3 MARCH 2022

NUMERICAL COMPUTING (CS325)

Course Instructor:
Sir Jamil Usmani

NUMERICAL COMPUTING (CS325)
PROJECT
LAB – 2

Group Members:

- Mohammad Basil Ali Khan (20K-0477)
- Ali Jodat (20K-0155)
- Abdul Ahad Shaikh (20K-0319)
- Mohammad Umer (20K-0225)

Project Title:

LAB 2: Interpolation and Polynomial Approximation

Aim:

To understand the fundamental concepts of scientific programming using python.

Description:

We selected three methods of Lab2.

1. Lag grange Interpolation
2. Newton Divided Difference
3. Newton Forward and Backward

First we have studied the algorithm of then we have written the programming of that method.

IDE and Programming Language:

We have chosen python programming language and IDE we are using is Visual Studio Code.

Library Used:

- ✓ Used panda library to make data frame

Implementation and Code Snippets:

- ### Lag grange Interpolation:

Formula:

$$f(x) = f_0\partial_0(x) + f_1\partial_1(x) + f_2\partial_2(x) + \cdots + f_N\partial_N(x)$$

Where $\partial_i(x)$ can be written as;

$$\partial_i(x) = \frac{\prod_{i=0}^{N}; i \neq j\ (x - x_j)}{\prod_{i=0}^{N}; i \neq j\ (x_i - x_j)}$$

Algorithm:

Step 1: Read number of data N.

Step 2: Read data Xi and Yi from I = 0 to I = N.

Step 3: Read value of independent variables say x whose corresponding value of dependent say y is to be determined.

Step 4: Initialize: y = 0

Step 5: For i = 0 to N

Set p = 1

For j =0 to N

If i ≠ j then

Calculate product = product * (x - Xj)/(Xi - Xj)

End If

Next j

Calculate y = y + product * Yi

Next i

Step 6: Display value of y as interpolated value.

Code Snippets:

```
Lagrange_Interpolation_Method.py


def EnterNumberOfData():
    data = input("Enter Number of Data points: ")
    return int(data)

def Enter_Value():
    val = input("Enter value x to find: ")
    return float(val)

def getData(num):
    for i in range(num):
        val1 = input("Enter x" + str(i) + ": ")
        val2 = input("Enter y" + str(i) + ": ")
        X_values.append(float(val1))
        Y_values.append(float(val2))
    return

Num_Of_Data = EnterNumberOfData()

Degree = Num_Of_Data - 1

x = Enter_Value()
y = 0

X_values = []
Y_values = []

getData(Num_Of_Data)
print("\n")
for i in range(Degree+1):
    product = 1
    for j in range(Degree+1):
        if j != i:
            product = product * ((x - X_values[j])/(X_values[i]-X_values[j]))
    y = y + Y_values[i] * product

print("\nResult: ")
print("x = " + str(x))
print("P(x) = " + str(y))
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\FAST\4th Semester\Numerical Computing\NC Project\Mid II Submission> python -u "e:\FAST\4th Semester\Numerical Computing\NC Project\Mi
d II Submission\Lagrange_Interpolation_Method.py"
Enter Number of Data points: 4
Enter value x to find: 3


Enter x0: 3.2
Enter y0: 22
Enter x1: 2.7
Enter y1: 17.8
Enter x2: 1
Enter y2: 14.2
Enter x3: 4.8
Enter y3: 38.3



Result:
x = 3.0
P(x) = 20.211960717301274
PS E:\FAST\4th Semester\Numerical Computing\NC Project\Mid II Submission>
```

- ## Newton Forward and Backward:

Formula:

- **Forward**

$$\boldsymbol{P_n(x) = \sum_{i=0}^{n} \binom{p}{i} \Delta^i f_i(x) \;; p = \frac{x - x_0}{h} \;; h = x_1 - x_0}$$

- **Backward**

$$\boldsymbol{P_n(x) = \sum_{i=0}^{n} \binom{p}{i} \nabla^i f_i(x) \;; p = \frac{x - x_n}{h} \;; h = x_1 - x_0}$$

Algorithm:

- **Forward**

Step 1: Read number of data (n)

Step 2: Read data points for x and y:

    For i = 0 to n-1

        Read Xi and Yi,0

    Next i

Step 3: Read calculation point where derivative is required (xp)

Step 4: Generate forward difference table

    For i = 1 to n-1

        For j = 0 to n-1-i

            Yj,i = Yj+1,i-1 - Yj,i-1

        Next j

    Next i

Step 5: Calculate finite difference: h = X1 - X0

Step 6: Set sum = 0 and sign = 1

Step 7: Calculate sum of different terms in formula to find derivatives using Newton's forward difference formula:

For i = 1 to n-1-index

term = (Yindex, i)i / i

sum = sum + sign * term

sign = -sign

Next i

Step 8: Divide sum by finite difference (h) to get result first_derivative = sum/h

Step 9: Display value of first_derivative

- **Backward**

Step 1: Read number of data (n)

Step 2: Read data points for x and y:

For i = 0 to n-1

Read Xi and Yi,0

Next i

Step 3: Read calculation point where derivative is required (xp)

Step 4: Generate backward difference table

For i = 1 to n-1

For j = n-1 to i (Step -1)

Yj,i = Yj,i-1 - Yj-1,i-1

Next j

Next i

Step 5: Calculate finite difference: h = X1 - X0

Step 6: Set sum = 0

Step 7: Calculate sum of different terms in formula to find derivatives using Newton's backward difference formula:

For i = 1 to index

term = (Yindex, i)i / i

sum = sum + term

Next i

Step 8: Divide sum by finite difference (h) to get result

first_derivative = sum/h

Step 9: Display value of first_derivative

Code Snippets:

- **Forward**

Newton_Forward_Formula_Method.py

```python
import pandas as pd

def Calculating_p(p, n):
    temp = p
    for i in range(1, n):
        temp = temp * (p - i)
    return float(temp)

def Factorial(n):
    Fact = 1
    for i in range(2, n + 1):
        Fact = Fact * i
    return int(Fact)

Num_of_data = int(input("Enter number of data: "))
print("\nEnter values of X: ")
x = []
for i in range(Num_of_data):
    num1 = input("X" + str(i) + " : ")
    x.append(num1)

y = [[0 for i in range(Num_of_data)] for j in range(Num_of_data)]
print("\nEnter f(x) values: ")
for i in range(Num_of_data):
    num2 = input("Y" + str(i) + " : ")
    y[i][0] = float(num2)

for i in range(1, Num_of_data):
    for j in range(Num_of_data - i):
        y[j][i] = float(y[j + 1][i - 1]) - float(y[j][i - 1])

print(pd.DataFrame(y))

value = float(input("\nEnter Value to Interpolate: "))

sum = y[0][0]
p = (float(value) - float(x[0])) / (float(x[1]) - float(x[0]))
for i in range(1,Num_of_data):
    sum = float(sum) + (Calculating_p(p, i) * float(y[0][i])) / Factorial(i)

print("\nValue at ", value, "is", sum)
```

- **Backward**

Newton_Backward_Formula_Method.py

```python
import pandas as pd

def Calculating_p(p, n):
    temp = p
    for i in range(1, n):
        temp = temp * (p + i)
    return float(temp)

def Factorial(n):
    Fact = 1
    for i in range(2, n + 1):
        Fact = Fact * i
    return int(Fact)

Num_of_data = int(input("Enter number of data: "))
print("\nEnter values of X: ")
x = []
for i in range(Num_of_data):
    num1 = input("X" + str(i) + " : ")
    x.append(num1)

y = [[0 for i in range(Num_of_data)] for j in range(Num_of_data)]
print("\nEnter f(x) values: ")
for i in range(Num_of_data):
    num2 = input("Y" + str(i) + " : ")
    y[i][0] = float(num2)

for i in range(1, Num_of_data):
    for j in range(Num_of_data-1, i-1, -1):
        y[j][i] = float(y[j][i - 1]) - float(y[j - 1][i - 1])

print(pd.DataFrame(y))
```

```python
value = float(input("\nEnter Value to Interpolate: "))

sum = float(y[Num_of_data-1][0])
p = (float(value) - float(x[Num_of_data-1])) / (float(x[1]) - float(x[0]))
for i in range(1,Num_of_data):
    sum = float(sum) + (Calculating_p(p, i) * float(y[Num_of_data-1][i])) / Factorial(i)

print("\nValue at ", value, "is", sum)
```

Output:

- ### Forward

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                  Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\FAST\4th Semester\Numerical Computing\NC Project\Mid II Submission> python -u "e:\FAST\4th Semester\Numerical Computing\NC Project\Mi
d II Submission\Newton_Forward_Formula_Method.py"
Enter number of data: 4

Enter values of X:
X0 : 1.7
X1 : 1.8
X2 : 1.9
X3 : 2

Enter f(x) values:
Y0 : 0.3979849
Y1 : 0.3399864
Y2 : 0.2818186
Y3 : 0.2238908
          0         1         2         3
0  0.397985 -0.057998 -0.000169  0.000409
1  0.339986 -0.058168  0.000240  0.000000
2  0.281819 -0.057928  0.000000  0.000000
3  0.223891  0.000000  0.000000  0.000000

Enter Value to Interpolate: 1.72

Value at  1.72 is 0.38641839040000003
PS E:\FAST\4th Semester\Numerical Computing\NC Project\Mid II Submission>
```

- ### Backward

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL                                  Code

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS E:\FAST\4th Semester\Numerical Computing\NC Project\Mid II Submission> python -u "e:\FAST\4th Semester\Numerical Computing\NC Project\Mi
d II Submission\Newton_Backward_Formula_Method.py"
Enter number of data: 4

Enter values of X:
X0 : 1.7
X1 : 1.8
X2 : 1.9
X3 : 2

Enter f(x) values:
Y0 : 0.3979849
Y1 : 0.3399864
Y2 : 0.2818186
Y3 : 0.2238908
          0         1         2         3
0  0.397985  0.000000  0.000000  0.000000
1  0.339986 -0.057998  0.000000  0.000000
2  0.281819 -0.058168 -0.000169  0.000000
3  0.223891 -0.057928  0.000240  0.000409

Enter Value to Interpolate: 1.72

Value at  1.72 is 0.3864183903999998
PS E:\FAST\4th Semester\Numerical Computing\NC Project\Mid II Submission>
```

- # Newton Divided Difference:

Formula:

$$f(x) = f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \ldots + (x - x_0)(x - x_1) \ldots (x - x_k\text{-}1) f[x_0, x_1, \ldots, x_k]$$

Where

$$f[x_0, x_1, \ldots, x_k] = (f[x_1, x_2, \ldots, x_k] - f[x_0, x_1, \ldots, x_{k-1}]) / (x_k - x_0)$$

Code Snippets:

Get Started | Newton_DividedDifference_method.py | Lagrange_Interpolation_Method.py

C: > Users > Dell > Documents > BS(CS)NUCES > BS(CS) Course related material > Numerical_Computing > Lab_2 > Newton_DividedDifference_method.py > printDif

```python

def EnterNumberOfData():
    data = input("Enter Number of Data points: ")
    return int(data)

def Enter_Value():
    val = input("Enter value x to find: ")
    return float(val)

def getData(num):
    for i in range(num):
        val1 = input("Enter x" + str(i) + ": ")
        val2 = input("Enter y" + str(i) + ": ")
        x.append(float(val1))
        y[i][0] = (float(val2))
    return

def product_x(i, value, x):
    prod = 1;
    for j in range(i):
        prod = prod * (value - x[j]);
    return prod;

def dividedDiffTable(x, y, n):

    for i in range(1, n):
        for j in range(n - i):
            y[j][i] = ((y[j][i - 1] - y[j + 1][i - 1]) /
                                     (x[j] - x[i + j]))
    return y

def applyFormula(value, x, y, n):
    sum = y[0][0];
    for i in range(1, n):
        sum = sum + (product_x(i, value, x) * y[0][i]);

    return sum;

def printDiffTable(y, n):
    print("\t DIVIDED DIFFERENCE TABLE")
    print("f(x) \t\t", end="")
    for z in range(1,n):
        print(str(z)+"DD  \t\t\t",end="")

    print("\n")
    for i in range(n):
```

```python
        print("\n")
        for i in range(n):
            for j in range(n - i):
                print(round(y[i][j], 7), "\t\t",
                                            end = " ");

            print(" ");

    n = EnterNumberOfData();
    y = [[0 for i in range(10)]
            for j in range(10)];
    x = [];

    getData(n)
    value = Enter_Value()
    y=dividedDiffTable(x, y, n);

    printDiffTable(y, n);

    print("\nValue at", value, "is",
            round(applyFormula(value, x, y, n), 7))
```

Output:

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

PS C:\Users\Dell> python -u "c:\Users\Dell\Documents\BS(CS)NUCES\BS(CS) Course related material\Numerical_Computing\Lab_2\Newton_
Enter Number of Data points: 5
Enter x0: 1.0
Enter y0: .7651977
Enter x1: 1.3
Enter y1: .6200860
Enter x2: 1.6
Enter y2: .4554022
Enter x3: 1.9
Enter y3: .2818186
Enter x4: 2.2
Enter y4: .1103623
Enter value x to find: 2.8
         DIVIDED DIFFERENCE TABLE
f(x)            1DD                     2DD                     3DD                     4DD

0.7651977               -0.4837057              -0.1087339              0.0658784               0.0018251
0.620086                -0.548946               -0.0494433              0.0680685
0.4554022               -0.578612               0.0118183
0.2818186               -0.571521
0.1103623

Value at 2.8 is -0.180286
PS C:\Users\Dell> python -u "c:\Users\Dell\Documents\BS(CS)NUCES\BS(CS) Course related material\Numerical_Computing\Lab_2\Newton_
```