Rotinas Desenvolvidas – P1T3 ALC

Matheus Lomba de Rezende Conde – DRE: 117085216

As rotinas desenvolvidas para o trabalho estão divididas em 4 agrupamentos: Main, Inputs, Métodos e Suporte.

Main: agrupamento principal por onde as funções são chamadas e o programa é executado.

```
from inputs import pontox, coordenadas, paresN, icod
from metodos import interpolacao, regressao
from sup import finaliza

#Inputs do programa
var_paresN = paresN()
var_icod = icod()
x = pontox()
listapontos = coordenadas(var_paresN)

if var_icod == 1:
    interpolacao(x, listapontos, var_paresN)
elif var_icod == 2:
    regressao(x, listapontos, var_paresN)

finaliza()
```

Inputs: agrupamento onde ficam as rotinas que requisitam a entrada de dados fornecidos pelo usuário.

```
from sup import finaliza

#---------------------------------------------------------------------

def pontox():
    print('=' * 30)
    while True:
        try:
            x = float(input('O programa deverá calcular o y de qual
ponto x? '))
            break
        except ValueError:
                print('Favor inserir um número (float).')
    return x

#---------------------------------------------------------------------

def coordenadas(paresN):

    listapontos = [[]]
    print('=' * 30)

    for i in range(0, paresN):
```

```
        for j in range(0, 2):
            try:
                if j == 0:
                    listapontos[i].append(float(input(f'{i + 1}°
Coordenada: x = ')))
                else:
                    listapontos[i].append(float(input(f'{i + 1}°
Coordenada: y = ')))
            except ValueError:
                print("Favor rodar o programa novamente e inserir um
número válido (float).")
                finaliza()
                exit(0)
        if i < paresN - 1:
            listapontos.append(list())

    # Printar Matriz
    print('=' * 30)
    print('Lista de Coordenadas:')
    for i in range(len(listapontos)):
        print(listapontos[i])

    return listapontos

#---------------------------------------------------------------------

def paresN():
    print('=' * 30)
    while True:
        try:
            var_paresN = int(input('Qual é o número de pares de pontos
(x,y)? '))
            if var_paresN < 1:
                print('Favor inserir um número inteiro positivo e não
nulo.')
            else:
                break
        except ValueError:
                print('Favor inserir um número inteiro.')
    return var_paresN


def icod():
    print('=' * 30)
    print('1 = Interpolação\n2 = Regressão')
    while True:
        try:
            var_icod = int(input('Qual método será utilizado? '))
            if var_icod != 1 and var_icod != 2:
                print('Favor inserir um valor válido para o método.')
            else:
                break
        except ValueError:
            print('Favor inserir um valor válido para o método (1 ou
2)')
    return var_icod
```

Métodos: agrupamento onde estão armazenados os métodos requisitados no trabalho para a solução de sistemas lineares.

```
import numpy as np
import math
#---------------------------------------------------------------------

def interpolacao(x, listapontos, paresN):

    phi = 1
    y = 0
    for i in range(0, paresN):
        for j in range(0, paresN):
           if j != i:
               phi *= (x - listapontos[j][0]) / (listapontos[i][0] -
listapontos[j][0])
        y += phi * listapontos[i][1]
        phi = 1

    print(f'y = {y}')

    return 0
#---------------------------------------------------------------------

def regressao(x, listapontos, paresN):

    matrizP = [[]]
    matrizY = []

    for i in range(0, paresN):
        for j in range(0, 2):
            if j == 0:
                matrizP[i].append(1/math.pow(math.e,
listapontos[i][0]))
            else:
                matrizP[i].append(math.log(listapontos[i][0], math.e))
        matrizY.append(listapontos[i][1])
        if i < paresN - 1:
            matrizP.append(list())

    matrizPt = np.transpose(matrizP)
    matrizA = np.dot(matrizPt, matrizP)
    matrizC = np.dot(matrizPt, matrizY)
    MatrizAinv = np.linalg.inv(matrizA)
    matrizB = np.dot(MatrizAinv, matrizC)

    y = (matrizB[0] / math.e ** x) + matrizB[1] * math.log(x, math.e)
    print(f'y = {y}')

    return 0
```

Suporte: agrupamento onde estão armazenadas funções de suporte, que auxiliam as funções principais a realizarem seus processos corretamente.

```
import numpy as np

def finaliza():
    # Impede que o terminal feche automaticamente assim que o programa
```

```
finaliza.
    input('Pressione qualquer tecla para finalizar...')
```