

Higher Institute of Engineering & Technology, El-Beheira

Computer Engineering Department

Third assignment in numerical analysis

**The numerical solution of a system of non-linear equations using Newton’s Method.)**

Under supervision of Dr.Mahmoud Gamal

|  |  |
| --- | --- |
| Team | ID |
| Mohamed Yosry El-Zarka | 19100 |
| Youssef Mohamed El-Sheheimy | 19124 |
| Omar Abd Al-Halim Khalil | 19138 |

**Source code in python: -**

1 import math

2 from sympy import \* #for differentiation & mathematical functions

3 import numpy as np #matrix operations

4

5 pi=3.141592653589793

6 e=2.718281828459045

7

8 print('Project for "Numerical analysis". under the supervision of Dr. Mahmoud Gamal')

9 print('by:')

10 print('\t\tMohamed Yosry ElZarka 19100')

11 print('\t\tYoussef Mohamed Elsheheimy 19124')

12 print('\t\tOmar Abd Al-Halim Khalil 19138\n')

13

14 print("This is a program to calculate the numerical solution of a system of non-linear equations using (Newton's method).\n")

15

16 print("""

17 you can use parentheses () in addition to the following mathematical operators:

18 (+ Add), (- Subtract), (\* Multiply), (/ Divide), (% Modulus), (// Floor division), (\*\* Exponent)

19 you can also use the following constants:

20 \t pi=3.141592653589793

21 \t e=2.718281828459045

22 note: Trigonometric functions sin(x), asin(x), cos(x), acos(x), tan(x), atan(x) 'equivalent of tan-1(x)' use radian values.

23       log(x,y)= log(x) / log(y) ,,, ln(x)

24 """)

25

26 def check\_equalization(recent\_x,previous\_x):

27     for i in range (0, len(recent\_x)):

28         if recent\_x[i] != previous\_x[i]:

29             return False

30     return True

31

32 decimal\_point\_precision=4

33

34 while True:

35     n=int( input('Enter the number of equations: ') )

36     equations, equations\_values, jacobian\_matrix, jacobian\_values= [] , [0]\*n , [] , []

37

38     for i in range(0,n):

39         equations.append( str(input("Enter the equation #{}:    0 = ".format(i+1))) )

40         jacobian\_matrix.append([])

41         jacobian\_values.append([0]\*n)

42     last\_x , current\_x= [] , []

43     for i in range(0,n):

44         last\_x.append( float(input("Enter the initial x{} = ".format(i+1))) )

45

46     for i in range(0,n): #partial differntiaion matrix

47         for j in range(0,n):

48             jacobian\_matrix[i].append(  diff( equations[i] , 'x'+str(j+1) ) )

49

50     print("\njacobian matrix=",jacobian\_matrix)

51

52     print("\n        ",end="")

53     for i in range(0,n):

54         print("x{}          ".format(i+1),end="")

55     print("")

56     print("i=0",end="")

57     for i in range(0,n):

58         last\_x[i]=round( last\_x[i] , decimal\_point\_precision )

59         print(" | %.4f | "%last\_x[i],end="")

60     print("")

61

62     dictionary\_of\_last\_x={}

63

64     for iterations in range(1,500): #maximum number of iterations is 500

65         for i in range(0,n): #updating the values of matrices

66             dictionary\_of\_last\_x['x'+str(i+1)]=last\_x[i]

67         for i in range(0,n):

68             equations\_values[i]=round( eval(equations[i],dictionary\_of\_last\_x) , decimal\_point\_precision)

69         for i in range(0,n):

70             for j in range(0,n):

71                 jacobian\_values[i][j]=round( eval(str(jacobian\_matrix[i][j]),dictionary\_of\_last\_x) , decimal\_point\_precision )

72

73         A = np.array(last\_x) #matrix declarations

74         B = np.array(jacobian\_values)

75         C = np.array(equations\_values)

76

77         current\_x=np.subtract(A, np.dot( np.linalg.inv(B)  , C ) ) #matrix operations

78

79         print("i={}".format(iterations),end="")

80         for i in range(0,n):

81             current\_x[i]=round( current\_x[i] , decimal\_point\_precision )

82             print(" | %.4f | "%current\_x[i],end="")

83         print("")

84         if check\_equalization(current\_x,last\_x):

85             break

86         last\_x=current\_x

87

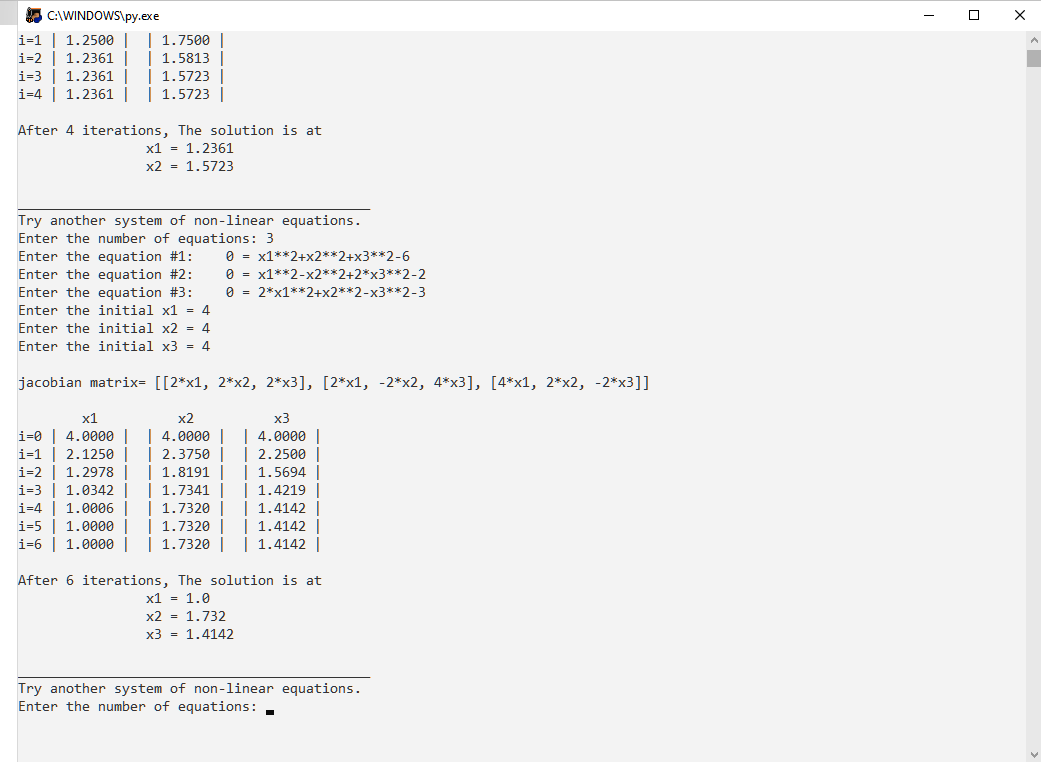
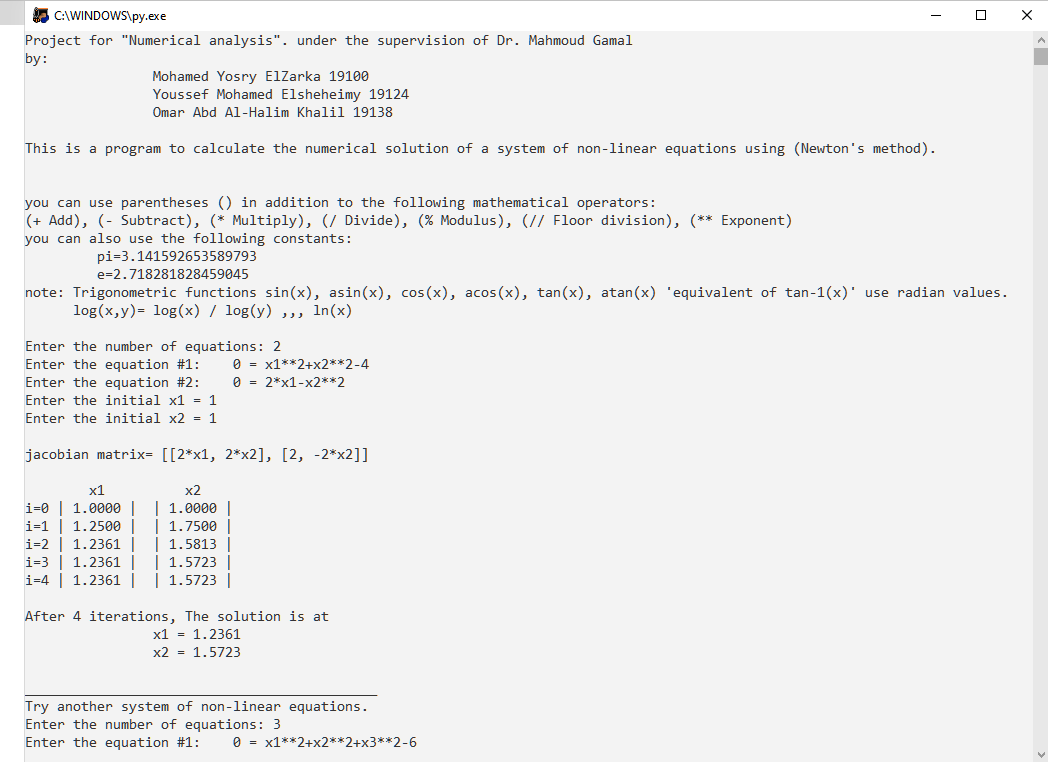
88     print("\nAfter",iterations,"iterations, The solution is at")

89     for i in range(0,n):

90         print("\t\tx{} =".format(i+1),last\_x[i])

91     print("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

92     print("Try another system of non-linear equations.")

******The program**