

§6 Numerical Calculus

One of the most basic but also most important applications of computers in physics is the evaluation of integrals and derivatives. Numerical evaluation of integrals is a particularly crucial topic since integrals occur widely in physics calculations and most cannot be done analytically in closed form while some can be. However, they can almost always be done on a computer. In this chapter, we examine a number of different techniques for evaluating integrals and derivatives, as well as taking a brief look at the related operation of interpolation.

6.1 Simple Methods for Numerical Integration

Suppose we want to evaluate the integral of a given function. Let us first consider the simplest case: the integral of a function of a single variable over a finite range. In this section, we study some simple methods for the numerical evaluation of such integral.

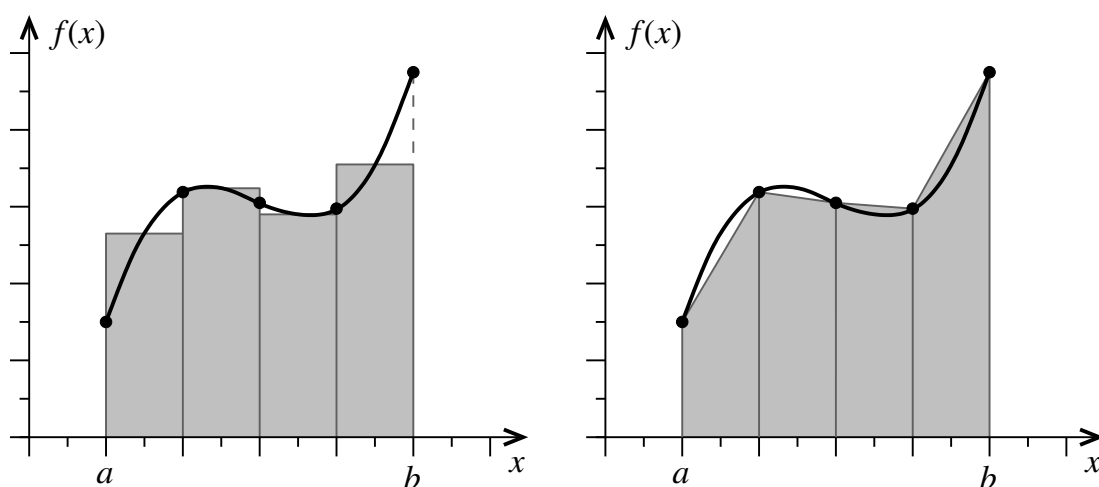


Figure 6.1: The approximation of the area under a curve by the rectangular method (left) and the trapezoidal rule (right).

Assume that we want to calculate the integral of a function $f(x)$ with respect to x from $x = a$ to $x = b$, i. e.

$$I(a, b) = \int_a^b f(x) dx$$

This is equivalent to calculating the area under the curve of $f(x)$ from $x = a$ to $x = b$. There is no known way to calculate such an area exactly in all cases on a computer. However, we can do it approximately by the **rectangular method** as shown on the left in Figure 6.1. In this method, we divide the area into rectangular slices, calculate the

area of each one, and then sum them up. But this is a pretty poor approximation. The area under the rectangles is not very close to the true area under the curve.

A better approach is the **trapezoidal rule** in which the area is divided into trapezoids rather than rectangles as shown on the right in Figure 6.1. The area under the trapezoids is a considerably better approximation to the area under the curve. This approach involves very little extra work and often gives much better results though it's simple.

Suppose we divide the interval from $x = a$ to $x = b$ into N slices of width $h = (b-a)/N$. Then the right-hand side of the k th slice falls at $x = a + kh$ and the left-hand side falls at $x = a + (k-1)h$. So the area of the trapezoid for this slice is


$$A_k = \frac{h}{2}[f(a + (k-1)h) + f(a + kh)] \quad (6.1)$$

This is called the trapezoidal rule. It gives us a trapezoidal approximation to the area under one slice of our function.

Our approximation for the area under the whole curve is the sum of the areas of the trapezoids for all N slices:

$$\begin{aligned} I(a, b) &\approx \sum_{k=1}^N A_k = h \left[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \dots + \frac{1}{2}f(b) \right] \\ &= h \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N-1} f(a+kh) \right] \end{aligned} \quad (6.2)$$

This is called the **extended trapezoidal rule** — it is the extension to many slices of the basic trapezoidal rule of Eq. (6.1). But we often refer to it simply as the trapezoidal rule. Beware that the quantity inside the square brackets is a sum over the values of $f(x)$ measured at equally spaced points in the integration domain, and we take a half of the values at the start and end points but one times the value at all the interior points.



Example 6.1

Let us use the trapezoidal rule to compute the integral $\int_0^4 (x^4 - 4x + 4) dx$. This is actually an integral we can do by hand. But it's a good first example because we can easily check if our program is working and how accurate an answer it gives. Below is the Python program `trapezoidal.py` which evaluates the integral using the trapezoidal rule with $N = 10$ slices.

```
# trapezoidal.py
```

```
# This program calculates the integral of the function  $x^4 - 4x + 4$ 
# from  $x = 0$  to  $x = 4$  using the trapezoidal rule.
# Last update on 26 Nov 2020 by F K Chow

def f(x):
    return  $x**4 - 4*x + 4$ 

N = 10
a, b = 0.0, 4.0
h = (b-a)/N
s = 0.5*f(a) + 0.5*f(b)

for k in range(1,N):
    s += f(a+k*h)

print("{0:.9f}".format(h*s))
```

This is a straightforward translation of the trapezoidal rule formula into computer code. We create a function that calculates the integrand, set up all the required constants, evaluate the sum for the integral $I(a, b)$ term by term, multiply it by h , and then output the result. Running this program produces the output **192.209920000** while the true answer is

$$\int_0^4 (x^4 - 4x + 4) dx = \left[\frac{1}{5}x^5 - 2x^2 + 4x \right]_0^4 = 188.8$$

So our calculation is moderately but not exceptionally accurate — the answer is off the true answer by about 2%.

We can make the calculation more accurate by increasing the number of slices. When N is larger, we approximate the area under the curve better though the program will also take longer to reach an answer since there are more terms in the sum to evaluate. If we increase the number of slices to $N = 100$ and run the program again, then we get **188.834132992**. The result is now accurate to 0.02% which is pretty good. And if we use $N = 1000$, then we get **188.800341333** which is accurate to 0.0002%. In the next section, we will study the accuracy of the trapezoidal rule in more detail.

The trapezoidal rule is a simple numerical integration methods, taking only a few lines of code as we have seen. But it is often perfectly adequate for calculations where no great accuracy is required. In many physics calculations, we don't need an answer accurate to

many significant figures. In such cases, the ease and simplicity of the trapezoidal rule can make it the method of choice. One should not turn up one's nose at simple methods like this. They play an important role and are widely used. In addition, the trapezoidal rule is the basis for several other more sophisticated methods of evaluating integrals including the adaptive methods in Section 6.3 and the Romberg integration method in Section 6.4.

However, there are also cases where a greater accuracy is required. As we have seen, we can increase the accuracy of the trapezoidal rule by increasing the number of steps N used in the calculation. But in some cases, particularly for integrands that are rapidly varying, a very large number of steps may be required to achieve the desired accuracy, which means the calculation can become slow. There are other more advanced schemes for calculating integrals that can achieve high accuracy while still arriving at an answer quickly. Here we are going to study one such scheme called **Simpson's rule**.

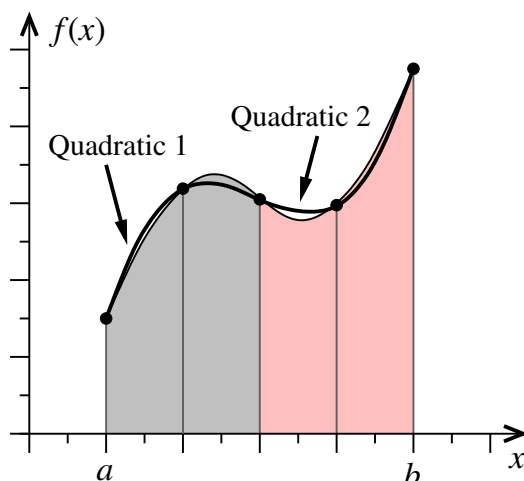


Figure 6.2: The approximation of the area under a curve by the Simpson's rule.

In practice, the trapezoidal rule estimates the area under a curve by approximating the curve with straight-line segments. We can often get a better result if we approximate the function instead with curves of some kind. Simpson's rule does this by using quadratic curves as shown in Figure 6.2. In order to specify a quadratic completely, one needs three points, not just two as with a straight line. So in this method we take a pair of adjacent slices and fit a quadratic through the three points that mark the boundaries of those slices. In Figure 6.2, there are two quadratics fitted to four slices. Simpson's rule involves approximating the integrand with quadratics in this way and then calculating the area under those quadratics, which gives an approximation to the area under the true curve.

Let us denote our integrand as $f(x)$ and the spacing of adjacent points as h just like before. And suppose for the purposes of argument that we have three points at $x = -h$,

0, and h . If we fit a quadratic $Ax^2 + Bx + C$ through these points, then by definition we will have:

$$f(-h) = Ah^2 - Bh + C, \quad f(0) = C, \quad f(h) = Ah^2 + Bh + C.$$

Solving these equations simultaneously for A , B , and C gives

$$A = \frac{1}{h^2} \left[\frac{1}{2}f(-h) - f(0) + \frac{1}{2}f(h) \right], \quad B = \frac{1}{2h} [f(h) - f(-h)], \quad f(0) = C,$$

And the area under the curve of $f(x)$ from $x = -h$ to $x = h$ is given approximately by the area under the quadratic

$$\int_{-h}^h (Ax^2 + Bx + C) dx = \frac{2}{3}Ah^3 + 2Ch = \frac{1}{3}h [f(-h) + 4f(0) + f(h)] \quad (6.3)$$

This is known as Simpson's rule. It gives us an approximation to the area under two adjacent slices of our function. Note that the final formula for the area involves only h and the value of the function at evenly spaced points, just as with the trapezoidal rule. So we don't actually have to worry about the details of fitting a quadratic as we use Simpson's rule. We just plug numbers into Eq. (6.3) and it gives us an answer. This makes Simpson's rule almost as simple to use as the trapezoidal rule, and yet Simpson's rule often gives much more accurate results.

To use Simpson's rule performing a general integral, we note that Eq. (6.3) does not depend on the fact that our three points lie at $x = -h$, 0, and h . If we were to slide the curve along the x -axis to either higher or lower values, the area underneath it would not change. So we can use the same rule for any three uniformly spaced points. Applying Simpson's rule involves dividing the domain of integration into many slices and using the rule to separately estimate the area under successive pairs of slices, then adding the estimates for all pairs to get the final answer. If we are integrating from $x = a$ to $x = b$ in slices of width h just like before, then the three points bounding the first pair of slices fall at $x = a$, $a + h$, and $a + 2h$, those bounding the second pair at $x = a + 2h$, $a + 3h$, and $a + 4h$, and so forth. Then the approximate value of the entire integral is given by

$$\begin{aligned} I(a, b) &\approx \frac{1}{3}h[f(a) + 4f(a + h) + f(a + 2h)] \\ &\quad + \frac{1}{3}h[f(a + 2h) + 4f(a + 3h) + f(a + 4h)] + \dots \\ &\quad + \frac{1}{3}h[f(a + (N - 2)h) + 4f(a + (N - 1)h) + f(b)] \end{aligned}$$

Notice that the total number of slices N must be even for this formula to work. Collecting


```
s += (4*f(a+(2*k-1)*h)+2*f(a+2*k*h))/3.0
print("{0:.9f}".format(h*s))
```

```

# Bessel function of first kind of order m using Simpson's rule
# with N = 1000 points and then makes a density plot of the Airy
# pattern produced by light of wavelength lam from a point source
# passing through a small circular aperture of radius a. The Airy
# pattern is plotted on a square screen of side length L located
# at a distance D from the aperture for maximum intensity I0 = 1.
# Last update on 7 Jan 2022 by F K Chow

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

lam, a = 500e-9, 1e-5    # In units of m
D, L = 0.1, 0.01        # In units of m
k = 2*np.pi/lam
Np = 1001

def f(q, m, x):
    """ Calculate the function f(q) = cos(m*q - x*sin(q)) """
    return np.cos(m*q - x*np.sin(q))

def J(m, x):
    """ Calculate the Bessel function of first kind of order m
        using Simpson's rule with N = 1000 points """
    N = 1000
    a, b = 0, np.pi
    h = (b-a)/N
    s = f(a, m, x) + f(b, m, x) + 4*f(a+(N-1)*h, m, x)
    for k in range(1, int(N/2)):
        s += 4*f(a+(2*k-1)*h, m, x) + 2*f(a+2*k*h, m, x)
    return h*s/(3.0*np.pi)

# Generate the data for the density plot of the Airy pattern
# Use the function np.errstate to suppress the division by zero
# warning for z = 0 and then set I(0) = 1 as  $\lim_{z \rightarrow 0} J_1(z)/z = 0.5$ 

```


In this program, the Airy pattern is plotted on a square screen of side length $L = 1$ cm located at a distance of $D = 10$ cm from the aperture for maximum intensity $I_0 = 1$. Figure 6.3 shows the plot of the Airy pattern generated by this program.



6.2 Errors on Numerical Integrals

Our numerical integrals are only approximations. As with most numerical calculations, there is usually a round-off error when we calculate an integral. But this is not the main source of error. The main source of error is the truncation error — the fact that our integration rules themselves are only approximations to the true integral. Both the trapezoidal and Simpson's rules calculate the area under a curve using an approximation (either linear or quadratic) to the integrand, not the integrand itself. How big an error does this approximation introduce?

Consider the integral $\int_a^b f(x) dx$ again and let us first look at the trapezoidal rule stated in Equation (6.2). To simplify our notation a little bit, let us define $x_k = a + kh$ as a shorthand for the positions at which we evaluate the integrand $f(x)$. We will refer to these positions as sample points. Now consider the slice of the integral that falls between x_{k-1} and x_k . By performing a Taylor series expansion of $f(x)$ about $x = x_{k-1}$, we obtain:

$$f(x) = f(x_{k-1}) + (x - x_{k-1})f'(x_{k-1}) + \frac{1}{2}(x - x_{k-1})^2 f''(x_{k-1}) + \dots$$

where f' and f'' denote the first and second order derivatives of f , respectively. Integrating this expression from x_{k-1} to x_k gives

$$\begin{aligned} \int_{x_{k-1}}^{x_k} f(x) dx &= f(x_{k-1}) \int_{x_{k-1}}^{x_k} dx + f'(x_{k-1}) \int_{x_{k-1}}^{x_k} (x - x_{k-1}) dx \\ &\quad + \frac{1}{2} f''(x_{k-1}) \int_{x_{k-1}}^{x_k} (x - x_{k-1})^2 dx + \dots \end{aligned}$$

Now we make the substitution $u = x - x_{k-1}$ in the above equation. Then we have

$$\begin{aligned} \int_{x_{k-1}}^{x_k} f(x) dx &= f(x_{k-1}) \int_0^h du + f'(x_{k-1}) \int_0^h u du + \frac{1}{2} f''(x_{k-1}) \int_0^h u^2 du + \dots \\ &= hf(x_{k-1}) + \frac{1}{2} h^2 f'(x_{k-1}) + \frac{1}{6} h^3 f''(x_{k-1}) + O(h^4) \end{aligned} \quad (6.5)$$

where $O(h^4)$ denotes the rest of the terms in the series, i. e. those in h^4 and higher, that we are neglecting.

We can do a similar expansion about $x = x_k$ and again integrate the resulting expression from $x = x_{k-1}$ to $x = x_k$ to obtain

$$\int_{x_{k-1}}^{x_k} f(x) dx = hf(x_k) - \frac{1}{2} h^2 f'(x_k) + \frac{1}{6} h^3 f''(x_k) - O(h^4). \quad (6.6)$$

Then taking the average of Equations (6.5) and (6.6) yields

$$\begin{aligned} \int_{x_{k-1}}^{x_k} f(x) dx &= \frac{1}{2}h[f(x_{k-1}) + f(x_k)] + \frac{1}{4}h^2[f'(x_{k-1}) - f'(x_k)] \\ &\quad + \frac{1}{12}h^3[f''(x_{k-1}) + f''(x_k)] + O(h^4) \end{aligned}$$

Finally, we sum this expression over all slices k to get the full integral that we want:

$$\begin{aligned} \int_a^b f(x) dx &= \sum_{k=1}^N \int_{x_{k-1}}^{x_k} f(x) dx = \frac{1}{2}h \sum_{k=1}^N [f(x_{k-1}) + f(x_k)] + \frac{1}{4}h^2[f'(a) - f'(b)] \\ &\quad + \frac{1}{12}h^3 \sum_{k=1}^N [f''(x_{k-1}) + f''(x_k)] + O(h^4) \end{aligned} \quad (6.7)$$

Let's take a closer look on this equation to see what's going on.

The first sum on the right-hand side of Eq. (6.7) is precisely equal to the trapezoidal rule in Eq. (6.2). When we use the trapezoidal rule, we evaluate only this sum and discard all the remaining terms. The size of the discarded terms, i. e. the rest of the series, measures the amount we would have to add to the trapezoidal rule value in order to get the true value of the integral. In other words, it is equal to the truncation error incurred by the use of the trapezoidal rule.

For the term in h^2 , almost all the terms of the sum have canceled out, leaving only the first and last terms (those evaluated at $x = a$ and $x = b$). Although we haven't shown it, a similar cancellation happens for the terms in all even powers of h .

Now we consider the term in h^3 and note the following useful fact: the sum in this term is just the trapezoidal rule approximation to the integral of $f''(x)$ over the interval from $x = a$ to $x = b$ to within an overall constant. Specifically, if we take Eq. (6.7) and make the substitution $f(x) \rightarrow f''(x)$ on the left-hand side, then we have

$$\int_a^b f''(x) dx = \frac{1}{2}h \sum_{k=1}^N [f''(x_{k-1}) + f''(x_k)] + O(h^2).$$

Multiplying this expression by $(1/6)h^2$ and rearranging, we then get

$$\frac{1}{12}h^3 \sum_{k=1}^N [f''(x_{k-1}) + f''(x_k)] = \frac{1}{6}h^2 \int_a^b f''(x) dx + O(h^4) = \frac{1}{6}h^2[f'(b) - f'(a)] + O(h^4)$$

since the integral of $f''(x)$ is just $f'(x)$. Substituting this result back into Eq. (6.7) and canceling some terms, we find that

$$\int_a^b f(x) dx = \frac{1}{2}h \sum_{k=1}^N [f(x_{k-1}) + f(x_k)] + \frac{1}{12}h^2[f'(a) - f'(b)] + O(h^4). \quad (6.8)$$

Thus the value of the terms dropped in using the trapezoidal rule, which equals the truncation error ϵ on the integral, is given to leading order in h by

$$\epsilon = \frac{1}{12}h^2[f'(a) - f'(b)]. \quad (6.9)$$

This is the **Euler-Maclaurin formula** giving the error of the trapezoidal rule. More precisely, it is the first term in the Euler-Maclaurin formula; the full formula keeps the terms to all orders in h . From Eq. (6.8), we can see that the next term in the series is of order h^4 . We might imagine it would be of order h^3 . But the h^3 term actually cancels out. Indeed, it's not difficult to show that only even powers of h survive in the full formula at all orders. So the next term after h^4 is h^6 , then h^8 , and so forth. However, we can neglect the h^4 and higher-order terms as long as h is small. The leading term in Eq. (6.8) is usually enough.

Equation (6.8) tells us that the trapezoidal rule is a *first-order* integration rule. It means that it's accurate up to and including terms proportional to h and the leading-order truncation error is of order h^2 . That is to say, a first-order rule is accurate to $O(h)$ and has an error of $O(h^2)$.

In addition to truncation error, there is also a round-off error in our calculation. As discussed in Section 5.2, this round-off error will have an approximate size of C times the value of the integral where C is about 10^{-16} in Python. Equation (6.9) tells us that the truncation error gets smaller as h gets smaller. So we can make our integral more accurate by using smaller h or a larger number of slices N . However, we should not make h so small that the truncation error becomes much smaller than the round-off error. Further decreases in h beyond this point will only make our program slower without improving the accuracy of our calculation significantly since accuracy will be dominated by the round-off error in such case.

Hence, decreases in h will only help us up to the point at which the truncation and round-off errors are roughly equal. It is the point where

$$\begin{aligned} \frac{1}{12}h^2[f'(a) - f'(b)] &\approx C \int_a^b f(x) dx \\ \Rightarrow h &\approx \sqrt{\frac{12 \int_a^b f(x) dx}{f'(a) - f'(b)}} C^{1/2} \end{aligned}$$

Then we can set $h = (b - a)/N$ to obtain

$$N \approx (b - a) \sqrt{\frac{f'(a) - f'(b)}{12 \int_a^b f(x) dx}} C^{-1/2}. \quad (6.10)$$

For example, if all the factors except the last one in the above equation are of order unity, then the round-off error will become important when $N \approx 10^8$. Alternatively speaking, this is the point at which the accuracy of the trapezoidal rule reaches the machine precision — the maximum accuracy with which the computer can represent the result. There is no point for increasing the number of integration slices beyond this point as the calculation will not become any more accurate. However, $N = 10^8$ would be an unusually large number of slices for the trapezoidal rule. It would be rare to use such a large number when equivalent accuracy can be achieved using much smaller N with a more accurate rule such as Simpson's rule. Thus, in most practical situations, we will be in the regime where the truncation error is the dominant source of inaccuracy. Then it is safe for us to assume that the round-off error can be ignored.

We can do an analogous error analysis for Simpson's rule. The algebra is similar but much more tedious. Here we'll just quote the results. For an integral of the function $f(x)$ over the interval from $x = a$ to $x = b$, the truncation error is given to leading order by

$$\epsilon = \frac{1}{180} h^4 [f'''(a) - f'''(b)]. \quad (6.11)$$

Thus Simpson's rule is a *third-order* integration rule with a fourth-order truncation error. It means that the error of Simpson's rule will typically be much smaller than that of the trapezoidal rule for small values of h . This explains why Simpson's rule gave superior results in Example 6.2 in Section 6.1.

The round-off error for Simpson's rule is again of order $C \int_a^b f(x) dx$. So the equivalent of Eq. (6.10) is

$$N \approx (b - a) \left(\frac{f'''(a) - f'''(b)}{180 \int_a^b f(x) dx} \right)^{1/4} C^{-1/4}.$$

If the leading factors are again roughly of order unity, then the round-off error will become important when $N \approx 10000$. Beyond this point Simpson's rule is so accurate that its accuracy exceeds the machine precision of the computer. By contrast with the case for the trapezoidal rule, $N = 10000$ is not an unusually large number of slices to be used in a calculation. Calculations with ten thousand slices can be done easily in a fraction of a second. You should keep in mind that there is no point to use more than a few thousand slices with Simpson's rule because the calculation will reach the limits of precision of the computer and larger values of N will do no further good.

Notice that it is not always guaranteed that Simpson's rule do better than the trapezoidal rule although the former one in general gives superior accuracy. It is because the

errors on these two rules also depend on derivatives of the integrand function via Equations (6.9) and (6.11). For example, it would be possible for $f'''(a)$ by bad luck to be large in some particular instance, making the error in Eq. (6.11) similarly large, and possibly worse than the error for the trapezoidal rule. It would be fair to say that Simpson's rule usually gives better results than the trapezoidal rule. But cautious scientist will bear in mind that it can do worse on occasion.

The Euler-Maclaurin formula for the trapezoidal rule, Eq. (6.9), and its equivalent for Simpson's rule, Eq. (6.11), allow us to calculate the error on our integrals provided that we have a known closed-form expression for the integrand $f(x)$ so that we can calculate the derivatives that appear in the formulas. Unfortunately, we have no such expression in many cases. For instance, the integrand may not be a mathematical function at all but a set of measurements made in the laboratory or it might itself be the output of another computer program. In such cases, we cannot differentiate the function and Equations (6.9) and (6.11) will not work. However, there is still a way to calculate the error.

Suppose that we are again evaluating an integral over the interval from $x = a$ to $x = b$. And assume that we are using the trapezoidal rule since it makes the argument simpler although the method described here extends to Simpson's rule too. Let us evaluate the integral with some number of steps N_1 so that the step size is $h_1 = (b - a)/N_1$.

Then here's the trick: we now double the number of steps and evaluate the integral again. In other words, we define a new number of steps $N_2 = 2N_1$ and a new step size $h_2 = (b - a)/N_2 = h_1/2$. Then we evaluate the integral again using the trapezoidal rule, giving a new answer which will normally be more accurate than the previous one. As we have seen, the trapezoidal rule introduces an error of order $O(h^2)$. It implies that we *quarter* the size of our error if the value of h is halved. Knowing this fact allows us to estimate how big is the error.

Suppose that the true value of our integral is I and let us denote our first estimate using the trapezoidal rule with N_1 steps by I_1 . The difference between the true value and the estimate, i. e. the error on the estimate, is proportional to h^2 . So let us write it as ch^2 where c is a constant. Then I and I_1 are related by $I = I_1 + ch_1^2$, neglecting higher-order terms. We can also write a similar formula for our second estimate I_2 of the integral with N_2 steps, namely, $I = I_2 + ch_2^2$. Equating the two expressions for I yields

$$I_2 - I_1 = ch_1^2 - ch_2^2 = 3ch_2^2,$$

where we have made use of the fact that $h_1 = 2h_2$. Rearranging this equation then gives

the error ϵ_2 on the second estimate of the integral:

$$\epsilon_2 = ch_2^2 = \frac{1}{3}(I_2 - I_1). \quad (6.12)$$

This expression can be either positive or negative, depending on which way the error happens to go. If we only want the absolute size of the error, then we can take the absolute value $(1/3)|I_2 - I_1|$, which in Python can be done using the built-in function `abs`.

This method gives us a simple way to estimate the error on the trapezoidal rule without using the Euler-Maclaurin formula. Indeed, even in cases where we could in principle use the Euler-Maclaurin formula, it is often simpler in practice to use the method of Eq. (6.12) instead which is easy to program and gives reliable answers.

As mentioned before, the same principle can be applied to integrals evaluated using Simpson's rule too. The equivalent of Eq. (6.12) in that case turns out to be

$$\epsilon_2 = \frac{1}{15}(I_2 - I_1).$$

The derivation of this expression is left as an exercise for you.

6.3 Number of Steps for Numerical Integration

So far we have not specified how to choose the number of steps N used in our integrals. In our previous calculations we just chose round numbers and looked to see if the results seem reasonable. This is fine for quick calculations. However, we want a more principled approach for serious physics problems. We may know in advance how many steps we want to use in some calculations. Sometimes we have a “budget”, i. e. a certain amount of computer time that we can spend on a calculation, and our goal is simply to make the most accurate calculation we can in the given amount of time. For example, if we know that we have time to do a thousand steps, then that's what we do.

But a more common situation is that we want to calculate the value of an integral to a given accuracy and we would like to know how many steps will be required. It should always be possible to meet our goal by using a large enough number of steps so long as the desired accuracy does not exceed the fundamental limit set by the machine precision of our computer — the round-off error that limits all calculations. At the same time, we want to avoid using more steps than necessary since more steps take more time and our calculation will become slower. Ideally we would like a number of steps N that gives us the accuracy we want and no more.

A simple way to achieve this is to start with a small value of N and repeatedly double it until we achieve the accuracy we want. As we saw in the Section 6.2, there is a simple formula, Eq. (6.12), for calculating the error on an integral when we double the number of steps. By using this formula with repeated doubling, we can evaluate an integral to exactly the accuracy we want.

The procedure is straightforward. We start off by evaluating the integral with some small number of steps N_1 . For instance, we might choose $N_1 = 10$. Then we double the number to $N_2 = 2N_1$, evaluate the integral again, and apply Eq. (6.12) to calculate the error. If the error is small enough to satisfy our accuracy requirements, then we're done and we have our answer; otherwise we keep on doubling the number of steps until we achieve the required accuracy. The error on the i th step of the process is given by the obvious generalization of Eq. (6.12):

$$\epsilon_i = \frac{1}{3}(I_i - I_{i-1}). \quad (6.13)$$

where I_i is the i th estimate of the integral. This method is an example of an **adaptive integration** method that varies its own parameters to get a desired answer.

A very nice feature of this method is that we don't need to recompute the entire integral again when we double the number of steps. We can reuse our previous calculation rather than just throwing it away. To see this, let's take a look at Figure 6.4. The top part of the figure depicts the locations of the sample points — the values of x at which the integrand is evaluated in the trapezoidal rule. The sample points are regularly spaced, and remember that the first and last points are treated differently from the others — the trapezoidal rule formula Eq. (6.2) specifies that the values of $f(x)$ at these points are multiplied by a factor of $1/2$ while the values at the interior points are multiplied by a factor of 1.

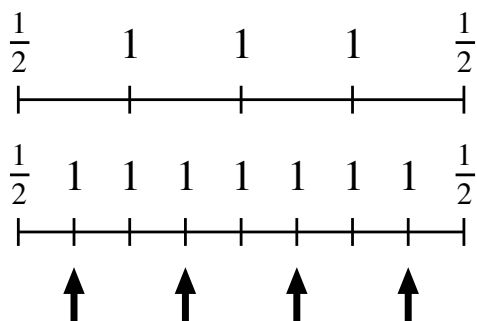


Figure 6.4: As we double the number of steps in the trapezoidal rule, we effectively add a new set of points, half way between the previous points, as indicated by the arrows.

The lower part of Fig. 6.4 shows what happens as we double the number of steps. This

adds an additional set of sample points half way between the old ones, as indicated by the arrows. Note that the original points are still included in the calculation and still carry the same multiplying factors as before — $1/2$ at the ends and 1 in the middle — while the new points are all multiplied by a simple factor of 1 . Thus we have all of the same terms in our trapezoidal rule sum that we had before (terms that we have already evaluated). But we also have a set of new ones which we have to add into the sum to calculate its full value. In the jargon of computational physics, we say that the sample points for the first estimate of the integral are nested inside the points for the second estimate.

To put this in mathematical terms, consider the trapezoidal rule at the i th step of the calculation. Let the number of slices at this step be N_i and the width of each slice be $h_i = (b - a)/N_i$. Note that on the previous step there were half as many slices of twice the width so that $N_{i-1} = (1/2)N_i$ and $h_{i-1} = 2h_i$. Then

$$\begin{aligned} I_i &= h_i \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N_i-1} f(a + kh_i) \right] \\ &= h_i \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{\substack{k \text{ even} \\ 2, \dots, N_i-2}} f(a + kh_i) + \sum_{\substack{k \text{ odd} \\ 1, \dots, N_i-1}} f(a + kh_i) \right] \end{aligned}$$

However,

$$\sum_{\substack{k \text{ even} \\ 2, \dots, N_i-2}} f(a + kh_i) = \sum_{k=1}^{N_i/2-1} f(a + 2kh_i) = \sum_{k=1}^{N_{i-1}-1} f(a + kh_{i-1}),$$

Hence,

$$I_i = \frac{1}{2}h_{i-1} \left[\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N_{i-1}-1} f(a + kh_{i-1}) \right] + h_i \sum_{\substack{k \text{ odd} \\ 1, \dots, N_i-1}} f(a + kh_i),$$

But the term $h_{i-1}[\dots]$ in this equation is precisely the trapezoidal rule estimate I_{i-1} of the integral on the previous iteration of the process. So

$$I_i = \frac{1}{2}I_{i-1} + h_i \sum_{\substack{k \text{ odd} \\ 1, \dots, N_i-1}} f(a + kh_i). \quad (6.14)$$

In effect, our old estimate gives us half of the terms in our trapezoidal rule sum and we only have to calculate the other half. In this way we avoid ever recalculating any term that has already been calculated, meaning that each term in our sum is calculated only once, regardless of how many levels of the calculation it's used in. This means it takes only about as much work to calculate I_i going through all the successive levels I_1, I_2, I_3, \dots

as it does to calculate I_i directly using the ordinary trapezoidal rule. Thus we pay very little extra price in terms of the running time of our program to use this adaptive method and we gain the significant advantage of a guarantee in the accuracy of the integral.

In summary, the entire process of the adaptive approach is as follows:

1. Choose an initial number of steps N_1 and decide on the target accuracy for the value of the integral. Calculate the first approximation I_1 to the integral using the chosen value of N_1 with the standard trapezoidal rule formula, Eq. (6.2).
2. Double the number of steps and then use Eq. (6.14) to calculate an improved estimate of the integral. Also calculate the error on that estimate from Eq. (6.13).
3. Stop if the absolute magnitude of the error is less than the target accuracy for the integral. Otherwise repeat from step 2.

The sum over odd values of k in Eq. (6.14) can be conveniently performed in Python with a for loop of the form `for k in range(1,N,2)`.

We can also derive a similar method for integrals evaluated using Simpson's rule. Again we double the number of steps on each iteration of the process and the equivalent of Eq. (6.13) is

$$\epsilon_i = \frac{1}{15}(I_i - I_{i-1}). \quad (6.15)$$

The equivalent of Eq. (6.14) is a little more complicated. We define

$$S_i = \frac{1}{3} \left[f(a) + f(b) + 2 \sum_{\substack{k \text{ even} \\ 2, \dots, N_i-2}} f(a + kh_i) \right], \quad (6.16)$$

$$T_i = \frac{2}{3} \sum_{\substack{k \text{ odd} \\ 1, \dots, N_i-1}} f(a + kh_i). \quad (6.17)$$

Then we can show that

$$S_i = S_{i-1} + T_{i-1} \quad \text{and} \quad I_i = h_i(S_i + 2T_i). \quad (6.18)$$

Thus for Simpson's rule the complete process of the adaptive approach is:

1. Choose an initial number of steps N_1 and a target accuracy. Calculate the sums S_1 and T_1 from Eqs. (6.16) and (6.17) and the initial value I_1 of the integral from Eq. (6.18).
2. Double the number of steps and then use Equations (6.17) and (6.18) to calculate the new values of S_i and T_i as well as the new estimate of the integral. Also calculate the error on that estimate from Eq. (6.15).


```

""" Calculate the function f1(x) = 2*x*sin(3*pi*x)^2 """
return 2*x*sin(3*pi*x)**2

def f2(x):
    """ Calculate the function f2(x) = 2*x^2*sin(3*pi*x)^2 """
    return 2*(x*sin(3*pi*x))**2

def adptrapezoidal(f, a, b, N, tol):
    """ Evaluate the integral of f for a <= x <= b using the adaptive
        trapezoidal rule with an accuracy of tol and an initial number
        of slices of N """
    error = 1
    h = (b - a)/N
    s = 0.5*(f(a) + f(b))
    for k in range(1, N):
        s += f(a+k*h)
    Io = s*h
    while abs(error) > tol:
        N *= 2
        h *= 0.5
        s = 0
        for k in range(1, N, 2):
            s += f(a+k*h)
        Ic = 0.5*Io + s*h
        error = (Ic - Io)/3.0
        Io = Ic
    return Ic

# Use the adaptive trapezoidal rule to evaluate the integrals for
# <x>/L and <x^2>/L^2 and find the position uncertainty in units of L
a, b = 0, 1
tol = 1e-8
N = 10
xe = adptrapezoidal(f1, a, b, N, tol) # xe = <x>/L
x2e = adptrapezoidal(f2, a, b, N, tol) # x2e = <x^2>/L

```

```
deltax = sqrt(x2e - xe**2) # deltax = position uncertainty/L

# Output the results of the calculation

print("For a particle in the n = 3 quantum state of ")
print("a 1D box of length L,")
print(" <x> = {:.8f}L, <x^2> = {:.8f}L^2, ".format(xe,x2e))
print(" position uncertainty Delta x = {:.8f}L.".format(deltax))
```

Below is the output of this program. We can see that it agrees with the theoretical value of $\Delta x = \sqrt{(1 - 2/3\pi^2)/12}L$.

For a particle in the $n = 3$ quantum state of a 1D box of length L ,

```
<x> = 0.500000000L, <x^2> = 0.32770438L^2,
position uncertainty Delta x = 0.27875505L.
```


6.4 Romberg Integration

We can do even better than the adaptive method of the last section with only a little more effort. Let us go back to the trapezoidal rule again. We have seen that the leading-order error on the trapezoidal rule at the i th step of the adaptive method can be written as ch_i^2 for some constant c and is given by Eq. (6.13) as

$$ch_i^2 = \frac{1}{3}(I_i - I_{i-1})$$

But by definition the true value of the integral is $I = I_i + ch_i^2 + O(h_i^4)$, where we are including the $O(h_i^4)$ term to remind us of the next term in the series. So we have

$$I = I_i + \frac{1}{3}(I_i - I_{i-1}) + O(h_i^4). \quad (6.19)$$

However, this expression is now accurate to third order and has only a fourth order error which is as accurate as Simpson's rule. And yet we calculated it using only our results from the trapezoidal rule with hardly any extra work. We are simply reusing the numbers that we have already calculated while carrying out the repeated doubling procedure in Section 6.3.

We can take this process further. Let us refine our notation a little and define

$$R_{i,1} = I_i, \quad R_{i,2} = I_i + \frac{1}{3}(I_i - I_{i-1}) = R_{i,1} + \frac{1}{3}(R_{i,1} - R_{i-1,1}).$$

Then, from Eq. (6.19), we have

$$I = R_{i,2} + c_2 h_i^4 + O(h_i^6). \quad (6.20)$$

where c_2 is another constant and we have made use of the fact that the series for I contains only even powers of h_i . Similarly,

$$I = R_{i-1,2} + c_2 h_{i-1}^4 + O(h_{i-1}^6) = R_{i-1,2} + 16c_2 h_i^4 + O(h_i^6).$$

Since the last two equations both give expressions for I , we can equate them and rearrange to yield

$$c_2 h_i^4 = \frac{1}{15}(R_{i,2} - R_{i-1,2}) + O(h_i^6).$$

Substituting this expression back into Eq. (6.20) gives

$$I = R_{i,2} + \frac{1}{15}(R_{i,2} - R_{i-1,2}) + O(h_i^6).$$

Now we have eliminated the h_i^4 term and generated an estimate accurate to fifth order with a sixth-order error!

We can continue this process, canceling out higher and higher order error terms and getting more and more accurate results. In general, if $R_{i,m}$ is an estimate calculated at the i th round of the doubling procedure and accurate to order h_i^{2m-1} with an error of order h_i^{2m} , then

$$I = R_{i,m} + c_m h_i^{2m} + O(h_i^{2m+2}), \quad (6.21)$$

$$I = R_{i-1,m} + c_m h_{i-1}^{2m} + O(h_{i-1}^{2m+2}) = R_{i-1,m} + 4^m c_m h_i^{2m} + O(h_i^{2m+2}).$$

Equating the two equations and rearranging, we obtain

$$c_m h_i^{2m} = \frac{1}{(4^m - 1)}(R_{i,m} - R_{i-1,m}) + O(h_i^{2m+2}). \quad (6.22)$$

Substituting this result back into Eq. (6.21) yields

$$I = R_{i,m+1} + O(h_i^{2m+2}),$$

where

$$R_{i,m+1} = R_{i,m} + \frac{1}{(4^m - 1)}(R_{i,m} - R_{i-1,m}), \quad (6.23)$$

which is accurate to order h_i^{2m+1} with an error of order h_i^{2m+2} . The calculation also gives us an estimate of the error — Equation (6.22) is precisely the error on $R_{i,m}$ — and hence we can determine the accuracy of our results.

In practice, we do the followings to make use of these results:

1. We calculate our first two estimates of the integral using the regular trapezoidal rule:
 $I_1 = R_{1,1}$ and $I_2 = R_{2,1}$.
2. From the estimates $R_{1,1}$ and $R_{2,1}$, we calculate the more accurate estimate $R_{2,2}$ using Eq. (6.23). This is as much as we can do with only these two starting estimates.
3. Now we calculate the next trapezoidal rule estimate $I_3 = R_{3,1}$ and we calculate $R_{3,2}$ and then $R_{3,3}$ from this estimate using Eq. (6.23).
4. At each successive stage, we compute one more trapezoidal rule estimate $I_i = R_{i,1}$. And we can calculate $R_{i,2}, \dots, R_{i,i}$ from it with very little extra effort.
5. We can also compute the error of each estimate using Eq. (6.22). It allows us to stop the calculation when the error on our estimate of the integral meets the desired target.

To illustrate how this process proceeds, Fig. 6.5 shows which values $R_{i,m}$ are needed to calculate further R s. Each row in the figure lists one trapezoidal rule estimate I_i followed by the other higher-order estimates it allows us to make. The arrows show which previous estimates go into the calculation of each new estimate via Eq. (6.23).

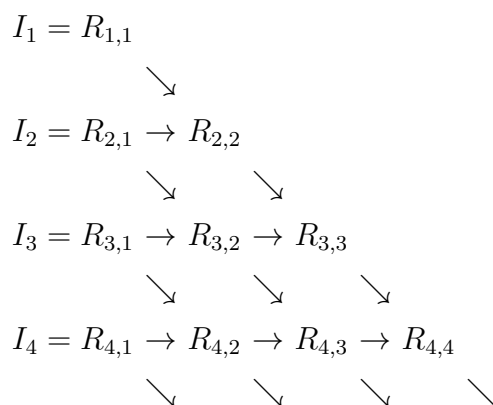


Figure 6.5: The values of $R_{i,m}$ required to calculate further R s using Eq. (6.23).

Notice how each fundamental trapezoidal rule estimate I_i allows us to go one step further with the calculation of $R_{i,m}$. The most accurate estimate we get from the whole process is the very last one. If we do n levels of the process, then the last estimate is $R_{n,n}$ and is accurate to order h_n^{2n-1} .

Errors on our estimates are given by Eq. (6.22). However, we should beware that the equation gives us the error on every estimate *except* the last one in each row (which is the one we really care about). The equation tells us that the error on $R_{n,n}$ would be $(R_{n,n} - R_{n-1,n})/(4^n - 1)$. But there is no $R_{n-1,n}$ and so we cannot use this formula in such case. In practice this means we have to content ourselves with the error estimate for the second last entry in each row, which is normally bigger than the error on the final entry.

The best we can say is that the final entry in the row is our most accurate estimate of the integral and that its error is at least as good as the error for the entry that precedes it, which is given by Eq. (6.22). This is not ideal. But in practice it's usually good enough.

This whole procedure is called **Romberg integration**. It's essentially an "add-on" to our earlier trapezoidal rule scheme: all the tough work is done in the trapezoidal rule calculations and the Romberg integration takes almost no extra computer time (although it does involve extra programming). The payoff is a value for the integral that is accurate to much higher order in h than the simple trapezoidal rule value (or even than Simpson's rule). Moreover, when it's used in an adaptive scheme that halts the calculation once the required accuracy is reached, the time needed to evaluate our integral can be reduced significantly because it reduces the number of trapezoidal rule steps we have to do.

The Romberg integration does have its limitations. We are essentially calculating the value of our integral by making a series expansion in powers of the step size h . It means that the method works best in cases where such power series converge rapidly. If one needs hundreds of terms in the series to get good convergence, then the method is not going to give us any advantage over the simple trapezoidal rule. This will happen if the integrand $f(x)$ is poorly behaved, say, it contains wild fluctuations or singularities, or it is noisy. If your integrand displays these types of properties, then Romberg integration is not a good choice. The simpler adaptive trapezoidal method of Section 6.3 will give better results. However, in cases where the integrand is smooth and well-behaved, Romberg integration can give significantly more accurate results in a remarkably faster time than either the trapezoidal or Simpson's rules.

6.5 Higher-order Integration methods

As we have seen, the trapezoidal rule is based on approximating an integrand $f(x)$ with straight-line segments, while Simpson's rule uses quadratics. We can create higher-order (and hence potentially more accurate) rules by using higher-order polynomials, fitting $f(x)$ with cubics, quartics, and so forth. The general form of the trapezoidal and Simpson's rules is

$$\int_a^b f(x) dx \approx \sum_{k=1}^N w_k f(x_k). \quad (6.24)$$

where x_k are the positions of the sample points at which we calculate the integrand and w_k are some set of weights. In the trapezoidal rule, the first and last weights are $1/2$ and

the others are all 1 (see Eq. (6.2)), while in Simpson's rule the weights are $1/3$ for the first and last slices and alternate between $4/3$ and $2/3$ for the other slices (see Eq. (6.4)).

The basic form for higher-order rules is the same. After fitting to the appropriate polynomial and integrating, we end up with a set of weights that multiply the values $f(x_k)$ of the integrand at evenly spaced sample points. Table 6.1 lists the weights up to quartic order. Higher-order integration rules of this kind are called **Newton-Cotes rules** and in principle they can be extended to any order that we like.

Table 6.1: The weights in the Newton-Cotes rules up to quartic order

Degree	Polynomial	Coefficients
1 (Trapezoidal rule)	Straight line	$\frac{1}{2}, 1, 1, \dots, 1, \frac{1}{2}$
2 (Simpson's rule)	Quadratic	$\frac{1}{3}, \frac{4}{3}, \frac{2}{3}, \frac{4}{3}, \frac{2}{3}, \dots, \frac{4}{3}, \frac{1}{3}$
3	Cubic	$\frac{3}{8}, \frac{9}{8}, \frac{9}{8}, \frac{3}{4}, \frac{9}{8}, \frac{9}{8}, \frac{3}{4}, \dots, \frac{9}{8}, \frac{3}{8}$
4	Quartic	$\frac{14}{45}, \frac{64}{45}, \frac{8}{15}, \frac{64}{45}, \frac{28}{45}, \frac{64}{45}, \frac{8}{15}, \frac{64}{45}, \frac{28}{45}, \dots, \frac{64}{45}, \frac{14}{45}$

But we can do better still. The point to note is that the trapezoidal rule is exact if the function being integrated is actually a straight line since then the straight-line approximation isn't an approximation at all. Similarly, Simpson's rule is exact if the function being integrated is a quadratic, and the k th Newton-Cotes rule is exact if the function being integrated is a degree- k polynomial. But if we have N sample points, then we could probably just fit one $(N - 1)$ th-degree polynomial to the whole integration interval, and get an integration method that is exact for $(N - 1)$ th-degree polynomials.

However, we can do even better than this. Here we have assumed that the sample points are evenly spaced. This has some significant advantages. Methods with evenly spaced points are relatively simple to program. And it's easy to increase the number of points by adding new points half way between the old ones, as we saw in Section 6.3. However, it is also possible to derive integration methods that use unevenly spaced points and they have their own advantages while they lack some of the above advantages. Specifically, they can give very accurate answers with only small numbers of points, making them particularly suitable for cases where we need to do integrals very fast or where evaluation of the integrand itself takes a long time.

Suppose we broaden our outlook into including rules of the form of Eq. (6.24), but where we are allowed to vary not only the weights w_k but also the positions x_k of the sample points. Any choice of positions is allowed, and particularly ones that are not

equally spaced. As we have mentioned, it is possible to create an integration method that is exact for polynomials up to degree $N - 1$ with N equally spaced points. Varying the positions of the points gives us N extra degrees of freedom, which suggests that it might be possible to create an integration rule that is exact for polynomials up to degree $2N - 1$ if all of those degrees of freedom are chosen correctly. For large values of N , this could give us the power to fit functions very accurately and thus can do very accurate integrals. Indeed, it turns out that it's possible to do this and the developments lead to the superbly accurate integration method known as **Gaussian quadrature**. But we will not discuss this method here as it's out of our current scope.

6.6 Choosing a Numerical Integration Method

We have studied a number of different integration methods in this chapter: the trapezoidal rule and Simpson's rule as well as adaptive versions of both plus Romberg integration. (Here we don't consider the rectangular method as it's usually a poor approximation.) You might ask at this point which one of these methods is the best? Which one should I use in practice if I need to evaluate an integral?

There is no unique answer to this question. Which method you should use depends on the particular problem that you are working on. However, a good general principle is that higher-order methods such as Romberg integration work best when applied to smooth, well-behaved functions. If your function is not smooth or poorly behaved in some way, then simpler methods, and particularly the trapezoidal rule, are the way to go. The reason is that any integration method knows the value of the integrand only at its sample points. If the integrand varies rapidly between the sample points, then such variation will not be reflected in the computed value of the integral, which can lead to inaccurate results. If you're evaluating an integral using only ten or twenty sample points, then it's crucial that those points give a good picture of the integrand. If it does not, then high-order methods using few sample points will not do a good job.

Bearing this principle in mind, here is a guide to the kinds of problems each of our integration methods is good for.

1. **The trapezoidal rule:** The trapezoidal rule is trivial to program and so is a good choice when you need a quick answer for an integral. It's not very accurate, but sometimes you don't need great accuracy. It uses equally spaced sample points which is appropriate for problems such as integrating data from laboratory experiments that

are sampled at uniform time intervals. The trapezoidal rule is also a good choice for poorly behaved functions, i. e. those that vary widely, contain singularities, or are noisy. It's usually a better choice for such functions than the other methods we have considered. In its adaptive form, it can also give us a guaranteed accuracy for an integral, although it may take more computer time to achieve that accuracy than other methods.

2. **Simpson's rule:** Simpson's rule has many of the benefits of the trapezoidal rule such as simplicity of programming and equally spaced sample points. It gives greater accuracy than the trapezoidal rule with the same number of sample points or the same accuracy with fewer points. But it relies on higher-order approximation of the integrand, which can lead to problems if the integrand is noisy or otherwise not smooth. So we should use it with caution if you are unsure of the nature of your integrand. Its adaptive form provides a result of guaranteed accuracy and does so faster than the equivalent trapezoidal rule calculation. But again it may be less suitable for poorly behaved integrands.
3. **Romberg integration:** When using equally spaced sample points, Romberg integration is a typical higher-order integration method. It gives remarkably accurate estimates of integrals with a minimum number of sample points, plus error estimates that allow you to halt the calculation once you have achieved the desired accuracy. Since it relies extrapolating answers from measurements of the integrand at only a few points, Romberg integration will not work well for wildly varying integrands, noisy integrands, or integrands with mathematically pathological behaviors such as singularities. It is best applied to smooth functions whose form can be determined accurately from only a small number of sample points.

Armed with these guidelines, you should be able to choose a suitable integration method for most problems you come up against in scientific computation.

6.7 Numerical Derivatives

As you have seen in the calculus books, the standard definition of a derivative is

$$\frac{df}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}.$$

The basic method for calculating numerical derivatives is precisely an implementation of this formula. Of course, we can't take the limit $h \rightarrow 0$ in practice. But we can make h

very small and then compute

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x)}{h}. \quad (6.25)$$

This approximation to the derivative is called the **forward difference** as it is measured in the forward (i. e. positive) direction from the point of interest x . You can think of it in geometric terms as shown in Fig. 6.6. It's simply the slope of the curve $f(x)$ measured over a small interval of width h in the forward direction from x .

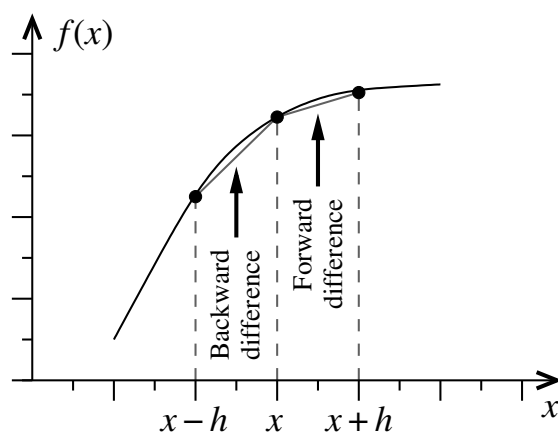


Figure 6.6: The forward and backward differences provide two different approximations to the derivative of a function $f(x)$ at the point x .

There is also the **backward difference**, which has the mirror image definition:

$$\frac{df}{dx} \approx \frac{f(x) - f(x-h)}{h}.$$

The forward and backward differences typically give almost the same answer and in many cases you can use either. Most often one uses the forward difference. There are a few special cases where one is preferred over the other, particularly when there is a discontinuity in the derivative of the function at the point x or when the domain of the function is bounded and you want the value of the derivative on the boundary, in which case only one or other of the two difference formulas will work.

Before using either the forward or backward difference, we must choose a value for h . To work out what the best value is we need to look at the errors and inaccuracies involved in calculating numerical derivatives. Calculations of derivatives using forward and backward differences are not perfectly accurate. There are two sources of error. The first one is a round-off error that has been discussed in Section 5.3. The second one is the truncation error that arises because we cannot take the limit $h \rightarrow 0$ and so our differences are not really true derivatives. By contrast with numerical integrals where

round-off error is usually negligible, it turns out that both sources of error are important when we calculate a numerical derivative.

To understand why both round-off and truncation errors are important, let us focus on the forward difference and consider the Taylor expansion of $f(x)$ about x :

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2f''(x) + \dots$$

where f' and f'' denote the first and second order derivatives of f . Rearranging this expression, we obtain

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{1}{2}hf''(x) + \dots \quad (6.26)$$

When we calculate the forward difference, we calculate only the first part on the right-hand side and neglect the term in $f''(x)$ plus all higher-order terms. The size of these neglected terms measures the truncation error on the forward difference. Thus, to leading order in h , the absolute magnitude of the truncation error is $(1/2)h|f''(x)|$. It is linear in h so that we should get more accurate answers if we use smaller values of h .

But now a problem arises: subtracting numbers from one another on a computer can give rise to big round-off errors (in fractional terms) if the numbers are close to one another. And that's exactly what happens here — the numbers $f(x+h)$ and $f(x)$ that we are subtracting will be very close to one another if h is very small. Thus we will get a large round-off error in our result if we make h too small. This puts us in a difficult situation: we want to make h small to make the forward difference approximation as accurate as possible, but if we make it too small we will get a large round-off error. To get the best possible answer, we need to find a compromise.

In Section 5.2, we saw that the computer can typically calculate a number $f(x)$ to an accuracy of $Cf(x)$ where the constant $C \approx 10^{-16}$ in Python. Since $f(x+h)$ is normally close in value to $f(x)$, the accuracy of their values will be also about the same. So the absolute magnitude of the total round-off error on $f(x+h) - f(x)$ will be about $2C|f(x)|$ in the worst case. It might be better than this if the two errors go in opposite directions and happen to cancel out. But we cannot assume that this will be the case. So the worst-case round-off error on the forward difference, Eq. (6.25), will be $2C|f(x)|/h$.

Meanwhile, the truncation error is about $(1/2)h|f''(x)|$ according to Eq. (6.26). Thus the total error ϵ on our estimate of the derivative (in the worst case) is

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{2}h|f''(x)|. \quad (6.27)$$

We want to find the value of h that minimizes this error. So we differentiate the above expression with respect to h and set the result equal to zero, which gives

$$\begin{aligned} -\frac{2C|f(x)|}{h^2} + \frac{1}{2}|f''(x)| &= 0 \\ \Rightarrow h &= \sqrt{4C \left| \frac{f(x)}{f''(x)} \right|} \end{aligned}$$

Substituting this result back into Eq. (6.27), we find that the minimum error on our derivative is

$$\epsilon = h|f''(x)| = \sqrt{4C|f(x)f''(x)|}. \quad (6.28)$$

Thus, for instance, if $f(x)$ and $f''(x)$ are of order 1, then we should choose h to be roughly of order \sqrt{C} , which is typically about 10^{-8} , and the error of our final result will be also about $\sqrt{C} \approx 10^{-8}$. A similar analysis can be applied to the backward difference and it gives the same final result. Thus we can at the best get about half of the usual numerical precision on our derivatives if we compute them using the forward or backward differences. If the precision is about 16 digits as in Python, then we can get 8 digits of precision on our derivatives. This is substantially poorer than most of the calculations we have seen so far and could be a significant source of error for calculations that require high accuracy.

We have seen that forward and backward differences are not very accurate. What can we do to improve the situation? A simple improvement is to use the **central difference**:

$$\frac{df}{dx} \approx \frac{f(x+h/2) - f(x-h/2)}{h}. \quad (6.29)$$

The central difference is similar to the forward and backward differences, approximating the derivative using the difference between two values of $f(x)$ at points a distance h apart. What's the difference is that the two points are now placed symmetrically around x , one at a distance $(1/2)h$ in the forward direction and the other at a distance $(1/2)h$ in the backward direction.

To calculate the truncation error on the central difference, we consider the Taylor series expansion

$$f\left(x \pm \frac{h}{2}\right) = f(x) \pm \frac{1}{2}hf'(x) + \frac{1}{8}h^2f''(x) \pm \frac{1}{48}h^3f'''(x) + \dots$$

Manipulating on this expansion, we obtain the expression for $f'(x)$:

$$f'(x) = \frac{f(x+h/2) - f(x-h/2)}{h} - \frac{1}{24}h^2f'''(x) + \dots$$

To leading order in h , the magnitude of the error is now $(1/24)h^2|f'''(x)|$, which is one order in h higher than before. Just like before, there is also a round-off error whose size is the same as in our previous calculation, having magnitude $2C|f(x)|/h$. So the magnitude of the total error on our estimate of the derivative (in the worst case) is

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{24}h^2|f'''(x)|. \quad (6.30)$$

Differentiating this expression to find its minimum and rearranging, we find that the optimal value of h is

$$h = \left(24C \left| \frac{f(x)}{f'''(x)} \right| \right)^{1/3}$$

Substituting this back into Eq. (6.30), we find the minimum error to be

$$\epsilon = \frac{1}{8}h^2|f'''(x)| = \left(\frac{9}{8}C^2[f(x)]^2|f'''(x)|\right)^{1/3}. \quad (6.31)$$

Thus, for instance, if $f(x)$ and $f''(x)$ are of order 1, then the ideal value of h is going to be around $C^{1/3}$, which is typically about 10^{-5} , and the error of our final result will be about $C^{2/3} \approx 10^{-10}$. Thus the central difference is indeed more accurate than the forward and backward differences, by a factor of 100 or so in this case, although we can achieve this accuracy by using a larger value of h . This may seem slightly surprising, but it is the correct result.



Example 6.5

As an sample application of the central difference, suppose we are given the values of a function $f(x)$ measured at evenly spaced sample points a distance h apart as shown in Fig. 6.7.

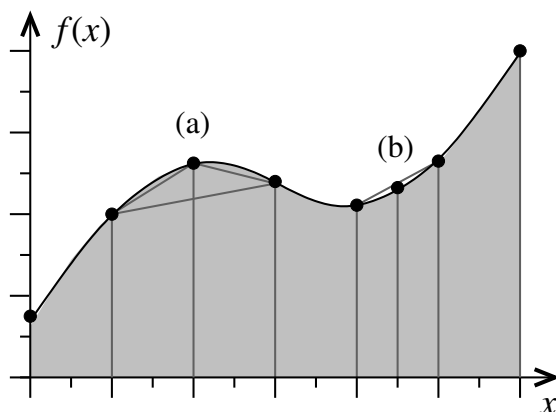


Figure 6.7: Computing the numerical derivative of a sample function using different approaches.

Now suppose we want to calculate the derivative of f at one of these points (case (a) in Fig. 6.7). We could use a forward or backward difference based on the sample at x and one of the adjacent ones, or we could use a central difference. However, if we use a central difference, which is based on points equally spaced on either side of x , then we must use the points at $x + h$ and $x - h$. We cannot use points at $x + h/2$ and $x - h/2$ as in Eq. (6.29) because there are no such points. The formula for the central difference in this case will thus be

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x-h)}{2h}. \quad (6.32)$$

This means that the interval between the points we use is $2h$ for the central difference, but only h for the forward and backward differences. So which will give a better answer? The central difference because it's a better approximation or the forward difference because of its smaller interval?

From Eq. (6.28), we can see that the error on the forward difference is $h|f''(x)|$. Besides, from Eq. (6.31), we see that the error on the central difference is $h^2|f'''(x)|/2$ with h replaced by $2h$. Which is smaller depends on the value of h . For the central difference to give the more accurate answer, we require $h^2|f'''(x)|/2 < h|f''(x)|$, i. e.

$$h < 2 \left| \frac{f''(x)}{f'''(x)} \right|$$

If h is larger than this, then the forward difference is actually the better approximation in this case.

But now suppose that instead of calculating the value of the derivative at one of the sample points itself, we want to calculate it at a point x lying halfway between two of the samples (case (b) in Fig. 6.7). Viewed from that point, we do have samples at $x + h/2$ and $x - h/2$. So now we can use the original form of the central difference, Eq. (6.29), with an interval only h wide like the forward difference. This calculation will give a more accurate answer, but only at the expense of calculating the result at a point between the samples.

One way to think about the numerical derivatives we have seen so far is that we are fitting a straight line through two points, such as the points $f(x)$ and $f(x+h)$, and then asking about the slope of that line at the point x . The trapezoidal rule does a similar thing for integrals, approximating a curve by a straight line between sample points and estimating the area under the curve using that line. We saw that we can make a higher-order (and usually better) approximation to an integral by fitting a quadratic

or higher-order polynomial instead of a straight line, and this led to the Simpson's and Newton-Cotes rules for integrals. We can take a similar approach with derivatives by fitting a polynomial to a set of sample points and then calculating the derivative of the polynomial at x .

For example, consider fitting a quadratic curve $y = ax^2 + bx + c$ to the function $f(x)$. We require three sample points to make the fit and the best results are obtained by placing the points symmetrically about the point of interest x as with the central difference. Suppose that we are interested in the derivative at $x = 0$. So we place our three points at $-h$, 0 , and $+h$, for some h that we choose. Requiring that our quadratic is equal to $f(x)$ at these three points gives us three equations thus:

$$ah^2 - bh + c = f(-h), \quad c = f(0), \quad ah^2 + bh + c = f(h). \quad (6.33)$$

In principle, we can now solve these equations for the three parameters a , b , and c . But we don't need the whole solution in this case because we don't need all of the parameters. Given the quadratic fit $y = ax^2 + bx + c$, the derivative of the curve at the point $x = 0$ is

$$\frac{dy}{dx} = [2ax + b]_{x=0} = b$$

So we need only the one parameter b , which we can get from Eq. (6.33) by subtracting the first equation from the third to give $2bh = f(h) - f(-h)$ and rearranging. Thus our approximation for the derivative at $x = 0$ is

$$\frac{df}{dx} \approx \frac{f(h) - f(-h)}{2h}.$$

We have done this calculation for the derivative at $x = 0$. However, the same result applies at any other point — we can slide the whole function up or down the x -axis to put any point x at the origin and then calculate the derivative from the formula above. Thus we can just write

$$\frac{df}{dx} \approx \frac{f(x+h) - f(x-h)}{2h} \quad (6.34)$$

for general x . This is the correct result for the quadratic approximation. But it's a disappointing result since Eq. (6.34) is nothing other than the central difference approximation for sample points $2h$ apart, which we already saw in Eq. (6.32). It means that the higher-order approximation has not helped us in this case.

However, going to higher orders does help. If we use a cubic or quartic approximation, we will get improved estimates of the derivative. At higher orders, there is a distinction between the odd- and even-order approximations. For the odd-order ones, the sample

points fall at “half-way” points just like the central difference of Eq. (6.29). For instance, we will choose the four sample points for the cubic approximation to fall at $x = -3h/2, -h/2, h/2$, and $3h/2$ that are symmetrically distributed about zero. On the other hand, the samples fall at “integer” points for even-order approximations. For instance, the five sample points for the quartic approximation fall at $-2h, -h, 0, h$, and $2h$.

The methodology for deriving the higher-order approximations follows the same pattern as for the quadratic case: we write down the required value of the polynomial at each of the sample points, which gives us a set of simultaneous equations in the polynomial coefficients. As before, we actually need only one of those coefficients — the coefficient of the linear term in the polynomial. Solving for this coefficient gives us our expression for the derivative. At each order the expression is a linear combination of the samples divided by h . We will not go through the derivations in detail. Table 6.2 gives the coefficients of the combinations for the first five approximations.

Table 6.2: The coefficients for central approximations to $f'(x)$ at $x = 0$

Degree	$f(-\frac{5}{2}h)$	$f(-2h)$	$f(-\frac{3}{2}h)$	$f(-h)$	$f(-\frac{1}{2}h)$	$f(0)$	$f(\frac{1}{2}h)$	$f(h)$	$f(\frac{3}{2}h)$	$f(2h)$	$f(\frac{5}{2}h)$	Error
1					-1		1					$O(h^2)$
2				$-\frac{1}{2}$				$\frac{1}{2}$				$O(h^2)$
3			$\frac{1}{24}$		$-\frac{27}{24}$		$\frac{27}{24}$		$-\frac{1}{24}$			$O(h^4)$
4		$\frac{1}{12}$		$-\frac{2}{3}$				$\frac{2}{3}$		$-\frac{1}{12}$		$O(h^4)$
5	$-\frac{3}{640}$		$\frac{25}{384}$		$-\frac{75}{64}$		$\frac{75}{64}$		$-\frac{25}{384}$		$\frac{3}{640}$	$O(h^6)$

Each of the approximations given in Table 6.2 is exact, apart from a round-off error, if the function being differentiated is actually a polynomial of the appropriate (or lower) degree so that the polynomial fit is a perfect one. However, this will not be the case for most of the time and there will be a truncation error involved in calculating the derivative. One can calculate this error to leading order of h for each of the approximations by a method analogous to our calculations for the forward, backward, and central differences: we perform Taylor series expansions about $x = 0$ to derive expressions for $f(x)$ at each of the sample points and then plug these expressions into the formula for the derivative. The order in h of the resulting error is listed in the final column of Table 6.2. Just like before, this truncation error must be balanced against the round-off error and a suitable value of h is chosen to minimize the overall error in the derivative.

An interesting point to note about Table 6.2 is that the coefficient for $f(0)$ in all

the approximations is zero. The value of the function exactly at the point of interest never plays a role in the evaluation of the derivative. Another point is that the order in h of the error given in the final column does not go up uniformly with the degree of the polynomial. It is the same for the even-degree polynomials as for the next-lower odd-degree ones. We saw a special case of this result for the quadratic: the quadratic fit just gives us an ordinary central difference and therefore must have an error $O(h^2)$ which is the same as the central difference derived from the linear fit. In general, the odd-degree approximations give us slightly more accurate results than the even-degree ones — the error is of the same order in h but the constant of proportionality is smaller. In addition, the odd-degree approximations require samples at the half-way points which can be inconvenient. As discussed in Example 6.5, we sometimes have samples at only the “integer” points, in which case we must use the even-degree approximations.

We can also derive numerical approximations for the second-order derivative of a function $f(x)$. By definition, the second-order derivative is the derivative of the first-order derivative. So we can calculate it by applying our first-order derivative formulas twice. For example, starting with the central difference formula, Eq. (6.29), we can write expressions for the first-order derivative at $x + h/2$ and $x - h/2$ as:

$$f'\left(x + \frac{h}{2}\right) \approx \frac{f(x+h) - f(x)}{h}, \quad f'\left(x - \frac{h}{2}\right) \approx \frac{f(x) - f(x-h)}{h}.$$

Then we apply the central difference again to get an expression for the second-order derivative:

$$f''(x) \approx \frac{f'(x+h/2) - f'(x-h/2)}{h} \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (6.35)$$

This is the simplest approximation for the second derivative.

We can also calculate the error on Eq. (6.35). We perform two Taylor series expansions of $f(x)$:

$$f(x \pm h) = f(x) \pm hf'(x) + \frac{1}{2}h^2f''(x) \pm \frac{1}{6}h^3f'''(x) + \frac{1}{24}h^4f''''(x) + \dots$$

Adding them together and rearranging, we find that

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}h^2f''''(x)$$

The first term on the right is our formula for the second-order derivative, Eq. (6.35), and the remaining terms measure the error. Thus the absolute error inherent in our approximation to the second-order derivative is $(1/12)h^2|f''''(x)|$ to leading order of h . We also need to take round-off error into account which contributes an error of roughly $C|f(x)|$

on each value of $f(x)$ so that the total round-off error in the numerator of Eq. (6.35) is $4C|f(x)|$ in the worst case and the round-off error on the whole expression is $4C|f(x)|/h^2$.

Then the complete error on the derivative is

$$\epsilon = \frac{4C|f(x)|}{h^2} + \frac{1}{12}h^2|f'''(x)| \quad (6.36)$$

Differentiating this expression with respect to h and setting the result to zero then gives an optimum value of h of

$$h = \left(48C \left| \frac{f(x)}{f'''(x)} \right| \right)^{1/4}$$

Substituting this expression back into Eq. (6.36) gives the size of the optimal error:

$$\epsilon = \frac{1}{6}h^2|f'''(x)| = \sqrt{\frac{4}{3}C|f(x)f'''(x)|}$$

Thus, for instance, if $f(x)$ and $f'''(x)$ are of order 1, then the optimal error will be roughly of order \sqrt{C} , which is typically about 10^{-8} . This is about the same accuracy as we found for the forward and backward difference approximations to the first derivative. Therefore, our expression for the second derivative is not very accurate. It is about as good as, but not better than, the forward difference. As mentioned above, there are higher-order approximations for the second derivative that can give more accurate answers. But Eq. (6.35) will be good enough for most of our purposes.

Example 6.6

Suppose an electron in the $2s$ state of the hydrogen atom whose radial probability density is equal to

$$P(r) = \frac{r^2}{8a_0^3} \left(4 - \frac{4r}{a_0} + \frac{r^2}{a_0^2} \right) e^{-r/a_0}$$

where $a_0 = (4\pi\epsilon_0\hbar^2)/(m_e e^2)$ is the Bohr radius. The radial probability density gives the probability per thickness of finding the electron within a thin spherical shell of radius r . So the most probable distance r_p of the electron from the nucleus is the value of r where $P(r)$ attains the maximum value. Below is the Python program `mostprobdist.py` which proves that $r_p = (3 + \sqrt{5})a_0$ for the electron by showing $P'(r_p) = 0$ and $P''(r_p) < 0$ using the central difference with step size $h = 0.00001a_0$.

```
# mostprobdist.py
# This program proves that  $r_p = (3 + \sqrt{5})a_0$  for the electron by
# showing  $P'(r_p) = 0$  and  $P''(r_p) < 0$  using the central difference
```

```

# with step size h = 0.00001a0. Here we use the transformation
# x = r/a0 for convenience.
# Written on 7 Jan 2022 by F K Chow

from numpy import exp,isclose,sqrt

def P(x):
    """ Calculate the  $P(x) = x^2(4 - 4x + x^2)e^{-x}/8$  """
    return x**2*(4 - 4*x + x**2)*exp(-x)/8.0

def cdiff1(P, x, h):
    """ Calculate first-order derivative of P(x) using the central
        difference with step size h """
    return (P(x + 0.5*h) - P(x - 0.5*h))/h

def cdiff2(P, x, h):
    """ Calculate second-order derivative of P(x) using the central
        difference with step size h """
    return (P(x + h) - 2*P(x) + P(x - h))/h**2

# Calculate P'(rp) and P''(rp) using the central difference
xp = 3 + sqrt(5)
h = 0.00001
dPdr = cdiff1(P, xp, h)
dP2dr2 = cdiff2(P, xp, h)

# Output the results of the calculation
print("For rp = {:.5f}a0, ".format(xp), end="")
print("P'(rp) = {:.5f}, P''(rp) = {:.5f}.".format(dPdr, dP2dr2))
print("It implies that rp is ", end="")
if (isclose(dPdr, 0) == 0) or (dP2dr2 >= 0):
    print("NOT ", end="")
print("the most probable distance of the electron\nfrom the nucleus.")

```

Here is the output of this program which gives the expected result.

and the distance marked y on Fig. 6.8 is given in terms of this slope by $y = m(x - a)$. The distance marked z is equal to $f(a)$. So we find that

$$f(x) \approx y + z = \frac{(b - x)f(a) + (x - a)f(b)}{b - a} \quad (6.37)$$

This is the fundamental formula of linear interpolation. In fact, this same formula can also be used to *extrapolate* the function to points outside the interval from a to b , although one should not extrapolate too far. The further you go, the less likely that the extrapolation would be accurate.

How accurate is the linear interpolation formula? The calculation of the error is similar to that for derivatives, making use of two Taylor expansions:

$$\begin{aligned} f(a) &= f(x) + (a - x)f'(x) + \frac{1}{2}(a - x)^2 f''(x) + \dots \\ f(b) &= f(x) + (b - x)f'(x) + \frac{1}{2}(b - x)^2 f''(x) + \dots \end{aligned}$$

Substituting these into Eq. (6.37), the terms in $f'(x)$ cancel. After rearranging the resulting expression, we find that

$$f(x) = \frac{(b - x)f(a) + (x - a)f(b)}{b - a} + \frac{1}{2}(a - x)(b - x)f''(x) + \dots$$

The first term on the right-hand side is our linear interpolation formula while the remaining terms are the error. Note that the leading-order error term vanishes as x tends to either a or b so that either $b - x$ or $a - x$ becomes small. And assuming $f''(x)$ varies slowly, the error will be largest in the middle of the interval. If we denote the width of the interval by $b - a = h$, then when we are in the middle we have $x - a = b - x = (1/2)h$ and the magnitude of the leading-order error is $(1/8)h^2|f''(x)|$. Thus, like the central difference formula for a first-order derivative, the worst-case error on a linear interpolation is $O(h^2)$. So we can make the interpolation more accurate by making h smaller.

By contrast with the case of derivatives, we do not need to be particularly careful about round-off error when using linear interpolation. The interpolation formula, Eq. (6.37), involves the sum of values of $f(x)$ at two closely spaced points instead of the difference. So we don't normally run into the accuracy problems that plague calculations that are based on subtractions.

Can we do better than linear interpolation? We cannot do better if the value of the function $f(x)$ is known at only two points as there is no better approximation in such case. However, if we know the function at more than two points, then there are several ways to improve on linear interpolation. The most obvious one is to interpolate

using higher-order polynomials. For example, if we have three points, then we can fit a quadratic through them, which will usually give a better match to the underlying curve. Fitting quadratics or higher polynomials leads to a set of higher-order methods known as **Lagrange interpolation** methods.

But this approach breaks down when the number of points becomes large. If we have a large number N of points, then you might think the best thing to do would be to fit an $(N - 1)$ th order polynomial through them. But it turns out this doesn't work because very high order polynomials tend to fluctuate rapidly between them and can deviate from the fitted points badly in the intervals between points. In this case, it's better to fit many lower-order polynomials such as quadratics or cubics to smaller sets of adjacent points. Unfortunately, the naive implementation of such a scheme gives rather uneven interpolations because the slope of the interpolation changes at the join-points between polynomials. An even better approach is to fit polynomials to the measured points and the derivatives at their ends so that one gets a function that goes through the points and has a smooth slope everywhere. Such interpolations are called **splines**. The most widely used type are **cubic splines**. But we won't go into these methods further here. For most of our purposes, linear interpolation will be good enough.