TO: Professor Andreas Linninger, Dr. Chang Sub Park & Grant Hartung

FROM: Kehinde Abioye

DATE: November 2, 2017

SUBJECT: Linearization and Newton-Raphson (Newton) method

Solving for nonlinear system of equations for concentration functions representing flux in and out of a cell using Newton's Method for

Introduction

Nonlinear ODEs are linearized by hand and in MATLAB using the Newton-Raphson method. Linearization is needed for nonlinear models to analyze a nonlinear function at a single point in a linear system. Taylor expansion can be used for this.

Methods

Part 1

The following 6 equations are conservation equation with reaction rate, k.

$$V\frac{dC_A}{dt} = V\left[-2k_1{C_A}^2C_B - 3k_2{C_A}^3C_C + k_3C_D\right] \tag{1}$$

$$V\frac{dC_B}{dt} = V\left[-k_1{C_A}^2C_B\right] \tag{2}$$

$$V\frac{dC_c}{dt} = V\left[-k_2{C_A}^3C_C + k_4C_F\right] \tag{3}$$

$$V\frac{dC_D}{dt} = V\left[2k_1{C_A}^2C_B + 3k_2{C_A}^3C_C - k_3C_D\right] \tag{4}$$

$$V\frac{dC_E}{dt} = V\left[k_1{C_A}^2C_B - k_4C_E\right] \tag{5}$$

$$V\frac{dC_F}{dt} = V\left[k_2{C_A}^3C_C - k_4C_F\right] \tag{6}$$

The equations were linearized and put in the matrix form, $\dot{C} = AC$.

Part 2

Equation 7 needs to be linearized.

$$V\frac{dc}{dt} = qc_{in} - qc - kc^3V \tag{7}$$

Part 3

Equation 8 and 9 was solved for using Newton-Raphson iteration method with initial guess x = [1 ;1].

$$f_1(x_1, x_2) = 2{x_1}^2 + x_2 - 1 = 0 \tag{8}$$

$$f_2(x_1, x_2) = -x_1 - 2{x_2}^2 + 1 = 0 \tag{9}$$

Part a:

The residual error contours were plotted for equations 8 and 9 using for loops, like in part 6 of memorandum 8, to create a matrix of residual error.

Part b:

The hand calculations for the Newton method are listed on a separate sheet of paper.

Part c:

Once the calculations in part b were done, they were put in a table like table 1 for three iterations. The calculations were verified in MATLAB.

k	$x^{(k)}$	$F(x^k)$	$F(x^k)^T F(x^k)$	$J(x^k) = [\nabla F(x^k)]$	$\Delta x^k = [J(x^k)]^{-1} F(x^k)$
0					
1					
2					
3					

Table 1. Solution of Newton-Raphson iterative method for each iteration, k

Part 4

Equations 10 and 11 were solved using the Newton-Raphson iterative method.

$$f_1(x_1, x_2) = 2x_1{}^2 + x_2 - 6 = 0 \tag{10}$$

$$f_2(x_1, x_2) = x_1 + 2x_2{}^2 - 3.5 = 0 \tag{11}$$

Part a:

The residual error contours were plotted for equations 10 and 11 using for loops to create a matrix of residual error.

Part b:

The Newton method was done in MATLAB and by hand with an initial guess of [2;1] and an alpha value of 1, 0.5, and 0.25. A table for each alpha value was created, as seen in table 1.

Part c:

The Newton method was performed in MATLAB and by hand with initial guess of [-1;-1] and an alpha value of 1, 0.5, and 0.25. A table for each alpha value was created, as seen in table 1.

Part 5

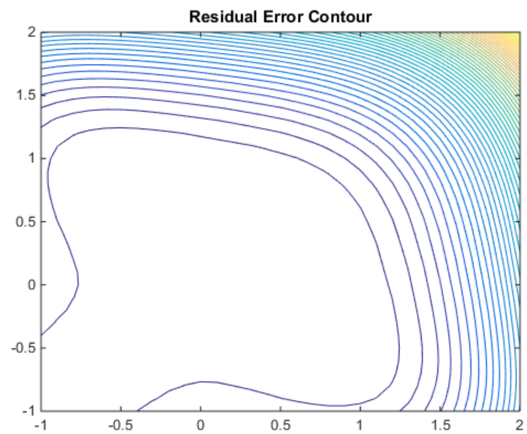
Equation 12 was solved using Taylor's series about $\theta_0 = 0$.

$$e^{i\theta} = \cos\theta + i\sin\theta \tag{12}$$

Results

Part 3a:

Residual error contour for equation 8 and 9:



Part 3d:

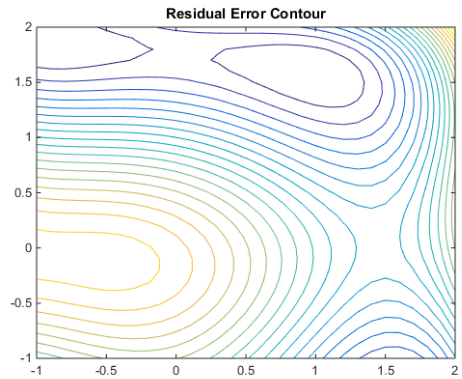
Equation 8 and 9 solved with Newton-Raphson iterative method with initial guess $\boldsymbol{x}(0)$ = [1; 1]:

Table 2. Solution of Newton-Raphson iterative method

k	$x^{(k)}$	$F(x^k)$	$F(x^k)^T F(x^k)$	$J(x^k) = [\nabla F(x^k)]$	$\Delta x^k = [J(x^k)]^{-1} F(x^k)$
0	$\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}2\\-2\end{bmatrix}$	8	$\begin{bmatrix}4 & 1\\-1 & -4\end{bmatrix}$	$\begin{bmatrix}-0.35\\-0.35\end{bmatrix}$
1	$\begin{bmatrix}0.6\\0.6\end{bmatrix}$	$\begin{bmatrix}0.32\\-0.32\end{bmatrix}$	0.205	$\begin{bmatrix}2.4 & 1\\-1 & -2.4\end{bmatrix}$	$\begin{bmatrix}0.094\\0.094\end{bmatrix}$
2	$\begin{bmatrix}0.506\\0.506\end{bmatrix}$	$\begin{bmatrix}0.018\\-0.018\end{bmatrix}$	0.036	$\begin{bmatrix}2.02 & 1\\-1 & -2.02\end{bmatrix}$	$\begin{bmatrix}0.006\\0.006\end{bmatrix}$
3	$\begin{bmatrix}0.5\\0.5\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	0	$\begin{bmatrix}2 & 1\\-1 & -2\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$

Part 4a:

Residual error contour for equations 10 and 11:



Part 4d:

Equations 10 and 11 solved with Newton-Raphson iterative method with an initial guess $\boldsymbol{x}(0)$ = [-1; -1]:

Table 3. Solution of Newton-Raphson iterative method for step size of α = 1

k	$x^{(k)}$	$F(x^k)$	$F(x^k)^T F(x^k)$	$J(x^k)= [\nabla F(x^k)]$	$\Delta x^k= [J(x^k)]^{-1}F(x^k)$
0	$\begin{bmatrix}-1\\-1\end{bmatrix}$	$\begin{bmatrix}-5\\-2.5\end{bmatrix}$	31.25	$\begin{bmatrix}-4 & 1\\1 & -4\end{bmatrix}$	$\begin{bmatrix}1.5\\1\end{bmatrix}$
1	$\begin{bmatrix}-2.5\\-2\end{bmatrix}$	$\begin{bmatrix}4.5\\2\end{bmatrix}$	24.25	$\begin{bmatrix}-10 & 1\\1 & -8\end{bmatrix}$	$\begin{bmatrix}-0.48\\-0.31\end{bmatrix}$
2	$\begin{bmatrix}-2.02\\-1.7\end{bmatrix}$	$\begin{bmatrix}0.46\\0.26\end{bmatrix}$	0.28	$\begin{bmatrix}-8.08 & 1\\1 & -6.8\end{bmatrix}$	$\begin{bmatrix}-0.063\\-0.047\end{bmatrix}$
3	$\begin{bmatrix}-1.95\\-1.65\end{bmatrix}$	$\begin{bmatrix}-0.045\\-0.005\end{bmatrix}$	0.0021	$\begin{bmatrix}-7.8 & 1\\1 & -6.6\end{bmatrix}$	$\begin{bmatrix}0.0059\\0.0016\end{bmatrix}$

Table 4. Solution of Newton-Raphson iterative method for step size of α = 0.5

k	$x^{(k)}$	$F(x^k)$	$F(x^k)^T F(x^k)$	$J(x^k)= [\nabla F(x^k)]$	$\Delta x^k= [J(x^k)]^{-1}F(x^k)$
0	$\begin{bmatrix}-1\\-1\end{bmatrix}$	$\begin{bmatrix}-5\\-2.5\end{bmatrix}$	31.25	$\begin{bmatrix}-4 & 1\\1 & -4\end{bmatrix}$	$\begin{bmatrix}1.5\\1\end{bmatrix}$
1	$\begin{bmatrix}-1.75\\-1.5\end{bmatrix}$	$\begin{bmatrix}-1.38\\-0.75\end{bmatrix}$	2.5	$\begin{bmatrix}-7 & 1\\1 & -6\end{bmatrix}$	$\begin{bmatrix}0.22\\0.16\end{bmatrix}$
2	$\begin{bmatrix}-1.86\\-1.58\end{bmatrix}$	$\begin{bmatrix}-0.66\\-0.37\end{bmatrix}$	0.57	$\begin{bmatrix}-7.44 & 1\\1 & -6.3\end{bmatrix}$	$\begin{bmatrix}0.099\\0.074\end{bmatrix}$
3	$\begin{bmatrix}-1.91\\-1.62\end{bmatrix}$	$\begin{bmatrix}-0.32\\-0.16\end{bmatrix}$	0.128	$\begin{bmatrix}-7.64 & 1\\1 & -6.4\end{bmatrix}$	$\begin{bmatrix}0.046\\0.031\end{bmatrix}$

Table 5. Solution of Newton-Raphson iterative method for step size of α = 0.25

k	**$x^{(k)}$**	**$F(x^k)$**	**$F(x^k)^T F(x^k)$**	**$J(x^k) = [\nabla F(x^k)]$**	**$\Delta x^k = [J(x^k)]^{-1} F(x^k)$**
0	$\begin{bmatrix}-1\\-1\end{bmatrix}$	$\begin{bmatrix}-5\\-2.5\end{bmatrix}$	31.25	$\begin{bmatrix}-4 & 1\\1 & -4\end{bmatrix}$	$\begin{bmatrix}1.5\\1\end{bmatrix}$
1	$\begin{bmatrix}-1.38\\-1.25\end{bmatrix}$	$\begin{bmatrix}-3.44\\-1.76\end{bmatrix}$	14.9	$\begin{bmatrix}-5.52 & 1\\1 & -5\end{bmatrix}$	$\begin{bmatrix}0.71\\0.49\end{bmatrix}$
2	$\begin{bmatrix}-1.55\\-1.37\end{bmatrix}$	$\begin{bmatrix}-2.57\\-1.29\end{bmatrix}$	8.3	$\begin{bmatrix}-6.2 & 1\\1 & -5.48\end{bmatrix}$	$\begin{bmatrix}0.46\\0.32\end{bmatrix}$
3	$\begin{bmatrix}-1.66\\-1.45\end{bmatrix}$	$\begin{bmatrix}-1.94\\-1.45\end{bmatrix}$	4.66	$\begin{bmatrix}-6.64 & 1\\1 & -5.8\end{bmatrix}$	$\begin{bmatrix}0.33\\0.22\end{bmatrix}$

Conclusion

This assignment was affective in teaching linearization of nonlinear ODEs. Newton-Raphson method and Taylor series expansion are useful in performing this task.

Discussion

Working out the problems by hand gave a deeper understanding on how linearization works. It is a way to visualize a complex system such as non-linear ODEs.

Appendix A

```
clc
clear all
close all

% Problem 3
nmax=5;
tol=1e-6;
x=[1;1]
% f1=2*x+y-1;
% f2=-x-(2*(y^2))+1;
fun = @(x) [(2*x(1,1)^2+x(2,1)-1);(-x(1,1)-2*x(2,1)^2+1)];
dfun = @(x) [4*x(1,1) 1;-1 -4*x(2,1)];
alfa=1;

x1 = -1:.1:2;
x2 = -1:.1:2;

for i=1:length(x1);
    for j=1:length(x2);
        resN = [(2*x1(i)^2)+x2(j)-1;(-x1(i)-2*x2(j)^2+1)];
        resSurf(i,j) = resN'*resN;
    end
end
contour(x1,x2,resSurf)
title('Residual Error Contour');

[xvect,resnorm] = newton(x,fun,dfun,nmax,alfa,tol)

% Problem 4
x=[2;1];
% x=[-1;-1];
% f1=2*(x^2)+y-6;
% f2=x+(2*(y^2))-3.5;
nmax=5;
tol=1e-6;
fun = @(x) [(2*x(1,1)^2+x(2,1)-6);(x(1,1)+2*x(2,1)^2-3.5)];
dfun = @(x) [4*x(1,1) 1;1 4*x(2,1)];
alfa=1;
alfa2=0.5;
alfa3=0.25;

x1 = -1:.1:2;
x2 = -1:.1:2;

for i=1:length(x1);
    for j=1:length(x2);
        resN = [(2*x1(i)^2+x2(j)-6);(x1(i)+2*x2(j)^2-3.5)];
        resSurf(i,j) = resN'*resN;
    end
end
figure
contour(x1,x2,resSurf,20)
title('Residual Error Contour');
```

```
[xvect,resnorm]= newton(x,fun,dfun,nmax,alfa,tol)
[xvect,resnorm]= newton(x,fun,dfun,nmax,alfa2,tol)
[xvect,resnorm]= newton(x,fun,dfun,nmax,alfa3,tol)

[xvect,resnorm]= newton(x,fun,dfun,nmax,alfa,tol)
[xvect,resnorm]= newton(x,fun,dfun,nmax,alfa2,tol)
[xvect,resnorm]= newton(x,fun,dfun,nmax,alfa3,tol)
```