Write a program and documentation for Runge Kutta's 8th order method with 100 iterations, print only 10

EXPLANATION:

The **"def runge_kutta_8th_order(f, x0, xn, N, y0):"** is Python function that implements the Runge-Kutta 8th order method, which is a numerical technique for solving ordinary differential equations. Here is an explanation of the function parameters:

• **f** is a function that will be passed into the main function as the differential equation, such as f(x, y) = x – y

• **x0** and **xn** are the initial and final values of the independent variable

• **N** is the number of steps to divide the interval **[x0, xn]** into, such as N = 100

• **y0** is the initial value of the dependent variable

h = (xn - x0) / N : This line calculates the step size h by dividing the interval [x0, xn] into N equal subintervals.

t = x0 : This line assigns the initial value of t to be x0, which is the lower bound of the interval.

w = y0 : This line assigns the initial value of w to be y0, which is the initial condition of the differential equation. w represents the approximate solution at each step.

• It creates two empty lists **xVal** and **yVal** to store the values of x and y for plotting

• It uses a loop to iterate over the subintervals from **x0** to **xn** with a step size of **h = (xn - x0) / N.**

• It calculates eight intermediate values **k1** to **k13** using the function **f** and the previous values of **t** and **w**.

• It updates the value of **w** using a weighted average of **k1** to **k13**.

• It updates the value of **t** by adding **h** to it.

• If **i** is less than or equal to 10, it prints the values of **t** and **w** rounded to four decimal places, separated by a tab

• It appends the values of **t** and **w** to the lists **xVal** and **yVal**, respectively

• After the loop ends, it creates a pandas data frame data with two columns: X and Y, containing the values from **xVal** and **yVal**

• It prints the result of plotting the data frame with X as the x-axis and Y as the y-axis, using the plot method of pandas

• It returns the final value of w as the approximate solution of the differential equation at **xn**.

Example usage

def ex1(x, y):

return x - y

x0 = 0

xn = 1

N = 100

y0 = 0.5

runge_kutta_8th_order(ex1, x0, xn, N, y0)

Explanation:

• The function ex1 defines the ODE as y' = x - y, where y' is the derivative of y with respect to x.

• The variables x0 and xn are the lower and upper bounds of the interval where the solution is sought, respectively. In this case, x0 = 0 and xn = 1.

• The variable N is the number of subintervals used to divide the interval [x0, xn]. The smaller the subinterval, the more accurate the solution. In this case, N = 100.

• The variable y0 is the initial value of the solution, i.e., y(x0) = y0. In this case, y0 = 0.5.

• The function runge_kutta_8th_order takes the ODE function, the interval bounds, the number of subintervals, and the initial value as arguments, and returns the approximate value of the solution at xn using the Runge-Kutta method of 8th order.

THE IMPLEMENTATION

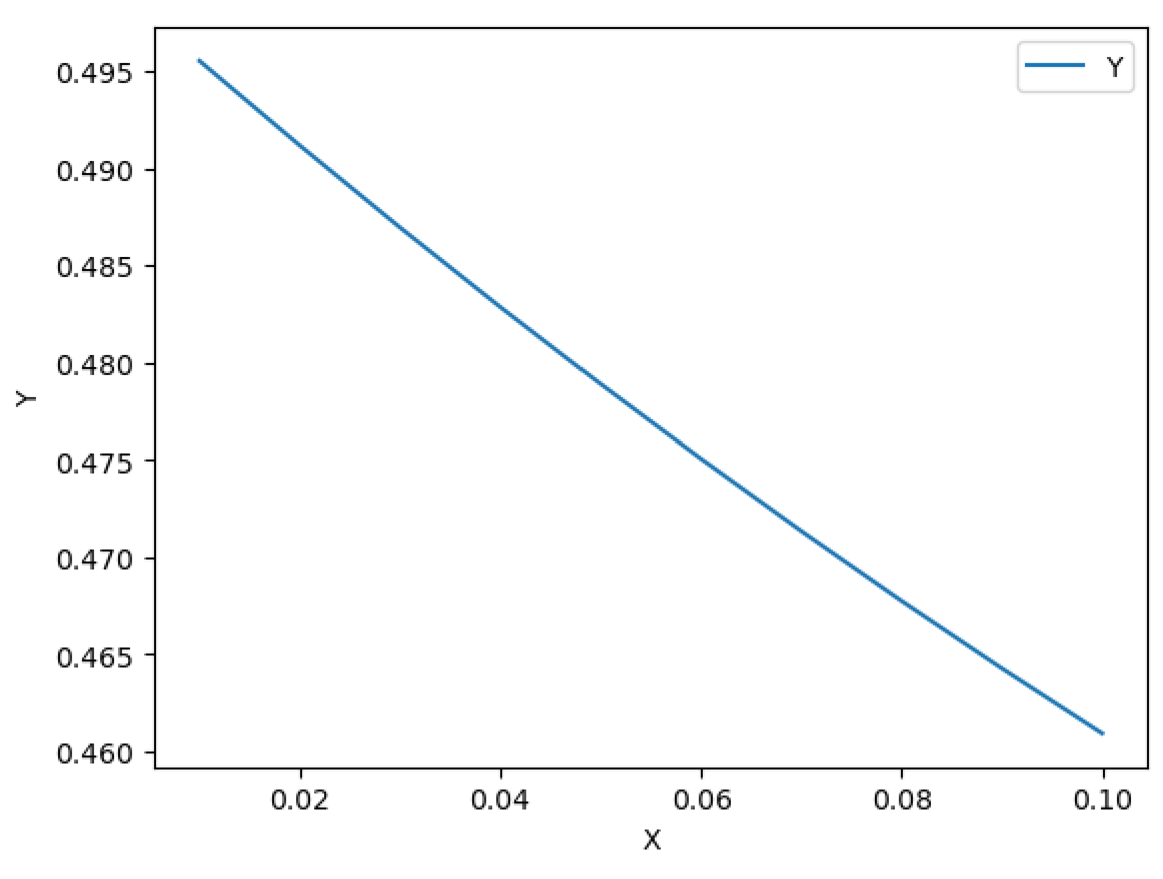
In [1]:

```python
import pandas as pd

def runge_kutta_8th_order(f, x0, xn, N, y0):
    h = (xn - x0) / N
    t = x0
    w = y0
    print("X \t\tY ")

    xVal = []
    yVal = []

    for i in range(1, N + 1):
        k1 = f(t, w)

        k2 = f(t + (2/27)*h, w + (2/27)*h*k1)

        k3 = f(t + (1/9)*h, w + ((1/36)*h)*(k1 + (3*k2)))

        k4 = f(t +(1/6)*h, w + ((1/24)*h)*(k1 + (3*k2)))

        k5 = f(t +(5/12)*h, w + ((1/48)*h)*((20*k1) - (75*k3) + (75*k4)))

        k6 = f(t + (1/2)*h, w + ((1/20)*h)*(k1 + (5*k4) + (4*k5)))

        k7 = f(t + (5/6)*h, w + ((1/108)*h)*((-25*k1) + (125*k4) - (260*k5) + (250*k6)))

        k8 = f(t + (1/6)*h, w + h*(((31/300)*k1) - ((61/225)*k5) - ((2/9)*k6) + ((13/900)*k7)))

        k9 = f(t + (2/3)*h, w + h*((2*k1) -((53/6)*k4) + ((704/45)*k5) - ((107/9)*k6)
                + ((67/90)*k7)+ (3*k8)))

        k10 = f(t + (1/3)*h, w + h*((-91/108)*k1) + ((23/108)*k4) - ((976/135)*k5)
                + ((311/54)*k6) - ((19/60)*k7) + ((17/6)*k8) - ((1/12)*k9))

        k11 = f(t + h, w + h*((2383/4100)*k1) - ((341/164)*k4) + ((4496/1025)*k5) - ((301/82)*k6)
                + ((2133/4100)*k7) + ((45/82)*k8) + ((45/164)*k9) + ((18/41)*k10))

        k12 = f(t, w + h*((3/205)*k1) - ((6/41)*k6) - ((3/205)*k7) - ((3/41)*k8) - ((3/41)*k9)
                + ((6/41)*k10))

        k13 = f(t + h, w + h*((-1777/4100)*k1) - ((341/164)*k4) + ((4496/1025)*k5) - ((289/82)*k6)
                + ((2193/4100)*k7) + ((51/82)*k8) + ((33/164)*k9) + ((12/41)*k10) + k12)

        w = w + h*(((272*k6) + (216*k7) + (216*k8) + (27*k9) + (27*k10) + (41*k12) + (41*k13))/840)
        t = x0 + i * h
        if i <= 10:
            print(t, "\t\t", round(w, 4))
            xVal.append(t)
            yVal.append(w)

    #pandas data frame
    data = pd.DataFrame({'X': xVal,'Y': yVal})

    #graph definition
    print((data.set_index('X').plot(y='Y',xlabel= 'X',ylabel = 'Y')))

    return w

# Example usage
def ex1(x, y):
    return x - y

x0 = 0
xn = 1
N = 100
y0 = 0.5
runge_kutta_8th_order(ex1, x0, xn, N, y0)
```

```
X               Y
0.01             0.4955
0.02             0.4912
0.03             0.4869
0.04             0.4828
0.05             0.4789
0.06             0.475
0.07             0.4713
0.08             0.4677
0.09             0.4643
0.1              0.4609
Axes(0.125,0.11;0.775x0.77)
```

Out[1]:

```
0.5448493636862826
```



Output:

The code prints a table of X and Y values, where X is the independent variable t and Y is the approximate solution w of the ODE at each step. The code also returns the final value of w as the output of the function.

The graph is a linear plot that shows the values of y as a function of x for the first 10 steps of the runge_kutta_8th_order function.

The graph shows that y decreases linearly as x increases from 0 to 0.1. The output value above the graph is the final value of y at x = 1, which is approximately 0.5448. The graph is displayed using the plot method of pandas, a Python library for data analysis.

REFERENCES:

Kolar Amir Taher, 2020: Comparison of numerical methods for solving a system of ordinary differential equations: accuracy, stability and efficiency

GROUP MEMBERS

TALABI DAMILARE HANNAH 21/7954

OLAIYA JOLAYAEMI 22/9796

DIVINE UKPAI 21/8200

FATOKUN DAMILOLA 21/8666

DAVID EMMANUEL ILERIOLUWA 21/9187