IS5306-NUMERICAL METHODS

Project Work: Group 03

Group Members:

EG/2021/4410	ARACHCHI W.A.H.U.W.
EG/2021/4412	ARACHCHI W.A.T.T.W.
EG/2021/4414	ARIVANAN V.
EG/2021/4415	ARIVARASAN J.
EG/2021/4417	ASHFAQ M.R.M.
EG/2021/4418	ATHAPATHTHU A.A.V.S.
EG/2021/4419	ATHTHANAYAKA K.A.L.G.
EG/2021/4424	BALASOORIYA J.M.
EG/2021/4426	BANDARA A.W.M.L.M.
EG/2021/4432	BANDARA K.M.T.O.N.
EG/2021/4433	BANDARA L.R.T.D.
EG/2021/4434	BANDARA M.M.D.L.
EG/2021/4437	BANDARA R.M.U.T.
EG/2021/4447	BRAVIN K.
EG/2021/4449	BULEGODAARACHCHI D.C.

Table of Contents

1 The Selected non linear equation and its relevance

$$f(x) = x^3 - x - 2 = 0$$

This equation has been selected due to its significance in modeling real-world systems where nonlinear relationships between variables need to be analyzed. It serves as a representative mathematical model that can be applied in a variety of scientific and engineering applications, including those in the computer engineering field.

To solve this equation, we aim to determine its root within the interval [1,2] [1,2], where the equation is expressed as:

$f(x)=x^3-x-2$

The selection of this nonlinear equation is motivated by its practical applications in computer engineering systems where nonlinear relationships often emerge. These relationships are commonly encountered in areas such as algorithm optimization, signal processing, and hardware design.

Example in Real-Life Scenario:

Consider the behavior of a computer processor or a circuit board under varying operational conditions:

- Thermal Management in Processors: As processors work, they experience varying thermal loads, which affect the overall performance. Nonlinear equations like

$f(x)=x^3-x-2$ can be used to model temperature changes across components as power consumption varies.

- Power Efficiency in Electronics: In systems with power regulation circuits, the relationship between voltage, current, and resistance can often be nonlinear, necessitating the use of nonlinear equations for accurate modeling and analysis.

These examples illustrate the importance of solving equations such as

$f(x)=x^3-x-2$

in predicting and optimizing system performance in computer hardware design and thermal management.

Nonlinear Equation Appears in Such Systems;

- Processor Performance and Heat Dissipation: Nonlinear equations are used to model how heat generated by processors influences the overall system performance. As clock speeds increase or system workloads change, thermal effects become nonlinear, requiring numerical methods to predict temperatures and adjust cooling mechanisms.
- Optimization Algorithms: Nonlinear equations appear in optimization problems, particularly in the context of algorithmic complexity and resource allocation. In scenarios such as load balancing or memory management, the relationship between system parameters is often nonlinear, making the need for accurate solutions critical.
- Signal Processing: Nonlinear systems are prevalent in signal processing, where algorithms need to handle varying frequencies or power levels. The relationship between input signals and outputs in certain filters or transformations is often nonlinear, and solving such equations is necessary to ensure precision in the processing systems.

This equation is highly relevant in computer engineering and computational applications, including:

- Circuit Design: Nonlinear equations are crucial when modeling the performance of integrated circuits under varying conditions like power fluctuations or heat exposure.
- Algorithm Efficiency: Cubic equations can be used to describe the computational complexity of certain algorithms, especially in scenarios where resource consumption grows nonlinearly with problem size, such as in cryptography or machine learning.
- Simulations and Modeling: In computational simulations, particularly those modeling processor behavior or electronic systems, solving nonlinear equations accurately is essential to predict performance, optimize resource usage, and ensure system stability.

2 Manual Solutions

2.1 Bisection Method

$f(x) = x^3 - x - 2 = 0$

$f(x) = x^3 - x - 2 = 0$, interval [1,2]

$f(1) = 1^3 - 1 - 2$

$= \underline{\underline{-2}}$ **$F(1) < 0$**

$f(2) = 2^3 - 2 - 2$

$= \underline{\underline{4}}$ **$F(2) > 0$**

Since f(1)<0. f(2) > 0; a root exists between 1 and 2

$$\mathbf{C} = \left(\frac{1+2}{2}\right)$$

$$= \underline{\underline{1.5}}$$

$f(1.5) = 1.5^3 - 1.5 - 2$

$= \underline{\underline{-0.125}}$

- Therefore, updated Interval [1.5,2]

Table 1:Iteration Table.

Iterations	**a**	**F(a)**	**b**	**F(b)**	**c**	**F(c)**	**update**
1	1.00000000	-2.00000000	2.00000000	4.00000000	1.50000000	-0.12500000	a=c
2	1.50000000	-0.12500000	2.00000000	4.00000000	1.75000000	1.60937500	b=c
3	1.50000000	-0.12500000	1.75000000	1.60937500	1.62500000	0.66601562	b=c
4	1.50000000	-0.12500000	1.62500000	0.66601562	1.56250000	0.25219727	b=c
5	1.50000000	-0.12500000	1.56250000	0.25219727	1.53125000	0.05911255	b=c

- So **1.53125000** considered as a **root** according to Bisection Method after 5 iterations

2.2 Newton-Raphson Method

f(x) = $x^3 - x - 2 = 0$

$f'(x) = 3x^2 - 1$

Initial guess :

$x_0 = 1.5$

Newton-Raphson formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f`(x_n)}$$

n = 0

$f(x_0) = (1.5)^3 - 1.5 - 2$

$f(x_0) = -0.125$

$f`(x_0) = 3 \times (1.5)^2 - 1$

$f`(x_0) = 5.75$

$x_1 = x_0 - \frac{f(x_0)}{f`(x_0)}$

$x_1 = 1.5 - \frac{(-0.125)}{5.75}$

$x_1 = 1.5 - 0.02174$

$\underline{\underline{x_1 = 1.52174}}$

$\varepsilon_1 = \left|\frac{x_1 - x_0}{x_1}\right| \times 100\ \%$

$\varepsilon_1 = \left|\frac{1.52174 - 1.5}{1.52174}\right| \times 100\ \%$

$\underline{\underline{\varepsilon_1 = 1.426\ \%}}$

n = 1

$$f(x_0) = (1.52174)^3 - 1.52174 - 2$$

$$f(x_0) = 0.00214$$

$$f\,`(x_0) = 3 \times (1.52174)^2 - 1$$

$$f\,`(x_0) = 5.94708$$

$$x_2 = x_1 - \frac{f(x_0)}{f\,`(x_0)}$$

$$x_2 = 1.52174 - \frac{(0.00214)}{5.94708}$$

$$x_2 = 1.5 - 0.00036$$

$$\underline{\underline{x_2 = 1.52138}}$$

$$\varepsilon_2 = \left|\frac{x_2 - x_1}{x_2}\right| \times 100\ \%$$

$$\varepsilon_2 = \left|\frac{1.52138 - 1.52174}{1.52138}\right| \times 100\ \%$$

$$\underline{\underline{\varepsilon_2 = 0.0236\ \%}}$$

n = 2

$f(x_0) = (1.52138)^3 - 1.52138 - 2$

$f(x_0) = 0.0000017$

$f`(x_0) = 3 \times (1.52138)^2 - 1$

$f`(x_0) = 5.94379$

$x_3 = x_2 - \frac{f(x_0)}{f`(x_0)}$

$x_3 = 1.52138 - \frac{(0.0000017)}{5.94379}$

$x_3 = 1.52138 - 0.00000$

$\underline{\underline{x_3 = 1.52138}}$

$\varepsilon_3 = \left|\frac{x_3 - x_2}{x_3}\right| \times 100\ \%$

$\varepsilon_1 = \left|\frac{1.52138 - 1.52138}{1.52138}\right| \times 100\ \%$

$\underline{\underline{\varepsilon_1 = 0\ \%}}$

Table 2:Iteration Table.

Iteration	x_n	$f(x_n)$	$f`(x_n)$	x_{n+1}	Absolute error (in%)
1	1.5	-0.125	5.7500	1.52174	1.4260
2	1.52174	0.00214	5.94708	1.52138	0.0236
3	1.52138	0.0000017	5.94379	1.52138	0.0000
4	1.52138	0.0000017	5.94379	1.52138	0.0000
5	1.52138	0.0000017	5.94379	1.52138	0.0000

2.3 Fixed Point Method

$\mathrm{f(x)} = \mathrm{x}^3 - \mathrm{x} - 2 = 0$

$\mathrm{g(x)} = \mathrm{x}$

$x = (x+2)^{\frac{1}{3}}$ -> $x_n = (x_n + 2)^{\frac{1}{3}}$

Therefore:

$g(x) = (x+2)^{\frac{1}{3}}$

Let interval as [1,2]

$x = 1$

$g(1)$ = $(1+2)^{\frac{1}{3}}$

= $\underline{\underline{1.442}}$

$x = 2$

$g(2)$ = $(2+2)^{\frac{1}{3}}$

= $\underline{\underline{1.5874}}$

$g(1)$ > 1 $g(2)$ < 2

Hence g(x) ϵ [1,2]

h(x) = g(x) -x ; ɏ x ϵ [1,2]

h(1) = g(1) -1 >0

h(2) = g(2) -2 <0

Derivative of g(x)

g'(x)=$\frac{1}{3}(x+2)^{\frac{-2}{3}}$

For the interval [1,2]:

x =1;	g'(1)	=	$\frac{1}{3}(1+2)^{\frac{-2}{3}}$	=	0.154
x =2;	g'(2)	=	$\frac{1}{3}(2+2)^{\frac{-2}{3}}$	=	0.118

Since |g'(x)|<1| for all x∈[1,2] the fixed-point iteration will converge.

Initial Guess:

$x_{0=}1.5$

n=0	x_1	$(1.5+2)^{\frac{1}{3}}$	=	1.51829
Error	$\|x_1 - x_0\|$	|1.51829-1.5|	=	0.01829
n=1	x_2	$(1.51829+2)^{\frac{1}{3}}$	=	1.52093
Error	$\|x_1 - x_0\|$	|1.52093-1.51829|	=	0.00264
n=2	x_3	$(1.52093+2)^{\frac{1}{3}}$	=	1.5213149
Error	$\|x_1 - x_0\|$	|1.5213149-1.52093|	=	0.00038
n=3	x_4	$(1.52131+2)^{\frac{1}{3}}$	=	1.52137
Error	$\|x_1 - x_0\|$	|1.52137-1.52131|	=	0.00006
n=4	x_5	$(1.52137+2)^{\frac{1}{3}}$	=	1.5213783
Error	$\|x_1 - x_0\|$	|1.52138-1.52137|	=	0.00001

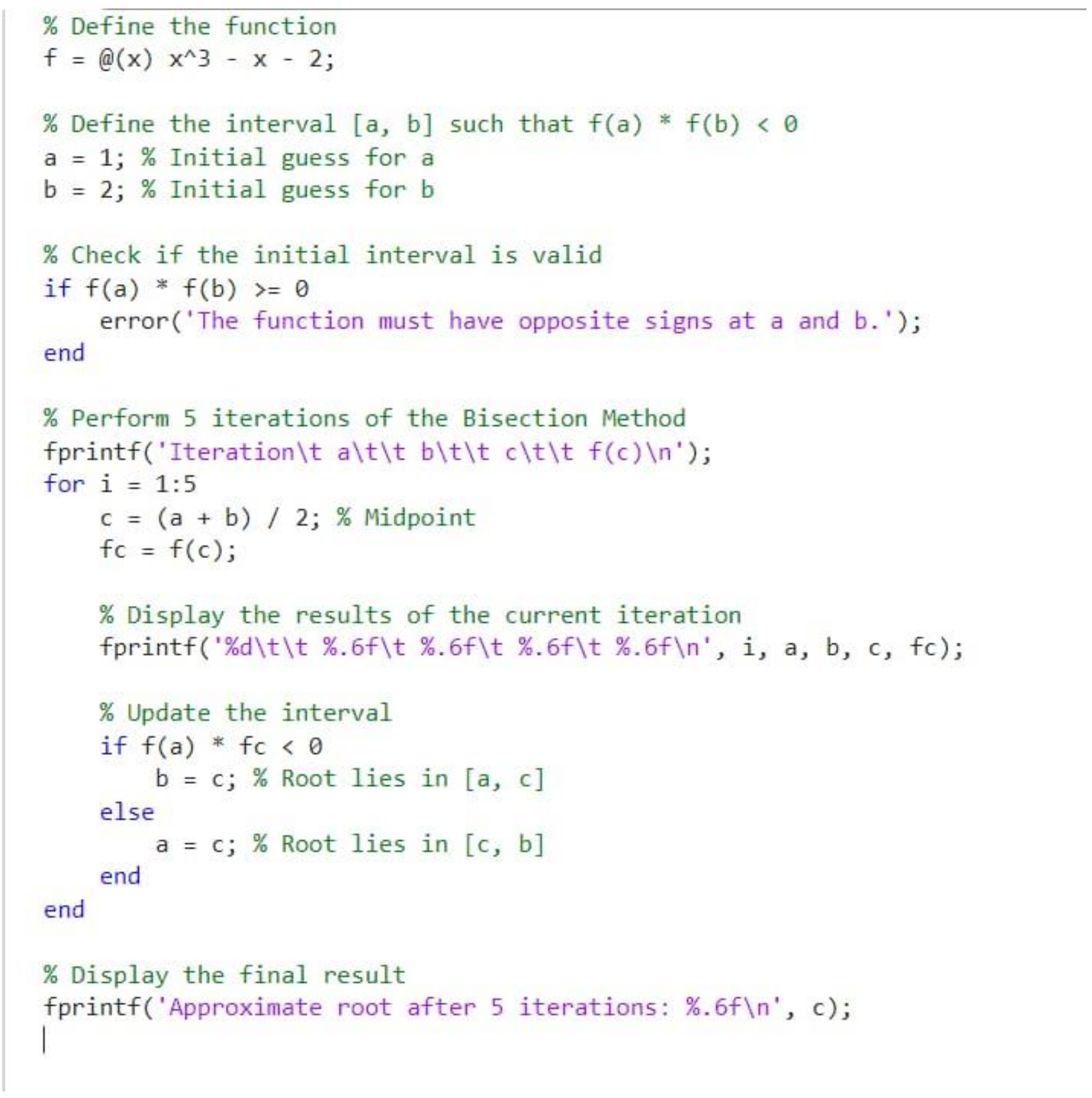
After 5 iterations, the approximate root of $x^3 - x - 2 = 0$ is x=1.545

Table 3:Iteration Table.

Iterations	x_n	x_{n++1}	**Error**
0	1.5	1.51829	0.01829
1	1.51829	1.52093	0.00264
2	1.52093	1.5213149	0.00038
3	1.5213149	1.52137	0.00006
4	1.52137	1.5213783	0.00001

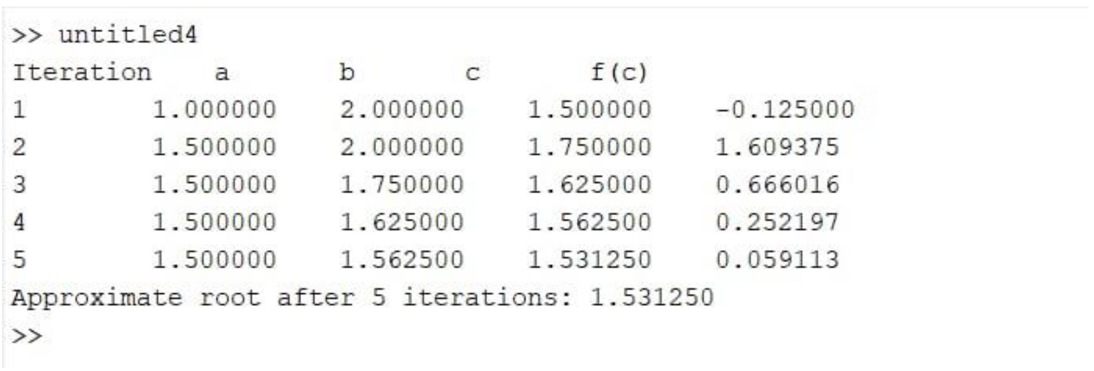
3 MathLab Implementation

3.1 Bisection Method



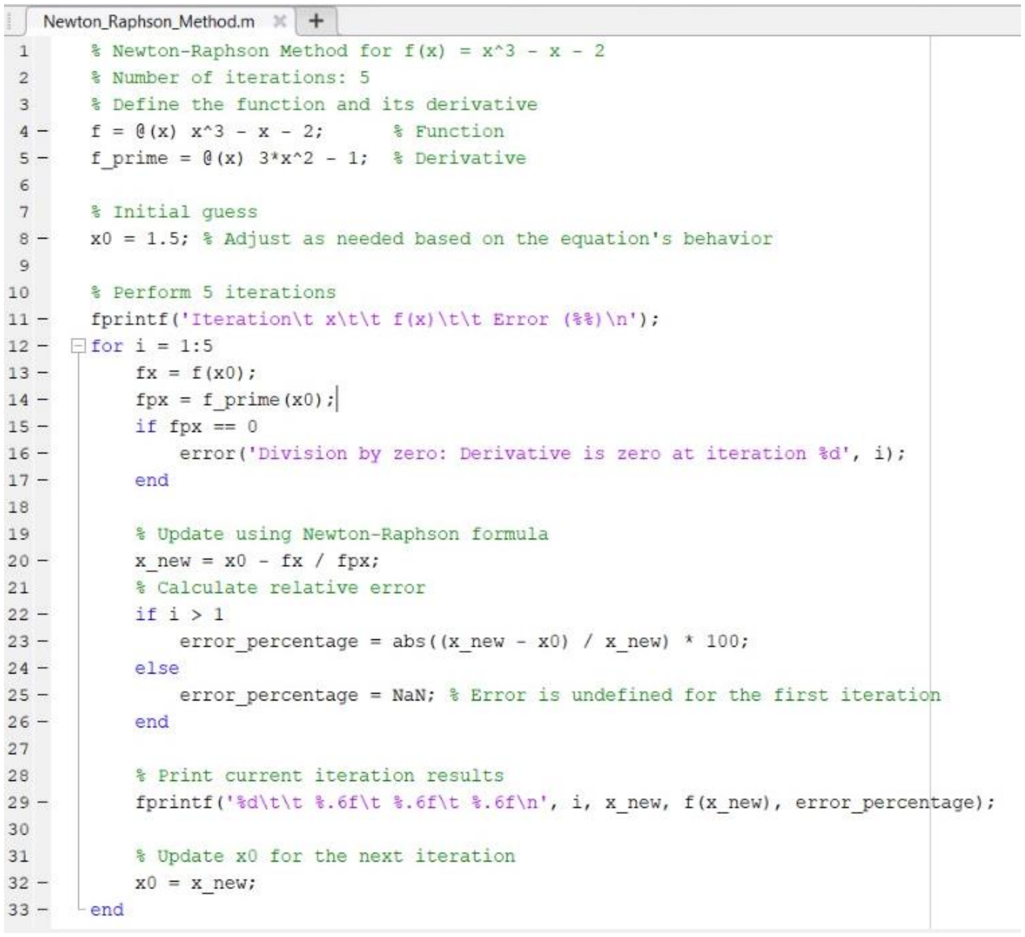
```
% Define the function
f = @(x) x^3 - x - 2;

% Define the interval [a, b] such that f(a) * f(b) < 0
a = 1; % Initial guess for a
b = 2; % Initial guess for b

% Check if the initial interval is valid
if f(a) * f(b) >= 0
    error('The function must have opposite signs at a and b.');
end

% Perform 5 iterations of the Bisection Method
fprintf('Iteration\t a\t\t b\t\t c\t\t f(c)\n');
for i = 1:5
    c = (a + b) / 2; % Midpoint
    fc = f(c);

    % Display the results of the current iteration
    fprintf('%d\t\t %.6f\t %.6f\t %.6f\t %.6f\n', i, a, b, c, fc);

    % Update the interval
    if f(a) * fc < 0
        b = c; % Root lies in [a, c]
    else
        a = c; % Root lies in [c, b]
    end
end

% Display the final result
fprintf('Approximate root after 5 iterations: %.6f\n', c);
```

Output:



```
>> untitled4
Iteration    a         b         c         f(c)
1         1.000000    2.000000    1.500000    -0.125000
2         1.500000    2.000000    1.750000    1.609375
3         1.500000    1.750000    1.625000    0.666016
4         1.500000    1.625000    1.562500    0.252197
5         1.500000    1.562500    1.531250    0.059113
Approximate root after 5 iterations: 1.531250
>>
```

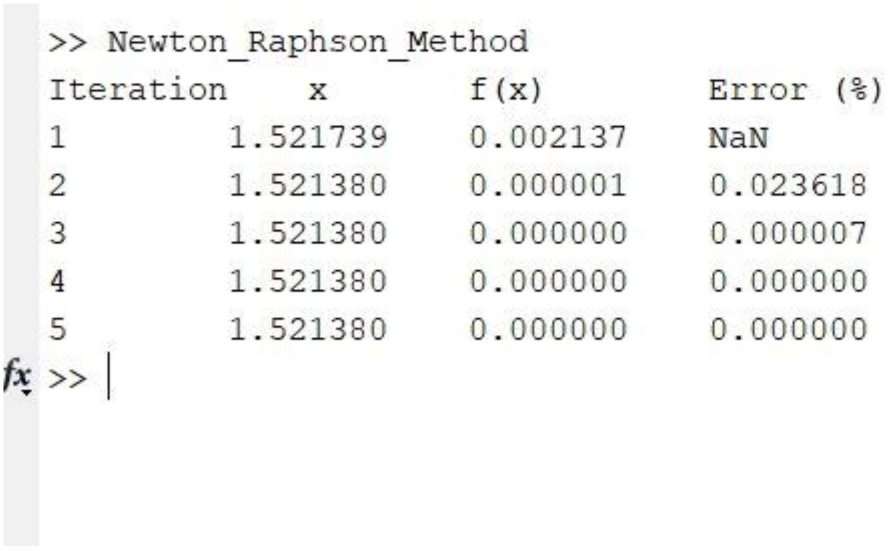
3.2 Newton-Raphson Method



Newton_Raphson_Method.m

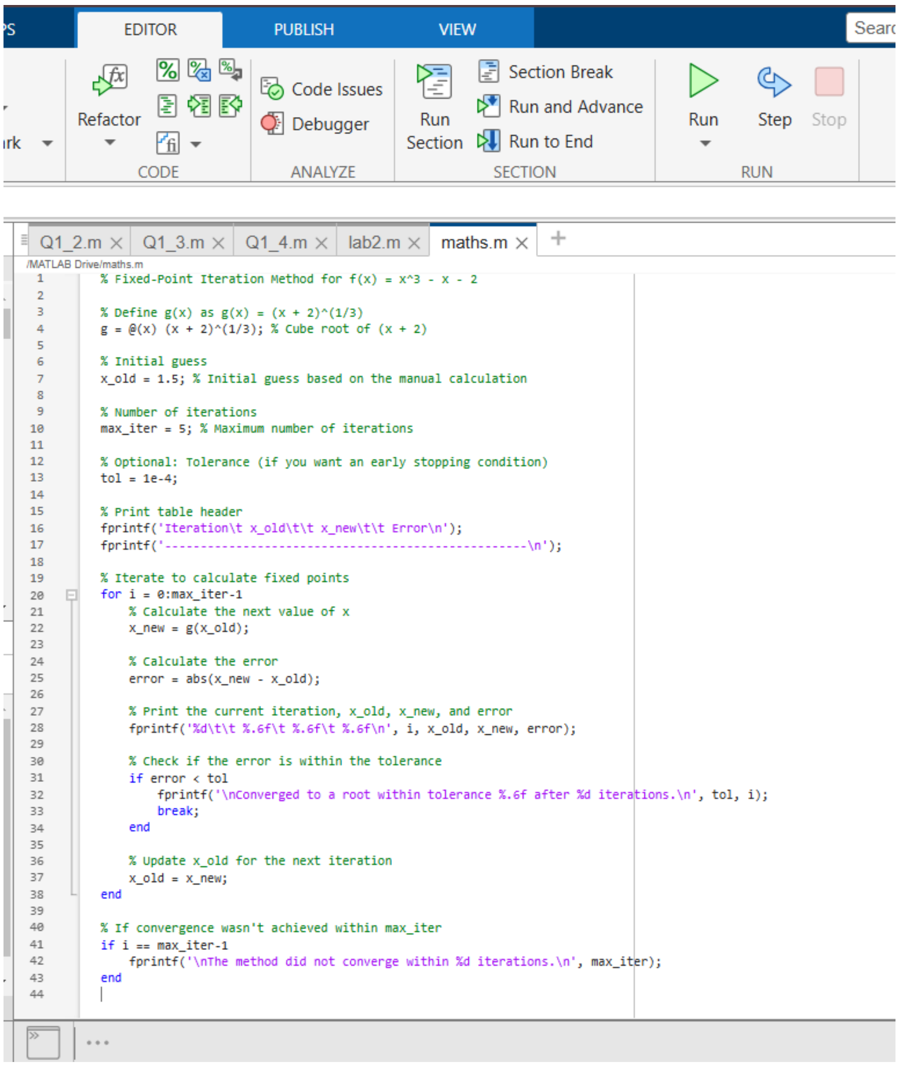
```
% Newton-Raphson Method for f(x) = x^3 - x - 2
% Number of iterations: 5
% Define the function and its derivative
f = @(x) x^3 - x - 2;      % Function
f_prime = @(x) 3*x^2 - 1;  % Derivative

% Initial guess
x0 = 1.5; % Adjust as needed based on the equation's behavior

% Perform 5 iterations
fprintf('Iteration\t x\t\t f(x)\t\t Error (%%)\n');
for i = 1:5
    fx = f(x0);
    fpx = f_prime(x0);
    if fpx == 0
        error('Division by zero: Derivative is zero at iteration %d', i);
    end

    % Update using Newton-Raphson formula
    x_new = x0 - fx / fpx;
    % Calculate relative error
    if i > 1
        error_percentage = abs((x_new - x0) / x_new) * 100;
    else
        error_percentage = NaN; % Error is undefined for the first iteration
    end

    % Print current iteration results
    fprintf('%d\t\t %.6f\t %.6f\t %.6f\n', i, x_new, f(x_new), error_percentage);

    % Update x0 for the next iteration
    x0 = x_new;
end
```

Output



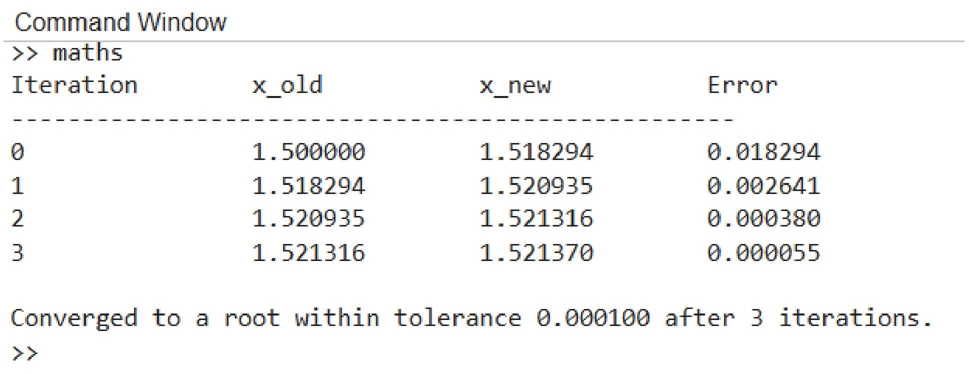
```
>> Newton_Raphson_Method
Iteration     x          f(x)          Error (%)
1          1.521739    0.002137     NaN
2          1.521380    0.000001     0.023618
3          1.521380    0.000000     0.000007
4          1.521380    0.000000     0.000000
5          1.521380    0.000000     0.000000
>> 
```

3.3 Fixed Point Method



```matlab
% Fixed-Point Iteration Method for f(x) = x^3 - x - 2

% Define g(x) as g(x) = (x + 2)^(1/3)
g = @(x) (x + 2)^(1/3); % Cube root of (x + 2)

% Initial guess
x_old = 1.5; % Initial guess based on the manual calculation

% Number of iterations
max_iter = 5; % Maximum number of iterations

% Optional: Tolerance (if you want an early stopping condition)
tol = 1e-4;

% Print table header
fprintf('Iteration\t x_old\t\t x_new\t\t Error\n');
fprintf('--------------------------------------------------\n');

% Iterate to calculate fixed points
for i = 0:max_iter-1
    % Calculate the next value of x
    x_new = g(x_old);

    % Calculate the error
    error = abs(x_new - x_old);

    % Print the current iteration, x_old, x_new, and error
    fprintf('%d\t\t %.6f\t %.6f\t %.6f\n', i, x_old, x_new, error);

    % Check if the error is within the tolerance
    if error < tol
        fprintf('\nConverged to a root within tolerance %.6f after %d iterations.\n', tol, i);
        break;
    end

    % Update x_old for the next iteration
    x_old = x_new;
end

% If convergence wasn't achieved within max_iter
if i == max_iter-1
    fprintf('\nThe method did not converge within %d iterations.\n', max_iter);
end
```

Output:



```
>> maths
Iteration        x_old          x_new          Error
--------------------------------------------------
0                1.500000       1.518294       0.018294
1                1.518294       1.520935       0.002641
2                1.520935       1.521316       0.000380
3                1.521316       1.521370       0.000055

Converged to a root within tolerance 0.000100 after 3 iterations.
>>
```

4 The Comparison and Discussion of the Findings.

Discussion

Accuracy of Roots Found

Using the Bisection Method the root converges to $x \approx 1.531$ with a specified tolerance.This method guarantees a root within the interval [a,b] as long as $f(a) \cdot f(b) < 0$. Its accuracy improves linearly, but achieving high precision requires more iterations. On the other hand, the Newton-Raphson Method achieves high accuracy quickly due to its quadratic convergence, especially when the initial guess is close to the root. So for $x_0 = 1.5$ (initial guess), the method converges to $x \approx 1.521$ within five iterations. Fixed-Point Iteration depends heavily on the choice of g(x); poorly chosen transformations can slow convergence or even prevent it.

Speed of Convergence

The Bisection Method converges slowly compared to the other two, with about 20 iterations needed to reach a precision of 10^{-6}. In contrast, the Newton-Raphson Method converges quadratically and typically requires only 4-5 iterations to achieve the same precision, provided the initial guess is good. Fixed-Point Iteration is slower than Newton-Raphson and can require as many as 15-20 iterations to achieve similar precision, depending on g(x).

Ease of Application and Challenges

The Bisection Method is simple and reliable, requiring no derivative calculations. However, it is slower and requires $f(a) \cdot f(b) < 0$, which may not always be easy to identify. The Newton-Raphson Method is efficient and fast but requires a good initial guess and accurate calculation of $f'(x)$. If the guess is far from the root or $f'(x)=0$, the method may fail. Fixed-Point Iteration is straightforward to implement once g(x) is defined, but selecting an appropriate g(x) is challenging. Poor choices can lead to divergence.

Recommendations

- For guaranteed convergence: Use the Bisection Method, which is best for reliable results when little is known about the function.
- For speed and efficiency: Use the Newton-Raphson Method, especially if a good initial guess and derivative are available.
- For simplicity: Use Fixed-Point Iteration, but only with a well-chosen transformation g(x).

Each method serves a unique purpose and is suited to different situations based on the problem's requirements.