Nama : Gina Nabila

NIM : 21120122130055

Mata Kuliah : Metode Numerik – Kelas D

GitHub :https://github.com/gin-na/Implementasi-Interpolasi_Metode-Numerik_Gina-Nabila

Pertemuan 10 – Implementasi Interpolasi

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 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
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

Jawaban:

1. Polinom Lagrange

Metode polinom interpolasi yang banyak digunakan dalam komputasi numerik adalah polinom Lagrange, Newton, dan Newton-Gregory. Bentuk umum polinom Lagrange derajat $\leq n$ untuk $(n+1)$ titik berbeda adalah:

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

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} = \frac{(x-x_0)(x-x_1)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)(x_i-x_1)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}$$

Source Code:

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

# Fungsi untuk melakukan interpolasi lagrange
def lagrange_interpolation(x_points, y_points, x):
n = len(x_points)
result = 0.0
for i in range(n):
term = y_points[i]
for j in range(n):
if i != j:
term *= (x - x_points[j]) / (x_points[i] - x_points[j])
result += term
return result

# Fungsi untuk memasukkan jumlah titik dan nilai
def get_user_input():
n = int(input("Masukkan jumlah titik data: "))
x_points = []
y_points = []
for i in range(n):
x = float(input(f"Masukkan nilai x({i}): "))
```

```
    y = float(input(f"Masukkan nilai y({i}): "))
    x_points.append(x)
    y_points.append(y)
    return x_points, y_points

    # Fungsi utama
    def main():
    # Mendapatkan nilai yang dimasukkan
    x_points, y_points = get_user_input()

    # Menampilkan titik data yang dimasukkan
    print("Titik data yang dimasukkan:")
    for x, y in zip(x_points, y_points):
    print(f"({x}, {y})")

    # Menghitung nilai interpolasi pada titik tertentu
    x_interpolate = float(input("Masukkan nilai x yang ingin
    diinterpolasi: "))
    y_interpolate = lagrange_interpolation(x_points, y_points,
    x_interpolate)
    print(f"Nilai interpolasi pada x = {x_interpolate} adalah y =
    {y_inter40polate}")

    # Membuat grafik interpolasi
    x_plot = np.linspace(min(x_points) - 1, max(x_points) + 1,
    1000)
    y_plot = [lagrange_interpolation(x_points, y_points, x) for x
    in x_plot]

    plt.plot(x_plot, y_plot, color='royalblue',
    label='Interpolasi Lagrange')
    plt.scatter(x_points, y_points, color='magenta', label='Titik
    Data')
    plt.scatter([x_interpolate,[y_interpolate],
    color='mediumpurple',zorder=5,label=f'Interpolasi
    (x={x_interpolate})')
    plt.xlabel('x')
    plt.ylabel('y')
```

```
    plt.title('Interpolasi Lagrange')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Menjalankan program utama
    if __name__ == "__main__":
    main()
```

Penjelasan alur kode:

1. **Input nilai titik data**

```
    # Fungsi untuk memasukkan jumlah titik dan nilai
    def get_user_input():
        n = int(input("Masukkan jumlah titik data: "))
        x_points = []
        y_points = []
        for i in range(n):
            x = float(input(f"Masukkan nilai x({i}): "))
            y = float(input(f"Masukkan nilai y({i}): "))
            x_points.append(x)
            y_points.append(y)
        return x_points, y_points
```

Dengan kode di atas, pengguna dapat memasukkan jumlah titik data dan nilai tiap titik.

2. Inisialisasi fungsi interpolasi Lagrange

```
    def lagrange_interpolation(x_points, y_points, x):
        n = len(x_points)
        result = 0.0
        for i in range(n):
            term = y_points[i]
            for j in range(n):
                if i != j:
                    term *= (x - x_points[j]) / (x_points[i] -
    x_points[j])
            result += term
        return result
```

Dengan kode di atas, Fungsi lagrange_interpolation(x_points, y_points, x) menghitung nilai interpolasi Lagrange pada titik x tertentu berdasarkan titik data yang dimasukkan. Fungsi ini mengikuti algoritma interpolasi Lagrange, yaitu dengan mengalikan setiap nilai y dengan basis polinomial Lagrange dan menambahkannya ke nilai hasil akhir.

3. Menghasilkan grafik interpolasi

```
# Membuat grafik interpolasi
    x_plot = np.linspace(min(x_points) - 1, max(x_points) + 1,
1000)
    y_plot = [lagrange_interpolation(x_points, y_points, x)
for x in x_plot]

    plt.plot(x_plot, y_plot, color='royalblue',
label='Interpolasi Lagrange')
    plt.scatter(x_points, y_points, color='magenta',
label='Titik Data')
    plt.scatter([x_interpolate], [y_interpolate],
color='mediumpurple', zorder=5, label=f'Interpolasi
(x={x_interpolate})')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Interpolasi Lagrange')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Setelah nilai interpolasi dihitung, sebuah grafik dibuat untuk menampilkan hasil interpolasi Lagrange. Rentang nilai x untuk plotting dihasilkan menggunakan `np.linspace` untuk mencakup rentang yang lebih luas dari titik data. Fungsi interpolasi Lagrange dihitung untuk setiap nilai x dalam rentang tersebut. Garis interpolasi dan titik data ditampilkan dalam grafik menggunakan `plt.plot` dan `plt.scatter`.

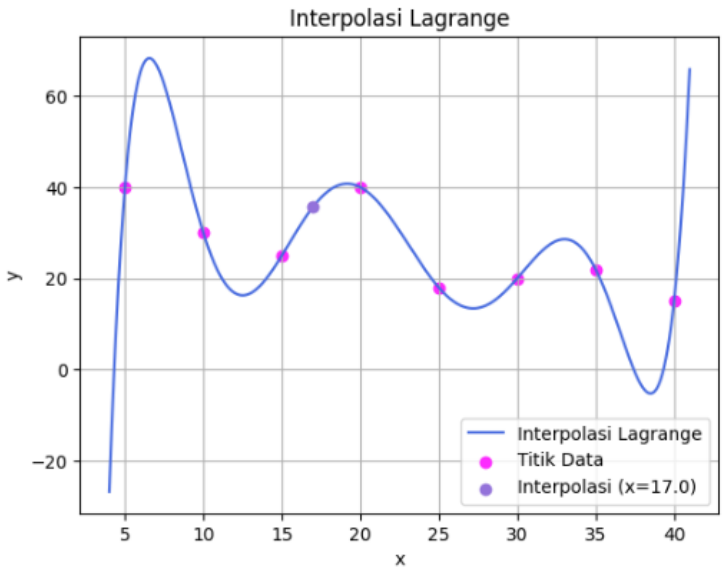
Code testing:

Dalam code testing, jumlah titik data dan nilainya disesuaikan dengan soal. Sehingga masukannya akan seperti berikut:

```
Masukkan jumlah titik data: 8
Masukkan nilai x(0): 5
Masukkan nilai y(0): 40
Masukkan nilai x(1): 10
Masukkan nilai y(1): 30
Masukkan nilai x(2): 15
Masukkan nilai y(2): 25
Masukkan nilai x(3): 20
Masukkan nilai y(3): 40
Masukkan nilai x(4): 25
Masukkan nilai y(4): 18
Masukkan nilai x(5): 30
Masukkan nilai y(5): 20
Masukkan nilai x(6): 35
Masukkan nilai y(6): 22
Masukkan nilai x(7): 40
Masukkan nilai y(7): 15
```

Dari nilai yang dimasukkan, akan dihasilkan jawaban nilai interpolasi dan grafik sebagai berikut:

```
Titik data yang dimasukkan:
(5.0, 40.0)
(10.0, 30.0)
(15.0, 25.0)
(20.0, 40.0)
(25.0, 18.0)
(30.0, 20.0)
(35.0, 22.0)
(40.0, 15.0)
Masukkan nilai x yang ingin diinterpolasi: 17
Nilai interpolasi pada x = 17.0 adalah y = 35.720872959999994
```



2. Polinom Newton

Dalam interpolasi Lagrange terdapat kelemahan, yaitu ketika terjadi penambahan atau pengurangan titik data yang diketahui maka bentuk polinom sebelumnya tidak dapat digunakan lagi. Hal ini tentu akan merepotkan karena setiap adanya perubahan kuantitas titik data harus dicari bentuk polinom dari awal lagi. Oleh karena itu, diperlukan solusi untuk mengatasi ini, yaitui dengan menggunakan interpolasi newton.

Tahapan pembentukan polinom newton adalah sebagai berikut:

$$p_1(x) = p_0(x) + a_1(x - x_0)$$
$$= a_0 + a_1(x - x_0)$$

$$p_2(x) = p_1(x) + a_2(x - x_0)(x - x_1)$$
$$= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

$$p_3(x) = p_2(x) + a_3(x - x_0)(x - x_1)(x - x_2)$$
$$= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$$

$$p_n(x) = p_{n-1}(x) + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$
$$= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$$
$$+ \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1})$$

Di mana, nilai konstanta a merupakan nilai selisih terbagi. Dengan nilai:

$$a_0 = f(x_0)$$
$$a_1 = f[x_1, x_0]$$
$$a_2 = f[x_2, x_1, x_0]$$
$$a_n = f[x_n, x_{n-1}, \dots, x_1, x_0]$$

Dengan demikian polinom newton dapat ditulis dalam bentuk lengkap sebagai berikut:

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

Source Code:

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


# Fungsi untuk menghitung konstanta a pada interpolasi Newton
def calculate_constants(x_points, y_points):
n = len(x_points)
constants = np.copy(y_points)
```

```
    for j in range(1, n):
    for i in range(n-1, j-1, -1):
    constants[i] = (constants[i] - constants[i-1]) / (x_points[i]
    - x_points[i-j])

    return constants

    # Fungsi untuk menghitung nilai interpolasi Newton pada titik
    x
    def newton_interpolation(x_points, constants, x):
    n = len(x_points)
    result = constants[-1]

    for i in range(n-2, -1, -1):
    result = result * (x - x_points[i]) + constants[i]

    return result

    # Fungsi untuk memasukkan jumlah titik dan nilai
    def get_user_input():
    n = int(input("Masukkan jumlah titik data: "))
    x_points = []
    y_points = []
    for i in range(n):
    x = float(input(f"Masukkan nilai x({i}): "))
    y = float(input(f"Masukkan nilai y({i}): "))
    x_points.append(x)
    y_points.append(y)
    return x_points, y_points

    # Fungsi utama
    def main():
    # Mendapatkan nilai yang dimasukkan
    x_points, y_points = get_user_input()

    # Menghitung koefisien interpolasi Newton
    constants = calculate_constants(x_points, y_points)
```

```
    # Menampilkan titik data yang dimasukkan
    print("Titik data yang dimasukkan:")
    for x, y in zip(x_points, y_points):
    print(f"({x}, {y})")

    # Menghitung nilai interpolasi pada titik tertentu
    x_interpolate = float(input("Masukkan nilai x yang ingin
    diinterpolasi: "))
    y_interpolate = newton_interpolation(x_points, constants,
    x_interpolate)
    print(f"Nilai interpolasi pada x = {x_interpolate} adalah y =
    {y_interpolate}")

    # Membuat grafik interpolasi
    x_plot = np.linspace(min(x_points) - 1, max(x_points) + 1,
    1000)
    y_plot = [newton_interpolation(x_points, constants, x) for x
    in x_plot]

    plt.plot(x_plot, y_plot, color='hotpink', label='Interpolasi
    Newton')
    plt.scatter(x_points, y_points, color='orange', label='Titik
    Data')
    plt.scatter([x_interpolate], [y_interpolate],
    color='blueviolet', zorder=5, label=f'Interpolasi
    (x={x_interpolate})')
    plt.xlabel('x (kg/mm^2)')
    plt.ylabel('y (jam)')
    plt.title('Interpolasi Newton')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Menjalankan program utama
    if __name__ == "__main__":
    main()
```

Penjelasan alur kode:

1. Input nilai titik data

```
# Fungsi untuk memasukkan jumlah titik dan nilai
def get_user_input():
n = int(input("Masukkan jumlah titik data: "))
x_points = []
y_points = []
for i in range(n):
x = float(input(f"Masukkan nilai x({i}): "))
y = float(input(f"Masukkan nilai y({i}): "))
x_points.append(x)
y_points.append(y)
return x_points, y_points
```

Dengan kode di atas, pengguna dapat memasukkan jumlah titik data dan nilai tiap titik

2. Melakukan perhitungan nilai konstanta a (selisih terbagi)

```
# Fungsi untuk menghitung konstanta a pada interpolasi Newton
def calculate_constants(x_points, y_points):
n = len(x_points)
constants = np.copy(y_points)
for j in range(1, n):
for i in range(n-1, j-1, -1):
constants[i] = (constants[i] - constants[i-1]) / (x_points[i]
- x_points[i-j])


return constants
```

Fungsi di atas digunakan dalam menghitung nilai konstanta a pada interpolasi newton atau disebut juga dengan selisih terbagi (*divided-difference*) di mana nilainya nanti akan digunakan dalam menghitung nilai interpolasi.

3. Menghitung nilai interpolasi

```
# Fungsi untuk menghitung nilai interpolasi Newton pada titik
x
def newton_interpolation(x_points, constants, x):
n = len(x_points)
result = constants[-1]
```

```
    for i in range(n-2, -1, -1):
    result = result * (x - x_points[i]) + constants[i]

    return result
```

Fungsi `newton_interpolation` menghitung nilai interpolasi Newton pada titik x yang diberikan. Ini dilakukan dengan menggunakan konstanta a interpolasi Newton yang telah dihitung sebelumnya. Nilai interpolasi dihitung menggunakan rumus:

$$\begin{aligned} p_n(x) &= p_{n-1}(x) + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) \\ &= a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) \\ &\quad + \dots + a_n(x - x_0)(x - x_1) \dots (x - x_{n-1}) \end{aligned}$$

4. Menghasilkan grafik interpolasi

```
    # Membuat grafik interpolasi
    x_plot = np.linspace(min(x_points) - 1, max(x_points) + 1,
    1000)
    y_plot = [newton_interpolation(x_points, constants, x) for x
    in x_plot]

    plt.plot(x_plot, y_plot, color='hotpink', label='Interpolasi
    Newton')
    plt.scatter(x_points, y_points, color='orange', label='Titik
    Data')
    plt.scatter([x_interpolate], [y_interpolate],
    color='blueviolet', zorder=5, label=f'Interpolasi
    (x={x_interpolate})')
    plt.xlabel('x (kg/mm^2)')
    plt.ylabel('y (jam)')
    plt.title('Interpolasi Newton')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Setelah nilai-nilai interpolasi dihitung, grafik interpolasi Newton dan titik data ditampilkan menggunakan Matplotlib. Kurva interpolasi dihasilkan menggunakan nilai-nilai `x_plot` yang dihasilkan secara linier dengan menggunakan NumPy `linspace`. Titik-titik data dan titik interpolasi juga ditampilkan pada grafik.

Code Testing:

Dalam code testing, jumlah titik data dan nilainya disesuaikan dengan soal. Sehingga masukannya akan seperti berikut:

```
Masukkan jumlah titik data: 8
Masukkan nilai x(0): 5
Masukkan nilai y(0): 40
Masukkan nilai x(1): 10
Masukkan nilai y(1): 30
Masukkan nilai x(2): 15
Masukkan nilai y(2): 25
Masukkan nilai x(3): 20
Masukkan nilai y(3): 40
Masukkan nilai x(4): 25
Masukkan nilai y(4): 18
Masukkan nilai x(5): 30
Masukkan nilai y(5): 20
Masukkan nilai x(6): 35
Masukkan nilai y(6): 22
Masukkan nilai x(7): 40
Masukkan nilai y(7): 15
```

Dari nilai yang dimasukkan, akan dihasilkan jawaban nilai interpolasi dan grafik sebagai berikut:

```
Titik data yang dimasukkan:
(5.0, 40.0)
(10.0, 30.0)
(15.0, 25.0)
(20.0, 40.0)
(25.0, 18.0)
(30.0, 20.0)
(35.0, 22.0)
(40.0, 15.0)
Masukkan nilai x yang ingin diinterpolasi: 17
Nilai interpolasi pada x = 17.0 adalah y = 35.72087296
```