

1. Submit your **report** as a **Microsoft Word** file on Canvas before the deadline.
2. **Annotate** all your answers in your submission using the following **format**:

Answer to Question X Part Y

3. Name your file in following format: **Lab6_SID_LastName_FirstName**
4. Type **your name** and **student ID** on the first page of your report.
5. Include your Python code at the end of your report.

10 marks

Q1) In this question, you will use the **bisection** method to compute the root of a function.

The Python code of the bisection method is provided to you at the end of this question. Some information about the Python code:

- You can enter the function $f(x)$ in the code. The code finds the root of $f(x)$ using the bisection method.
- The Python code performs 20 iterations of the bisection method.
- It generates the matrix X .
- The first column of X is the left endpoint of the interval (x_l) in each iteration of the bisection method.
- The second column of X is the midpoint of the interval (x_r) in each iteration of the bisection method.
- The third column of X is the right endpoint (x_u) in each iteration of the bisection method.
- The fourth column of X is the percentage of the relative error in each iteration.
- The code prints the root of the function.
- The Python code has the vector “**Itr**” that contains the index of iterations.
- You need to enter the function $f(x)$, and x_l and x_u in the code.

Consider the function $f(x) = x^4 - 7x^3 - 30x^2 + 400x - 1000$.

a) Enter $f(x)$ in Python. Find the root of the function by running the code for $x_l = 4.5$ and $x_u = 6$. Enter the estimate of the root in your report.

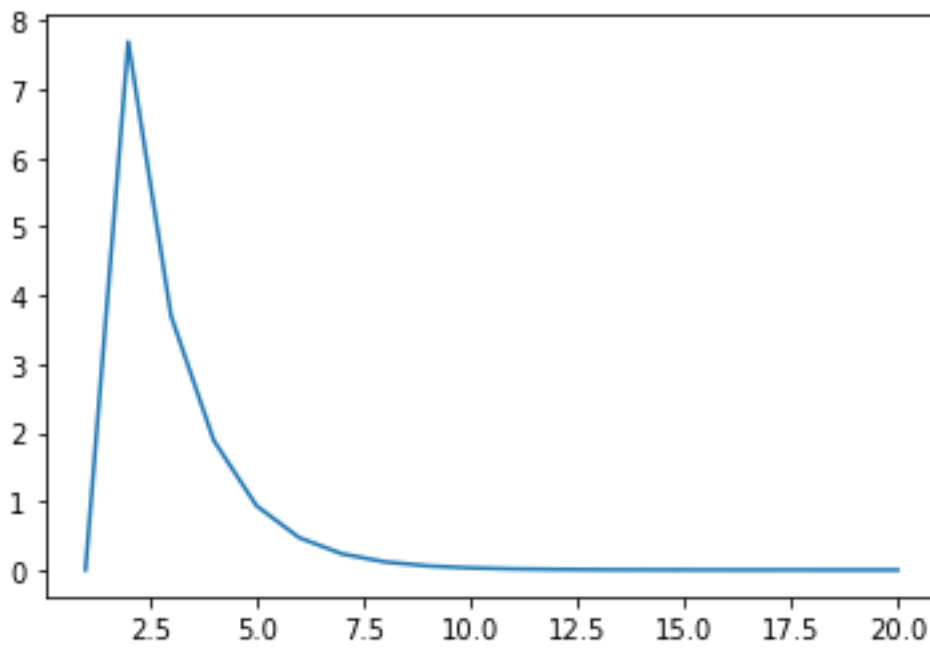
The root is 4.999999523162842

b) Evaluate the function at the root of the function. Include the answer in your report.

-3.576278345462924e-05

c) Plot the percentage of the relative error versus the number of iterations. Include your plot in your report.

Hint: The percentage of the relative error is in the last column of the matrix.



d) How will the error behave as the number of iterations becomes large? Why?

The error goes to zero as the algorithm converges.

Python code of the bisection:

```

def func(x):
    f= Enter your function here
    return f

import numpy as np
import matplotlib.pyplot as plt
import math
Xu=0;
Xl=0;
Iteration=20
d= np.zeros(Iteration)
S=(Iteration,4)
X = np.array(np.zeros(S))
X[0,0]=Xl
X[0,2]=Xu
Xo=0;
for i in range(0,Iteration-1):
    X[i,1]=(X[i,0]+X[i,2])/2;
    if func(X[i,0])*func(X[i,1])<0:
        X[i+1,0]=X[i,0]
        X[i+1,2]=X[i,1]
    if func(X[i,2])*func(X[i,1])<0:
        X[i+1,0]=X[i,1]
        X[i+1,2]=X[i,2]
    if i>0:
        X[i,3]=np.abs((X[i,1]-Xo)/X[i,1])*100
        Xo=X[i,1]

i=i+1
X[i,1]=(X[i,0]+X[i,2])/2
X[i,3]=np.abs((X[i,1]-Xo)/X[i,1])*100
print('The root is',X[i,1])
Itr= np.linspace(1, Iteration, Iteration)

```

function $f(x)$
 x_u
 x_l
Matrix X
 X_r
 ϵ_k
Number of iterations

Q2) In this question, you will use the **secant** method to compute the root of a function.

The Python code of the secant method is provided to you at the end of this question. Some information about the Python code:

- You can enter the function $f(x)$ in the code. It will find the root of $f(x)$ using the secant method.
- The Python code performs 8 iterations of the secant method.
- It generates the matrix X .
- The first column of X is the estimate of the root at each iteration.
- The second column of X is the percentage of the relative error in each iteration.
- The code prints the root of the function.
- The Python code has the vector “**Itr**” that contains the index of iterations.
- You need to enter the function $f(x)$, and x_0 and x_{-1} in the code.

Consider the function $f(x) = x^3 - 2x - 1$.

a) Enter $f(x)$ in Python. Find the root of the function by running the code for $x_0 = 2$ and $x_{-1} = 1$. Enter the estimate of the root in your report.

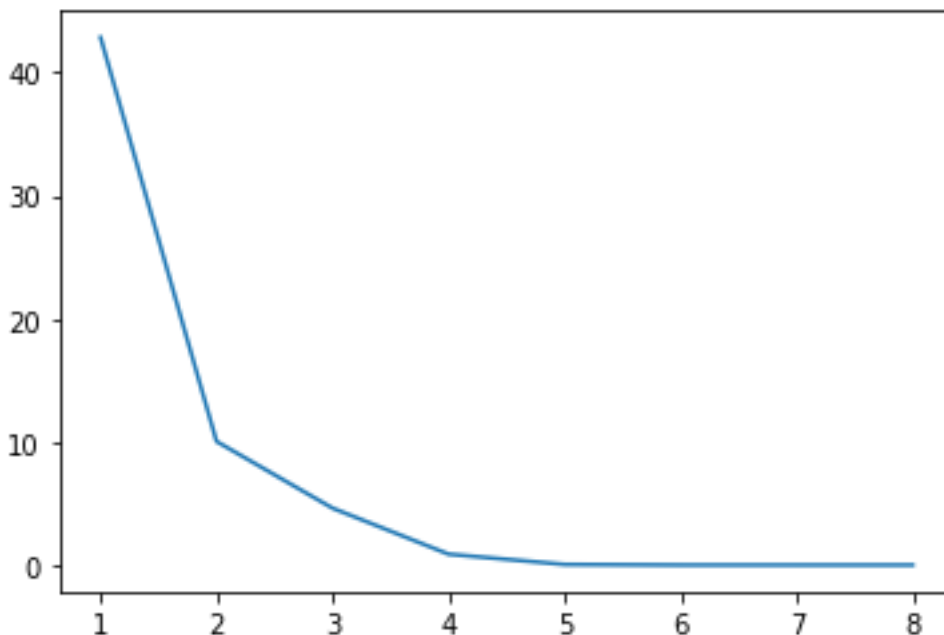
The root is 1.6180339887498638

b) Evaluate the function at the root of the function. Include the answer in your report.

-1.8207657603852567e-13

c) Plot the percentage of the relative error versus the number of iterations. Include your plot in your report.

Hint: The percentage of the relative error is in the last column of the matrix.



d) How will the error behave as the number of iterations becomes large? Why?

The error goes to zero as the algorithm converges.

Python code of the secant method:

```

def func(x):
    f= Enter your function here ← function  $f(x)$ 
    return f

import numpy as np
import matplotlib.pyplot as plt
import math
X0=1; ←  $x_0$ 
X_1=0; ←  $x_{-1}$ 
Iteration=8
d= np.zeros(Iteration)
S=(Iteration,2)
X = np.array(np.zeros(S))
X[0,0]=X0-(func(X0)*(X0-X_1))/(func(X0)-func(X_1))
X[0,1]=np.abs((X[0,0]-X0)/X[0,0])*100
X[1,0]=X[0,0]-(func(X[0,0])*(X[0,0]-X0))/(func(X[0,0])-func(X0))
X[1,1]=np.abs((X[1,0]-X[0,0])/X[1,0])*100

for i in range(2,Iteration-1):
    X[i,0]=X[i-1,0]-(func(X[i-1,0])*(X[i-1,0]-X[i-2,0]))/(func(X[i-1,0])-func(X[i-2,0])) ←  $x_k$ 
    X[i,1]=np.abs((X[i,0]-X[i-1,0])/X[i,0])*100 ←  $\epsilon_k$ 

print('The root is',X[i,0])
Itr= np.linspace(1, Iteration, Iteration) ← Number of iterations

```

Name:

Student ID:

Q1)