Numerical Differentiation Using Difference Table.

Aim:- Implement numerical differentiation using a difference table in python.

Provide a fn $y = f(x)$ & a set of data points. Compute the numerical derivative at specific points using the forward difference method.

Discuss the sensitivity of numerical differentiation to the choice of step size.

- Present physics problems like compute the velocity or acceleration of a particle based on position data.

Theory :-

Numerical Differentiation.

Consider a set of tabulated values (x_i, y_i), $i = 1, 2, \ldots n$ of a fn. The process of computing the derivatives or derivatives of that fn at some value of x from the given set of value is called Numerical Differentiation.

If the value of x are equispaced & the derivative is required near the begining of the table of values (x_i, y_i) we employ Gregory - Newton - Forward Interpolation Formula:

It is written as,

$$y = y_0 + P\Delta y_0 + \frac{P(P-1)}{2!}\Delta^2 y_0 + \frac{P(P-1)(P-2)}{3!}\Delta^2 y_0$$

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$
x_0	y_0	$\Delta y_0 = y_1 - y_0$	$\Delta^2 y_0 = \Delta y_1 - \Delta y_0$	$\Delta^3 y_0 = \Delta^2 y_1 - \Delta^2 y_0$
x_1	y_1	$\Delta y_1 = y_2 - y_1$	$\Delta^2 y_1 = \Delta y_2 - \Delta y_1$	
x_2	y_2	$\Delta y_2 = y_3 - y_2$		
x_3	y_3			

This is a forward difference Table.

Principle :- Derivative of a function can be calculated using Newton's forward difference interpolation formula.

$$y = y_0 + P\Delta y_0 + \frac{P(P-1)\Delta^2 y_0}{2!} + \frac{P(P-1)(P-2)\Delta^2 y_0}{3!} + \cdots$$

where $P = \dfrac{x - x_0}{h}$

x = test value for which the derivative are to be determined

x_0 & y_0 = Data at the beginning of the tabulated set (x_1, y_1) of the function.

h = Successive difference between x_1's.

$\Delta y_0, \Delta^2 y_0$ = are the first & second order forward differences of y_i

$$\frac{dy}{dx} = \frac{1}{h}\left[\Delta y_0 + \frac{(2P+1)}{2!}\Delta^2 y_0 + \frac{(3P^2 - 6P + 2)}{3!}\Delta^3 y_0 + \cdots\right] \quad (2)$$

$$\frac{d^2y}{dx^2} = \frac{1}{h^2}\left[\Delta^2 y_0 + (P-1)\Delta^3 y_0 + \cdots\right] \quad (3)$$

using equation (2) & (3) we can find the first & second derivative of the function of the desired value of x

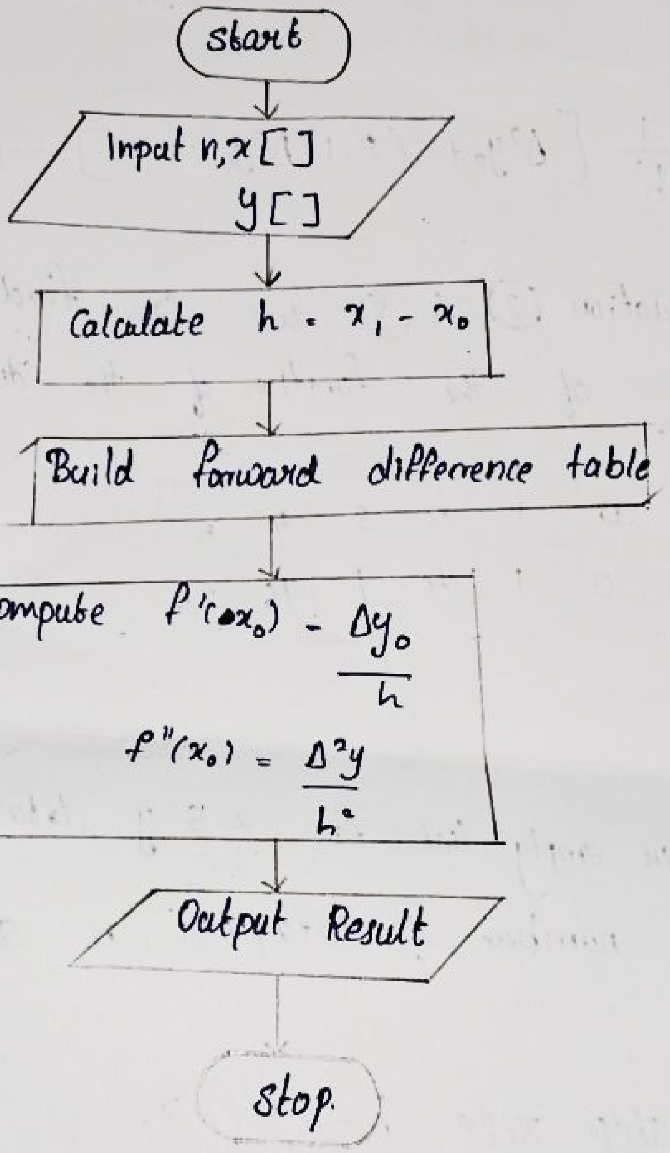
Principle :-

x	0	1	2	3	4	5
$y = f(x)$	0	1	4	9	16	25

Algorith.

1. Create an empty list for x & y data.
2. Input the number of data point, n & corresponding x, y values
3. Calculate step size $h = x_1 - x_0$
4. Create a forward difference table from y - values
5. Compute $f'(x_0) \approx \Delta y_0 / h$ & second derivative $f''(x_0) = \frac{\Delta^2 y_0}{h^2}$
6. Output result.

Flow chart



Procedure :- Code.

```
import math
x = [0, 1, 2, 3, 4, 5]
y = [0, 1, 4, 9, 16, 25]
n = len(x)
h = x[1] - x[0]
```

```
def forward_difference_table(x,y):
  n= len(y)
  diff_table =[y.copy()]
for i in range (1,n):
     row =[]
     for j in range (n-i):
        delta= diff_table[i-1][j+1]-diff_table[i-1][j]
~~[j+1]-diff_table[i-1][j]~~
       row.append(delta)
     diff_table.append(row)
return diff_table
Print("forward difference table:")
diff_table=forward_difference_table(x,y)
for row in diff_table:
     Print (row)
x_test=1.5
P=(x_test-x[0])/h
dy=(1/h)*(diff_table[1][0]+((2*P-1)/
math.factorial(2))*diff_table[2]
[0]+((3*P**2-6*P+2)/
math.factorial(3))*diff_table[3][0])
d2y=(1/h**2)*(diff_table[2][0]+
(P-1)*diff_table[3][0])
Print(f"At x={x_test}:")
Print(f"First derivative(dy/dx)≈{dy}")
Print(f"Second derivative (d²y/dx²)≈{d2y}")
```

⇒ output → Forward difference table : [0, 1, 4, 9, 16, 25]

[1, 3, 5, 7, 9]

[2, 2, 2, 2]

[0, 0, 0]

[0, 0]

[0]

At $x = 1.5$

First Derivative $(dy/dx) \approx 3.0$

Second Derivative $(d^2y/dx^2) \approx 2.0$