NUMERICAL DIFFERENTIATION USING PYTHON

AIM: To learn how to numerically differentiate the functions.

THEORY:

The derivative of a function of a single variable at a chosen input value, when it exists, is the slope of the tangent line to the graph of the function at that point. The tangent line is the best linear approximation of the function near that input value. For this reason, the derivative is often described as the "instantaneous rate of change", the ratio of the instantaneous change in the dependent variable to that of the independent variable.

In order to numerically calculate the derivative of the given function we employ *Taylor series* expansion to derive finite-divided-difference approximation of derivatives.

The forward Taylor series expansion of function f(x) around point x_i [predicting value of f(x) at x_{i+1}].

$$fxi+1=fxi+f'xih+f''xi2h2+\ldots \quad \text{----------(1)}$$

where h=xi+1-xi

Which can be solved for

$$f'xi=fxi+1-fxih-f''xi2h+O(h2) \quad \text{----------(2)}$$

where O(h2)represents the remaining terms of the series.

To get the expression for f''xi, we expand the function f(x) (for point xi+2) around point x_i.

$$fxi+2=fxi+f'xi2h+f''xi2(2h)2+\ldots \quad \text{----------(3)}$$

Now equation-1 can be multiplied by 2 and subtracted from equation-3 to give

$$fxi+2-2fxi+1=-fxi+f''xih2+\ldots \quad \text{----------(4)}$$

Which can be solved for $f''xi=fxi+2-fxi+1+fxih2+\Omega(h)$ ----------(5)

This is inserted in equation-2 to get

$$f'xi=fxi+1-fxih-fxi+2-fxi+1+fxih2h+\zeta(h2) \quad \text{----------(6)}$$

Rearranging the terms in the above equation,

$$f'xi=-fxi+2+4fxi+1-3fxi2h+\psi(h2) \quad \text{----------(7)}$$

The first term in equation 7 can be used as first derivative of f(x) at point x_i with error $\psi(h2)$. For small h, $\psi(h2)$ term can be ignored. Therefore we can write

$$f'xi=-fxi+2+4fxi+1-3fxi2h$$

PYTHON CODES

Example:

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

def f(x):
   y=x**2
   return y
x=np.linspace(0,50, 500)
a=0 #minimum range
b=50#maximum range
n= 500#step size
h= (b-a)/n
i=0
derv=[]
while i<n-2:
   dervj=(-f(x[i+2])+4*f(x[i+1])-(3*f(x[i])))/(2*h)
   derv.append(dervj)
   i=i+1
print('differentiated value=',derv[-1])
plt.plot(x[0:498],derv)
plt.title('Result')
plt.xlabel('x')
```

```
plt.ylabel('derv')
plt.show()
```

1(a). For step size 500

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

def f(x):
   y=np.sqrt(x)*np.exp(-x)
   return y
x=np.linspace(0.2,2, 500)
a=0.2 #minimum range
b=2 #maximum range
n= 500#step size
h= (b-a)/n
i=0
derv=[]
while i<n-2:
   dervj=(-f(x[i+2])+4*f(x[i+1])-(3*f(x[i])))/(2*h)
   derv.append(dervj)
   i=i+1
print('differentiated value=',derv[-1])
plt.plot(x[0:498],derv)
plt.title('Result')
plt.xlabel('x')
plt.ylabel('derv')
plt.show()
```

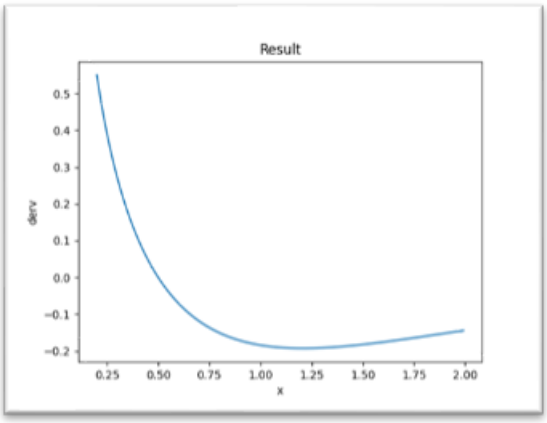
1(b). For step size 50

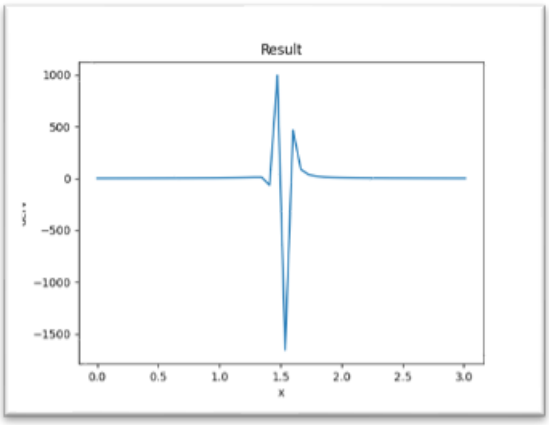
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
   y=(np.sin(x))**2/np.cos(x)
   return y
x=np.linspace(0,np.pi, 50)
a=0 #minimum range
b=np.pi#maximum range
n= 50#step size
h= (b-a)/n
i=0
derv=[]
while i<n-2:
   dervj=(-f(x[i+2])+4*f(x[i+1])-(3*f(x[i])))/(2*h)
   derv.append(dervj)
   i=i+1
print('differentiated value=',derv[-1])
plt.plot(x[0:500],derv)
plt.title('Result')
plt.xlabel('x')
plt.ylabel('derv')
plt.show()
```

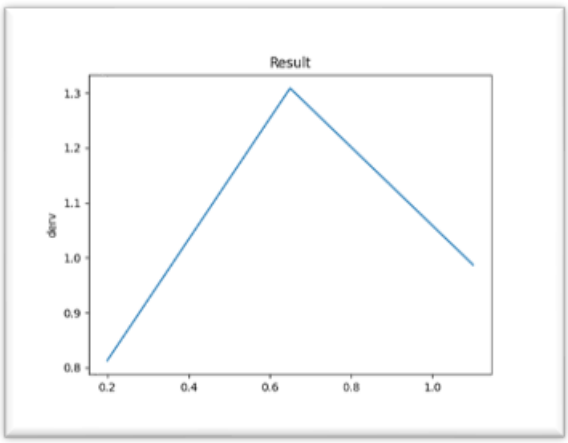
1(c). For step size 5

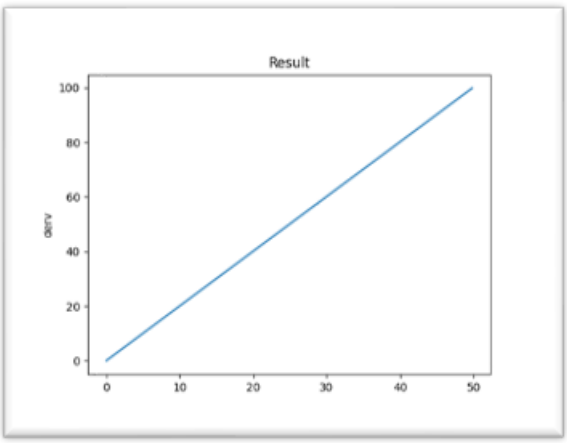
```
import numpy as np
import matplotlib.pyplot as plt
```

```
def f(x):
    y=x**2/np.sqrt(x**3+1)
    return y
x=np.linspace(0.2,2, 5)
a=0.2 #minimum range
b=2 #maximum range
n= 5#step size
h= (b-a)/n
i=0
derv=[]
while i<n-2:
    dervj=(-f(x[i+2])+4*f(x[i+1])-(3*f(x[i])))/(2*h)
    derv.append(dervj)
    i=i+1
print('differentiated value=',derv[-1])
plt.plot(x[0:3],derv)
plt.title('Result')
plt.xlabel('x')
plt.ylabel('derv')
plt.show()
```









CONCLUSION: Numerical differentiation is a valuable technique for estimating derivatives when analytical solutions are unavailable or impractical. Python, with libraries like NumPy, provides a convenient environment for implementing and experimenting with various numerical differentiation methods. When using numerical differentiation, it's important to choose appropriate methods and step sizes to balance accuracy and stability for our specific problem.