

ES – 304
Numerical Analysis
Assignment # 3

Name: Syed Muhammad Ashhar Shah

Reg No: 2020478

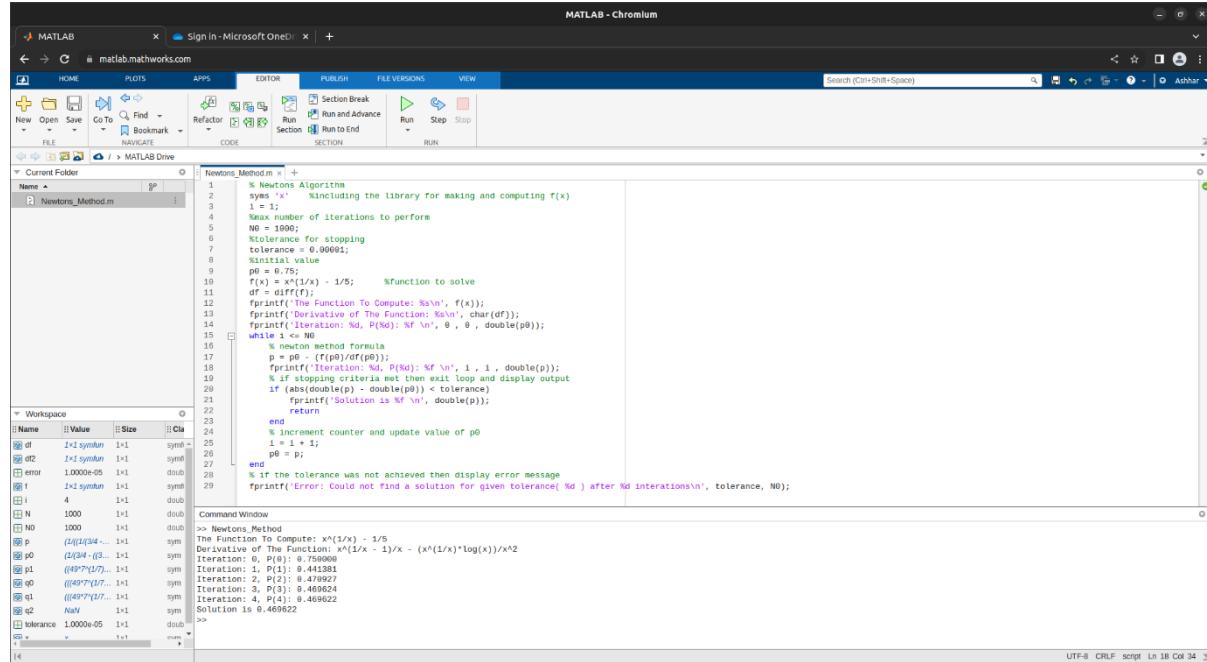
Section: B-CS



$$\text{Q1. } F(x) = x^{(1/x)} - 1/5, \quad [0,2]$$

Root: 0.469622

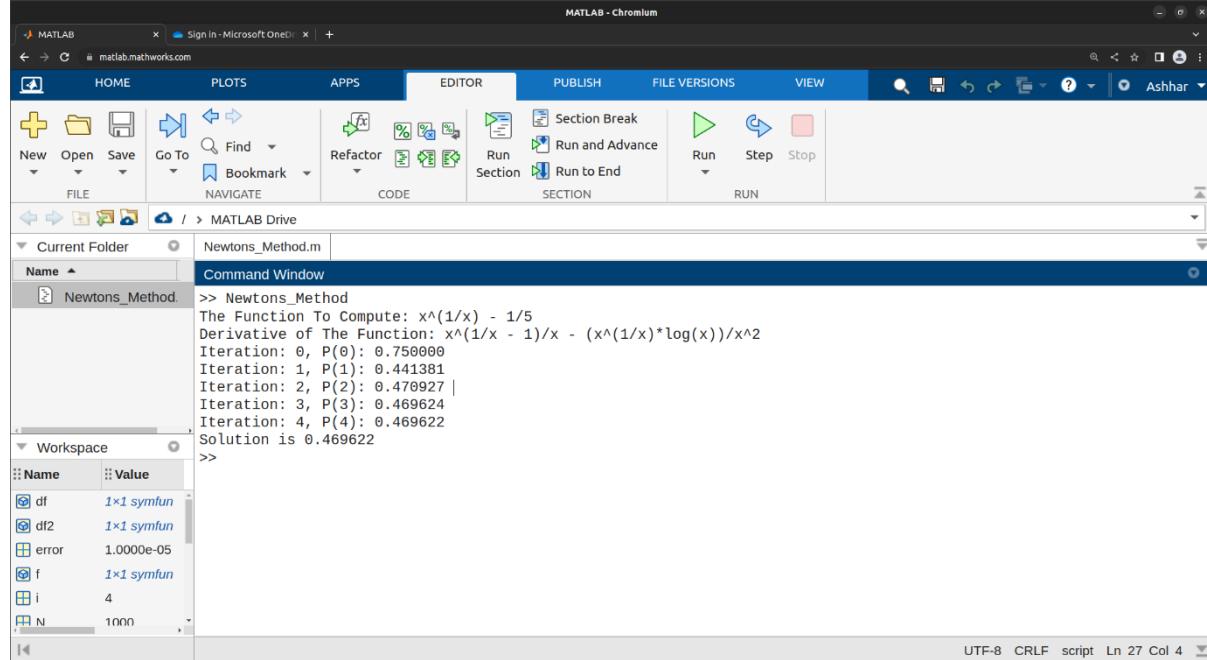
Using Newtons Method:



```

MATLAB - Chromium
MATLAB | Sign in - Microsoft OneDrive | matlab.mathworks.com
HOME PLOTS APPS EDITOR PUBLISH FILE VERSIONS VIEW
New Open Save Go To Find Bookmarks Refactor Run Section Run and Advance Run Step Stop
FILE NAVIGATE CODE SECTION RUN
Current Folder Name Newtons_Method.m
1 % Newtons Algorithm
2 % Including the library for making and computing f(x)
3 % Number of iterations to perform
4 NO = 1000;
5 % Tolerance for stopping
6 tol = 1e-000001;
7 % Initial value
8 p0 = 0.75;
9
10 f(x) = x^(1/x) - 1/5; % Function to solve
11 df = diff(f(x));
12 df2 = diff(df);
13 derror = 1.0000e-05;
14
15 while i <= NO
16     % Newtons Method formula
17     p = p0 - (f(p0)/df(p0));
18     fprintf('Iteration: %d, P(%d): %f\n', i, double(p));
19     % If stopping criteria met then exit loop and display output
20     if (abs(double(p) - double(p0)) < tolerance)
21         fprintf('Solution is %f\n', double(p));
22         return
23     end
24     % Increment counter and update value of p0
25     i = i + 1;
26     p0 = p;
27
28 % If the tolerance was not achieved then display error message
29 fprintf('Error: Could not find a solution for given tolerance (%d) after %d iterations\n', tolerance, NO);
30
31 end
32
33 Command Window
>> Newtons_Method
The Function To Compute: x^(1/x) - 1/5
Derivative of The Function: x^(1/x - 1)/x - (x^(1/x)*log(x))/x^2
Iteration: 0, P(0): 0.750000
Iteration: 1, P(1): 0.441381
Iteration: 2, P(2): 0.470927 |
Iteration: 3, P(3): 0.469624
Iteration: 4, P(4): 0.469622
Solution is 0.469622
34

```



```

MATLAB - Chromium
MATLAB | Sign in - Microsoft OneDrive | matlab.mathworks.com
HOME PLOTS APPS EDITOR PUBLISH FILE VERSIONS VIEW
New Open Save Go To Find Bookmarks Refactor Run Section Run and Advance Run Step Stop
FILE NAVIGATE CODE SECTION RUN
Current Folder Name Newtons_Method.m
Command Window
>> Newtons_Method
The Function To Compute: x^(1/x) - 1/5
Derivative of The Function: x^(1/x - 1)/x - (x^(1/x)*log(x))/x^2
Iteration: 0, P(0): 0.750000
Iteration: 1, P(1): 0.441381
Iteration: 2, P(2): 0.470927 |
Iteration: 3, P(3): 0.469624
Iteration: 4, P(4): 0.469622
Solution is 0.469622
>>

```

Using Modified Newton's Method:

The screenshot shows the MATLAB Editor window with the script file 'Modified_Newtons_Method.m' open. The code implements the Modified Newton's Method to find the root of the function $f(x) = x^{1/x} - 1/5$. The script includes comments explaining the steps: defining the function, setting initial values, calculating derivatives, and performing iterations until convergence or reaching a maximum number of iterations (NO).

```

% Modified Newtons Algorithm
syms 'x'
% including the library for making and computing f(x)
% max number of iterations to perform
NO = 1000;
% tolerance for stopping
tolerance = 0.00001;
% initial value
p0 = 0.5;
% first initial guess
f(x) = x^(1/x) - 1/5;
df = diff(f);
df2 = diff(df);
fprintf('The Function To Compute: %s\n',f(x));
fprintf('Derivative of The Function: %s\n', char(df));
fprintf('2nd Derivative of The Function: %s\n', char(df2));
fprintf('Iteration: %d, P(%d): %f \n', 0 , 0 , double(p0));
while i <= NO
    % modified newton method formula
    p = p0 - ((p0)*df(p0)) / ((df(p0))^2);
    fprintf('Iteration: %d, P(%d): %f \n', i , 1 , double(p));
    % if stopping criteria met then exit loop and display output
    if (abs(double(p) - double(p0)) < tolerance)
        fprintf('Solution is %f \n', double(p));
        fprintf('Number of Iterations Done: %d ', i);
        return;
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p;
end
% if the tolerance was not achieved then display error message
fprintf('Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n', tolerance, NO);

```

The Command Window shows the execution of the script and the resulting output, which includes the function definition, derivative, 2nd derivative, iteration details, and the final solution.

The screenshot shows the MATLAB Command Window displaying the output of the script 'Modified_Newtons_Method.m'. The output provides the function definition, its first and second derivatives, and the iterative process for finding the root. The final output shows the solution is 0.469622 and the number of iterations done is 4.

```

>> Modified_Newtons_Method
The Function To Compute: x^(1/x) - 1/5
Derivative of The Function: x^(1/x - 1)/x - (x^(1/x)*log(x))/x^2
2nd Derivative of The Function: (x^(1/x - 2)*(1/x - 1) - (x^(1/x - 1)*log(x))/x^2)/x - x^(1/x - 1)*x^2 - x^(1/x - 1)/x^2 - x^(1/x - 1)
Iteration: 0, P(0): 0.500000
Iteration: 1, P(1): 0.469013
Iteration: 2, P(2): 0.469621
Iteration: 3, P(3): 0.469621
Iteration: 4, P(4): 0.469622
Solution is 0.469622
Number of Iterations Done: 4
>>

```

Using Secant Method:

The screenshot shows the MATLAB Editor window with the script `Secant_Method.m` open. The code implements the Secant method for solving the equation $x^{1/x} - 1/5 = 0$. It starts by defining the function $f(x) = x^{1/x} - 1/5$, setting initial values $p_0 = 1$ and $p_1 = 1.5$, and specifying a tolerance of 0.00001 . The loop iterates until the absolute difference between successive approximations is less than the tolerance. The command window at the bottom shows the iterations and the final result.

```

% Secant Algorithm
% Including the library for making and computing f(x)
syms 'x'
i = 2;
% max number of iterations to perform
N0 = 1000;
%tolerance for stopping
tolerance = 0.00001;
% initial value
p0 = 1; % first initial guess
p1 = 1.5; % second initial guess
f(x) = x^(1/x) - 1/5;
fprintf('The Function To Compute: \n',f(x));
fprintf('Iteration: %d, P(%d): %f \n', 0 , 0 , double(p0));
for i=2:N0
    p2 = p1 - (q1 * (p1 - p0)) / (q1 - q0);
    if abs(p2-p1) < tolerance
        fprintf('Solution is %f \n', double(p2));
        fprintf('Number of Iterations Done: %d \n', i);
        return
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p1;
    p1 = p2;
    q0 = q1;
    q1 = f(p1);
end
% if the tolerance was not achieved then display error message
if (q1 > tolerance) & (double(p1) < tolerance)
    fprintf('Error: Could not find a solutions for given tolerance( %d ) after %d iterations\n', tolerance, N0);
end

```

The screenshot shows the MATLAB Command Window with the script `Secant_Method.m` running. The output displays the iterative steps of the Secant method, starting from the initial guess $p_0 = 1$ and $p_1 = 1.5$, and converging to the solution $x = 0.469622$ after 6 iterations.

```

>> Secant_Method
The Function To Compute: x^(1/x) - 1/5
Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 0.500000
Iteration: 2, P(2): 0.466667
Iteration: 3, P(3): 0.469524
Iteration: 4, P(4): 0.469622
Iteration: 5, P(5): 0.469622
Iteration: 6, P(6): 0.469622
Solution is 0.469622
Number of Iterations Done: 6
>>

```

$$\text{Q2. } F(x) = e^x - 2^{-x} + 2\cos(x) - 6, [0, 2]$$

Root: 1.944462

Using Newton's Method:

The screenshot shows the MATLAB IDE interface. The current folder contains the script `Newtons_Method.m`. The workspace shows variables `df`, `f`, `i`, `N`, `NO`, `p0`, `p`, `tolerance`, and `x`. The command window displays the execution of the script, showing the iterative process of the Newton-Raphson method until convergence at `x = 1.944462`.

```

MATLAB - Chromium
Newtons_Method.m | +
1 % Modified Newtons Algorithm
2 syms 'x' %Including the library for making and computing T(x)
3 i = 1;
4 % max number of iterations to perform
5 NO = 1000;
6 %Tolerance for stopping
7 tolerance = 0.00001;
8 % initial value (Pi/2 / 2)
9 p0 = 1;
10 f(x) = exp(x) - 2^(-x) + 2*cos(x) - 6;
11 df(x) = diff(f);
12 fprintf('The Function To Compute: %s\n',f(x));
13 fprintf('Derivative of The Function: %s\n', char(df));
14 fprintf('Iterations: %d, P(%d): %f \n', 0, 0, double(p0));
15 while i < NO
16     %Newton's method formula
17     p = p0 - (f(p0)/df(p0));
18     fprintf('Iteration: %d, P(%d): %f \n', i , 1 , double(p));
19     % if stopping criteria met then exit loop and display output
20     if abs(p-p0) < tolerance
21         fprintf('Solution is %f \n', double(p));
22         fprintf('Number of Iterations Done: %d \n', i);
23         return
24     end
25     % increment counter and update value of p0
26     i = i + 1;
27     p0 = p;
28 end
29 % if the tolerance was not achieved then display error message
30 fprintf('Error: Could not find a solutions for given tolerance( %d ) after %d iterations\n', tolerance, NO);

>> Newtons_Method
The Function To Compute: 2*cos(x) + exp(x) - 1/2^x - 6
Derivative of The Function: exp(x) - 2^sin(x) + log(2)/2^x
Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 2.954618
Iteration: 2, P(2): 2.367677
Iteration: 3, P(3): 2.043683
Iteration: 4, P(4): 1.951074
Iteration: 5, P(5): 1.944494
Iteration: 6, P(6): 1.944462
Iteration: 7, P(7): 1.944462
Solution is 1.944462
Number of Iterations Done: 7

```

This screenshot continues the execution of the `Newtons_Method.m` script. The command window shows the final iteration where the tolerance was not reached, resulting in an error message. The workspace variables remain the same as in the previous screenshot.

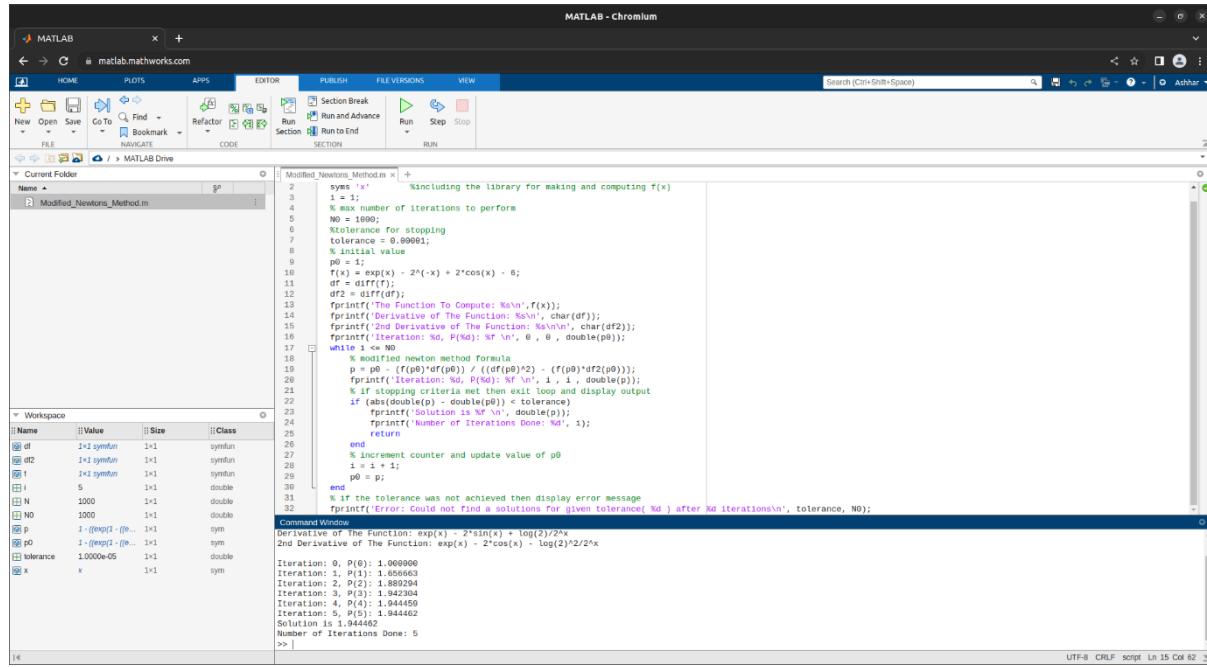
```

MATLAB - Chromium
Newtons_Method.m | +
27 p0 = p;
28 end
29 % if the tolerance was not achieved then display error message
30 fprintf('Error: Could not find a solutions for given tolerance( %d ) after %d iterations\n', tolerance, NO);

>> Newtons_Method
The Function To Compute: 2*cos(x) + exp(x) - 1/2^x - 6
Derivative of The Function: exp(x) - 2^sin(x) + log(2)/2^x
Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 2.954618
Iteration: 2, P(2): 2.367677
Iteration: 3, P(3): 2.043683
Iteration: 4, P(4): 1.951074
Iteration: 5, P(5): 1.944494
Iteration: 6, P(6): 1.944462
Iteration: 7, P(7): 1.944462
Solution is 1.944462
Number of Iterations Done: 7
>> |


```

Using Modified Newton's Method:



The screenshot shows the MATLAB IDE interface with the following details:

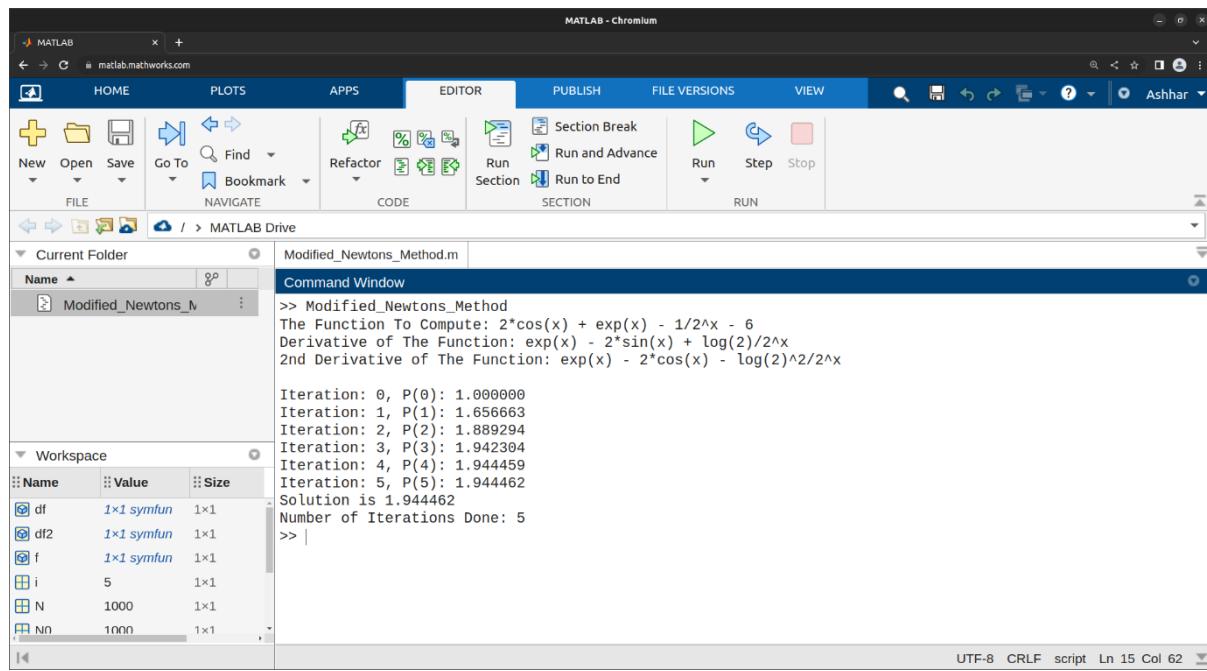
- Editor Tab:** Contains the script file `Modified_Newtons_Method.m` with the following code:

```

1 %including the library for making and computing f(x)
2 % max number of iterations to perform
3 N = 1000;
4 tolerance = 0.00001;
5 % initial value
6 p0 = 1;
7 f = @(x) exp(x) - 2*(x) + 2*cos(x) - 6;
8 df = diff(f);
9 df2 = diff(df);
10 fprintf('The Function To Compute: %s\n',f);
11 fprintf('1st Derivative of The Function: %s\n',df);
12 fprintf('2nd Derivative of The Function: %s\n',df2);
13 fprintf('Iteration: %d, P(%d): %f\n', 1, 1, double(p0));
14 fprintf('Number of Iterations Done: %d\n', 1);
15 while 1 < N
16     p = p0 - ((p0)*(df(p0))) / ((df(p0))^2 - (f(p0)*(df2(p0))));
17     fprintf('Iteration: %d, P(%d): %f\n', i, i, double(p));
18     if (abs(double(p) - double(p0)) < tolerance)
19         fprintf('Solution is %f\n', double(p));
20         fprintf('Number of Iterations Done: %d\n', i);
21         return
22     end
23     % increment counter and update value of p0
24     i = i + 1;
25     p0 = p;
26 end
27 % if stopping criteria met then exit loop and display output
28 if (abs(double(p) - double(p0)) < tolerance)
29     fprintf('Solution is %f\n', double(p));
30 else
31     fprintf('The tolerance was not achieved then display error message');
32 end
33 fprintf('Error: Could not find a solutions for given tolerance( %d ) after %d iterations\n', tolerance, NO);

```

- Command Window:** Displays the execution results and iterations.



The screenshot shows the MATLAB IDE interface with the following details:

- Editor Tab:** Contains the script file `Modified_Newtons_Method.m` with the following code:

```

>> Modified_Newtons_Method
The Function To Compute: 2*cos(x) + exp(x) - 1/2^x - 6
Derivative Of The Function: exp(x) - 2*sin(x) + log(2)/2^x
2nd Derivative Of The Function: exp(x) - 2*cos(x) - log(2)^2/2^x

Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 1.656663
Iteration: 2, P(2): 1.889294
Iteration: 3, P(3): 1.942304
Iteration: 4, P(4): 1.944459
Iteration: 5, P(5): 1.944462
Solution is 1.944462
Number of Iterations Done: 5
>> |

```

Using Secant Method:

The screenshot shows the MATLAB IDE interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, FILE VERSIONS, and VIEW. The current window is the EDITOR tab, displaying the code for 'Secant_Method.m'. The code implements the Secant Method to solve the equation $2\cos(x) + \exp(x) - 1/2^x - 6 = 0$. It defines symbols for df, df2, f, and i. It sets initial values p0 = 1.5 and p1 = 1.75. The script then enters a loop where it calculates intermediate values q0, q1, and q2, and updates p0 and p1. It prints iteration details and exits the loop if the tolerance is met or reaches a maximum of 1000 iterations. The workspace table shows variables df, df2, f, i, NO, n, and x.

```

% Secant Algorithm
syms 'x' %including the library for making and computing f(x)
i=1; % max number of iterations to perform
NO = 1000;
%tolerance for stopping
tolerance = 0.0001;
q0 = 0;
%initial values
p0 = 1.5; % first initial guess
p1 = 1.75; % second initial guess
f(x) = exp(x) - 2^x - 2*cos(x);
for i=1:NO
    fval = f(x);
    fprintf('Iteration: %d, P(%d): %f\n', i, 0, double(p0));
    fprintf('Iteration: %d, P(%d): %f\n', i, 1, double(p1));
    q0 = f(p0);
    q1 = f(p1);
    q2 = q0*(p1-p0)/(q1-q0);
    if abs(double(p1)-double(p0)) < tolerance
        fprintf('Solution is %f', double(p1));
        fprintf('Number of Iterations Done: %d', i);
        return;
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p1;
    p1 = q2;
    q0 = q1;
    q1 = f(p1);
end
% if the tolerance was not achieved then display error message

```

The screenshot shows the MATLAB IDE interface. The top menu bar includes HOME, PLOTS, APPS, EDITOR, PUBLISH, FILE VERSIONS, and VIEW. The current window is the COMMAND WINDOW tab. The command `>> Secant_Method` is run, and the output shows the iterative process of the Secant Method to find the root of the equation. The solution is found to be 1.944462 after 7 iterations. The workspace table shows variables df, df2, f, i, NO, n, and x.

```

>> Secant_Method
The Function To Compute: 2*cos(x) + exp(x) - 1/2^x - 6
Iteration: 0, P(0): 1.500000
Iteration: 1, P(1): 1.750000
Iteration: 2, P(2): 2.020450
Iteration: 3, P(3): 1.933520
Iteration: 4, P(4): 1.943880
Iteration: 5, P(5): 1.944467
Iteration: 6, P(6): 1.944462
Iteration: 7, P(7): 1.944462
Solution is 1.944462
Number of Iterations Done: 7
>>

```

$$Q3. F(x) = (x+2) * (x+1)^2 * (x-2), \quad [0,2]$$

Root: 2.00000

Using Newton's Method:

The screenshot shows the MATLAB IDE interface. The code in the editor is as follows:

```

% Newton's Method x = -3
% Newton's Method Algorithm
% including the library for making and computing f(x)
i = 1;
% max number of iterations to perform
N = 1000;
% tolerance for stopping
tolerance = 0.00001;
% initial value
p0 = -5;
f = @(x) (x+2) * (x+1)^2 * (x-2);
df = diff(f);
for i = 1:N
    % Newton method formula
    p = p0 - f(p0)/df(p0);
    if abs(double(p) - double(p0)) < tolerance
        fprintf('Solution is %f\n', double(p));
        fprintf('Number of Iterations Done: %d', i);
        return
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p;
end
% if the tolerance was not achieved then display error message
fprintf('Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n', tolerance, N);

```

The command window output shows the iteration process:

```

The Function To Compute: (x + 1)^2*(x - 2)*(x + 2)
Derivative of The Function: (x + 1)^2*(x - 2) + (x + 1)^2*(x + 2) + (2*x + 2)*(x - 2)*(x + 2)
Iteration: 0, P(0): 1.500000
Iteration: 1, P(1): 2.593750
Iteration: 2, P(2): 2.186985
Iteration: 3, P(3): 2.026069
Iteration: 4, P(4): 2.000064
Iteration: 5, P(5): 2.000000
Iteration: 6, P(6): 2.000000
Solution is 2.000000
Number of Iterations Done: 6
>>

```

The screenshot shows the MATLAB IDE interface. The code in the editor is as follows:

```

>> Newtons_Method
The Function To Compute: (x + 1)^2*(x - 2)*(x + 2)
Derivative of The Function: (x + 1)^2*(x - 2) + (x + 1)^2*(x + 2) + (2*x + 2)*(x - 2)*(x + 2)
Iteration: 0, P(0): 1.500000
Iteration: 1, P(1): 2.593750
Iteration: 2, P(2): 2.186985
Iteration: 3, P(3): 2.026069
Iteration: 4, P(4): 2.000064
Iteration: 5, P(5): 2.000000
Iteration: 6, P(6): 2.000000
Solution is 2.000000
Number of Iterations Done: 6
>> |

```

Using Modified Newton's Method:

The screenshot shows the MATLAB interface with the following details:

- Editor Tab:** The current file is `Modified_Newtons_Method.m`. The code implements the Modified Newton's Method for the function $f(x) = (x+2)^2 \cdot (x+1)^2 \cdot (x-2)$.
- Code:**

```
% Modified Newton's Method
% Including the library for making and computing f(x)
syms 'x'
1 = 1;
% max number of iterations to perform
N0 = 1000;
% tolerance for stopping
tolerance = 0.00001;
% initial value
p0 = -1;
f(x) = (x+2)^2 * (x+1)^2 * (x-2);
df = diff(f);
df2 = diff(df);
fprintf('The Function To Compute: %a\n', f(x));
fprintf('1st Derivative of The Function: %a\n', char(df));
fprintf('2nd Derivative of The Function: %a\n', char(df2));
fprintf('Iteration: %d, P(%d): %f\n', 0, 0, double(p0));
for i = 1:N0
    p = p0 - ((f(p0)*df(p0)) / ((df(p0))^2 - (f(p0)*df2(p0))));
    fprintf('Iteration: %d, P(%d): %f\n', i, 1, double(p));
    if (abs(p - p0) < tolerance)
        fprintf('Solution is %f\n', double(p));
        fprintf('Number of Iterations Done: %d', i);
        return;
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p;
end
% if the tolerance was not achieved then display error message
fprintf('Error: Could not find a solutions for given tolerance( %d ) after %d iterations\n', tolerance, N0);

```
- Workspace:** Shows variables `df`, `df2`, `f`, `i`, `N`, `N0`, `p`, `p0`, `tolerance`, and `x`.
- Command Window:** Displays the execution of the script and the resulting iterations.

The screenshot shows the MATLAB interface with the following details:

- Editor Tab:** The current file is `Modified_Newtons_Method.m`. The code is identical to the one in the first screenshot.
- Code:** The same script code is shown.
- Command Window:** Shows the command `>> Modified_Newtons_Method` and the resulting output of the iterations.
- Workspace:** Shows the same set of variables as the first screenshot.

Using Secant Method:

MATLAB - Chromium

File Home Plots Apps Editor Publish File Versions View

Search (Ctrl+Shift+Space)

New Open Save Go To Find Bookmarks Refactor Navigate

FILE CODE SECTION RUN

Current Folder

Secant_Method.m

```
% Secant Method
% Secant Algorithm
% including the library for making and computing f(x)
% max number of iterations to perform
N0 = 1000;
%Tolerance for stopping
tolerance = 0.00001;
% initial value
p0 = 1.5; % First initial guess
p1 = 1.75; % Second initial guess
f0 = f(p0); f1 = f(p1);
printf("The Function To Solve: \n",f(x));
printf("Iteration: %d, P(0): %f\n", 0 , 0 , double(p0));
printf("Iteration: %d, P(1): %f\n", 1 , 1 , double(p1));
q0 = f(p0);
q1 = f(p1);
while 1 <= N0
    % secant method formula
    p = p1 - (q1 * (p1 - p0)) / (q1 - q0);
    printf("Iteration: %d: %f\n", i , i , double(p));
    % if stopping criteria met then exit loop and display output
    if(abs(double(p) - double(p0)) < tolerance)
        printf("Solution is %f\n", double(p));
        printf("Number of Iterations done: %d\n");
        return;
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p1;
    p1 = p;
    q0 = q1;
    q1 = f(p);
end
% if the tolerance was not achieved then display error message
printf("Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n", tolerance, N0);
```

Command Window

```
Iteration: 5, P(5): 2.000000
Iteration: 6, P(6): 1.999997
Iteration: 7, P(7): 2.000000
Iteration: 8, P(8): 2.000000
Solution is 2.000000
Number of Iterations Done: 8
>>
```

The screenshot shows the MATLAB interface in a browser window titled "MATLAB - Chromium". The top menu bar includes HOME, PLOTS, APPS, EDITOR (selected), PUBLISH, FILE VERSIONS, and VIEW. Below the menu is a toolbar with icons for New, Open, Save, Go To, Find, Refactor, Section Break, Run Section, Run and Advance, Run to End, Step, and Stop. The left sidebar shows the Current Folder path as "MATLAB Drive / Current Folder" and contains files "Secant_Method.m" and "Secant_Method.rw". The workspace table lists variables: df (1x1 symfun), df2 (1x1 symfun), f (1x1 symfun), i (8, 1x1), NO (1000, 1x1), and n (146505654, 1x1). The Command Window displays the following code and output:

```
>> Secant_Method
The Function To Compute: (x + 1)^2*(x - 2)*(x + 2)
Iteration: 0, P(0): 1.500000
Iteration: 1, P(1): 1.750000
Iteration: 2, P(2): 2.210660
Iteration: 3, P(3): 1.951189
Iteration: 4, P(4): 1.991457
Iteration: 5, P(5): 2.000396
Iteration: 6, P(6): 1.999997
Iteration: 7, P(7): 2.000000
Iteration: 8, P(8): 2.000000
Solution is 2.000000
Number of Iterations Done: 8
>>
```

$$Q4. F(x) = x \sin(x) - 1/2 \cos(2x), \quad [0,2]$$

Root: 0.517085

Using Newton's Method:

The screenshot shows the MATLAB interface with the following details:

- Editor Tab:** The file `Newtons_Method.m` is open, containing the MATLAB script for the Newton's Method algorithm.
- Command Window:** The output shows the execution of the script, including the function definition, derivative calculation, iteration steps, and final solution.
- Workspace:** The workspace variables are listed, including `f`, `df`, `i`, `N`, `NO`, `p`, `p0`, and `tolerance`.

```

Newtons_Method.m % x
1 % Modified Newtons Algorithm
2 % Including the library for making and computing f(x)
3 syms 'x'
4 i = 1;
5 % number of iterations to perform
6 NO = 1000;
7 % tolerance for stopping
8 tolerance = 0.00001;
9 % initial value (0+2 / 2)
10 p0 = 1;
11 f(x) = (x * sin(x)) - ((1/2)*cos(2*x));
12 df = diff(f);
13 % The Function To Compute: x * sin(x);
14 % Derivative of The Function: sin(2*x) + sin(x) + x*cos(x)
15 fprintf('Iteration: %d, P(%d): %f\n', i, NO);
16 while i <= NO
17     % Newton Method formula
18     p = p0 - ((p0)/df(p0));
19     fprintf('Iteration: %d, P(%d): %f\n', i, NO);
20     % if stopping criteria met then exit loop and display output
21     if (abs(p - double(p0)) < tolerance)
22         fprintf('Solution is %f\n', double(p));
23         fprintf('Number of Iterations Done: %d', i);
24         return
25     end
26     % increment counter and update value of p0
27     i = i + 1;
28     p0 = p;
29 end
30 % if the tolerance was not achieved then display error message
31 fprintf('Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n', tolerance, NO);

```

Command Window Output:

```

>> Newtons_Method
The Function To Compute: x*sin(x) - cos(2*x)/2
Derivative of The Function: sin(2*x) + sin(x) + x*cos(x)
Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 0.541898
Iteration: 2, P(2): 0.517484
Iteration: 3, P(3): 0.517085
Iteration: 4, P(4): 0.517085
Solution is 0.517085
Number of Iterations Done: 4
>>

```

The screenshot shows the MATLAB interface with the following details:

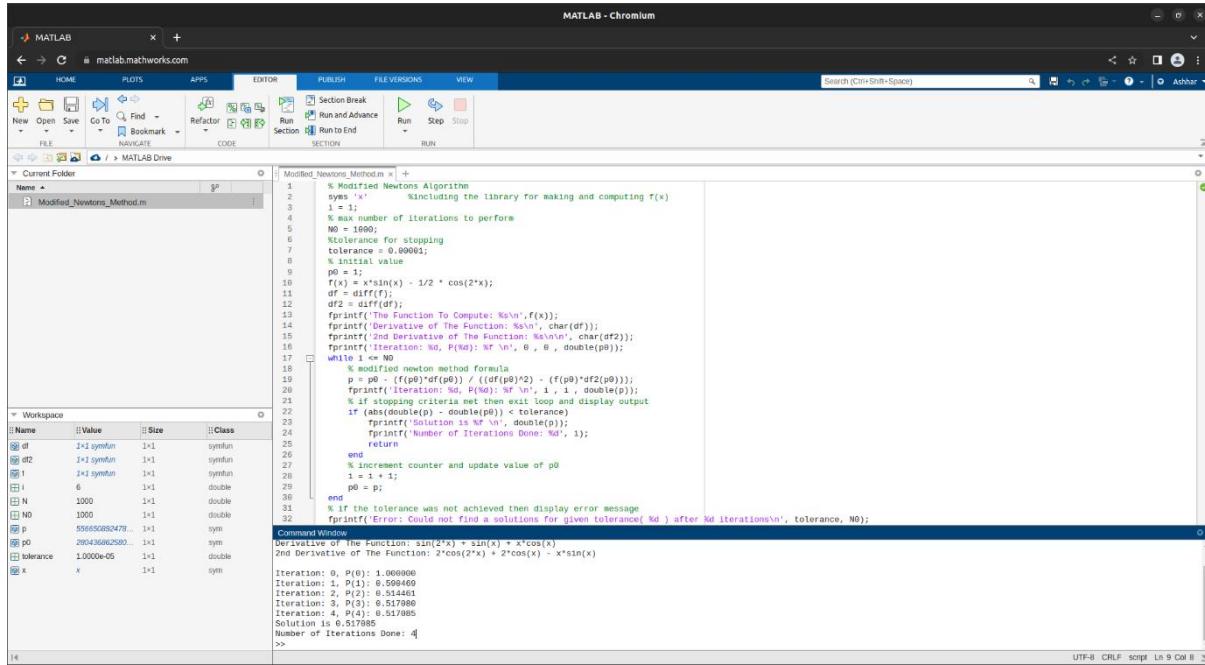
- Editor Tab:** The file `Newtons_Method.m` is open, containing the MATLAB script for the Newton's Method algorithm.
- Command Window:** The output shows the execution of the script, including the function definition, derivative calculation, iteration steps, and final solution.
- Workspace:** The workspace variables are listed, including `f`, `df`, `i`, `N`, `NO`, `n`, and `p`.

```

>> Newtons_Method
The Function To Compute: x*sin(x) - cos(2*x)/2
Derivative of The Function: sin(2*x) + sin(x) + x*cos(x)
Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 0.541898
Iteration: 2, P(2): 0.517484
Iteration: 3, P(3): 0.517085
Iteration: 4, P(4): 0.517085
Solution is 0.517085
Number of Iterations Done: 4
>>

```

Using Modified Newton's Method:



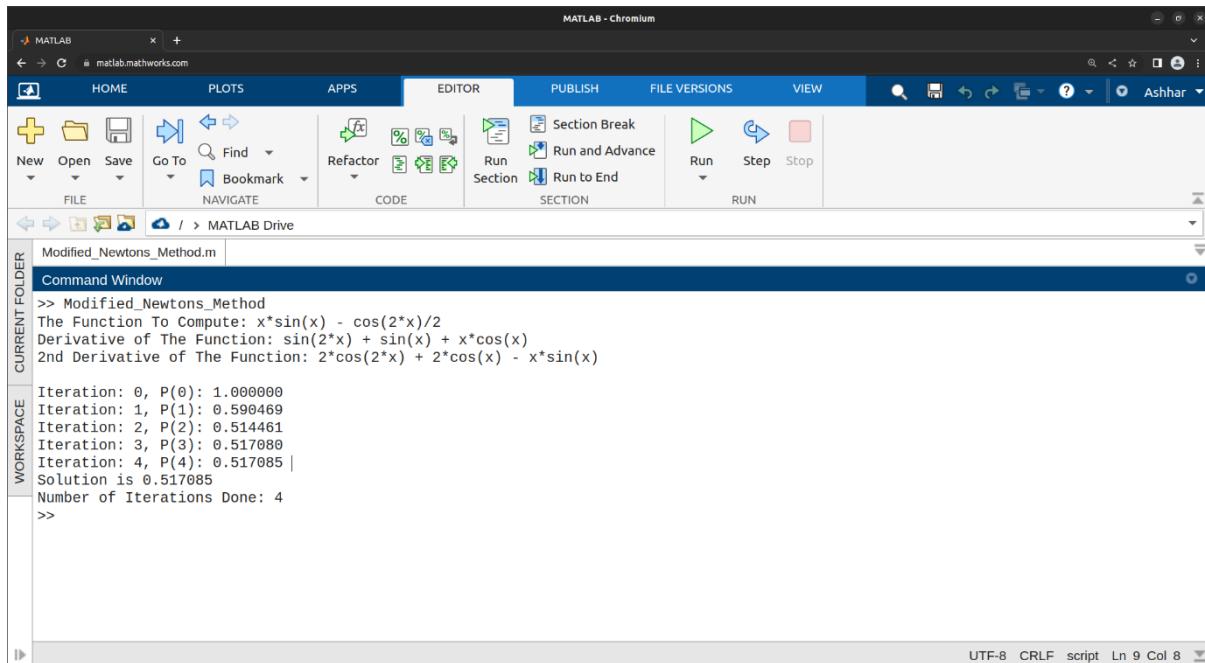
The screenshot shows the MATLAB Editor window with the file `Modified_Newtons_Method.m` open. The code implements the modified Newton's method to solve the equation $x \sin(x) - \cos(2x) = 0$. It includes comments explaining the variables and the iterative process. The workspace pane shows variables like `f`, `df`, `p`, and `N`. The command window at the bottom displays the iteration steps and the final solution.

```

MATLAB - Chromium
MATLAB | matlab.mathworks.com | Search (Ctrl+Shift+Space)
HOME PLOTS APPS EDITOR PUBLISH FILE VERSIONS VIEW
FILE New Open Save Go To Find Refactor Run and Advance Run Section Run to End Step Stop SECTION RUN
Current Folder / > MATLAB Drive
Modified_Newtons_Method.m
1 % Modified Newtons Algorithm
2 % Including the library for making and computing f(x)
3 % Number of iterations to perform
4 N0 = 1000;
5 %tolerance for stopping
6 tolerance = 0.00001;
7 % initial value
8 p0 = 0;
9 f(x) = x*sin(x) - 1/2 * cos(2*x);
10 df = diff(f);
11 df2 = diff(df);
12 fprintf(' Function To Compute: %s\n',f(x));
13 fprintf('Derivative of The Function: %s\n',char(df));
14 fprintf('2nd Derivative of The Function: %s\n',char(df2));
15 fprintf('Iteration %d, P(%d): %f\n', 0 , 0 , double(p0));
16 while abs(double(p) - double(p0)) > tolerance
17 % modified newton method formula
18 p = p0 - ((p0)*df(p0)) / ((df(p0))^2) - (f(p0)*df2(p0));
19 fprintf('Iteration: %d, P(%d): %f\n', 1 , double(p));
20 if (abs(double(p) - double(p0)) < tolerance)
21 fprintf('Solution is %f\n', double(p));
22 else
23 fprintf('Number of Iterations Done: %d', 1);
24 end
25 return
26 % increment counter and update value of p0
27 i = i + 1;
28 p0 = p;
29 end
30 % If the tolerance was not achieved then display error message
31 if (abs(double(p) - double(p0)) < tolerance)
32 fprintf('Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n', tolerance, N0);
33 end
34
Command Window
Derivative of The Function: sin(2*x) + sin(x) + x*cos(x)
2nd Derivative of The Function: 2*cos(2*x) + 2*cos(x) - x*sin(x)

Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 0.590469
Iteration: 2, P(2): 0.514461
Iteration: 3, P(3): 0.517080
Iteration: 4, P(4): 0.517085 |
Solution is 0.517085
Number of Iterations Done: 4
>>

```



The screenshot shows the MATLAB Command Window with the script `Modified_Newtons_Method.m` running. The output displays the iterations of the modified Newton's method, starting from $P(0)$ and converging to a solution of approximately 0.517085 after 4 iterations.

```

MATLAB - Chromium
MATLAB | matlab.mathworks.com | Search (Ctrl+Shift+Space)
HOME PLOTS APPS EDITOR PUBLISH FILE VERSIONS VIEW
FILE New Open Save Go To Find Refactor Run and Advance Run Section Run to End Step Stop SECTION RUN
CURRENT FOLDER / > MATLAB Drive
Modified_Newtons_Method.m
Command Window
>> Modified_Newtons_Method
The Function To Compute: x*sin(x) - cos(2*x)/2
Derivative of The Function: sin(2*x) + sin(x) + x*cos(x)
2nd Derivative of The Function: 2*cos(2*x) + 2*cos(x) - x*sin(x)

Iteration: 0, P(0): 1.000000
Iteration: 1, P(1): 0.590469
Iteration: 2, P(2): 0.514461
Iteration: 3, P(3): 0.517080
Iteration: 4, P(4): 0.517085 |
Solution is 0.517085
Number of Iterations Done: 4
>>

```

Using Secant Method:

MATLAB - Chromium

```

% Secant Algorithm
syms 'x' %Including the library for making and computing f(x)
i = 2;
% Number of iterations to perform
NO = 1000;
% Tolerance for stopping
tolerance = 0.00001;
% initial values
p0 = 0.5; % First initial guess
p1 = 1; % second initial guess
f(x) = x*sin(x) - cos(2*x)/2;
fprintf('The Function To Compute: %s\n',f(x));
for(Iteration=0; Iteration<NO; Iteration++)
    fprintf('Iteration: %d, P(%d): %f\n', Iteration, NO, double(p0));
    fprintf('Iteration: %d, P(%d): %f\n', Iteration, NO, double(p1));
    q0 = f(p0);
    q1 = f(p1);
    p2 = p1 - (q1 * (p1 - p0)) / (q1 - q0);
    if (abs(double(p2)-double(p1)) < tolerance)
        fprintf('Solution is %f\n', double(p2));
        fprintf('Number of Iterations Done: %d', Iteration);
        return;
    end
    % increment counter and update value of p0
    i = i + 1;
    p0 = p1;
    p1 = p2;
    q0 = q1;
    q1 = f(p1);
end
if (tolerance was not achieved then display error message
fprintf('Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n', tolerance, NO);

```

Workspace

Name	Value	Size	Class
df	1x1 symfun	1x1	symfun
df2	1x1 symfun	1x1	symfun
i	1x1 symfun	1x1	symfun
NO	8	1x1	double
n	1000	1x1	double
p0	1465056545...	1x1	sym
p1	3839379797...	1x1	sym
q0	8093023732...	1x1	sym
q1	-152416086...	1x1	sym
x	1092373007...	1x1	sym
q2	NAV	1x1	sym
tolerance	1.0000e-05	1x1	double

Command Window

```

>> Secant_Method
The Function To Compute: x*sin(x) - cos(2*x)/2
Iteration: 0, P(0): 0.500000
Iteration: 1, P(1): 1.000000
Iteration: 2, P(2): 0.514092
Iteration: 3, P(3): 0.516573
Iteration: 4, P(4): 0.517086
Iteration: 5, P(5): 0.517085
Iteration: 6, P(6): 0.517085
Solution is 0.517085
Number of Iterations Done: 6
>>

```

MATLAB - Chromium

```

% Secant Method
The Function To Compute: x*sin(x) - cos(2*x)/2
Iteration: 0, P(0): 0.500000
Iteration: 1, P(1): 1.000000
Iteration: 2, P(2): 0.514092
Iteration: 3, P(3): 0.516573
Iteration: 4, P(4): 0.517086
Iteration: 5, P(5): 0.517085
Iteration: 6, P(6): 0.517085
Solution is 0.517085
Number of Iterations Done: 6
>>

```

Workspace

Name	Value	Size	Class
df	1x1 symfun	1x1	symfun
df2	1x1 symfun	1x1	symfun
f	1x1 symfun	1x1	symfun
i	6	1x1	double
NO	1000	1x1	double
n	(cos(2)/4 -	1x1	sym

Command Window

```

>> Secant_Method
The Function To Compute: x*sin(x) - cos(2*x)/2
Iteration: 0, P(0): 0.500000
Iteration: 1, P(1): 1.000000
Iteration: 2, P(2): 0.514092
Iteration: 3, P(3): 0.516573
Iteration: 4, P(4): 0.517086
Iteration: 5, P(5): 0.517085
Iteration: 6, P(6): 0.517085
Solution is 0.517085
Number of Iterations Done: 6
>>

```

$$Q5. F(x) = \ln(x-1) + \cos(x-1), [0,2]$$

Root: 1.397748

Using Newton's Method:

```

MATLAB - Chromium
MATLAB HOME APPS EDITOR PUBLISH FILE VERSIONS VIEW
New Open Save Go To Find Refactor Run Step Stop RUN
FILE NAVIGATE CODE SECTION
Search (Ctrl+Shift+Space) Ashhar
Current Folder Name ▾
Newton_Method.m
1 % Modified Newtons Algorithm
2 % Including the library for making and computing f(x)
3 i = 1;
4 % max number of iterations to perform
5 N = 1000;
6 %tolerance for stopping
7 tolerance = 0.00001;
8 % initial value
9 p0 = 1;
10 f = log(p0-1) + cos(p0);
11 df = diff(f);
12 fprintf('The Function To Compute: %s\n',f(x));
13 fprintf('Derivative of The Function: %s\n',char(df));
14 fprintf('Iteration: %d, P(%d): %f\n', 0 , p0, double(p0));
15 while i <= N
16     % newton method formula
17     p = p0 - f/(df);
18     fprintf('Iteration: %d, P(%d): %f\n', i , p, double(p));
19     % if stopping criteria met then exit loop and display output
20     if (abs(double(p)) - double(p0)) < tolerance
21         fprintf('Solution is %f\n', double(p));
22         fprintf('Number of Iterations Done: %d', i);
23         return
24     end
25     % increment counter and update value of p0
26     i = i + 1;
27     p0 = p;
28 end
29 % if the tolerance was not achieved then display error message
30 fprintf('Error: Could not find a solutions for given tolerance(%d ) after %d iterations\n', tolerance, N);

```

Command Window

```

>> Newtons_Method
The Function To Compute: cos(x - 1) + log(x - 1)
Derivative of The Function: 1/(x - 1) - sin(x - 1)
Iteration: 0, P(0): 1.569906
Iteration: 1, P(1): 1.378707
Iteration: 2, P(2): 1.397136
Iteration: 3, P(3): 1.397748
Iteration: 4, P(4): 1.397748
Solution is 1.397748
Number of Iterations Done: 4
>>

```

```

MATLAB - Chromium
MATLAB HOME APPS EDITOR PUBLISH FILE VERSIONS VIEW
New Open Save Go To Find Refactor Run Step Stop RUN
FILE NAVIGATE CODE SECTION
Search (Ctrl+Shift+Space) Ashhar
Current Folder Name ▾
Newton_Method.m
Command Window
>> Newtons_Method
The Function To Compute: cos(x - 1) + log(x - 1)
Derivative of The Function: 1/(x - 1) - sin(x - 1)
Iteration: 0, P(0): 1.560000
Iteration: 1, P(1): 1.378707
Iteration: 2, P(2): 1.397136
Iteration: 3, P(3): 1.397748
Iteration: 4, P(4): 1.397748
Solution is 1.397748
Number of Iterations Done: 4
>>

```

Using Modified Newton's Method:

The screenshot shows the MATLAB IDE interface with the following details:

- Editor Tab:** The script file `Modified_Newtons_Method.m` is open, containing the MATLAB code for the modified Newton's method.
- Command Window:**

```

Command Window
Iteration: 0, P(0): 1.500000
Iteration: 1, P(1): 1.412681
Iteration: 2, P(2): 1.398115
Iteration: 3, P(3): 1.397749
Iteration: 4, P(4): 1.397748
Solution is 1.397748
Number of Iterations Done: 4
>>

```
- Status Bar:** Shows "UTF-8 CRLF script Ln 12 Col 16".

The screenshot shows the MATLAB IDE interface with the following details:

- Command Window:**

```

Command Window
>> Modified_Newtons_Method
The Function To Compute: cos(x - 1) + log(x - 1)
Derivative of The Function: 1/(x - 1) - sin(x - 1)
2nd Derivative of The Function: - cos(x - 1) - 1/(x - 1)^2

Iteration: 0, P(0): 1.500000 |
Iteration: 1, P(1): 1.412681
Iteration: 2, P(2): 1.398115
Iteration: 3, P(3): 1.397749
Iteration: 4, P(4): 1.397748
Solution is 1.397748
Number of Iterations Done: 4
>>

```
- Status Bar:** Shows "UTF-8 CRLF script Ln 13 Col 35".

Using Secant Method:

The screenshot shows the MATLAB IDE interface with the following details:

- Editor Tab:** The current file is `Secant_Method.m`. The code implements the Secant Method to solve the equation $f(x) = \cos(x - 1) + \log(x - 1)$. It starts with initial guesses $p_0 = 1.5$ and $p_1 = 1.55$, and iterates until the tolerance is reached or the maximum number of iterations (1000) is exceeded.
- Command Window:**

```
>> Secant_Method
The Function To Compute: cos(x - 1) + log(x - 1)
Iteration: 0, P(0): 1.250000
Iteration: 1, P(1): 1.500000
Iteration: 2, P(2): 1.423384
Iteration: 3, P(3): 1.393117
Iteration: 4, P(4): 1.397952
Iteration: 5, P(5): 1.397750
Iteration: 6, P(6): 1.397748
Iteration: 7, P(7): 1.397748
Solution is 1.397748
Number of Iterations Done: 7
>>
```
- Workspace:** Shows variables `df`, `df2`, `f`, `i`, `NO`, `P`, `P0`, `P1`, `q0`, `q1`, `q2`, and `x`.

The screenshot shows the MATLAB IDE interface with the following details:

- Editor Tab:** The current file is `Secant_Method.m`. The code implements the Secant Method to solve the equation $f(x) = \cos(x - 1) + \log(x - 1)$. It starts with initial guesses $p_0 = 1.5$ and $p_1 = 1.55$, and iterates until the tolerance is reached or the maximum number of iterations (1000) is exceeded.
- Command Window:**

```
>> Secant_Method
The Function To Compute: cos(x - 1) + log(x - 1)
Iteration: 0, P(0): 1.397752
Iteration: 1, P(1): 1.397750
Iteration: 2, P(2): 1.397748
Iteration: 3, P(3): 1.397748
Iteration: 4, P(4): 1.397748
Iteration: 5, P(5): 1.397748
Iteration: 6, P(6): 1.397748
Iteration: 7, P(7): 1.397748
Solution is 1.397748
Number of Iterations Done: 7
>>
```
- Workspace:** Shows variables `df`, `df2`, `f`, `i`, `NO`, `n`, and `x`.