Spring 2022 Class Project

Numerical Methods / Numerical Analysis

MATLAB Implementation

Instructor: Shahzad Ahmad | Total Marks: 20

Akif Ejaz - BSCE19042

1. **Problem Statement:**

   Implement Newton-Raphson method using MATLAB to compute the drag coefficient c needed for a parachutist of mass **m = First Two Digits of Your Registration Number ÷ 2 kg** to have a velocity of **Second Last Digit of Your Registration Number + 40 m/s** after free falling for time **t = Last Digit of your Registration Number + 5 secs**. *Note:* The acceleration due to gravity is 9.81 m/s^2.

   The drag coefficient is given by

$$f(c) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) - v$$

a. Formulate an iterative formula for the Newton-Raphson method

Using the given equation for estimating c as:

$$f(c) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) - v$$

Now, we have to find out 'f(c) = 0'. We will be using the NR method for this problem. The MATLAB code for this is given below:

```
m = 19/2;          %m = First Two Digits of Your Registration Number ÷ 2
v = 4 + 40;        % Second Last Digit of Your Registration Number + 40 m/s
t = 2 + 5;         %time t = Last Digit of your Registration Number + 5 secs
gravity = 9.8;

%Setting x as symbolic variable that representing c
syms x;

% Input Section
%v(t) = g*m/c *(1- exp(-(c/m)*t) );          % orignal equation of v
y = gravity*m/x *( 1 - exp(-(x/m)*t) ) - v;  % equation for estimating c

a = 3;        %a = input('Enter initial guess: ');
e = 0.0001;   %e = input('Tolerable error: ');
N = 100;      %N = input('Enter maximum number of steps: ');
funct = [];
error = [];

% Initializing step counter
```

```
step = 1;

% Finding derivate of given function
g = diff(y,x);

% Finding Functional Value
fa = eval(subs(y,x,a));
while abs(fa)> e
    fa = eval(subs(y,x,a));
    ga = eval(subs(g,x,a));
    if ga == 0
        disp('ERROR: Division by zero.');
        break;
    end
    b = a - fa/ga;
    fprintf('step num=%d\t\tApproximated value=%f\t\tFunction Value When 
f(a)=%f\n',step,a,fa);
    a = b;
    if step>N
        disp('ERROR: Not convergent ! TRY CHANGE THE INITIAL GUESS a ');
        break;
    end

funct(step) = a;          % saving the value of step in a vector
error(step) =((1.31-a)/1.31)*100;    % vector for error at every iteration
step = step + 1;
end

fprintf('Approximated Value for f(c) is %f\n', a);

% true value plot
x_axis = 0:1:step-2;
tem = 1.31;
const = @(x_axis)(tem).*x_axis.^(0);
plot(x_axis, const(x_axis),'--','linewidth',2); hold on;

% calculated numerically
x_axis = 0:1:step-2;    % axis for plot
plot(x_axis,funct,'linewidth',2);

xlabel("Step n"); ylabel("Approx. value of func");
title("Plot of True Value & Approximated Values Over Steps");
legend('true value','approx value');
```

b. Choose an appropriate initial guess to start iterations in order to achieve convergence. If the solution diverges re-choose the initial guess.

For my case roll no: 19042, trying out the initial guess which is greater then 0 and less than 10 gives me the approximated value without divergence with limited steps.

let's say for initial guess 3 I get as:

MATLAB Output:

```
Command Window

step num=1      Approximated value=3.000000     Function Value When f(a)=-16.369252
step num=2      Approximated value=0.557951     Function Value When f(a)=12.247651
step num=3      Approximated value=1.192319     Function Value When f(a)=1.648620
step num=4      Approximated value=1.306914     Function Value When f(a)=0.041976
step num=5      Approximated value=1.309987     Function Value When f(a)=0.000029
Approximated Value for f(c) is 1.309989
>>
```

Figure 1: Output of the code with initial guess '3'

Approximated Value for f(c) is = **1.309989.** I'm getting the value within 5 iterations.

c. Calculate the approximated error after every iteration and tabulate your results.

The approximated error is computed as:

$$Et = \frac{true\ value - approximated\ value}{true\ value} \times 100\%$$

Iteration (step)	True Value	Approximated	Et %
1	1.31	3.0	57.408318
2	1.31	0.557951	8.9833145
3	1.31	1.192319	0.2355850
4	1.31	1.306914	0.0010293
5	1.31	1.309987	0.0008672

MATLAB Code for Error Calculation:

```
%%  Error Plot

x_axis = 0:1:step-2;    % axis for plot
plot(x_axis,error,'linewidth',2);
title("Error from NR Method for f(c)  ");
xlabel("Step n"); ylabel("Error");
```

MATLAB Output:

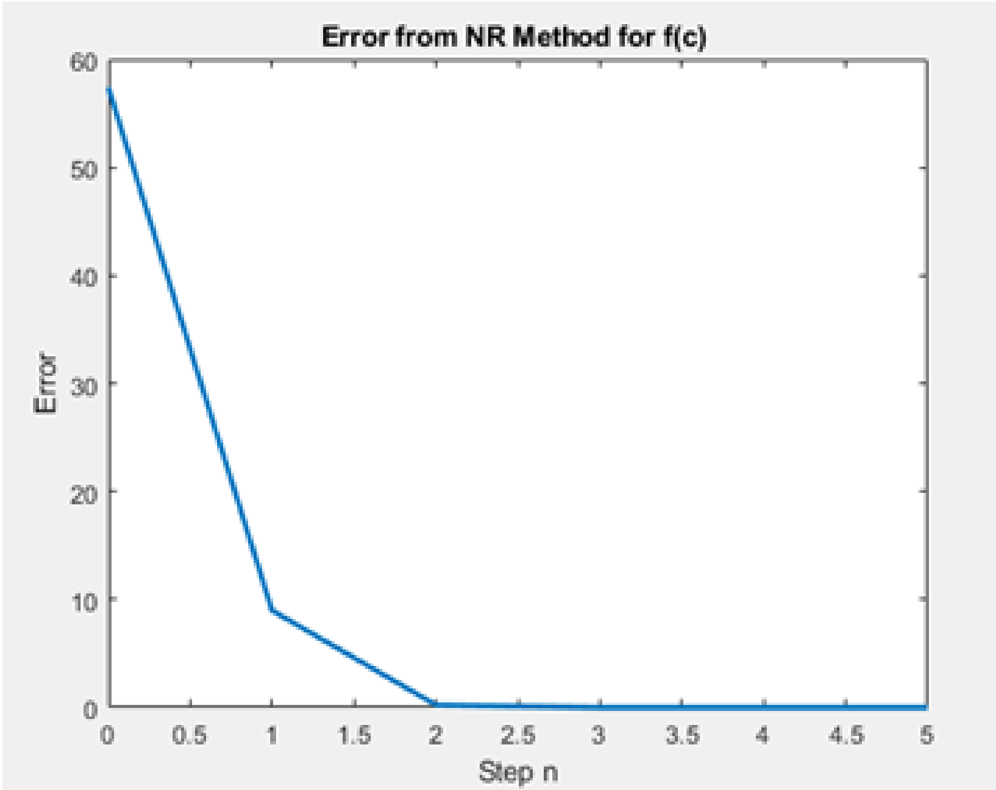


Figure 2: Error plot for NR

d. The ending criteria of the numerical computation is such that the consecutive calculations have a precision of 1e-4 (**For Even Registration Number**) and 1e-5 (**For Odd Registration Number**).

With even roll num (19042) I have chosen 0.0001 error for ending the loop. as line no 14 in my code above (part a)

MATLAB CODE:

```
e = 0.0001;    %e = input('Tolerable error: ');
```

e. Plot the computed drag coefficient values with respect to the number of iterations to show convergence.

With the initial guess as 3 the plot for drag coefficient values with respect to the number of iterations is as follow:

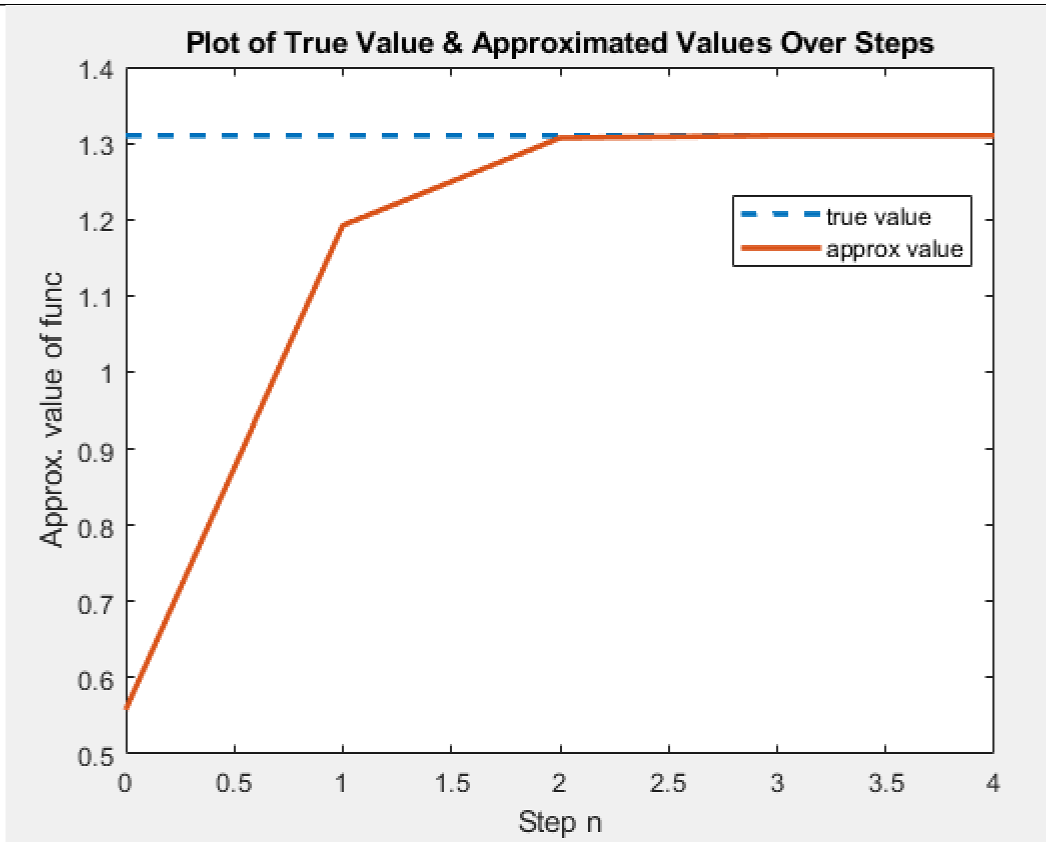


Figure 3: Plot Approximated Values VS Step sizes

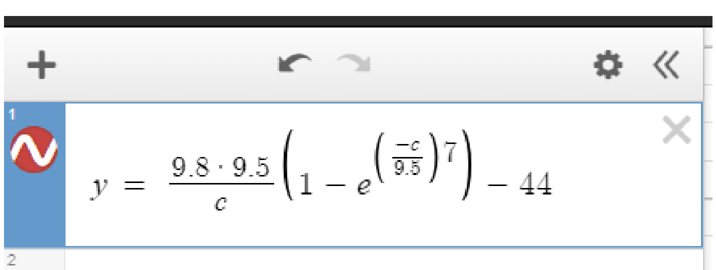
f. Validate the computed value.

So, in order to validate I just using the traditional method. With different values trying to plot the equation as given for calculation drag cofficient and identifying where the f(c) is equal to 0.

The Given equation is:

$$f(c) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) - v$$

Input this equation with all the values substituting, in graphing calculator we get:



We get something like this: [source: Desmos | Graphing Calculator]

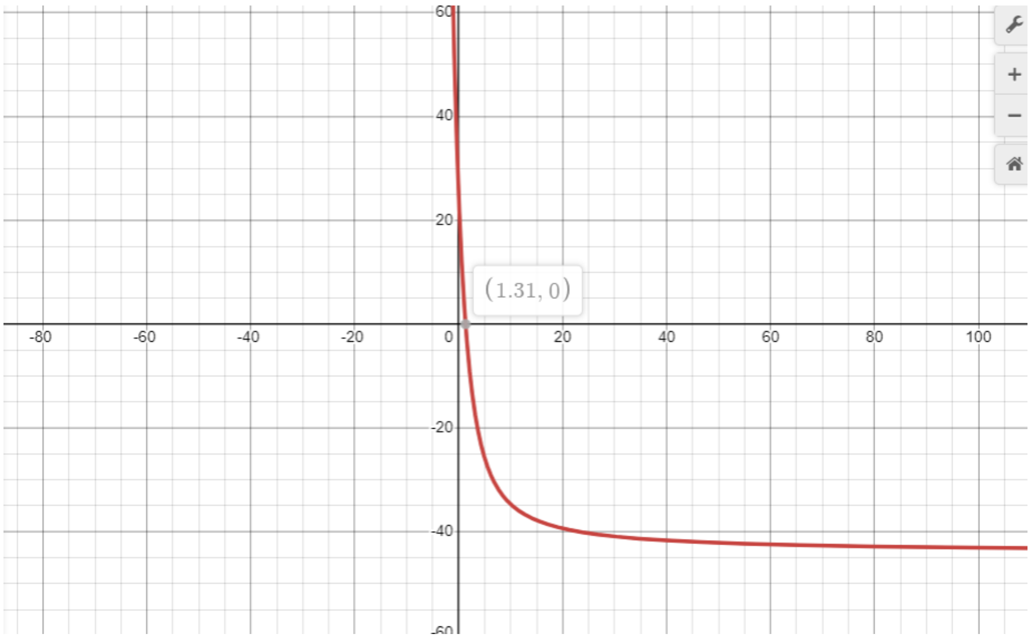


Figure 4: Plot for the given equation f(c)

We can see that function has value 0 at around **1.31**. From our NR method we get value around **1.309989.** We can clearly see that our value matches with the true value. Hence, (with some error) we are sure that this calculation is correct.