5 Numerical differentiation and rounding errors.

So far we have demonstrated the use of python for numerical integration and plotting functions.

In this notebook you will learn how to:

- carry out simple numerical differentation,
- the importance of selecting the best algorithm,
- make estimates of errors,
- recognise the perils of trying to get precision by brute force.

5.1 Numerical differentiation

NOTE. Although this notebook shows you the principles by which you can do numerical differentiation, it is something you should avoid if you can. Whenever, you can code an analytical differential you should do so.

5.1.1 Forward and backward difference methods

You may remember that when you were introduced to calculus you defined the differential of a function as,

$\frac{df(x)}{dx} = \lim_{\delta x \to 0} \frac{f(x+\delta x) - f(x)}{\delta x}$, where, without going into detail $f(x)$ has to be differentiable. For example, if $f(x) = x^2$ we find

$$\frac{df(x)}{dx} = \lim_{\delta x \to 0} \frac{(x+\delta x)^2 - x^2}{\delta x} = \lim_{\delta x \to 0} \frac{x^2 + 2x\delta x + (\delta x)^2 - x^2}{\delta x} = \lim_{\delta x \to 0} \frac{2x\delta x + (\delta x)^2}{\delta x} = 2x$$

In order to carry out numerical differentiation (or integration as we done already) we need to make one step back as there is no way we can take the step $\delta x \to 0$. We will therefore write:

$\frac{df(x)}{dx} \simeq \frac{f(x+\Delta x) - f(x)}{\Delta x}$ where Δx is a small but finite (discrete) number. The smaller we can make Δx the more closely we approximate $\frac{df}{dx}$.

A problem we have with this method is we don't have any idea how the size of Δx affects the accuracy in which we can determine $\frac{df}{dx}$. We also don't know how small we can make Δx in our calculations. We can however do better if we consider Taylor's expansion that is a formal result from calculus.

$f(x + \Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) + \ldots$. [Equation (1)]

where we have written $\frac{df}{dx} = f'$ etc. Re-arranging this we can write

$f'(x) \simeq \frac{1}{\Delta x}[f(x + \Delta x) - f(x) - \frac{\Delta x^2}{2!} f''(x) + \ldots]$. [Equation (2)]

or $f'(x) \simeq \frac{1}{\Delta x}[f(x + \Delta x) - f(x) - E(\Delta x) + \ldots]$

where $E(\Delta x) = \frac{\Delta x^2}{2!} f''(x) + \ldots$

From E we can then see that the error in our approximation of $f'(x)$ is of the order Δx.

We could equally well have written, using the Taylor expansion of $f(x - \Delta x)$

$f(x - \Delta x) = f(x) - \Delta x f'(x) + \frac{\Delta x^2}{2!} f''(x) + \ldots$. [Equation (3)]

so that

$f'(x) \simeq \frac{1}{\Delta x}[f(x) - f(x - \Delta x) + \frac{\Delta x^2}{2!}f''(x) + \ldots]$ [Equation 4]

Equations 2 & 4 are both valid and are known as the *forward* and *backward* difference methods and as you can see they will both have th same error estimator (order Δx).

5.1.2 Central difference methods

An alternative method of numerical differentiation is to subtract equation [3] from equation [1] from which we obtain the result

$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + E(\Delta x)$

This is known as the *central* difference method.

We also find that the $\frac{\Delta x^2}{2!}f''(x)$ term in the error term cancels in this expression so that in this case the error term is of the order $(\Delta x)^2$ rather than Δx. Hence the *central* difference method is considerably more accurate for a given Δx and this method would be used in preference.

Let's see how this works in a bit of code. The function we are differentiating is $f(x) = x^3 + 5x^2 + 3$ so $f'(x) = 3x^2 + 10x$. $f(x)$ is defined in the code as a function and can be easily changed if required. Input different values of x and compare your numerical results by the different methods with the analytical result. Note also in the code where we have defined the value of Δx we are using.

In [4]:

```
def f(x):
    ans=x**3+5*x**2+4
    return(ans)

xval = float(input('Enter xval =>'))
deltx = 0.01 # Define the value of delta X we will be using.

diffx = (f(xval+deltx)-f(xval-deltx))/2/deltx
fdiffx = (f(xval+deltx)-f(xval))/deltx

print("For delta x = ",deltx)
print ("df/dx (central difference) at x = ",xval," = ",diffx)
print ("df/dx (forward difference) at x = ",xval," = ",fdiffx)
```

```
Enter xval =>1
For delta x =  0.01
df/dx (central difference) at x =  1.0  =  13.000100000000003
df/dx (forward difference) at x =  1.0  =  13.080099999999995
```

Experiment with different values of x and changing the definition of $f(x)$ and Δx. You should be able to see how much better the result is for the central difference method.

You may think you can get better and better estimates by making Δx smaller and smaller. However, ultimately you will run into numerical precision errors as you approach dividing the small number generated from a difference Δf divided by another small number (Δx). In the code above try making Δx smaller and smaller (by factors of 10) until you reach Δx = 0.0000000000000001. What do you notice? (NB. you may need to restart the kernel occasionally if you do this).

In the code block below write some code to look at $f(x+\Delta)-f(x)$ and Δx in the code above. This should give you a better understanding of the issue.

In [ ]:

```
# Enter your code to print out the volues df and delta(x).
```

5.2 Summary

This has been a short notebook but it has contained some important ideas, notably how we can represent differerntials in our numerical programming and the importance in understanding the underlying mathematics properly in order to produce the most accurate results. In general you should avoid numerical differentiation. It is always preferable to use an analytical result for the derivative.

In this notebook you have learnt:

- how to calculate the derivative of a known 1D function using forward, backward and central difference methods,
- how using the central difference method gives a considerably improved accuracy,
- how the errors may be estimated,
- what happens if you try to be too accurate.