UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ

DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEPEL

COORDENAÇÃO DE ENGENHARIA ELÉTRICA - COELE

GABRIEL AUGUSTO SILVA BATISTA

TRABALHO 5

São João del-Rei – MG

Outubro de 2023

O código é uma implementação dos métodos de interpolação polinomial de Lagrange e Newton em Python. Ele usa a biblioteca NumPy para operações matemáticas e a biblioteca Matplotlib para plotar os gráficos.

Importação de Bibliotecas: O código começa importando as bibliotecas necessárias: NumPy e Matplotlib. NumPy é uma biblioteca para a linguagem Python, adicionando suporte a grandes matrizes e matrizes multidimensionais, juntamente com uma grande coleção de funções matemáticas de alto nível. Matplotlib é uma biblioteca para criar visualizações estáticas, animadas e interativas em Python.

Funções Auxiliares: Em seguida, o código define duas funções auxiliares: coef_newton e eval_newton.

- A função coef_newton calcula os coeficientes do polinômio interpolador de Newton. Ela cria uma matriz de zeros com dimensões n x n (onde n é o número de pontos dados) e preenche a primeira coluna com os valores Y. Em seguida, ela preenche o restante da matriz usando a fórmula das diferenças divididas de Newton. A função retorna a primeira linha da matriz, que contém os coeficientes do polinômio interpolador.
- A função eval_newton avalia o polinômio interpolador de Newton em um ponto específico. Ela começa definindo o polinômio como o último coeficiente e então entra em um loop onde multiplica o polinômio atual pelo termo (x - X[n-k]) e adiciona o próximo coeficiente. O resultado é o valor do polinômio no ponto x.

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

# Função para calcular os coeficientes do polinômio interpolador de Newton
def coef_newton(X, Y):
    n = len(X)
    coef = np.zeros([n, n]) # matriz de coeficientes
    coef[:,0] = Y # primeira coluna é Y

    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[0, :] # retorna a primeira linha

# Função para calcular o valor do polinômio interpolador de Newton em um ponto
def eval_newton(coef, X, x):
    n = len(X) - 1
    p = coef[n]
    for k in range(1,n+1):
        p = coef[n-k] + (x - X[n-k])*p
    return p
```

Dados: O código define três conjuntos de dados, cada um contendo arrays NumPy de valores X e Y.

```python
# Dados
conjuntos = [
    {"X": np.array([0, 2, 4, 6, 8, 10]), "Y": np.array([0.9, 2, 2.8, 3.1, 5.9, 6])},
    {"X": np.array([0, 2, 4, 6, 8]), "Y": np.array([1, 9.389, 58.598, 409.429, 2988.958])},
    {"X": np.array([0, 2, 4, 6, 8, 10, 12, 14, 16]), "Y": np.array([1, 7, 21, 22, 34, 34.5, 35, 64.5, 65])}
]
```

Interpolação: Para cada conjunto de dados, o código realiza as seguintes operações:

1. **Interpolação de Lagrange**: O código calcula o polinômio interpolador de Lagrange usando a função np.polyfit do NumPy. Esta função ajusta um polinômio de grau n-1 aos pontos dados (onde n é o número de pontos), no sentido dos mínimos quadrados. O resultado é um objeto np.poly1d, que representa um polinômio unidimensional. O código então avalia este polinômio no ponto X=5.2 usando a chamada ao método do objeto np.poly1d. Finalmente, ele imprime os coeficientes do polinômio.
2. **Interpolação de Newton**: O código calcula os coeficientes do polinômio interpolador de Newton usando a função coef_newton definida anteriormente. Ele então avalia este polinômio no ponto X=5.2 usando a função eval_newton. Finalmente, ele imprime os coeficientes do polinômio.

```python
for i, conjunto in enumerate(conjuntos):
    X = conjunto["X"]
    Y = conjunto["Y"]

    # Polinômio interpolador de Lagrange
    P_Lagrange = np.poly1d(np.polyfit(X,Y,len(X)-1))

    # Valor da função f(X) para X=5.2
    f_5_2_Lagrange = P_Lagrange(5.2)

    # Coeficientes do polinômio interpolador na forma canônica
    coef_Lagrange = P_Lagrange.coefficients

    print(f"Conjunto {i+1} - Método de Lagrange:")
    print(f"f(5.2) = {f_5_2_Lagrange}")
    print(f"Coeficientes: {coef_Lagrange}")

    # Polinômio interpolador de Newton
    coef_Newton = coef_newton(X,Y)

    # Valor da função f(X) para X=5.2
    f_5_2_Newton = eval_newton(coef_Newton,X,5.2)

    print(f"\nConjunto {i+1} - Método de Newton:")
    print(f"f(5.2) = {f_5_2_Newton}")
    print(f"Coeficientes: {coef_Newton}\n")
```

Plotagem: Finalmente, o código plota os pontos de dados originais e as funções polinomiais interpoladoras de Lagrange e Newton usando a biblioteca Matplotlib. Ele cria uma figura, plota os pontos originais em vermelho, cria um array linearmente espaçado para o eixo X (para uma representação mais suave da função), avalia os polinômios nos pontos deste array e plota as funções resultantes.

```python
    # Plot do gráfico
    plt.figure()

    plt.plot(X,Y,'ro') # pontos originais em destaque

    X_plot = np.linspace(np.min(X),np.max(X),500)

    Y_plot_Lagrange = P_Lagrange(X_plot)

    plt.plot(X_plot,Y_plot_Lagrange,'b-', label='Lagrange') # função polinomial de Lagrange

    Y_plot_Newton = [eval_newton(coef_Newton,X,x) for x in X_plot]

    plt.plot(X_plot,Y_plot_Newton,'g-', label='Newton') # função polinomial de Newton

    plt.title(f"Conjunto {i+1} - Métodos de Lagrange e Newton")

    plt.legend()

    plt.show()
```

Saída do programa:

```
Conjunto 1 - Método de Lagrange:
f(5.2) = 2.786892799999994
Coeficientes: [-0.00296875  0.06770833 -0.51979167  1.54166667 -0.94833333  0.9       ]

Conjunto 1 - Método de Newton:
f(5.2) = 2.7868928
Coeficientes: [ 0.9         0.55       -0.0375     -0.00416667  0.00833333 -0.00296875]

Conjunto 2 - Método de Lagrange:
f(5.2) = 146.8936336000022
Coeficientes: [   4.33925521  -46.6376875   163.42947917 -170.82775      1.        ]

Conjunto 2 - Método de Newton:
f(5.2) = 146.89363360000002
Coeficientes: [1.         4.1945     5.1025     5.433375   4.33925521]

Conjunto 3 - Método de Lagrange:
f(5.2) = 19.81004902407744
Coeficientes: [ 2.69329737e-05 -1.92522321e-03  5.59353299e-02 -8.50000000e-01
  7.23161892e+00 -3.40510417e+01  8.07393353e+01 -6.81976190e+01
  1.00000000e+00]

Conjunto 3 - Método de Newton:
f(5.2) = 19.810049024
Coeficientes: [ 1.00000000e+00  3.00000000e+00  1.00000000e+00 -4.37500000e-01
  1.17187500e-01 -2.38281250e-02  3.73263889e-03 -4.16976687e-04
  2.69329737e-05]
```