

# PENJELASAN SOURCE CODE IMPLEMENTASI INTERPOLASI

## MATA KULIAH METODE NUMERIK

Nama : Novendra Anugrah Fitriatmoko  
NIM : 21120122130074  
Kelas : A

### A. Polinom Lagrange

Berikut ini source code yang digunakan untuk menyelesaikan permasalahan yang diberikan menggunakan Polinom Lagrange.

```
x = [5, 10, 15, 20, 25, 30, 35, 40]
y = [40, 30, 25, 40, 18, 20, 22, 15]

import matplotlib.pyplot as plt

def lagrange_interpolasi(x, y, xi):
    def L(k, xi):
        Lk = 1
        for i in range(len(x)):
            if i != k:
                Lk *= (xi - x[i]) / (x[k] - x[i])
        return Lk

    yi = []
    for xi in xi:
        yi.append(sum(y[k] * L(k, xi) for k in range(len(x))))
    return yi

x_values = [i for i in range(5, 41)]
y_values_lagrange = lagrange_interpolasi(x, y, x_values)

plt.plot(x, y, 'o', label='Data asli')
plt.plot(x_values, y_values_lagrange, '-', label='Interpolasi
Lagrange')
plt.xlabel('Tegangan (kg/mm2)')
plt.ylabel('Waktu patah (jam)')
plt.legend()
plt.show()
```

Penjelasan source code:

- Source code dimulai dengan memasukkan nilai  $x$  dan  $y$  yang telah diberikan pada permasalahan.
- Kemudian meng-import library “matplotlib.pyplot as plt” guna membuat grafik untuk hasil akhir nanti.
- Selanjutnya adalah pendeklarasian interpolasi lagrange guna menghitung nilai interpolasi lagrange pada setiap titik “xi”.
  - “def L(k, xi):” berguna menghitung polinom dasar lagrange dengan indeks “k”.
  - “Lk = 1” merupakan inisialisasi nilai basis lagrange.
  - “for i in range(len(x)):” merupakan perulangan atau loop yang digunakan untuk menghitung hasil polinom lagrange.
  - “yi = []” merupakan inisialisasi untuk menyimpan hasil perhitungan nilai interpolasi untuk setiap “xi”.
  - “yi.append(sum(y[k] \* L(k, xi) for k in range(len(x))))” guna menghitung nilai interpolasi lagrange pada setiap “xi” dan disimpan pada “yi”.
- Kemudian, kode “x\_values = [i for i in range(5, 41)]” digunakan untuk membuat nilai dari  $5 \leq x \leq 40$  sesuai dengan yang diminta pada permasalahan.
- “y\_values\_lagrange = lagrange\_interpolasi(x, y, x\_values)” berguna untuk menghitung nilai interpolasi lagrange pada setiap “x\_values”.
- Langkah terakhir yaitu membuat source code untuk menampilkan grafik interpolasi lagrange.
  - “plt.plot(x, y, 'o', label='Data asli’)” berguna menampilkan tanda “o” sebagai penanda “Data Asli”.
  - “plt.plot(x\_values, y\_values\_lagrange, '-', label='Interpolasi Lagrange’)” guna membuat hasil interpolasi lagrange sebagai garis (‘-’) dan memiliki label “‘Interpolasi Lagrange”.
  - “plt.xlabel('Tegangan (kg/mm<sup>2</sup>)’)” berfungsi membuat label “‘Tegangan (kg/mm<sup>2</sup>)” pada sumbu x.

- “plt.ylabel('Waktu patah (jam)')” berfungsi membuat label “'Waktu patah (jam)’” pada sumbu y.
- “plt.legend()” sebagai pembeda antara hasil interpolasi dan data asli.
- “plt.show()” guna menampilkan grafik hasil.

## B. Polinom Newton

Berikut ini source code yang digunakan untuk menyelesaikan permasalahan yang diberikan menggunakan Polinom Lagrange.

```
import matplotlib.pyplot as plt

x = [5, 10, 15, 20, 25, 30, 35, 40]
y = [40, 30, 25, 40, 18, 20, 22, 15]

def divided_diff(x, y):
    n = len(x)
    coef = [[0] * n for _ in range(n)]
    for i in range(n):
        coef[i][0] = y[i]
    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[i][i] for i in range(n)]

def newton_interpolasi(x, y, xi):
    coef = divided_diff(x, y)
    n = len(x)
    yi = []
    for xi_val in xi:
        term = coef[0]
        for k in range(1, n):
            prod = coef[k]
            for j in range(k):
                prod *= (xi_val - x[j])
            term += prod
        yi.append(term)
    return yi

def verify_interpolation_at_data_points(x, y):
```

```

y_interpolated = newton_interpolasi(x, y, x)
print("Verifikasi hasil interpolasi pada titik data asli:")
for xi, yi, yi_interpolated in zip(x, y, y_interpolated):
    print(f"x = {xi}, y_asli = {yi}, y_interpolasi = {yi_interpolated}")

verify_interpolation_at_data_points(x, y)

x_values = [i for i in range(5, 41)]
y_values_newton = newton_interpolasi(x, y, x_values)

plt.plot(x, y, 'o', label='Data asli')
plt.plot(x_values, y_values_newton, '-', label='Interpolasi Newton')
plt.xlabel('Tegangan (kg/mm2)')
plt.ylabel('Waktu patah (jam)')
plt.legend()
plt.show()

```

Penjelasan source code:

- Pertama dimulai dengan meng-import library “matplotlib.pyplot as plt” guna membuat grafik untuk hasil akhir nanti.
- Selanjutnya mendeklarasikan nilai x dan y yang telah diberikan pada permasalahan.
- Kemudian “def divided\_diff(x, y)” gunakan menghitung divided differences yang digunakan dalam interpolasi newton.
  - `n = len(x)` guna mendapatkan panjang dari daftar x.
  - `coef = [[0] * n for _ in range(n)]` berfungsi membuat matriks coef berukuran n x n dan menginisiasinya dengan 0.
  - `for i in range(n)` sebagai loop untuk mengisi kolom pertama matriks coef dengan nilai-nilai y.
  - `coef[i][0] = y[i]` berfungsi mengisi kolom pertama matriks coef dengan nilai-nilai y.
- Kemudian “for j in range(1, n):” berperan sebagai perulangan untuk menghitung nilai divided differences.
- Selanjutnya membuat method untuk menghitung hasil interpolasi newton “def newton\_interpolasi(x, y, xi):”.

- `"coef = divided_diff(x, y)"` guna memanggil fungsi `divided_diff` untuk mendapatkan divided differences.
- `"n = len(x)"` untuk mendapatkan panjang dari daftar `x`.
- `"yi = []"` inialisasi daftar untuk menyimpan nilai interpolasi pada setiap `xi`.
- `"for xi in xi"` loop melalui setiap nilai dalam `xi`.
- `"term = coef[0]"` inialisasi `term` dengan nilai divided difference pertama.
- `"for k in range(1, n)"` loop untuk menghitung nilai interpolasi.
- `"prod = coef[k]"` inialisasi `prod` dengan nilai divided difference ke-`k`.
- `"for j in range(k)"` loop untuk menghitung produk yang dibutuhkan untuk nilai interpolasi.
- `"prod *= (xi - x[j])"` menghitung produk untuk nilai interpolasi.
- `"term += prod"` menambahkan produk ke `term`.
- `"yi.append(term)"` menambahkan nilai interpolasi ke daftar `yi`.
- `"return yi"` mengembalikan daftar nilai interpolasi `yi`.
- Selanjutnya membuat verifikasi hasil interpolasi pada titik data asli `"def verify_interpolation_at_data_points(x, y)"`.
- `"verify_interpolation_at_data_points(x, y)"` memanggil fungsi verifikasi yang telah dideklarasikan sebelumnya.
- Kemudian, kode `"x_values = [i for i in range(5, 41)]"` digunakan untuk membuat nilai dari  $5 \leq x \leq 40$  sesuai dengan yang diminta pada permasalahan.
- `"y_values_lagrange = newton_interpolasi(x, y, x_values)"` berguna untuk menghitung nilai interpolasi lagrange pada setiap `"x_values"`.
- Langkah terakhir yaitu membuat source code untuk menampilkan grafik interpolasi newton.
  - `"plt.plot(x, y, 'o', label='Data asli')"` berguna menampilkan tanda "o" sebagai penanda "Data Asli".
  - `"plt.plot(x_values, y_values_lagrange, '-', label='Interpolasi Lagrange')"` guna membuat hasil interpolasi

lagrange sebagai garis ('-') dan memiliki label "'Interpolasi Newton".

- `plt.xlabel('Tegangan (kg/mm2)')` berfungsi membuat label `"Tegangan (kg/mm2)"` pada sumbu x.
- `plt.ylabel('Waktu patah (jam)')` berfungsi membuat label `"Waktu patah (jam)"` pada sumbu y.
- `plt.legend()` sebagai pembeda antara hasil interpolasi dan data asli.
- `plt.show()` guna menampilkan grafik hasil.

### C. Testing Code

Berikut ini adalah testing code yang digunakan untuk menampilkan hasil dari dua source pada masing – masing metode yang digunakan.

```
from PolinomLagrange import lagrange_interpolasi
from PolinomNewton import newton_interpolasi
import matplotlib.pyplot as plt

def test_interpolation():
    x = [5, 10, 15, 20, 25, 30, 35, 40]
    y = [40, 30, 25, 40, 18, 20, 22, 15]

    xi_test = [7, 12, 17, 22, 27, 32, 37]

    y_lagrange_test = lagrange_interpolasi(x, y, xi_test)
    print("Hasil interpolasi Lagrange pada xi_test:")
    for xi, yi in zip(xi_test, y_lagrange_test):
        print(f"x = {xi}, y = {yi}")

    y_newton_test = newton_interpolasi(x, y, xi_test)
    print("Hasil interpolasi Newton pada xi_test:")
    for xi, yi in zip(xi_test, y_newton_test):
        print(f"x = {xi}, y = {yi}")

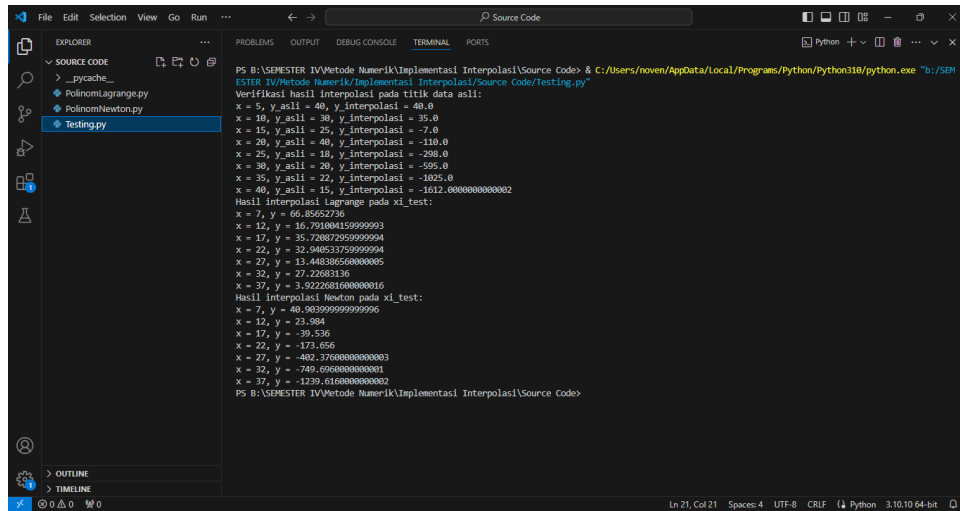
test_interpolation()
```

Penjelasan source code:

- Pertama, kita meng-import terlebih dahulu library dan function yang akan digunakan.

- `“from PolinomLagrange import lagrange_interpolasi”` meng-import function `“lagrange_interpolasi”` dari modul `“PolinomLagrange”`.
- `“from PolinomNewton import newton_interpolasi”` meng-import function `“newton_interpolasi”` dari modul `“PolinomNewton”`.
- `“import matplotlib.pyplot as plt”` meng-import library `“matplotlib.pyplot as plt”`.
- Mendefinisikan function untuk menguji interpolasi `“def test_interpolation():”`.
- Selanjutnya menambahkan nilai x dan y yang telah diketahui dari permasalahan yang diberikan.
- Kemudian mendefinisikan titik uji `“xi_test = [7, 12, 17, 22, 27, 32, 37]”`.
- Selanjutnya menghitung interpolasi lagrange menggunakan titik uji.
  - `“y_lagrange_test = lagrange_interpolasi(x, y, xi_test)”` untuk menghitung hasil interpolasi lagrange pada setiap titik uji.
  - `“print("Hasil interpolasi Lagrange pada xi_test:")”` guna mencetak hasil interpolasi lagrange.
  - `“for xi, yi in zip(xi_test, y_lagrange_test):”` perulangan untuk setiap nilai uji dan hasil interpolasi lagrange.
  - `“print(f"x = {xi}, y = {yi}")”` mencetak nilai ‘xi’ dan hasil interpolasi ‘yi’.
- Kemudian menghitung interpolasi newton menggunakan titik uji.
  - `“y_newton_test = newton_interpolasi(x, y, xi_test)”` untuk menghitung hasil interpolasi newton pada setiap titik uji.
  - `“print("Hasil interpolasi Newton pada xi_test:")”` guna mencetak hasil interpolasi newton.
  - `“for xi, yi in zip(xi_test, y_lagrange_test):”` perulangan untuk setiap nilai uji dan hasil interpolasi newton.
  - `“print(f"x = {xi}, y = {yi}")”` mencetak nilai ‘xi’ dan hasil newton ‘yi’.
- `“test_interpolation()”` berfungsi untuk memanggil function test agar dapat berjalan.

## D. Hasil Running



```
PS B:\SEMESTER IV\Metode Numerik\Implementasi Interpolasi\Source Code> & C:/Users/noven/AppData/Local/Programs/Python/Python310/python.exe "b:/SEMESTER IV\Metode Numerik\Implementasi Interpolasi\Source Code\Testing.py"
Verifikasi hasil interpolasi pada titik data asli:
x = 5, y asli = 40, y interpolasi = 40.0
x = 10, y asli = 30, y interpolasi = 35.0
x = 15, y asli = 25, y interpolasi = -7.0
x = 20, y asli = 40, y interpolasi = -110.0
x = 25, y asli = 18, y interpolasi = -298.0
x = 30, y asli = 20, y interpolasi = -595.0
x = 35, y asli = 22, y interpolasi = -1025.0
x = 40, y asli = 15, y interpolasi = -1612.0000000000002
Hasil interpolasi Lagrange pada xl_test:
x = 7, y = 66.85652750
x = 12, y = 16.791004150000003
x = 17, y = 35.728872959999994
x = 22, y = 32.948533759999994
x = 27, y = 13.448385600000005
x = 32, y = 27.22683136
x = 37, y = 3.9222681600000016
Hasil interpolasi Newton pada xl_test:
x = 7, y = 40.903999999999996
x = 12, y = 23.6984
x = 17, y = -39.536
x = 22, y = -173.656
x = 27, y = -482.37600000000003
x = 32, y = -740.69600000000001
x = 37, y = -1239.6160000000002
PS B:\SEMESTER IV\Metode Numerik\Implementasi Interpolasi\Source Code>
```

