**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].

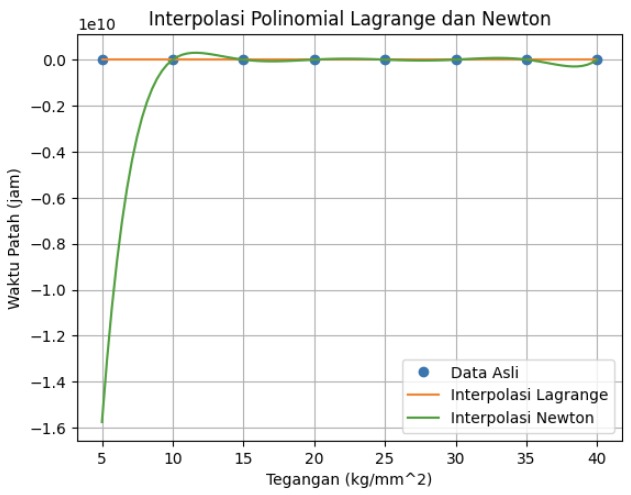
|  |
| --- |
| # Interpolasi dengan Lagrange  x\_interpolasi = np.linspace(5, 40, 100)  y\_interpolasi\_lagrange = []  for xi in x\_interpolasi:  yi = interpolasi\_lagrange(xi, tegangan, waktu\_patah)  y\_interpolasi\_lagrange.append(yi)  # Interpolasi dengan Newton  y\_interpolasi\_newton = []  for 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:





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.