P}} - \mathbf{F}(\mathbf{u}(s)) = \mathbf{0} \tag{1a}$$

   with reference load, $\bar{\mathbf{P}} = -(0.99\ \text{kN})\,\mathbf{j}$, load intensity factor γ and displacement $\mathbf{u}$.

   Find the tangents $\partial\mathbf{R}/\partial\gamma$ and $\partial\mathbf{R}/\partial\mathbf{u}$ as you will need them in what follows.

   You should have all the necessary parts for this question from Assignment #1.

2. Develop a, or adjust your existing Newton method to incrementally find γ and $\mathbf{u}$ using displacement control on the vertical displacement, $v = \bar{v}$. The respective constraint equation is

$$g(\mathbf{u}) := \mathbf{e}_v \cdot \mathbf{u} - \bar{v} = 0 \tag{1b}$$

   with

$$\Delta\mathbf{u} \Rightarrow \left\{ \begin{array}{c} u \\ v \end{array} \right\} \quad \text{and} \quad \Delta\mathbf{e}_v \Rightarrow \left\{ \begin{array}{cc} 0 & 1 \end{array} \right\} . \tag{1c}$$

3. Plot load level γ versus vertical displacement v, as well as γ versus horizontal displacement u for the converged points on the equilibrium path. Observe how only one of them actually looks like the curve you (might have) expected.

4. Add a plot u versus v to explore yet another view of the solution path. Try and overlay this curve on a contour plot of $F_x(u, v)$ and on a contour plot for $F_y(u, v)$[3].

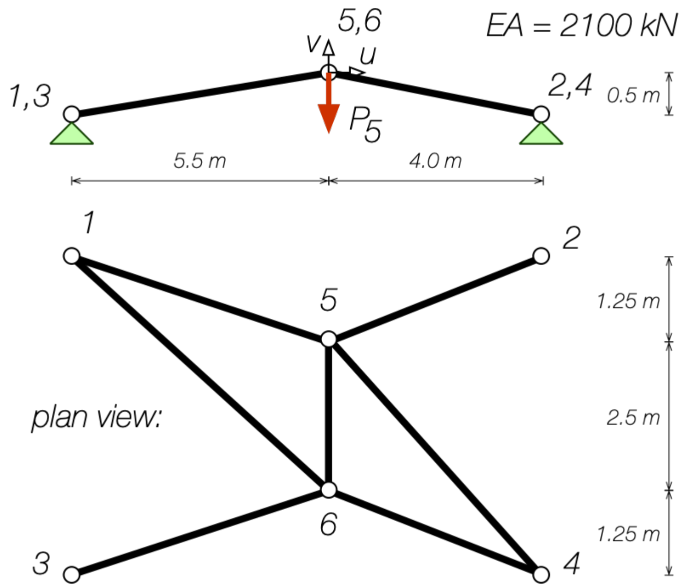
[1] This is where load control will fail.

[2] This is where displacement control will fail.

[3] The first should show that the solution path is hugging the $P_x(u, v) = 0$ contour line. The latter is like a road going through a hilly landscape with elevation along the road being $\lambda(s)\bar{P}$.

Problem 2-2: Going into higher dimensions

Now let's take the problem into the third dimension and look at a 3D-truss system with 6 nodes and 7 truss members as shown.



Only node 5 shall be loaded by a vertical load, $P_5 = \gamma \bar{P}$.

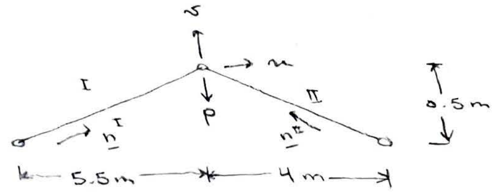
(a) Adjust your code to accommodate the higher dimensional system. At this point, it is highly advisable to drop the brute-force approach and use appropriate functions or my TrussElement class to represent each system component. Assembly may still be done beforehand, though transitioning to a more generic concept will be helpful for future assignments.

(b) Use the vertical displacement $v_5 = \bar{v}$ as control parameter and find the equilibrium path using displacement control. Trace the equilibrium path until members 3–6 and 4–6 are in tension.

(c) Present load-displacement diagrams by plotting the load factor γ against displacement components of nodes 5 and 6.

(d) Plot the planar view of the equilibrium path for nodes 5 and 6 (can be in one or two plots).

①



USING DISPLACEMENT CONTROL TO TRACK EQUILIBRIUM PATH

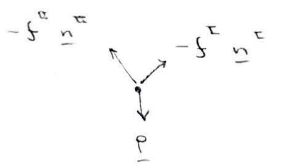
1.1 DERIVE RELATIONSHIP BTW DISPLACEMENTS AND FORCES ON FREE NODES

- HENKEY STRAIN : $\varepsilon = \ln \lambda = \frac{1}{2} \ln\left(\frac{\ell}{L}\right)^2$
- $\sigma = E\varepsilon$
- DEFORMED SYSTEM EQUILIBRIUM
- A = CONST.

INTERNAL FORCE MAGNITUDE :

$$f^I = EA\varepsilon^I \qquad f^{II} = EA\varepsilon^{II}$$

NODAL EQUILIBRIUM :



$$\underline{P} = f^I \underline{n}^I + f^{II} \underline{n}^{II}$$

CONSTRAINT :

↳ CONTROLLING VERTICAL DEFLECTION, $\bar{v}$

$$g(\underline{u}) = \underline{u} \cdot \underline{e}_K - \bar{v} = 0$$

CONSIDER RESIDUAL

$$\underline{R}(\gamma, \underline{u}) = \gamma \bar{\underline{P}} - \underline{F}(\underline{u}) = 0$$

WHERE $\underline{F}(\underline{u}) = f^I \underline{n}^I + f^{II} \underline{n}^{II}$, $\underline{n}^e = \frac{1}{\ell^e}\left[\underline{L}^e + \underline{u}_j^e\right]$

AND $\bar{\underline{P}} = \begin{Bmatrix} 0 \\ -0.99 \end{Bmatrix} \text{kN}$

$$\gamma \underline{P} = \underline{F}(\underline{u})$$

FINDING TANGENTS :

$$\boxed{\frac{\partial \underline{R}}{\partial \gamma} = \bar{\underline{P}}}$$

$$\boxed{\frac{\partial \underline{R}}{\partial \underline{u}} = -\frac{\partial \underline{F}}{\partial \underline{u}} = -\underline{\underline{K}}_T = -\left(\frac{EA}{\ell^I} \underline{n}^I \otimes \underline{n}^I + \frac{f^I}{\ell^I} \underline{\underline{P}}^I\right) - \left(\frac{EA}{\ell^{II}} \underline{n}^{II} \otimes \underline{n}^{II} + \frac{f^{II}}{\ell^{II}} \underline{\underline{P}}^{II}\right)}$$

$$; \underline{\underline{P}}^e = \underline{\underline{I}} - \underline{n}^e \otimes \underline{n}^e$$

① 1.2 DISPLACEMENT CONTROL $\Rightarrow v = \bar{v}$

↳ CONSTRAINT EQUATION: $g(\underline{u}) := \underline{e}_v \cdot \underline{u} - \bar{v} = 0$

$$\Delta \underline{u} = \begin{Bmatrix} u \\ v \end{Bmatrix} \qquad \underline{e}_v = \{0 \;\; 1\}$$

$$\begin{Bmatrix} \underline{R}(\lambda, \underline{u}) \\ g(\underline{u}) \end{Bmatrix} = \begin{bmatrix} \underline{\underline{K}}_T(\underline{u}) & -\underline{\bar{P}} \\ -\underline{e}_v & 0 \end{bmatrix} \begin{Bmatrix} \Delta \underline{u} \\ \Delta \lambda \end{Bmatrix}$$

$$\underline{R}(\lambda, \underline{u}) = \underline{\underline{K}}_T(\underline{u}) \Delta \underline{u} - \underline{\bar{P}} \Delta \lambda$$

$$\Delta \underline{u} = \underline{\underline{K}}_T^{-1} \cdot \underline{R} + \underline{\underline{K}}_T^{-1} \underline{\bar{P}} \Delta \lambda = \Delta \underline{u}_0 + \underline{u}_1 \Delta \lambda$$

$$g(\underline{u}) + \underline{e}_v \cdot \Delta \underline{u} = 0$$

$$g(\underline{u}) + \underline{e}_v \Delta \underline{u}_0 + \underline{e}_v \underline{u}_1 \Delta \lambda = 0$$

↳ $\Delta \lambda = \dfrac{-(g(\underline{u}) + \underline{e}_v \Delta \underline{u}_0)}{\underline{e}_v \underline{u}_1}$

↳ $\Delta \underline{u} = \Delta \underline{u}_0 + \underline{u}_1 \Delta \lambda$

LET $\Delta \underline{q} = \begin{Bmatrix} \Delta \underline{u} \\ \Delta \lambda \end{Bmatrix}$

ALGORITHM:

- INITIALIZE @ $\underline{u} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$

↳ $\underline{R}(\lambda, \underline{u}) = \underline{0}$ ↳ $\bar{v} = -\delta$ (I CHOOSE THIS INCREMENT)

↳ $\Delta \underline{u}_0 = \underline{0}$ ↳ $g(\underline{u}) = \underline{u} \cdot \underline{e}_v - \bar{v} = \delta$

↳ $\underline{u}_1 = \underline{\underline{K}}_T^{-1} \underline{\bar{P}}$

$$\Delta \lambda = \frac{-\delta}{\underline{\underline{K}}_T^{-1} \underline{\bar{P}}} \Rightarrow \Delta \underline{u} = \Delta \underline{u}_0 + \underline{u}_1 \Delta \lambda$$

UPDATE:

↳ DIRECTION VECTORS $\underline{n}^e$ ↳ POSITION, $\underline{u} = \underline{u} + \Delta \underline{u}$

↳ $\underline{\underline{K}}_T(\underline{u})$

↳ $\underline{R} = \underline{\underline{K}}_T(\underline{u}) \Delta \underline{u} - \underline{\bar{P}} \Delta \lambda$

↳ $g(\underline{u}) = -\underline{e}_v \cdot \Delta \underline{u}$

CHECK $\|\tilde{\underline{R}}\|$ VS. TOLERANCE

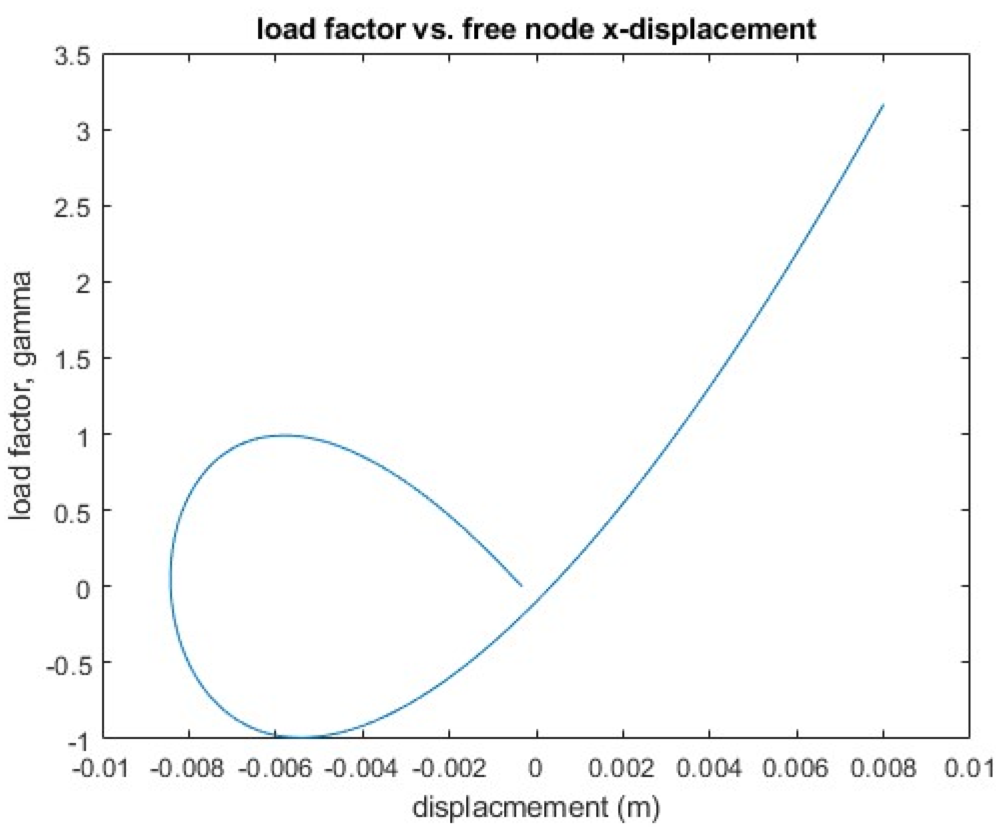
CESG 506 HW2 - NEWTON RAPHSON w/ DISPLACEMENT CONTROL

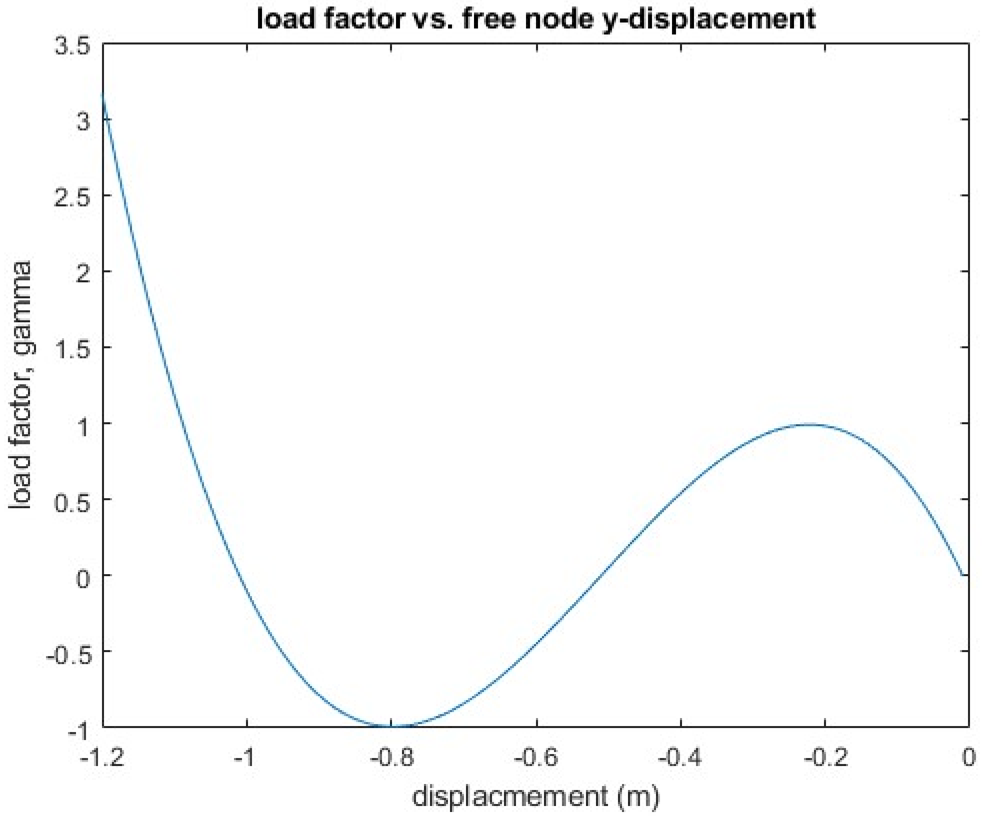
```
clear; clc;

%%-----PROBLEM SPECIFIC PARAMETERS-----%%
EA = 2100; %kN
W1 = 5.5; %m
W2 = 4.0; %m
H = 0.5; %m
L1_vec = [W1,H];
L2_vec = [W1,H]-[W1+W2,0];
L1 = sqrt(dot(L1_vec,L1_vec));
L2 = sqrt(dot(L2_vec,L2_vec));
N1 = L1_vec./L1;
N2 = L2_vec./L2;
Pref = [0; -0.99]; %kN

%%-----ITERATIVE NEWTON RAPHSON ALGORITHM-----%%

%%-----Initialization-----%%
u_bar = [0,0]; %Initially undeformed displacement
gamma = 0; %load factor
k_tan = EA/L1*(N1'*N1) + EA/L2*(N2'*N2); %Initial Tangent Stiffness
ev = [0,1]; % y-direction vector
k_tot = [k_tan, -Pref; -ev,0]; %Appended stiffness matrix

vert = 0; %total vertical deflection
delta = -0.01; %Vertical controlled increment of free node (m)
vert_limit = 1.2;
vert_num = vert_limit/abs(delta);

v_disp = zeros(1,vert_num);
u_disp = zeros(1,vert_num);
load_factor = zeros(1,vert_num);
Fx = zeros(1,vert_num);
Fy = zeros(1,vert_num);

tick = 0
while abs(vert) < vert_limit
    tick = tick + 1;
    load_factor(tick) = gamma;
    v_disp(tick) = delta*tick;
    R = [0;0];
    g = -delta;
    R_tilde = [R;g];
    del_q = k_tot\R_tilde;
    delta_u = [del_q(1),del_q(2)];
    del_gamma = del_q(3);

    tol = 0.001;
```

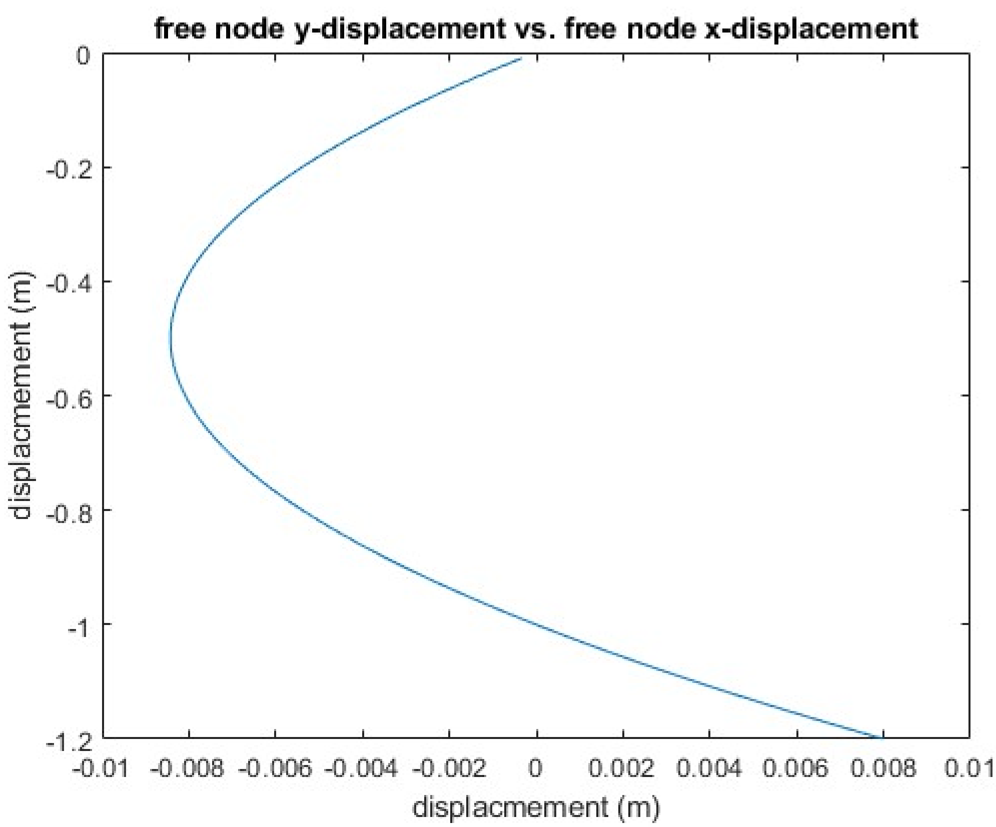
```
    tock = 0;
    error = abs(delta);

    while error > tol
        tock = tock + 1;
        u_bar = u_bar + delta_u;
        gamma = gamma + del_gamma;

        %%----Find Internal Force Vector at Updated Position-----%%
        l1_vec = L1_vec + u_bar;
        l2_vec = L2_vec + u_bar;
        l1_mag = sqrt(dot(l1_vec,l1_vec));
        l2_mag = sqrt(dot(l2_vec,l2_vec));
        n_1 = l1_vec./l1_mag;
        n_2 = l2_vec./l2_mag;
        lambda_1 = l1_mag/L1;
        lambda_2 = l2_mag/L2;
        eps_1 = log(lambda_1);
        eps_2 = log(lambda_2);
        f_1 = EA*eps_1;
        f_2 = EA*eps_2;
        F_int = f_1.*n_1 + f_2.*n_2;

        %%-----Find Residuals-----%%
        R = -F_int' + Pref*gamma;
        g = ev*u_bar' - tick*delta;
        R_tilde = [R;g];
        error = norm(R_tilde);

        %%-----Update Tangent Stiffness-----%%
        k1 = (EA/l1_mag).*(n_1'*n_1) + (f_1/l1_mag).*(eye(2)-
n_1'*n_1);
        k2 = EA/l2_mag.*(n_2'*n_2) + f_2/l2_mag.*(eye(2)-n_2'*n_2);
        k_tan = k1 + k2;
        k_tot = [k_tan, -Pref; -ev,0];

        %%-----Calculate New Displacements-----%%
        del_q = k_tot\R_tilde;
        delta_u = [del_q(1),del_q(2)];
        del_gamma = del_q(3);

        if tock > 25
            break
        else
        end
        u_disp(tick) = u_bar(1);
    end
    vert = vert + delta;
    Fx(tick) = F_int(1);
    Fy(tick) = F_int(2);
end

plot(u_disp,load_factor)
title('load factor vs. free node x-displacement')
```

```
xlabel('displacmement (m)')
ylabel('load factor, gamma')

figure
plot(v_disp,load_factor)
title('load factor vs. free node y-displacement')
xlabel('displacmement (m)')
ylabel('load factor, gamma')

figure
plot(u_disp,v_disp)
title('free node y-displacement vs. free node x-displacement')
xlabel('displacmement (m)')
ylabel('displacmement (m)')
```

```
tick =

     0
```




load factor vs. free node y-displacement
load factor, gamma
displacmement (m)
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0


free node y-displacement vs. free node x-displacement
displacmement (m)
displacmement (m)
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-0.01
-0.008
-0.006
-0.004
-0.002
0
0.002
0.004
0.006
0.008
0.01

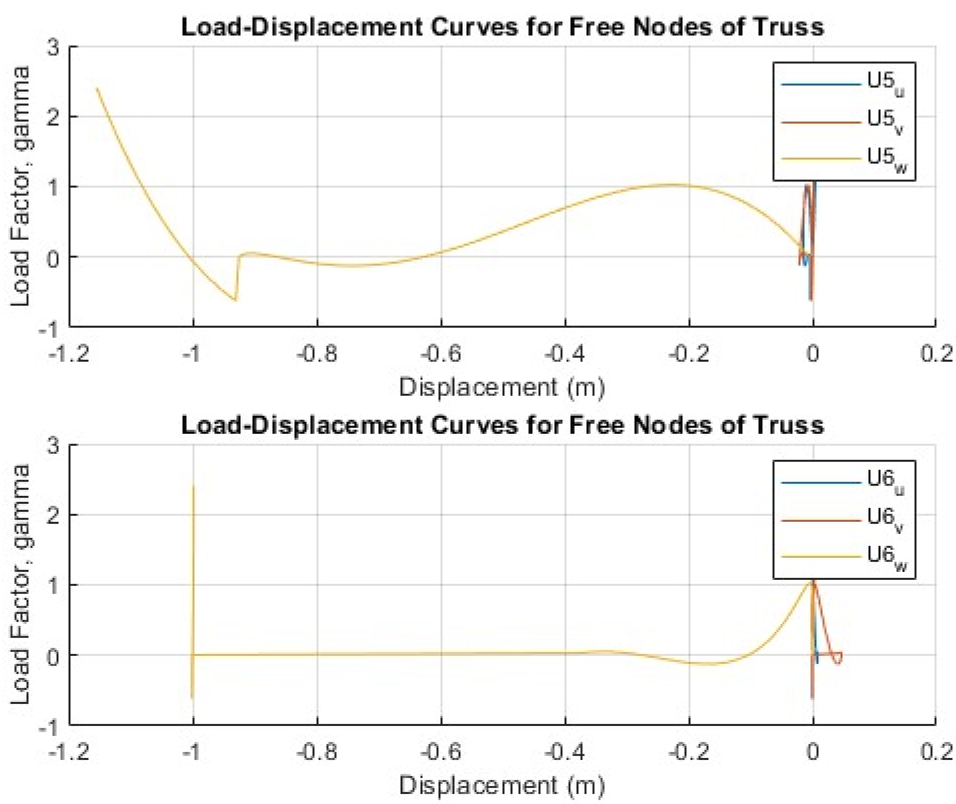
CESG 506 HW2 - DISPLACEMENT CONTROL FRAME SYSTEM

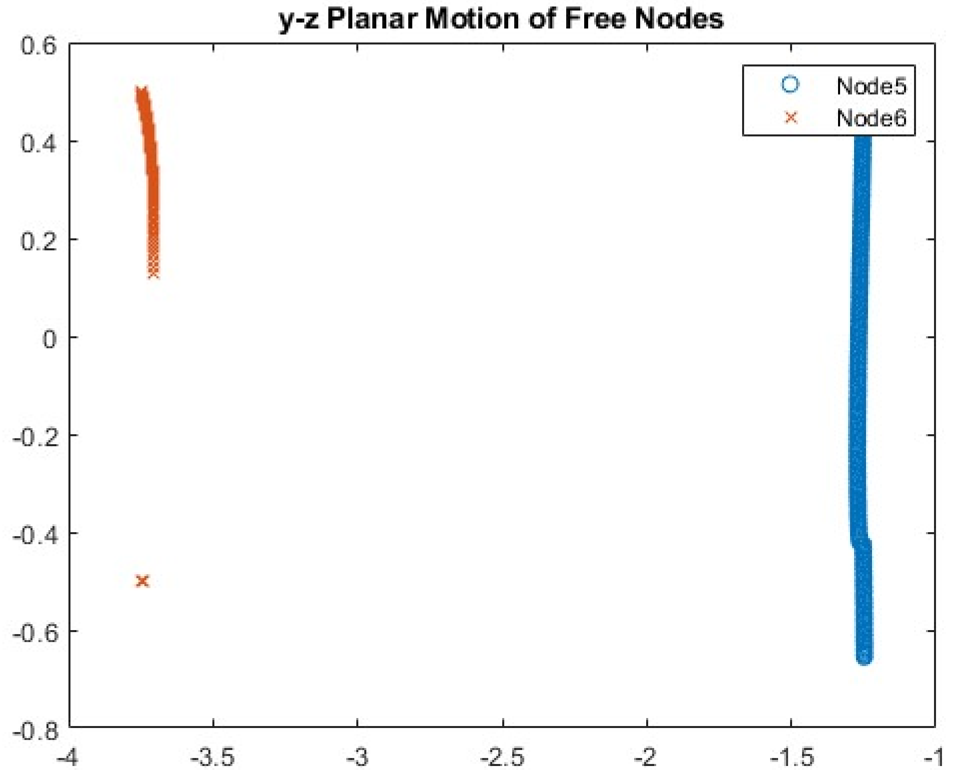
```
clear; clc;
%%-----PROBLEM SPECIFIC PARAMETERS-----%%
EA = 2100; %kN
Pref = [0; 0; -0.99; 0; 0; 0]; %kN
ek = [0,0,1,0,0,0];

%%-----DEFINE SYSTEM LAYOUT-----%%
layout = [0, 0, 0, 0; %defines original nodal layout (x,y,z,fixity)
          9.5, 0, 0, 0; %row m is node_m coordinates
          0, -5, 0, 0;
          9.5, -5, 0, 0;
          5.5, -1.25, 0.5, 1;
          5.5, -3.75, 0.5, 1];

logic = [1,1,5; %defines element connectivity (element#,nodei,nodej)
         2,1,6;
         3,2,5;
         4,3,6;
         5,4,5;
         6,4,6;
         7,5,6];

for i = 1:length(logic)
    Length{i} = layout(logic(i,3),(1:3))-layout(logic(i,2),(1:3));
end

%%-----INITIALIZATION-----%%
disp5 = [0,0,0];
disp6 = [0,0,0];
u_bar = [disp5,disp6];
diff = disp6 - disp5;

%%-----Get Initial Element Stiffnesses-----%%
for m = 1:length(logic) %m = element number
    if logic(m,3)==5
        [F{m},k{m}] = stiffness(EA, Length{m}, disp5, 3);
    elseif logic(m,2)~=5
        [F{m},k{m}] = stiffness(EA, Length{m}, disp6, 3);
    else
        [F{m},k{m}] = stiffness(EA, Length{m}, diff, 3);
    end
end

kff1 = k{1} + k{5} + k{3} + k{7};
kff2 = -k{7}; %Should use fixity values from layout instead of
 hardcoding
kff3 = -k{7};
kff4 = k{2} + k{4} + k{6} + k{7};
```

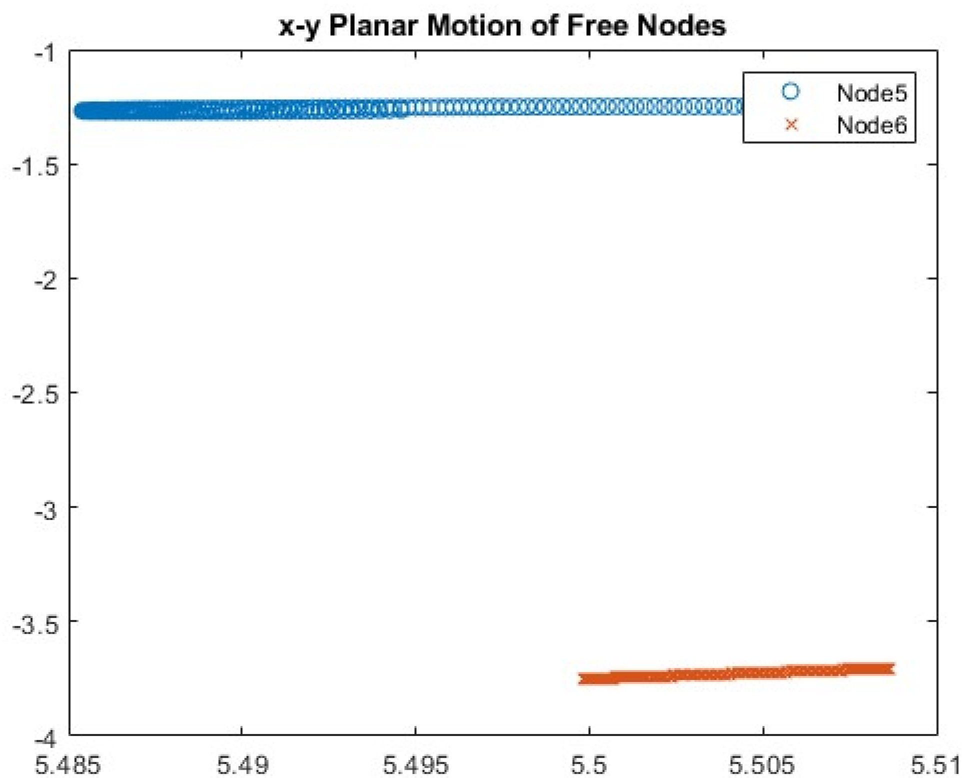
```
k_tan = [kff1,kff2;kff3,kff4];

k_tot = [k_tan,-Pref;-ek,0]; %Appended stiffness matrix

%%----ITERATIVE NEWTON-RAPHSON ALGORITHM-----%%
vert = 0; %total vertical deflection (z-dir)(controlled variable)
delta = -0.005; %Vertical controlled increment of node 5 (m)

u_disp5 = [];
v_disp5 = [];
w_disp5 = [];
u_disp6 = [];
v_disp6 = [];
w_disp6 = [];

load_factor = [];
gamma = 0;
F36 = 0;
F46 = 0;

tick = 0;
while abs(vert) < 1.15
    tick = tick + 1;
    load_factor(tick) = gamma;
    w_disp5(tick) = delta*tick;
    R = [0;0;0;0;0;0];
    g = -delta;
    R_tilde = [R;g];
    del_q = k_tot\R_tilde;
    delta_u = [del_q(1),del_q(2),del_q(3),del_q(4),del_q(5),del_q(6)];
    del_gamma = del_q(length(k_tot));

    tol = 1e-8;
    tock = 0;
    error = abs(delta);

    while error > tol
        tock = tock + 1;
        u_bar = u_bar + delta_u;
        disp5 = [u_bar(1),u_bar(2),u_bar(3)];
        disp6 = [u_bar(4),u_bar(5),u_bar(6)];
        diff = disp6 - disp5;
        gamma = gamma + del_gamma;

        %%----Find Internal Force and stiffness at Updated
 Position-----%%
        F_int = [0,0,0,0,0,0];
        for m = 1:7 %m = element number
            if logic(m,3)==5
                [F{m},k{m}] = stiffness(EA, Length{m}, disp5, 3);
                F_int5 = F{m};
                F_int6 = [0,0,0];
            elseif logic(m,2)~=5
                [F{m},k{m}] = stiffness(EA, Length{m}, disp6, 3);
```

```
                F_int5 = [0,0,0];
                F_int6 = F{m};
            else
                [F{m},k{m}] = stiffness(EA, Length{m}, diff, 3);
                F_int5 = -F{m}; %points opposite of element n vector
                F_int6 = F{m};
            end
            F_int = F_int + [F_int5,F_int6];
            F36 = norm(F{4});
            F46 = norm(F{6});
        end
        kff1 = k{1} + k{2} + k{3} + k{7};
        kff2 = -k{7}; %Should use fixity values from layout instead of
hardcoding
        kff3 = -k{7};
        kff4 = k{2} + k{4} + k{6} + k{7};
        k_tan = [kff1,kff2;kff3,kff4];
        k_tot = [k_tan,-Pref;-ek,0]; %Appended stiffness matrix

        %%-----Find Residuals-----%%
        R = -F_int' + Pref*gamma;
        g = ek*u_bar' - tick*delta;
        R_tilde = [R;g];
        error = norm(R_tilde);

        %%-----Calculate New Displacements-----%%
        del_q = k_tot\R_tilde;
        delta_u =
[del_q(1),del_q(2),del_q(3),del_q(4),del_q(5),del_q(6)];
        del_gamma = del_q(length(k_tot));

        if tock > 1000
            break
        else
        end
    end
    u_disp5(tick) = u_bar(1);
    v_disp5(tick) = u_bar(2);
    u_disp6(tick) = u_bar(4);
    v_disp6(tick) = u_bar(5);
    w_disp6(tick) = u_bar(6);
    vert = vert + delta;
end
subplot(2,1,1)
grid on
hold on
plot(u_disp5,load_factor)
plot(v_disp5,load_factor)
plot(w_disp5,load_factor)
legend('U5_u','U5_v','U5_w')
xlabel('Displacement (m)')
ylabel('Load Factor, gamma')
title('Load-Displacement Curves for Free Nodes of Truss')
subplot(2,1,2)
```

```
grid on
hold on
plot(u_disp6,load_factor)
plot(v_disp6,load_factor)
plot(w_disp6,load_factor)
legend('U6_u','U6_v','U6_w')
title('Load-Displacement Curves for Free Nodes of Truss')
xlabel('Displacement (m)')
ylabel('Load Factor, gamma')

figure
plot(layout(5,2)+v_disp5,layout(5,3)+w_disp5,'o')
hold on
plot(layout(6,2)+v_disp6,layout(6,3)+w_disp6,'x')
legend('Node5','Node6')
title('y-z Planar Motion of Free Nodes')

figure
plot(layout(5,1)+u_disp5,layout(5,2)+v_disp5,'o')
hold on
plot(layout(6,1)+u_disp6,layout(6,2)+v_disp6,'x')
legend('Node5','Node6')
title('x-y Planar Motion of Free Nodes')
```




y-z Planar Motion of Free Nodes
Node5
Node6
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-4
-3.5
-3
-2.5
-2
-1.5
-1


x-y Planar Motion of Free Nodes
Node5
Node6
-1
-1.5
-2
-2.5
-3
-3.5
-4
5.485
5.49
5.495
5.5
5.505
5.51