

9.2: Trapezium Rule

The **trapezium rule** is a simple extension of the mid-point integration rule. Instead of calculating the area of a sequence of rectangles, we calculate the area of a sequence of trapeziums, as shown in Fig. 9.2.1. As before, we divide the integration region into N equal segments; however, we now consider the *end-points* of these segments, rather than their mid-points. These is a set of $N + 1$ positions $\{x_0, \dots, x_N\}$, such that

$$x_n = a + n \Delta x, \quad \Delta x \equiv \frac{b - a}{N}. \quad (9.2.1)$$

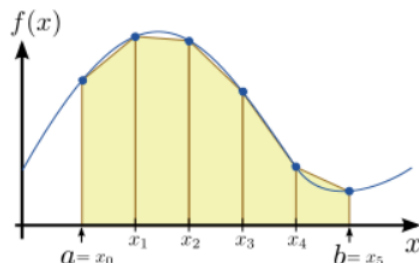


Figure 9.2.1: Computing a definite integral using the trapezium rule.

Note that $a = x_0$ and $b = x_N$. By using the familiar formula for the area of each trapezium, we can approximate the integral as

$$\mathcal{I}^{\text{trapz}} = \sum_{n=0}^{N-1} \Delta x \left(\frac{f(x_n) + f(x_{n+1})}{2} \right) \quad (9.2.2)$$

$$= \Delta x \left[\left(\frac{f(x_0) + f(x_1)}{2} \right) + \left(\frac{f(x_1) + f(x_2)}{2} \right) + \dots + \left(\frac{f(x_{N-1}) + f(x_N)}{2} \right) \right] \quad (9.2.3)$$

$$= \Delta x \left[\frac{f(x_0)}{2} + \left(\sum_{n=1}^{N-1} f(x_n) \right) + \frac{f(x_N)}{2} \right]. \quad (9.2.4)$$

9.2.1 Python Implementation of the Trapezium Rule

In Scipy, the trapezium rule is implemented by the `trapz` function. It usually takes two array arguments, `y` and `x`; then the function call `trapz(y, x)` returns the trapezium rule estimate for $\int y \, dx$, using the elements of `x` as the discretization points, and the elements of `y` as the values of the integrand at those points.

Note

Matlab compatibility note: Be careful of the order of inputs! For the Scipy function, it's `trapz(y, x)`. For the Matlab function of the same name, the inputs are supplied in the opposite order, as `trapz(x, y)`.

Note that the discretization points in `x` need not be equally-spaced. Alternatively, you can call the function as `trapz(y, dx=s)`; this performs the numerical integration assuming equally-spaced discretization points, with spacing `s`.

Here is an example of using `trapz` to evaluate the integral $\int_0^\infty \exp(-x^2) dx = 1$:

```
>>> from scipy import *
>>> x = linspace(0,10,25)
>>> y = exp(-x)
>>> t = trapz(y,x)
>>> print(t)
1.01437984777
```

9.2.2 Numerical Error for the Trapezium Rule

From a visual comparison of Fig. 9.1.1 (for the mid-point rule) and Fig. 9.2.1 (for the trapezium rule), we might be tempted to conclude that the trapezium rule should be more accurate. Before jumping to that conclusion, however, let's do the actual calculation of the numerical error. This is similar to our above calculation of the mid-point rule's numerical error. For the trapezium rule, however, it's convenient to consider a *pair* of adjacent segments, which lie between the three discretization points $\{x_{n-1}, x_n, x_{n+1}\}$.

As before, we perform a Taylor expansion of $f(x)$ in the vicinity of x_n :

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{f''(x_n)}{2}(x - x_n)^2 + \frac{f'''(x_n)}{6}(x - x_n)^3 + \dots \quad (9.2.5)$$

If we integrate $f(x)$ from x_{n-1} to x_{n+1} , the result is

$$\mathcal{I}_n \equiv \int_{x_n - \Delta x}^{x_n + \Delta x} f(x) dx \quad (9.2.6)$$

$$= 2f(x_n)\Delta x + f'(x_n) \int_{x_n - \Delta x}^{x_n + \Delta x} (x - x_n) dx + \frac{f''(x_n)}{2} \int_{x_n - \Delta x}^{x_n + \Delta x} (x - x_n)^2 dx + \dots \quad (9.2.7)$$

As before, every other term on the right-hand side integrates to zero. We are left with

$$\mathcal{I}_n = 2f(x_n)\Delta x + \frac{f''(x_n)}{3}\Delta x^3 + O(\Delta x^5) + \dots \quad (9.2.8)$$

This has to be compared to the estimate produced by the trapezium rule, which is

$$\mathcal{I}_n^{(\text{trapz})} = \Delta x \left[\frac{f(x_{n-1})}{2} + f(x_n) + \frac{f(x_{n+1})}{2} \right]. \quad (9.2.9)$$

If we Taylor expand the first and last terms of the above expression around x_n , the result is:

$$\mathcal{I}_n^{(\text{trapz})} = \frac{\Delta x}{2} \left[f(x_n) - f'(x_n)\Delta x + \frac{f''(x_n)}{2}\Delta x^2 - \frac{f'''(x_n)}{6}\Delta x^3 + \frac{f''''(x_n)}{24}\Delta x^4 + O(\Delta x^4) \right] \quad (9.2.10)$$

$$+ \Delta x f(x_n) \quad (9.2.11)$$

$$+ \frac{\Delta x}{2} \left[f(x_n) + f'(x_n)\Delta x + \frac{f''(x_n)}{2}\Delta x^2 + \frac{f'''(x_n)}{6}\Delta x^3 + \frac{f''''(x_n)}{24}\Delta x^4 + O(\Delta x^4) \right] \quad (9.2.12)$$

$$= 2f(x_n)\Delta x + \frac{f''(x_n)}{2}\Delta x^3 + O(\Delta x^5). \quad (9.2.13)$$

Hence, the numerical error for integrating over this pair of segments is

$$\mathcal{E}_n \equiv |\mathcal{I}_n - \mathcal{I}_n^{\text{trapz}}| = \frac{|f''(x_n)|}{6}\Delta x^3 \sim O\left(\frac{1}{N^3}\right). \quad (9.2.14)$$

And the total numerical error goes as

$$\mathcal{E}_{\text{total}} \sim O\left(\frac{1}{N^2}\right), \quad (9.2.15)$$

which is the same scaling as the mid-point rule! Despite our expectations, the trapezium rule isn't actually an improvement over the mid-point rule, as far as numerical accuracy is concerned.

This page titled [9.2: Trapezium 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.