LAB EXPERIMENT # 2: Root finding using Newton Rapson method and Secant method

Name & Roll No.: Ashraf Al- Khalique, 1801171	**Grade (20)**
Registration Number: 351	
Lab Section: C-2	

2.1 Objectives

- To determine roots of an equation in a single variable using 'Newton Rapson method' & 'Secant method'.
- To understand the MATLAB implementation of the 'Newton Rapson method' & 'Secant method'.
- To analyze of results using different initial values and different ranges of error.

2.2 Theory

2.2.1 Newton Rapson method

The Newton-Raphson method is a method for finding the roots of nonlinear equations that is one of the most commonly used root-finding algorithms due to its simplicity and speed. A function's root is the point at which f(x)=0. Many equations have several roots. Every odd degree real polynomial has an odd number of real roots. Newton-Raphson is an iterative approach that starts with an estimate at the root. The method employs both the derivative of the function f′(x) and the original function f(x), and so works only when the derivative is known.

2.2.2 Secant method

The Secant Approach is a numerical method for solving equations with a single unknown. One disadvantage of Newton's approach is that it requires evaluating f'(x) at multiple places, which may be impractical for some f values (x). The secant approach avoids this problem by approximating the derivative with a finite difference. As a result, rather of a tangent line through one point on the graph of f(x), a secant line across two points on the graph of f(x) is used to approximate the root of f(x).

2.3 Apparatus

2.3.1 MATLAB

2.4 <u>Newton Rapson method</u>

2.4.1 Algorithm

<u>Step: 1</u> Define function *f(x)*

Step: 2 Lower ‘*xl*’ and upper ‘*xu*’ values are guessed for the root in such a way that ‘$f(xl)*f(xu) < 0$’

Step: 3 The approximation of the root is determined by ‘$x=xl-(f(xl)/fd(xl))/2$’

Step: 4 Then, take $xl=x$.

Step: 5 If $abs((x-xl)/x)*100 > \varepsilon \,||\, abs((x-xl)/x)*100 > \varepsilon$ accuracy, then go to step 3 otherwise step 6.

Step: 6 Display *x2* as root.

Step: 7 Stop.

2.4.2 Tolerable Error

Defining the approximation error as

% Error = (| Xnew - Xold | / Xnew) * 100

An acceptance level of error says, (E) is input by the user. Then the program is terminated when,

% Error <= E

2.4.3 Pseudocode

```
1. Start
2. Define function f= @x
3. Input
   xl=input ('Lower value xl= ');
   xu=input ('Upper value xu= ');
   Emax=input ('Error= ');
   Imax=input ('Imax= ');
4. if (f(xl)*f(xu)) >0
   print "Incorrect initial guesses"
   goto 3
   End If
5. Do
   x=xl-(f(xl)/fd(xl))/2
   xl=x
   while abs((x-xl)/x)*100 > e
6. Print root as x
7. Stop
```

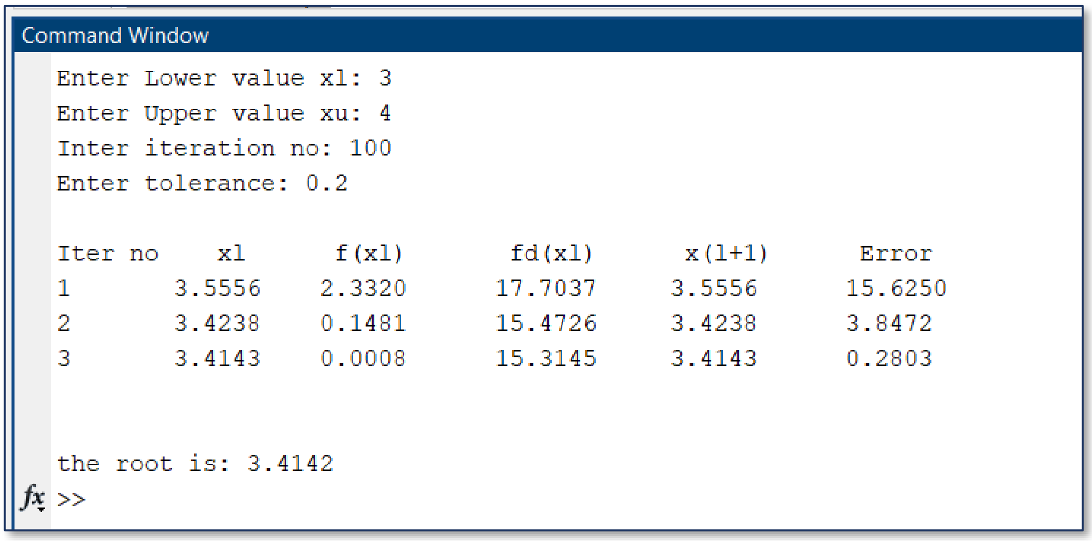
2.4.4 MATLAB Code

```
clc;
clear all;
close all;
f=@(x) (x^3)-(2*x^2)-(6*x)+4;
fd=@(x) (3*x^2)-(4*x)-6;
xl=input('Enter Lower value xl: ');
```

```
xu=input('Enter Upper value xu: ');
Imax=input('Inter iteration no: ');
Emax=input('Enter tolerance: ');
fprintf('\nIter no\t   xl\t   f(xl)\t   fd(xl)\t   x(l+1)\t
Error\n')
if f(xl)*f(xu)<0
    for i=1:Imax
        x=xl-(f(xl)/fd(xl));
        e=abs((x-xl)/x)*100;
        xl=x;
        if e<=Emax;
            break
        end
        fprintf('%d\t    %.4f\t  %.4f\t  %.4f\t  %.4f\t
%.4f\n',i,xl,f(xl),fd(xl),x,e)
    end
end
fprintf('\n\nthe root is: %.4f\n',x);
```

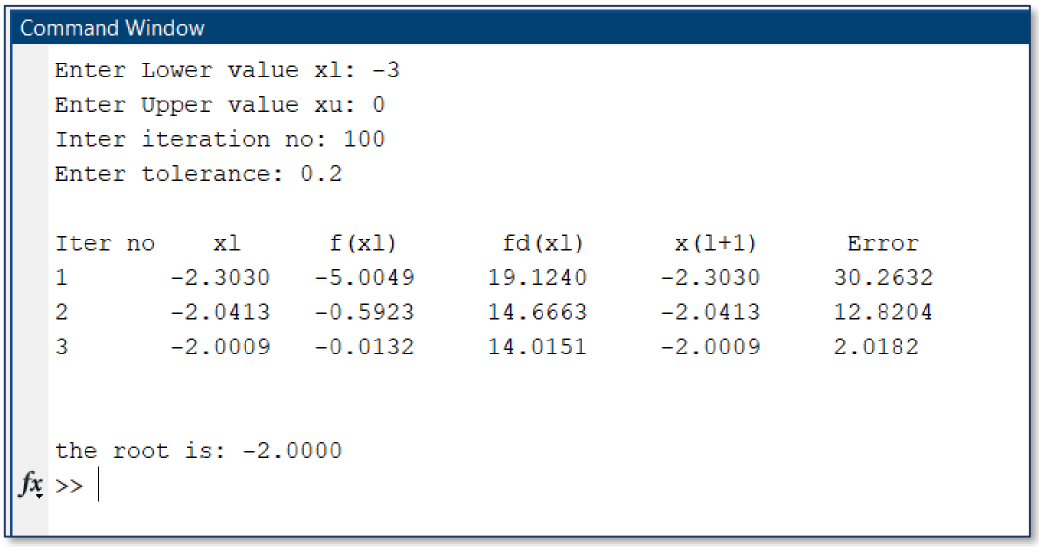
2.4.5 MATLAB Output

- Iteration for error <= 0.2% with interval [3,4]



```
Enter Lower value xl: 3
Enter Upper value xu: 4
Inter iteration no: 100
Enter tolerance: 0.2

Iter no      xl        f(xl)        fd(xl)        x(l+1)        Error
1          3.5556     2.3320       17.7037       3.5556        15.6250
2          3.4238     0.1481       15.4726       3.4238        3.8472
3          3.4143     0.0008       15.3145       3.4143        0.2803


the root is: 3.4142
>>
```

- Iteration for error <= 0.2% with interval [-3,0]



```
Enter Lower value xl: -3
Enter Upper value xu: 0
Inter iteration no: 100
Enter tolerance: 0.2

Iter no     xl        f(xl)       fd(xl)      x(l+1)      Error
1        -2.3030   -5.0049     19.1240     -2.3030     30.2632
2        -2.0413   -0.5923     14.6663     -2.0413     12.8204
3        -2.0009   -0.0132     14.0151     -2.0009     2.0182


the root is: -2.0000
>>
```

<u>2.5 Secant method</u>

2.5.1 Algorithm

<u>**Step: 1**</u> Define function *f(x)*

<u>**Step: 2**</u> Lower '*x0*' and upper '*x1*' values are guessed for the root in such a way that '*f(x0)*f(x1)* < 0'

<u>**Step: 3**</u> The approximation of the root is determined by '*x2=(x0*f(x1)-x1*f(x0))/(f(x1)-f(x0))*'

<u>**Step: 4**</u> Then,

- Take *x0=x1*.
- Next take, *x1=x2*.

<u>**Step: 5**</u> If *abs((x2-x1)/x2)*100* > ε accuracy, then go to step 3 otherwise step 6.

<u>**Step: 6**</u> Display *x2* as root.

<u>**Step: 7**</u> Stop.

2.5.2 Tolerable Error

Defining the approximation error as

% Error = (| Xnew - Xold | / Xnew) * 100

An acceptance level of error says, (E) is input by the user. Then the program is terminated when,

% Error <= E

2.5.3 Pseudocode

```
1. Start
2. Define function f= @x
3. Input
```

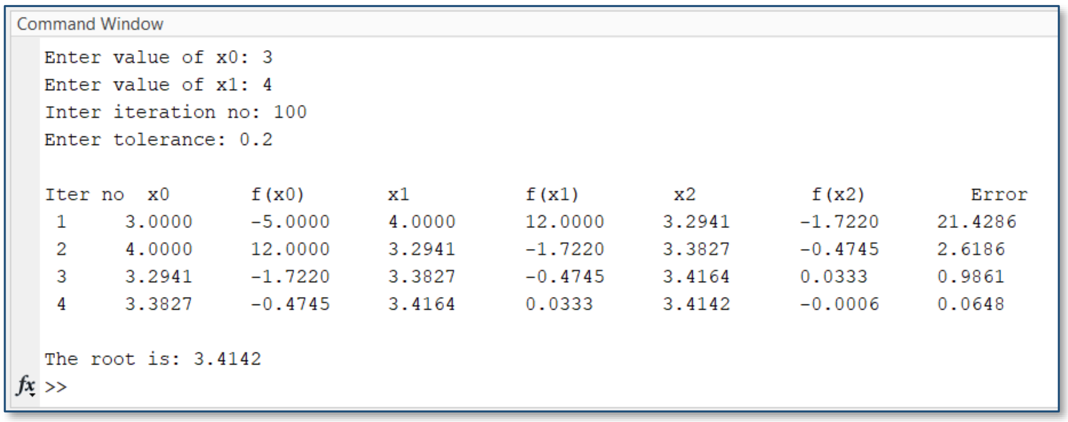
```
      X0=input ('Enter value x0= ');
      X1=input ('Enter value x1= ');
      Emax=input ('Error= ');
      n=input ('Iteration no. ');
   4. if (f(x0)*f(x1)) >0

           print "Incorrect initial guesses"
           goto 3
         end if
   5. Do
      x2=(x0*f(x1)-x1*f(x0))/(f(x1)-f(x0))
      x0 = x1
      x1 = x2
      end if

      while abs((x2-x1)/x2)*100 > e
   8. Print root as x2
   6. Stop
```

2.5.4 MATLAB Code

```
clc;
clear all;
close all;
f=@(x) (x^3)-(2*x^2)-(6*x)+4;
x0=input('Enter value of x0: ');
x1=input('Enter value of x1: ');
n=input('Inter iteration no: ');
Emax=input('Enter tolerance: ');
fprintf('\nIter no\t x0\t      f(x0)\t      x1\t      f(x1)\t
x2\t          f(x2)\t      Error\n')
if f(x0)*f(x1)<0
    for i=1:n
        x2=(x0*f(x1)-x1*f(x0))/(f(x1)-f(x0));
        e1=abs((x2-x1)/x2)*100;
        fprintf(' %d\t   %.4f\t  %.4f\t  %.4f\t  %.4f\t
%.4f\t  %.4f\t  %.4f\n',i,x0,f(x0),x1,f(x1),x2,f(x2),e1)
        if e1<=Emax;
break
end
        x0=x1;
x1=x2;
end
end
fprintf('\nThe root is: %.4f\n',x2);
```

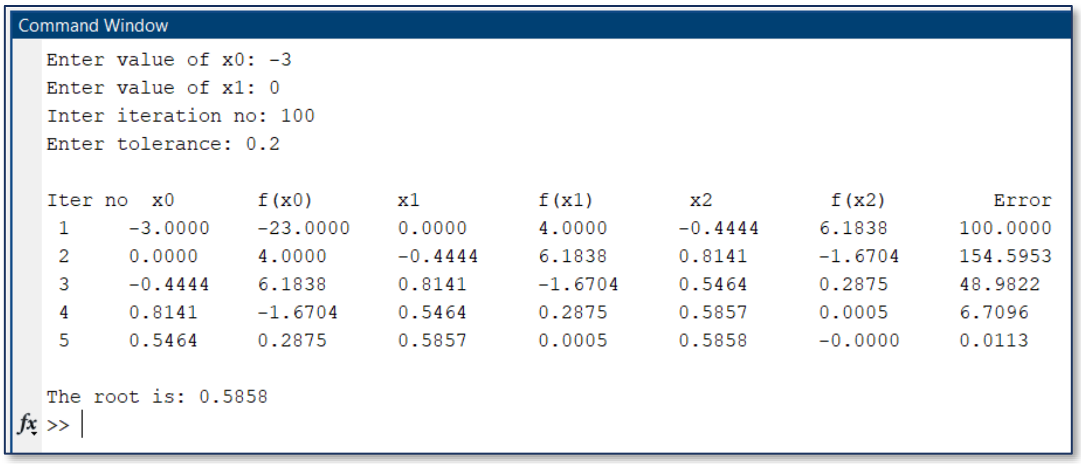
2.5.5 MATLAB Output

- Iteration for error <= 0.2% with interval [3,4]



```
Command Window
Enter value of x0: 3
Enter value of x1: 4
Inter iteration no: 100
Enter tolerance: 0.2

Iter no  x0        f(x0)       x1        f(x1)      x2        f(x2)      Error
 1      3.0000    -5.0000     4.0000    12.0000    3.2941    -1.7220    21.4286
 2      4.0000    12.0000     3.2941    -1.7220    3.3827    -0.4745    2.6186
 3      3.2941    -1.7220     3.3827    -0.4745    3.4164    0.0333     0.9861
 4      3.3827    -0.4745     3.4164    0.0333     3.4142    -0.0006    0.0648

The root is: 3.4142
fx >>
```

- Iteration for error <= 0.2% with interval [-3,0]



```
Command Window
Enter value of x0: -3
Enter value of x1: 0
Inter iteration no: 100
Enter tolerance: 0.2

Iter no  x0        f(x0)       x1        f(x1)      x2        f(x2)      Error
 1      -3.0000   -23.0000    0.0000    4.0000     -0.4444   6.1838     100.0000
 2      0.0000    4.0000      -0.4444   6.1838     0.8141    -1.6704    154.5953
 3      -0.4444   6.1838      0.8141    -1.6704    0.5464    0.2875     48.9822
 4      0.8141    -1.6704     0.5464    0.2875     0.5857    0.0005     6.7096
 5      0.5464    0.2875      0.5857    0.0005     0.5858    -0.0000    0.0113

The root is: 0.5858
fx >> |
```

2.6 Discussion & Analysis

- How does the choice of the initial interval affect the solution?

Using Newton Rapson method, when the interval was [3,4], the root was 3.4142 at a tolerance of 0.2%. The root, on the other hand, was -2 at a tolerance of 0.2% for the interval [-3,0]. Again, with Secant approach, root was 3.4142 at interval [3,4] at 0.2% tolerance and 0.5858 at interval [-3,0] at 0.2% tolerance. Because changing the interval affects both the root position and the root value. That is how the initial interval influences the solutions.

- Discuss the convergence of the Newton Rapson method and Secant method.

The Newton Rapson method has a linear convergence. This strategy iteratively reduces the value to get to the root. It is a straightforward procedure. But the Secant technique is not necessarily convergent. It occasionally displays output in a divergent condition. In the experiment, x2 was convergent for the range [3,4]. However, for the range [-3,0], x2 was divergent.

- How the speed of convergence and the error tolerance are related?

Convergence speed improves as tolerance is increased. However, the accuracy falls due to the log step size, which increases truncation error. When the tolerance was set to 2%, the step was reduced and the convergence was quick. Step rose by 0.2%, and convergence was delayed.