Tugas Modul II

Interpolasi

Dibuat Oleh :

Nashirudin Baqiy

24060119130045

Asisten Praktikum :

Muhammad Rizqi Arya Pradana

Ibnu Nahwitama

DEPARTEMEN INFORMATIKA

FAKULTAS SAINS DAN MATEMATIKA

UNVERSITAS DIPONEGORO

2020

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Tujuan

Mahasiswa dapat membuat program interpolasi dengan metode polinom Newton dan metode polinom Lagrange.

1.2 Rumusan Permasalahan

1. Gunakan Program Interpolasi Newton untuk menghitung x = 2.5

x	0.0	1.0	2.0	3.0	4.0
f(x)	1.0000	0.5403	-0.4161	-0.9900	-0.6536

2. Gunakan Program Interpolasi Lagrange untuk menghitung x = 323.5

x	321.0	322.8	324.2	325.0
f(x)	2.50651	2.50893	2.51081	2.51188

BAB II
DASAR TEORI

2.1 Pendahuluan Teori

Diketahui pasangan data $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1}), (x_n, y_n)$.Bagaimana mencari y untuk nilai x lain yang dikehendaki? Fungsi kontinu $f(x)$ direpresentasikan n +1 data (Gambar 1). Interpolasi Polynomial meliputi pencarian polynomial derajad n yang melalui n +1 titik. Metode yang sering dipakai adalah metode interpolasi Newton dann metode interpolasi Lagrange.

2.2 Metode Polinom Newton

Diberikan n+1 titik data, $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, jarak titik x dengan selang yang sama, sebagai

$$P_n(x) = b_0 + b_1(x - x_0) + \cdots + b_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Atau dapat ditulis sebagai

$$P_{n(x)} = P(x_0 + hu) = y_0 + u\Delta^1 y_0 + \frac{u(u-1)}{2!}\Delta^2 y_0 + \cdots + \frac{u(u-1)\dots(u-n+1)}{2!}\Delta^n y_0$$

Dengan

$$\frac{x - x_0}{h} = u$$

Algortima Interpolasi Newton ke depan :

1. Baca data masukan pasangan x dan y
2. Berikan nilai x yang dicari y nya.
3. Hitung h
4. Lakukan inisialisasi :
   Sum = y0
   R = (x-x0)/h
5. Lakukan iterasi proses berikut untuk i=1 sampai n-1
   a. Product =1
   b. Untuk j=0 sampai i-1 lakukan perhitungan
   Product = product*(u-j)/(j+1)
   c. Lakukan penjumlahan
   Sum = sum +product * $\Delta^i y_0$
6. Kembalikan nilai sum sebagai hassil perhitungan

2.3 Metode Polinom Lagrange

Diberikan n+1 titik data, $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, jarak titik x dengan selang yang tidak sama. Dengan $f(x_i) = y_i$

Polinom Lagrange diberikan oleh

$$P_{n(x)} = \sum_{i=0}^{n} L\_i(x) f(x_i)$$

Dengan n dalam $f_{n(x)}$ merupakan derajad order n yang mengaproksimasi fungsi $y = f(x)$ diberikan $n + 1$ titik data sebagai. $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1}), (x_n, y_n)$, dan

$$L_{i(x)} = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Algoritma Interpolasi Lagrange :

1. Baca data masukan pasangan x dan y
2. Lakukan inisialisasi :
   Sum = y_0
   R = (x-x_0)/h
3. Lakukan iterasi proses berikut untuk i=1 sampai n-1
   a. Product =y_i
   b. Untuk j=0 sampai i-1 lakukan perhitungan
      Product = Product*(x-x_j)/($x_i - x_j$)
   c. Lakukan penjumlahan
   Sum = sum +product * $\Delta^i y_0$
4. Kembalikan nilai sum sebagai hasil perhitungan

BAB III
PEMBAHASAN

3.1 Penyelesaian dengan Metode Polinom Newton

3.1.1 Source Code

```
# Metode Polinom Newton
# 24060119130045 - Nashirudin Baqiy

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#data pada tabel
x = [0.0, 1.0, 2.0, 3.0, 4.0]
y = [1.0, 0.5403, -0.4161, -0.9900, -0.6536] #y=f(x)
xinput = 2.5

def newton_interpolation(x, y, xi):
    #Panjang/jumlah data point
    n = len(x)
    #inisialisasi selisih terbagi
    fdd = [[None for x in range(n+1)] for x in range(n)]
    #nilai f(x) pada derajat yang berbeda
    yint = [None for x in range(n)]

    #menemukan perbedaan pembagi
    for i in range(n):
        fdd[i][0] = x[i]
    for i in range(n):
        fdd[i][1] = y[i]
    for j in range(1,n):
        for i in range(n-j):
            fdd[i][j+1] = (fdd[i+1][j] - fdd[i][j])/(x[i+j]-x[i])

    fdd_table = pd.DataFrame(fdd,
columns=['x','f(x)','f[1]','f[2]','f[3]','f[4]'])
    print(fdd_table)

    #interpolasi xinput
    xterm = 1
    yint[0] = fdd[0][1]
    for order in range(1, n):
        xterm = xterm * (xi - x[order-1])
        yint2 = yint[order-1] + fdd[0][order+1]*xterm
        yint[order] = yint2

    return yint[order]

a = newton_interpolation(x, y, xinput)

print(
```

```
    )
print('Hasil interpolasi newton untuk x = %f adalah %f.' % (xinput,
a))
```

3.1.2 Hasil Command Prompt

```
(base) F:\TUGAS\Praktikum\Metnum\Tugas2\py>python "polinom newton.py"
     x    f(x)    f[1]     f[2]      f[3]      f[4]
0  0.0  1.0000 -0.4597 -0.24835  0.146533 -0.014642
1  1.0  0.5403 -0.9564  0.19125  0.087967       NaN
2  2.0 -0.4161 -0.5739  0.45515       NaN       NaN
3  3.0 -0.9900  0.3364      NaN       NaN       NaN
4  4.0 -0.6536     NaN      NaN       NaN       NaN

Hasil interpolasi newton untuk x = 2.500000 adalah -0.792086.
```

3.1.3 Excel

i	x	F[ ]	F[,]	F[, ,]	F[, , ,]	F[, , , ,]
0	0	1	-0,4597	-0,24835	0,14653	-0,01464
1	1	0,5403	-0,9564	0,19125	0,08797	
2	2	-0,4161	-0,5739	0,45515		
3	3	-0,99	0,3364			
4	4	-0,6536				

$P_3(x)$ untuk y pada x =		2,5
$P_3(2,5)$ =	-0,792086	

3.1.4 Penjelasan

Program untuk metode polinom newton di atas merupakan penerjemahan dari algoritma yang diberikan Modul Numerik 2020. Pertama mendefinisikan x,y, dan xinput secara hardcode. Kemudian n dapat dicari dari len(x) (*banyaknya x*) dan didapatkan $n = 4$. Untuk penghitungan menyesuaikan algoritma yang diberikan yaitu pertama perulangan(n) menghitung $f[1]$ sampai $f[n]$ yang masing-masing perulangan(n) terdapat $f[j]$ untuk iterasi i menghasilkan (Selanjutnya ditulis $f[i][j]$)

$$(f[i][j-1] - f[i-1][j-1])/(x[i+j] - x[i])$$

seperti yang terlihat pada tabel. Kemudian menjalankan perulangan (1,n) perhitungan

$$xterm = xterm * (xi - x[order-1])$$

dan

$$yint2 = yint[order-1] + fdd[0][order+1] * xterm$$

secara berulang. $xterm$ diinisiasi dengan 1 dan inisiasi $yint = y$. Kemudian kembalikan nilai $yint2$ / $yint[n]$ sebagai nilai polinom lagrange dari x yang dicari.

3.2 Penyelesaian dengan Metode Polinom Lagrange

3.1.1 Source Code

```
# Metode Polinom Lagrange
# 24060119130045 - Nashirudin Baqiy

import numpy as np
import pandas as pd

x = [321, 322.8, 324.2, 325]
y = [2.50651, 2.50893, 2.51081, 2.51188]
xp = 323.5

def lagrange(x, y, xin):
    #Panjang/jumlah data
    n = len(x)
    # Inisialisasi nilai kardinal
    lr = [[0 for x in range (3)] for x in range(n)]
    Sum = 0
    for i in range(n):
        lr[i][0] = x[i]
    for i in range(n):
        lr[i][1] = y[i]

    # Metode Interpolasi Lagrange
    for i in range(n):
        p = 1
        for j in range(n):
            if i != j:
                p = p * (xin-x[j]) / (x[i]-x[j])
        lr[i][2] = p
        Sum = Sum + lr[i][2] * lr[i][1]

    lr_table = pd.DataFrame(lr, columns=['x','f(x)','l(i)'])
    print(lr_table)
    return Sum

a = lagrange(x, y, xp)
print(
    )
print('Hasil interpolasi lagrange untuk x = %f adalah %f.' % (xp, a))
```

3.1.2 Hasil Command Prompt

```
(base) F:\TUGAS\Praktikum\Metnum\Tugas2\py>python "polinom lagrange.py"
       x     f(x)      l(i)
0  321.0  2.50651 -0.031901
1  322.8  2.50893  0.473485
2  324.2  2.51081  0.732422
3  325.0  2.51188 -0.174006

Hasil interpolasi lagrange untuk x = 323.500000 adalah 2.509871.
```

3.1.3 Excel

i	x	f(x)	l(i)
0	321	2,50651	-0,0319
1	322,8	2,50893	0,473485
2	324,2	2,51081	0,732422
3	325	2,51188	-0,17401
f(x) untuk x =			323,5
$P_3(323,5)$ =	2,509871		

3.1.4 Penjelasan

Program untuk metode polinom newton di atas merupakan penerjemahan dari algoritma yang diberikan Modul Numerik 2020. Pertama mendefinisikan x,y, dan xinput secara hardcode. Kemudian n dapat dicari dari len(x) (*banyaknya x*) dan didapatkan $n = 4$. Untuk penghitungan menyesuaikan algoritma yang diberikan yaitu perulangan (sampai n) inisiasi $p = 1$ dan $l(i) = y_i$ juga melakukan perulangan (sampai n) menghitung

$$l(i) = l(i) * (x - x_j)/(x_i - x_j) \text{ jika } i \neq j$$

dan menghitung

$$Sum = Sum + f(i) * l(i)$$

Mengembalikan nilai Sum untuk hasil polinom lagrange dari x yang dicari.

BAB IV
PENUTUPAN

4.1 Kesimpulan

Nilai fungsi dari suatu x dapat dicari dengan Metode Polinom Newton dan Metode Polinom Lagrange. Langkah-langkah yang tersedia dapat dibuat dengan menggunakan Bahasa *Python* dan menggunakan Spreadsheet seperti excel sehingga tidak perlu melakukan iterasi secara sistem manual seperti yang ada pada *proses* di atas. Hasil kerja dari sistem program tersebut adalah sebagai berikut.

1. Pada soal pertama tentang implementasi Metode Polinom Newton, dicari nilai $f(x)$ dengan $x = 2.5$. Hasil yang diperoleh menggunakan Metode Polinom Newton yang telah dibuat dalam Bahasa *Python* dan Excel adalah -0.792086.
2. Pada soal kedua tentang implementasi Metode Polinom Lagrange, dicari nilai $f(x)$ dengan $x = 323.5$. Hasil yang diperoleh menggunakan Metode Polinom Newton yang telah dibuat dalam Bahasa *Python* dan Excel adalah 2.509871.