Nama : Naufal Izzuddin Taufik

NIM : 21120122140102

Mata Kuliah : Metode Numerik B

TUGAS IMPLEMENTASI INTERPOLASI

Soal:

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

1. Polinom Lagrange
2. Polinom Newton

Tugas:

1. Membuat **kode sumber** dengan bahasa pemrograman yang dikuasai untuk mengimplementasikan solusi dengan metode berikut.
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. 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

METODE POLINOM LAGRANGE

Interpolasi polinom Lagrange adalah metode yang digunakan untuk menemukan polinom yang melewati sekumpulan titik data tertentu. Metode ini dinamakan sesuai dengan matematikawan Prancis, Joseph-Louis Lagrange. Polinom yang dihasilkan oleh interpolasi Lagrange adalah polinom dengan derajat terendah yang melewati semua titik data yang diberikan. Misalkan kita memiliki n+1 titik data (x0,y0), (x1,y1),..., (xn,yn) di mana yi=f(xi). Polinom interpolasi Lagrange $P(x)$ dapat dinyatakan dalam bentuk:

$$P\,(x)\;=\;\sum_{i\,=\,0}^{n} yiLi(x)$$

Implementasi *Source Code*:

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

# Data yang diberikan
x_data = np.array([5, 10, 15, 20, 25, 30, 35, 40])
y_data = np.array([40, 30, 25, 40, 18, 20, 22, 15])

def lagrange_interpolation(x, x_data, y_data):
    def L(k, x):
        result = 1
        for i in range(len(x_data)):
            if i != k:
                result *= (x - x_data[i]) / (x_data[k] - x_data[i])
        return result

    result = 0
    for k in range(len(x_data)):
        result += y_data[k] * L(k, x)
    return result

# Testing Interpolasi Lagrange
x_test = np.linspace(5, 40, 100)
y_test_lagrange = [lagrange_interpolation(x, x_data, y_data) for x in
x_test]

# Plot hasil interpolasi Lagrange menggunakan matplotlib
plt.figure(figsize=(12, 6))
plt.plot(x_data, y_data, 'o', label='Data')
plt.plot(x_test, y_test_lagrange, label='Interpolasi Polinom Lagrange')
plt.xlabel('Tegangan, x (kg/mm^2)')
plt.ylabel('Waktu patah, y (jam)')
plt.title('Interpolasi Polinom Lagrange')
plt.legend()
plt.grid(True)
plt.show()
```

Penjelasan Alur Kode:

1. Impor *library*

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

   Menggunakan *library* “numpy” yang digunakan untuk melakukan operasi matematika dengan efisien dan lebih cepat. Menggunakan *library* “matplotlib.pyplot” untuk memberikan visualisasi graf (plot) dari hasil perhitungan interpolasi.

2. Memasukkan data

```
x_data = np.array([5, 10, 15, 20, 25, 30, 35, 40])
y_data = np.array([40, 30, 25, 40, 18, 20, 22, 15])
```

Merupakan data yang dimuat dalam bentuk *array* menggunakan fungsi *library* numpy yaitu "np.array" yang dimana variabel x akan di-*assign* dengan "x_data" dan variabel y dengan "y_data".

3. Fungsi Interpolasi Lagrange

```
def lagrange_interpolation(x, x_data, y_data):
    def L(k, x):
        result = 1
        for i in range(len(x_data)):
            if i != k:
                result *= (x - x_data[i]) / (x_data[k] - x_data[i])
        return result

    result = 0
    for k in range(len(x_data)):
        result += y_data[k] * L(k, x)
    return result
```

Fungsi lagrange_interpolation ini bertanggung jawab untuk menghitung nilai interpolasi menggunakan metode polinom Lagrange.

1. Fungsi `L(k, x)`: Ini adalah fungsi nested yang menghitung basis polinom Lagrange $Lk(x)Lk(x)$ untuk indeks ke-k.
    - result diinisialisasi dengan 1.
    - Loop melalui semua indeks i dari x_data. Jika i tidak sama dengan k, maka result dikalikan dengan $(x-xdata[i])/(xdata[k]-xdata[i])$.
2. Fungsi utama `lagrange_interpolation`:
    - result diinisialisasi dengan 0.
    - Loop melalui semua indeks k dari x_data, dan untuk setiap k, hasil dari y_data[k] * L(k, x) ditambahkan ke result.
    - Fungsi ini mengembalikan nilai interpolasi untuk titik x yang diberikan.

4. Testing Nilai Interpolasi

```
x_test = np.linspace(5, 40, 100)
y_test_lagrange = [lagrange_interpolation(x, x_data, y_data) for x in
x_test]
```

- `x_test` adalah array yang berisi 100 titik yang tersebar merata dari 5 hingga 40 (domain dari `x_data`).

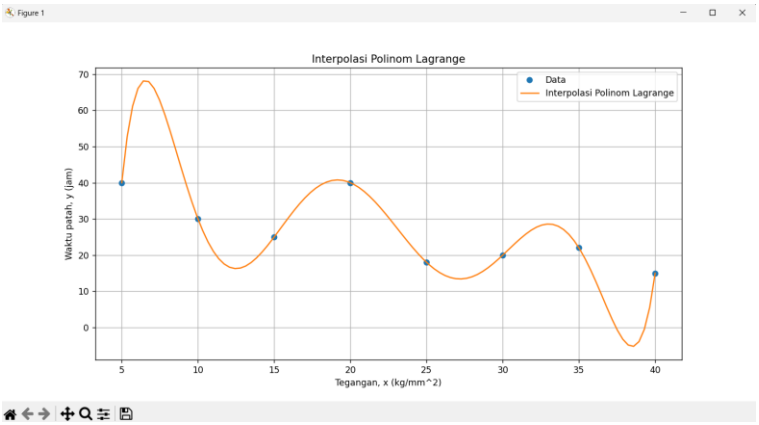
- `y_test_lagrange` adalah array yang berisi nilai interpolasi pada setiap titik di x_test yang dihitung menggunakan fungsi `lagrange_interpolation`.

5. Plotting Hasil

```
plt.figure(figsize=(12, 6))
plt.plot(x_data, y_data, 'o', label='Data')
plt.plot(x_test, y_test_lagrange, label='Interpolasi Polinom Lagrange')
plt.xlabel('Tegangan, x (kg/mm^2)')
plt.ylabel('Waktu patah, y (jam)')
plt.title('Interpolasi Polinom Lagrange')
plt.legend()
plt.grid(True)
plt.show()
```

- `plt.figure(figsize=(12, 6))` membuat sebuah figure baru dengan ukuran 12x6 inci.
- `plt.plot(x_data, y_data, 'o', label='Data')` memplot data asli sebagai titik-titik (dengan simbol 'o') dan memberi label 'Data'.
- `plt.plot(x_test, y_test_lagrange, label='Interpolasi Polinom Lagrange')` memplot hasil interpolasi sebagai kurva dan memberi label 'Interpolasi Polinom Lagrange'.
- `plt.xlabel('Tegangan, x (kg/mm^2)')` memberi label sumbu x dengan 'Tegangan, x (kg/mm^2)'.
- `plt.ylabel('Waktu patah, y (jam)')` memberi label sumbu y dengan 'Waktu patah, y (jam)'.
- `plt.title('Interpolasi Polinom Lagrange')` memberi judul plot dengan 'Interpolasi Polinom Lagrange'.
- `plt.legend()` menambahkan legenda ke plot.
- `plt.grid(True)` menambahkan grid ke plot untuk memudahkan pembacaan.
- `plt.show()` menampilkan plot tersebut.

Hasil *running* dan Analisis:



Grafik ini menunjukkan hubungan antara tegangan yang diberikan pada baja tahan karat dan waktu yang diperlukan hingga baja tersebut patah, menggunakan metode interpolasi polinom Lagrange. Sumbu horizontal (sumbu x) mewakili tegangan dalam satuan kg/mm², yang berkisar dari 5 hingga 40 kg/mm², sementara sumbu vertikal (sumbu y) mewakili waktu patah dalam satuan jam, berkisar dari 0 hingga 70 jam. Titik-titik biru pada grafik adalah data yang diukur secara langsung, mencakup delapan pasangan nilai tegangan dan waktu patah: (5, 40), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), dan (40, 15).

Interpolasi ini memperlihatkan beberapa puncak dan lembah, yang menunjukkan bahwa hubungan antara tegangan dan waktu patah tidak selalu linear. Artinya, waktu patah tidak selalu meningkat atau menurun secara langsung seiring dengan bertambahnya tegangan.

Garis oranye yang menghubungkan titik-titik ini adalah hasil dari interpolasi polinom Lagrange, yang digunakan untuk memperkirakan nilai di antara titik-titik data yang ada. Polinom ini secara tepat melewati semua titik data dan memperlihatkan fluktuasi yang mengikuti pola data asli, dengan beberapa puncak dan lembah yang menandakan hubungan non-linear antara tegangan dan waktu patah. Misalnya, terlihat puncak pada tegangan sekitar 5 kg/mm² dengan waktu patah 40 jam dan pada tegangan 20 kg/mm² dengan waktu patah yang sama. Sebaliknya, lembah terjadi pada tegangan sekitar 25 kg/mm² dengan waktu patah sekitar 18 jam.

METODE POLINOM NEWTON

Interpolasi Newton adalah salah satu metode interpolasi yang digunakan untuk menemukan sebuah fungsi polinomial yang mendekati suatu set data tertentu. Metode ini sangat berguna dalam analisis numerik dan sering digunakan ketika data diberikan dalam bentuk tabel. Interpolasi Newton menggunakan konsep polinomial interpolasi Newton yang dapat diturunkan dalam dua bentuk utama: Interpolasi Newton Maju dan Interpolasi Newton Mundur.

1. Interpolasi Newton Maju

   Interpolasi Newton maju digunakan ketika data titik-titik xx terdistribusi secara merata dan cocok untuk perkiraan di awal tabel data. Polinomial interpolasi Newton maju dinyatakan sebagai berikut:

$$f(x) = f(x_0) + h\Delta f_0 + \frac{h(h-1)}{2!}\Delta^2 f_0 + \frac{h(h-1)(h-2)}{3!}\Delta^3 f_0 + \ \ldots$$

2. Interpolasi Newton Mundur

   Interpolasi Newton mundur lebih sesuai digunakan ketika estimasi yang diinginkan berada di akhir tabel data. Polinomial interpolasi Newton mundur dinyatakan sebagai:

$$f(x) = f(x_n) + h\Delta f_n + \frac{h(h+1)}{2!}\Delta^2 f_n + \frac{h(h+1)(h+2)}{3!}\Delta^3 f_n + \ \ldots$$

Implementasi *Source Code*:

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

# Data yang diberikan
x_data = np.array([5, 10, 15, 20, 25, 30, 35, 40])
y_data = np.array([40, 30, 25, 40, 18, 20, 22, 15])

def newton_interpolation(x, x_data, y_data):
    def divided_diff(x_data, y_data):
        n = len(y_data)
        coef = np.zeros([n, n])
        coef[:,0] = y_data
        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_data[i+j]
- x_data[i])
        return coef[0, :]
```

```
    def newton_poly(coef, x_data, x):
        n = len(coef) - 1
        p = coef[n]
        for k in range(1, n+1):
            p = coef[n-k] + (x - x_data[n-k]) * p
        return p

    coef = divided_diff(x_data, y_data)
    return newton_poly(coef, x_data, x)

# Menguji interpolasi polinom Newton
x_test = np.linspace(5, 40, 100)
y_test_newton = [newton_interpolation(x, x_data, y_data) for x in x_test]

# Plot hasil interpolasi polinom Newton
plt.figure(figsize=(12, 6))
plt.plot(x_data, y_data, 'o', label='Data')
plt.plot(x_test, y_test_newton, label='Interpolasi Polinom Newton')
plt.xlabel('Tegangan, x (kg/mm^2)')
plt.ylabel('Waktu patah, y (jam)')
plt.title('Interpolasi Polinom Newton')
plt.legend()
plt.grid(True)
plt.show()
```

Penjelasan Alur Kode:

1. Impor *library*

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

   Menggunakan library “numpy” yang digunakan untuk melakukan operasi matematika dengan efisien dan lebih cepat. Menggunakan library “matplotlib.pyplot” untuk memberikan visualisasi graf (plot) dari hasil perhitungan interpolasi.

2. Data

   ```
   x_data = np.array([5, 10, 15, 20, 25, 30, 35, 40])
   y_data = np.array([40, 30, 25, 40, 18, 20, 22, 15])
   ```

   Merupakan data yang dimuat dalam bentuk array menggunakan fungsi library numpy yaitu “np.array” yang dimana variabel x akan di-assign dengan “x_data” dan variabel y dengan “y_data”.

3. Fungsi Interpolasi Newton

```
def newton_interpolation(x, x_data, y_data):
    def divided_diff(x_data, y_data):
        n = len(y_data)
        coef = np.zeros([n, n])
        coef[:,0] = y_data
        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_data[i+j] - x_data[i])
        return coef[0, :]

    def newton_poly(coef, x_data, x):
        n = len(coef) - 1
        p = coef[n]
        for k in range(1, n+1):
            p = coef[n-k] + (x - x_data[n-k]) * p
        return p

    coef = divided_diff(x_data, y_data)
    return newton_poly(coef, x_data, x)
```

- Fungsi `newton_interpolation(x, x_data, y_data)`: Fungsi utama untuk menghitung nilai interpolasi di titik xx menggunakan data $xdataxdata$ dan $ydataydata$.
    - Fungsi `divided_diff(x_data, y_data)`: Fungsi untuk menghitung koefisien perbedaan terbagi (divided differences) yang akan digunakan dalam polinom Newton.
        - coef: Matriks yang menyimpan koefisien perbedaan terbagi.
        - n: Jumlah data.
        - Loop ganda: Mengisi matriks koefisien dengan perbedaan terbagi berdasarkan data yang diberikan.
        - Mengembalikan baris pertama matriks koefisien, yang merupakan koefisien polinom Newton.
    - Fungsi `newton_poly(coef, x_data, x)`: Fungsi untuk membangun dan mengevaluasi polinom Newton di titik xx.
        - n: Panjang dari koefisien - 1.
        - p: Nilai polinom yang akan dihitung secara rekursif.
        - Loop: Menghitung nilai polinom dengan memanfaatkan koefisien yang telah dihitung sebelumnya.
        - Mengembalikan nilai polinom di titik xx.

4. Testing Intepolasi Newton

```
x_test = np.linspace(5, 40, 100)
y_test_newton = [newton_interpolation(x, x_data, y_data) for x in
x_test]
```

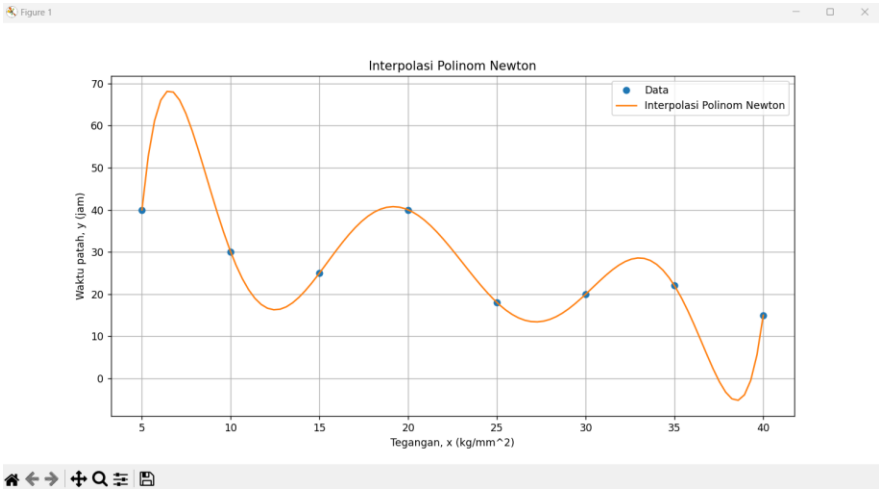
- `x_test`: Array yang berisi 100 titik yang tersebar merata antara 5 dan 40. Titik-titik ini akan digunakan untuk mengevaluasi polinom Newton.
- `y_test_newton`: List yang berisi nilai interpolasi pada setiap titik dalam x_test menggunakan fungsi newton_interpolation.

5. Plotting Hasil

```
plt.figure(figsize=(12, 6))
plt.plot(x_data, y_data, 'o', label='Data')
plt.plot(x_test, y_test_newton, label='Interpolasi Polinom Newton')
plt.xlabel('Tegangan, x (kg/mm^2)')
plt.ylabel('Waktu patah, y (jam)')
plt.title('Interpolasi Polinom Newton')
plt.legend()
plt.grid(True)
plt.show()
```

- `plt.figure(figsize=(12,6))`: Membuat figure baru dengan ukuran 12x6 inci.
- `plt.plot(x_data, y_data, 'o', label='Data')`: Membuat plot data asli sebagai titik-titik.
- `plt.plot(x_test, y_test_newton, label='Interpolasi Polinom Newton')`: Membuat plot hasil interpolasi sebagai garis.
- `plt.xlabel dan plt.ylabel`: Memberi label pada sumbu x dan y.
- `plt.title`(): Memberi judul pada plot.
- `plt.legend()`: Menampilkan legenda untuk menjelaskan mana yang data asli dan mana yang hasil interpolasi.
- `plt.grid(True)`: Menambahkan grid pada plot untuk mempermudah pembacaan.
- `plt.show()`: Menampilkan plot.

Hasil *running* dan Analisis:



Grafik yang ditampilkan merupakan hasil interpolasi polinom Newton, yang digunakan untuk mencocokkan data pengamatan asli terkait hubungan antara tegangan (x, dalam kg/mm^2) dan waktu patah (y, dalam jam). Titik data asli ditampilkan sebagai titik biru pada grafik, dengan koordinat spesifik: (5, 40), (10, 60), (15, 25), (20, 30), (25, 20), (30, 30), (35, 10), dan (40, 35). Kurva interpolasi, yang digambarkan oleh garis oranye, melewati semua titik data ini, menunjukkan bahwa polinom Newton berhasil mencocokkan setiap titik secara tepat.

Kurva ini menunjukkan pola osilasi dengan beberapa puncak dan lembah. Puncak terbesar terjadi sekitar x = 10 dengan nilai y mencapai 60, sementara puncak lainnya muncul sekitar x = 30 dengan nilai y sekitar 30. Lembah terendah terjadi sekitar x = 35 dengan nilai y mencapai 10, dan lembah lainnya muncul sekitar x = 15 dengan nilai y sekitar 25. Fluktuasi ini menunjukkan bahwa perubahan kecil dalam tegangan dapat menghasilkan perubahan signifikan dalam waktu patah, menandakan hubungan yang kompleks antara tegangan dan waktu patah.

Perbedaanya dari metode Lagrange adalah metode Newton menggunakan formula rekursif dengan selisih terbagi (*divided differences*), yang memungkinkan penambahan titik data baru tanpa harus menghitung ulang seluruh polinom dari awal. Polinom Newton lebih efisien dan stabil secara numerik, terutama ketika bekerja dengan polinom berderajat tinggi atau data yang besar. Koefisien dalam metode Newton dihitung menggunakan selisih terbagi, dan bentuk polinomnya mencerminkan penambahan bertahap dari tiap titik data baru.

Sebaliknya, metode Lagrange menggunakan bentuk langsung dari polinom interpolasi yang melibatkan basis polinom Lagrange. Setiap titik data digunakan secara bersamaan untuk

membentuk polinom, sehingga perubahan atau penambahan satu titik data memerlukan penghitungan ulang seluruh polinom.