ECSE 343: Numerical Methods in Engineering

Assignment 4

Due Date: 9th April 2021

Student Name: Hongtao Xu

Student ID: 260773785

Please type your answers and write you code in this .mlx script. If you choose to provide the handwritten answers, please scan your answers and include those in SINGLE pdf file.

Please submit this **.mlx** file along with the **PDF** copy of this file.

Note: You can write the function in the appendix section using the provided stencils.

Question 1: **Nonlinear Equations for univariate case.**

a) Bisection Method is used for tfinding the roots of a continuous function, $f(x)$, given endpoints;a, b with $f(a).f(b) < 0$. The interval $[a\ b]$contains a root, xr, because $f(a)$ and $f(b)$ have opposite signs.

Bisection method bisects the given interval at the midpoint, $c = \frac{a+b}{2}$ and then chooses a new interval based such that end of point of interval have opposite signs, i.e., if $f(a).f(c) < 0$, then the new interval is set to $[a\ c]$, else if $f(c).f(b) < 0$ then the interval is set to $[c\ b]$.

The above procedure is repeated until the following two conditions are simultaneously met,

- The function value at the interval is sufficiently small, i.e., $\|f(c)\|2 < \epsilon\text{tolerance}$
- The new interval is sufficiently small, i.e., $\|a-b\|2 < \epsilon\text{tolerance}$

Implement the Bisection Method in the cell below to find the root of $f(x) = x^4 - \square 2\,x^2 - 4$ in the interval $[-3\ \ 3]$ using $\epsilon\text{tolerance} = 10^{-5}$. **Show the number of iterations the bisection method took to converge.**

Use the cell below to implement your code.

Note: There is no need to write the function for Bisection Method. However, if you wish to implement the function, use the appendix.

```
f1 = -1;
f2 = 1;
t = 1;
s = 1;
b= -3;
a = 3;
c = 3;
tol = 1e-5;
while t >= tol & s >=tol
if f1*f2 < 0
a = c;
else
b = c;
end
c = (b+a)/2;
f1 = b^4-2*b^2-4;
f2 = c^4-2*c^2-4;
fb = a^4-2*a^2-4;
s = norm(b-a);
t = norm(f2/2);
end
result = c
```

```
result = -1.7989
```

```
% the number of iterations the bisecton method took to converge is here
```

b) **Newton-Raphson**

Bisection Method requires the given function, $f(x)$ to be continuous, Newton-Raphson requires $f(x)$ to be continuous and differentiable. While the Newton-Raphson method converges faster than the bisection method, however, Newton-Raphson method does not guarantee convergence. Newton-Raphson works by linearising the $f(x)$, at the given initial guess, $x^{(0)}$as shown below

$$f(x) = f(x^{(0)}) + f(x^{(0)})'(x - x^{(0)})$$

Setting the expression on the right to zero, provides an approximation to the root, and we obtain

$$x^{(1)} = x^{(0)} - \frac{f(x^{(0)})}{f(x^{(0)})'}$$

Then, the function, $f(x)$, is again linearised using approximation, $x^{(1)}$as the initial guess, to obtain new approximation of the root. After k iterations the new approximation for the root becomes,

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f(x^{(k)})'}$$

The above procedure is repeated untill the following two convergence condtions are simultaneously met,

- The function value at new guess point is sufficiently small, i.e., $\|f(x^{(k+1)})\|2 < \epsilon\text{tolerance}$
- The difference between the two consecutive solutions is sufficiently small, i.e., $\|x^{(k+1)} - x^{(k)}\|2 < \epsilon\text{tolerance}$

Implement the **Newton-Raphson** Method in the cell below to find the root of $f(x) = x^4 - \square 2\,x^2 - 4$ use intial guess of $x^{(0)} = 3$ using $\epsilon\text{tolerance} = 10^{-5}$. If the convergence is not reached in 100 iterations, quit the algorithm by displaying an error message indicating that Newton-Raphson failed to converge.

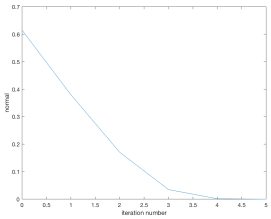
Show the number of iterations the Newton-Raphson method took to converge.

Also plot the 2-Norm of difference between consecutive solutions $\|x(k+1) - x(k)\|2$ for each iteration.

```
x0 = 3;
x1 = x0;
f1 = x1^4-2*x1^2-4;
fx = zeros(2,100);
tol = 1e-5;
ts = 1;
x = 0;
t = 1;
while t >= tol && ts >=tol
fx(1,x+1) = x;
df = 4*x1^3-4*x1;
x1 = x1-f1/df;
f1 = x1^4-2*x1^2-4;
ts = norm(x1-x0);
t = norm(f1);
fx(2,x+1) = ts;
if x == 100
ts = 1e-6;
t = 1e-6;
fprintf("Iteration over 101 times, Newton-Raphson can't converge")
end
x = x+1;
x0 = x1;
end
if x < 100
numberofit = x-1
finalx = x0
end
```

```
numberofit = 5
finalx = 1.7989
```

```
% the number of iterations the Newton-Raphson method took to converge is here
plot(fx(1,1:x),fx(2,1:x))
ylabel('normal')
xlabel('iteration number')
```



c) In this part implement the **Bisection Method**, and the **Newton-Raphson Method** to find the roots of $f(x) = arctan(x)$. Use ϵtolerance $= 10^{-5}$, to stop the algorithms. For bisection method use the interval $[-4\ 4]$. For Newton-Rapshon use $x^{(0)} = 4$ as initial guess. Comment on your findings.

Note: The derivative of $f(x) = arctan(x)$ is $\frac{1}{x2+1}$

If the convergence is not reached for Newto-Raphson in 100 iterations, quit the function by displaying an error message indicating that Newton-Raphson failed to converge. Plot the 2-Norm of difference between consecutive solutions $\|x(k+1) - x(k)\|2$ for each iteration, what can you conclude from the plot?

Use the cells below to implement both methods.

```
f = 1;
s = 1;
c0 = 4;
b = -1;
ar= -4;
br = 4;
cf = 1;
tol = 1e-5;
```

```
while f >= tol & s >=tol
if b*cf < 0
br = c0;
elseif fb*cf < 0
ar = c0;
end
b = atan(ar);
fb = atan(br);
c0 = (ar+br)/2;
cf = atan(c0);
s = norm(ar-br);
f = norm(cf/2);
if b == 0
c0 = ar
f = 0;
s = 0;
```

```
elseif fb ==0
c0 = ar
f = 0;
s = 0;
elseif cf ==0
f = 0;
s = 0;
end
end
```

```
result = c0
```

```
result = 0
```

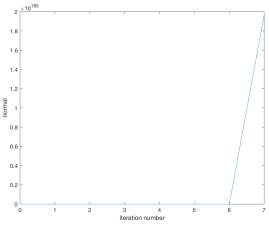
```
f = 1;
s = 1;
x = 0;
xa = 4;
xb = xa;
tol = 1e-5;
fa = atan(xb);
fx = zeros(2,100);
```

```
while f >= tol && s >=tol
fx(1,x+1) = x;
df = 1/(xb^2+1);
xb = xb-fa/df;
fa = atan(xb);
s = norm(xb-xa);
f = norm(fa);
fx(2,x+1) = s;
if x == 100
s = 1e-6;
f = 1e-6;
fprintf("Iteration over 101 times, Newton-Raphson can't converge")
end
xa = xb;
x = x+1;
end
```

```
if x < 100
finalx = xa
numberofit = x-1
end
```

```
finalx = NaN
numberofit = 9
```

```
plot(fx(1,1:x),fx(2,1:x))
ylabel('normal')
xlabel('iteration number')
```




```
% from the conclusion of the plot, the proper initial guess is very important, we should always pick the most proper ini
% I can see that my graph stops when it reaches nan because my initial is -4,  the graph would jump to the high value di
```

d) In this part re-implement **Newton-Raphson Method** from part c) to find the roots of $f(x) = arctan(x)$.

Use $x^{(0)} = 1.40$ as initial guess. Comment on your findings.

Implement your code in the cell below. Did the Newton-Raphson converge to the root of $f(x)$? What can you conclude?

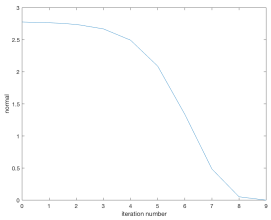
```
f = 1;
s = 1;
x = 0;
ka = 1.39;
kb = ka;
tol = 1e-5;
x1 = atan(kb);
fx = zeros(2,100);
```

```
while f >= tol && s >=tol
fx(1,x+1) = x;
df = 1/(kb^2+1);
kb = kb-x1/df;
x1 = atan(kb);
s = norm(kb-ka);
f = norm(x1);
```

```
fx(2,x+1) = s;
if x == 100
s = 1e-6;
f = 1e-6;
fprintf("Iteration over 101 times, Newton-Raphson can't converge")
end
ka = kb;
x = x+1;
end
if x < 100
finalx = ka
numberofit = x-1
end
```

```
finalx = 5.7004e-13
numberofit = 9
```

```
plot(fx(1,1:x),fx(2,1:x))
ylabel('normal')
xlabel('iteration number')
```




```
% the Newton-Raphson converge to the root of f(x) in the end, I can conclude that the initial value is very important.
% the lower initial guess 1.39/1.4 makes my graph moving slower and more accurate.
```

Question 2: **Nonlinear Equations for N-variables.**

Newton-Raphson Method can be used to solve the system of nonlinear equations of N-variables, $[x1,\ x2,\ \cdots,\ xn]$ shown below,

$$f1(x1,x2,\cdots,xN)=0$$
$$f2(x1,x2,\cdots,xN)=0$$
$$\vdots$$
$$fN(x1,x2,\cdots,xN)=0$$

The above equations can be written as a vector valued function as shown below,

$$\boldsymbol{f}(\boldsymbol{X})=\begin{bmatrix} f1(x1,x2,\cdots,xN) \\ f2(x1,x2,\cdots,xN) \\ \vdots \\ fN(x1,x2,\cdots,xN) \end{bmatrix}$$

where $\boldsymbol{X}=[x1,x2,\cdots,xN]$ is the vector containing all the variables.

Following the idea of unidimensional Newton-Rapshson, we start with an initial guess $\boldsymbol{X}^{(0)}$ and we use this to linearise the function $\boldsymbol{f}(\boldsymbol{X})$ around $\boldsymbol{X}^{(0)}$

$$\boldsymbol{f}(\boldsymbol{X})=\ f(\boldsymbol{X}^{(0)})+\left(\frac{\partial}{\partial \boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X}^{(0)})\right)(\boldsymbol{X}-\boldsymbol{X}^{(0)})$$

Setting the expresson on the left side to zero, we find the approximation of the solution vector. After k iteration the approximation for the solution vector can be written as

$$\boldsymbol{X}^{(k+1)}=\boldsymbol{X}^{(k)}-\left(\frac{\partial}{\partial \boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X}^{(k)})\right)^{-1}\boldsymbol{f}(\boldsymbol{X}^{(k)})$$

where $\boldsymbol{f}(\boldsymbol{X}^{(k)})$ is the vector valued fucntion evaluated at $\boldsymbol{X}^{(k)}$ and $\left(\frac{\partial}{\partial \boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X}^{(k)})\right)$ is the jacobian evaluated at $\boldsymbol{X}^{(k)}$. The structure of the Jacobian is given by,

$$\frac{\partial}{\partial \boldsymbol{X}}\boldsymbol{f}(\boldsymbol{X})=\begin{bmatrix} \frac{\partial f1}{\partial x1} & \frac{\partial f1}{\partial x2} & \cdots & \frac{\partial f1}{\partial xn} \\ \frac{\partial f2}{\partial x1} & \frac{\partial f2}{\partial x2} & \cdots & \frac{\partial f2}{\partial xn} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial fn}{\partial x1} & \frac{\partial fn}{\partial x2} & \cdots & \frac{\partial fn}{\partial xn} \end{bmatrix}$$

The above procedure is repeated untill the following two convergence condtions are simultaneously met,

- The function value at new guess point is sufficiently small, i.e., $\|\boldsymbol{f}(\boldsymbol{X}^{(k+1)})\|2<\epsilon\text{tolerance}$
- The difference between the two consecutive solutions is sufficiently small, i.e., $\|\boldsymbol{X}^{(k+1)}-\boldsymbol{X}^{(k)}\|2<\epsilon\text{tolerance}$

Find a solution of the following system of three nonlinear equations using Newton-Raphson method.

$$x1\,x2-x3^{2}=1$$
$$x1\ \ x2\,x3-x1^{2}+x2^{2}=2$$
$$e^{x1}-e^{x2}-x3=0$$

Note that this can be expressed as

$$f(X) = \begin{bmatrix} x1\,x2 - x3^2 - 1 \\ x1\,x2\,x3 - x1^2 + x2^2 - 2 \\ e^{x1} - e^{x2} - x3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \text{where } X = [x1\ x2\ x3]^T$$

a) Write a MATLAB function named *evaluateEquations(x)*, that evaluates the above equation at a given input vector.

Use the framework of the function provided in the Appendix.

b) Write a MATLAB function named *evaluateJacobian()*, that evaluates the Jacobian of the above equations.

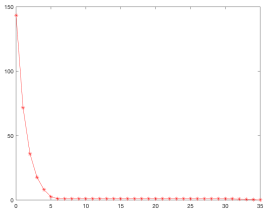
Use the framework of the function provided in the Appendix.

c) Implement the Newton-Raphson method, use the function named *NewtonRaphson()* to write the code for this. Use initial guess, $X^{(0)} = [35\ 0\ 2]^T$. Plot the 2-Norm of difference between consecutive solutions $\|X^{(k+1)} - X^{(k)}\|_2$ for each iteration.

```
Xguess = [35 0 2]';
[X,iter,delta] = NewtonRaphson(Xguess,1e-5)
```

```
X = 3×1
    1.6684
    1.4192
    1.1695
iter = 36
delta = 2×100
         0    1.0000    2.0000    3.0000    4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000   11.00
  143.3750   71.7122   35.7811   17.6406    8.0595    2.5852    1.0000    1.0000    1.0000    1.0000    1.0000    1.00
```

```
plot(delta(1,1:iter),delta(2,1:iter),'r-*')
```

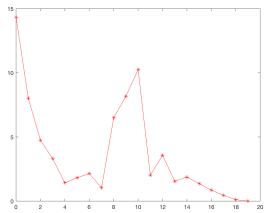


d) Run the part c) using the intial guess $X^{(0)} = [0.15\ 0\ 0]^T$ Explain the differences. **Why is the result in part (c) different than the result obtained in part (d)?**

```
Xguess = [0.15 0 0]';
[X2,iter2,deta2] = NewtonRaphson(Xguess,1e-5)
```

```
X2 = 3×1
   -0.7162
   -1.4935
    0.2640
iter2 = 20
deta2 = 2×100
         0    1.0000    2.0000    3.0000    4.0000    5.0000    6.0000    7.0000    8.0000    9.0000   10.0000   11.00
   14.3375    8.0308    4.7487    3.3089    1.4267    1.8252    2.1498    1.0356    6.4912    8.1770   10.2587    2.02
```

```
plot(deta2(1,1:iter2),deta2(2,1:iter2),'r-*')
```



```
% the reuslt in part c different than part b because they have different initial guess,
% the function has many different roots.
```

Appendix

Question 2) a)

```
function f  = evaluateEquations(x)
f = [x(1)*x(2)-x(3)^2-1;x(1)*x(2)*x(3)-x(1)^2+x(2)^2-2;exp(x(1))-exp(x(2))-x(3)];
end
```

Question 2) b)

```
function J  = evaluateJacobian(x)
syms x1 x2 x3
l = length(x);
sum = [x1 x2 x3];
J = sym('J',[3 3])';
s = [x1*x2-x3^2-1;x1*x2*x3-x1^2+x2^2;exp(x1)-exp(x2)-x3];
for i = 1:l
for j = 1:l
J(i,j) = diff(s(i),sum(j));
```

```
end
end
J = subs(J, {x1 x2 x3}, x');
end
```

Question 2) c)

```
function [Xout,iterations,delta]  = NewtonRaphson(Xguess,tolerance)
f = 1;
s = 1;
x = 0;
k1 = Xguess;
k2 = k1;
t = tolerance;
y = zeros(2,100);
x1 = evaluateEquations(k2);
while s >= t && f >=t
y(1,x+1) = x;
df = evaluateJacobian(k2);
k2 = double(k2-df\x1);
s = double(norm(k2-k1,"inf"));
x1 = evaluateEquations(k2);
f = double(norm(x1,"inf"));
y(2,x+1) = s;
if x == 100
s = 1e-6;
f = 1e-6;
fprintf("Iteration over 101 times, Newton-Raphson can't converge")
end
k1 = k2;
x = x+1;
end
if x < 100
delta = y;
Xout = k1;
iterations = x-1;
end
end
```