

Numerical Analysis Project Report

1- Karim M. Elsayad	- 6023
2- Hagar Omar Abouroumia	- 5943
3- Marwan Medhat El Nahas	- 5365
4- Youssef Ahmed Walid	- 5491
5- Ahmed M. Abdelrehim	- 5684

Contents:

- 1- Functions and code
- 2- How to Use
- 3- Installation guide
- 4- Sample runs

1 - Code Organization

Functions:

Bisection Function:

```
function ret = Bisection(app)
    x1, xu, funcm maxIter are given

    if func(x1) * func(xu) > 0
        No solution!

    else
        for i=1:1:MaxIter

            % compute midpoin xr
            xr = (xu + x1)/2;

            % compute test = f(x1) * f(xr)
            test = func(x1) * func(xr);

            if test < 0
                xr_old = xu;
                xu = xr;
            elseif test > 0
                xr_old = x1;
                x1 = xr;

            if test == 0
                ea = 0;
            else
                % approximate relative error
                ea = abs((xr - xr_old)/xr) * 100;

            if ea < Precision
                break;
```

```
ret = [xr, i];
```

Bisection Notes:

After choosing Bisection in the GUI, initialize $xr_old=0$, we take the input of the user in xl (x lower) & xu (x upper) , Then we take the function input (function expression) and get the value of $Function(xl)$ & $Function(xu)$ If $Function(xl) * Function(xu) > 0$ display No Solution Because they have the same sign.

Else we loop from $i=1$ to the Maximum Number of iterations taken from the input of the user in GUI or break if The Error = (absolute of $(xr-xr_old) * 100/xr$) is $<$ Precision.

At each iteration xr (new root) = $(xl+xu)/2$, Then get

$Function(xl) * Function(xr)$ if (< 0 $xr_old = xu$, $xu=xr$,iterate once again) else if (> 0 $xr_old = xl$, $xl=xr$,iterate once again) else(= 0 break)

Returns the xr , Number of Iterations of the Final Iteration which is the desired Root.

1) False Position (Regula Falsi) Function:

```
function ret = FalsePosition(app)
    xl = XlEditField_False.Value;
    xu = XuEditField_False.Value;
    xr_old=0;
    % on solution if both starting points have same sign
    if func(xl)*func(xu) > 0
        disp("No Solution!");
        xr = 0;
        i = 0;
        ErrorLabel.Text = "No solution in range";
    else
        for i=1:1:MaxIter
            % compute midpoin xr
            % minimize function calls
            fu = func(xu);
            fl = func(xl);
            xr = (xl * fu - xu * fl)/(fu - fl);

            % compute test = f(xl) * f(xr)
            test = fl * func(xr);
            if test < 0
                xr_old = xu;
                xu = xr;
            elseif test > 0
                xr_old = xl;
                xl = xr;
            end
            if test == 0
                ea = 0;
            else
                % approximate relative error
                ea = abs((xr - xr_old)/xr) * 100;
            end
            if ea < Precision
                break;
            end
        end
    end

    ret = [xr, i];
```

False Position (Regula Falsi) Notes:

After choosing False Position in the GUI, initialize $xr_old=0$, we take the input of the user in x_l (x lower) & x_u (x upper) , Then we take the function input (function expression) and get the value of $Function(x_l)$ & $Function(x_u)$ If $Function(x_l) * Function(x_u) > 0$ display No Solution Because they have the same sign.

Else we loop from $i=1$ to the Maximum Number of iterations taken from the input of the user in GUI or break if The Absolute Error = $(\text{absolute of } (xr - xr_old) * 100 / xr)$ is $<$ Precision.

At each iteration xr (new root) $= (x_l F(x_u) - x_u F(x_l)) / (F(x_u) - F(x_l))$,

Then get $Function(x_l) * Function(xr)$ if $(< 0$ $xr_old = x_u$, $x_u = xr$, iterate- once again) else if $(> 0$ $xr_old = x_l$, $x_l = xr$, iterate once again) else $(= 0$ break)

Returns the xr , Number of Iterations of the Final Iteration which is the desired Root.

2) Fixed Point Function:

```
function ret = Fixedpoint(app)
    x0 = X0EditField_Fixed.Value;
    syms gFunc(x)
    gFunc(x) = str2sym(gxEditField.Value);
    xr = x0;
    for i=1:1:MaxIter
        xr_old = xr;
        xr = gFunc(xr_old);
        ea = abs((xr - xr_old)/xr) * 100;

        if (ea < Precision)
            break
        end
    end
    ret = [xr, i];
end
```

Fixed Point Notes:

After choosing Fixed Point in the GUI, we take the input of the user for x_0 & $gFunc(x)$, initialize $xr = x_0$, we loop from $i=1$ to Maximum Number of Iteration taken from the input of the user in GUI.

Each Iteration make $x_{old} = xr$, calculate $xr = gFunc(xr_{old})$,

calculate Absolute Error = (absolute of $(xr - xr_{old}) * 100/xr$)

if ($< Precision$) break.

Returns the xr , Number of Iterations of the Final Iteration which is the desired Root.

Newton Raphson Function:

```
function ret = NewtonRaphson(app)
    xr_old = X0EditField_Newton.Value;
    for i=1:1:MaxIter
        xr = xr_old - ((func(xr_old))/(funcDiff(xr_old)));
        ea = abs((xr - xr_old)/xr) * 100;
        if (ea < Precision)
            break
        end
    end
    ret = [xr, i];
end
```

Newton Raphson Notes:

After choosing Fixed Point in the GUI, we take the input of the user for xr_old we loop from $i=1$ to Maximum Number of Iteration taken from the input of the user in GUI.

Each iteration $xr = xr_old - (\text{Function}(xr_old) / \text{Derivative}(\text{Func}(xr_old)))$,

calculate Absolute Error = $(\text{absolute of } (xr - xr_old) * 100 / xr)$

if ($< \text{Precision}$) break.

Returns the xr , Number of Iterations of the Final Iteration which is the desired Root.

3) Secant Function:

```
function ret = Secant(app)
    xi_old = X0EditField_Secant.Value;
    xi = X1EditField_Secant.Value;
    for i=1:1:MaxIter
        xiplus = xi - (func(xi)*(xi_old -
xi))/(func(xi_old)-func(xi));
        ea = abs((xiplus-xi)/xiplus)*100;
        if (ea < Precision)
            break
        end
        xi_old = xi;
        xi = xiplus;
    end
    ret = [xiplus, i];
end
```

-> Secant Notes:

After choosing Fixed Point in the GUI, we take the input of the user for xi & xi_old we loop from i=1 to Maximum Number of Iteration taken from the input of the user in GUI.

Each iteration $x_{i+1} = x_i - (f(x_i) * (x_i - x_{i-1})) / (f(x_i) - f(x_{i-1}))$

calculate Absolute Error = $(|x_i - x_{i-1}| * 100) / x_i$

if (< Precision) break.

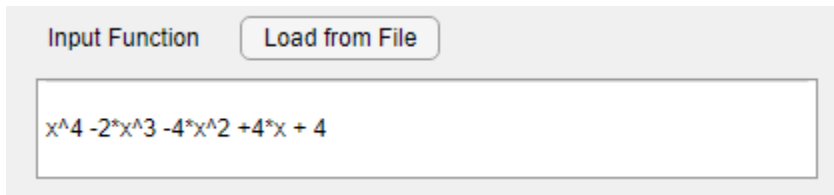
Else : $x_{i-1} = x_i$ & $x_i = x_{i+1}$.

Returns the xiplus, Number of Iterations of the Final Iteration which is the desired Root.

How to use

- No implicit multiplication. Use `*`. (wrong: `2x`, correct: `2 * x`). It's only a text box that doesn't provide advance parsing capabilities.
- Only enter the Left-hand side of the equation, RHS is assumed to be zero. Adding an equal sign results in an error.
- The buttons under the Input box tell what functions are known to work.
- You can use "load from file" button to read a function from a file, provided it's written in the same format mentioned above

- Example of an input function



The screenshot shows a user interface for an application. At the top, there are two buttons: "Input Function" and "Load from File". Below these buttons is a large text input box. Inside this box, the mathematical expression $x^4 - 2x^3 - 4x^2 + 4x + 4$ is entered. The text is rendered in a monospaced font, with superscripts for the powers of x.

Installation Guide:

As this app is made using app designer, it requires version 2016 or later to run.

- Put the three provided files (analyze, single_step_mode, plotFunction) into a folder. Alternatively, use the matlab function `appinfo = matlab.apputil.install(appfile)` to install using the provided `Analyzer_1.mlappinstall` file.
- After files are placed somewhere, add that place to path, or change current directory to that place. Ultimately, you want the files to be visible for matlab search, as it's the case with all functions.
- Run analyze through the matlab console/cli.

Sample Runs

Bisection

Analyzer

Input Function

Load from File

$x^4 - 2x^3 - 4x^2 + 4x + 4$

Sin

Cos

Tan

exp

sqrt

nthroot

Max Iterations

Precision

Plot Function

Solve!

Try Single Step with Bisection

X

i

Ea

Time elapsed

Method

☒ Bisection

☐ False-position

☐ Fixed point

☐ Newton-Raphson

☐ Secant

Xl

Xu

False position

Analyzer

Input Function

Load from File

$x^4 - 2x^3 - 4x^2 + 4x + 4$

Sin

Cos

Tan

exp

sqrt

nthroot

Max Iterations

50

Precision

1.000000e-02

Plot Function

Solve!

X

-1.414050906320

i

18

Ea

0.007367

Time elapsed

0.2499

Try Single Step with Bisection

Method

☐ Bisection

☒ False-position

☐ Fixed point

☐ Newton-Raphson

☐ Secant

Xl

0

Xu

0

Xl

-2

Xu

-1

	Iteration	xr	f(xr)	Precision Error
1	1	-1.0769	-1.1037	7.142
2	2	-1.1547	-1.0952	6.733
3	3	-1.2254	-0.9731	5.769
4	4	-1.2835	-0.7809	4.527
5	5	-1.3273	-0.5760	3.298
6	6	-1.3581	-0.3985	2.268
7	7	-1.3787	-0.2636	1.496
8	8	-1.3921	-0.1692	0.959
9	9	-1.4005	-0.1065	0.603
10	10	-1.4058	-0.0662	0.375
11	11	-1.4091	-0.0409	0.231
12	12	-1.4111	-0.0251	0.142
13	13	-1.4123	-0.0154	0.087
14	14	-1.4130	-0.0094	0.053
15	15	-1.4135	-0.0057	0.032
16	16	-1.4138	-0.0035	0.019
17	17	-1.4139	-0.0021	0.012
18	18	-1.4141	-0.0013	0.007
19	0	0	4	
20	0	0	4	

Fixed point

Analyzer

Input Function

Load from File

$x^2 - 2x + 1$

X

1.048387096774

Sin

Cos

Tan

exp

sqrt

nthroot

i

20

Max Iterations

20

Plot Function

Solve!

Ea

0.2347

Precision

1.000000e-01

Time elapsed

0.131

Try Single Step with Bisection

Method

☐ Bisection

☐ False-position

☒ Fixed point

☐ Newton-Raphson

☐ Secant

Xl

0

Xu

2

Xl

0

Xu

0

X0

2.5

g(x)

$(2x-1)/x$

	Iteration	xr	f(xr)	Precision Error
1	1	1.6000	0.3600	56.2500
2	2	1.3750	0.1406	16.3636
3	3	1.2727	0.0744	8.0357
4	4	1.2143	0.0459	4.8128
5	5	1.1765	0.0311	3.2143
6	6	1.1500	0.0225	2.3018
7	7	1.1304	0.0170	1.7308
8	8	1.1154	0.0133	1.3493
9	9	1.1034	0.0107	1.0817
10	10	1.0938	0.0088	0.8867
11	11	1.0857	0.0073	0.7401
12	12	1.0789	0.0062	0.6272
13	13	1.0732	0.0054	0.5383
14	14	1.0682	0.0046	0.4670
15	15	1.0638	0.0041	0.4091
16	16	1.0600	0.0036	0.3613
17	17	1.0566	0.0032	0.3214
18	18	1.0536	0.0029	0.2878
19	19	1.0508	0.0026	0.2592
20	20	1.0484	0.0023	0.2347

Newton Raphson

Analyzer

Input Function

Load from File

x^2 -2* x +1

SinCosTanexpsqrtnthroot

Max Iterations20

Precision1.000000e-01

Plot FunctionSolve!

Try Single Step with Bisection

X1.000732421875

i11

Ea0.07319

Time elapsed0.09087

Method

Bisection

False-position

Fixed point

Newton-Raphson

Secant

Xl0

Xl0

X00

X02.5

X00

	Iteration	xr	f(xr)	Precision Error
1	1	1.7500	0.5625	42.8571
2	2	1.3750	0.1406	27.2727
3	3	1.1875	0.0352	15.7895
4	4	1.0938	0.0088	8.5714
5	5	1.0469	0.0022	4.4776
6	6	1.0234	5.4932e-04	2.2901
7	7	1.0117	1.3733e-04	1.1583
8	8	1.0059	3.4332e-05	0.5825
9	9	1.0029	8.5831e-06	0.2921
10	10	1.0015	2.1458e-06	0.1463
11	11	1.0007	5.3644e-07	0.0732
12	0	0	1	0
13	0	0	1	0
14	0	0	1	0
15	0	0	1	0
16	0	0	1	0
17	0	0	1	0
18	0	0	1	0
19	0	0	1	0
20	0	0	1	0

Secant Method

Analyzer

Input Function

Load from File

2^x-x -x

X

0.641185744754

Sin

Cos

Tan

exp

sqrt

nthroot

i

4

Max Iterations

5

Precision

1.000000e-02

Plot Function

Solve!

Ea

0.0003923

Time elapsed

0.07228

Try Single Step with Bisection

Method

☐ Bisection

☐ False-position

☐ Fixed point

☐ Newton-Raphson

☒ Secant

Xl

0

Xu

2

Xl

0

Xu

0

X0

0

g(x)

0

X0

0

X0

0

X1

1

	Iteration	xr	f(xr)	Precision Error
1	1	0.6667	-0.0367	50.0000
2	2	0.6403	0.0013	4.1248
3	3	0.6412	-3.6336e-06	0.1452
4	4	0.6412	-3.5986e-10	3.9229e-04
5	0	0	1	0

Single Step Mode

