GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

MATH214 NUMERICAL ANALYSIS

2020 – 2021 FALL TERM

Project - 1

Student's

Number : 200102002087

Name and Surname : Alican Bayındır

Preparation date: 26.10.2020 -> 28.10.2020

Upload date: 28.10.2020

Used formula;

$$\boxed{E(x) = \frac{1}{4\pi\varepsilon_0}\left[\frac{q_1(x-x_1)}{|x-x_1|^3} + \frac{q_2(x-x_2)}{|x-x_2|^3} + \frac{q_3(x-x_3)}{|x-x_3|^3} + \frac{q_4(x-x_4)}{|x-x_4|^3}\right]}$$

This formula is implemented into MATLAB as;

```
% Formula of electrical field
E = @(x) (1/4*pi*(1/36*pi)*10^-9)*((13*(x-(-7)/abs(x-(-7))^3)) ...
    + (9*(x-(-4)/abs(x-(-4))^3)) + (6*(x-11/abs(x-11)^3)) ...
    + (3*(x-14/abs(x-14)^3)));
```

However, this code works fine to handle the bisection method, Newton-Raphson method and the secant method as seen from figures 1, 2 and 3.

```
m_200102002087_bayindir_PR1_MATH214.m
COMMAND WINDOW
The bisection is -0.025062!
The bisection found 34. iteration!
```

Figure 1: The Bisection Method s result.

```
The procedure succeded in iteration 7. and the value is: -0.025062
```

Figure 2: The Newton-Raphson Method's result.

```
The procedure succeded in iteration 6. and the value is: -0.025062
```

Figure 3: The Secant Method's result.

And the table’s of all methods can be seen below as figures 4, 5, 6.

Iterations	Results	Errors
1	3.5	3.5
2	0.25	-3.25
3	-1.375	-1.625
4	-0.5625	0.8125
5	-0.15625	0.40625
6	0.046875	0.20313
7	-0.054688	-0.10156
8	-0.0039063	0.050781
9	-0.029297	-0.025391
10	-0.016602	0.012695
11	-0.022949	-0.0063477
12	-0.026123	-0.0031738
13	-0.024536	0.0015869
14	-0.02533	-0.00079346
15	-0.024933	0.00039673
16	-0.025131	-0.00019836
17	-0.025032	9.9182e-05
18	-0.025082	-4.9591e-05
19	-0.025057	2.4796e-05
20	-0.025069	-1.2398e-05
21	-0.025063	6.1989e-06
22	-0.02506	3.0994e-06
23	-0.025061	-1.5497e-06
24	-0.025062	-7.7486e-07
25	-0.025062	3.8743e-07
26	-0.025062	1.9372e-07
27	-0.025062	9.6858e-08
28	-0.025062	-4.8429e-08
29	-0.025062	2.4214e-08
30	-0.025062	1.2107e-08
31	-0.025062	6.0536e-09
32	-0.025062	-3.0268e-09
33	-0.025062	1.5134e-09
34	-0.025062	7.567e-10

Figure 4: The Result-Error table of bisection method.

```
The procedure succeded in iteration 6. and the value is: -0.025062
my_table =

  6×3 table

    Iterations     Results       Errors
    __________    _________    __________

        1               3.5           3.5
        2         -0.013754        3.5138
        3         -0.024903      0.011149
        4         -0.025059    0.00015611
        5         -0.025062       2.2e-06
        6         -0.025062    3.1006e-08
```

Figure 5: The Result-Error table of Newton-Raphson Method.

```
The procedure succeded in iteration 7. and the value is: -0.025062
my_table =

  7×3 table

    Iterations     Results       Errors
    __________    _________    ___________

        1                 0    -5.2289e-11
        2          -0.58203       -0.58203
        3           0.10329        0.68532
        4         -0.025774       -0.12907
        5         -0.025062     0.00071162
        6         -0.025062     7.0738e-07
        7         -0.025062    -4.1046e-12
```

Figure 6: The Result-Error table of The Secant Method.

As seen in figure 5 using Newton-Raphson's method is far more quick than others but the problem here is that my code does not work well. The reason for that might be the format of numbers. All 3 method's last iterations has some problems in results but error seems work well. The problem in Newton's method might be the derivative of the function but I tried different methods to differantiate it solve the problem. Unfortunately, none of them worked well and solved the problem. You can see my code in APPENDIX – A part and also the code will be delivered with this file to MS-Teams.

As specified in the project manual I changed tolerance value from 10^-10 to 10^-15 and the results can be seen below;

Iterations	Results	Errors
1	3.5	3.5
2	0.25	-3.25
3	-1.375	-1.625
4	-0.5625	0.8125
5	-0.15625	0.40625
6	0.046875	0.20313
7	-0.054688	-0.10156
8	-0.0039063	0.050781
9	-0.029297	-0.025391
10	-0.016602	0.012695
11	-0.022949	-0.0063477
12	-0.026123	-0.0031738
13	-0.024536	0.0015869
14	-0.02533	-0.00079346
15	-0.024933	0.00039673
16	-0.025131	-0.00019836
17	-0.025032	9.9182e-05
18	-0.025082	-4.9591e-05
19	-0.025057	2.4796e-05
20	-0.025069	-1.2398e-05
21	-0.025063	6.1989e-06
22	-0.02506	3.0994e-06
23	-0.025061	-1.5497e-06
24	-0.025062	-7.7486e-07
25	-0.025062	3.8743e-07
26	-0.025062	1.9372e-07
27	-0.025062	9.6858e-08
28	-0.025062	-4.8429e-08
29	-0.025062	2.4214e-08

Figure 7: The result of bisection method when tolerance value changed to 10^-15.

18	-0.025082	-4.9591e-05
19	-0.025057	2.4796e-05
20	-0.025069	-1.2398e-05
21	-0.025063	6.1989e-06
22	-0.02506	3.0994e-06
23	-0.025061	-1.5497e-06
24	-0.025062	-7.7486e-07
25	-0.025062	3.8743e-07
26	-0.025062	1.9372e-07
27	-0.025062	9.6858e-08
28	-0.025062	-4.8429e-08
29	-0.025062	2.4214e-08
30	-0.025062	1.2107e-08
31	-0.025062	6.0536e-09
32	-0.025062	-3.0268e-09
33	-0.025062	1.5134e-09
34	-0.025062	7.567e-10
35	-0.025062	-3.7835e-10
36	-0.025062	1.8917e-10
37	-0.025062	-9.4587e-11
38	-0.025062	4.7294e-11
39	-0.025062	2.3647e-11
40	-0.025062	1.1823e-11
41	-0.025062	5.9117e-12
42	-0.025062	2.9559e-12
43	-0.025062	-1.4779e-12
44	-0.025062	7.3896e-13
45	-0.025062	3.6948e-13
46	-0.025062	1.8474e-13
47	-0.025062	9.2371e-14
48	-0.025062	4.6185e-14
49	-0.025062	-2.3093e-14
50	-0.025062	-1.1546e-14
51	-0.025062	-5.7732e-15
52	-0.025062	2.8866e-15
53	-0.025062	-1.4433e-15
54	-0.025062	-7.2164e-16

Figure 8: The continued result of bisection method when tolerance value changed to 10^-15

Iterations	Results	Errors
1	3.5	3.5
2	-0.013754	3.5138
3	-0.024903	0.011149
4	-0.025059	0.00015611
5	-0.025062	2.2e-06
6	-0.025062	3.1006e-08
7	-0.025062	4.37e-10
8	-0.025062	6.159e-12
9	-0.025062	8.6795e-14
10	-0.025062	1.2247e-15

Figure 9: The result of Newton's method when tolerance changed to 10^-15.

8×3 table

Iterations	Results	Errors
1	0	-5.2289e-12
2	-0.58203	-0.58203
3	0.10329	0.68532
4	-0.025774	-0.12907
5	-0.025062	0.00071162
6	-0.025062	7.0738e-07
7	-0.025062	-4.1046e-12
8	-0.025062	3.4694e-18

Figure 10: The result of Secant method when tolerance changed to 10^-15.

So, as seen in figures 7, 8, 9 and 10 we got closer to the result and error of the procedures became smaller. In the first result of secant method should not be 0 but there is an error here again. In order to approach to the result we need to make our tolerance smaller it causes increase of iterations but it makes our results more precise. Again, the problem in my results segment in table might be because of the format of the numbers. I could not change it because my MATLAB frequently crushes.

As specified in the project manual I changed tolerance value from 10^-10 to 10^-16 and the results can be seen below;

Iterations	Results	Errors
1	3.5	3.5
2	0.25	-3.25
3	-1.375	-1.625
4	-0.5625	0.8125
5	-0.15625	0.40625
6	0.046875	0.20313
7	-0.054688	-0.10156
8	-0.0039063	0.050781
9	-0.029297	-0.025391
10	-0.016602	0.012695
11	-0.022949	-0.0063477
12	-0.026123	-0.0031738
13	-0.024536	0.0015869
14	-0.02533	-0.00079346
15	-0.024933	0.00039673
16	-0.025131	-0.00019836
17	-0.025032	9.9182e-05
18	-0.025082	-4.9591e-05
19	-0.025057	2.4796e-05
20	-0.025069	-1.2398e-05
21	-0.025063	6.1989e-06
22	-0.02506	3.0994e-06
23	-0.025061	-1.5497e-06
24	-0.025062	-7.7486e-07
25	-0.025062	3.8743e-07
26	-0.025062	1.9372e-07
27	-0.025062	9.6858e-08
28	-0.025062	-4.8429e-08
29	-0.025062	2.4214e-08
30	-0.025062	1.2107e-08

Figure 12: The result of bisection method when tolerance value changed to 10^-16

26	-0.025062	1.9372e-07
27	-0.025062	9.6858e-08
28	-0.025062	-4.8429e-08
29	-0.025062	2.4214e-08
30	-0.025062	1.2107e-08
31	-0.025062	6.0536e-09
32	-0.025062	-3.0268e-09
33	-0.025062	1.5134e-09
34	-0.025062	7.567e-10
35	-0.025062	-3.7835e-10
36	-0.025062	1.8917e-10
37	-0.025062	-9.4587e-11
38	-0.025062	4.7294e-11
39	-0.025062	2.3647e-11
40	-0.025062	1.1823e-11
41	-0.025062	5.9117e-12
42	-0.025062	2.9559e-12
43	-0.025062	-1.4779e-12
44	-0.025062	7.3896e-13
45	-0.025062	3.6948e-13
46	-0.025062	1.8474e-13
47	-0.025062	9.2371e-14
48	-0.025062	4.6185e-14
49	-0.025062	-2.3093e-14
50	-0.025062	-1.1546e-14
51	-0.025062	-5.7732e-15
52	-0.025062	2.8866e-15
53	-0.025062	-1.4433e-15
54	-0.025062	-7.2164e-16
55	-0.025062	-3.6082e-16
56	-0.025062	-1.8041e-16
57	-0.025062	9.0206e-17

Figure 11: The continued result of bisection method when tolerance value changed to 10^-16

Iterations	Results	Errors
1	3.5	3.5
2	-0.013754	3.5138
3	-0.024903	0.011149
4	-0.025059	0.00015611
5	-0.025062	2.2e-06
6	-0.025062	3.1006e-08
7	-0.025062	4.37e-10
8	-0.025062	6.159e-12
9	-0.025062	8.6795e-14
10	-0.025062	1.2247e-15

Figure 13: The result of Newton's method when tolerance changed to 10^-16.

8×3 table

Iterations	Results	Errors
1	0	-5.2289e-12
2	-0.58203	-0.58203
3	0.10329	0.68532
4	-0.025774	-0.12907
5	-0.025062	0.00071162
6	-0.025062	7.0738e-07
7	-0.025062	-4.1046e-12
8	-0.025062	3.4694e-18

Figure 14: The result of Secant method when tolerance changed to 10^-16.

The bisection method did more calculation than before since we increased the tolerance of the algorithm to be more precise. On the other hand, other algorithms did not change. I guess my algorithm is not succesfull to use. The algorithm failed to calculate the results. As seen above in figures 12 to 14. I used ONLINE MATLAB (can be seen in the pictures in APPENDIX part) to code these three algorithms since I was not able to use downloaded version of MATLAB.

APPENDIX

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```
%% Alican Bayındır started: 26.10.2020 - finished: 28.10.2020
% Project - 1 MATH214
% This script contains the Bisection method, the Newton's method
% and the Secant method
clear all
clc
% 1st Bisection method

% Formula of electrical field
E = @(x) (1/4*pi*(1/36*pi)*10^-9)*((13*(x-(-7)/abs(x-(-7))^3)) ...
     + (9*(x-(-4)/abs(x-(-4))^3)) + (6*(x-11/abs(x-11)^3)) ...
     + (3*(x-14/abs(x-14)^3)));

% The closed interval is [-3,10] so a = -3, b = 10;
x_lower = -3;
x_upper = 10;

%tolerance value that is given by project guide
tol = 10^-9;

%c provides us loop
c = 0;
i = 1;
FA = E(x_lower);
```

COMMAND WINDOW

m_200102002087_bayindir_PR1_MATH214.m

```
i = 1;
FA = E(x_lower);

% Bisection algorithm
while c < 1
    % Finding middle point and assigning it to a variable p
    p = x_lower + (x_upper - x_lower) / 2;
    FP = E(p);

    % The part below is to handle the table
    i_for_table = (1:i); % iterations
    p_for_table(i) = p; % results

    %if the result found
    if (FP == 0 || (x_upper - x_lower) / 2 < tol)
        fprintf('The bisection is %f!\n', p);
        fprintf('The bisection found %d. iteration! \n', i);
        % To a proper result for errors, I needed to change the format
        format shortEng
        o = 1;
        % calculating and creating errors' array
        while o <= i
            if o == 1
            error(1) = p_for_table(1) - E(0);
```

COMMAND WINDOW

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```matlab
        while o <= i
            if o == 1
            error(1) = p_for_table(1) - E(0);
            else
            error(o) = p_for_table(o) - p_for_table(o - 1);
            end
        o = o + 1;
        end
        %Creating table
        my_table = table(i_for_table', p_for_table', error', ...
            'VariableNames',{'Iterations' 'Results' 'Errors'})
        return
    end
    %if not found initialize variables and add 1 to iteration.
    i = i + 1;

    %Initializing variables to find new middle point
    if FA*FP > 0
        x_lower = p;
        FA = FP;
    else
        x_upper = p;
    end
end
```

COMMAND WINDOW

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```matlab
%% 2nd Newton-Raphson method
clear all
clc

% Formula of electrical field
E = @(x) (1/4*pi*(1/36*pi)*10^-9)*((13*(x-(-7)/abs(x-(-7))^3)) ...
    + (9*(x-(-4)/abs(x-(-4))^3)) + (6*(x-11/abs(x-11)^3)) ...
    + (3*(x-14/abs(x-14)^3)));

% Tolerance value
tol = 10^-9;
c = 0;
i = 1;
% p0 is the initial approximation value. I am going to choose
% mid point of [-3,10] interval
p0 = 3.5;
% Derivated version of E
E_derivated = (262495630976900031*sign(p0 - 11))/(193428131138340667952988 16* ...
    abs(p0 - 11)^4) - (72385158845145237*sign(p0 + 7))/(386856262276681335905976 32* ...
    abs(p0 + 7)^4) - (7158971753915463*sign(p0 + 4))/(96714065569170333976494 08* ...
    abs(p0 + 4)^4) + (16704267425802747*sign(p0 - 14))/(193428131138340667952988 16* ...
    abs(p0 - 14)^4) + 8219560161902939/38685626227668133590597632;

% Newton - Raphson method's algorithm
```

COMMAND WINDOW

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```matlab
% Newton - Raphson method's algorithm
while c < 1
    p = p0 - E(p0) / E_derivated;

    % The part below is to handle the table
    i_for_table = (1:i); %iteration' array
    p_for_table(i) = p0; %results' array
        % if result found
        if abs(p - p0) < tol
        fprintf('The procedure succeded in iteration %d. and the value is: %f', i, p);
        % to show proper errors, we need to change the format
        format shortEng
        o = 1;
        % editing errors' array
        while o <= i
            if o == 1
            error(1) = abs(p_for_table(1) - E(0));
            else
            error(o) = abs(p_for_table(o) - p_for_table(o - 1));
            end
        o = o + 1;
        end
        %create table
        my_table = table(i_for_table', p_for_table', error','VariableNames',{'Iterations' 'Results' 'Errors'})
```

COMMAND WINDOW

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```matlab
            %create table
            my_table = table(i_for_table', p_for_table', error','VariableNames'
            return
            end
        %initialize variables
        i = i + 1;
        p0 = p;
end
%% 3rd The Secant Method
clear all
clc

% Formula of electrical field
E = @(x) (1/4*pi*(1/36*pi)*10^-9)*((13*(x-(-7)/abs(x-(-7))^3)) ...
    + (9*(x-(-4)/abs(x-(-4))^3)) + (6*(x-11/abs(x-11)^3)) ...
    + (3*(x-14/abs(x-14)^3)));

%tolerance value
tol = 10^-9;
c = 0;
i = 2;

% upper point and lower point
p0 = -3;
```

COMMAND WINDOW

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```
% upper point and lower point
p0 = -3;
p1 = 10;

q0 = E(p0);
q1 = E(p1);

%The secant method's algorithm
while c < 1
    p = p1 - q1 * (p1-p0) / (q1-q0);

    % The part below is to handle the table
    i_for_table = (1:i); %iterations
    p_for_table(i) = p; %results

    %if results found
    if abs(p-p1) < tol
        fprintf('The procedure succeded in iteration %d. and the value is: %f', i, p)
        % to show proper errors we need to change the format
        format shortEng
        o = 1;
        %editing errors' array
        while o <= i
```

COMMAND WINDOW

MATLAB Drive

m_200102002087_bayindir_PR1_MATH214.m

```
        fprintf('The procedure succeded in iteration %d. and the value is: %f', i, p)
        % to show proper errors we need to change the format
        format shortEng
        o = 1;
        %editing errors' array
        while o <= i
            if o == 1
            error(1) = p_for_table(1) - E(0);
            else
            error(o) = p_for_table(o) - p_for_table(o - 1);
            end
        o = o + 1;
        end
        %create table
        my_table = table(i_for_table', p_for_table', error', ...
            'VariableNames',{'Iterations' 'Results' 'Errors'})
        return
    end
    %initialize variables
    i = i + 1;
    p0 = p1;
    q0 = q1;
    p1 = p;
    q1 = E(p);
```

COMMAND WINDOW