MM3110 - Expt 2 - Non Linear Equations

Vishal S
MM14B048

R Mythreyi
MM15B022

1 Question 1

The equation we need to solve is cubic in nature and hence it is non-linear. Since, it is a cubic equation a maximum of three real roots can be expected and it will have at least one real root. The equation takes the form :

$$x^3 - 6x^2 + 3x + 10 = 0 \tag{1}$$

This can be factorised as :

$$x^2(x-5) - x(x-5) - 2(x-5) = 0$$
$$(x-5)(x^2 - x - 2) = 0$$
$$(x-5)(x-2)(x+1) = 0$$

Thus, the solutions for equation 1 are $x = 5$, 2 and -1 . The same can be seen by plotting the function (see Figure 1). A GNU Octave loop can be written to solve the equation using *Newton-Raphson Method*.

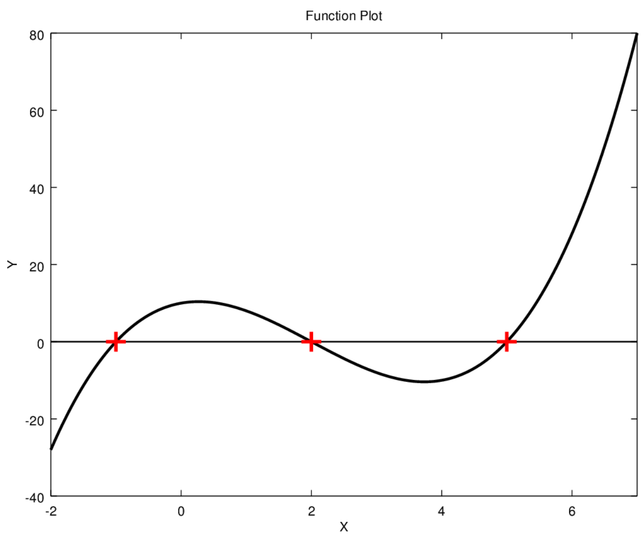

Figure 1: Plot of $x^3 - 6x^2 + 3x + 10 = 0$.

Definition 1. *Newton-Raphson Method* is an iterative method. In this method an approximate value for the root of the equation is calculated based on the output of the previous iteration. Initially, a guess is provided to start the iteration. Mathematically, the method can be written as ,

$$x_i = x_{i-1} - \frac{F(x_{i-1})}{F'(x_{i-1})} \tag{2}$$

Where,

x_{i-1} is the value from the previous iteration,
x_i is the value obtained from this iteration,
$F(x)$ is the function whose roots has to be found, and
$F'(x)$ is the derivative of the function with respect to x.

In this example the function $F(x)$ is a polynomial and hence its derivative can be analytically found. If the derivative cannot be obtained analytically, we can approximate the value of the derivative at x_{i-1} using central/forward/backward schemes from the functionś Taylor Series expansion. Geometrically, $(x_i, 0)$ is the intersection of the x-axis and the tangent of the graph of $F(x)$ at $(x_{i-1}, F(x_{i-1}))$. This iteration is carried on until $F(x_i)$ is equal to 0, or $|x_i - x_{i-1}|$ is less than the predefined convergence limit. The root to which the iteration converges is depends on the initial guess. Care should be taken such that there is no extremum between the guess and the root because $F'(x)$ becomes zero at extrema and our method would break down.

The above iteration can be written as a script in GNU OCTAVE.

```
function f = fun (x)
  f = power(x,3) - 6* power(x, 2) + 3*x + 10 ;
endfunction

function f = fundash (x)
  f = 3* power(x, 2) - 12*x + 3 ;
endfunction

init_guess = 2.49; % can be changed to new values
delX = 1; % can be any number greater than 0.0001 in our case
new_guess = 1; % can be anything
old_guess = init_guess
i = 0;

while (delX > 0.0001)
  fprintf("\n");
  fprintf("Iteration ");
  i
  fprintf("\n");
  if(xdash == 0 )
        printf("Derivative becomes zero. Please choose a different initial guess.");
  endif
  new_guess =  old_guess - (fun(old_guess) / fundash(old_guess))
  delX =  abs(new_guess -  old_guess)
  old_guess = new_guess;
  i++;
endwhile

fprintf("\n");
best_guess = new_guess
```

The variable `init_guess` can be changed to new values, to reach all the roots that we know exist. The output of the above code for `init_guess` of 2.49, -1.49 and 4.51 is the following:

```
 ------------------
init_guess =  2.4900

Iteration i = 0

new_guess =  1.9716
delX =  0.51842

Iteration i =  1

new_guess =  2.0000
delX =  0.028424

Iteration i =  2

new_guess =  2
```

```
delX =    5.1017e-06

best_guess =  2

 ------------------

 ------------------
init_guess = -1.4900

Iteration i = 0

new_guess = -1.0870
delX =  0.40299

Iteration i =  1

new_guess = -1.0035
delX =  0.083462

Iteration i =  2

new_guess = -1.0000
delX =  0.0035391

Iteration i =  3

new_guess = -1.0000
delX =    6.2674e-06

best_guess = -1.0000

 ------------------

 ------------------
init_guess =  4.5100

Iteration i = 0

new_guess =  5.1945
delX =  0.68450

Iteration i =  1

new_guess =  5.0164
delX =  0.17807

Iteration i =  2

new_guess =  5.0001
delX =  0.016299

Iteration i =  3
```

```
new_guess =  5.0000
delX =    1.3331e-04

Iteration i =  4

new_guess =  5
delX =    8.8854e-09

best_guess =  5

 ------------------
```

2 Question 2

The given equation :

$$x^2 + 2\log(|x|) = 0 \tag{3}$$

can be solved by plotting as well as by using the built-in GNU Octave function `fzero()`. The plot of the function is in Figure 2. To use the function `fzero()`, we need to pass as argument, two values of x that we know straddle a zero of the function. The first argument in the function `fzero()` usage in the code shows the initial guesses used to find the zeros. The below GNU

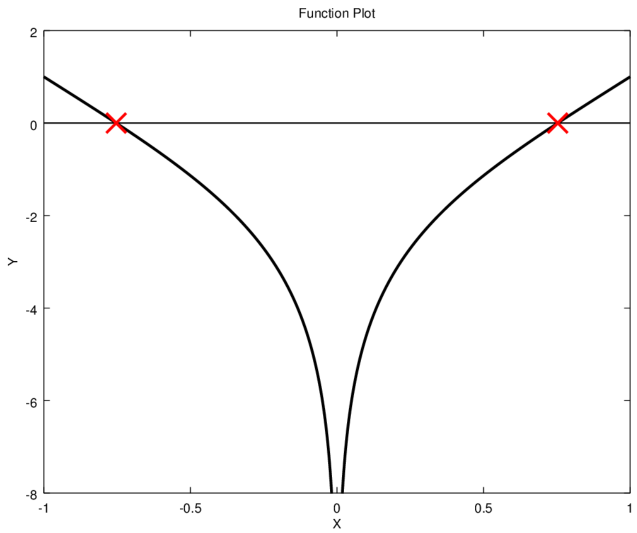

Figure 2: Plot of $x^2 + 2\log(|x|) = 0$.

Octave code can be used to solve the function for both its roots. We need to make sure while plotting that $x = 0$ is excluded for the function blows up to $-\infty$ at $x = 0$.

```
x1 = -1:0.001:-0.01;
x2 = 0.01:0.001:1;
x = [x1 , x2]; % we need to exclude 0 in the range.

y = @(x) power(x,2) + 2*log(abs(x));
zer = @(x) 0 * x;
hold on;
plot(x, y(x), 'k', 'linewidth',2);
plot(x, zer(x), 'k', 'linewidth', 1);
plot(z,[0 0],'rx','linewidth',2, 'markersize',15);
ylim([-8 2]); xlabel("X"); title("Function Plot"); ylabel("Y");

z(1) = fzero(y, [-1 -0.5]);
z(2) = fzero(y, [1 0.5]);
```

```
z % z holds the zeros of our equation.
```

The output for the above code is:

```
z =

  -0.75309   0.75309
```

3 Question 3

The question can solved using the following GNU OCTAVE code.

```
fprintf("Positive number chosen: "); a = 0.39875
fprintf("\n");
init_guess = 0.9
delX = 1; % can be any number greater than 0.0001 in our case
new_guess = 1; % can be anything
old_guess = init_guess;
i = 0;

while (delX > 0.0001)
  fprintf("\n");
  fprintf("Iteration ");
  i
  fprintf("\n");
  new_guess = ((a / old_guess) + old_guess ) / 2
  delX = abs(new_guess - old_guess)
  old_guess = new_guess;
  i++;
endwhile

fprintf("\n");
best_guess = new_guess
```

The output for the above code is:

```
 ------------------
Positive number chosen: a =  5

init_guess =  3

Iteration i = 0

new_guess =  2.3333
delX =  0.66667

Iteration i =  1

new_guess =  2.2381
delX =  0.095238

Iteration i =  2

new_guess =  2.2361
delX =  0.0020263

Iteration i =  3

new_guess =  2.2361
```

```
delX =    9.1814e-07

best_guess =  2.2361

 ------------------

 ------------------
Positive number chosen: a =  345

init_guess =  18

Iteration i = 0

new_guess =  18.583
delX =  0.58333

Iteration i =  1

new_guess =  18.574
delX =  0.0091555

Iteration i =  2

new_guess =  18.574
delX =    2.2564e-06

best_guess =  18.574

 ------------------

 ------------------
Positive number chosen: a =  49

init_guess =  4

Iteration i = 0

new_guess =  8.1250
delX =  4.1250

Iteration i =  1

new_guess =  7.0779
delX =  1.0471

Iteration i =  2

new_guess =  7.0004
delX =  0.077456

Iteration i =  3

new_guess =  7.0000
```

```
delX =    4.2851e-04

Iteration i =  4

new_guess =  7
delX =    1.3116e-08

best_guess =  7

 ------------------

 ------------------
Positive number chosen: a =  0.39875

init_guess =  0.90000

Iteration i = 0

new_guess =  0.67153
delX =  0.22847

Iteration i =  1

new_guess =  0.63266
delX =  0.038866

Iteration i =  2

new_guess =  0.63147
delX =  0.0011938

Iteration i =  3

new_guess =  0.63147
delX =    1.1285e-06

best_guess =  0.63147

 ------------------
```

The *divide and average method* is nothing but the *Newton-Raphson method* recast in another form. From definition 1 and equation 2 we have,

$$x_i = x_{i-1} - \frac{F(x_{i-1})}{F'(x_{i-1})} \qquad \text{(2 revisited)}$$

Since we are trying to solve the square root of a natural number, $F(x)$ is defined as

$$x^2 - a = 0$$

Similarly $F'(x)$ is defined as,

$$\frac{dF(x)}{dx} = 2x,$$

Thus equation 2 can now be written as:

$$x_i = x_{i-1} - \frac{{x_{i-1}}^2 - a}{2x_{i-1}}$$
$$x_i = x_{i-1} - \frac{x_{i-1}}{2} + \frac{a}{2x_{i-1}}$$
$$x_i = \frac{x_{i-1}}{2} + \frac{a}{2x_{i-1}}$$

$$x_i = \frac{x_{i-1} + \frac{a}{x_{i-1}}}{2} \tag{4}$$

Equation 4 is same as what is given in the question as *divide and average method*. Thus, we've proved that it's a recast form of the *Newton-Raphson Method*.

4 Question 4

In this question, we will be solving two non-linear functions simultaneously . The equations that need to be solved are

$$x^2 + y^2 = 17$$
$$y = x^2 + 2x - 5$$

These two functions can be plotted using a predefined function in MATLAB called `fimplicit()`. This function can be used to plot implicit functions. The plot is given below in Figure 3. From

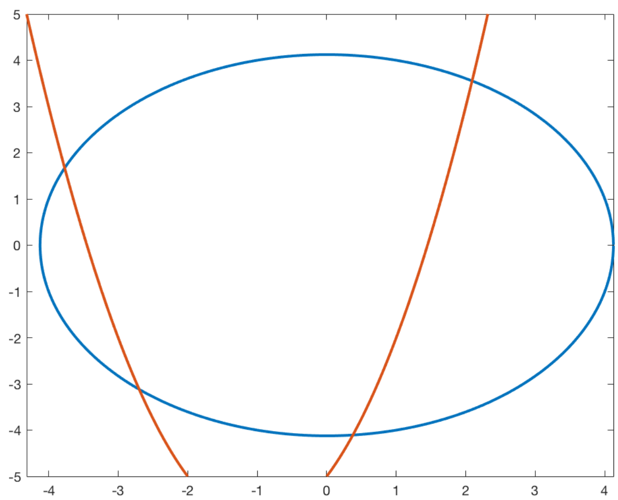

Figure 3: The plot obtained using `fimplicit()`.

the plot, we can infer that there are 4 solutions to this set of non-linear equations. We can solve these equations using another predefined function `fsolve`, which is an extrapolation of the `fzero()` function used in Question 2 to a system of Non-Linear Equations. The MATLAB code given below was used to solve the question:

```
fun = @non_linear;
%x0 = [1.5  3];
%x0 = [0  0 ];
%x0 =  [-2.5  -3];
%x0 = [-4  3]
x0 = [1  1]

x = fsolve(fun,x0);
```

```
x

fimplicit(@(x,y)x.^2 + y.^2 - 17,'LineWidth' , 2.0)
hold on;
fimplicit(@(x,y)x.^2 + 2*x - 5 - y,'LineWidth' , 2.0)
function F = non_linear(x)

F(1) = x(1)*x(1) + x(2)*x(2) - 17;
F(2) = x(1)*x(1) + 2*x(1) - x(2) - 5 ;
end
```

The output of the code is given below. Different solutions can be obtained by varying the initial guess `x0`.

```
------------------
x0 =

     1     1

x =

    2.0909    3.5536
------------------

------------------
x0 =

    -4     3

x =

   -3.7696    1.6705
------------------

------------------
x0 =

   -2.5000   -3.0000

x =

   -2.6976   -3.1182
------------------

------------------
x0 =

     0     0

x =

    0.3763   -4.1059
------------------
```

`fimplicit()` by default assumes the domain to be $[-5, 5]$ and 151 points along each direction. This corresponds to an incremental value of 0.0662. This default `MeshDensity` of 151 points can be changed using the *Name-Value Pair Arugument* (`'MeshDensity', value`) inside the `fimplicit()` function. The importance of the above value is that, the accurate determination of the shape of the plot is influenced by the incremental value. The plot becomes better for smaller values of incremental value. However, for smaller values of incremental values the size of the array becomes big and the run time also increases. The same plot for a `MeshDensity` of `10` (versus `151`) is given in Figure 4.

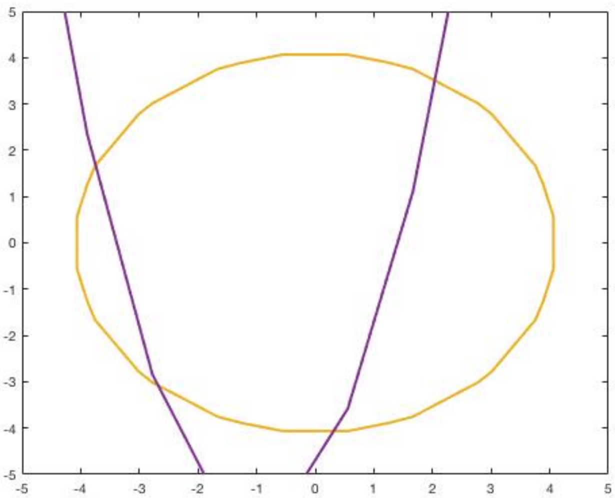

Figure 4: Plot using `MeshDensity` of 10.