# NC Assignment 1

# 11762 Muhammad Kashif

## Question 1 Derivative

## Code

```
import math as m
import sympy as s
x = s.Symbol('x')
eq = s.sqrt(x)-s.cos(x)
ans = s.diff(eq).evalf(subs={x:2})
print("True Value = ",ans)
a = m.sqrt(2+0.3)-m.cos(2+0.3)
b = m.sqrt(2)-m.cos(2)
apprx = (a-b)/0.3
print("Approximatly value = ",apprx)
print("True Error = ",abs(apprx-ans))
```

## Output

```
True Value =  1.26285081741896
Approximatly value =  1.1749690372329877
True Error =  0.0878817801859677
```

## Question 2 RegulaFalsi Method

## Code

```
def f(x):
   return m.exp(-2*x)+4*x**2-36
def Regulafalsi(xL,xU):
   for i in range(15):
      xR = ((xL*f(xU))-(xU*f(xL)))/(f(xU)-f(xL))
      print("  ",i,"\t\t",round(xL,4),"\t",round(xU,4),"\t",round(xR,4),"\t",abs(round(f(xR),4)))

      if f(xR)<0:
         xU = xR
      else:
         xL = xR
      if abs(f(xR))<= 0.0001:
         break
print("================================================================================")
print("                    Regula Falsi Method")
print("================================================================================")
print("Iteration\txL\t\txU\t\txR\t\tTolarance")
Regulafalsi(1,2)
```

## Output

```
================================================================================
                            Regula Falsi Method
================================================================================
Iteration       xL              xU              xR              Tolarance
   0             1       2       3.6815          18.2155
   1             3.6815          2       2.8796          2.8274
   2             3.6815          2.8796          2.9874          0.2994
   3             3.6815          2.9874          2.9986          0.0307
   4             3.6815          2.9986          2.9998          0.0031
   5             3.6815          2.9998          2.9999          0.0003
   6             3.6815          2.9999          2.9999          0.0
```

# Question 3 Muller's Method

# Code

```
import numpy as np
def f(x):
   return 2*x - np.exp(x) + 3*np.cos(x)
def muller( x0, x1, x2):
   for i in range(100):
      h1 = x1 - x0
      h2 = x2 - x1
      d1 = (f(x1) - f(x0)) / h1
      d2 = (f(x2) - f(x1)) / h2
      d = (d2 - d1) / (h2 + h1)
      b = d2 + h2*d
      D = np.sqrt(b**2 - 4*f(x2)*d)
      if abs(b-D) < abs(b+D):
         E = b + D
      else:
         E = b - D
      h = -2*f(x2) / E
      p = x2 + h
      print("  ",i,"\t\t",round(x0,5),"\t",round(x1,5),"\t",round(x2,5),"\t",round(p,5),"\t",round(abs(h),5))
      if abs(h) < 0.00001:
         break
      x0 = x1
      x1 = x2
      x2 = p
print("================================================================================")
print("                    Muller's Method")
print("================================================================================")
print("Iteration\tx0\t\tx1\t\tx2\t\txR\t\tTolarance")
muller(-2, 0, 2)
```

# Output

```
================================================================================
                            Muller's Method
================================================================================
Iteration       x0              x1              x2              xR              Tolarance
   0             -2      0       2       1.12278         0.87722
   1             0       2       1.12278         1.23721         0.11443
   2             2       1.12278         1.23721         1.2397          0.00249
   3             1.12278         1.23721         1.2397          1.23971         1e-05
   4             1.23721         1.2397          1.23971         1.23971         0.0
```

# Question 4 Prove A(BC)=AB+BC

## Code

```
import numpy as np
A= np.array([[1,2,3],[4,5,6]])
B= np.array([[2,3],[4,5],[6,7]])
C = np.array([[9,3],[1,2]])
result1 = np.array([[0,0],[0,0],[0,0]])
result = np.array([[0,0],[0,0]])
result2 = np.array([[0,0],[0,0],[0,0]])
result3 = np.array([[0,0],[0,0]])
print("A(BC) = AB + BC")
#BC
for i in range(3):
   for j in range(2):
      for k in range(2):
         result1[i,j] +=B[i,k]*C[k,j]
#A(BC)
for i in range(2):
   for j in range(2):
      for k in range(3):
         result[i,j] += A[i,k]*result1[k,j]
print("A(BC)")
print(result)
#AB
for i in range(2):
   for j in range(2):
      for k in range(3):
         result3[i,j] +=A[i,k]*B[k,j]
print("AB")
print(result3)
print("BC")
print(result1)
print("AB + BC is not possible because the size of the matrix is not same")
```

## Output

```
A(BC) = AB + BC
A(BC)
[[286 152]
 [655 350]]
AB
[[28 34]
 [64 79]]
BC
[[21 12]
 [41 22]
 [61 32]]
AB + BC is not possible because the size of the matrix is not same
```