RULES

1. This is the **version 1.0**. In case there are any corrections for the solutions of Exercise 1, we will post an updated version on our website. You can follow the changes in the exercises by the **Version History** section below.

Version History

v1.0 Exercise is released.

v.1.1 The code in Question 4, Part B is changed

1. The root of the function is solved by the following methods in the interval [0,2] and the results are given below:

a) Bisection Method:

Step#	Xlower	f(Xlower)	Xupper	f(Xupper)	Xroot	f(Xroot)	$Error_{RA}$ (%)
0	0.000000	-2.000000	2.000000	4.389056	1.000000	-0.281718	-
1	1.000000	-0.281718	2.000000	4.389056	1.500000	1.481689	33.333333
2	1.000000	-0.281718	1.500000	1.481689	1.250000	0.490343	20.000000
3	1.000000	-0.281718	1.250000	0.490343	1.125000	0.080217	11.111111
4	1.000000	-0.281718	1.125000	0.080217	1.062500	-0.106404	5.882353
5	1.062500	-0.106404	1.125000	0.080217	1.093750	-0.014551	2.857143
6	1.093750	-0.014551	1.125000	0.080217	1.109375	0.032463	1.408451
7	1.093750	-0.014551	1.109375	0.032463	1.101563	0.008864	0.709220
8	1.093750	-0.014551	1.101563	0.008864	1.097656	-0.002867	0.355872
9	1.097656	-0.002867	1.101563	0.008864	1.099609	0.002993	0.177620
10	1.097656	-0.002867	1.099609	0.002993	1.098633	0.000062	0.088889
11	1.097656	-0.002867	1.098633	0.000062	1.098145	-0.001403	0.044464
12	1.098145	-0.001403	1.098633	0.000062	1.098389	-0.000671	0.022227
13	1.098389	-0.000671	1.098633	0.000062	1.098511	-0.000305	0.011112
14	1.098511	-0.000305	1.098633	0.000062	1.098572	-0.000122	0.005556
15	1.098572	-0.000122	1.098633	0.000062	1.098602	-0.000030	0.002778

METU CIVIL ENGINEERING DEPARTMENT

b) Regula-Falsi (False Position) Method:

Step#	Xlower	f(Xlower)	Xupper	f(Xupper)	Xroot	f(Xroot)	$Error_{RA}$ (%)
0	0.000000	-2.000000	2.000000	4.389056	0.626071	-1.129753	-
1	0.626071	-1.129753	2.000000	4.389056	0.907327	-0.522309	30.998356
2	0.907327	-0.522309	2.000000	4.389056	1.023530	-0.217000	11.353118
3	1.023530	-0.217000	2.000000	4.389056	1.069533	-0.085982	4.301254
4	1.069533	-0.085982	2.000000	4.389056	1.087411	-0.033418	1.644062
5	1.087411	-0.033418	2.000000	4.389056	1.094306	-0.012890	0.630153
6	1.094306	-0.012890	2.000000	4.389056	1.096958	-0.004957	0.241766
7	1.096958	-0.004957	2.000000	4.389056	1.097977	-0.001904	0.092790
8	1.097977	-0.001904	2.000000	4.389056	1.098369	-0.000731	0.035618
9	1.098369	-0.000731	2.000000	4.389056	1.098519	-0.000281	0.013673
10	1.098519	-0.000281	2.000000	4.389056	1.098576	-0.000108	0.005249
11	1.098576	-0.000108	2.000000	4.389056	1.098598	-0.000041	0.002015

c) Newton-Raphson Method:

Step#	X_i	$f(X_i)$	$f'(X_i)$	X_{i+1}	$Error_{RA}$ (%)
1	0.3	-1.65014	1.349859	1.522455	80.2949797
2	1.522455	1.583462	4.583462	1.176982	29.352449
3	1.176982	0.244566	3.244566	1.101604	6.84249284
4	1.101604	0.00899	3.00899	1.098617	0.27195443
5	1.098617	1.34E-05	3.000013	1.098612	0.00040708

d) As you can see from the above calculations, Newton-Raphson Method is faster than Regula-Falsi and Bisection Methods in addition to the fact that Regula-Falsi Method is faster than Bisection Method. The reason why N-R is fastest is it has quadratic convergence if the root is simple like this case. Regula-Falsi method is generally faster than Bisection Method since the algorithm is improved.

2.

a) If the problem is solved by Secant-Method with initial guesses 0.3 and 0.4 and much lower error tolerance using MATLAB, the following codes can be used:

Code 1: Defining function in a different **.m* file called *givenfunction.m.*

```
function y=givenfunction(x)
y=exp(x)-3;
end
```

METU - CIVIL ENGINEERING DEPARTMENT

Code 2: Secant Method that uses the pre-defined function called *givenfunction*.

```
clear all
clc

%% SECANT METHOD
format long
%%% INPUT
tol=5*10^-7;
err(1,1)=tol+1;
%initial guesses
x(1,1)=0.3;
x(2,1)=0.4;
%%%

i=1;
while err(i,1)>=tol

%find the value of the function at first initial guess
f_value(i,1)=givenfunction(x(i,1));

%find the value of the function at second initial guess
f_value(i+1,1)=givenfunction(x(i+1,1));

%find new x value
x(i+2,1)=x(i+1,1)-(f_value(i+1,1)*(x(i+1,1)-x(i,1)))/(f_value(i+1,1)-f_value(i));
%find the value of function at new x
f_value(i+2,1)=givenfunction(x(i+2,1));

err(i+1,1)=100*abs((x(i+2,1)-x(i+1,1))/x(i+2,1)); %find percent relative error
i=i+1; %update i

    %%Maximum iteration number criteria
    iteration=i-1;
    if iteration>=20
        break;
    end

end
iter=(1:i-1)'; %number of iterations

%displaying results in tabulated form
result=[iter x(1:i-1,:) x(2:i,:) f value(1:i-1,:) f value(2:i,:) x(3:i+1,:) err(2:i,:)]
```

The result is:

Step#	X_{i-1}	X_i	$f(X_{i-1})$	$f(X_i)$	X_{i+1}	$Error_{RA}$ (%)
1	0.300000	0.400000	-1.650141	-1.508175	1.462350	72.646776
2	0.400000	1.462350	-1.508175	1.316092	0.967301	51.178386
3	1.462350	0.967301	1.316092	-0.369165	1.075745	10.080764
4	0.967301	1.075745	-0.369165	-0.067825	1.100153	2.218608
5	1.075745	1.100153	-0.067825	0.004625	1.098595	0.141819
6	1.100153	1.098595	0.004625	-0.000053	1.098612	0.001608
7	1.098595	1.098612	-0.000053	0.000000	1.098612	0.000001
8	1.098612	1.098612	0.000000	0.000000	1.098612	0.000000

METU-CE

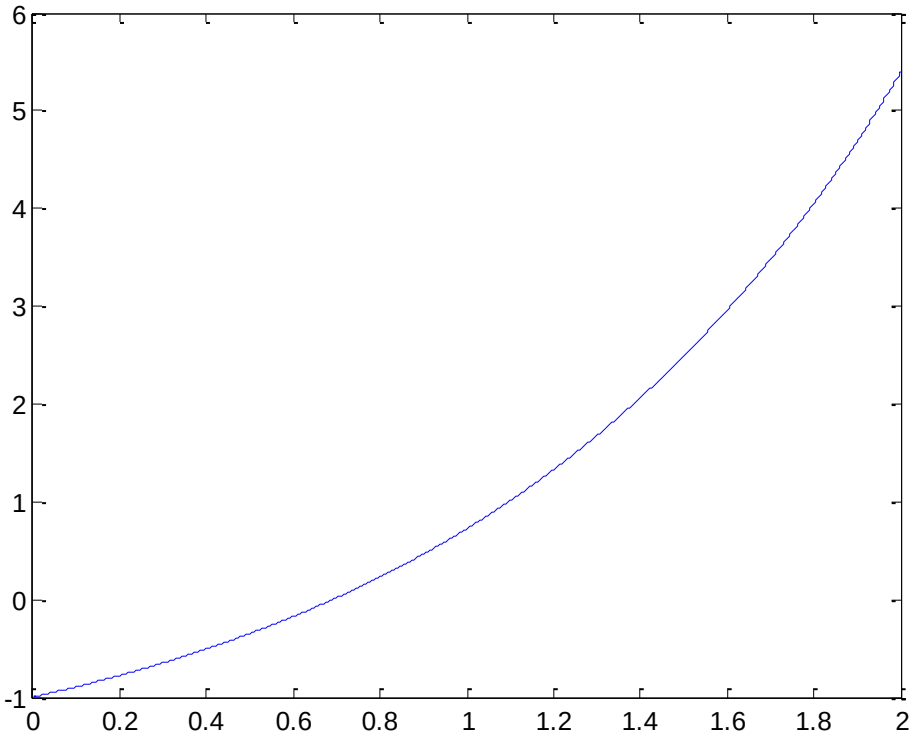
b) To plot the function in the given interval, the following code can be used:

```
clear all
clc

x=0:0.0001:2; %x values from 0 to 2 increasing by 0.0001
y=exp(x)-2;   %corresponding function values

plot(x,y)     %plotting x vs y
```

The resulting figure is given below:



c) The results would not change.

d) As an example, if we use $x_0=0$ and $x_1=2$, the results and convergence velocity do not change with the same error criteria.

METU-CIV

3)

a) The Newton-Raphson method is not defined when the derivative of the function is zero at the initial guess. That initial guess where the N-R method will not work can be found as:

$$f'(x) = 1 - \sin x$$
$$f'(x_0) = 1 - \sin x_0 = 0 \rightarrow x_0 = -1.5708$$

b) The results can be found by Secant Method as:

Step#	X_{i-1}	X_i	$f(X_{i-1})$	$f(X_i)$	X_{i+1}	$Error_{RA}$ (%)
1	0.0000	0.2500	-1.0000	-0.7189	0.8894	71.8912
2	0.2500	0.8894	-0.7189	0.2595	0.7198	23.5618

c) The computation of the derivative of the function at each iteration is the drawback of Newton-Raphson Method. If the function has a complicated expression one can choose approximating the root by Secant Method.

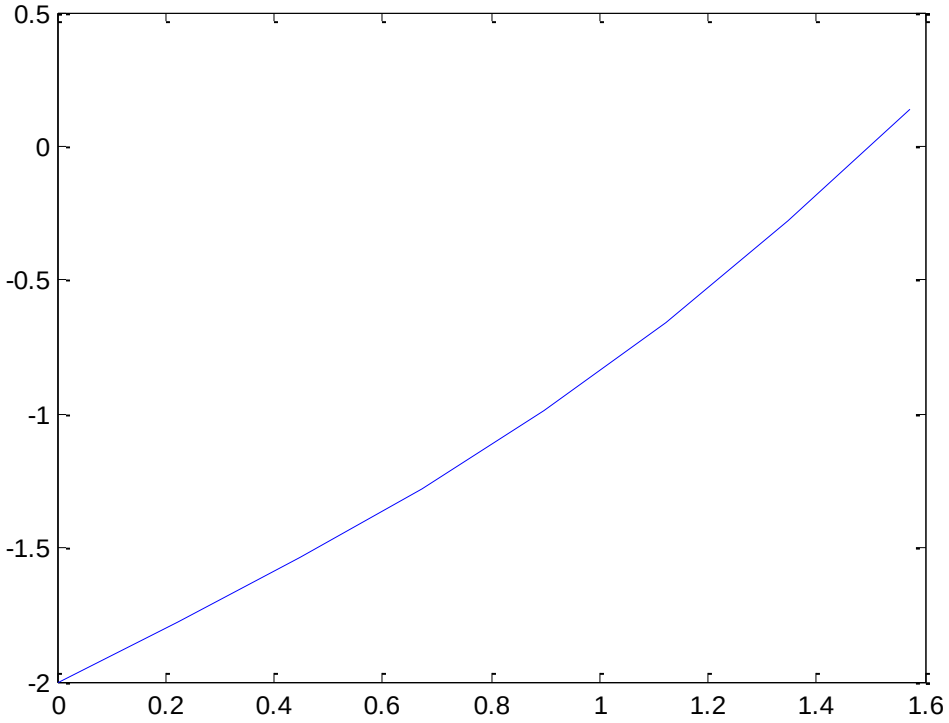
4)

a) To plot the function in the given interval using 6 points within the interval the following code can be used:

```
clear all
clc
x=0:pi/14:pi/2;    %x values from 0 to 2 increasing by 0.0001
y=2*x-2-sin(x);    %corresponding function values
plot(x,y)          %plotting x vs y
```

METU - CIVIL ENGINEERING DEPARTMENT

The resulting figure is given below:



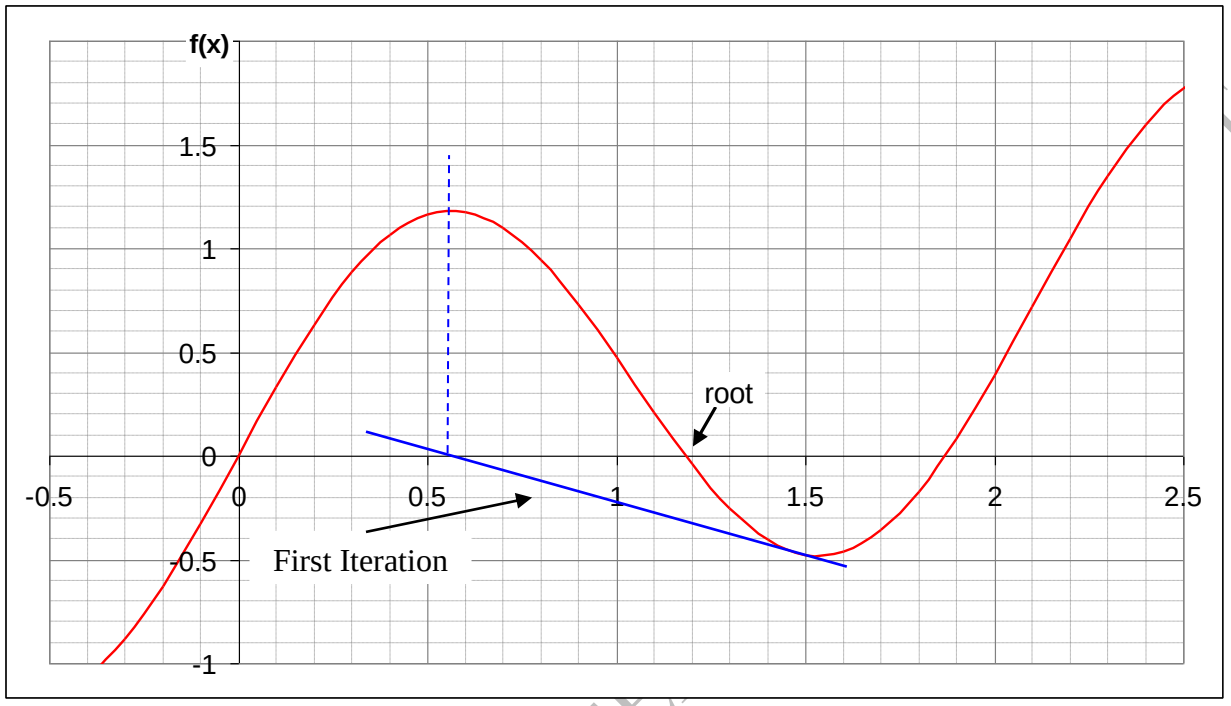
b) Using error tolerance as: $\varepsilon_s = 0.05$. *The concept of finding error tolerance in the question is not included in the content of this course. So use error tolerance as 0.05.*

Step#	X_{i-1}	X_i	$f(X_{i-1})$	$f(X_i)$	X_{i+1}	$Error_{RA}$ (%)
1	0.75	1.00	-1.18	-0.84	1.62	38.21
2	1.00	1.62	-0.84	0.24	1.48	9.20
3	1.62	1.48	0.24	-0.03	1.50	1.08
4	1.48	1.50	-0.03	0.00	1.50	0.03

METU - CIVIL ENGIN

5)

a) The wrong initial guess is 1.5 since the derivative of the function at that point is close to 0.



Similarly the further iterations can be shown on the graph.

b) The results can be tabulated for two iteration and given below:

Step#	**X_i**	**$f(X_i)$**	**$f'(X_i)$**	**X_{i+1}**	**$Error_{RA}$ (%)**
1	1.00000	0.47445	-2.63664	1.17995	15.25035
2	1.17995	0.00551	-2.43190	1.18221	0.19179

METU - CIVIL ENGINEERING