CEE 576: Nonlinear Finite Elements - Fall 2016
Homework 1

Bhavesh Shrimali (NetID: bshrima2)

September 9, 2016

Solutions

Solution 1:

The given sets of parameters are used to calculate the number of steps required for the Newton Raphson to converge to an error value of 0.036 or less, starting from an initial error of 0.9:

- (a) c = 1.0, k = 1.2 : Takes 20 steps
- (b) c = 0.5, k = 1.0 : Takes 11 steps

Table 1: Convergence of the Newton Raphson Method

No of Step	Case (a)	Case (b)
1	0.9	0.9
2	0.881234	0.63
3	0.85923	0.441
4	0.833549	0.3087
5	0.803743	0.21609
6	0.76938	0.151263
7	0.730077	0.105884
8	0.685556	0.074119
9	0.635698	0.051883
10	0.580633	0.036318
11	0.520813	0.025423
12	0.457107	
13	0.39086	
14	0.323911	
15	0.25853	
16	0.197248	
17	0.142566	
18	0.096565	
19	0.060504	
20	0.034525	

Problem 2:

For problem 2 we assume isoparametric mapping such that

$$x = \sum_{a=1}^{N} N_a^e(\zeta) \;\; x_a^e$$

thus

$$J = \frac{\partial x}{\partial \zeta} = \sum_{a=1}^{N} N_{a,\zeta}^e(\zeta) \;\; x_a^e$$

Due to the cumbersome nature of the calculations for any arbitrary x_1^e, x_3^e and x_3^e Matlab is used to simplify them and later, for reference, calculations are shown for some specific values of x_i^e.

Matlab Code for part (c)

Attached overleaf. Several comments are worthwhile making here. $\mathbf{K1_i}$ and $\mathbf{K2_i}$ are respectively the asymmetric and the symmetric part of the consistent tangent. In the code these refer to just the numerical values of the computed matrices. Thus to summarize:

- $\mathbf{K1_1} = \frac{\partial K}{\partial u}(\zeta_1)\mathbf{K1_1}(\mathbf{code})$
- $\mathbf{K1_2} = \frac{\partial K}{\partial u}(\zeta_2)\mathbf{K1_2}(\mathbf{code})$
- $\mathbf{K2_1} = \frac{\partial K}{\partial u}(\zeta_1)\mathbf{K2_1}(\mathbf{code})$
- $\mathbf{K2_2} = \frac{\partial K}{\partial u}(\zeta_2)\mathbf{K2_2}(\mathbf{code})$

where

$$\zeta_1 = \frac{-1}{\sqrt{3}} \quad \text{and} \quad \zeta_2 = \frac{1}{\sqrt{3}}$$

```
%Nonlinear Finite Element Method : Fall 2016
clear all; close all; clc;
syms xe1 he xe2 xe3 xc de1 de2 de3 real;

Jacb = 0.5*xe1*(2*xc-1) - 2*xc*xe2 + 0.5*xe3*(2*xc+1);
N1 = xc*(xc-1)/2;
N2 = (1-xc)*(1+xc);
N3 = xc*(xc+1)/2;

N = [N1*N1 N1*N2 N1*N3;
     N1*N2 N2*N2 N2*N3;
     N1*N3 N2*N3 N3*N3];

 N = N*Jacb;
 Fext_matrix = simplify(subs(N,xc,-(3)^(-0.5)) + subs(N,xc,(3)^(-0.5)))
 % The above is true, in general, for any given values of xe1, xe2, xe3
 % However numerical evaluation is presented corresponding to the following
 % values

 J_1 = 1/Jacb;
 Na = [diff(N1,xc);diff(N2,xc);diff(N3,xc)];
 Na =simplify(Na);
```

```
Nb = [N1 N2 N3];

d = Na' * [de1;de2;de3];
d = simplify(d);
K_matrix1 = J_1 * Na * Nb * d;
K_matrix1 = simplify(K_matrix1);

K_matrix2 = J_1 * (Na) * (Na)';

xc = -(3)^(-0.5);
K1_1 = simplify(subs(K_matrix1));
K2_1 = simplify(subs(K_matrix2));

xc = (3)^(-0.5);
K1_2 = simplify(subs(K_matrix1));
K2_2 = simplify(subs(K_matrix2));

% Sample Calculations for some specific values of xe_i
xe1 = 0;
xe2 = he/2;
xe3 = he;

Fext_matrix = simplify(subs(Fext_matrix));
K1_1 = simplify(subs(K1_1))
K1_2 = simplify(subs(K1_2))
K2_1 = simplify(subs(K2_1))
K2_2 = simplify(subs(K2_2))
```

Solution 3:

Part A: Newton Raphson

The algorithm, step-wise, of the Newton Raphson method is described as follows:

- The function, in this case the absolute difference between the external and internal force, or the residual is determined at the start of the problem. We take input all the required parameters, such as the tolerance, step size, number of steps etc.
- The external force is discretized in a number of steps as desired.
- For each step, an iteration counter is initialized, which keeps a track of the number of iterations inside of each step
- For each iteration, we solve the corresponding linearized problem and obtain Δd_n. The displacement at each iteration is then updated as
$$d_n^i = d_n^i + \Delta d_n^i$$
It is important to note that the slope is also updated in the Newton Raphson algorithm at each iteration.
- Residual Check is performed to verify if the residual is within the specified tolerance
- If the check is satisfactory, then we proceed as usual with the next step
- If, however, the check is not satisfied then we iterate further.
- Move from step 1 through n and plot the value of force at each step.

Part A: Modified Newton Raphson

The algorithm, step-wise, of the Modified Newton Raphson method is described as follows:

- The function, in this case the absolute difference between the external and internal force, or the residual is determined at the start of the problem. We take input all the required parameters, such as the tolerance, step size, number of steps etc.
- The external force is discretized in a number of steps as desired.
- For each step, an iteration counter is initialized, which keeps a track of the number of iterations inside of each step. Outside of the iteration loop the slope is calculated corresponding to the first value of $d_n^i \mid_{i=1}$
- For each iteration, we solve the corresponding linearized problem and obtain Δd_n. The displacement at each iteration is then updated as

  $$d_n^i = d_n^i + \Delta d_n^i$$

  It is important to note that the slope is **NOT** updated in the Modified Newton Raphson algorithm at each iteration.
- Residual Check is performed to verify if the residual is within the specified tolerance
- If the check is satisfactory, then we proceed as usual with the next step
- If, however, the check is not satisfied then we iterate further.
- Move from step 1 through n and plot the value of force at each step.

Matlab Code: Newton Raphson

```
%Nonlinear Finite Element Method : Fall 2016

% HW Assignment #1
% Problem #3
%Date: 09/02/2016

syms x;
N = (0.19*x^3 -2*x^2 + 6*x)*exp(0.02*x);
slope = diff(N,x);
Fint = double(subs(N,0));
Fext = 0.5:0.5:5.5;
count = 1;
epsilon = 1e-12;
dn = 0;
y = linspace(0,7,100);
displacement = zeros (length(Fext),1);
Internal_force = zeros (length(Fext),1);

for i=1:length(Fext)
    Res_temp = abs(double(Fext(i)-Fint));
    del_d_temp = Res_temp/double(subs(slope,dn));
    dn = dn+del_d_temp;
    Fint = double(subs(N,dn));

    Res(i) = abs(double(Fext(i)-Fint));
    j =1;
```

```
    Res_int = Res(i)
    Resi = Res_int
    while Resi  >  (epsilon)*Res_int
        j
        Residual(j,i) = Resi;
        s = double(subs(slope,dn));
        del_d = double(Resi/s);
        dn = dn+del_d;
        F_int(j,1) = double(subs(N,dn));
        Resi = double(Fext(i) - F_int(j,1))
        Residual(j+1,i) = Resi;
        j=j+1;
    end
    displacement(i,1) = dn;
    Internal_force(i,1) = F_int(j-1,1);
    Fint = double(subs(N,dn));
end

N_plot = double(subs(N,y));
plot(y,N_plot,displacement,Internal_force,'o');
grid on;
xlabel('Displacement (d)');
ylabel('Internal Force (F_{int})');
legend('Actual Curve','Obtained Curve');
title('F_{int} vs d (Newton Raphson)');
filename = 'NR.xlsx';
xlswrite(filename,Residual);
Fall = [displacement Internal_force];
filename = 'NRFint.xlsx';
xlswrite(filename,Fall);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Matlab Code: Modified Newton Raphson

```
%Nonlinear Finite Element Method : Fall 2016

% HW Assignment #1
% Problem #3
%Date: 09/02/2016

syms x;
N = (0.19*x^3 -2*x^2 + 6*x)*exp(0.02*x);
slope = diff(N,x);
Fint = double(subs(N,0));
Fext = 0.5:0.5:5.5;
count = 1;
epsilon = 1e-12;
dn = 0;
y = linspace(0,7,100);
displacement = zeros (length(Fext),1);
Internal_force = zeros (length(Fext),1);
```

```
for i=1:length(Fext)
    Res_temp = abs(double(Fext(i)-Fint));
    s = double(subs(slope,dn));
    del_d_temp = Res_temp/s;
    dn = dn+del_d_temp;
    Fint = double(subs(N,dn));

    Res(i) = abs(double(Fext(i)-Fint));
    j =1;
    Res_int = Res(i)
    Resi = Res_int
    while Resi > (epsilon)*Res_int
        Residual(j,i) = Resi;
        del_d = double(Resi/s);
        dn = dn+del_d;
        F_int(j,1) = double(subs(N,dn));
        Resi = double(Fext(i) - F_int(j,1))
        Residual(j+1,i) = Resi;
        j=j+1;

    end
    displacement(i,1) = dn;
    Internal_force(i,1) = F_int(j-1,1);
    Fint = double(subs(N,dn));
end
N_plot = double(subs(N,y));
plot(y,N_plot,displacement,Internal_force,'o');
grid on;
xlabel('Displacement (d)');
ylabel('Internal Force (F_{int})');
legend('Actual Curve','Obtained Curve');
title('F_{int} vs d (Newton Raphson)');
filename = 'MNR.xlsx';
xlswrite(filename,Residual);
Fall = [displacement Internal_force];
filename = 'MNRFint.xlsx';
xlswrite(filename,Fall);
```

Force Displacement: Newton Raphson

displacement	Internal Force
0.085614154	0.5
0.176262953	1
0.272804143	1.5
0.37636149	2
0.488455047	2.5
0.611226311	3
0.747858955	3.5
0.903452692	4
1.087138521	4.5
1.318638855	5
1.662398726	5.5

Table 2: Force-Displacement for Newton Raphson

Force Displacement: Modified Newton Raphson

displacement	Internal Force
0.085614154	0.5
0.176262953	1
0.272804143	1.5
0.37636149	2
0.488455047	2.5
0.611226311	3
0.747858955	3.5
0.903452692	4
1.087138521	4.5
1.318638855	5
1.662398726	5.5

Table 3: Force-Displacement for Modified Newton Raphson

Residual Tables: Modified Newton Raphson

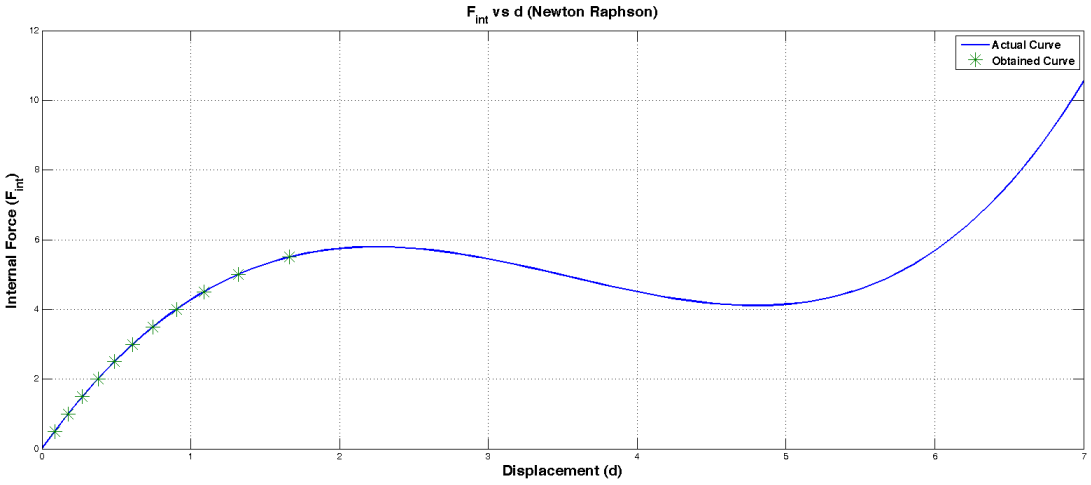
Table 4: Modified Newton Raphson

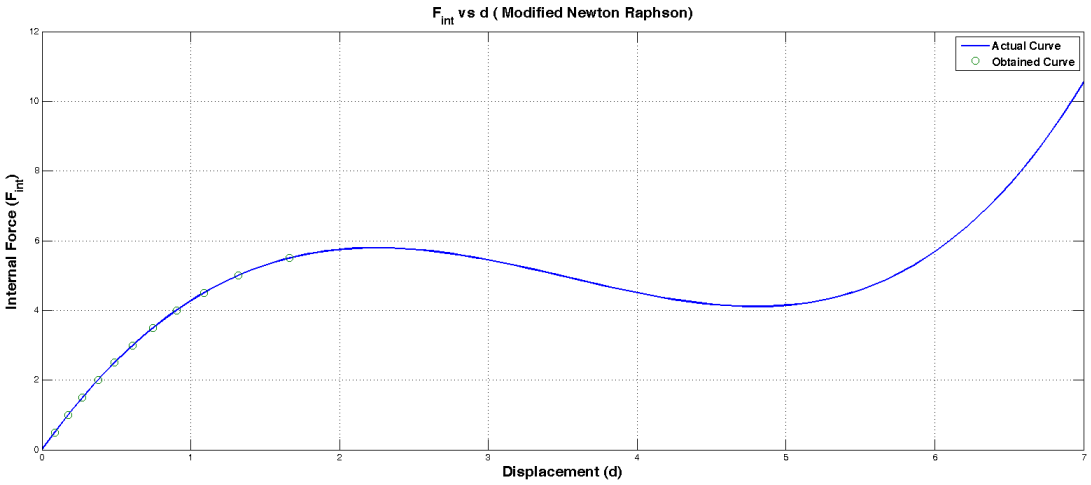
i	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10	Step 11
1	0.013	0.014	0.0156	0.0173	0.0195	0.022	0.026	0.032	0.04	0.0546	0.0865
2	7E-04	8E-04	0.001	0.0012	0.0015	0.002	0.003	0.004	0.007	0.0124	0.0317
3	4E-05	5E-05	6E-05	9E-05	0.0001	2E-04	3E-04	5E-04	0.001	0.003	0.0128
4	2E-06	3E-06	4E-06	6E-06	1E-05	2E-05	3E-05	7E-05	2E-04	0.0007	0.0053
5	1E-07	2E-07	3E-07	4E-07	8E-07	2E-06	4E-06	1E-05	3E-05	0.0002	0.0023
6	5E-09	9E-09	2E-08	3E-08	7E-08	2E-07	4E-07	1E-06	6E-06	4E-05	0.001
7	3E-10	5E-10	1E-09	2E-09	5E-09	1E-08	5E-08	2E-07	1E-06	1E-05	0.0004
8	2E-11	3E-11	7E-11	2E-10	4E-10	1E-09	5E-09	2E-08	2E-07	3E-06	0.0002
9	8E-13	2E-12	4E-12	1E-11	4E-11	1E-10	6E-10	3E-09	3E-08	6E-07	7E-05
10	4E-14	1E-13	3E-13	8E-13	3E-12	1E-11	6E-11	4E-10	5E-09	2E-07	3E-05
11	2E-15	6E-15	2E-14	6E-14	2E-13	1E-12	7E-12	6E-11	9E-10	4E-08	1E-05
12			1E-15	4E-15	2E-14	1E-13	7E-13	8E-12	2E-10	9E-09	6E-06
13						9E-15	8E-14	1E-12	3E-11	2E-09	2E-06
14							9E-15	1E-13	5E-12	6E-10	1E-06
15								2E-14	8E-13	1E-10	5E-07
16									1E-13	3E-11	2E-07
17									2E-14	8E-12	8E-08
18										2E-12	4E-08
19										5E-13	2E-08
20										1E-13	6E-09
21										3E-14	3E-09
22											1E-09
23											5E-10
24											2E-10
25											9E-11
26											4E-11
27											2E-11
28											7E-12
29											3E-12
30											1E-12
31											6E-13
32											2E-13
33											1E-13
34											4E-14

Table 5: Newton Raphson

i	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10	Step 11
1	0.013	0.014	0.016	0.017	0.02	0.022	0.026	0.032	0.04	0.0546	0.0865
2	1E-05	1E-05	2E-05	2E-05	3E-05	5E-05	9E-05	2E-04	3E-04	0.001	0.0054
3	5E-12	1E-11	2E-11	4E-11	1E-10	3E-10	1E-09	4E-09	3E-08	4E-07	3E-05
4			2E-16						-0	5E-14	7E-10

Plots





Residual Reduction: Modified Newton Raphson

Table 6: Residual Reduction in Modified Newton Raphson

i	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10	Step 11
1											
2	0.012	0.013	0.015	0.016	0.018	0.02	0.023	0.028	0.033	0.0422	0.0548
3	6E-04	8E-04	9E-04	0.001	0.001	0.002	0.002	0.004	0.005	0.0094	0.0189
4	3E-05	4E-05	6E-05	8E-05	1E-04	2E-04	3E-04	5E-04	9E-04	0.0023	0.0075
5	2E-06	3E-06	4E-06	6E-06	9E-06	2E-05	3E-05	6E-05	2E-04	0.0005	0.0031
6	1E-07	1E-07	2E-07	4E-07	8E-07	1E-06	3E-06	9E-06	3E-05	0.0001	0.0013
7	5E-09	9E-09	2E-08	3E-08	6E-08	1E-07	4E-07	1E-06	5E-06	3E-05	0.0006
8	3E-10	5E-10	1E-09	2E-09	5E-09	1E-08	4E-08	2E-07	8E-07	8E-06	0.0002
9	1E-11	3E-11	6E-11	2E-10	4E-10	1E-09	4E-09	2E-08	1E-07	2E-06	0.0001
10	8E-13	2E-12	4E-12	1E-11	3E-11	1E-10	5E-10	3E-09	3E-08	5E-07	4E-05
11	4E-14	1E-13	3E-13	8E-13	3E-12	1E-11	5E-11	4E-10	4E-09	1E-07	2E-05
12	2E-15	6E-15	2E-14	6E-14	2E-13	1E-12	6E-12	5E-11	8E-10	3E-08	8E-06
13	0	0	1E-15	4E-15	2E-14	9E-14	7E-13	7E-12	1E-10	7E-09	3E-06
14			0	0	0	9E-15	7E-14	9E-13	2E-11	2E-09	1E-06
15			0	0	0	0	9E-15	1E-13	4E-12	4E-10	6E-07
16			0	0	0	0	0	2E-14	7E-13	1E-10	3E-07
17			0	0	0	0	0	0	1E-13	3E-11	1E-07
18			0	0	0	0	0	0	2E-14	6E-12	5E-08
19			0	0	0	0	0	0	0	2E-12	2E-08
20			0	0	0	0	0	0	0	4E-13	9E-09
21			0	0	0	0	0	0	0	9E-14	4E-09
22			0	0	0	0	0	0	0	3E-14	2E-09
23			0	0	0	0	0	0	0	0	7E-10
24			0	0	0	0	0	0	0	0	3E-10
25			0	0	0	0	0	0	0	0	1E-10
26			0	0	0	0	0	0	0	0	5E-11
27			0	0	0	0	0	0	0	0	2E-11
28			0	0	0	0	0	0	0	0	1E-11
29			0	0	0	0	0	0	0	0	4E-12
30			0	0	0	0	0	0	0	0	2E-12
31			0	0	0	0	0	0	0	0	7E-13
32			0	0	0	0	0	0	0	0	3E-13
33			0	0	0	0	0	0	0	0	1E-13
34			0	0	0	0	0	0	0	0	6E-14

Residual Reduction: Newton Raphson

Table 7: Residual Reduction in Newton Raphson

i	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	Step 10	Step 11
1											
2	0.013	0.014	0.016	0.017	0.02	0.022	0.026	0.032	0.04	0.0536	0.0811
3	1E-05	1E-05	2E-05	2E-05	3E-05	5E-05	9E-05	2E-04	3E-04	0.001	0.0054
4	5E-12	1E-11	2E-11	4E-11	1E-10	3E-10	1E-09	4E-09	3E-08	4E-07	3E-05

Soft Spots and Overall performance

- The Newton Raphson Algorithm converges fairly quickly because the stiffness (slope) is updated in each iteration. This is costly but in turn enhances the overall performance of the Algorithm
- In case of the Modified Newton Raphson Algorithm the stiffness is not updated at every iteration, hence the cost is saved, but it takes many more iterations to converge to a similar tolerance level when compared to the Newton-Raphson Algorithm.
- Both the algorithms, when used with the incremental load method are unable to capture the fall in the actual force as the slope equals zero at $F = 6$
- When encountered with unloading (decreasing F) a similar problem could occur as explained in the previous point.