

9.3: Simpson's Rule

Our analysis of the mid-point rule and the trapezium rule showed that both methods have $O(1/N^2)$ numerical error. In both cases, the error can be traced to the same source: the fact that the integral estimate over each segment differs from the Taylor series result in the *second-order* term—the term proportional to $f''(x_n)$. This suggests a way to improve on the numerical integration result: we could take a weighted average of the mid-point rule and the trapezium rule, such that the *second-order* numerical errors from the two schemes cancel each other out! This is the numerical integration method known as **Simpson's rule**.

To be precise, let's again consider a pair of adjacent segments, which lie between the equally-spaced discretization points $\{x_{n-1}, x_n, x_{n+1}\}$. As derived above, the integral over these segments can be Taylor expanded as

$$\mathcal{I}_n = 2f(x_n)\Delta x + \frac{f''(x_n)}{3}\Delta x^3 + O(\Delta x^5) + \dots \quad (9.3.1)$$

By comparison, the mid-point rule and trapezium rule estimators for the integral are

$$\mathcal{I}_n^{\text{mp}} = 2f(x_n)\Delta x \quad (9.3.2)$$

$$\mathcal{I}_n^{\text{trapz}} = 2f(x_n)\Delta x + \frac{f''(x_n)}{2}\Delta x^3 + O(\Delta x^5). \quad (9.3.3)$$

Hence, we could take the following weighted average:

$$\mathcal{I}_n^{\text{simp}} = \frac{1}{3}\mathcal{I}_n^{\text{mp}} + \frac{2}{3}\mathcal{I}_n^{\text{trapz}} = 2f(x_n)\Delta x + \frac{f''(x_n)}{3}\Delta x^3 + O(\Delta x^5). \quad (9.3.4)$$

Such a weighted average would match the Taylor series result up to $O(\Delta x^4)$! (You can check for yourself that the $O(\Delta x^5)$ terms differ.) In summary, Simpson's rule for this set of three points can be written as

$$\mathcal{I}_n^{\text{simp}} = \frac{1}{3}\left[2f(x_n)\Delta x\right] + \frac{2}{3}\Delta x\left[\frac{f(x_{n-1})}{2} + f(x_n) + \frac{f(x_{n+1})}{2}\right] \quad (9.3.5)$$

$$= \frac{\Delta x}{3}\left[f(x_{n-1}) + 4f(x_n) + f(x_{n+1})\right]. \quad (9.3.6)$$

The total numerical error, over a set of $O(N)$ segments, is $O(1/N^4)$. That is an improvement of two powers of $1/N$ over the trapezium and mid-point rules! What's even better is that it involves exactly the same number of arithmetic operations as the trapezium rule. This is as close to a free lunch as you can get in computational science.

9.3.1 Python Implementation of Simpson's Rule

In Scipy, Simpson's rule is implemented by the `scipy.integrate.simps` function, which is defined in the `scipy.integrate` submodule. Similar to the `trapz` function, this can be called as either `simps(y,x)` or `simps(y,dx=s)` to estimate the integral $\int y \, dx$, using the elements of `x` as the discretization points, with `y` specifying the set of values for the integrand.

Because Simpson's rule requires dividing the segments into pairs, if you specify an even number of discretization points in `x` (i.e. an odd number of segments), the function deals with this by doing a trapezium rule estimate on the first and last segments. Usually, the error is negligible, so don't worry about this detail

Here is an example of `simps` in action:

```
>>> from scipy import *
>>> from scipy.integrate import simps
>>> x = linspace(0,10,25)
>>> y = exp(-x)
>>> t = simps(y,x)
>>> print(t)
1.00011864276
```

For the same number of discretization points, the trapezium rule gives 1.01438 the exact result is 0.9999546... Clearly, Simpson's rule is more accurate.

This page titled [9.3: Simpson's Rule](#) is shared under a [CC BY-SA 4.0](#) license and was authored, remixed, and/or curated by [Y. D. Chong](#) via [source content](#) that was edited to the style and standards of the LibreTexts platform.