

Nama : Femas Arianda Rizki

NIM : 21120122130080

Kelas : Metode Numerik – B

1. Lagrange Polynomial Interpolation

Kode sumber, kode testing, dan plot grafik:

```
# Nama      : Femas Arianda Rizki
# NIM       : 21120122130080
# Kelas     : Metode Numerik - B

# Kode Sumber
import matplotlib.pyplot as plt
import numpy as np

def lagrange_interpolation(points, x):
    """
    Menghitung nilai interpolasi Lagrange pada titik x.

    Parameters:
    points (list of tuple): Daftar titik (x, y).
    x (float): Nilai x yang ingin diinterpolasi.

    Returns:
    float: Nilai interpolasi pada titik x.
    """
    # variabel untuk menyimpan hasil akhir dari interpolasi
    total = 0.0
    # variabel untuk menyimpan jumlah titik dalam points
    n = len(points)
    # looping I, untuk tiap titik xi, yi dalam points
    for i in range(n):
        # menyimpan nilai x dan y dari titik ke-i
        xi, yi = points[i]
        # menyimpan nilai yi untuk nanti dikalikan
        term = yi

        # looping II, untuk tiap titik xj, yj dalam daftar points,
        # kecuali jika i dan j sama
        for j in range(n):
            if i != j:
                # mendapatkan nilai x dari titik j ke dalam
                # variabel xj
                xj, _ = points[j]
                term *= (x - xj) / (xi - xj)

        total += term
```

```

        return total

# Kode Testing, contoh penyelesaian problem
points = [(5, 10), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), (40, 15)]
x = 33
interpolated_value = lagrange_interpolation(points, x)
print(f"P({x}) = {interpolated_value:.5f}")
print(f"Jadi nilai interpolasi pada x = {x} yaitu {interpolated_value:.5f}")

# Plot Grafik hasil interpolasi dengan 5 <= x <= 40
x_values = np.arange(5, 40.1, 0.1)
y_values = [lagrange_interpolation(points, x) for x in x_values]

plt.plot(x_values, y_values, "r")
plt.grid()
plt.xlim(0, 42)

plt.show()

```

Penjelasan kode:

a. Import library

```

import matplotlib.pyplot as plt
import numpy as np

```

- `import matplotlib.pyplot as plt`: Mengimpor library `matplotlib.pyplot` dan memberi alias `plt` untuk memudahkan penulisan. Library ini digunakan untuk membuat grafik.
- `import numpy as np`: Mengimpor library `numpy` dan memberi alias `np`. Library ini digunakan untuk komputasi numerik yang efisien, terutama untuk bekerja dengan array dan fungsi matematika.

b. Definisi fungsi interpolasi lagrange

```

def lagrange_interpolation(points, x):
    """
    Menghitung nilai interpolasi Lagrange pada titik x.

    Parameters:
    points (list of tuple): Daftar titik (x, y).
    x (float): Nilai x yang ingin diinterpolasi.

    Returns:
    float: Nilai interpolasi pada titik x.
    """

```

- `def lagrange_interpolation(points, x):` Mendefinisikan fungsi `lagrange_interpolation` yang menerima dua parameter: `points` (daftar titik (x, y)) dan `x` (nilai x yang ingin diinterpolasi).
- `""" ... """`: Docstring yang menjelaskan fungsi, parameter yang diterima, dan nilai yang dikembalikan oleh fungsi.

c. Inisialisasi variabel

```
# variabel untuk menyimpan hasil akhir dari interpolasi
total = 0.0
# variabel untuk menyimpan jumlah titik dalam points
n = len(points)
```

- `total = 0.0`: Menginisialisasi variabel `total` untuk menyimpan hasil akhir interpolasi.
- `n = len(points)`: Menghitung jumlah titik dalam `points` dan menyimpannya dalam variabel `n`.

d. Looping untuk menghitung interpolasi

```
# looping I, untuk tiap titik xi, yi dalam points
for i in range(n):
    # menyimpan nilai x dan y dari titik ke-i
    xi, yi = points[i]
    # menyimpan nilai yi untuk nanti dikalikan
    term = yi

    # looping II, untuk tiap titik xj, yj dalam daftar
    # points, kecuali jika i dan j sama
    for j in range(n):
        if i != j:
            # mendapatkan nilai x dari titik j ke dalam
            # variabel xj
            xj, _ = points[j]
            term *= (x - xj) / (xi - xj)

    total += term

return total
```

- `for i in range(n)`: Looping pertama untuk mengiterasi setiap titik (x_i, y_i) dalam `points`.
- `xi, yi = points[i]`: Mendapatkan nilai x dan y dari titik ke- i .
- `term = yi`: Inisialisasi `term` dengan nilai y_i untuk digunakan dalam perhitungan.
- `for j in range(n)`: Looping kedua untuk mengiterasi setiap titik (x_j, y_j) dalam `points`, kecuali saat $i == j$.
- `if i != j`: Mengecek apakah indeks i tidak sama dengan j .
- `xj, _ = points[j]`: Mendapatkan nilai x dari titik ke- j .
- `term *= (x - xj) / (xi - xj)`: Menghitung nilai interpolasi Lagrange.
- `total += term`: Menambahkan `term` ke `total`.

- `return total`: Mengembalikan hasil interpolasi.

e. Kode testing

```
# Kode Testing, contoh penyelesaian problem
points = [(5, 10), (10, 30), (15, 25), (20, 40), (25, 18),
(30, 20), (35, 22), (40, 15)]
x = 33
interpolated_value = lagrange_interpolation(points, x)
print(f"P({x}) = {interpolated_value:.5f}")
print(f"Jadi nilai interpolasi pada x = {x} yaitu
{interpolated_value:.5f}")
```

- `points = [(5, 10), ...]`: Mendefinisikan daftar titik (x, y).
- `x = 33`: Menetapkan nilai x yang akan diinterpolasi.
- `interpolated_value = lagrange_interpolation(points, x)`: Memanggil fungsi `lagrange_interpolation` untuk menghitung nilai interpolasi pada x.
- `print(f"P({x}) = {interpolated_value:.5f}")`: Menampilkan hasil interpolasi.
- `print(f"Jadi nilai interpolasi pada x = {x} yaitu {interpolated_value:.5f}")`: Menampilkan hasil interpolasi dengan penjelasan.

f. Plot grafik hasil interpolasi

```
# Plot Grafik hasil interpolasi dengan 5 <= x <= 40
x_values = np.arange(5, 40.1, 0.1)
y_values = [lagrange_interpolation(points, x) for x in
x_values]

plt.plot(x_values, y_values, "r")
plt.grid()
plt.xlim(0, 42)

plt.show()
```

- `x_values = np.arange(5, 40.1, 0.1)`: Membuat array `x_values` dari 5 hingga 40 dengan interval 0.1 menggunakan `numpy`.
- `y_values = [lagrange_interpolation(points, x) for x in x_values]`: Menghitung nilai interpolasi untuk setiap x dalam `x_values` dan menyimpannya dalam `y_values`.
- `plt.plot(x_values, y_values, "r")`: Membuat plot `x_values` versus `y_values` dengan warna merah ("r").
- `plt.grid()`: Menampilkan grid pada grafik.

- `plt.xlim(0, 42)`: Menetapkan batas x pada grafik dari 0 hingga 42.
- `plt.show()`: Menampilkan grafik.

2. Newton Polynomial Interpolation

Kode sumber, kode testing, dan plot grafik:

```
# Nama      : Femas Arianda Rizki
# NIM       : 21120122130080
# Kelas     : Metode Numerik - B

# Kode Sumber
import numpy as np
import matplotlib.pyplot as plt

def newton_interpolation(points, x):
    """
    Menghitung nilai interpolasi Newton pada titik x.

    Parameters:
    points (list of tuple): Daftar titik (x, y).
    x (float): Nilai x yang ingin diinterpolasi.

    Returns:
    float: Nilai interpolasi pada titik x.
    """
    n = len(points)
    # Membuat tabel selisih terbagi
    divided_diff = np.zeros((n, n))
    # Memasukkan nilai y ke kolom pertama dari tabel selisih terbagi
    for i in range(n):
        divided_diff[i][0] = points[i][1]

    # Menghitung selisih terbagi
    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]) / (points[i+j][0] - points[i][0])

    # Menghitung nilai interpolasi pada x
    result = divided_diff[0][0]
    product = 1.0
    for i in range(1, n):
        product *= (x - points[i-1][0])
        result += divided_diff[0][i] * product
```

```

        return result

# Kode Testing, contoh penyelesaian problem
points = [(5, 10), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), (40, 15)]
x = 33
interpolated_value = newton_interpolation(points, x)
print(f"P({x}) = {interpolated_value:.5f}")
print(f"Jadi nilai interpolasi pada x = {x} yaitu {interpolated_value:.5f}")

# Plot Grafik hasil interpolasi dengan 5 <= x <= 40
x_values = np.arange(5, 40.1, 0.1)
y_values = [newton_interpolation(points, x) for x in x_values]

plt.plot(x_values, y_values, "b")
plt.grid()
plt.xlim(0, 42)
plt.xlabel('x')
plt.ylabel('Interpolated y')
plt.title('Interpolasi Polinomial Newton')

plt.show()

```

Penjelasan kode:

a. Import library

```

import numpy as np
import matplotlib.pyplot as plt

```

- `numpy` digunakan untuk operasi numerik, seperti pembuatan tabel selisih terbagi.
- `matplotlib.pyplot` digunakan untuk plotting grafik hasil interpolasi.

b. Fungsi newton interpolation

```

def newton_interpolation(points, x):
    """
    Menghitung nilai interpolasi Newton pada titik x.

    Parameters:
    points (list of tuple): Daftar titik (x, y).
    x (float): Nilai x yang ingin diinterpolasi.

    Returns:
    float: Nilai interpolasi pada titik x.
    """

```

- Definisi fungsi `newton_interpolation` untuk menghitung nilai interpolasi Newton pada titik `x`.
- Parameter `points` adalah daftar titik `(x, y)` yang diketahui.

- Parameter x adalah nilai x yang ingin diinterpolasi.
- Fungsi mengembalikan nilai interpolasi pada titik x .

c. Inisialisasi dan tabel selisih terbagi

```
n = len(points)
# Membuat tabel selisih terbagi
divided_diff = np.zeros((n, n))
# Memasukkan nilai y ke kolom pertama dari tabel selisih
terbagi
for i in range(n):
    divided_diff[i][0] = points[i][1]
```

- `n` menyimpan jumlah titik dalam `points`.
- `divided_diff` adalah tabel selisih terbagi, diinisialisasi sebagai array 2D berukuran $n \times n$ dengan nilai awal 0.
- Nilai y dari tiap titik (x , y) diisi ke kolom pertama `divided_diff`.

d. Menghitung selisih terbagi

```
# Menghitung selisih terbagi
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]) / (points[i+j][0] - points[i][0])
```

- Menghitung selisih terbagi menggunakan dua loop bersarang.
- Loop luar dengan indeks j berjalan dari 1 hingga $n-1$.
- Loop dalam dengan indeks i berjalan dari 0 hingga $n-j-1$.
- Setiap elemen `divided_diff[i][j]` dihitung berdasarkan perbedaan elemen di kolom sebelumnya dibagi selisih x dari titik yang sesuai.

e. Menghitung nilai interpolasi

```
# Menghitung nilai interpolasi pada x
result = divided_diff[0][0]
product = 1.0
for i in range(1, n):
    product *= (x - points[i-1][0])
    result += divided_diff[0][i] * product

return result
```

- `result` diinisialisasi dengan nilai `divided_diff[0][0]`.
- `product` diinisialisasi dengan nilai 1.0.
- Loop dengan indeks i berjalan dari 1 hingga $n-1$ untuk menghitung nilai interpolasi pada x .
- `product` diperbarui dengan mengalikan $(x - \text{points}[i-1][0])$.

- `result` diperbarui dengan menambahkan `divided_diff[0][i] * product`.
- Nilai `result` dikembalikan sebagai hasil interpolasi.

f. Kode testing

```
points = [(5, 10), (10, 30), (15, 25), (20, 40), (25, 18),
(30, 20), (35, 22), (40, 15)]
x = 33
interpolated_value = newton_interpolation(points, x)
print(f"P({x}) = {interpolated_value:.5f}")
print(f"Jadi nilai interpolasi pada x = {x} yaitu
{interpolated_value:.5f}")
```

- `points` adalah daftar titik yang diketahui.
- `x` adalah nilai yang ingin diinterpolasi (33).
- `interpolated_value` menyimpan hasil interpolasi menggunakan fungsi `newton_interpolation`.
- Hasil interpolasi dicetak.

g. Plot grafik hasil interpolasi

```
# Plot Grafik hasil interpolasi dengan 5 <= x <= 40
x_values = np.arange(5, 40.1, 0.1)
y_values = [newton_interpolation(points, x) for x in x_values]

plt.plot(x_values, y_values, "b")
plt.grid()
plt.xlim(0, 42)

plt.show()
```

- `x_values` adalah array nilai `x` dari 5 hingga 40 dengan interval 0.1.
- `y_values` adalah daftar nilai interpolasi yang dihitung untuk setiap `x` dalam `x_values` menggunakan fungsi `newton_interpolation`.
- Plot hasil interpolasi dengan warna biru ("b").
- Menambahkan grid pada plot.
- Membatasi sumbu-x dari 0 hingga 42.
- Menampilkan plot.