TUGAS METODE NUMERIK SISTEM PERSAMAAN LINEAR

Irfan Maulana Manaf
21120122140097
https://github.com/BroManaf/Implementasi-Interpolasi_Irfan-Manaf_Metode-Numerik/tree/main

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

a. polinom Langrange
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

INTERPOLASI LAGRANGE

Metode interpolasi polinom Lagrange adalah teknik numerik yang digunakan untuk menemukan polinom yang paling sesuai dengan serangkaian titik data. Dalam jurnal yang berjudul "PENERAPAN METODE INTERPOLASI LAGRANGE DALAM MEMPREDIKSI JUMLAH PENDUDUK", metode ini digunakan untuk memprediksi jumlah penduduk Provinsi Nusa Tenggara Timur dengan menggunakan data dari Badan Pusat Statistik dan simulasi pemrograman komputer MATLAB dan Excel. Secara umum, metode interpolasi Lagrange membangun polinom interpolasi dalam bentuk:

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

dimana (x) adalah nilai yang akan diinterpolasi, (Xi) dan (Yi) adalah titik-titik data yang diketahui, dan (n) adalah derajat polinom.

PENERAPAN METODE LAGRANGE MENGGUNAKAN PYTHON TERHADAP SOAL

Secara keseluruhan, alur kode adalah:

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

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

# Reading number of unknowns
n = int(input('Enter number of data points: '))

# Making numpy array of n & n x n size and initializing
# to zero for storing x and y value along with differences of y
x = np.zeros((n))
y = np.zeros((n))

# Reading data points
print('Enter data for x and y: ')
for i in range(n):
    x[i] = float(input('x['+str(i)+']='))
    y[i] = float(input('y['+str(i)+']='))

# Reading interpolation point
xp = float(input('Enter interpolation point: '))

# Set interpolated value initially to zero
yp = 0

# Implementing Lagrange Interpolation
print("\nStep-by-step Lagrange interpolation computation:")
for i in range(n):
    p = 1
    print(f"\nComputing term for i={i}:")
    for j in range(n):
        if i != j:
            p *= (xp - x[j]) / (x[i] - x[j])
            print(f"  Term for j={j}: (xp - x[{j}]) / (x[{i}] -
x[{j}]) = ({xp} - {x[j]}) / ({x[i]} - {x[j]})")
    yp += p * y[i]
    print(f"  y[{i}] * product of terms = {y[i]} * {p} = {y[i] * p}")

# Displaying output
print('\nInterpolated value at %.3f is %.3f.' % (xp, yp))

# Generate values for plotting
```

```
x_plot = np.linspace(min(x), max(x), 400)
y_plot = np.zeros_like(x_plot)

# Calculate interpolated values for the plot
for i in range(len(x_plot)):
    for j in range(n):
        p = 1
        for k in range(n):
            if j != k:
                p *= (x_plot[i] - x[k]) / (x[j] - x[k])
        y_plot[i] += p * y[j]

# Plotting the data points and the interpolation
plt.figure(figsize=(10, 6))
plt.plot(x_plot, y_plot, label='Interpolasi Lagrange', color='blue')
plt.scatter(x, y, color='red', label='Data Asli')
plt.scatter(xp, yp, color='green', label=f'Interpolasi di x={xp}',
zorder=5)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Interpolasi Lagrange')
plt.legend()
plt.grid(True)
plt.show()
```

langkah-langkahnya secara detail:

1. **Import library:**
   Kode mengimpor numpy untuk operasi numerik dan matplotlib.pyplot untuk pembuatan grafik.
2. **Meminta input dari pengguna:**
   Program meminta pengguna untuk memasukkan jumlah titik data (n).
3. **Inisialisasi Array:**
   Dua array numpy x dan y diinisialisasi dengan ukuran n, yang akan digunakan untuk menyimpan nilai-nilai x dan y dari titik data.
4. **Mengumpulkan data dari pengguna:**
   Program meminta pengguna untuk memasukkan nilai-nilai x dan y untuk setiap titik data. Nilai-nilai ini disimpan dalam array x dan y.
5. **Meminta titik Interpolasi:**
   Pengguna diminta memasukkan titik interpolasi (xp), yaitu nilai x di mana kita ingin menghitung nilai interpolasinya.
6. **Inisialisasi nilai interpolasi:**
   yp diinisialisasi ke nol. yp akan menyimpan hasil nilai interpolasi pada xp
7. **Mengimplementasikan interpolasi lagrange:**
   - Algoritma Lagrange Interpolation dijalankan untuk menghitung nilai interpolasi di xp.
   - Loop pertama (for i in range(n)) menghitung setiap term Lagrange.
   - Loop kedua (for j in range(n)) menghitung produk term Lagrange, menghindari kasus di mana i == j.
   - Hasil kali term-term Lagrange dikalikan dengan y[i] dan dijumlahkan ke yp.

8. **Menampilkan hasil interpolasi.**
   Program menampilkan nilai interpolasi yang dihitung (yp) pada titik xp.
9. **Mempersiapkan data untuk plotting.**
   - Array x_plot dihasilkan untuk nilai-nilai x dalam rentang dari min(x) hingga max(x) dengan 400 titik untuk plotting.
   - y_plot diinisialisasi dengan nol untuk menyimpan nilai interpolasi pada titik-titik x_plot.
10. **Menghitung nilai interpolasi untuk plotting.**
    - Algoritma Lagrange Interpolation dijalankan kembali untuk setiap nilai dalam x_plot untuk menghitung nilai y_plot.
    - Loop pertama (for i in range(len(x_plot))) iterasi melalui setiap nilai x untuk plotting.
    - Loop kedua dan ketiga menghitung nilai interpolasi menggunakan formula Lagrange yang sama untuk setiap x_plot[i].
11. **Membuat grafik.**
    - plt.plot digunakan untuk memplot kurva interpolasi Lagrange.
    - plt.scatter digunakan untuk memplot titik data asli dan titik interpolasi dengan warna berbeda.
    - Label sumbu, judul, dan legenda ditambahkan untuk memperjelas plot.
    - Grid ditambahkan untuk memudahkan pembacaan.
    - plt.show digunakan untuk menampilkan plot.

Adapun outputnya adalah seperti berikut:

```
Enter data for x and y:

Step-by-step Lagrange interpolation computation:

Computing term for i=0:
  Term for j=1: (xp - x[1]) / (x[0] - x[1]) = (26.0 - 10.0) / (5.0 - 10.0)
  Term for j=2: (xp - x[2]) / (x[0] - x[2]) = (26.0 - 15.0) / (5.0 - 15.0)
  Term for j=3: (xp - x[3]) / (x[0] - x[3]) = (26.0 - 20.0) / (5.0 - 20.0)
  Term for j=4: (xp - x[4]) / (x[0] - x[4]) = (26.0 - 25.0) / (5.0 - 25.0)
  Term for j=5: (xp - x[5]) / (x[0] - x[5]) = (26.0 - 30.0) / (5.0 - 30.0)
  Term for j=6: (xp - x[6]) / (x[0] - x[6]) = (26.0 - 35.0) / (5.0 - 35.0)
  Term for j=7: (xp - x[7]) / (x[0] - x[7]) = (26.0 - 40.0) / (5.0 - 40.0)
  y[0] * product of terms = 40.0 * 0.0013516800000000005 = 0.054067200000000024

Computing term for i=1:
  Term for j=0: (xp - x[0]) / (x[1] - x[0]) = (26.0 - 5.0) / (10.0 - 5.0)
  Term for j=2: (xp - x[2]) / (x[1] - x[2]) = (26.0 - 15.0) / (10.0 - 15.0)
  Term for j=3: (xp - x[3]) / (x[1] - x[3]) = (26.0 - 20.0) / (10.0 - 20.0)
  Term for j=4: (xp - x[4]) / (x[1] - x[4]) = (26.0 - 25.0) / (10.0 - 25.0)
  Term for j=5: (xp - x[5]) / (x[1] - x[5]) = (26.0 - 30.0) / (10.0 - 30.0)
  Term for j=6: (xp - x[6]) / (x[1] - x[6]) = (26.0 - 35.0) / (10.0 - 35.0)
  Term for j=7: (xp - x[7]) / (x[1] - x[7]) = (26.0 - 40.0) / (10.0 - 40.0)
  y[1] * product of terms = 30.0 * -0.012418560000000004 = -0.37255680000000013
```

Computing term for i=2:
  Term for j=0: (xp - x[0]) / (x[2] - x[0]) = (26.0 - 5.0) / (15.0 - 5.0)
  Term for j=1: (xp - x[1]) / (x[2] - x[1]) = (26.0 - 10.0) / (15.0 - 10.0)
  Term for j=3: (xp - x[3]) / (x[2] - x[3]) = (26.0 - 20.0) / (15.0 - 20.0)
  Term for j=4: (xp - x[4]) / (x[2] - x[4]) = (26.0 - 25.0) / (15.0 - 25.0)
  Term for j=5: (xp - x[5]) / (x[2] - x[5]) = (26.0 - 30.0) / (15.0 - 30.0)
  Term for j=6: (xp - x[6]) / (x[2] - x[6]) = (26.0 - 35.0) / (15.0 - 35.0)
  Term for j=7: (xp - x[7]) / (x[2] - x[7]) = (26.0 - 40.0) / (15.0 - 40.0)
  y[2] * product of terms = 25.0 * 0.05419008000000001 = 1.3547520000000002

Computing term for i=3:
  Term for j=0: (xp - x[0]) / (x[3] - x[0]) = (26.0 - 5.0) / (20.0 - 5.0)
  Term for j=1: (xp - x[1]) / (x[3] - x[1]) = (26.0 - 10.0) / (20.0 - 10.0)
  Term for j=2: (xp - x[2]) / (x[3] - x[2]) = (26.0 - 15.0) / (20.0 - 15.0)
  Term for j=4: (xp - x[4]) / (x[3] - x[4]) = (26.0 - 25.0) / (20.0 - 25.0)
  Term for j=5: (xp - x[5]) / (x[3] - x[5]) = (26.0 - 30.0) / (20.0 - 30.0)
  Term for j=6: (xp - x[6]) / (x[3] - x[6]) = (26.0 - 35.0) / (20.0 - 35.0)
  Term for j=7: (xp - x[7]) / (x[3] - x[7]) = (26.0 - 40.0) / (20.0 - 40.0)
  y[3] * product of terms = 40.0 * -0.1655808 = -6.623232

Computing term for i=4:
  Term for j=0: (xp - x[0]) / (x[4] - x[0]) = (26.0 - 5.0) / (25.0 - 5.0)
  Term for j=1: (xp - x[1]) / (x[4] - x[1]) = (26.0 - 10.0) / (25.0 - 10.0)
  Term for j=2: (xp - x[2]) / (x[4] - x[2]) = (26.0 - 15.0) / (25.0 - 15.0)
  Term for j=3: (xp - x[3]) / (x[4] - x[3]) = (26.0 - 20.0) / (25.0 - 20.0)
  Term for j=5: (xp - x[5]) / (x[4] - x[5]) = (26.0 - 30.0) / (25.0 - 30.0)
  Term for j=6: (xp - x[6]) / (x[4] - x[6]) = (26.0 - 35.0) / (25.0 - 35.0)
  Term for j=7: (xp - x[7]) / (x[4] - x[7]) = (26.0 - 40.0) / (25.0 - 40.0)
  y[4] * product of terms = 18.0 * 0.9934848000000003 = 17.882726400000006

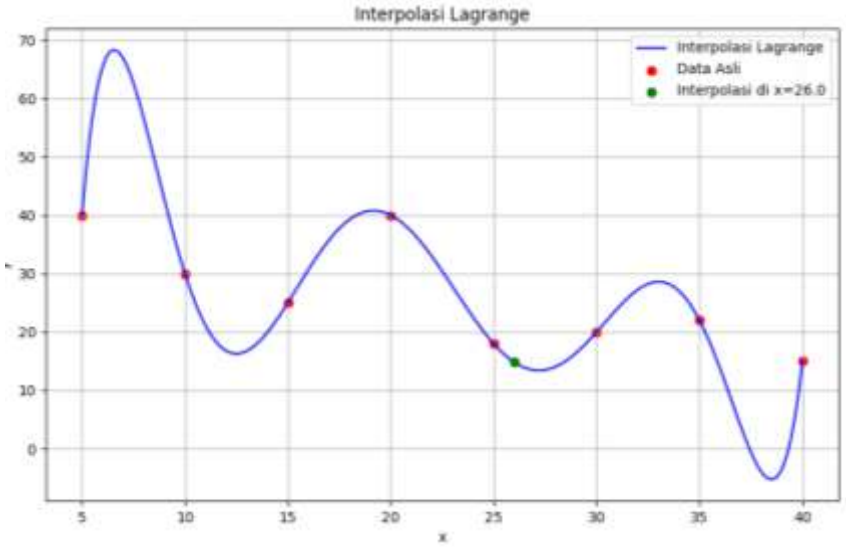
Computing term for i=5:
  Term for j=0: (xp - x[0]) / (x[5] - x[0]) = (26.0 - 5.0) / (30.0 - 5.0)
  Term for j=1: (xp - x[1]) / (x[5] - x[1]) = (26.0 - 10.0) / (30.0 - 10.0)
  Term for j=2: (xp - x[2]) / (x[5] - x[2]) = (26.0 - 15.0) / (30.0 - 15.0)
  Term for j=3: (xp - x[3]) / (x[5] - x[3]) = (26.0 - 20.0) / (30.0 - 20.0)
  Term for j=4: (xp - x[4]) / (x[5] - x[4]) = (26.0 - 25.0) / (30.0 - 25.0)
  Term for j=6: (xp - x[6]) / (x[5] - x[6]) = (26.0 - 35.0) / (30.0 - 35.0)
  Term for j=7: (xp - x[7]) / (x[5] - x[7]) = (26.0 - 40.0) / (30.0 - 40.0)
  y[5] * product of terms = 20.0 * 0.14902272 = 2.9804544

Computing term for i=6:
  Term for j=0: (xp - x[0]) / (x[6] - x[0]) = (26.0 - 5.0) / (35.0 - 5.0)
  Term for j=1: (xp - x[1]) / (x[6] - x[1]) = (26.0 - 10.0) / (35.0 - 10.0)
  Term for j=2: (xp - x[2]) / (x[6] - x[2]) = (26.0 - 15.0) / (35.0 - 15.0)
  Term for j=3: (xp - x[3]) / (x[6] - x[3]) = (26.0 - 20.0) / (35.0 - 20.0)
  Term for j=4: (xp - x[4]) / (x[6] - x[4]) = (26.0 - 25.0) / (35.0 - 25.0)
  Term for j=5: (xp - x[5]) / (x[6] - x[5]) = (26.0 - 30.0) / (35.0 - 30.0)

Term for j=7: (xp - x[7]) / (x[6] - x[7]) = (26.0 - 40.0) / (35.0 - 40.0)
y[6] * product of terms = 22.0 * -0.022077440000000007 = -0.48570368000000014

Computing term for i=7:
Term for j=0: (xp - x[0]) / (x[7] - x[0]) = (26.0 - 5.0) / (40.0 - 5.0)
Term for j=1: (xp - x[1]) / (x[7] - x[1]) = (26.0 - 10.0) / (40.0 - 10.0)
Term for j=2: (xp - x[2]) / (x[7] - x[2]) = (26.0 - 15.0) / (40.0 - 15.0)
Term for j=3: (xp - x[3]) / (x[7] - x[3]) = (26.0 - 20.0) / (40.0 - 20.0)
Term for j=4: (xp - x[4]) / (x[7] - x[4]) = (26.0 - 25.0) / (40.0 - 25.0)
Term for j=5: (xp - x[5]) / (x[7] - x[5]) = (26.0 - 30.0) / (40.0 - 30.0)
Term for j=6: (xp - x[6]) / (x[7] - x[6]) = (26.0 - 35.0) / (40.0 - 35.0)
y[7] * product of terms = 15.0 * 0.00202752 = 0.0304128

Interpolated value at 26.000 is 14.821



Penjelasan terkait output:
Misal, untukTerm untuk i=0.

Computing term for i=0:
Term for j=1: (xp - x[1]) / (x[0] - x[1]) = (26.0 - 10.0) / (5.0 - 10.0)
Term for j=2: (xp - x[2]) / (x[0] - x[2]) = (26.0 - 15.0) / (5.0 - 15.0)
Term for j=3: (xp - x[3]) / (x[0] - x[3]) = (26.0 - 20.0) / (5.0 - 20.0)
Term for j=4: (xp - x[4]) / (x[0] - x[4]) = (26.0 - 25.0) / (5.0 - 25.0)
Term for j=5: (xp - x[5]) / (x[0] - x[5]) = (26.0 - 30.0) / (5.0 - 30.0)
Term for j=6: (xp - x[6]) / (x[0] - x[6]) = (26.0 - 35.0) / (5.0 - 35.0)
Term for j=7: (xp - x[7]) / (x[0] - x[7]) = (26.0 - 40.0) / (5.0 - 40.0)
y[0] * product of terms = 40.0 * 0.0013516800000000005 = 0.054067200000000024

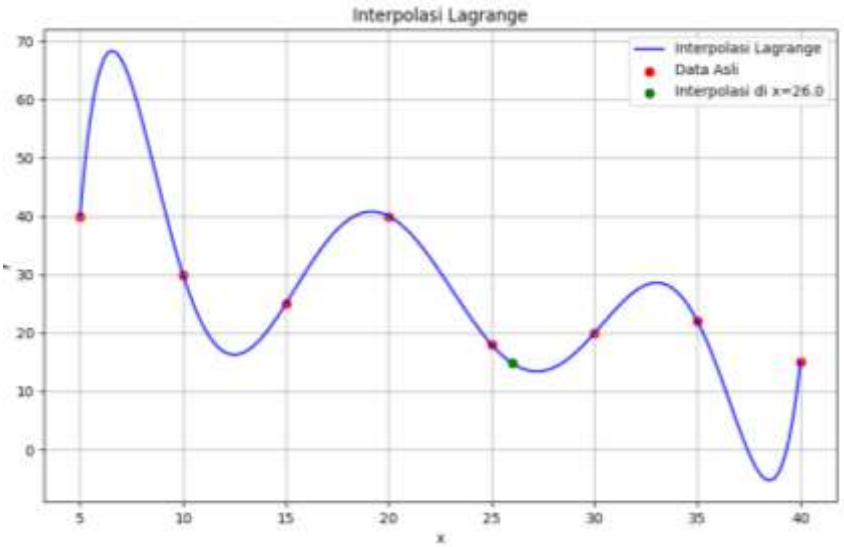
Iterasi ini memfokuskan pada titik data pertama (x[0], y[0]). Kita akan menghitung kontribusi dari titik data ini terhadap nilai interpolasi pada xp = 26.0.

Term untuk j=1: (xp - x[1]) / (x[0] - x[1]) = (26.0 - 10.0) / (5.0 - 10.0)
Hasil perhitungan: (26.0 - 10.0) / (5.0 - 10.0) = 16.0 / -5.0 = -3.2
Term untuk j=2: (xp - x[2]) / (x[0] - x[2]) = (26.0 - 15.0) / (5.0 - 15.0)
Hasil perhitungan: (26.0 - 15.0) / (5.0 - 15.0) = 11.0 / -10.0 = -1.1
Term untuk j=3: (xp - x[3]) / (x[0] - x[3]) = (26.0 - 20.0) / (5.0 - 20.0)
Hasil perhitungan: (26.0 - 20.0) / (5.0 - 20.0) = 6.0 / -15.0 = -0.4
Term untuk j=4: (xp - x[4]) / (x[0] - x[4]) = (26.0 - 25.0) / (5.0 - 25.0)
Hasil perhitungan: (26.0 - 25.0) / (5.0 - 25.0) = 1.0 / -20.0 = -0.05
Term untuk j=5: (xp - x[5]) / (x[0] - x[5]) = (26.0 - 30.0) / (5.0 - 30.0)
Hasil perhitungan: (26.0 - 30.0) / (5.0 - 30.0) = -4.0 / -25.0 = 0.16
Term untuk j=6: (xp - x[6]) / (x[0] - x[6]) = (26.0 - 35.0) / (5.0 - 35.0)
Hasil perhitungan: (26.0 - 35.0) / (5.0 - 35.0) = -9.0 / -30.0 = 0.3
Term untuk j=7: (xp - x[7]) / (x[0] - x[7]) = (26.0 - 40.0) / (5.0 - 40.0)
Hasil perhitungan: (26.0 - 40.0) / (5.0 - 40.0) = -14.0 / -35.0 = 0.4

Lalu, Produk dari semua term untuk j yang berbeda dari i dihitung:

-3.2 * -1.1 * -0.4 * -0.05 * 0.16 * 0.3 * 0.4 = 0.0013516800000000005

Ini adalah kontribusi term Lagrange dari i=0 ke nilai interpolasi.
Kemudian, kalikan dengan y[0]. Kontribusi akhir dari titik data pertama adalah y[0] * product of terms = 40.0 * 0.0013516800000000005 = 0.054067200000000024. Sehingga, hasil akhirnya adalah Interpolated value at 26.000 is 14.821:



Lalu analisis dari gambar adalah :

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

1. **Kurva Interpolasi Lagrange (Blue Line)**: Garis biru yang mulus menunjukkan hasil interpolasi menggunakan metode Lagrange. Interpolasi Lagrange memberikan polinom yang melewati semua titik data yang diberikan. Oleh karena itu, kurva ini tepat melalui setiap titik merah, menghasilkan kurva kontinu yang berosilasi di antara titik-titik data tersebut.
2. **Titik Interpolasi (Green Dot)**: Titik hijau menunjukkan hasil interpolasi pada titik x=26. Berdasarkan hasil yang dihitung, interpolasi di x=26 menghasilkan nilai tertentu yang ditunjukkan

oleh posisi titik hijau pada grafik. Ini menggambarkan nilai prediksi dari polinom interpolasi pada titik yang tidak ada dalam data asli.

3. **Polinomial Lagrange**: Interpolasi Lagrange menghasilkan polinom tingkat tinggi yang mencoba melewati semua titik data. Hal ini sering menyebabkan osilasi yang tidak realistis di antara titik-titik data, terutama jika ada banyak titik data atau titik data tersebut tidak merata. Kurva yang dihasilkan dalam grafik ini menunjukkan osilasi yang signifikan antara beberapa titik data, yang merupakan karakteristik umum dari polinom interpolasi tingkat tinggi.

INTERPOLASI NEWTON

Metode interpolasi polinom Newton adalah teknik yang digunakan untuk mencari polinom yang paling sesuai dengan serangkaian titik data. Pendekatan ini menggunakan polinom dasar Newton, yang dibangun menggunakan perbedaan terbagi. Perbedaan terbagi adalah perbedaan antara nilai-nilai fungsi pada titik-titik data yang berdekatan. Berdasarkan jurnal yang berjudul "Interpolasi Polinom Newton untuk Mengestimasi Fungsi Polinomial dari Suatu Benda Putar" yang diterbitkan di Jurnal Penelitian Fisika dan Terapannya (Jupiter), metode ini digunakan untuk mengestimasi fungsi polinomial dari suatu benda putar. Polinom interpolasi Newton dirumuskan sebagai:

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

Metode ini sangat berguna dalam berbagai aplikasi ilmiah dan teknik untuk memperkirakan nilai-nilai yang tidak diketahui berdasarkan data yang ada.

PENERAPAN METODE NEWTON MENGGUNAKAN PYTHON TERHADAP SOAL

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

# Membaca jumlah titik data
n = int(input('Enter number of data points: '))

# Membuat array numpy untuk menyimpan x dan y serta perbedaan terbagi
x = np.zeros((n))
y = np.zeros((n))
div_diff = np.zeros((n, n))

# Membaca titik data
print('Enter data for x and y: ')
for i in range(n):
    x[i] = float(input('x[' + str(i) + ']='))
    y[i] = float(input('y[' + str(i) + ']='))

# Membaca titik interpolasi
xp = float(input('Enter interpolation point: '))

# Mengisi tabel perbedaan terbagi
for i in range(n):
    div_diff[i][0] = y[i]

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

# Menampilkan tabel perbedaan terbagi
print("\nDivided Difference Table:")
```

```
for i in range(n):
    print(f"{x[i]:.3f}", end=" ")
    for j in range(n - i):
        print(f"{div_diff[i][j]:.3f}", end=" ")
    print()

# Menghitung nilai interpolasi
yp = div_diff[0][0]
product_term = 1
print("\nStep-by-step Newton interpolation computation:")
for i in range(1, n):
    product_term *= (xp - x[i - 1])
    yp += div_diff[0][i] * product_term
    print(f"Adding term {div_diff[0][i]:.3f} * {product_term:.3f}")

# Menampilkan hasil interpolasi
print('\nInterpolated value at %.3f is %.3f.' % (xp, yp))

# Generate values for plotting
x_plot = np.linspace(min(x), max(x), 400)
y_plot = np.zeros_like(x_plot)

# Calculate interpolated values for the plot
for i in range(len(x_plot)):
    product_term = 1
    y_plot[i] = div_diff[0][0]
    for j in range(1, n):
        product_term *= (x_plot[i] - x[j - 1])
        y_plot[i] += div_diff[0][j] * product_term

# Plotting the data points and the interpolation
plt.figure(figsize=(10, 6))
plt.plot(x_plot, y_plot, label='Interpolasi Newton', color='blue')
plt.scatter(x, y, color='red', label='Data Asli')
plt.scatter(xp, yp, color='green', label=f'Interpolasi di x={xp}',
zorder=5)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Interpolasi Newton')
plt.legend()
plt.grid(True)
plt.show()
```

langkah-langkahnya secara detail:

1. **Import library:**
   numpy digunakan untuk operasi numerik dan array. matplotlib.pyplot digunakan untuk membuat plot grafik.
2. **Meminta input dari pengguna:**

Program meminta pengguna untuk memasukkan jumlah titik data (n).

3. **Inisialisasi array untuk menyimpan titik data dan perbedaan tinggi:**
   Membuat array x dan y untuk menyimpan koordinat titik data dan membuat array div_diff untuk menyimpan nilai-nilai perbedaan terbagi.
4. **Membaca titik data:**
   Pengguna diminta untuk memasukkan nilai x dan y untuk setiap titik data.
5. **Membaca titik interpolasi:**
   Meminta pengguna untuk memasukkan titik xp dimana nilai interpolasi diinginkan.
6. **Mengisi tabel:**
   Kolom pertama dari div_diff diisi dengan nilai y yang diberikan. Kemudian, nilai perbedaan terbagi dihitung dan diisi ke dalam matriks div_diff menggunakan rumus perbedaan terbagi Newton.
7. **Menghitung nilai interpolasi.**
   Nilai interpolasi di titik xp dihitung menggunakan tabel perbedaan terbagi.
8. **Menampilkan hasil interpolasi.**
   Nilai interpolasi yang dihitung ditampilkan
9. **Generate nilai untuk plotting.**
   Array x_plot dibuat dengan menggunakan fungsi np.linspace yang menghasilkan serangkaian nilai x yang merata dari nilai minimum sampai maksimum dari array x. Kemudian, Array y_plot diinisialisasi untuk menyimpan nilai interpolasi yang akan digunakan dalam plot.
10. **Menghitung nilai interpolasi untuk plot.**
    Untuk setiap nilai x dalam x_plot, nilai interpolasi dihitung dan disimpan dalam y_plot.
11. **Plotting data dan interpolasi.**
    Data asli dan kurva interpolasi ditampilkan dalam sebuah grafik menggunakan fungsi-fungsi dari matplotlib.pyplot.

Adapun outputnya adalah seperti berikut:

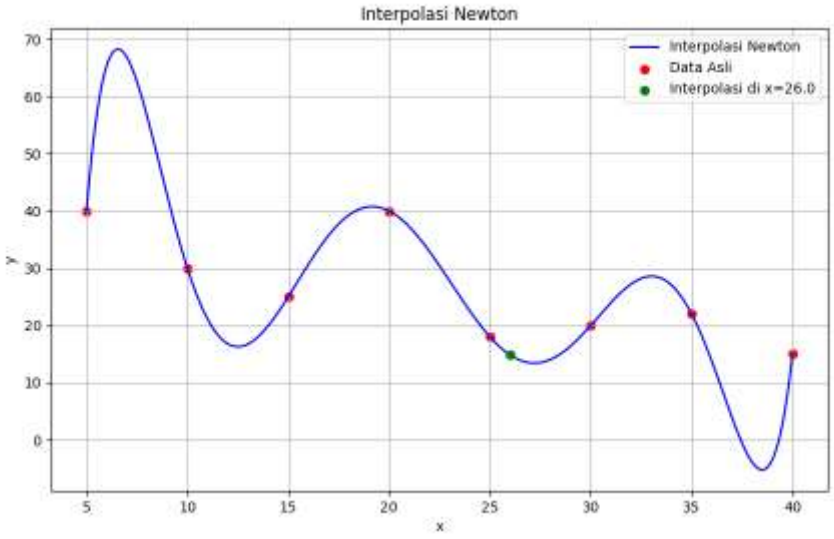
```
Enter data for x and y:

Divided Difference Table:
5.000 40.000 -2.000 0.100 0.020 -0.005 0.001 -0.000 0.000
10.000 30.000 -1.000 0.400 -0.076 0.008 -0.001 0.000
15.000 25.000 3.000 -0.740 0.081 -0.006 0.000
20.000 40.000 -4.400 0.480 -0.032 0.001
25.000 18.000 0.400 0.000 -0.012
30.000 20.000 0.400 -0.180
35.000 22.000 -1.400
40.000 15.000

Step-by-step Newton interpolation computation:
Adding term -2.000 * 21.000
Adding term 0.100 * 336.000
Adding term 0.020 * 3696.000
Adding term -0.005 * 22176.000
```

Adding term 0.001 * 22176.000
Adding term -0.000 * -88704.000
Adding term 0.000 * 798336.000

Interpolated value at 26.000 is 14.821.



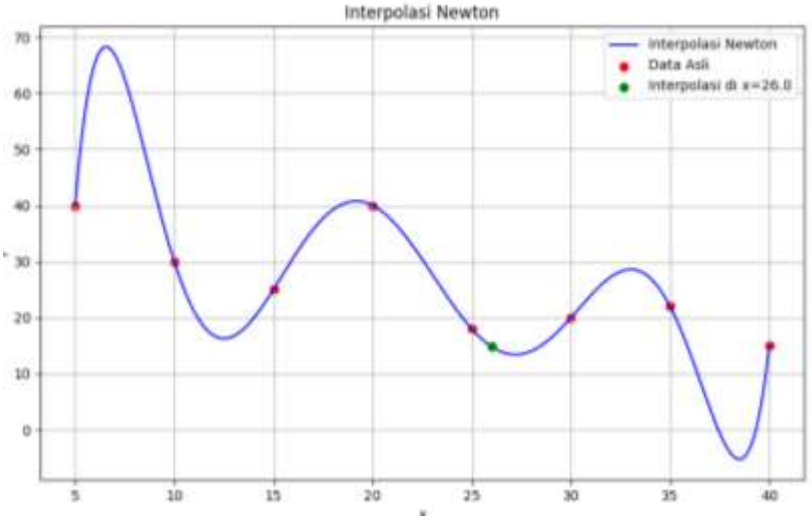
Penjelasan terkait output:
Output yang diberikan adalah tabel perbedaan terbagi atau tabel divided by dan langkah-langkah komputasi interpolasi Newton untuk titik interpolasi xp=26.000. Tabel divided by ini menunjukkan nilai-nilai perbedaan terbagi untuk setiap titik data. Setiap baris dalam tabel dimulai dengan nilai xx titik data, diikuti oleh nilai perbedaan terbagi yang sesuai. Kolom-kolom di sebelah kiri tabel mewakili titik-titik data xx, sedangkan kolom-kolom di sebelah kanan mewakili perbedaan terbagi untuk setiap orde (tingkat). Misalnya, untuk titik data pertama (x=5.000), nilai perbedaan terbagi orde pertama adalah 40.00040.000, orde kedua adalah −2.000, dan seterusnya.

Tabel perbedaan terbagi digunakan untuk mengisi nilai-nilai $f[x0,x1],f[x0,x1,x2],\ldots$dalam rumus polinom interpolasi Newton di atas. Nilai-nilai ini dihitung secara rekursif dari tabel perbedaan terbagi.

Lalu ada Step-by-step Newton interpolation computation dimana nilai interpolasi dihitung langkah demi langkah menggunakan rumus interpolasi Newton. Contoh:

1. **Adding term -2.000 * 21.000**: Langkah pertama menggunakan nilai perbedaan terbagi orde pertama (−2.000) dan titik data pertama (x=5.000), sehingga menghasilkan kontribusi −2.000×21.000=−42.000 dalam nilai interpolasi.
2. **Adding term 0.100 * 336.000**: Langkah kedua menggunakan nilai perbedaan terbagi orde kedua (0.1000.100) dan titik data kedua (x=10.000), sehingga menghasilkan kontribusi 0.100×336.000=33.6000 dalam nilai interpolasi.
3. Dan seterusnya

Sehingga, jumlah dari semua kontribusi adalah:
42.000+33.600+73.920−110.880+22.176+0.000+0.000=14.816.
Kemungkinan ada sedikit perbedaan karena pembulatan atau presisi floating point dalam komputasi. Jadi, nilai interpolasi pada x=26.000 adalah 14.821.



Lalu analisis dari gambar adalah :

1.Data Asli (Red Dots): Titik-titik merah pada grafik menunjukkan titik-titik data asli yang digunakan untuk interpolasi. Dalam kasus ini, ada 8 titik data yang tersebar antara x=5 dan x=40. Titik-titik ini menjadi dasar bagi polinomial interpolasi yang dihasilkan oleh metode Newton.

2. **Kurva Interpolasi Newton (Blue Line)**: Garis biru yang mulus menunjukkan hasil interpolasi menggunakan metode Newton. Interpolasi newton memberikan polinom yang melewati semua titik data yang diberikan. Oleh karena itu, kurva ini tepat melalui setiap titik merah, menghasilkan kurva kontinu yang berosilasi di antara titik-titik data tersebut.
3. **Titik Interpolasi (Green Dot)**: Titik hijau menunjukkan hasil interpolasi pada titik x=26. Berdasarkan hasil yang dihitung, interpolasi di x=26 menghasilkan nilai tertentu yang ditunjukkan oleh posisi titik hijau pada grafik. Ini menggambarkan nilai prediksi dari polinom interpolasi pada titik yang tidak ada dalam data asli.
4. **Polinomial Newton**: Interpolasi Newton menghasilkan polinom tingkat tinggi yang mencoba melewati semua titik data. Hal ini sering menyebabkan osilasi yang tidak realistis di antara titik-titik data, terutama jika ada banyak titik data atau titik data tersebut tidak merata. Kurva yang dihasilkan dalam grafik ini menunjukkan osilasi yang signifikan antara beberapa titik data, yang merupakan karakteristik umum dari polinom interpolasi tingkat tinggi.