**PENJELASAN SOURCE CODE IMPLEMENTASI INTERPOLASI**

**MATA KULIAH METODE NUMERIK**

Nama : Novendra Anugrah Fitriatmoko

NIM : 21120122130074

Kelas : A

1. **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 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/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.

1. **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 digunaakn 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 menginisialisasinya 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 sebagi perulangan untuk menghitung nilai divided differences.
* Selanjutnya membuat metfhod 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 = []” inisialisasi daftar untuk menyimpan nilai interpolasi pada setiap xi.
* “for xi in xi” loop melalui setiap nilai dalam xi.
* “term = coef[0]” inisialisasi term dengan nilai divided difference pertama.
* “for k in range(1, n)” loop untuk menghitung nilai interpolasi.
* “prod = coef[k]” inisialisasi 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 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.

1. **Testing Code**

Berikut ini adalah testing code yang digunakan untuk menapilkan 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 diguanakan.
* “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 ynag 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 memangil function test agar dapat berjalan.

1. **Hasil Running**





