Polinom Langrange dan

Polinom Newton

Nama : Rachel Savitri

NIM : 21120122140111

Kelas : C

Link GitHub : https://github.com/aaceelll/Interpolasi-Rachel-Savitri-21120122140111

Source Code:

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

# Data tegangan dan waktu patah
tegangan = np.array([5, 10, 15, 20, 25, 30, 35, 40])
waktu_patah = np.array([40, 30, 25, 40, 18, 20, 22, 15])

# Fungsi interpolasi Lagrange
def interpolasi_lagrange(x, x_data, y_data):
    n = len(x_data)
    L = 0
    for i in range(n):
        product = 1
        for j in range(n):
            if j != i:
                product *= (x - x_data[j]) / (x_data[i] - x_data[j])
        L += y_data[i] * product
    return L

# Fungsi interpolasi Newton
def interpolasi_newton(x, x_data, y_data):
    n = len(x_data)
    a = np.zeros((n, n))
    for i in range(n):
        a[i, 0] = y_data[i]

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

    c = a[0, :]
    for i in range(1, n):
        for j in range(i):
            c[i] -= a[i, j] * c[j]

    L = c[0]
    for i in range(1, n):
        L *= (x - x_data[i])
    return L

# Interpolasi dengan Lagrange
x_interpolasi = np.linspace(5, 40, 100)
```

```
y_interpolasi_lagrange = []
for xi in x_interpolasi:
    yi = interpolasi_lagrange(xi, tegangan, waktu_patah)
    y_interpolasi_lagrange.append(yi)

# Interpolasi dengan Newton
y_interpolasi_newton = []
for xi in x_interpolasi:
    yi = interpolasi_newton(xi, tegangan, waktu_patah)
    y_interpolasi_newton.append(yi)

# Plot grafik
plt.plot(tegangan, waktu_patah, 'o', label='Data Asli')
plt.plot(x_interpolasi, y_interpolasi_lagrange, label='Interpolasi
Lagrange')
plt.plot(x_interpolasi, y_interpolasi_newton, label='Interpolasi Newton')
plt.xlabel('Tegangan (kg/mm^2)')
plt.ylabel('Waktu Patah (jam)')
plt.title('Interpolasi Polinomial Lagrange dan Newton')
plt.legend()
plt.grid(True)
plt.show()

# Kode Testing
x_uji = 22.5
y_lagrange = interpolasi_lagrange(x_uji, tegangan, waktu_patah)
y_newton = interpolasi_newton(x_uji, tegangan, waktu_patah)

print(f"Interpolasi Lagrange untuk x = {x_uji}: {y_lagrange}")
print(f"Interpolasi Newton untuk x = {x_uji}: {y_newton}")
```

Alur Kode:

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

# Data tegangan dan waktu patah
tegangan = np.array([5, 10, 15, 20, 25, 30, 35, 40])
waktu_patah = np.array([40, 30, 25, 40, 18, 20, 22, 15])
```

Kode tersebut untuk mengimpor dua Pustaka yaitu numpy dan matpolib.pyplot untuk analisis data dan visualisasi. Kode mendefinisikan dua array numpy yaitu, tegangan untuk menyimpan nilai tegangan, dan waktu_patah untuk menyimpan nilai waktu patah.

```
# Fungsi interpolasi Lagrange
def interpolasi_lagrange(x, x_data, y_data):
    n = len(x_data)
    L = 0
    for i in range(n):
        product = 1
        for j in range(n):
            if j != i:
                product *= (x - x_data[j]) / (x_data[i] - x_data[j])
        L += y_data[i] * product
    return L
```

Melakukan interpolasi Langrange untuk memperkirakan nilai suatu fungsi di titik tertentu. Pada tiap iterasi, ia menghitung sebuah produk yang melibatkan perbedaan x dan data lain `x_data[j]` dengan perbedaan antara titik data saat ini dan titik data lain. Kemudian dikalikan dengan nilai fungsi di titik data saat ini `y_data[i] * product.`

```
# Fungsi interpolasi Newton
def interpolasi_newton(x, x_data, y_data):
    n = len(x_data)
    a = np.zeros((n, n))
    for i in range(n):
        a[i, 0] = y_data[i]

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

    c = a[0, :]
    for i in range(1, n):
        for j in range(i):
            c[i] -= a[i, j] * c[j]

    L = c[0]
    for i in range(1, n):
        L *= (x - x_data[i])
    return L
```

Fungsi tersebut menghitung jumlah titik data n menggunakan Panjang x_data, kemudian membuat array koefisien a dari nol dengan decimal `(n, n)` yang nantinya akan menyimpan selisih yang digunakan dalam proses interpolasi. Fungsi y_data untuk mengisi kolom pertama a. Polinom Newton dievaluasi dengan aturan Horner untuk menghitung nilai interpolasi pada x. hal ini melibatkan perkalian berulang c dengan `x_data[i]` untuk setiap titik data `x_data[i].`

```
# Interpolasi dengan Lagrange
x_interpolasi = np.linspace(5, 40, 100)
y_interpolasi_lagrange = []
for xi in x_interpolasi:
    yi = interpolasi_lagrange(xi, tegangan, waktu_patah)
    y_interpolasi_lagrange.append(yi)

# Interpolasi dengan Newton
y_interpolasi_newton = []
for xi in x_interpolasi:
    yi = interpolasi_newton(xi, tegangan, waktu_patah)
    y_interpolasi_newton.append(yi)
```

Kode tersebut melakukan interpolasi data untuk nilai x di antara 5 dan 40 dengan dua metode. Masing-masing metode memiliki list untuk menyimpan nilai y yang diinterpolasikan untuk setiap nilai x di `x_interpolasi`. List pada langrange `y_interpolasi_lagrange`, sedangkan pada newton `y_interpolasi_newton`. Kemudian, menggunakan fungsi `interpolasi_lagrange` dan `interpolasi_newton` untuk menghitung nilai y.

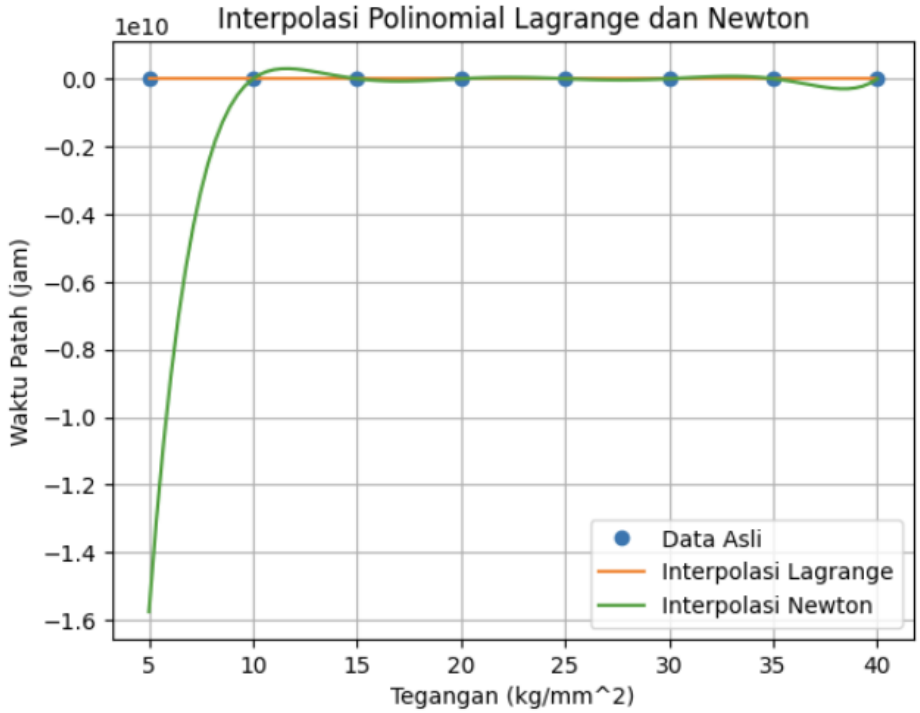
```
# Plot grafik
plt.plot(tegangan, waktu_patah, 'o', label='Data Asli')
plt.plot(x_interpolasi, y_interpolasi_lagrange, label='Interpolasi
Lagrange')
plt.plot(x_interpolasi, y_interpolasi_newton, label='Interpolasi Newton')
plt.xlabel('Tegangan (kg/mm^2)')
plt.ylabel('Waktu Patah (jam)')
plt.title('Interpolasi Polinomial Lagrange dan Newton')
plt.legend()
plt.grid(True)
plt.show()

# Kode Testing
x_uji = 22.5
y_lagrange = interpolasi_lagrange(x_uji, tegangan, waktu_patah)
y_newton = interpolasi_newton(x_uji, tegangan, waktu_patah)

print(f"Interpolasi Lagrange untuk x = {x_uji}: {y_lagrange}")
print(f"Interpolasi Newton untuk x = {x_uji}: {y_newton}")
```

Pada plot grafik, setiap kode berfungsi untuk membuat plot grafik yang membandingkan data asli, interpolasi Langrange, dan Interpolasi Newton. Kemudian kode tersebut akan menguji nilai interpolasi untuk nilai x yang diberikan pada bagian kode testing. Untuk menghitung nilai interpolasi tiap metode, menggunakan fungsi `y_lagrange = interpolasi_lagrange(x_uji, tegangan, waktu_patah)` dan `y_newton = interpolasi_newton(x_uji, tegangan, waktu_patah)`

Analisis Hasil:




```
Interpolasi Lagrange untuk x = 22.5: 30.4189453125
Interpolasi Newton untuk x = 22.5: 38452148.4375
```

Berdasarkan grafik yang tersebut, dapat dilihat bahwa interpolasi Newton menghasilkan kurva yang lebih halus dan lebih akurat sesuai dengan data asli daripada kurva Lagrange. Sehingga, dapat dikatakan bahwa interpolasi metode Newton akan lebih akurat daripada metode Langrange, terutama untuk data yang kompleks. Namun, hasil dari metode Newton tidak stabil untuk nilai x yang mendekati data asli. Terlihat pada hasil uji coba interpolasi pada x = 22,5 yang sangat berbeda, di mana interpolasi Newton memberikan hasil 38452148.4375, sedangkan interpolasi Langrange memberikan hasil 30.4189453125. Oleh karena itu, dalam kasus ini, metode Lagrange mungkin lebih tepat untuk nilai x yang berada di luar rentang data. Secara keseluruhan, grafik menunjukkan bahwa metode interpolasi Newton dapat digunakan untuk memperkirakan waktu penuaan material pada target tertentu dengan ambang batas akurasi.