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 Alur Kode**:

1. 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.

1. 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.

1. 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.

1. 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 (xi, yi) dalam points.
* xi, yi = points[i]: Mendapatkan nilai x dan y dari titik ke-i.
* term = yi: Inisialisasi term dengan nilai yi untuk digunakan dalam perhitungan.
* for j in range(n): Looping kedua untuk mengiterasi setiap titik (xj, yj) 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.

1. Kode testing

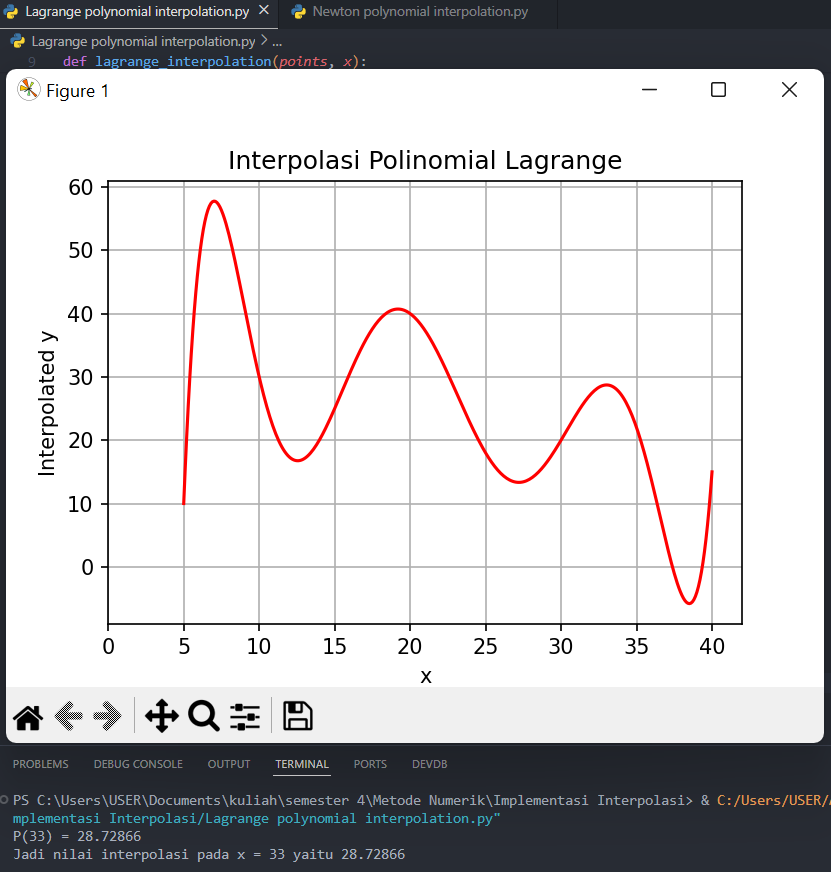
|  |
| --- |
| # Kode Testing, contoh penyelesaian problem  points = [(5, 10), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), (40, 15)]  x = 33  interpolated\_value = lagrange\_interpolation(points, x)  print(f"P({x}) = {interpolated\_value:.5f}")  print(f"Jadi nilai interpolasi pada x = {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.

1. 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.



**Analisis Hasil:**

* **Hasil Interpolasi pada x = 33**
* Hasil interpolasi pada titik x = 33 adalah 28.72866
* Artinya, nilai yang diinterpolasi pada titik x = 33 adalah sekitar 28.72866
* **Grafik Hasil Interpolasi**
* Grafik menunjukkan pola yang mengikuti titik-titik yang diketahui dengan baik.
* Grafik menunjukkan bahwa interpolasi polinomial Lagrange secara keseluruhan cocok dengan data yang diberikan.

**Penjabarannya:**

* Titik-titik data yang diketahui (5, 10), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), (40, 15) digunakan untuk melakukan interpolasi.
* Hasil interpolasi pada titik x = 33 adalah sekitar 28.72866, yang didapatkan dari perhitungan polinomial Lagrange.
* Grafik hasil interpolasi menunjukkan bahwa polinomial Lagrange mampu menginterpolasi data dengan baik, mengikuti pola data yang ada.

Dengan demikian, metode interpolasi polinomial Lagrange juga dapat digunakan untuk memperkirakan nilai di antara titik-titik data yang diketahui dengan cukup baik.

1. **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:

1. 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.

1. 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.

1. 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 x n dengan nilai awal 0.
* Nilai y dari tiap titik (x, y) diisi ke kolom pertama divided\_diff.

1. 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.

1. 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 - points[i-1][0]).
* result diperbarui dengan menambahkan divided\_diff[0][i] \* product.
* Nilai result dikembalikan sebagai hasil interpolasi.

1. Kode testing

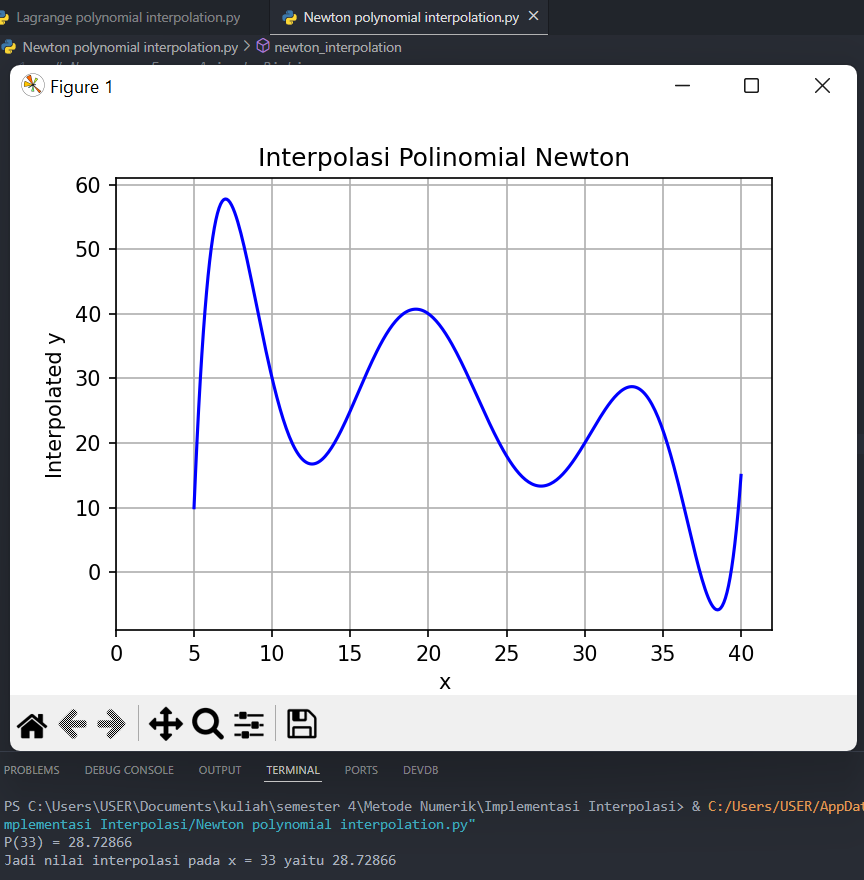
|  |
| --- |
| points = [(5, 10), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), (40, 15)]  x = 33  interpolated\_value = newton\_interpolation(points, x)  print(f"P({x}) = {interpolated\_value:.5f}")  print(f"Jadi nilai interpolasi pada x = {x} yaitu {interpolated\_value:.5f}") |

* 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.

1. 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.

****

**Analisis Hasil:**

* **Hasil Interpolasi pada x = 33**
* Hasil interpolasi pada titik x = 33 adalah 28.72866
* Artinya, nilai yang diinterpolasi pada titik x = 33 adalah sekitar 28.72866
* **Grafik Hasil Interpolasi**
* Grafik menunjukkan pola yang mengikuti titik-titik yang diketahui dengan baik.
* Grafik menunjukkan bahwa interpolasi polinomial Newton secara keseluruhan cocok dengan data yang diberikan.

**Penjabarannya:**

* Titik-titik data yang diketahui (5, 10), (10, 30), (15, 25), (20, 40), (25, 18), (30, 20), (35, 22), (40, 15) digunakan untuk melakukan interpolasi.
* Hasil interpolasi pada titik x = 33 adalah sekitar 28.72866, yang didapatkan dari perhitungan polinomial Newton.
* Grafik hasil interpolasi menunjukkan bahwa polinomial Newton mampu menginterpolasi data dengan baik, mengikuti pola data yang ada.

Dengan demikian, metode interpolasi polinomial Newton dapat digunakan untuk memperkirakan nilai di antara titik-titik data yang diketahui dengan cukup baik.