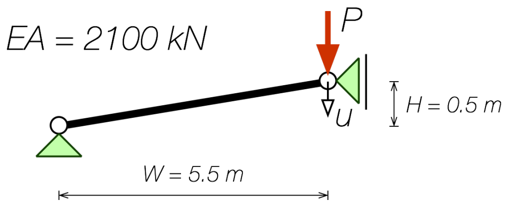
Problem 1-1: Study of different formulations for large deformation problems.



We looked at alternative formulations for strain, the constitutive relation, and the way to formulate equilibrium. For reference, here are the four strain measures:

$$\epsilon = \frac{\ell - L}{L} \tag{1a}$$

$$E = \frac{1}{2}\frac{\ell^2 - L^2}{L^2} \tag{1b}$$

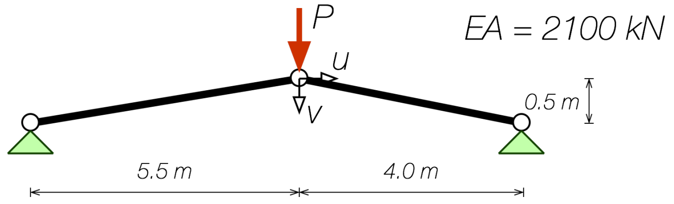
$$\varepsilon = \frac{1}{2}\frac{\ell^2 - L^2}{\ell^2} \tag{1c}$$

$$\tilde{\varepsilon} = \frac{1}{2}\ln\left(\frac{\ell^2}{L^2}\right) \tag{1d}$$

Assume the cross section area, A, does not change during the deformation process.

1. Derive the force-displacement relation for all possible combinations (8 options) of kinematic relations and equilibrium on the undeformed and the deformed configuration and compare them in a single plot for $0 \leq u \leq 2.5\,H$. Include the linear solution in the same plot.
2. Include a brief discussion on why some formulations may be better than others. Which one would you pick if you had to design this structure?

Problem 1-2: Two degree of freedom (2 DOF) problem – Load control.



This problem expands problem 1-1 to a small system with two degrees of freedom, u and v. Due to the non-symmetric nature of the problem, the vertical load P will activate both degrees of freedom.

1. Using Henkey 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. i.e.,

$$\left\{ \begin{array}{c} P_x \\ P_y \end{array} \right\} = \left\{ \begin{array}{c} F_x(u,v) \\ F_y(u,v) \end{array} \right\} \tag{2a}$$

2. Find the tangent stiffness matrix of the system. Keep it simple and do not substitute long expressions into your answer.

3. Formulate the condition $P_x = 0$ (no horizontal load) to the system of equations. Will this allow you to eliminate the horizontal displacement from the system of equations? Why? Why not?

4. Write a simple program that allows you to solve for the displacement vector using the generalized Newton method (also known as Newton-Raphson method) with

$$\left\{ \begin{array}{c} P_x \\ P_y \end{array} \right\} = \lambda \left\{ \begin{array}{c} 0 \\ P_{cr} \end{array} \right\} \tag{2b}$$

   with $\lambda = 0, 0.25, 0.5, 0.75, 0.99, 0.999$. List the residual for each iteration step at each of the load levels in a table.

   Hint: you will need to iterate to find P_{cr} even before solving for the given load levels. You may use your code to work up to that limit or use any other iterative or graphic technique to find that limit.

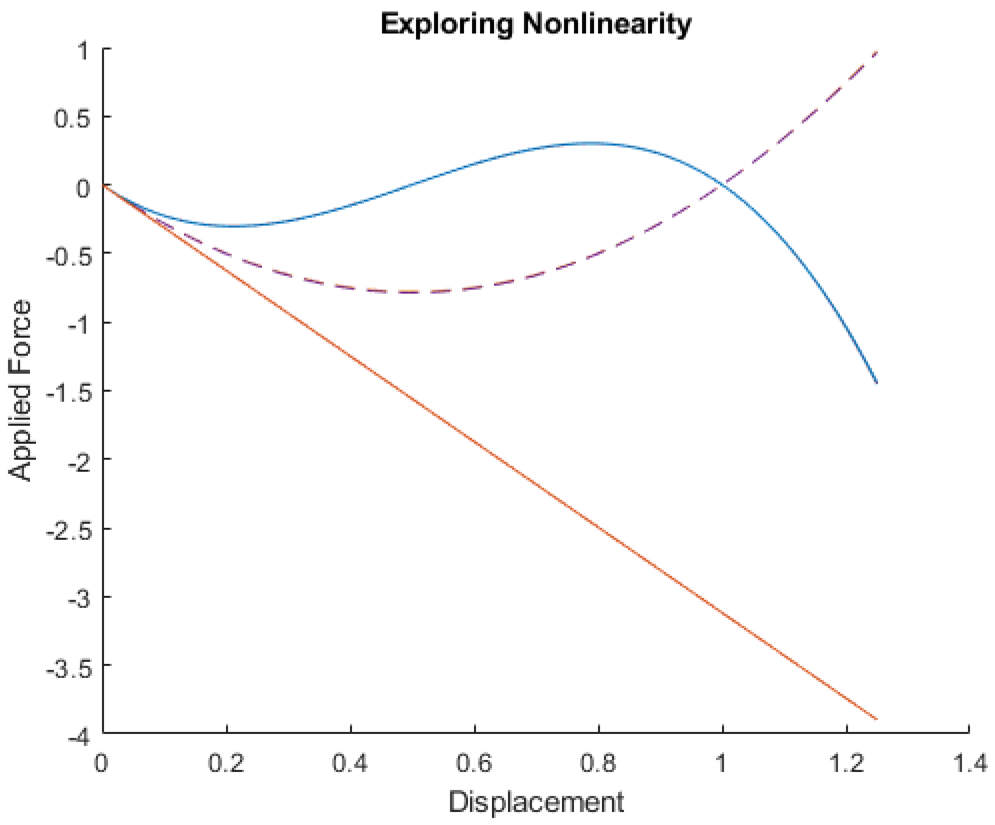
5. To what error limit could you compute the solution? Did that limit depend on the load level?

   Present the relative error for each load level in a table with one row per iteration step and one column per load level. Combine the error versus iteration step for all cases in a single semi-log plot (the relative error goes on the logarithmic axis).

CESG 506 HW1 - DIFFERENT FORMULATIONS FOR LARGE DEFORMATION PROBLEMS

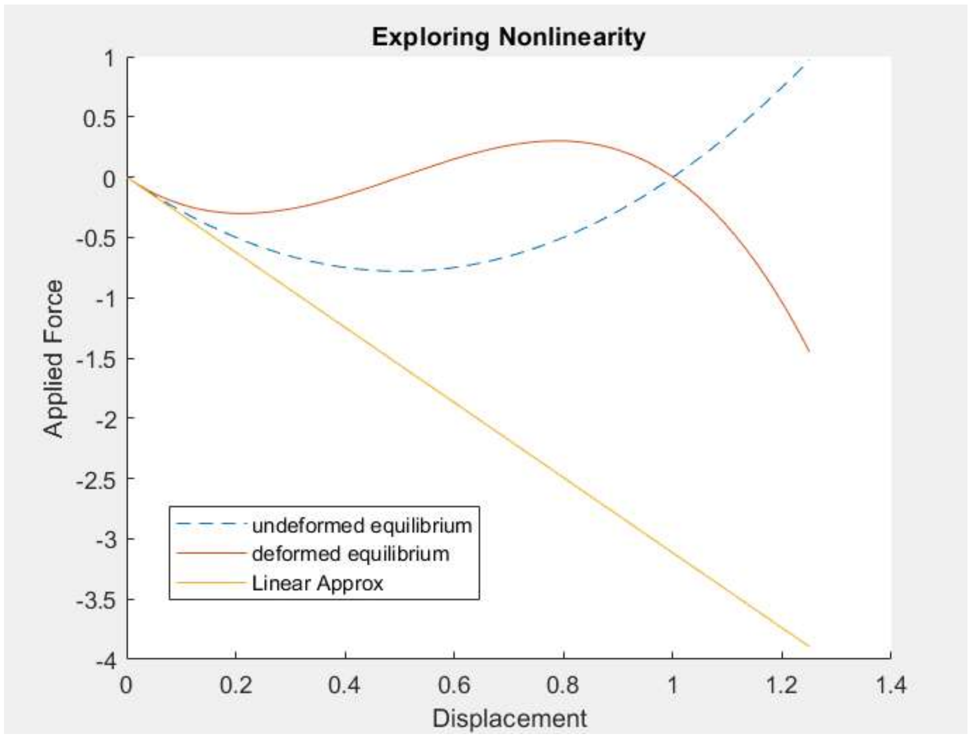
```
clear;clc;
% PROBLEM SPECIFIC PARAMETERS
EA = 2100; %kN (Axial Stiffness)
W = 5.5; %m (Horizontal component of length, fixed)
H = 0.5; %m (Vertical component of length, varies)
L = sqrt(W^2+H^2); %m (Undeformed Length)
num_points = 1500;
u = linspace(0,2.5*H,num_points); %m (Change in vertical component)
l = sqrt(W^2+(H-u).^2); %m (Deformed Length)
N = [cos(atan(H/W));
     sin(atan(H/W))]; %Undeformed direction vector (Normalized)
n = [cos(atan((H-u)/W));
     sin(atan((H-u)/W))]; %Deformed direction vector (Normalized)


%%-----FOUR DIFFERENT STRAIN MEASURES-----%%
% ENGINEERING STRAIN
e = (l-L)/L;
% GREEN-LAGRANGE STRAIN
EE = 1/2*(l.^2-L^2)/L^2;
% HENKEY (NATURAL) STRAIN
eps = 1/2*log(l.^2/L^2);
% ALMANSI STRAIN
eps_a = 1/2*(l.^2-L^2)./l.^2;

%%-----INTERNAL FORCE MAGNITUDES-----%%
f_e = EA.*e;
f_EE = EA.*EE;
f_eps = EA.*eps;
f_eps_a = EA.*eps_a;

%%-----EQUILIBRIUM ON DEFORMED VS. UNDEFORMED CONFIGURATIONS-----%%
% UNDEFORMED CONFIGURATION
fu_e = f_e.*N;
fu_EE = f_EE.*N;
fu_eps = f_eps.*N;
fu_eps_a = f_eps_a.*N;

% DEFORMED CONFIGURATION
fd_e = f_e.*n;
fd_EE = f_EE.*n;
fd_eps = f_eps.*n;
fd_eps_a = f_eps_a.*n;

%%-----PLOTTING-----%%
% UNDEFORMED CONFIGURATION
Pu_e = zeros(1,num_points);
Pu_EE = zeros(1,num_points);
```

```
Pu_eps = zeros(1,num_points);
Pu_eps_a = zeros(1,num_points);

for i = 1:num_points
    Pu_e(i) = fu_e(2,i);
    Pu_EE(i) = fu_EE(2,i);
    Pu_eps(i) = fu_eps(2,i);
    Pu_eps_a(i) = fu_eps_a(2,i);
end

hold on
plot(u,Pu_e,'--')
plot(u,Pu_EE,'--')
plot(u,Pu_eps,'--')
plot(u,Pu_eps_a,'--')

% DEFORMED CONFIGURATION
Pd_e = zeros(1,num_points);
Pd_EE = zeros(1,num_points);
Pd_eps = zeros(1,num_points);
Pd_eps_a = zeros(1,num_points);

for i = 1:num_points %Getting magnitudes of internal forces
    Pd_e(i) = fd_e(2,i);
    Pd_EE(i) = fd_EE(2,i);
    Pd_eps(i) = fd_eps(2,i);
    Pd_eps_a(i) = fd_eps_a(2,i);
end

plot(u,Pd_e)
plot(u,Pd_EE)
plot(u,Pd_eps)
plot(u,Pd_eps_a)

%LINEAR APPROXIMATION
k = EA/L*(N*N');
d{1} = [];
P = [];
for i = 1:num_points
    d{i} = [0 u(i)];
    P(i) = k(2,:)*d{i}';
end

plot(u,-P)
title('Exploring Nonlinearity')
xlabel('Displacement')
ylabel('Applied Force')
% legend('undeformed equilibrium','','','','deformed
 equilibrium','','','','Linear Approx')
```



Published with MATLAB® R2019b

Problem 1 discussion



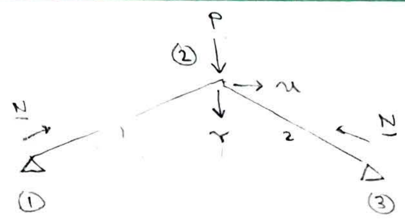
-The linear approximation diverges from the actual nonlinear behavior quickly as seen in the plot above.

- The nonlinear strain formulations coupled with equilibrium on the undeformed sections tell us that after displacements of 2*H, an upwards force will result in a downwards displacement. This seems absurd.

If this structure were expected to deflect less than 0.2 m downwards, I would design using the linear approximation, but for any deflection greater than that I would use nonlinear formulation on the deformed configuration, with Almansi strain because Almansi strain is formulated with respect to the undeformed configuration.

CESG 506 | HW 1 | BILL GALIK

(2)



1. DERIVE RELATIONSHIP BETWEEN DISPLACEMENTS AND FORCES ON NODES

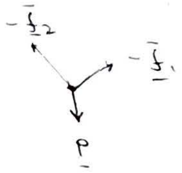
HENKEY STRAIN: $\bar{\varepsilon}_1 = \ln \lambda_1$

$\bar{\varepsilon}_2 = \ln \lambda_2$

LINEAR MATERIAL LAW: $\bar{f}_1 = EA\bar{\varepsilon}_1$

$\bar{f}_2 = EA\bar{\varepsilon}_2$

NODAL EQUILIBRIUM:



$\boxed{\underline{P} = f_1 \underline{n}_1 + f_2 \underline{n}_2}$; $\underline{n}_e = \frac{1}{l_e} \underline{l}_e = \frac{1}{l_e}\left[\underline{L}_e + (\underline{u}_j)_e - (\underline{u}_i)_e\right]$

$(\underline{u}_i)_e = \underline{0}$

$\underline{n}_e = \frac{1}{l_e}\left[\underline{L}_e + \underline{u}_{j e}\right]$

2. FIND TANGENT STIFFNESS MATRIX

$\underline{R}(u) = \underline{R}(u_0) + \frac{d\underline{R}}{du}\Delta \underline{u}$; $\frac{d\underline{R}}{d\underline{u}} = -\frac{d\underline{f}}{d\underline{u}} = -\underline{\underline{K}}$ (TANGENT STIFFNESS)

$[\underline{K}^e] = \begin{bmatrix} k^e & -k^e \\ -k^e & k^e \end{bmatrix}$ $\quad k^e = \frac{EA}{l} n \otimes n + \frac{f}{l} P_n$

$P_n = \underline{I} - n \otimes n$

$\boxed{k^1 = \frac{EA}{l_1} n_1 \otimes n_1 + \frac{f_1}{l_1} P_1 \quad \text{AND} \quad k^2 = \frac{EA}{l_2} n_2 \otimes n_2 + \frac{f_2}{f_2} P_2}$

SYSTEM STIFFNESS

$$\begin{Bmatrix} \underline{P}_1 \\ \underline{P}_2 \\ \underline{P}_3 \end{Bmatrix} = \begin{bmatrix} k^1 & -k^1 & 0 \\ -k^1 & \boxed{k^1 + k^2} & -k^2 \\ 0 & -k^2 & k^2 \end{bmatrix} \begin{Bmatrix} \underline{u}_1 \\ \underline{u}_2 \\ \underline{u}_3 \end{Bmatrix}$$

TANGENT STIFFNESS

$\underline{P}_2 = [\underline{k}^1 + \underline{k}^2]\,\underline{u}_2$ $\quad \boxed{\underline{\underline{K}}^T = \underline{k}^1 + \underline{k}^2}$

② CONT'D

$\begin{Bmatrix} n_x \\ n_y \end{Bmatrix} \{ n_x \quad n_y \}$

3. CONSIDERING THAT $P_x = 0$

$n_y n_x$

$$\begin{Bmatrix} 0 \\ -P \end{Bmatrix} = \begin{bmatrix} K_{11}^T & K_{12}^T \\ K_{21}^T & K_{22}^T \end{bmatrix} \begin{Bmatrix} u \\ -v \end{Bmatrix}$$

OFF DIAGONAL TERMS OF ELEMENTAL STIFFNESS

$$\frac{EA}{l_e} \underline{n}_e \otimes \underline{n}_e - \frac{\bar{f}_e}{l_e} \underline{n}_e \otimes \underline{n}_e \qquad \bar{f}_e = EA\bar{\varepsilon}_e$$

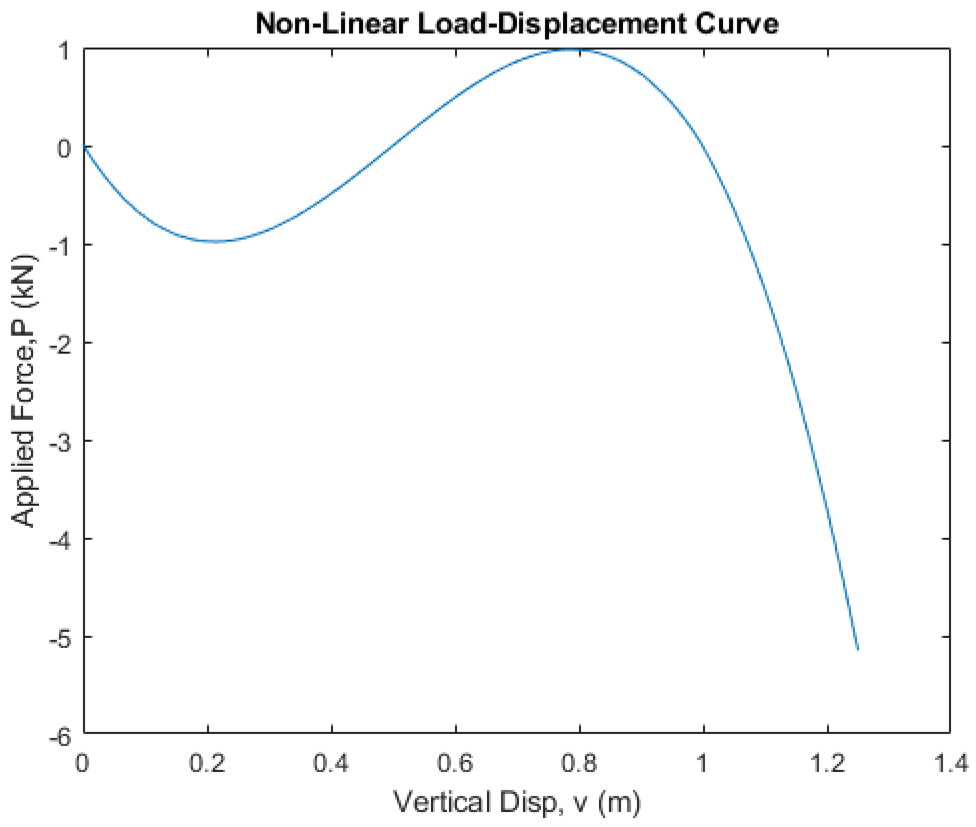
$$K_{21}^T = \Sigma_e \left[\frac{EA}{l_e} (1 - \bar{\varepsilon}_e) n_{e_y} n_{e_x} \right]$$

TO ELIMINATE HORIZONTAL COMPONENT OF DISPLACEMENT REQUIRES THAT $\bar{\varepsilon}_e = 1$

CESG 506 HW1 - SNAP-THROUGH LOAD

```
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;

%%-----DISPLACEMENT CONTROL (SOLVING FREE NODE TRAJECTORY-----%%
v = linspace(0,2.5*H,100); %Controlling downwards displacement
u_storage = zeros(1,100); %Will be used to store indices of u-values
 that satisfy equilibrium. One associated with each v disp.
F_int_x = zeros(100,250);
F_int_x(:,1) = 1; %Assign any positive value to the first column. Used
 to give initial sign value of F_x in horizontal disp loops
for i = 1:100 %loop through y displacements
    for j = 2:250  %loop through x-displacements
        u = j/5000;
        u_bar = [-u,-v(i)]; %displacement of free node. Down due to
 applied force. Left to maintain x-direction equil'b
        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;
        F_int_x(i,j) = F_int(:,1);

        if F_int_x(i,j)*F_int_x(i,j-1) < 0
            u_storage(i) = -u; %Finding and storing the u
 displacements associated with v(i)
        else
        end

    end
end
```

```
%%-----SOLVING AND PLOTTING INTERNAL FORCE (y) FOR FREE NODE
 TRAJECTORY-----%%
FF_int_y = zeros(1,100);
for ii = 1:100
    vv = v(ii); %has opposite sign
    uu = u_storage(ii); %already has correct sign
    uu_bar = [uu, -vv]; %signs both correct
    el1_vec = L1_vec + uu_bar;
    el2_vec = L2_vec + uu_bar;
    el1_mag = sqrt(dot(el1_vec,el1_vec));
    el2_mag = sqrt(dot(el2_vec,el2_vec));
    nn_1 = el1_vec./el1_mag;
    nn_2 = el2_vec./el2_mag;
    llambda_1 = el1_mag/L1;
    llambda_2 = el2_mag/L2;
    eeps_1 = log(llambda_1);
    eeps_2 = log(llambda_2);
    ff_1 = EA*eeps_1;
    ff_2 = EA*eeps_2;
    FF_int = ff_1.*nn_1 + ff_2.*nn_2;
    FF_int_y(ii) = FF_int(:,2);
end

plot(v,FF_int_y)
xlabel('Vertical Disp, v (m)')
ylabel('Applied Force,P (kN)')
title('Non-Linear Load-Displacement Curve')
```



Published with MATLAB® R2019b

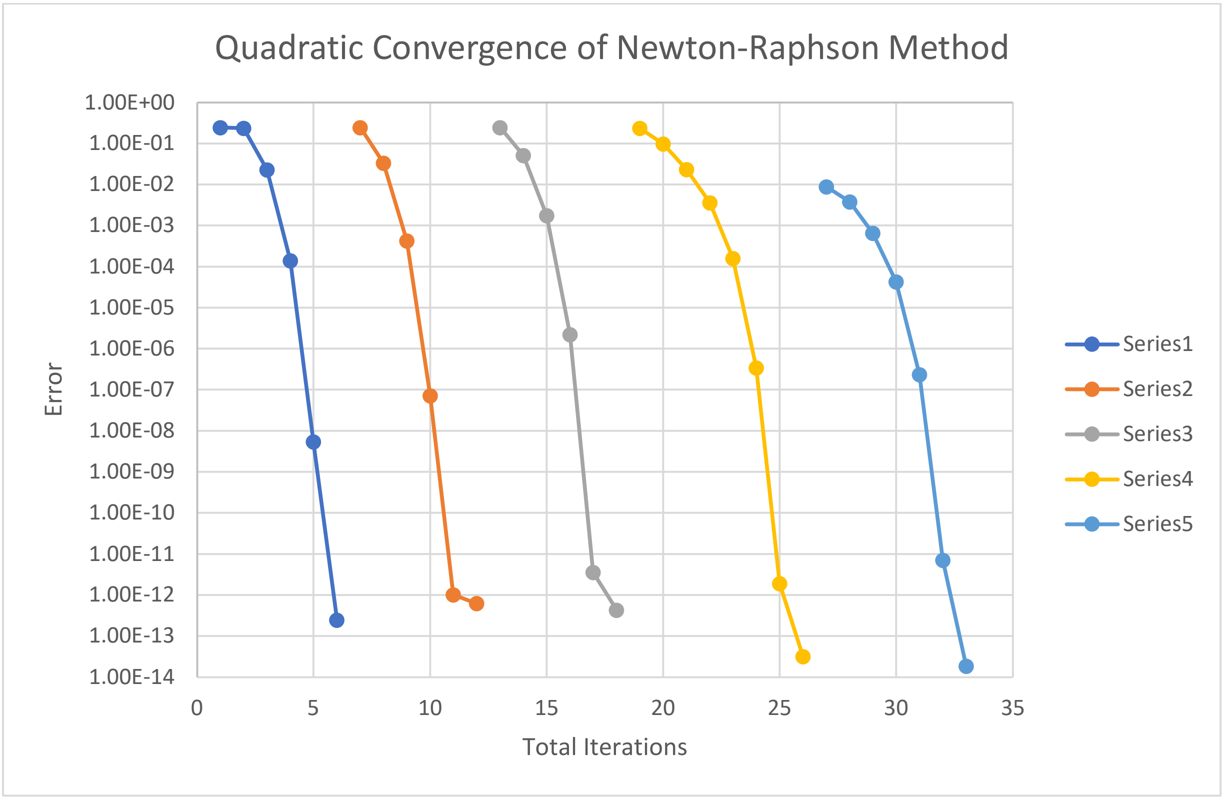
CESG 506 HW1 - NEWTON RAPHSON ITER-ATION METHOD

```
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;
Pcr = [0; 0.9817]; %kN
gamma = [0,0.25,0.5,0.75,0.99,0.999]; %load factors for iterative NR
 method

%%-----ITERATIVE NEWTON RAPHSON ALGORITHM-----%%
Residual = [];
for i = 1:(length(gamma)-1)
    if i == 1 %starting iteration at undeformed configuration
        u_bar = [0,0];
        ticker = 0;
        R = gamma(2)*Pcr; %for zero displacement, internal force = 0
        Residual(ticker+1,i) = norm(R);
        k_init = EA/L1*(L1_vec'*L1_vec) + EA/
L2*(L2_vec'*L2_vec); %note that 2nd term absent b/c int force = 0 for
 no init disp
        h = -inv(k_init)*R;
    else
        ticker = 0;
        k_init = k_tan; %k_tan calculated in first while loop
        R = gamma(i+1)*Pcr + F_int';
        Residual(ticker+1,i) = norm(R);
        h = -inv(k_init)*R;
    end
tol = 1e-12;
while norm(R) > tol
    ticker = ticker + 1;
    %%-----UPDATE POSTION VECTOR-----%%
    u_bar = u_bar + h';

    %%----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;

    %%-----CALCULATE RESIDUAL FORCE FOR UPDATED POSITION-----%%
    R = gamma(i+1)*Pcr + F_int';
    Residual(ticker+1,i) = norm(R);
    error = norm(R);

    %%-----CALCULATE UPDATED 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;

    %%-----CALCULATE NEW DISPLACEMENT DELTA (h)-----%%
    h = -inv(k_tan)*R;

end
% formatSpec = 'Gamma is %3.2f \n';
% fprintf(formatSpec,gamma)
line = '\n Displacement ';
number = 'at load level %1.0f ';
is = 'is: \n';
print = '%6.5f \n';
fprintf(line)
fprintf(number,i+1)
fprintf(is)
fprintf(print,u_bar)
end

fprintf('\n TABLE OF RESIDUALS: \n')
T = array2table(Residual);
disp(T)


 Displacement at load level 2 is:
-0.00086
-0.02623

 Displacement at load level 3 is:
-0.00184
-0.05806

 Displacement at load level 4 is:
-0.00305
-0.10087

 Displacement at load level 5 is:
-0.00515
```

```
-0.18871

 Displacement at load level 6 is:
-0.00547
-0.20452

 TABLE OF RESIDUALS:
    Residual1     Residual2     Residual3     Residual4     Residual5 
    __________    __________    __________    __________    __________

       0.24543       0.24542       0.24542       0.23561     0.0088353
       0.23561      0.033573      0.051429      0.097843     0.0038286
      0.022902    0.00042631     0.0017481       0.02329    0.00065872
    0.00013951    7.1409e-08    2.2122e-06     0.0036042    4.2768e-05
    5.3708e-09    1.0063e-12    3.5624e-12    0.00015697     2.331e-07
    2.4475e-13    6.2196e-13    4.3262e-13    3.4298e-07    7.0848e-12
             0             0             0    1.9024e-12    1.8664e-14
             0             0             0    3.1838e-14             0
```

Published with MATLAB® R2019b

Problem 2 – Part 5 Discussion



Note that the value of Pcr I used in my matlab iterative code was accurate to four decimals.

For each load step, the iterative NR method had no issues attaining a solution accuracy within 1e-12. However, when I decreased the tolerance to 1e-15, the method could not converge for any of the load steps. The shallow slopes in the figure below shows where the algorithm stalled around 1e-13.

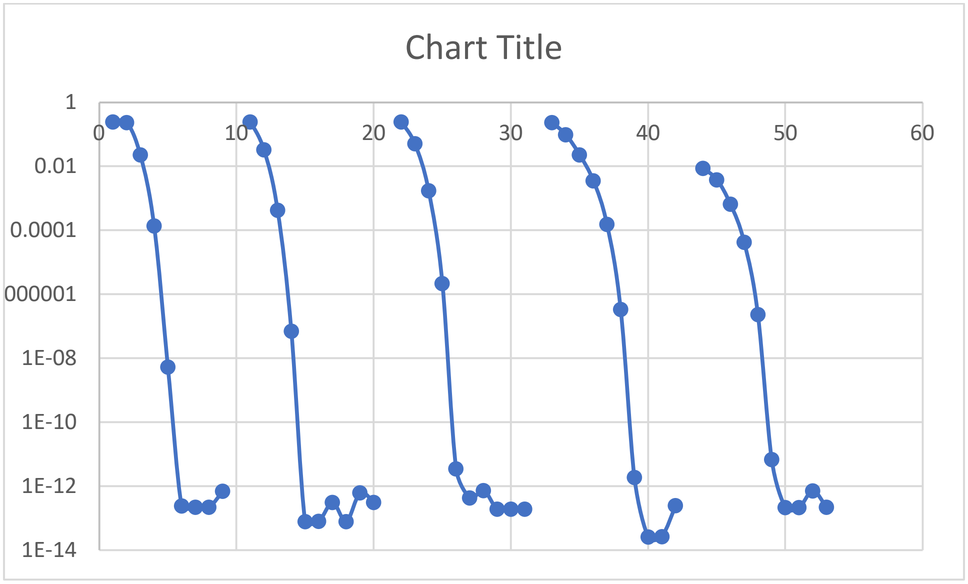


Chart Title
1E-08
1E-10
1E-12
1E-14