

Nama : Annisa Charisma Wijayanti
NIM : 211220122140086
Mata Kuliah : Metode Numerik – Kelas D
Jurusan : Teknik Komputer
Link Github : https://github.com/annisacharisma/Implementasi-Interpolarisasi_Annisa-Charisma-Wijayanti_Metode-Numerik

IMPLEMENTASI INTERPOLASI

Diinginkan aplikasi untuk mencari solusi dari problem pengujian yang memperoleh data terbatas (data terlampir) dengan interpolasi masing-masing menggunakan metode:

1. polinom Langrange
2. polinom Newton

Tugas mahasiswa:

1. Mahasiswa membuat kode sumber dengan bahasa pemrograman yang dikuasai untuk mengimplementasikan solusi di atas
2. Sertakan kode testing untuk menguji kode sumber tersebut untuk menyelesaikan problem dalam gambar. Plot grafik hasil interpolasi dengan $5 \leq x \leq 40$
3. Mengunggah kode sumber tersebut ke Github dan setel sebagai publik. Berikan deskripsi yang memadai dari project tersebut
4. Buat dokumen docx dan pdf yang menjelaskan alur kode dari (1), analisis hasil, dan penjabarannya.

- Sebuah pengukuran fisika telah dilakukan untuk menentukan hubungan antara tegangan yang diberikan kepada baja tahan-karat dan waktu yang diperlukan hingga baja tersebut patah. Delapan nilai tegangan yang berbeda dicobakan, dan data yang dihasilkan adalah

Tegangan, x (kg/mm ²)	5	10	15	20	25	30	35	40
Waktu patah, y (jam)	40	30	25	40	18	20	22	15

1. Polinom Langrange

Metode Lagrange disukai karena konsepnya sederhana. Namun, secara komputasi metode ini kurang efisien dibanding metode Newton. Secara umum derajat $\leq n$ untuk $(n+1)$ titik berbeda, persamaan polinom Lagrange dirumuskan:

$$p_n(x) = \sum_{i=0}^n a_i L_i(x) = a_0 L_0(x) + a_1 L_1(x) + \dots + a_n L_n(x)$$

$$\text{dimana } L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{(x - x_j)}{(x_i - x_j)} \text{ dan } a_i = y_i$$

Source Code:

```
import numpy as np
import matplotlib.pyplot as plt

# Lagrange Interpolation
def lagrange_interpolation(x, y, x_val):
    n = len(x)
    result = 0
    for i in range(n):
        term = y[i]
        for j in range(n):
            if j != i:
                term *= (x_val - x[j]) / (x[i] - x[j])
        result += term
    return result

# input data
arr_x = np.array([5, 10, 15, 20, 25, 30, 35, 40])
arr_y = np.array([40, 30, 25, 40, 18, 20, 22, 15])

# The x value for which y needs to be interpolated
x_val = 12

# Compute the interpolated value at x_val
y_val = lagrange_interpolation(arr_x, arr_y, x_val)
print(f"Hasilnya adalah: {y_val}")

# Plotting the results
x_vals = np.linspace(min(arr_x), max(arr_x), 100)
y_vals = [lagrange_interpolation(arr_x, arr_y, x) for x in x_vals]

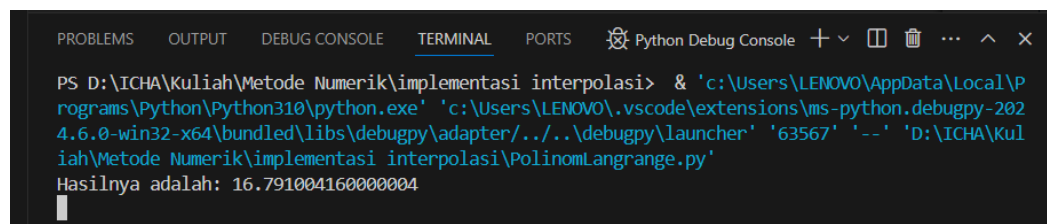
plt.plot(arr_x, arr_y, 'o', label='Data points')
plt.plot(x_vals, y_vals, label='Lagrange interpolation')
plt.axvline(x_val, color='r', linestyle='--',
            label=f'Interpolated x = {x_val}')
plt.axhline(y_val, color='g', linestyle='--',
            label=f'Interpolated y = {y_val:.2f}')
```

```
plt.legend()
plt.xlabel('Tegangan, x (Kg/mm^2)')
plt.ylabel('Waktu patahan, y (jam)')
plt.title('Lagrange Interpolation')
plt.grid(True)
plt.show()
```

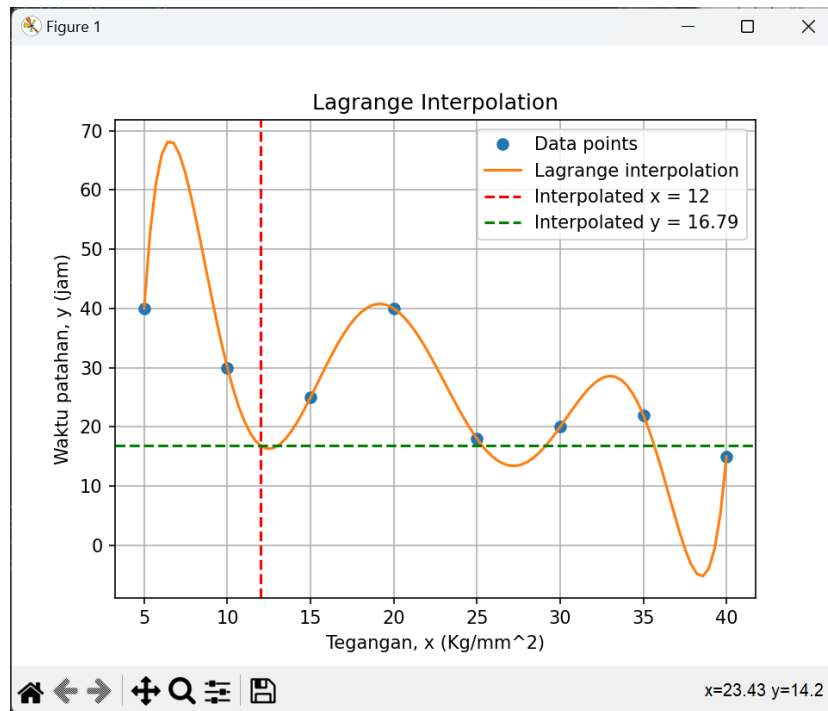
Penjelasan Source Code:

- Fungsi `lagrange_interpolation` menerima tiga parameter, yaitu `x` yang berisi nilai-nilai `x` dari data titik, `y` array dari nilai-nilai `y` yang sesuai, dan `x_val` adalah nilai `x` yang ingin dicari nilai interpolasinya. Fungsi ini menghitung nilai interpolasi dengan membentuk polinom Lagrange. Setiap iterasi dalam fungsi menghitung sebuah polinom basis Lagrange untuk setiap titik data, yang kemudian digunakan untuk membentuk polinom interpolasi akhir.
- Selanjutnya mendefinisikan dua array `arr_x` dan `arr_y` yang berisi data nilai dari input yang diberikan.
- Menentukan nilai `x` pada titik yang ingin dihitung interpolasi nilai `y` nya menggunakan fungsi `x_val`
- Selanjutnya memanggil fungsi `lagrange_interpolation` dengan data titik dan `x_val` untuk menghitung nilai `y` yang diinterpolasi pada `x_val`. Hasil ini disimpan dalam `y_val`.
- Untuk visualisasi, nilai-nilai `x` dalam rentang minimum dan maksimum `arr_x` dibuat dalam array `x_vals` menggunakan `np.linspace`.
- Plot grafik dibuat menggunakan `matplotlib`, yang menampilkan titik-titik data asli, kurva interpolasi Lagrange, serta garis-garis vertikal dan horizontal untuk menunjukkan titik interpolasi `x_val` dan nilai `y` yang diinterpolasi.

Hasil:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console + - [ ] [X] ... ^ X
PS D:\ICHA\Kuliah\Metode Numerik\implementasi interpolasi> & 'c:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\LENOVO\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter\..\debugpy\launcher' '63567' '--' 'D:\ICHA\Kuliah\Metode Numerik\implementasi interpolasi\PolinomLagrange.py'
Hasilnya adalah: 16.791004160000004
```



2. Polinom Newton

Persamaan polinomial pada Newton memudahkan komputasi karena fungsinya bisa disederhanakan dengan cara faktorisasi. Metode ini menggunakan konsep perbedaan terbagi (divided differences) untuk membangun polinom interpolasi secara iteratif. Interpolasi polinom Newton dapat didefinisikan sebagai berikut:

$$P(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Koefisien a_i adalah perbedaan terbagi yang dihitung dengan:

- $a_0 = f[x_0]$
- $a_1 = f[x_1, x_0]$
- \vdots
- $a_n = f[x_n, x_{n-1}, \dots, x_0]$

Perbedaan terbagi ($f[x_n, x_{n-1}, \dots, x_0]$) dihitung sebagai:

- $f[x_0] = y_0$
- $f[x_1, x_0] = \frac{f[x_1] - f[x_0]}{x_1 - x_0}$
- $f[x_2, x_1, x_0] = \frac{f[x_2, x_1] - f[x_1, x_0]}{x_2 - x_0}$
- \vdots

Source Code:

```
import numpy as np
import matplotlib.pyplot as plt

# Newton Interpolation
def newton_interpolation(x, y, x_val):
    n = len(y)
    # Create a divided difference table
    coef = np.zeros([n, n])
    coef[:, 0] = y

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

    # Evaluate the interpolation polynomial at x_val
    result = coef[0, 0]
    product_term = 1
    for i in range(1, n):
        product_term *= (x_val - x[i - 1])
        result += coef[0, i] * product_term

    return result

# input data
arr_x = np.array([5, 10, 15, 20, 25, 30, 35, 40])
arr_y = np.array([40, 30, 25, 40, 18, 20, 22, 15])

# The x value for which y needs to be interpolated
x_val = 12

# Compute the interpolated value at x_val
y_val = newton_interpolation(arr_x, arr_y, x_val)
print(f"Hasilnya adalah: {y_val}")

# Plotting the results
x_vals = np.linspace(min(arr_x), max(arr_x), 100)
y_vals = [newton_interpolation(arr_x, arr_y, x) for x in x_vals]

plt.plot(arr_x, arr_y, 'o', label='Data points')
plt.plot(x_vals, y_vals, label='Newton interpolation')
plt.axvline(x_val, color='r', linestyle='--', label=f'Interpolated
x = {x_val}')
plt.axhline(y_val, color='g', linestyle='--', label=f'Interpolated
y = {y_val:.2f}')
plt.legend()
plt.xlabel('Tegangan, x (Kg/mm^2)')
plt.ylabel('Waktu patahan, y (jam)')
plt.title('Newton Interpolation')
plt.grid(True)
plt.show()
```

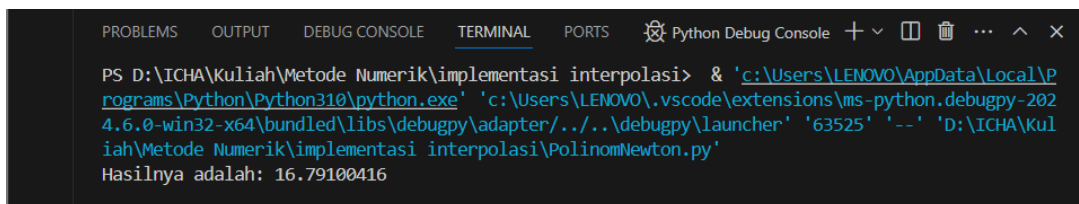
Penjelasan Source Code:

- Fungsi `newton_interpolation` digunakan untuk menghitung nilai interpolasi menggunakan metode Newton. Fungsi tersebut menerima tiga parameter, yaitu `x` : array

yang berisi titik-titik x (independen), y array yang berisi titik-titik y (dependen), dan x_val adalah nilai x yang ingin diinterpolasi.

- Inisialisasi ' n ' adalah panjang array y , dan $coef$ adalah tabel perbedaan terbagi (divided difference table) yang diinisialisasi sebagai matriks nol dengan ukuran $n \times n$.
- Selanjutnya, tabel perbedaan terbagi dihitung melalui loop ganda yang mengisi tabel $coef$ dengan perbedaan terbagi.
- Setelah itu, nilai interpolasi di x_val dihitung dengan cara menginisialisasi variabel $result$ dengan nilai $coef[0, 0]$ dan kemudian mengalikan produk dari $(x_val - x[i - 1])$ serta menambahkan kontribusi dari setiap koefisien ke dalam $result$. Fungsi ini kemudian mengembalikan nilai interpolasi di x_val .
- Lalu mendefinisikan dua array arr_x dan arr_y yang berisi data nilai dari input yang diberikan
- Menentukan nilai x pada titik yang ingin dihitung interpolasi nilai y nya menggunakan fungsi x_val
- Menggunakan fungsi `newton_interpolation` untuk menghitung nilai interpolasi di x_val dan mencetak hasilnya.
- Untuk memvisualisasikan hasil, rentang nilai x dibuat dari nilai minimum hingga maksimum arr_x menggunakan `np.linspace`. Nilai y untuk rentang x tersebut dihitung dengan menerapkan fungsi `newton_interpolation` pada setiap nilai dalam x_vals . Plot titik-titik data asli dan hasil interpolasi dibuat menggunakan `plt.plot`. Garis vertikal pada x_val dan garis horizontal pada y_val ditambahkan untuk menandai nilai interpolasi.

Hasil:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console + - [ ] [x] ... ^ x
PS D:\ICHA\Kuliah\Metode Numerik\implementasi interpolasi> & 'c:\Users\LENOVO\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\LENOVO\.vscode\extensions\ms-python.debugpy-2024.6.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '63525' '--' 'D:\ICHA\Kuliah\Metode Numerik\implementasi interpolasi\PolinomNewton.py'
Hasilnya adalah: 16.79100416
```

