NUMERICAL DIFFERENTIATION USING NEWTON BACKWARD DIFFERENCE METHOD IN PYTHON

SUBMITTED BY: UMANG KANCHAN

SUBMITTED TO: Dr. AYESHA CHOUDHARY

ASSIGNMENT 7B

NUMERICAL METHODS (CS-406)

2023

import numpy as np

from scipy.interpolate import CubicSpline

def newton\_backward\_diff(*x*, *y*, *h*, *x0*):

    n = len(x)

    dfdx = np.zeros(n)

    for i in range(n-1, 0, -1):

        if i == n-1:

            # use backward difference formula for the last point

            dfdx[i] = (3\*y[i] - 4\*y[i-1] + y[i-2]) / (2\*h)

        elif i == n-2:

            # use backward difference formula for the second to last point

            dfdx[i] = (y[i] - 4\*y[i-1] + 3\*y[i-2]) / (2\*h)

        else:

            # use central difference formula for all other points

            dfdx[i] = (y[i+1] - y[i-1]) / (2 \* h)

    # interpolate the derivative value at x0 using a cubic spline

    cs = CubicSpline(x, dfdx)

    dfdx0 = cs(x0, 1)

    return dfdx0

# take x and y coordinates as input from the user

n = *int*(input("Enter the number of data points: "))

x = np.zeros(n)

y = np.zeros(n)

for i in range(n):

    x[i] = *float*(input("Enter x[{}]: ".format(i)))

    y[i] = *float*(input("Enter y[{}]: ".format(i)))

# compute the derivative using Newton's backward difference formula

h = *float*(input("Enter the step size h: "))

x0 = *float*(input("Enter the value of x at which to compute the derivative: "))

dfdx0 = newton\_backward\_diff(x, y, h, x0)

# print the derivative value at x0

print("f'({}) = {}".format(x0, dfdx0))