

University of Waterloo

ECE204 Lab Report

Simulation Assignment #3

Section: 202

Kehan Xing (k5xing)

Jing Liu (j762liu)

Submission Date: 2019-10-31

Q1. Create a generic function to solve a system of nonlinear equation using newton Raphson method. The function should accept any acceptable initial condition, variables as symbolic variable and nonlinear equation. Use your program to solve the following two non-linear equations:

$$f=[4*x^2+y^2-13, x^2+y^2-10];$$

$$f=[2*x-y-\exp(-x), -x+2*y-\exp(-y)];$$

The iteration should stop when the relative approximate error is less than 0.5%.

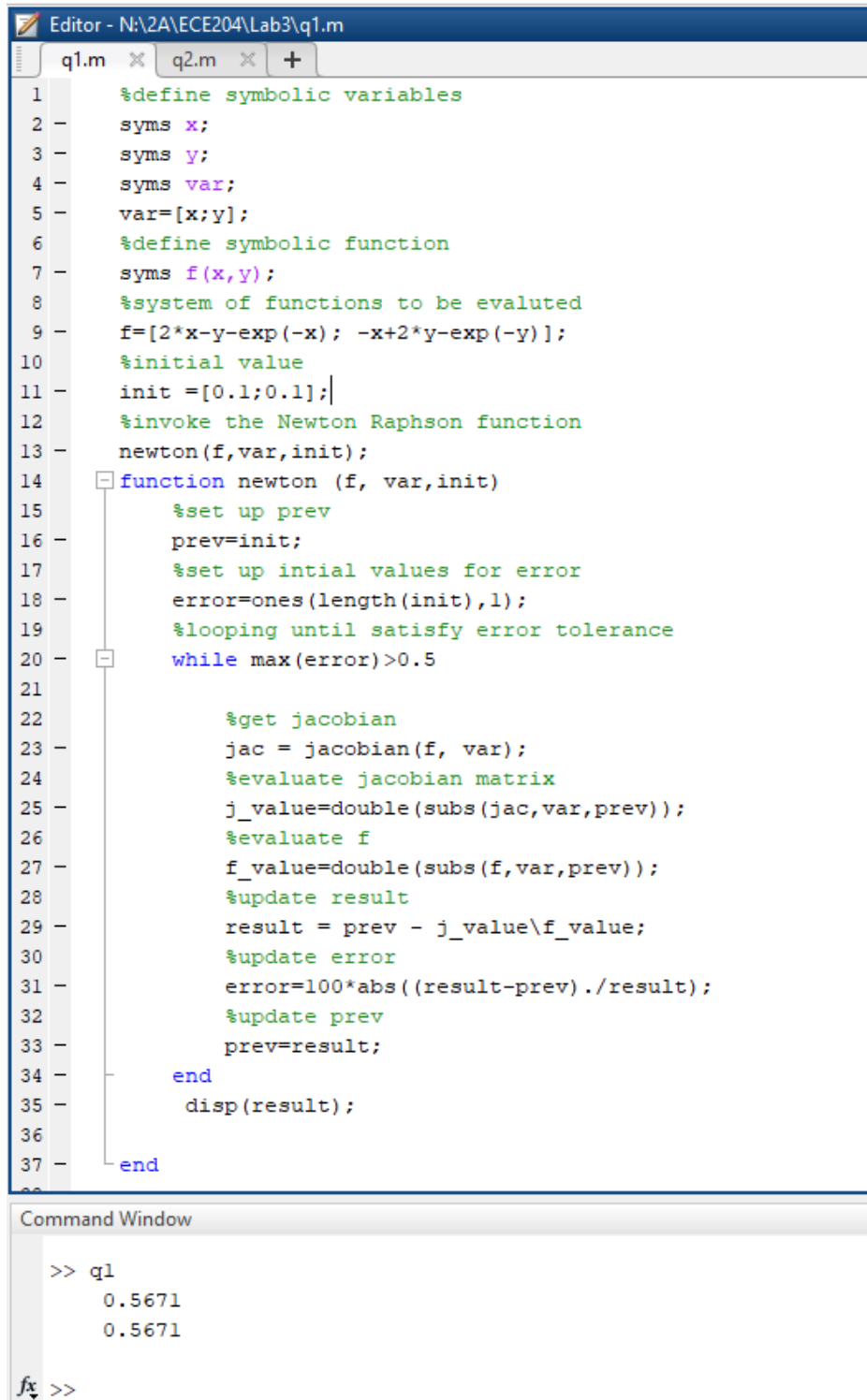
Code:

```
%define symbolic variables
syms x;
syms y;
syms var;
var=[x;y];
%define symbolic function
syms f(x,y);
%system of functions to be evaluated
f=[2*x-y-exp(-x); -x+2*y-exp(-y)];
%initial value
init =[0.1;0.1];
%invoke the Newton Raphson function
newton(f,var,init);
function newton (f, var,init)
    %set up prev
    prev=init;
    %set up intial values for error
    error=ones(length(init),1);
    %looping until satisfy error tolerance
    while max(error)>0.5

        %get jacobian
        jac = jacobian(f, var);
        %evaluate jacobian matrix
        j_value=double(subs(jac,var,prev));
        %evaluate f
        f_value=double(subs(f,var,prev));
        %update result
        result = prev - j_value\f_value;
        %update error
        error=100*abs((result-prev)./result);
        %update prev
        prev=result;
    end
    disp(result);
end
```

Display:

For $f=[2*x-y-\exp(-x), -x+2*y-\exp(-y)]$



The image shows a MATLAB Editor window with a script named q1.m. The script defines symbolic variables x and y, a system of functions f, and an initial value init. It then calls a function newton(f, var, init). The newton function is defined as a function handle that takes f, var, and init as inputs. It sets up prev, error, and loops until the error is less than 0.5. It calculates the Jacobian matrix, evaluates it, and updates the result and error. The final result is displayed.

```
Editor - N:\2A\ECE204\Lab3\q1.m
q1.m x q2.m x +
1 %define symbolic variables
2 - syms x;
3 - syms y;
4 - syms var;
5 - var=[x;y];
6 %define symbolic function
7 - syms f(x,y);
8 %system of functions to be evaluated
9 - f=[2*x-y-exp(-x); -x+2*y-exp(-y)];
10 %initial value
11 - init=[0.1;0.1];
12 %invoke the Newton Raphson function
13 - newton(f,var,init);
14 function newton (f, var,init)
15     %set up prev
16     prev=init;
17     %set up intial values for error
18     error=ones(length(init),1);
19     %looping until satisfy error tolerance
20     while max(error)>0.5
21
22         %get jacobian
23         jac = jacobian(f, var);
24         %evaluate jacobian matrix
25         j_value=double(subs(jac,var,prev));
26         %evaluate f
27         f_value=double(subs(f,var,prev));
28         %update result
29         result = prev - j_value\f_value;
30         %update error
31         error=100*abs((result-prev)./result);
32         %update prev
33         prev=result;
34     end
35     disp(result);
36
37 end

Command Window
>> q1
    0.5671
    0.5671

fx >>
```

For $f=[4x^2+y^2-13, x^2+y^2-10]$;

```
Editor - N:\2A\ECE204\Lab3\q1.m
q1.m x q2.m +
1 %define symbolic variables
2 syms x;
3 syms y;
4 syms var;
5 var=[x;y];
6 %define symbolic function
7 syms f(x,y);
8 %system of functions to be evaluated
9 %f=[2*x-y-exp(-x); -x+2*y-exp(-y)];
10 f=[4*x^2+y^2-13; x^2+y^2-10];
11 %initial value
12 init=[0.1;0.1];
13 %invoke the Newton Raphson function
14 newton(f,var,init);
15 function newton (f, var,init)
16     %set up prev
17     prev=init;
18     %set up intial values for error
19     error=ones(length(init),1);
20     %looping until satisfy error tolerance
21     while max(error)>0.5
22
23         %get jacobian
24         jac = jacobian(f, var);
25         %evaluate jacobian matrix
26         j_value=double(subs(jac,var,prev));
27         %evaluate f
28         f_value=double(subs(f,var,prev));
29         %update result
30         result = prev - j_value\f_value;
31         %update error
32         error=100*abs((result-prev)./result);
33         %update prev
34         prev=result;
35     end
36     disp(result);
37
38 end

Command Window
>> q1
    1.0000
    3.0000

fx >>
```

Q2. Using the information in example 9, in the set of notes titled “Roots of nonlinear equations”, find the temperature of an RTD that measure a resistance of:

75Ω and 250Ω

Use both bisection and Newton Raphson method. The iteration should stop when the relative approximate error is less than or equal 0.1%. Show the needed number of iterations of both methods. The program should be general and accept any resistance value.

Code:

```
syms T;
syms f1(T);
syms f2(T);
%take resistance
R = input('Enter the resistance value of R:');

f1= 5.775*(10^(-7))*T^2-3.9083*(10^(-3))*T+(R/100-1);
f2=5.775*(10^(-7))*T^2-3.9083*(10^(-3))*T+4.183*(10^(-12))*(T-100)*T^3+R/100-1;

%check which function to use depends on R
if R<100
    newrap(f2, -100,1);
    bisection(f2,-200,0,1);
else
    newrap(f1, 300,1);
    bisection(f1,0,850,1);
end

%bisection function
function bisection(f,init_1,init_2,counter)
    value_1=subs(f,init_1);
    value_2=subs(f,init_2);

    %update prev
    prev=init_1;

    %new root
    result=(init_1+init_2)/2;

    %get f(T) using new root
    value_new=subs(f,result);

    %recursively call bisection function until relative error < 0.1%
    if(abs((result-prev)/result)<=0.001)
        disp(['Bisection: ',num2str(double(result))]);
        disp(['Iterations using Bisection: ', num2str(counter)]);
    %update the range of the root
    elseif value_new*value_1<0
```

```

        %keep tracking of iterations
        counter = counter + 1;
        bisection(f,init_1,result,counter);
    elseif value_new*value_2<0
        counter = counter + 1;
        bisection(f,result,init_2,counter);
    end
end

%Newton Raphson function
function newrap(f, init, counter)
    %update prev
    prev = init;
    %differentiate the function
    df = diff(f);
    %new root
    new_T = double(prev - subs(f, prev)/subs(df, prev));

    %recursively call Newton Raphson function until relative error < 0.1%
    if(abs(new_T - prev) <= 0.001)
        disp(['Newton Raphson: ', num2str(double(new_T))]);
        disp(['Iterations using Newton Raphson: ', num2str(counter)]);
    else
        %keep tracking of iterations
        counter = counter + 1;
        newrap(f, new_T,counter);
    end
end

end

```

Display:

For R = 75Ω

Command Window

```

>> q2
Enter the resistance value of R:75
Newton Raphson: -63.3294
Iterations using Newton Raphson: 3
Bisection: -63.3301
Iterations using Bisection: 12

```

 >> |

For $R = 250\Omega$

Command Window

```
>> q2
```

```
Enter the resistance value of R:250
```

```
Newton Raphson: 408.45
```

```
Iterations using Newton Raphson: 3
```

```
Bisection: 408.606
```

```
Iterations using Bisection: 12
```

```
fx >> |
```

