CEE576 Assignment#3

Hao Yin

Question 1

Assume we have a linear system with $\boldsymbol{K}, \boldsymbol{K}$ not necessarily symmetric positive-definite. Use the minimization of residual idea to derive an explicit formula for $S^{(i)}$. What conditions on $\boldsymbol{K}$, K would be necessary to generally improve the solution in each iteration?

Derivation of the explicit formula for $s^{(i)}$:
Minimization of the residual:
Select $s^{(i)}$ such that $\boldsymbol{R}^{(i+1)} = \boldsymbol{F} - \boldsymbol{K}\boldsymbol{d}^{(i+1)}$ has zero component in the direction $\Delta\boldsymbol{d}^{(i)}$

$$\boldsymbol{0} = \Delta\boldsymbol{d}^{(i)} \cdot \boldsymbol{R}^{(i+1)}$$

since $\boldsymbol{d}^{(i+1)} = \boldsymbol{d}^{(i)} + s^{(i)}\Delta\boldsymbol{d}^{(i)}$, now $\boldsymbol{R}^{(i+1)}$ becomes:

$$\boldsymbol{R}^{(i+1)} = \boldsymbol{F} - \boldsymbol{K}\left(\boldsymbol{d}^{(i)} + s^{(i)}\Delta\boldsymbol{d}^{(i)}\right)$$

now pick $s^{(i)}$ such that the dot product becomes 0:

$$\Delta\boldsymbol{d}^{(i)} \cdot \left[\boldsymbol{F} - \boldsymbol{K}\left(\boldsymbol{d}^{(i)} + s^{(i)}\Delta\boldsymbol{d}^{(i)}\right)\right] = \boldsymbol{0}$$

$$s^{(i)} = \frac{\Delta\boldsymbol{d}^{(i)}\boldsymbol{F} - \Delta\boldsymbol{d}^{(i)}\boldsymbol{K}\boldsymbol{d}^{(i)}}{\Delta\boldsymbol{d}^{(i)}\boldsymbol{K}\Delta\boldsymbol{d}^{(i)}}$$

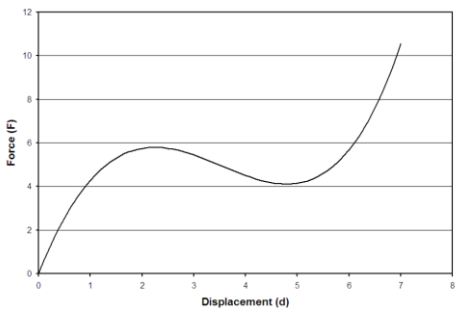
Conditions for improving the solution in each iteration:

1. $\boldsymbol{K}$ is real
2. $\boldsymbol{K}$ is nearly positive definite
3. Restarting the iteration

Question 2

Consider the relation:

$$N = (0.19d^3 - 2d^2 + 6d)e^{0.02d}$$



Use *arc-length control* with $b = 1/2$ in steps of $\delta a = 0.25$ to obtain the force-displacement curve in the range of $d \in [0,6]$ by the *modified Newton method* with *line search*. Use tolerance parameter $\varepsilon = 10^{-12}$.

Write in detail the algorithm and the solution procedure, (i.e., the system of equations to be solved). Also provide the following.

(a) Draw a comprehensive flow chart.

(b) Write a computer program to solve the problem.

(c) Attach the hard copy of the source code.

(d) Upload the soft copy of the source code to compass. In a word file provide the set of instructions to run your code and verify the results.

(e) Run your program for $\mathrm{b} = 0.5$
a. Make a table, listing the computed results, i.e., indicate the number of iterations required for each load step and the residual reduction in each iteration.
b. Plot the force-displacement curve obtained from the program.
c. For *any one of the load steps*, can you pictorially show the process of iterative convergence of your method.

(f) Rerun your program for $\mathrm{b} = 0.8$
a. Make a table, listing the computed results, i.e., indicate the number of iterations required for each load step and the residual reduction in each iteration.
b. Plot the force-displacement curve obtained from the program.
c. For *any one of the load steps*, can you pictorially show the process of iterative convergence of your method.

(g) Write your comments on the overall performance of the arc-length algorithm.

(h) With the help of the above problems, describe and discuss the soft *spots* for the algorithm.

(i) Discuss possible improvements to the algorithm.

(a) The algorithm and the solution procedure of arc-length control method:

1. Initialization:

$$n = 0,\ d_0 = 0,\ F_0^{int^{(0)}} = N(d_0)$$

2. Initialization for each load step:

$i = 0,\ d_{n+1}^{(0)} = d_n,\ a = 0,\ \lambda = 0,$

$\delta d_{n+1}^{(0)} = \delta\lambda_{n+1}^{(0)} = \delta a,\ F_{n+1}^{int\ (0)} = F_n^{int},$

$k_n^{(0)} = \frac{\partial N}{\partial d}\left(d_n^{(0)}\right)$ (modified Newton method), $q = \frac{F^{ext}}{k_n^{(0)}}$

for the sign of $\delta\lambda_{n+1}^{(0)}$, if both of following two conditions hold,

(1) $det\widetilde{\boldsymbol{K}}(d_n) = -det\widetilde{\boldsymbol{K}}(d_{n-1})$
(2) $det\widetilde{\boldsymbol{K}}(d_n)$ changed sign by passing through zero
then,

$$sign\left(\delta\lambda_{n+1}^{(0)}\right) = -sign\left(\delta\lambda_n^{(0)}\right)$$

otherwise,

$$sign\left(\delta\lambda_{n+1}^{(0)}\right) = sign\left(\delta\lambda_n^{(0)}\right)$$

3. Solve for both the displacement $\Delta d^{(i)}$ and the load parameter $\Delta\lambda^{(i)}$:

$$\begin{bmatrix} k_{n+1}^{(i)} & -F_{n+1}^{ext} \\ \left(\frac{\partial f}{\partial \delta d}\right)^T & \frac{\partial f}{\partial \delta\lambda} \end{bmatrix} \begin{Bmatrix} \Delta d^{(i)} \\ \Delta\lambda^{(i)} \end{Bmatrix} = \begin{Bmatrix} \lambda_{n+1}^{(i)} F_{n+1}^{ext} - F_{n+1}^{int}\left(d_{n+1}^{(i)}\right) \\ \delta a - f_{n+1}^{(i)} \end{Bmatrix}$$

where:

$$k_{n+1}^{(i)} = k_n^{(0)}$$

$$\left(\frac{\partial f}{\partial \delta d}\right)_{n+1}^{(i)} = \frac{c k_n^{(0)} \delta d^{(i)}}{f_{n+1}^{(i)}}$$

$$c = \frac{(1-b)}{q k_n^{(0)} q}$$

$$\left(\frac{\partial f}{\partial \delta\lambda}\right)_{n+1}^{(i)} = \frac{b\delta\lambda^{(i)}}{f_{n+1}^{(i)}}$$

$d_{n+1}^{(i)}$ ------- last converged value from last step

4. Use Secant Method to update line search parameter $s^{(i)}$:

$$0 = G\left(s^{(i)}\right) = \Delta d^{(i)} \cdot \left(\lambda_{n+1}^{(i)} F^{ext} - F^{int}\left(d_{n+1}^{(i)} + s^{(i)}\Delta d^{(i)}\right)\right)$$

1) Choose $s_0 = 0,\ s_1 = 0.1$
2) $s^{(k+1)} = s^{(k)} - G\left(s^{(k)}\right)\frac{s^{(k)} - s^{(k-1)}}{G\left(s^{(k)}\right) - G\left(s^{(k-1)}\right)}$
3) Iterate until $G\left(s^{(k)}\right) < \frac{1}{2}G(0)$

5. Update displacement and load parameter using $\Delta d_{n+1}^{(i+1)}$ and load parameter $\Delta\lambda^{(i)}$:

$$\lambda_{n+1}^{(i+1)} = \lambda_{n+1}^{(i)} + \Delta\lambda^{(i)}$$

$$d_{n+1}^{(i+1)} = d_{n+1}^{(i)} + s^{(i)}\Delta d^{(i)}$$

Residual Check:

$$\left\|R^{(i+1)}\right\| = \lambda_{n+1}^{(i)} F^{ext} - F^{int}(d_{n+1}^{(i)})$$

if $\left\|R^{(i+1)}\right\| \leq \varepsilon\left\|R^{(0)}\right\| \cap f_{n+1}^{(i)} - \delta a \leq \varepsilon\delta a$, proceed to next load step, $n = n + 1$

else, proceed to next iteration, $i = i + 1$

Procedure:

1) Initialization
2) Find both the displacement $\Delta d^{(i)}$ and the load parameter $\Delta\lambda^{(i)}$
3) Use Secant Method to update line search parameter $s^{(i)}$
4) Update displacement using $d_{n+1}^{(i+1)}$ and load parameter $\Delta\lambda^{(i)}$
5) Residual Check

(b) Comprehensive flow chart:

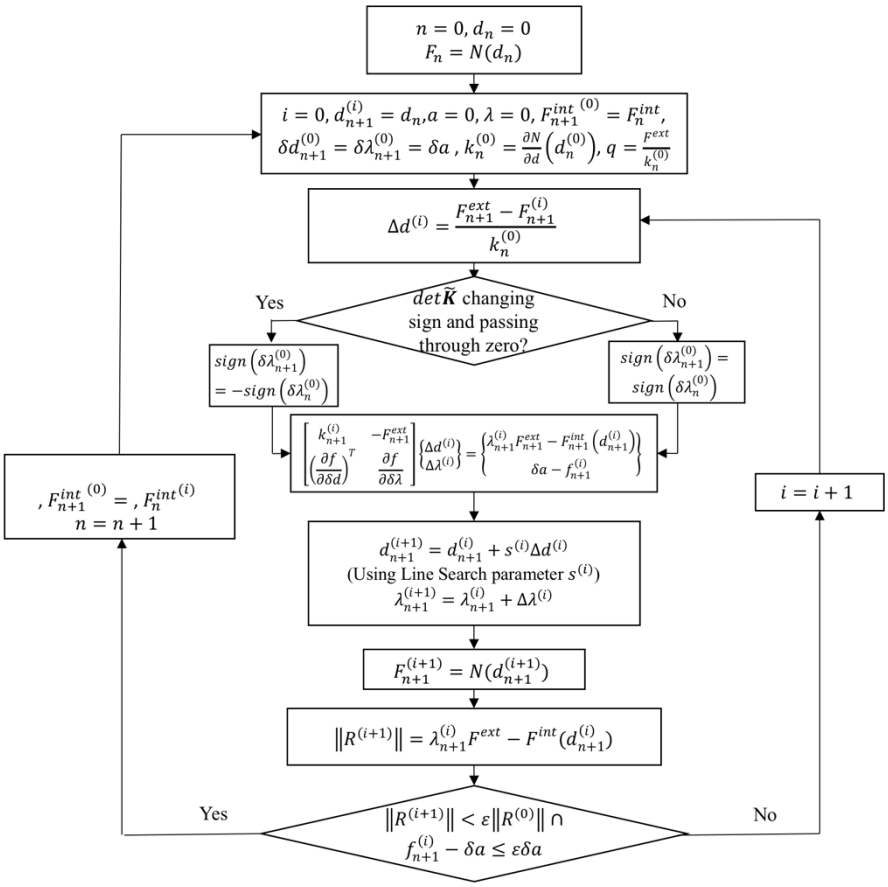


Figure 1. Flow chart of arc-length control method

(c) List of computed results:

Table 1. Computed results of arc-length control method (b=0.5)

(d) Force-displacement curve

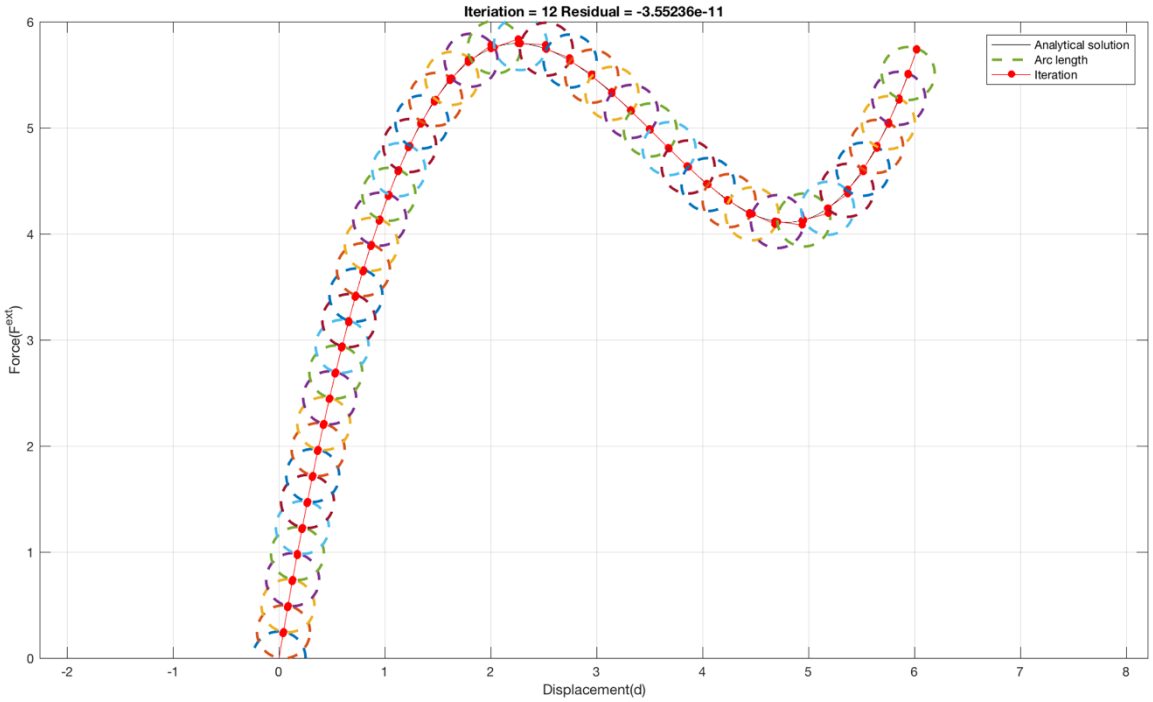


Figure 2. Force-displacement curve calculated by arc-length method

(e) Process of iterative convergence of arc-length control method (b=0.5)

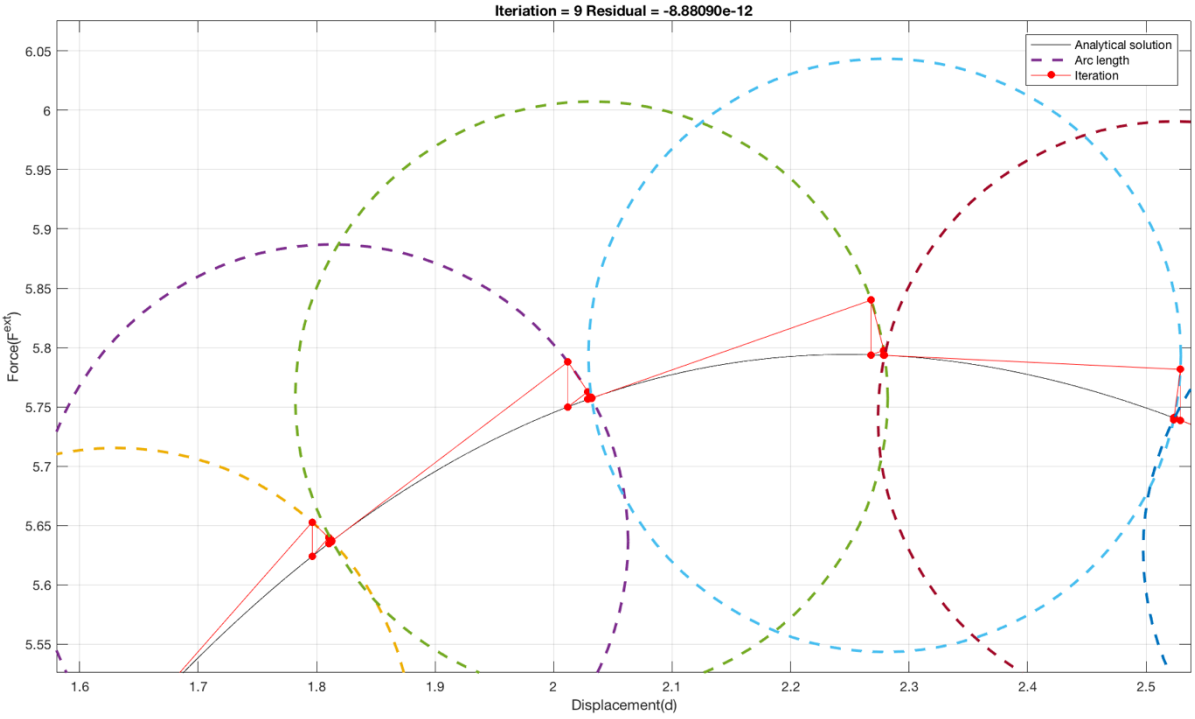


Figure 3. Process of iterative convergence calculated by arc-length method (b=0.5, n=10)

(f) List of computed results:

Table 2. Computed results of arc-length control method (b=0.8)

i	$\|R^{(i)}\|$	i	$\|R^{(i)}\|$
Load step #1 ($n=0$)		Load step #4 ($n=3$)	
0	0.0130	0	0.0173
1	6.7899×10^{-4}	1	0.0012
2	3.6025×10^{-5}	2	8.6985×10^{-5}
3	1.9127×10^{-6}	3	6.2247×10^{-6}
4	1.0156×10^{-7}	4	4.4548×10^{-7}
5	5.3923×10^{-9}	5	3.1882×10^{-8}
6	2.8631×10^{-10}	6	2.2817×10^{-9}
7	1.5202×10^{-11}	7	1.6329×10^{-10}
8	8.0724×10^{-13}	8	1.1686×10^{-11}
9	4.2910×10^{-14}	9	8.3644×10^{-13}
10	2.2204×10^{-15}	10	6.0396×10^{-14}
		11	4.2188×10^{-15}
Load step #2 ($n=1$)		Load step #5 ($n=4$)	
0	0.0142	0	0.0195
1	8.0965×10^{-4}	1	0.0015
2	4.6987×10^{-5}	2	1.2518×10^{-4}
3	2.7291×10^{-6}	3	1.0138×10^{-5}
4	1.5852×10^{-7}	4	8.2113×10^{-7}
5	9.2077×10^{-9}	5	6.6510×10^{-8}
6	5.3483×10^{-10}	6	5.3872×10^{-9}
7	3.1065×10^{-11}	7	4.3636×10^{-10}
8	1.8044×10^{-12}	8	3.5344×10^{-11}
9	1.0469×10^{-13}	9	2.8630×10^{-12}
10	6.2172×10^{-15}	10	2.3181×10^{-13}
		11	1.8652×10^{-14}
Load step #3 ($n=2$)		Load step #6 ($n=5$)	
0	0.0156	0	0.0224
1	9.8227×10^{-4}	1	0.0020
2	6.2915×10^{-5}	2	1.8965×10^{-4}
3	4.0337×10^{-6}	3	1.7697×10^{-5}
4	2.5864×10^{-7}	4	1.6517×10^{-6}
5	1.6584×10^{-8}	5	1.5417×10^{-7}
6	1.0633×10^{-9}	6	1.4389×10^{-8}
7	6.8178×10^{-11}	7	1.3430×10^{-9}
8	4.3714×10^{-12}	8	1.2535×10^{-10}
9	2.8044×10^{-13}	9	1.1700×10^{-11}
10	1.8208×10^{-14}	10	1.0925×10^{-12}

11	8.8818×10^{-16}	11	1.0125×10^{-13}
		12	9.7700×10^{-15}

i	$\|R^{(i)}\|$	i	$\|R^{(i)}\|$
Load step #7 ($n = 6$)		Load step #10 ($n = 9$)	
0	0.0262	0	0.0546
1	0.0028	1	0.0124
2	3.0774×10^{-4}	2	0.0030
3	3.3895×10^{-5}	3	7.2687×10^{-4}
4	3.7345×10^{-6}	4	1.7778×10^{-4}
5	4.1147×10^{-7}	5	4.3512×10^{-5}
6	4.5337×10^{-8}	6	1.0652×10^{-5}
7	4.9953×10^{-9}	7	2.6078×10^{-6}
8	5.5039×10^{-10}	8	6.3845×10^{-7}
9	6.0643×10^{-11}	9	1.5630×10^{-7}
10	6.6822×10^{-12}	10	3.8267×10^{-8}
11	7.3630×10^{-13}	11	9.3685×10^{-9}
12	8.0824×10^{-14}	12	2.2936×10^{-9}
13	8.4377×10^{-15}	13	5.6152×10^{-10}
Load step #8 ($n = 7$)		14	1.3747×10^{-10}
0	0.0317	15	3.3656×10^{-11}
1	0.0041	16	8.2396×10^{-12}
2	5.4971×10^{-4}	17	2.0171×10^{-12}
3	7.3949×10^{-5}	18	4.9383×10^{-13}
4	9.9536×10^{-6}	19	1.2079×10^{-13}
5	1.3398×10^{-6}	20	2.9310×10^{-14}
6	1.8036×10^{-7}	Load step #11 ($n = 10$)	
7	2.4278×10^{-8}	0	0.0865
8	3.2681×10^{-9}	1	0.0317
9	4.3993×10^{-10}	2	0.0128
10	5.9220×10^{-11}	3	0.0053
11	7.9710×10^{-12}	4	0.0023
12	1.0729×10^{-12}	5	9.6128×10^{-4}
13	1.4477×10^{-13}	6	4.0989×10^{-4}
14	1.9540×10^{-14}	7	1.7495×10^{-4}
Load step #9 ($n = 8$)		8	7.4701×10^{-5}
0	0.0400	9	3.1903×10^{-5}
1	0.0066	10	1.3626×10^{-5}
2	0.0011	11	5.8200×10^{-6}
3	1.9639×10^{-4}	12	2.4859×10^{-6}
4	3.4037×10^{-5}	13	1.0618×10^{-6}
5	5.9005×10^{-6}	14	4.5352×10^{-7}

6	1.0229×10^{-6}	15	1.9371×10^{-7}
7	1.7733×10^{-7}	16	8.2741×10^{-8}
8	3.0742×10^{-8}	17	3.5341×10^{-8}
9	5.3295×10^{-9}	18	1.5095×10^{-8}
10	9.2393×10^{-10}	19	6.4477×10^{-9}
11	1.6017×10^{-10}	20	2.7540×10^{-9}
12	2.7768×10^{-11}	21	1.1763×10^{-9}
13	4.8139×10^{-12}	22	5.0244×10^{-10}
14	8.3400×10^{-13}	23	2.1461×10^{-10}
15	1.4388×10^{-13}	24	9.1666×10^{-11}
16	2.5757×10^{-14}	25	3.9154×10^{-11}
		26	1.6724×10^{-11}
		27	7.1436×10^{-12}
		28	3.0500×10^{-12}
		29	1.3030×10^{-12}
		30	5.5689×10^{-13}
		31	2.3714×10^{-13}
		32	1.0214×10^{-13}
		33	4.2633×10^{-14}

(g) Force-displacement curve

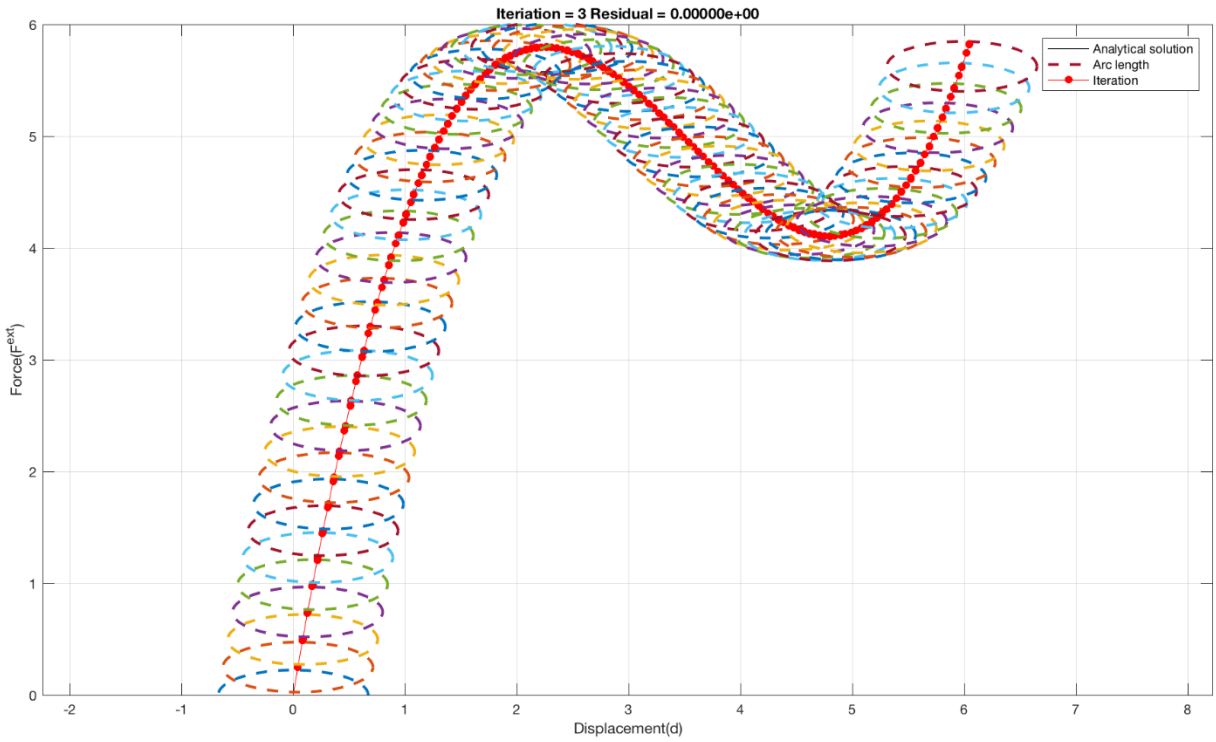


Figure 4. Force-displacement curve calculated by arc-length method (b=0.8)

(h) Process of iterative convergence of arc-length control method (b=0.8)

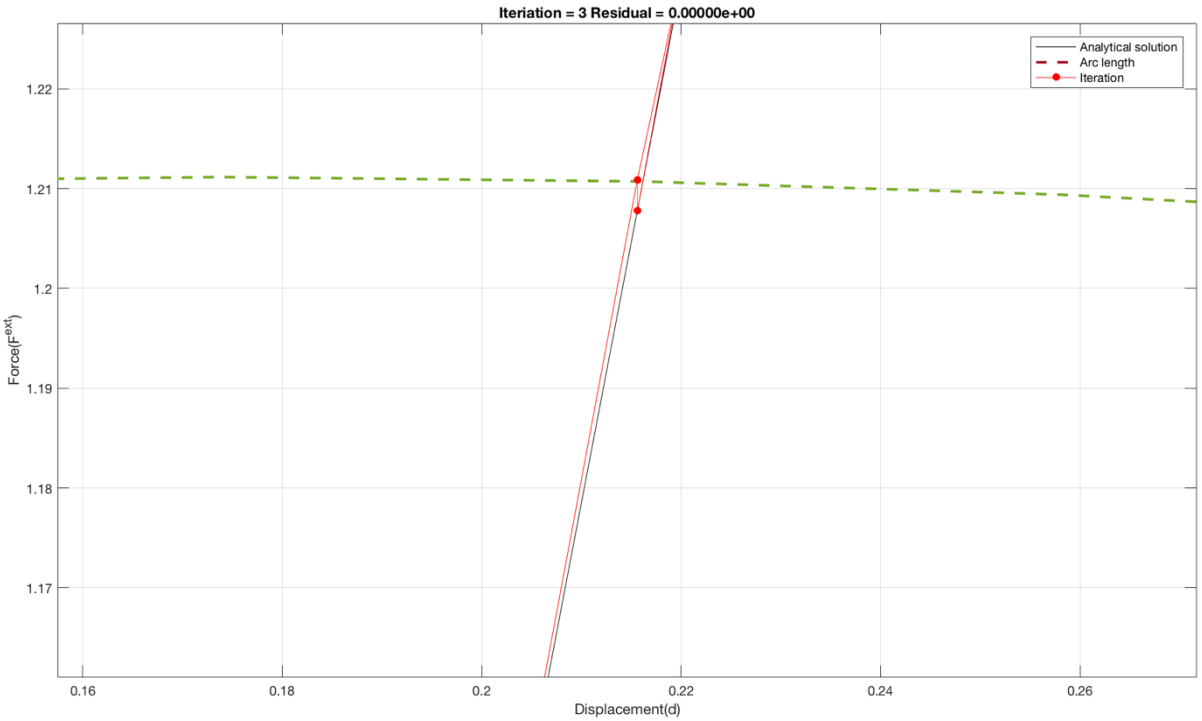


Figure 5. Process of iterative convergence calculated by arc-length method (b=0.8, n=1)

(i) Comments, soft spots and possible improvements on the performance of arc-length method

The arc-length method is an efficient method in solving non-linear systems of equations when there exist some critical points (i.e. dynamic snap under load control and dynamic snap under displacement control exist). In terms of a simple mechanical loading-unloading problem, a critical point could be interpreted as the point at which the loaded body cannot support an increase of the external forces and an instability occurs.

(j) Soft spots and possible improvements on the performance of arc-length method

The arc-length method has a very critical drawback: every iteration, the solver determines two sets of solutions (this is no surprise since a circle would always intersect a curve into two points), however, if we can't properly choose the set of solutions, then the solution would "evolves backward". One of the possible improvement is using line search method to supervise the direction the arc-length method evolving.

(k) Source Code

arc_length_new.m

```
clear; clc;


% Plot the analytical solution
%--------------------------------------------------------------------------
x = 0:0.001:6.0;
for i = 1:length(x)
    y(i) = calc_F(x(i));
end
psi = 1.0;
deltalL = 0.25;
%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1)
set(1,'Color','w')
set(1,'Position',[300 80 550*2 300*2])
set(gca,'Fontsize',16)
h1=plot(x,y,'k'); hold on
grid on
xlim([0 7])
ylim([0 6])
%h2=ezplot(['x.-^2+','(y.*',num2str(psi),').^2=',num2str(deltalL^2)],[0 pi./2]);
%set(h2,'LineWidth',2,'LineStyle','--');
axis equal


%--------------------------------------------------------------------------
% 1D Arch-length method with Modified newton raphson iteration
%--------------------------------------------------------------------------
% CEE576 Assignment#3
% Hao Yin, haoyin2@illinois.edu
% References:
% Hughes and Ferencz, Nonlinear FEM...
%--------------------------------------------------------------------------


%==========================================================================
% Initialization


% Assign storage space
load_step_limit = 500;
iteration_limit = 1000;


delta_d_matrix = zeros(load_step_limit,iteration_limit);
d_ini_matrix = zeros(load_step_limit,1);
```

```
d_new_matrix = zeros(load_step_limit,iteration_limit);
R_matrix = zeros(load_step_limit,iteration_limit);
F_int_matrix = zeros(load_step_limit,iteration_limit);
k_matrix = zeros(load_step_limit,1);
delta_lambda_matrix = zeros(load_step_limit,iteration_limit);
lambda_ini_matrix = zeros(load_step_limit,1);
lambda_new_matrix = zeros(load_step_limit,iteration_limit);
f_matrix = zeros(load_step_limit,iteration_limit);

tol = 1*10^(-12);
b = 1.0;
F_ext = 1.0; % here either assign the value of "q" or the value of "F_ext"
%q = 1.0;

delta_a = 0.25;

d0 = 0;
d = 0;
d_new = 0;

lambda0 = 0;
lambda = 0;
lambda_new = 0;

n = 1; % corresponding to n=0 in lecture note

while d_new <= 6
    % Initialization for each load step (i=1)
    lambda_ini_matrix(n) = lambda_new;
    d_ini_matrix(n) = d_new;
    k = calc_dF(d_new);
    k_matrix(n) = k;
    %q = F_ext/k;
    %c = (1-b)/(q*k*q);

    delta_d = sqrt(delta_a^2/(1+k^2));
    delta_lambda = delta_d * k;

    d_new_matrix(n,1) = d_new + delta_d;
    lambda_new_matrix(n,1) = lambda_new + delta_lambda;

    F_int(n,1) = calc_F(d_new_matrix(n,1));

    R(n,1) = lambda_new_matrix(n,1) - F_int(n,1);
```

```
    a1 = k^2 + 1;
    a2 = -2*k^2*d_new_matrix(n,1)-2*d_ini_matrix(n)+2*k*F_int(n,1)-
2*k*lambda_ini_matrix(n);
    a3 = d_ini_matrix(n)^2+k^2*d_new_matrix(n,1)^2-
2*k*d_new_matrix(n,1)*F_int(n,1)+2*k*d_new_matrix(n,1)*lambda_ini_matrix(n)+F_int(
n,1)^2-2*F_int(n,1)*lambda_ini_matrix(n)+lambda_ini_matrix(n)^2-0.25^2;

    d_new_matrix(n,2) = delta_d_solver(a1,a2,a3);

    lambda_new_matrix(n,2) = k*(d_new_matrix(n,2)-d_new_matrix(n,1))+F_int(n,1);

    F_int(n,2) = calc_F(d_new_matrix(n,2));
    R(n,2) = lambda_new_matrix(n,2) - F_int(n,2);

    % plot the iterative convergence
    line([d_ini_matrix(n) d_new_matrix(n,1)],[calc_F(d_ini_matrix(n))
lambda_new_matrix(n,1)],'LineStyle','-','Color',[1 0 0])
    plot(d_new_matrix(n,1),calc_F(d_new_matrix(n,1)),'ro','Markersize',5,'Markerfac
ecolor','r'); hold on
    line([d_new_matrix(n,1) d_new_matrix(n,1)],[lambda_new_matrix(n,1)
calc_F(d_new_matrix(n,1))],'LineStyle','-','Color',[1 0 0])

    line([d_new_matrix(n,1) d_new_matrix(n,2)],[calc_F(d_new_matrix(n,1))
lambda_new_matrix(n,2)],'LineStyle','-','Color',[1 0 0])
    plot(d_new_matrix(n,2),calc_F(d_new_matrix(n,2)),'ro','Markersize',5,'Markerfac
ecolor','r'); hold on
    line([d_new_matrix(n,2) d_new_matrix(n,2)],[lambda_new_matrix(n,2)
calc_F(d_new_matrix(n,2))],'LineStyle','-','Color',[1 0 0])
      %
    i = 2;

    while abs(R(n,i)) > abs(tol*R(n,1))
        a1 = k^2 + 1;
        a2 = -2*k^2*d_new_matrix(n,i)-2*d_ini_matrix(n)+2*k*F_int(n,i)-
2*k*lambda_ini_matrix(n);
        a3 = d_ini_matrix(n)^2+k^2*d_new_matrix(n,i)^2-
2*k*d_new_matrix(n,i)*F_int(n,i)+2*k*d_new_matrix(n,i)*lambda_ini_matrix(n)+F_int(
n,i)^2-2*F_int(n,i)*lambda_ini_matrix(n)+lambda_ini_matrix(n)^2-0.25^2;

        d_new_matrix(n,i+1) = delta_d_solver(a1,a2,a3);
        lambda_new_matrix(n,i+1) = k*(d_new_matrix(n,i+1)-
d_new_matrix(n,i))+F_int(n,i);
```

```
        F_int(n,i+1) = calc_F(d_new_matrix(n,i+1));
        R(n,i+1) = lambda_new_matrix(n,i+1) - F_int(n,i+1);
        Residual = R(n,i+1);
        i = i + 1;

        if i>=2
            % plot the iterative convergence
            line([d_new_matrix(n,i-1) d_new_matrix(n,i)],[calc_F(d_new_matrix(n,i-
1)) lambda_new_matrix(n,i)],'LineStyle','-','Color',[1 0 0])
            plot(d_new_matrix(n,i),calc_F(d_new_matrix(n,i)),'ro','Markersize',5,'M
arkerfacecolor','r'); hold on
            line([d_new_matrix(n,i) d_new_matrix(n,i)],[lambda_new_matrix(n,i)
calc_F(d_new_matrix(n,i))],'LineStyle','-','Color',[1 0 0])
            %
        end
    end

    lambda_new = lambda_new_matrix(n,i);
    d_new = d_new_matrix(n,i);

    % plot the arc-length circle for each load step

    theta = 0:pi/50:2*pi;
    xunit = delta_a * cos(theta) + d_ini_matrix(n);
    yunit = delta_a * sin(theta) + calc_F(d_ini_matrix(n));
    h2 = plot(xunit,yunit);
    set(h2,'LineWidth',2,'LineStyle','--');

    % plot the iterative convergence
    num_of_non_zero_elements = sum(d_new_matrix(n,:)~=0);
    h3=plot(d_new_matrix(n,1:num_of_non_zero_elements),lambda_new_matrix(n,1:num_of
_non_zero_elements)*F_ext,'r-o','Markersize',5,'Markerfacecolor','r');

    title(['Iteriation = ',num2str(i),' ','Residual = ',num2str(Residual * 9999,
'%10.5e\n')])
    legend([h1 h2 h3],'Analytical solution','Arc length','Iteration')
    xlabel('Displacement(d)')
    ylabel('Force(F^{ext})')
    %
    pause(0.2)

    n = n + 1;
    %
end
```

arc_length_ellipse.m

```
clear; clc;

% Plot the analytical solution
%-------------------------------------------------------------------------
x = 0:0.001:6.0;
for i = 1:length(x)
    y(i) = calc_F(x(i));
end
psi = 1.0;
deltalL = 0.25;
%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1)
set(1,'Color','w')
set(1,'Position',[300 80 550*2 300*2])
set(gca,'Fontsize',16)
h1=plot(x,y,'k'); hold on
grid on
xlim([0 7])
ylim([0 6])
%h2=ezplot(['x.-^2+','(y.*',num2str(psi),').^2=',num2str(deltalL^2)],[0 pi./2]);
%set(h2,'LineWidth',2,'LineStyle','--');
axis equal

%-------------------------------------------------------------------------
% 1D Arch-length method with Modified newton raphson iteration
%-------------------------------------------------------------------------
% CEE576 Assignment#3
% Hao Yin, haoyin2@illinois.edu
% References:
% Hughes and Ferencz, Nonlinear FEM...
%-------------------------------------------------------------------------

%=========================================================================
% Initialization

% Assign storage space
load_step_limit = 500;
iteration_limit = 1000;

delta_d_matrix = zeros(load_step_limit,iteration_limit);
d_ini_matrix = zeros(load_step_limit,1);
d_new_matrix = zeros(load_step_limit,iteration_limit);
R_matrix = zeros(load_step_limit,iteration_limit);
```

```
F_int_matrix = zeros(load_step_limit,iteration_limit);
k_matrix = zeros(load_step_limit,1);
delta_lambda_matrix = zeros(load_step_limit,iteration_limit);
lambda_ini_matrix = zeros(load_step_limit,1);
lambda_new_matrix = zeros(load_step_limit,iteration_limit);
f_matrix = zeros(load_step_limit,iteration_limit);

tol = 1*10^(-12);
b = 0.8;
F_ext = 1.0; % here either assign the value of "q" or the value of "F_ext"
%q = 1.0;
k0 = 6;
delta_a = 0.25;

d0 = 0;
d = 0;
d_new = 0;

lambda0 = 0;
lambda = 0;
lambda_new = 0;

n = 1; % corresponding to n=0 in lecture note

while d_new <= 6
    % Initialization for each load step (i=1)
    lambda_ini_matrix(n) = lambda_new;
    d_ini_matrix(n) = d_new;
    k = calc_dF(d_new);
    k_matrix(n) = k;
    q = F_ext/k0;
    %c = (1-b)/(q*k*q);

    delta_d = sqrt(delta_a^2/((1-b)/(q^2)+b*k0^2));
    delta_lambda = delta_d * k;

    d_new_matrix(n,1) = d_new + delta_d;
    lambda_new_matrix(n,1) = lambda_new + delta_lambda;

    F_int(n,1) = calc_F(d_new_matrix(n,1));

    R(n,1) = lambda_new_matrix(n,1) - F_int(n,1);

    a1 = b*k^2 + (1-b)/(q^2);
```

```
    a2 = -2*b*k^2*d_new_matrix(n,1)-2*d_ini_matrix(n)*((1-
b)/(q^2))+2*b*k*F_int(n,1)-2*b*k*lambda_ini_matrix(n);
    a3 = (1-b)/(q^2)*d_ini_matrix(n)^2+b*k^2*d_new_matrix(n,1)^2-
2*b*k*d_new_matrix(n,1)*F_int(n,1)+2*b*k*d_new_matrix(n,1)*lambda_ini_matrix(n)+b
*F_int(n,1)^2-2*b*F_int(n,1)*lambda_ini_matrix(n)+b*lambda_ini_matrix(n)^2-
0.25^2;

    d_new_matrix(n,2) = delta_d_solver(a1,a2,a3);

    lambda_new_matrix(n,2) = k*(d_new_matrix(n,2)-d_new_matrix(n,1))+F_int(n,1);

    F_int(n,2) = calc_F(d_new_matrix(n,2));
    R(n,2) = lambda_new_matrix(n,2) - F_int(n,2);

    % plot the iterative convergence
    line([d_ini_matrix(n) d_new_matrix(n,1)],[calc_F(d_ini_matrix(n))
lambda_new_matrix(n,1)],'LineStyle','-','Color',[1 0 0])
    plot(d_new_matrix(n,1),calc_F(d_new_matrix(n,1)),'ro','Markersize',5,'Markerf
acecolor','r'); hold on
    line([d_new_matrix(n,1) d_new_matrix(n,1)],[lambda_new_matrix(n,1)
calc_F(d_new_matrix(n,1))],'LineStyle','-','Color',[1 0 0])

    line([d_new_matrix(n,1) d_new_matrix(n,2)],[calc_F(d_new_matrix(n,1))
lambda_new_matrix(n,2)],'LineStyle','-','Color',[1 0 0])
    plot(d_new_matrix(n,2),calc_F(d_new_matrix(n,2)),'ro','Markersize',5,'Markerf
acecolor','r'); hold on
    line([d_new_matrix(n,2) d_new_matrix(n,2)],[lambda_new_matrix(n,2)
calc_F(d_new_matrix(n,2))],'LineStyle','-','Color',[1 0 0])
      %
    i = 2;

    while abs(R(n,i)) > abs(tol*R(n,1))
        %xxx = xxxx
        a1 = b*k^2 + (1-b)/(q^2);
        a2 = -2*b*k^2*d_new_matrix(n,1)-2*d_ini_matrix(n)*((1-
b)/(q^2))+2*b*k*F_int(n,1)-2*b*k*lambda_ini_matrix(n);
        a3 = (1-b)/(q^2)*d_ini_matrix(n)^2+b*k^2*d_new_matrix(n,1)^2-
2*b*k*d_new_matrix(n,1)*F_int(n,1)+2*b*k*d_new_matrix(n,1)*lambda_ini_matrix(n)+b
*F_int(n,1)^2-2*b*F_int(n,1)*lambda_ini_matrix(n)+b*lambda_ini_matrix(n)^2-
0.25^2;

        d_new_matrix(n,i+1) = delta_d_solver(a1,a2,a3);
        lambda_new_matrix(n,i+1) = k*(d_new_matrix(n,i+1)-
d_new_matrix(n,i))+F_int(n,i);
```

```
        F_int(n,i+1) = calc_F(d_new_matrix(n,i+1));
        R(n,i+1) = lambda_new_matrix(n,i+1) - F_int(n,i+1);
        Residual = R(n,i+1);
        i = i + 1

        if i>=2
            % plot the iterative convergence
            line([d_new_matrix(n,i-1) d_new_matrix(n,i)],[calc_F(d_new_matrix(n,i-
1)) lambda_new_matrix(n,i)],'LineStyle','-','Color',[1 0 0])
            plot(d_new_matrix(n,i),calc_F(d_new_matrix(n,i)),'ro','Markersize',5,'
Markerfacecolor','r'); hold on
            line([d_new_matrix(n,i) d_new_matrix(n,i)],[lambda_new_matrix(n,i)
calc_F(d_new_matrix(n,i))],'LineStyle','-','Color',[1 0 0])
            %
        end
    end

    lambda_new = lambda_new_matrix(n,i);
    d_new = d_new_matrix(n,i);

    % plot the arc-length circle for each load step

    theta = 0:pi/50:2*pi;
    xunit = sqrt((1-b)/(q^2)) * delta_a * cos(theta) + d_ini_matrix(n);
    yunit = sqrt(b) * delta_a * sin(theta) + calc_F(d_ini_matrix(n));
    h2 = plot(xunit,yunit);
    set(h2,'LineWidth',2,'LineStyle','--');

    % plot the iterative convergence
    num_of_non_zero_elements = sum(d_new_matrix(n,:)~=0);
    h3=plot(d_new_matrix(n,1:num_of_non_zero_elements),lambda_new_matrix(n,1:num_
of_non_zero_elements)*F_ext,'r-o','Markersize',5,'Markerfacecolor','r');

    title(['Iteriation = ',num2str(i),' ','Residual = ',num2str(Residual * 9999,
'%10.5e\n')])
    legend([h1 h2 h3],'Analytical solution','Arc length','Iteration')
    xlabel('Displacement(d)')
    ylabel('Force(F^{ext})')
    %
    pause(0.2)
    n = n + 1;
    %
end
```