Rabu, 29 Mei 2024

Nama : Kadek Wisnu Parijata Putra

NIM : 21120122140036

Prodi : Teknik Komputer / 2022

Mata Kuliah : Metode Numerik / D

Github : https://github.com/wisnuprjt/Implementasi-Interpolasi_Kadek-Wisnu_Metode-Numerik.git

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

a. Polinom Lagrange
b. 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 <= x <= 40

Sebuah pengukuran fisika telah dilakukan untuk menentukan hubungan antara tegangan yang diberikan kepada baja tahan-karat dan waktu yang diperlukan hingga baja tersebut patah. delapa 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

Jawaban :

a. Interpolasi Lagrange adalah Teknik Metode Numerik yang digunakan untuk menentukan Polinom/Interpolasi yang sesuai dengan serangkaian titik data. Secara umum, metode interpolasi Lagrange membangun polinom interpolasi dalam bentuk rumus sebagai berikut:

$$L(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

Code Testing **Interpolasi Lagrange menggunakan bahasa pemrograman Phyton** *:*

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

def lagrange_interpolation(x, y, x_values):
    def L(k, x_point):
        L_k = 1
        for i in range(len(x)):
            if i != k:
                L_k *= (x_point - x[i]) / (x[k] - x[i])
        return L_k

    def P(x_point):
        result = 0
        for k in range(len(y)):
            result += y[k] * L(k, x_point)
        return result

    return [P(xi) for xi in x_values]

# Data dari gambar
x = [5, 10, 15, 20, 25, 30, 35, 40]
y = [40, 30, 25, 40, 18, 20, 22, 15]

# Titik-titik untuk menguji interpolasi
x_test = np.linspace(5, 40, 500)
y_test = lagrange_interpolation(x, y, x_test)

# Titik-titik interpolasi untuk ditampilkan pada grafik (pilih beberapa
secara manual)
x_interpolasi = [7, 12, 17, 22, 27, 32, 37]  # Titik-titik yang dipilih
secara manual
y_interpolasi = lagrange_interpolation(x, y, x_interpolasi)

# Plot hasil interpolasi
plt.figure(figsize=(10, 6))
plt.plot(x_test, y_test, label='Polinomial Lagrange')
plt.scatter(x, y, color='red', label='Titik yang diketahui')
plt.scatter(x_interpolasi,  y_interpolasi,  color='green',  label='Titik
interpolasi', zorder=5)
```

```
plt.title('Interpolasi Polinomial Lagrange')
plt.xlabel('Tegangan, x (kg/mm²)')
plt.ylabel('Waktu patah, y (jam)')
plt.legend()
plt.grid(True)
plt.show()
```

Langkah-langkah :

1. **Import Library**

   Kode ini mengimport dua library yaitu numpy untuk operasi bilangan numerik dan matplotlib.pyplot untuk pembuatan grafik dan visualisasi data.

2. **Definisi Fungsi Lagrange Interpolation**

   Fungsi ‘lagrange_interpolation’ bertujuan untuk melakukan interpolasi Lagrange. fungsi ini meliputi ‘x’ daftar nilai-nilai dari titik data, ‘y’ daftar nilai-nilai yang berkorespondensi dengan nilai nilai, dan ‘x_values’ daftar nilai nilai x Dimana polynomial interpolasi akan dilakukan evaluasi.

3. **Menghitung Nilai Polinomial**

   Bagian ini mendefinisikan fungsi `P(x_point)` untuk menghitung nilai polinomial Lagrange pada titik `x_point`:

   - `result` diinisialisasi dengan nilai 0.

   - Loop dengan indeks `k` berjalan dari 0 hingga \(n-1 \) (dimana \(n \) adalah panjang daftar `y`) untuk menghitung dan menjumlahkan kontribusi setiap basis polinomial \(L_k(x) \) yang dikalikan dengan nilai \(y[k] \).

4. **Mengembalikan Nilai Polinomial**

   Fungsi `lagrange_interpolation` digunakan mengembalikan daftar nilai polinomial yang dihitung untuk setiap nilai dalam `x_values` menggunakan list comprehension.

5. **Data dan Titik Uji**

   x = [5, 10, 15, 20, 25, 30, 35, 40]

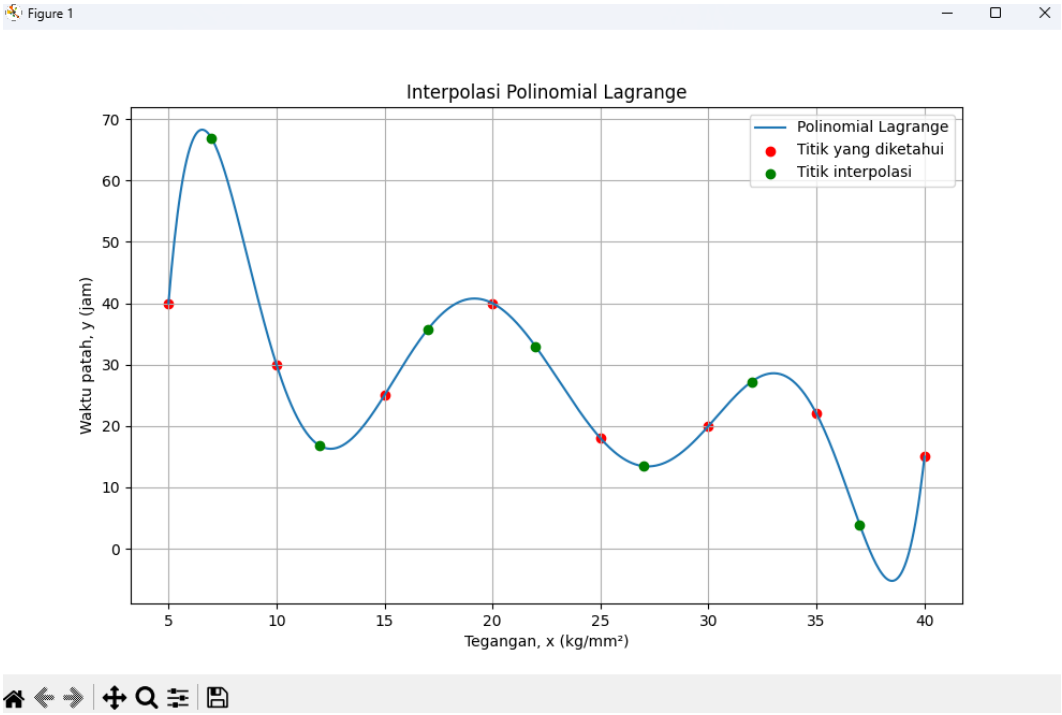
   y = [40, 30, 25, 40, 18, 20, 22, 15]

   x_test = np.linspace(5, 40, 500)

   y_test = lagrange_interpolation(x, y, x_test)

6. Plot hasil Interpolasi

Bagian ini membuat grafik hasil interpolasi menggunakan `matplotlib` :



Gambar 1.1 *Output* Grafik Polinomial Lagrange

7. Output

Output dari kode ini adalah grafik yang menunjukkan:

- **Kurva Polinomial Lagrange (Garis Biru)** yang sesuai dengan titik data yang diketahui. Interpolasi Lagrange memberikan polinom yang melewati semua titik data. karena itu, kurva ini tepat melalui setiap titik merah, yang menghasilkan kurva kontinu untuk berosilasi di antara titik-titik data tersebut

- **Titik data Asli (Titik Merah)** Titik merah pada grafik menunjukkan titik-titik data asli yang digunakan untuk interpolasi. Analisis kasus ini, ada 8 titik data asli yang tersebar antara x=5 dan x=40 Titik-titik ini menjadi dasar bagi polinomial interpolasi yang dihasilkan oleh metode Lagrange.

- **Titik interpolasi (Titik Hijau)** Titik Hijau pada grafik menunjukkan hasil interpolasi. Berdasarkan hasil yang dihitung. Ini menggambarkan nilai prediksi dari polinom interpolasi pada titik yang tidak ada dalam data asli.

b. Interpolasi Newton adalah Teknik yang efisien digunakan untuk membangun polinomial yang paling sesuai dengan serangkaian titik data. Metode ini menggunakan *divided differences* untuk menghitung koefisien polinomial secara rekursif, sehingga memungkinkan pembaruan yang mudah pada polinomial tanpa harus menghitung ulang seluruhnya. Polinom Interpolasi Newton dirumuskan sebagai berikut:

$$P(x) = f[x_0] + fx_0, x_1 + fx_0, x_1, x_2(x - x_1) + \ \ldots \ + fx_0, x_1, \ \ldots, x_n(x - x_1) \ldots (x - x_{n-1})$$

Code Testing **Interpolasi Newton menggunakan bahasa pemrograman Phyton** *:*

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

def newton_interpolation(x, y, x_values):
    # Menghitung divided differences
    n = len(x)
    divided_diff = np.zeros((n, n))
    divided_diff[:, 0] = y

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

    # Menghitung nilai polinomial di titik-titik x_values
    def P(x_point):
        result = divided_diff[0, 0]
        term = 1.0
        for i in range(1, n):
            term *= (x_point - x[i - 1])
            result += divided_diff[0, i] * term
        return result

    return [P(xi) for xi in x_values]

# Data dari gambar
x = [5, 10, 15, 20, 25, 30, 35, 40]
y = [40, 30, 25, 40, 18, 20, 22, 15]
```

```
# Titik-titik untuk menguji interpolasi
x_test = np.linspace(5, 40, 500)
y_test = newton_interpolation(x, y, x_test)

# Titik-titik interpolasi untuk ditampilkan pada grafik (pilih beberapa
secara manual)
x_interpolasi = [7, 12, 17, 22, 27, 32, 37]
y_interpolasi = newton_interpolation(x, y, x_interpolasi)

# Plot hasil interpolasi
plt.figure(figsize=(10, 6))
plt.plot(x_test, y_test, label='Polinomial Newton')
plt.scatter(x, y, color='red', label='Titik yang diketahui')
plt.scatter(x_interpolasi, y_interpolasi, color='green', label='Titik
interpolasi', zorder=5)
plt.title('Interpolasi Polinomial Newton')
plt.xlabel('Tegangan, x (kg/mm²)')
plt.ylabel('Waktu patah, y (jam)')
plt.legend()
plt.grid(True)
plt.show()
```

Langkah-langkah :

1. **Import Library**

   Kode ini mengimport dua library yaitu numpy untuk operasi bilangan numerik dan matplotlib.pyplot untuk pembuatan grafik dan visualisasi data.

2. **Definisi Fungsi Newton Interpolation**

   Fungsi ‘newton_interpolation’ bertujuan untuk melakukan interpolasi Newton. fungsi ini meliputi ‘x’ daftar nilai-nilai dari titik data, ‘y’ daftar nilai-nilai yang berkorespondensi dengan nilai nilai, dan ‘x_values’ daftar nilai nilai x Dimana polynomial interpolasi akan dilakukan evaluasi.

3. **Menghitung Nilai Polinomial**

   Bagian ini mendefinisikan fungsi `P(x_point)` untuk menghitung nilai polinomial Newton pada titik `x_point`:

   - `result` diinisialisasi dengan nilai 0.

   - Loop dengan indeks `k` berjalan dari 0 hingga $n-1$ (dimana n adalah panjang daftar `y`) untuk menghitung dan menjumlahkan kontribusi setiap basis polinomial $L_k(x)$ yang dikalikan dengan nilai $y[k]$.

4. **Data dan Titik Uji**

   x = [5, 10, 15, 20, 25, 30, 35, 40]

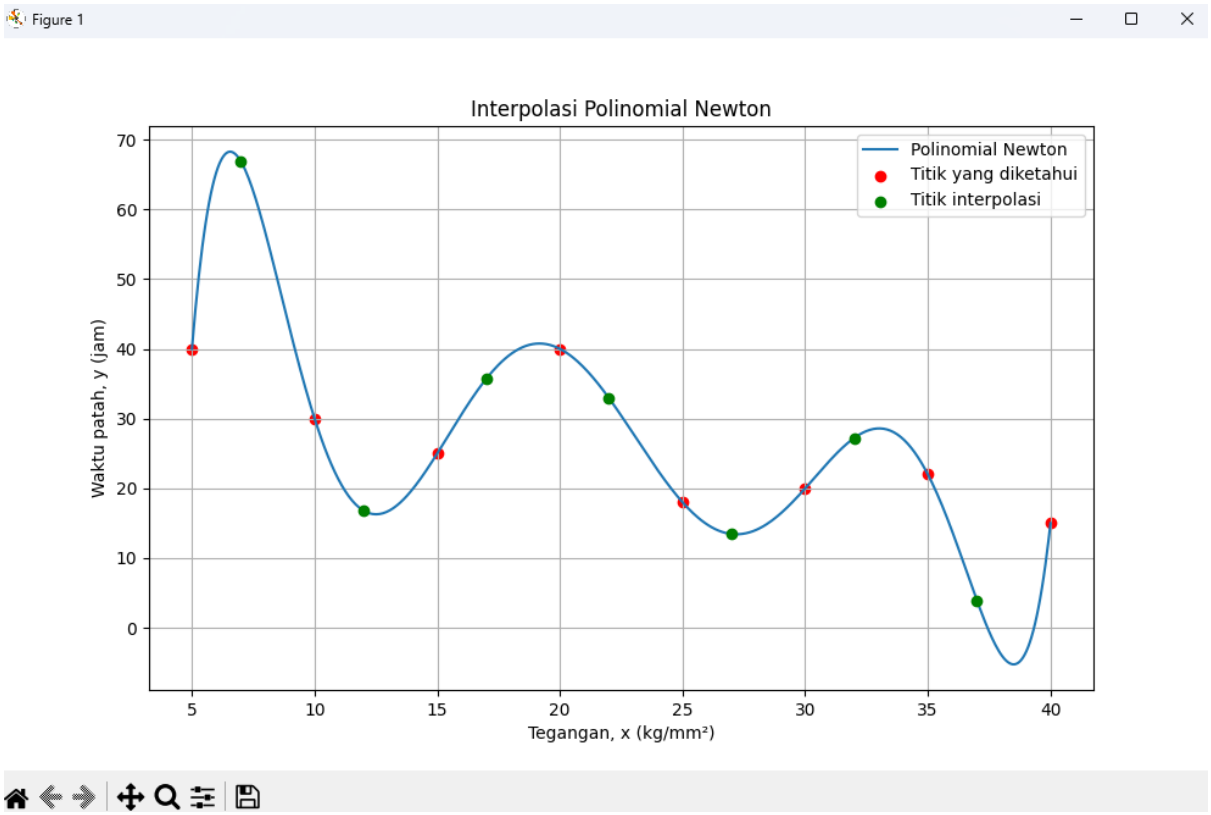
   y = [40, 30, 25, 40, 18, 20, 22, 15]

   x_test = np.linspace(5, 40, 500)

   y_test = newton_interpolation(x, y, x_test)

5. **Plot hasil Interpolasi**

   Bagian ini membuat grafik hasil interpolasi menggunakan `matplotlib` :



Gambar 2.1 *Output* Grafik Polinomial Newton

6. **Output**

   Output dari kode ini adalah grafik yang menunjukkan:

   - **Kurva Polinomial Newton (Garis Biru)** yang sesuai dengan titik data yang diketahui. Interpolasi Newton memberikan polinom yang melewati semua titik data. karena itu, kurva ini tepat melalui setiap titik merah, yang menghasilkan kurva kontinu untuk berosilasi di antara titik-titik data tersebut

- **Titik data Asli (Titik Merah)** Titik merah pada grafik menunjukkan titik-titik data asli yang digunakan untuk interpolasi. Analisis kasus ini, ada 8 titik data asli yang tersebar antara x=5 dan x=40 Titik-titik ini menjadi dasar bagi polinomial interpolasi yang dihasilkan oleh metode Newton.
- **Titik interpolasi (Titik Hijau)** Titik Hijau pada grafik menunjukkan hasil interpolasi. Berdasarkan hasil yang dihitung. Ini menggambarkan nilai prediksi dari polinom interpolasi pada titik yang tidak ada dalam data asli.