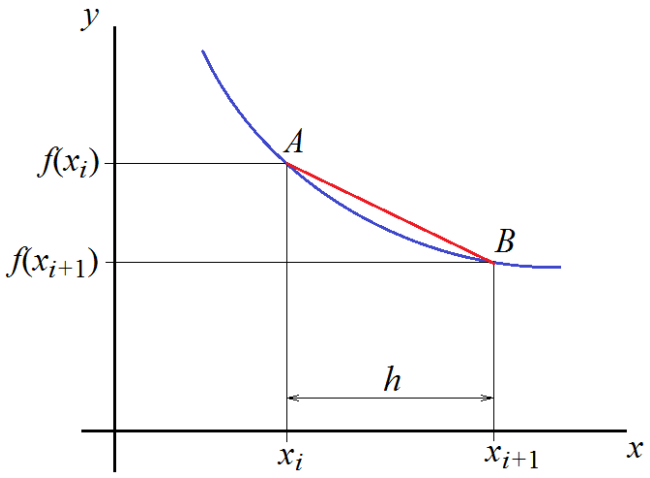
IV. Numerical Differentiation

Differentiation has numerous applications in science and engineering since most relationships of physical formulas and data are differentials with respect to position and/or time. Although most mathematical equations and formulas can be differentiated analytically, many curves and data sets obtained from experiments, natural observations, mechanically generated paths and other empirical methods require numerical differentiation techniques to perform accurate analysis.

The Finite Differences Approximations

The finite differences method is used in finding the derivatives of a formula or data by taking differences of $y(x)$ values with respect to differences of x. In case of equal differences of x, the difference Δx is commonly designated as h, which is known as the *step size*, as shown in the figure.



The simplest form of differentiation can be obtained by finding the slope of the line AB which is approximately equal to the slope of tangent to the curve at middle between x_i and x_{i+1}. The smaller step size is used, the more accurate slope can be obtained. The sequence of the slope values will yield the approximate first derivative of the given data.

Theoretically, the finite differences approximation is derived from Taylor's series expansion of $f(x)$. The expansion of $f(x_{i+1})$ can be written as

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + \cdots$$

So,

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{f''(x_i)}{2!}h - \frac{f'''(x_i)}{3!}h^2 + \cdots$$

By omitting the terms containing the second and higher derivatives, the difference formula becomes

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + O(h)$$

This formula is known as *forward difference* because the second point after x_i is selected in the positive direction of x-axis. The term $O(h)$ indicates to the error resulting from the truncation made to obtain the difference. More derivations are performed to improve the accuracy of the method.

The following formulas represent the differentiations by using forward, central and backward finite differences up to the fourth derivative.

a) Forward finite differences. Error: $O(h)$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$$
$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2}$$
$$f'''(x_i) = \frac{f(x_{i+3}) - 3f(x_{i+2}) + 3f(x_{i+1}) - f(x_i)}{h^3}$$
$$f''''(x_i) = \frac{f(x_{i+4}) - 4f(x_{i+3}) + 6f(x_{i+2}) - 4f(x_{i+1}) + f(x_i)}{h^4}$$

b) Central finite differences. Error: $O(h^2)$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$
$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$$
$$f'''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + 2f(x_{i-1}) - f(x_{i-2})}{2h^3}$$
$$f''''(x_i) = \frac{f(x_{i+2}) - 4f(x_{i+1}) + 6f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{h^4}$$

c) Backward finite differences. Error: $O(h)$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h}$$
$$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2}$$
$$f'''(x_i) = \frac{f(x_i) - 3f(x_{i-1}) + 3f(x_{i-2}) - f(x_{i-3})}{h^3}$$
$$f''''(x_i) = \frac{f(x_i) - 4f(x_{i-1}) + 6f(x_{i-2}) - 4f(x_{i-3}) + f(x_{i-4})}{h^4}$$

As it can simply be noticed form the formulas, the central differences consider the same number of points to the left and right of x_i, while in backward differences the points are taken to the left of x_i. This affects the results significantly.

Example: Find the first and second derivatives of the following polynomial at $x = 0.1$

$$f(x) = 0.1x^5 - 0.2x^3 + 0.1x - 0.2$$

Solution: the analytical solution

$$f'(x) = 0.5x^4 - 0.6x^2 + 0.1$$

$$f''(x) = 2.0x^3 - 1.2x$$

Substituting x = 0.1 yields

$$f'(0.1) = 0.09405 \text{ and } f''(0.1) = -0.118$$

Let's start the coding with the forward difference approximation, take h = 0.05.

Step 1: Define the given polynomial as an inline function to calculate the $f(x)$ values, the step size and given x value.

```
f = lambda x: 0.1*x**5 - 0.2*x**3 + 0.1*x - 0.2
h = 0.05
x = 0.1
```

Step2: Since the step size is constant, it can be used in evaluating $f(x)$ terms by setting $x_i \rightarrow$ `x`, $x_{i+1} \rightarrow$ `x+h`, $x_{i+2} \rightarrow$ `x+2*h` and so on. The first two differential formulas of forward differences become

```
dff1 = (f(x+h) - f(x))/h
dff2 = (f(x+2*h) - 2*f(x+h) + f(x))/h**2
print('f\'(%f) = %f'%(x,dff1))
print('f\'\'(%f) = %f'%(x,dff2))
```

At this stage the program can run and yield the following output

```
f'(0.100000) = 0.090632
f''(0.100000) = -0.172875
```

Now, let's add the differentials by using the central and backward differences and compare the results. Thus, the final code will be

```
f = lambda x: 0.1*x**5 - 0.2*x**3 + 0.1*x - 0.2
h = 0.05
x = 0.1

# Forward differences approximation
dff1 = (f(x+h) - f(x))/h
dff2 = (f(x+2*h) - 2*f(x+h) + f(x))/h**2
print('Solution by forward differences:')
print('f\'(%f) = %f'%(x,dff1))
print('f\'\'(%f) = %f'%(x,dff2))

# Central differences approximation
dfc1 = (f(x+h) - f(x-h))/(2*h)
dfc2 = (f(x+h) - 2*f(x) + f(x-h))/h**2
print('Solution by central differences:')
print('f\'(%f) = %f'%(x,dfc1))
print('f\'\'(%f) = %f'%(x,dfc2))
```

```
# Backward differences approximation
dfb1 = (f(x) - f(x-h))/h
dfb2 = (f(x) - 2*f(x-h) + f(x-2*h))/h**2
print('Solution by backward differences:')
print('f\'(%f) = %f'%(x,dfb1))
print('f\'\'(%f) = %f'%(x,dfb2))
```

The output:

```
Solution by forward differences:
f'(0.100000) = 0.090632
f''(0.100000) = -0.172875

Solution by central differences:
f'(0.100000) = 0.093576
f''(0.100000) = -0.117750

Solution by backward differences:
f'(0.100000) = 0.096519
f''(0.100000) = -0.059625
```

By comparing values obtained from the program with the analytical solution, we notice that the central differences yielded the most accurate solution in the first derivative (0.5% error) while forward and backward differences were less accurate (3.6% and 2.6%, respectively). The similar conclusion can be drawn for the second derivatives too.

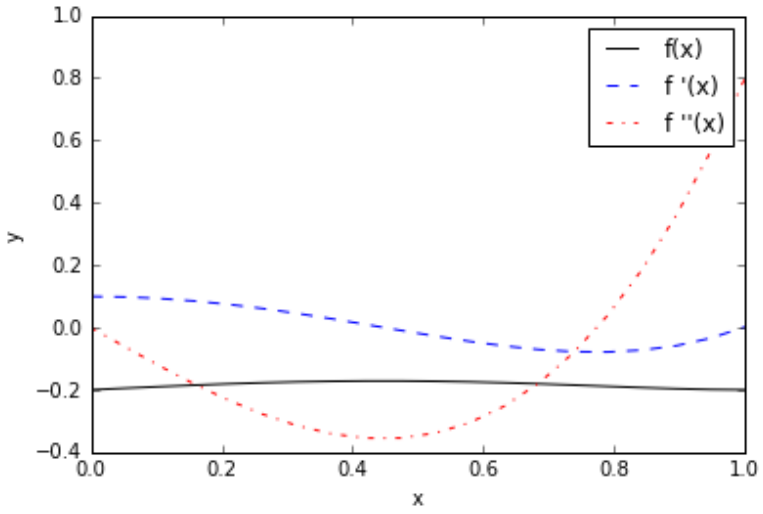
In order to plot derivative curves over a given domain, the difference formulas can be applied on an array of x values within the domain. For example, if the curves of the given function and its derivatives for [0, 1] by using central differences is required the program can be written as

```
import numpy as np                  # array module
import matplotlib.pyplot as plt     # plot module
f = lambda x: 0.1*x**5 - 0.2*x**3 + 0.1*x - 0.2
h = 0.05
x = np.linspace(0,1,11)       # array of 11 elements from 0 to 1

# Central differences approximation
dfc1 = (f(x+h) - f(x-h))/(2*h)
dfc2 = (f(x+h) - 2*f(x) + f(x-h)) / h**2

# Plotting results
plt.plot(x,f(x),'-k', x,dfc1,'--b', x,dfc2,'-.r')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(['f(x)','f \'(x)','f \'\'(x)'])
plt.show()
```

The output graph



When the function $f(x)$ is given, the increment of x values are not necessarily equal to differentiation step size. In the program above, the x values range from 0 to 1 by increment of 0.1 while the step size, h, is 0.05. The x increments determine the number of points that the differentials are computed at, on the other hand, the step size affects the accuracy of the derivatives at each point.

Instead of a function $f(x)$, if a data set is given as (x, y) points, interpolation or curve fitting can be applied to generate a polynomial that fits the given points. Then, the resulting polynomial is differentiated analytically or numerically to obtain the required derivatives.

Differentiation in SciPy

The numerical differentiation function `derivative()` from `scipy.misc` computes the n-th derivative of a given function at `x0` by using a central difference formula with spacing `dx`. The solution of the example will as following

```
>>> from scipy.misc import derivative
>>> f = lambda x: 0.1*x**5 - 0.2*x**3 + 0.1*x - 0.2
>>> y = derivative(f, 0.1, 0.05)    # x0=0.1, dx=0.05, n=1 (default)
>>> print(y)
0.093575625
>>> y2 = derivative(f, 0.1, 0.05,2) # n=2 for second order derivative
>>> print(y2)
-0.11775
```

These value are equal to those obtained by using the central differences code.

For more information about `derivative()`:

https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.misc.derivative.html