Nama : Namira Nurfaliani

NIM : 21120122140135

Kelas : Metode Numerik/C

Implementasi Interpolasi Menggunakan Metode Polinom Newton

https://github.com/iranamira/metnum_Pertemuan10_Namira-Nurfaliani

Penerapan Metode Lagrange terhadap soal adalah menggunakan Bahasa pemograman python. Alur untuk menemukan hasil dari implementasi interpolasi dengan menggunakan Polinom Newton :

1. Mempersiapkan input data dan titik interpolasi.
2. Menghitung nilai interpolasi menggunakan metode Newton.
3. Menampilkan hasil interpolasi.
4. Menghitung nilai interpolasi untuk rentang nilai x dan y untuk plotting.
5. Membuat dan menampilkan grafik interpolasi dan titik data asli.

Source Code :

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

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

def newton_divided_diff(x, y):
    """
    Fungsi untuk menghitung tabel selisih terbagi Newton.

    Args:
    x (numpy array): array dari nilai x yang diketahui
    y (numpy array): array dari nilai y yang diketahui

    Returns:
    numpy array: tabel selisih terbagi Newton
    """
    n = len(y)
    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])

    return coef

def newton_interpolation(x_data, y_data, x):
    """
    Fungsi untuk menghitung nilai interpolasi Newton pada titik x.
```

```
    Args:
    x_data (numpy array): array dari nilai x yang diketahui
    y_data (numpy array): array dari nilai y yang diketahui
    x (float): nilai x yang ingin diinterpolasi

    Returns:
    float: nilai y hasil interpolasi
    """
    coef = newton_divided_diff(x_data, y_data)
    n = len(x_data)
    y_interp = coef[0,0]
    for i in range(1, n):
        term = coef[0,i]
        for j in range(i):
            term *= (x - x_data[j])
        y_interp += term
    return y_interp

# Testing interpolasi pada beberapa titik
test_points = np.linspace(5, 40, 100)
interpolated_values = [newton_interpolation(x, y, point) for point in
test_points]

# Plotting hasil interpolasi
plt.figure(figsize=(10, 6))
plt.plot(test_points, interpolated_values, label='Interpolasi Newton',
color='blue')
plt.scatter(x, y, color='red', label='Data asli')
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 Kode Program :

1. Import Library

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

- `Numpy` : digunakan untuk operasi matematis pada array.
- `matplotlib.pyplot` : digunakan untuk membuat plot grafik.

2. Data yang diberikan

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

- `x` : adalah array yang berisi nilai tegangan dalam satuan kg/mm^2.
- `y` : adalah array yang berisi waktu patah dalam satuan jam.

3. Fungsi untuk menghitung table selisih terbagi newton

```
def newton_divided_diff(x, y):
    """
```

```
    Args:
    x (numpy array): array dari nilai x yang diketahui
    y (numpy array): array dari nilai y yang diketahui

    Returns:
    numpy array: tabel selisih terbagi Newton
    """
    n = len(y)
    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])

    return coef
```

- `n` : adalah panjang dari array y.
- `coef` : adalah matriks n x n yang diisi dengan nol menggunakan np.zeros
- `coef[:,0]` = y : mengisi kolom pertama dari coef dengan nilai-nilai y.
- Dua nested loop digunakan untuk menghitung selisih terbagi:
- Loop luar `j` : bergerak dari 1 ke n-1.
- Loop dalam `i` : bergerak dari 0 ke n-j.
- Setiap elemen coef[i][j] dihitung sebagai selisih antara elemen berikutnya dan elemen saat ini dibagi dengan perbedaan nilai x yang sesuai.

4. Fungsi untuk menghitung nilai interpolasi Newton pada titik x

```
def newton_interpolation(x_data, y_data, x):
    """
    Fungsi untuk menghitung nilai interpolasi Newton pada titik x.

    Args:
    x_data (numpy array): array dari nilai x yang diketahui
    y_data (numpy array): array dari nilai y yang diketahui
    x (float): nilai x yang ingin diinterpolasi

    Returns:
    float: nilai y hasil interpolasi
    """
    coef = newton_divided_diff(x_data, y_data)
    n = len(x_data)
    y_interp = coef[0, 0]
    for i in range(1, n):
        term = coef[0, i]
        for j in range(i):
            term *= (x - x_data[j])
        y_interp += term
    return y_interp
```

- `Coef` : adalah tabel selisih terbagi yang dihitung dari newton_divided_diff.
- `n` : adalah panjang dari array x_data.
- `y_interp` : diinisialisasi dengan elemen pertama dari coef.
- Loop `i` digunakan untuk menambahkan setiap term `y_interp`
- `term` : adalah nilai awal dari `coef[0, i]`

- Nested loop `j` mengalikan `term` dengan (x - x_data[j]) untuk setiap `j` dari 0 ke i-1.
- `term` kemudian ditambahkan ke y_interp.

5. Menghitung nilai interpolasi

```
test_points = np.linspace(5, 40, 100)
interpolated_values = [newton_interpolation(x, y, point) for point
in test_points]
```

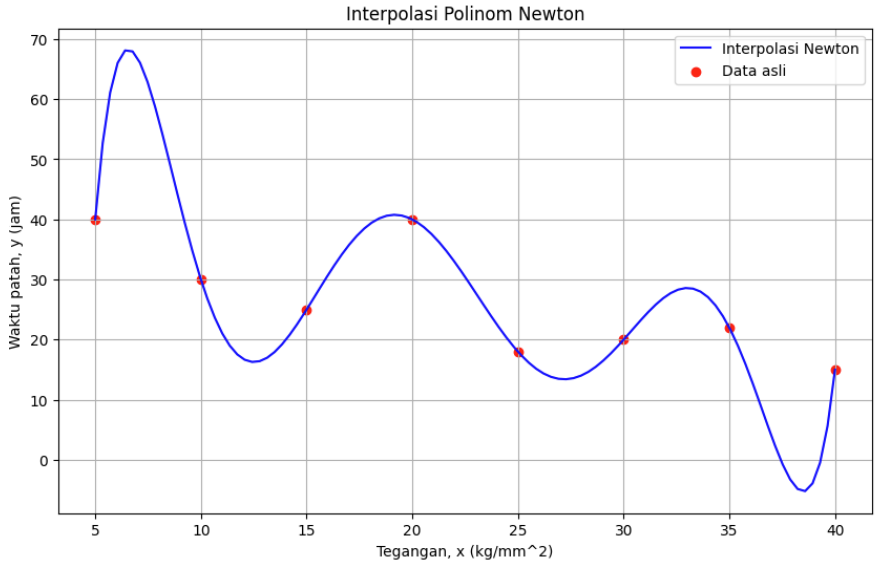
- `test_points` : adalah array yang terdiri dari 100 nilai yang tersebar merata antara 5 dan 40.
- `interpolated_values` : adalah daftar yang berisi hasil interpolasi untuk setiap titik dalam test_points.

6. Plotting hasil interpolasi

```
plt.figure(figsize=(10, 6))
plt.plot(test_points, interpolated_values, label='Interpolasi
Newton', color='blue')
plt.scatter(x, y, color='red', label='Data asli')
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` : Membuat figure baru dengan ukuran 10x6 inci.
- `plt.plot` : Menggambar kurva interpolasi (test_points vs. interpolated_values) dengan warna biru.
- `plt.scatter` : Menggambar titik data asli (x vs. y) dengan warna merah.
- `plt.xlabel` dan `plt.ylabel` : Menambahkan label untuk sumbu x dan y.
- `plt.title` : Menambahkan judul pada grafik.
- `plt.legend()` : Menambahkan legenda untuk membedakan antara data asli dan interpolasi.
- `plt.grid(True)` : Mengaktifkan grid pada grafik.
- `plt.show()` : Menampilkan grafik dengan plt.show().

Analisi Hasil dari menggunakan metode Polinom Newton



Gambar Hasil dari Polinom Newton 1

1. **Kesesuaian Interpolasi dengan Data Asli:** Garis biru yang merupakan hasil interpolasi Newton melewati semua titik data asli (ditandai dengan titik merah), menunjukkan bahwa interpolasi polinomial Newton telah berhasil menghubungkan semua titik data yang diberikan.
2. **Polinomial yang Kompleks:** Polinomial interpolasi menunjukkan perilaku yang kompleks, dengan beberapa titik ekstrim lokal (puncak dan lembah) yang cukup tajam. Pada interpolasi polinomial tinggi karena polinomial tersebut cenderung berosilasi, terutama ketika ada banyak titik data.
3. **Titik-titik Ekstrim:** Terdapat beberapa puncak dan lembah yang signifikan antara titik data, misalnya, antara tegangan 5 kg/mm² hingga 10 kg/mm² terdapat puncak yang sangat tinggi dan setelah tegangan 35 kg/mm² terdapat penurunan yang tajam.
4. **Kelancaran Kurva:** Secara keseluruhan, kurva interpolasi terlihat cukup halus namun tetap menunjukkan beberapa perubahan yang tajam di beberapa tempat.

Kesimpulan

1. Garis biru yang merupakan hasil interpolasi Newton melewati semua titik data asli (ditandai dengan titik merah), menunjukkan bahwa interpolasi polinomial Newton telah berhasil menghubungkan semua titik data yang diberikan.
2. Polinomial interpolasi menunjukkan perilaku yang kompleks, dengan beberapa titik ekstrim lokal (puncak dan lembah) yang cukup tajam. Hal ini umum terjadi pada

interpolasi polinomial tinggi karena polinomial tersebut cenderung berosilasi, terutama ketika ada banyak titik data.

3. Terdapat beberapa puncak dan lembah yang signifikan antara titik data, misalnya, antara tegangan 5 kg/mm² hingga 10 kg/mm² terdapat puncak yang sangat tinggi dan setelah tegangan 35 kg/mm² terdapat penurunan yang tajam.
4. Polinomial interpolasi tidak hanya menghubungkan titik data tetapi juga menghasilkan fluktuasi di antara titik-titik tersebut.