13 January 2025 – PHYS 6260
Prof. John Wise
Chapter 5: Integrals and Derivatives

After covering the capabilities of programs for scientific computing, we can turn our attention to the fundamentals of it — evaluating integrals and derivatives. While some domains and physical processes can be described by an exact functional form, others do not being dependent on the derivatives and integrals of the variables inside the system. Here in this chapter we will investigate different techniques for such operations.

5.1: Fundamental Methods for Evaluating Integrals

Let's start from the most simple example, evaluating an integral of a known function over some finite range. We will study a range of methods, but we will start with the most straightforward, the **trapezoidal rule**.

The Trapezoidal Rule

Consider a function $f(x)$ that we want to integrate from a to b, which we denote as

$$I(a,b) = \int_a^b f(x)\,dx \tag{1}$$

There is no known way to exactly computationally calculate an integral, but there are several approximate methods. Consider splitting up the area under the curve into N pieces. The most simple approach is to consider these pieces as rectangles, shown in the left panel of Figure 1. However, this is a pretty poor approximation as seen in the figure. We can better

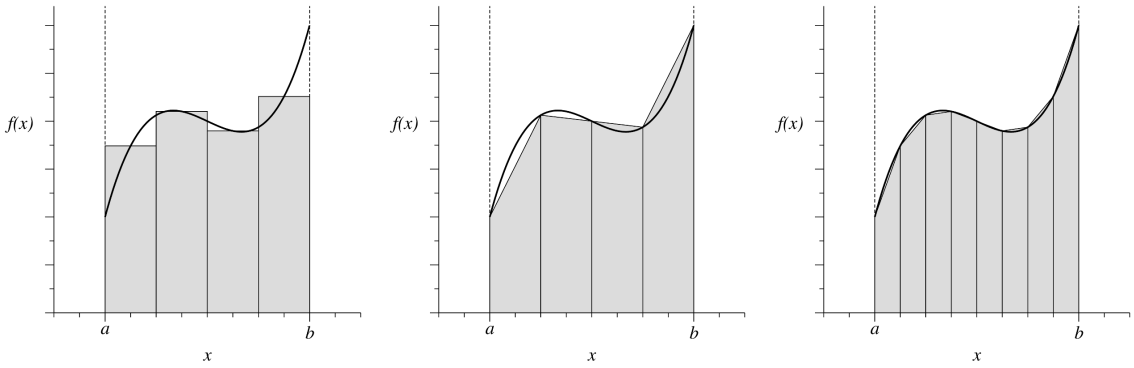

Figure 1: Estimating the area under a curve: (left) rectangular slices, (center) trapezoidal rule, (right) trapezoidal rule with more slices.

approximate the area in the slice with little effort by considering a trapezoid as the slice, as

shown in the middle plot of Figure 1. Each slice has a width $h = (b-a)/N$ with the k-th slice having edges at $x = a + (k-1)h$ and $x = a + kh$, resulting in an area of

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

which is the basis of the **trapezoidal rule**. Extending this to the entire integral, we sum all of the slices, giving

$$\begin{aligned} I(a,b) \simeq \sum_{k=1}^{N} A_k &= \frac{h}{2}\sum_{k=1}^{N}[f(a + (k-1)h) + f(a + kh)] &(3)\\ &= h\left[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \ldots + \frac{1}{2}f(b)\right] &(4)\end{aligned}$$

$$\boxed{I(a,b) \simeq h\left[\frac{1}{2}(f(a) + f(b)) + \sum_{k=1}^{N-1} f(a + kh)\right]} \tag{5}$$

Notice that this value is just the summation of all interior slices and half of the value of the boundary slices.

Example 5.1: Integrating a function

Let's use the trapezoidal rule to compute the integral of $f(x) = x^4 - 2x + 1$ from $x = [0, 2]$. We can do this by hand (the exact answer is 4.4), but it's a good example to test the accuracy of the trapezoidal rule because we have an exact result to compare against. Here's a program that performs the integral (in a loop for clarity) with $N = 10$,

```
def f(x):
    return x**4 - 2*x + 1

N = 10
a = 0.0
b = 2.0
h = (b-a)/N

s = 0.5 * (f(a) + f(b))
for k in range(1,N):
    s += f(a + k*h)
print(h*s)
```

Running this program gives the result `4.50656`, which is off by about 2%, which is decent for such a small number of slices. Increasing N to 100, we obtain `4.40107` (0.02% accuracy) and with $N = 1000$, we find the integral is `4.40001` (2×10^{-4} accuracy).

Simpson's Rule

The trapezoidal rule is perfectly adequate for most calculations when great accuracy isn't needed, but we can approximate the area under the curve better by creating a parabolic function that approximates the function around the k-th point. This is especially useful for functions that rapidly vary, where increasing N does not greatly help the accuracy of the trapezoidal rule.

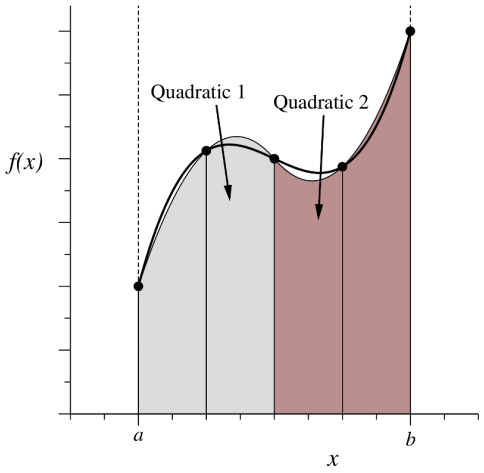

Figure 2: Simpson's rule with the same function as Figure 1.

To create a parabolic representation of the function around the k-th point, we need three points instead of two as with the trapezoidal rule. Fitting a parabola $Ax^2 + Bx + C$ through the points at $x = -h, 0, h$, we have

$$f(-h) = Ah^2 - Bh + C, \quad f(0) = C, \quad f(h) = Ah^2 + Bh + C. \tag{6}$$

Solving for the constants, we find

$$A = \frac{1}{h^2}\left[\frac{1}{2}f(-h) - f(0) + \frac{1}{2}f(h)\right], \quad B = \frac{1}{2H}\left[f(h) - f(-h)\right], \quad C = f(0). \tag{7}$$

The integral under this curve from $-h$ to $+h$ is given by

$$\int_{-h}^{h} (Ax^2 + Bx + C)\, dx = \frac{2}{3}Ah^3 + 2Ch = \frac{h}{3}[f(-h) + 4f(0) + f(h)], \tag{8}$$

which is the basis for **Simpson's rule**. Note that this expression doesn't depend on the location of x, only that the points are equally spaced. Thus, we can slide this parabola around different x-values, and it would still hold. Using this idea, we can apply Equation (8) to any x-value and evaluate the integral from a to b, summing all of the slices,

$$I(a,b) \simeq \frac{h}{3}[f(a) + 4f(a+h) + f(a+2h)] \tag{9}$$

$$+ \frac{h}{3}[f(a+2h) + 4f(a+3h) + f(a+4h)] + \ldots \tag{10}$$

$$+ \frac{h}{3}[f(a+(N-2)h) + 4f(a+(N-1)h) + f(b)]. \tag{11}$$

Note that Simpson's rule requires an even number of slices. Collecting terms, we obtain

$$I(a,b) \simeq \frac{h}{3}[f(a)+4f(a+h)+2f(a+2h)+4f(a+3h)+\ldots+f(b)] \tag{12}$$

$$= \frac{h}{3}\left[f(a)+f(b)+4\sum_{\substack{\text{odd } k \\ 1\ldots N-1}} f(a+kh)+2\sum_{\substack{\text{even } k \\ 2\ldots N-2}} f(a+kh)\right] \tag{13}$$

The sums over even and odd terms of the function can either be evaluated in a loop with the iterators `range(1,N,2)` and `range(2,N,2` or in array arithmetic with `f(x[1::2])` and `f(x[2::2])`, where `x = np.linspace(0,2,N+1)` is an array with equally spaced values of x. Alternatively, one can loop through the N slices, evaluating the following summation

$$I(a,b) \simeq \frac{h}{3}\left[f(a)+f(b)+4\sum_{k=1}^{N/2} f(a+(2k-1)h)+2\sum_{k=1}^{N/2-1} f(a+2kh)\right]. \tag{14}$$

Comparing this to the trapezoidal rule, it is a little more complicated but not too much. It is still straightforward to create a program that uses Simpson's rule.

Applying Simpson's rule to Example 5.1, we find that the result is `4.400427` with $N = 10$, which is accurate to less than 0.01%, orders of magnitude better than the trapezoidal rule. Results with higher values of N are even better. Depending on the complexity of the function and required accuracy, either the trapezoidal or Simpson's rule are valid choices, but Simpson's rule is usually the best choice because it requires less operations and gives better accuracy.

5.2: Errors on Integrals

The errors that come from numerical integration are *approximation errors* not rounding errors as we explored in Chapter 4. This is apparent in Figures 1 and 2 the slices don't exactly follow the function. We can estimate these errors by calculating the difference between the slices and a Taylor expansion around the k-th point at $x_k = a + kh$. The Taylor expansion around x_{k-1} is given by

$$f(x) = f(x_{k-1}) + (x-x_{k-1})f'(x_{k-1}) + \frac{1}{2}(x-x_{k-1})^2 f''(x_{k-1}) + \ldots \tag{15}$$

where f' and f'' denote that first and second spatial derivatives of $f(x)$. We can integrate the Taylor expansion from x_{k-1} to x_k, using the substitution $u = x - x_{k-1}$,

$$\begin{aligned}\int_{x_{k-1}}^{x_k} f(x)\,dx &= f(x_{k-1})\int_{x_{k-1}}^{x_k} dx + f'(x_{k-1})\int_{x_{k-1}}^{x_k}(x-x_{k-1})dx + \\ &\quad \frac{1}{2}f''(x_{k-1})\int_{x_{k-1}}^{x_k}(x-x_{k-1})^2dx + \ldots \qquad (16)\\ &= f(x_{k-1})\int_0^h du + f'(x_{k-1})\int_0^h u du + \frac{1}{2}f''(f_{k-1})\int_0^h u^2 du + \ldots \qquad (17)\\ &= hf(x_{k-1}) + \frac{1}{2}h^2 f'(x_{k-1}) + \frac{1}{6}h^3 f''(x_{k-1}) + O(h^4) \qquad (18)\end{aligned}$$

where $O(h^4)$ denotes the rest of the series with orders higher than h^4.

To obtain a better estimate, we will take the average of the integral with Taylor expansions around x_{k-1} and x_k. Performing the same exercise around x_k instead of x_{k-1}, we find that

$$\int_{x_{k-1}}^{x_k} f(x)dx = hf(x_k) - \frac{1}{2}h^2 f'(x_k) + \frac{1}{6}h^3 f''(x_k) + O(h^4) \qquad (19)$$

The average of these two integrals is

$$\begin{aligned}\int_{x_{k-1}}^{x_k} f(x)dx &= \frac{1}{2}h[f(x_{k-1}) + f(x_k)] + \frac{1}{4}h^2[f'(x_{k-1}) - f'(x_k)] + \qquad (20)\\ &\quad \frac{1}{12}h^3[f''(x_{k-1}) + f''(x_k)] + O(h^4)\end{aligned}$$

Finally we can sum this expression over all slices,

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{k=1}^{N}\int_{x_{k-1}}^{x_k} f(x)dx \qquad (21)\\ &= \frac{1}{2}h\sum_{k-1}^{N}[f(x_{k-1}) + f(x_k)] + \frac{1}{4}h^2[f'(a) - f'(b)] + \\ &\quad \frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1}) + f''(x_k)] + O(h^4)\end{aligned}$$

By inspecting this sum, we see that the trapezoidal rule is just the first sum, where the additional terms sum to the approximation error. The second term only depends on the first spatial derivative at the ends and doesn't depend on the interior values. A similar behavior happens at other even powers of h.

We can interpret the h^3 term by seeing that the sum is just the trapezoidal rule for $f''(x)$,

$$\int_a^b f''(x)dx = \frac{1}{2}h\sum_{k-1}^{N}[f''(x_{k-1}) + f''(x_k)] + O(h^2) \qquad (22)$$

To express the h^3 term in Equation 21, we multiply by $h^2/6$ and re-arrange

$$\frac{1}{12}h^3\sum_{k=1}^{N}[f''(x_{k-1})+f''(x_k)] = \frac{1}{6}h^2\int_a^b f''(x)dx + O(h^4) \quad (23)$$

$$= \frac{1}{6}h^2[f'(b)-f'(a)] + O(h^4), \quad (24)$$

where the last step is valid because the integral of $f''(x)$ is just $f'(x)$. Now we can substitute this result into Equation (21),

$$\int_a^b f(x)dx = \frac{1}{2}h\sum_{k=1}^{N}[f(x_{k-1}+f(x_k)] + \frac{1}{12}h^2[f'(a)-f'(b)] + O(h^4). \quad (25)$$

Thus the approximation error (to leading order) for the trapezoidal rule is

$$\epsilon = \frac{1}{12}h^2[f'(a)-f'(b)], \quad (26)$$

which the first term for the *Euler-Maclaurin formula* that quantifies the error on the trapezoidal rule. This tells us that the trapezoidal rule is a **first-order accurate** method, $O(h)$ and has second-order errors, $O(h^2)$.

One can do a similar exercise for Simpson's rule, finding

$$\epsilon = \frac{1}{90}h^4[f'''(a)-f'''(b)]. \quad (27)$$

Thus, Simpson's rule is third-order accurate with $O(h^4)$ errors.

We can use the approximation error ϵ to determine how many slices (N) are needed to approach machine precision $C \simeq 10^{-16}$. To do so, we need to evaluate this against a known function $f(x)$ whose integral can be expressed analytically. We first set the error to the analytical result times C,

$$\epsilon = \frac{1}{90}h^4[f'''(a)-f'''(b)] \simeq C\int_a^b f(x)\,dx \quad (28)$$

Solving for h, we find

$$h \simeq \sqrt{\frac{90\int_a^b f(x)dx}{f'''(a)-f'''(b)}}C^{1/4}, \quad (29)$$

and by using $h=(b-a)/N$, we find that

$$N = (b-a)\sqrt{\frac{f'''(a)-f'''(b)}{90\int_a^b f(x)dx}}C^{-1/4}, \quad (30)$$

slices are needed to approach machine precision. Using $C=10^{-16}$, we find that $N=10^4$ slices are needed with Simpson's rule. Anything beyond this value of N will not result in a more accurate answer. By contrast, the trapezoidal rule requires $N \simeq 10^8$ slices to be this accurate. Furthermore, one can see that the accuracy on the trapezoidal and Simpson's rule depends on the spatial derivative at the boundaries. When they are large (for any reason), then the errors will be large. Later, we will cover methods that avoid this behavior.

Practical estimation of errors

The Euler-Maclaurin formula is only valid for integrals with a closed-form solution. In many scientific calculations, no closed-form solution is known. Fortunately, there is a way around this deficiency.

Let's consider an integral being evaluated from $x = a$ to $x = b$ with N_1 slices, giving a step size $h_1 = (b - a)/N_1$. Using the trapezoidal rule, we can determine a numerical value for this integral. Here's the trick: let's double the number of slices to $N_2 = 2N_1$ with a new step size $h_2 = (b - a)/N_2 = h_1/2$ and re-evaluate the integral.

The trapezoidal rule has an inherent error of $O(h^2)$, thus by halving the step size, we cut the error by one-quarter. Going back to the initial integral, we can express the true value as $I = I_1 + ch_1^2$ (neglecting higher order terms), where I_1 is the numerical integral with N_1 steps. We can also write a similar formula for the integral with N_2 steps as $I = I_2 + ch_2^2$. Equating these two expressions, we find

$$I_2 - I_1 = ch_1^2 - ch_2^2 = 3ch_2^2, \tag{31}$$

where we have used $h_1 = 2h_2$. Rearranging this expression to find the error $\epsilon_2 = ch_2^2$, we find

$$\boxed{\epsilon_2 = \frac{1}{3}(I_2 - I_1)} \tag{32}$$

This gives us a better practical method of calculating the approximation error of any integral, closed-form or not. For Simpson's rule, a similar calculation can be done to find

$$\boxed{\epsilon_2 = \frac{1}{15}(I_2 - I_1)} \tag{33}$$

5.3: Choosing the number of steps

In the last lecture, we found that the approximation error of the trapezoidal rule will decrease as $(I_2 - I_1)/3$ if the resolution is doubled. Usually when running a simulation, there is a limited amount of computer cores and time allocated on the machine. With this "budget" in mind, we would like to complete some calculation at some resolution to some accuracy. Here we will cover one method on how to make these estimations with doubling the resolution or step size.

A simple way is to start with a small number N of steps and see how the error of the solution decreases as N is doubled. On the i-th doubling step, the error is

$$\epsilon_i = (I_i - I_{i-1})/3 \tag{34}$$

This method of doubling the resolution is known as an **adaptive integration** method. The nice part of doubling the resolution is that we've already computed half of the functional

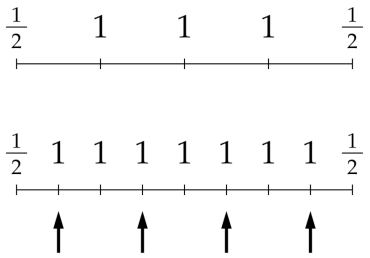

Figure 3: **Doubling the number of steps in the trapezoidal rule.**

values necessary for the integral, shown in the Figure. In the Figure, the top shows each point being multiplied by the appropriate factor. The bottom shows the system when the number of steps is doubled. Notice that there are only $N/2-1$ new points to calculate. These new points are known as *nested* points. Retaining the old solution greatly increases the computational efficiency of the solver, only needing to compute the functional value once at each point.

In mathematical terms, we can consider the trapezoidal rule at the i-th step of the calculation. In this step, there are N_i slices and the width is $h_i = (b-a)/N_i$, and the number of slices are doubled in each step. Therefore, the integral is

$$I_i = h_i \left[\frac{1}{2} f(a) + \frac{1}{2} f(b) + \sum_{k=1}^{N_i - 1} f(a + k h_i) \right] \tag{35}$$

$$= h_i \left[\frac{1}{2} f(a) + \frac{1}{2} f(b) + \sum_{\substack{\text{even } k \\ 2 \ldots N_i - 2}} f(a + k h_i) + \sum_{\substack{\text{odd } k \\ 1 \ldots N_i - 1}} f(a + k h_i) \right] \tag{36}$$

However,

$$\sum_{\substack{\text{even } k \\ 2 \ldots N_i - 2}} f(a + k h_i) = \sum_{k=1}^{N_i/2 - 1} f(a + 2k h_i) = \sum_{k=1}^{N_{i-1} - 1} f(a + k h_{i-1}). \tag{37}$$

Putting these together, we find that

$$I_i = \frac{1}{2} h_{i-1} \left[\frac{1}{2} f(a) + \frac{1}{2} f(b) + \sum_{k=1}^{N_{i-1} - 1} f(a + k h_{i-1}) \right] + h_i \sum_{\substack{\text{odd } k \\ 1 \ldots N_i - 1}} f(a + k h_i) \tag{38}$$

Seeing that the first term is just the trapezoidal rule from the previous [$(i-1)$-th] step, refining the integral as N is double is simply

$$I_i = \frac{1}{2} I_{i-1} + h_i \sum_{\substack{\text{odd } k \\ 1 \ldots N_i - 1}} f(a + k h_i) \tag{39}$$

Because this method builds upon the previous result, there is very little price to pay in performing the integral adaptively. The entire process follows these steps:

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.
2. Double the number of steps and calculate the improved estimate of the integral.
3. If the absolute magnitude of the error is less than the target accuracy for the integral, halt the process. Otherwise repeat from step #2.

Adaptive Simpson's Rule

An adaptive version of Simpson's rule is similar to the adaptive trapezoidal rule but a little more complicated. First, the error on the i-th step is

$$\epsilon_i = \frac{1}{15}(I_i - I_{i-1}) \tag{40}$$

To calculate the integral on this step, let's define two variables that track the discrete sum and how its refined,

$$S_i = \frac{1}{3}\left[f(a) + f(b) + 2\sum_{\substack{\text{even } k \\ 2\ldots N_i-2}} f(a + kh_i)\right] \tag{41}$$

$$T_i = \frac{2}{3}\sum_{\substack{\text{odd } k \\ 1\ldots N_i-1}} f(a + kh_i) \tag{42}$$

Using a method that is similar to the description of the adaptive trapezoidal rule, we can show that

$$S_i = S_{i-1} + T_{i-1} \quad \text{and} \quad I_i = h_i(S_i + 2T_i) \tag{43}$$

5.4: Romberg Integration

We can take adaptive methods a step further and try to increase the approximation errors by increasing the order of accuracy with respect to the step size h. We'll use the adaptive trapezoidal rule as an example. In the i-th step, the leading-order error is

$$ch_i^2 = \frac{1}{3}(I_i - I_{i-1}), \tag{44}$$

using the doubling technique in the previous section. We can include this term in the numerical integral,

$$I = I_i + \frac{1}{3}(I_i - I_{i-1}) + O(h_i^4) \tag{45}$$

making the result third-order accurate, which is as accurate as Simpson's rule.

We can generalize this method to an arbitrary number of iterations, correcting for the errors. The following procedures is called **Romberg integration**[1]. However, this method has limitations, where it works best when the power series converges rapidly. If one needs hundreds of terms in the series to get good convergence, then the method is not going to provide any significant advantages. Furthermore if the function varies rapidly, has singularities, or is noisy, then Romberg integration does not perform well.

With the caveats stated, we can look at the method. First let's introduce some notation $R_{i,j}$ is the j-th term of the polynomial of the i-th step. The first two terms comes from the trapezoidal rule,

$$R_{i,1} = I_i, \quad R_{i,2} = I_i + (I_i - I_{i-1})/3 = R_{i,1} + (R_{i,1} - R_{i-1,1})/3. \tag{46}$$

However from Equation (45) for the i-th and $(i-1)$-th steps, we have

$$I = R_{i,2} + c_2 h_i^4 + O(h^6) \tag{47}$$

$$I = R_{i-1,2} + c_2 h_{i-1}^4 + O(h_{i-1}^6) = R_{i-1,2} + 16 c_2 h_i^4 + O(h_i^6). \tag{48}$$

Equating these two expressions, we find that

$$c_2 h_i^4 = \frac{1}{15}(R_{i,2} - R_{i-1,2}) + O(h_i^6) \tag{49}$$

Substituting this error back into Equation (47), we find

$$I = R_{i,2} + \frac{1}{15}(R_{i,2} - R_{i-1,2}) + O(h_i^6). \tag{50}$$

Notice that we have effectively made this into a fifth-order accurate solution by eliminating the h^4 terms.

As you can imagine, one can continue this iterative process, canceling out higher and higher order errors. For a solution in the m-th iteration of canceling errors, the numerical solution will be

$$I = R_{i,m+1} + O(h_i^{2m+2}), \tag{51}$$

that has a leading-order error of order $(2m+2)$, where

$$R_{i,m+1} = R_{i,m} + \frac{1}{4^m - 1}(R_{i,m} - R_{i-1,m}) \tag{52}$$

that has an error of

$$c_m h_i^{2m} = \frac{1}{4^m - 1}(R_{i,m} - R_{i-1,m}) + O(h_i^{2m+2}) \tag{53}$$

This procedure looks like the following. The arrows show which previous estimates go into the calculation of each new one through Equation (52).

In practice, here are the steps to perform Romberg integration:

[1]Romberg integration is a more general technique of *Richardson extrapolation*, in which high-order estimates of quantities are calculated iteratively from lower-order ones.

$$
\begin{array}{ccccccc}
I_1 \equiv R_{1,1} & & & & & & \\
& \searrow & & & & & \\
I_2 \equiv R_{2,1} & \rightarrow & R_{2,2} & & & & \\
& \searrow & & \searrow & & & \\
I_3 \equiv R_{3,1} & \rightarrow & R_{3,2} & \rightarrow & R_{3,3} & & \\
& \searrow & & \searrow & & \searrow & \\
I_4 \equiv R_{4,1} & \rightarrow & R_{4,2} & \rightarrow & R_{4,3} & \rightarrow & R_{4,4} \\
& \searrow & & \searrow & & \searrow & & \searrow
\end{array}
$$

1. Calculate the first two estimates of the integral with the regular trapezoidal rule: $I_1 \equiv R_{1,1}$ and $I_2 \equiv R_{2,1}$.
2. Calculate a more accurate estimate $R_{2,2}$ from the two starting estimates.
3. Calculate the next trapezoidal rule estimate $I_3 \equiv R_{3,1}$ and from this calculate $R_{3,2}$ and $R_{3,3}$.
4. At each successive stage, we compute one more trapezoidal rule estimate $I_i \equiv R_{i,1}$ and then the remaining terms.
5. From each estimate, we can also calculate the error (Equation 53) that allows us to halt the calculation when the error on our estimate is less than some *tolerance*.

5.5: Higher-order Integration

We've seen in this Chapter that the approximation error (or equivalently the accuracy order) decreases as the fitting function increases its order. For example, the trapezoidal rule uses linear functions, and Simpson's rule uses parabolas. The general form of the numerical integration with polynomial fitting functions is

$$\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w_k f(x_k), \tag{54}$$

where x_k are the positions of the sample points and w_k are the weights. We can make similar methods as the trapezoidal and Simpson's rules, but with different coefficients. These higher-order integration rules are known as *Newton-Cotes* formulas.

There is one major restriction that we've been placing on integration limits: *uniform spacing of the slices*. But what if we relaxed this restriction? If the points were allowed to exist closer together (farther away) when the function is varying rapidly (slowly), this will dramatically increase the accuracy of the numerical integration. This method is known as *Gaussian quadrature*, which we will cover next lecture.

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

5.6: Gaussian Quadrature

Here we will derive the method of Gaussian Quadrature, which is a very accurate integration method of smoothly varying function. It does not work well with functions with sharp changes or ones that are noisy. For these functions, the trapezoidal or Simpson's rule is a better choice. The key difference in this method is the use of N non-uniform sample points x_k.

Non-uniform sample points

The first task is to approximate the function through N sample points with a polynomial, which we will integrate from $x = a$ to $x = b$. To fit N points, we need to use a polynomial of degree $N - 1$ with the method of *interpolating polynomials*. Let's consider the following quantity:

$$\phi_k(x) = \prod_{\substack{m=1\ldots N \\ m\neq k}} \frac{x - x_m}{x_k - x_m} \tag{55}$$

$$= \frac{x - x_1}{x_k - x_1} \times \ldots \times \frac{x - x_{k-1}}{x_k - x_{k-1}} \times \frac{x - x_{k+1}}{x_k - x_{k+1}} \times \ldots \times \frac{x - x_N}{x_k - x_N} \tag{56}$$

You can see that the product includes all N points expect the point x_k. For k values from 1 to N, this defines N *different polynomials*. When this quantity is evaluated at one of the sample points $x = x_m$, we get

$$\phi_k(x_m) = \begin{cases} 1 & \text{if } m = k, \\ 0 & \text{if } m \neq k, \end{cases}$$

which is the *Kronecker delta function* δ_{km}. To create a polynomial through the entire function, we consider a linear combination of these polynomials,

$$\Phi(x) = \sum_{k=1}^{N} f(x_k)\phi_k(x). \tag{57}$$

Double-checking, we can evaluate this function at a sample point $x = x_m$, finding

$$\Phi(x_m) = \sum_{k=1}^{N} f(x_k)\phi_k(x_m) = \sum_{k=1}^{N} f(x_k)\delta_{km} = f(x_m). \tag{58}$$

Now we can integrate the polynomial defined in Equation (57) from a to b,

$$\int_a^b f(x)dx \simeq \int_a^b \Phi(x)dx = \int_a^b \sum_{k=1}^{N} f(x_k)\phi_k(x)dx \tag{59}$$

$$= \sum_{k=1}^{N} f(x_k) \int_a^b \phi_k(x)dx \tag{60}$$

We can compare this to the *Newton-Cotes formula* from the last lecture, and identify the integrals in the previous equation as the needed weights for numerical integration,

$$w_k = \int_a^b \phi_k(x)dx \tag{61}$$

Thus, we have found a general method for creating an integration rule for any set of sample points x_k.

We derived Gaussian quadrature for a integral from a to b, but it can be easily rescaled to arbitrary integration limits. For historical reasons, the sample points and weights are given for integration limits from –1 to 1,

$$w_k = \int_{-1}^{1} \phi_k(x)dx. \tag{62}$$

However, we can easily rescale this to any integration limit, making it more narrow or wide and/or slicing the sample points and weights to different x values. This can be thought as changing variables when integrating.

For instance, suppose that we want to map the Gaussian quadrature weights from [–1, 1] to $[a, b]$. The new sample points x'_k will be altered by

$$x'_k = \frac{1}{2}(b-a)x_k + \frac{1}{2}(b+a), \tag{63}$$

effectively spreading or narrowing the spacing of the sample points, while keeping the ratios of their separation constant. Similarly, the weights will have to be rescaled by a corresponding factor by

$$w'_k = \frac{1}{2}(b-a)w_k \tag{64}$$

After this rescaling and shifting of the Gaussian quadrature points and weights, we can express the integral as

$$\int_a^b f(x)dx \simeq \sum_{k=1}^{N} w'_k f(x'_k) \tag{65}$$

Sample points for Gaussian Quadrature

The previous derivation of a general integration method gives us a procedure to calculate the integration weights but not the location of the sample points x_k. It is possible to choose the N sample points so that the numerical integral is *exact*.

For an exact result, we need a polynomial of degree $2N-1$. The proof is pretty involved and is left to the appendix of the book. The end result is that the sample points x_k should be chosen to correspond to the zeros of the N^{th} Legendre polynomial[2] $P_N(x)$ (see the Figure below), rescaled to the limits of integration.

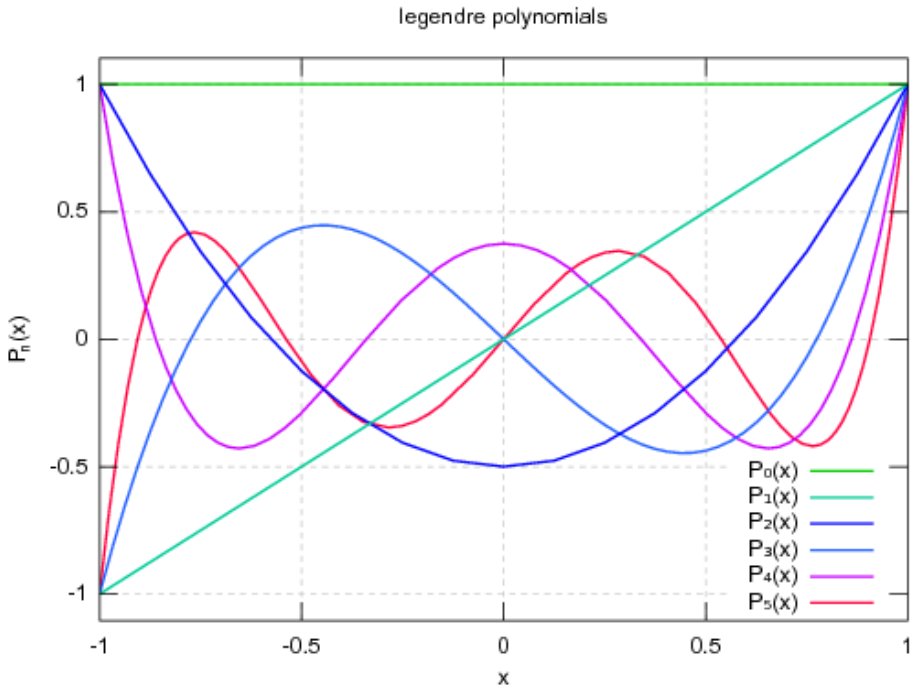

Figure 4: Legendre polynomials up to $N = 5$.

The weights are determined from spatial derivative of P_N at the sample point x_k,

$$w_k = \left[\frac{2}{1-x^2} \left(\frac{dP_N}{dx} \right)^{-1} \right]_{x=x_k} \tag{66}$$

These weights are somewhat computationally intensive, but they only have to calculated once for a given value of N. This method is very accurate for smooth functions for a small N, which made it very convenient before the advent of computers. You can find an example code `gaussxw.py` on Canvas (Files / course-docs) that calculates these weights. The Figure below shows the sample points and their corresponding weights for $N = 10$ and $N = 100$.

[2]These are solutions to Legendre's differential equation and appear when solving Laplace's equation. They are a set of orthonormal polynomials, which can be used in linear combinations.

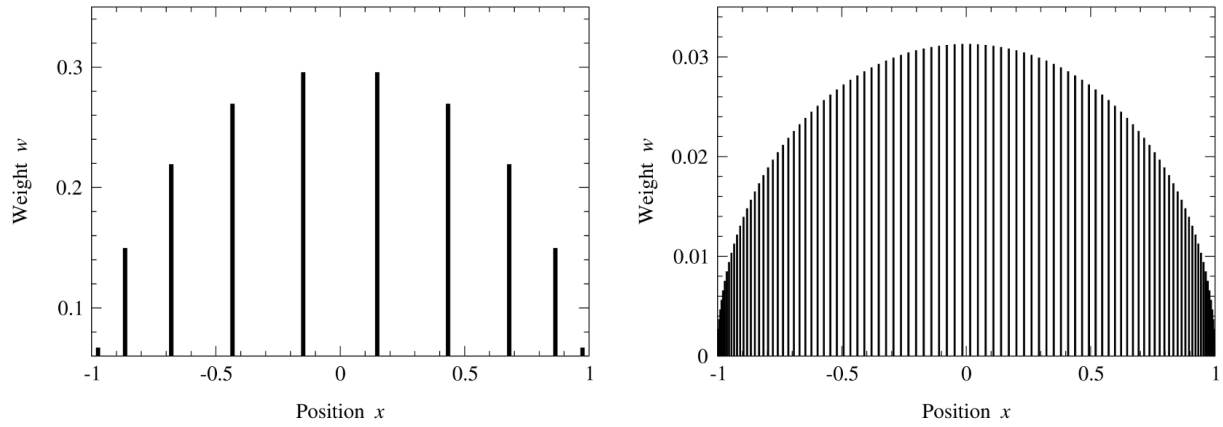

Figure 5: Sample points and weights for Gaussian quadrature.

Example 5.2: Gaussian integral of a simple function

Consider again the integral we previously evaluated, $\int_0^2 (x^4 - 2x + 1)dx$, which has a value of 4.4. Using Gaussian quadrature with $N = 3$, calculate the integral.

```
from gaussxw import gaussxw

def f(x):
    return x**4 - 2*x + 1

N = 3
a = 0.0
b = 2.0

# Calculate the rescaled and shifted sample points and weights
x, w = gaussxw(N)
xp = 0.5*(b-a)*x + 0.5*(b+a)
wp = 0.5*(b-a)*w

# Perform the integration
s = 0.0
for k in range(N):
    s += wp[k] * f(xp[k])

print(s)
```

Note how the function `gaussxw(N)` returns two variables and how they can be directed into two variables instead of a list. This function returns the Gaussian quadrature sample

points and weights in these variables, which we can then rescale to the integration window [0,2]. With $N = 3$, you should get the result of `4.4`, which is exactly the value. This is impressive, considering that we are only evaluating the function at three points. But in hindsight, this is expected because we are integrating over a polynomial, on which this method is based.

5.7: Choosing an Integration Method

We have covered four different integration methods: trapezoidal rule, Simpson's rule, Romberg integration, and Gaussian quadrature. Which one is best? There is no right answer to such a general question, and the choice depends on the integral being evaluated. One general rule-of-thumb is that Romberg or Gaussian integration gives very accurate results for smoothly varying functions with very few sample points. If the function is noisy or not smooth, then the trapezoidal rule or Simpson's rule is the best choice. Below you will find some guidelines on which method is best, given the situation.

The trapezoidal rule is trivial to implement and is a good choice if a quick answer is needed without much accuracy. It uses uniformly spaced sample points, which is appropriate for lab data sampled at constant time intervals and/or poorly behaved (widely varying, discontinuities, noisy) functions. In its adaptive form (Section 5.3), it can give us a guaranteed accuracy for an integral, although it may take more slices and thus more computing time to achieve some level of accuracy.

Simpson's rule has the same benefits as the trapezoidal rule but with better accuracy with the same number of points, or the same accuracy with fewer points. However since we're approximating the function with parabolas, this can lead to large errors when the data are noisy or ill-behaved.

Romberg integration is the quintessential higher-order integration method that gives excellent accuracy with a limited amount of sample points. It does not work well with integrands that widely vary and/or are noisy. It is best applied to smooth functions.

Gaussian quadrature has many of the same advantages and disadvantages of Romberg integration. However it is very easy to program, perhaps the simplest method. It uses non-uniform spacing of the sample points, unlike the previous three methods. If one is trying to integrate equally spaced data points, for instance from an experiment, Gaussian quadrature might not be the best choice. The hard work is determining the sample points and their weights. This calculations are usually performed before the integration or stored in a lookup table. It is the highest order integration method and perhaps the most accurate.

With these guidelines, you should be able to choose the best method for a given problem.

5.8: Integrals over Infinite Ranges

Very often in physics, we encounter infinite integrals, like $\int_0^\infty f(x)dx$. For example, one can calculate the fraction of electrons with some temperature T that have kinetic energies greater than the ionization potential of hydrogen that can ionize hydrogen through a collision. In practice, we cannot integrate to infinity, but we can re-express the integration by a change of variables. For an integral evaluated from 0 to ∞, we can use the standard change of variables,

$$z = \frac{x}{1+x} \qquad x = \frac{z}{1-z}. \tag{67}$$

Then $dx = dz/(1-z)^2$, and the integral can be re-expressed with a finite integration window,

$$\int_0^\infty f(x)dx = \int_0^1 \frac{1}{(1-z)^2} f\left(\frac{z}{1-z}\right) dz. \tag{68}$$

In general, there are other changes of variables,

$$z = \frac{x}{c+x} \qquad z = \frac{x^\gamma}{1+x^\gamma} \tag{69}$$

for any c or γ, that could be applied to functions, but one needs to experiment with the transformation to see how the integrand and limits change.

If we want to evaluate an integral from a to ∞, we can make the following change,

$$z = \frac{x-a}{1+x-a}, \quad x = \frac{z}{1-z} + a, \quad dx = \frac{dz}{(1-z)^2}, \tag{70}$$

resulting in

$$\int_a^\infty f(x)dx = \int_0^1 \frac{1}{(1-z)^2} f\left(\frac{z}{1-z} + a\right) dz \tag{71}$$

For integrals over the whole real domain ($-\infty \to \infty$), we could split the integral into two parts and perform the first method in the section. There is another substitution that we can make to avoid this complication, resulting in a more elegant solution,

$$x = \frac{z}{1-z^2}, \qquad dx = \frac{1+z^2}{(1-z^2)^2}\, dz. \tag{72}$$

Substituting these variables into the integral, we find that

$$\int_{-\infty}^\infty f(x)dx = \int_{-1}^1 \frac{1+z^2}{(1-z^2)^2} f\left(\frac{z}{1-z^2}\right) dz. \tag{73}$$

There is an even better substitution, though!

$$x = \tan z, \qquad dx = \frac{dz}{\cos^2 z} \tag{74}$$

This gives the following integral,

$$\int_{-\infty}^\infty f(x)dx = \int_{-\pi/2}^{\pi/2} \sec^2 z\, f(\tan z)\, dz \tag{75}$$

Example 5.3: Integrating over an infinite range

We can modify the previous example (5.2) that used Gaussian quadrature to calculate the following infinite integral,

$$I = \int_0^\infty e^{-t^2}\,dt. \tag{76}$$

We can re-express this as a finite integral,

$$I = \int_0^1 \frac{e^{-z^2/(1-z)^2}}{(1-z)^2}\,dz \tag{77}$$

Using $N = 50$, write a program that solves this integral.

```
from gaussxw import gaussxwab
from math import exp

def f(z):
    return exp(-z**2/(1-z)**2)/(1-z)**2

N = 50
a = 0.0
b = 1.0
x, w = gaussxwab(N, a, b)
s = 0.0
for k in range(N):
    s += w[k] * f(x[k])
print(s)
```

This integral has an exact result of $\sqrt{\pi}/2 = 0.886226925453...$. You will see that Gaussian quadrature recovers the answer to machine precision with only 50 slices.

5.9: Multiple Integrals

Physics problems are usually multi-dimensional, barring some symmetry or simplification, and we can generalize the integration methods in this chapter to 2D, 3D, and beyond. As an example, let's look at a two-dimensional integral,

$$I = \int_0^1 \int_0^1 f(x,y)dxdy, \tag{78}$$

which can be rewritten as

$$I = \int_0^1 F(y)\,dy, \quad \text{where} \quad F(y) = \int_0^1 f(x,y)\,dx \tag{79}$$

This can split into two integrals, first evaluating $F(y)$ with a suitable set of y values, then we can use these values in the integral over x, exactly (conceptually) like how an integral is performed analytically. Using Gaussian quadrature with an equal number N of points in each dimension, we have

$$F(y) \simeq \sum_{i=1}^{N} w_i f(x_i, y) \quad \text{and} \quad I \simeq \sum_{j=1}^{N} w_j F(y_j). \tag{80}$$

Alternatively, we can use the *Gauss-Legendre product formula*,

$$I \simeq \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j f(x_i, y_i), \tag{81}$$

which can be thought as a two-dimensional version of Gaussian quadrature. The figure below shows the non-uniform spacing of the sample points in 2D.

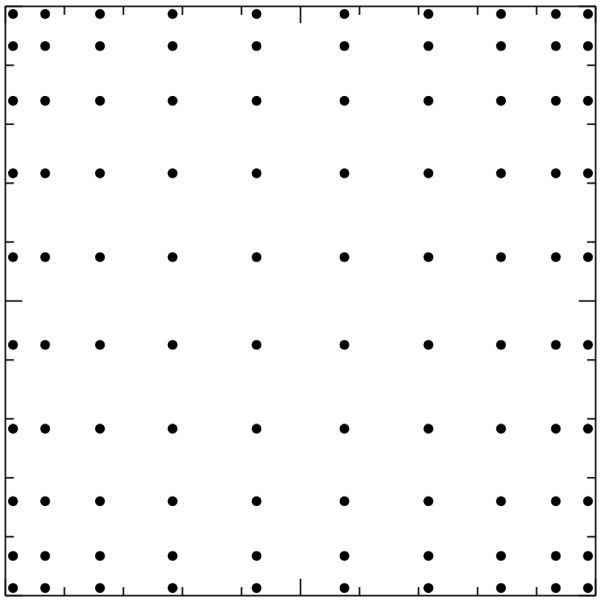

Figure 6: Sample points for Gaussian quadrature in two dimensions

There are other methods of how to choose the sample points, which I will leave the details to the book. It becomes complicated when there exists complex integration domains. We will be covering a random method, named *Monte Carlo*, that uses a random sample of points to evaluate the integral. We will be covering this method much later in the semester.

5.10: Derivatives

The basic numerical techniques of differentiation are easier than numerical integration. Numerical derivatives are used less often than integrals because they can be calculated analytically for any function, which is not the case for integrals. Furthermore, there are some

practical problems with numerical derivatives with round-off errors, so they are not used as often. However, they are very important when solving partial differential equations, which we will cover in Chapter 9.

Forward and Backward Differences

The standard mathematical definition of a derivative is

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}. \tag{82}$$

The basic method for numerical derivatives implements this exact formula, however we make h very small instead of letting it approach zero,

$$\frac{df}{dx} \simeq \frac{f(x+h) - f(x)}{h}. \tag{83}$$

This is called a **forward difference** because we're calculating the derivative with a data point in the positive direction of the evaluation point. The **backward difference** calculates the slope in the opposite direction,

$$\frac{df}{dx} \simeq \frac{f(x) - f(x-h)}{h}. \tag{84}$$

Both of these methods are illustrated in Figure 7. The forward and backward differences give about the same answer. Most methods use the forward difference, but there are a few special cases, particularly with discontinuities or boundaries in the forward direction, where the backward difference is preferred.

Errors

We calculated error estimates for integrals, and we can apply the same method for derivatives in computing the Taylor expansion of $f(x)$ about x:

$$f(x+h) = f(x) + hf'(x) + \tfrac{1}{2}h^2 f''(x) + \dots \tag{85}$$

where the primes indicate a spatial derivative with respect to x. Solving for $f'(x)$, we find

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \tfrac{1}{2}hf''(x) + \dots \tag{86}$$

Thus to leading order, the approximation error is $\frac{1}{2}h|f''(x)|$, linear in h. In other words, decreasing the step size h, the errors will decrease by the same order.

Numerical derivatives have the inherent limitation of taking differences of two similar numbers, where rounding errors come into play. It is worthwhile to determine what is the

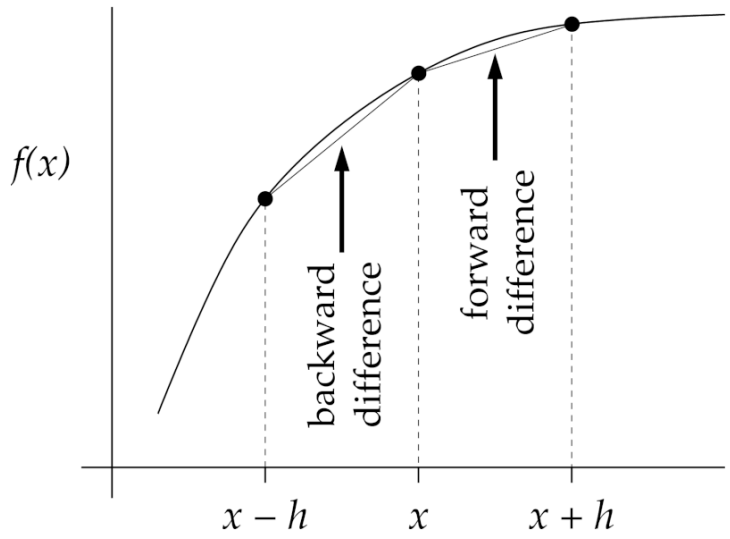

Figure 7: Forward and backward differences

smallest step size h that can be chosen before rounding errors dominate. If $C = 10^{-16}$ is the rounding error, then the accuracy for the difference $f(x+h) - f(x)$ will be (in the worst case) about $2C|f(x)|$. Putting this error into the difference formula, we have an overall error of $2C|f(x)|/h$. Now we can compare the rounding and approximation error, $\frac{1}{2}h|f''(x)|$. In the worst case scenario, the total error is the sum of them,

$$\epsilon = \frac{2C|f(x)|}{h} + \frac{1}{2}h|f''(x)|. \tag{87}$$

We want to minimize the error, so we differentiate with respect to h and find the where it equals zero, giving

$$h = \sqrt{4C\left|\frac{f(x)}{f''(x)}\right|}. \tag{88}$$

We can substitute this step size back into Equation (87) and find the minimized error,

$$\epsilon = h|f''(x)| = \sqrt{4C|f(x)f''(x)|} \tag{89}$$

If $f(x)$ and $f''(x)$ are about one, then the minimal error occurs when $h \approx \sqrt{C} = 10^{-8}$.

Central differences and higher-order approximations

Central differencing, which uses the points on the *negative and positive* sides,

$$\frac{df}{dx} \simeq \frac{f(x+h/2) - f(x-h/2)}{h} \tag{90}$$

is much more accurate than forward and backward differences. Its approximation error, $\frac{1}{24}h^2|f'''(x)|$, can be found by taking the sum of the Taylor expansions of $f(x+h/2)$ and

$f(x - h/2)$. Using the same minimization method, the total error is minimal when $h \simeq C^{1/3} \simeq 10^{-5}$, giving an error of $\sim 10^{-10}$, a factor of 100 times better than the forward and backward differences.

However there are practical difficulties with having the sample points within the slices. Usually this is mitigated by taking the difference between the two adjacent points,

$$\frac{df}{dx} \simeq \frac{f(x+h) + f(x-h)}{2h}. \tag{91}$$

This form of a central difference is derived by fitting a parabola through the left, central, and right points and then taking its derivative.

Higher order polynomials can be fit through the sample points, further improving the accuracy of the numerical derivatives. All of the odd-order polynomials have sample points at "halfway" points (e.g. $\pm\frac{1}{2}h$), and the even-order fits have sample points at "integer" points, making it more convenient. The table below shows the coefficients associated with the sample points when calculating the numerical derivative.

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	$-\frac{3}{640}$		$\frac{25}{384}$		$-\frac{75}{64}$		$\frac{75}{64}$		$-\frac{25}{384}$		$\frac{3}{640}$	$O(h^6)$

Table 5.1: Coefficients for numerical derivatives. The coefficients for central approximations to the first derivative of $f(x)$ at $x = 0$. To derive the full expression for an approximation, multiply the samples listed in the top row of the table by the coefficients in one of the other rows, then divide by h. For instance, the cubic approximation would be $[\frac{1}{24}f(-\frac{3}{2}h) - \frac{27}{24}f(-\frac{1}{2}h) + \frac{27}{24}f(\frac{1}{2}h) - \frac{1}{24}f(\frac{3}{2}h)]/h$. For derivatives at points other than $x = 0$ the same coefficients apply—one just uses the appropriate sample points around the value x of interest. The final column of the table gives the order of the approximation error on the derivative.

Second derivatives

We can determine the approximations for a second derivative $f''(x)$ by taking the difference of the numerical approximation of $f'(x)$. Let's start with the central difference formula at the halfway points,

$$f'(x + h/2) \simeq \frac{f(x+h) - f(x)}{h}, \qquad f'(x - h/2) \simeq \frac{f(x) - f(x-h)}{h}, \tag{92}$$

To obtain the second derivative, we apply the central difference once again,

$$f''(x) \simeq \frac{f'(x+h/2) - f'(x-h/2)}{h} \tag{93}$$

$$= \frac{[f(x+h) - f(x)]/h - [f(x) - f(x-h)]/h}{h} \tag{94}$$

$$= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{95}$$

This is the simplest expression for the second derivative, which we will be heavily using in Chapter 9 with partial differential equations. There exist higher-order approximations, but we will not use them in this class.

Similar to the central difference of the first derivative, we can calculate the approximation error to be $\frac{1}{12}h^2|f''''(x)|$ by taking the Taylor expansions of $f(x+h)$ and $f(x-h)$ and subtracting them. The error is roughly $\sqrt{C} \simeq 10^{-8}$, which is the same accuracy as the forward and backward differences.

Partial derivatives

There are many physics equations that use partial derivatives, which are taken with respect to *only one variable* in the function that has multiple variables. The same methods of ordinary differentiation hold for partial derivatives. For instance, the central differences of $f(x, y)$ with respect to x and y are

$$\frac{\partial f}{\partial x} = \frac{f(x+h/2, y) - f(x-h/2, y)}{h} \tag{96}$$

$$\frac{\partial f}{\partial y} = \frac{f(x, y+h/2) - f(x, y-h/2)}{h} \tag{97}$$

The same generalizations can be done for the second derivatives.

Derivatives of noisy data

As we've discussed before, noisy data is prevalent in physics data. An example dataset and its derivative are shown in Figure 8. One can imagine taking the derivative of noisy data makes the data even noisier because of the sharp small-scale changes. Fortunately, there are ways to smooth the derivatives of noisy data.

1. Increase the value of h so that the total error (Equation 87 with C as the data noise) is minimized.

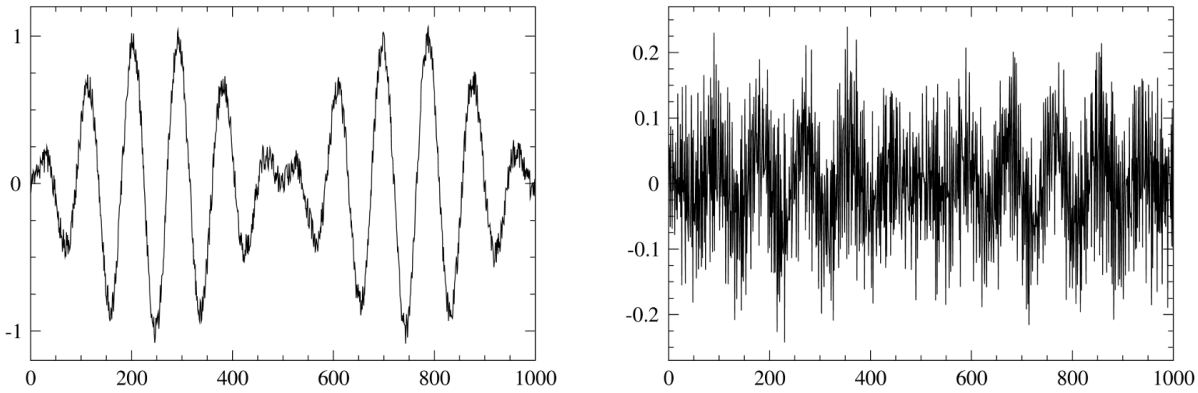

Figure 8: A noisy dataset (top) and its derivative (bottom).

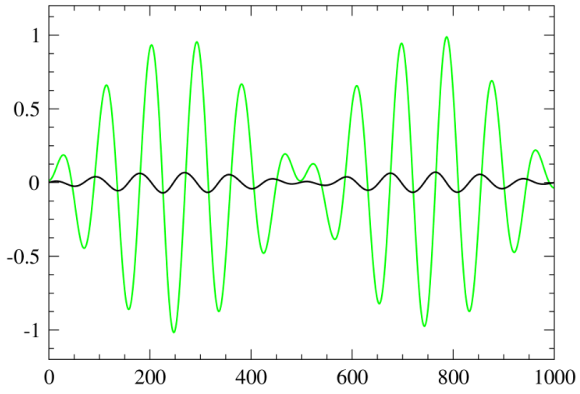

Figure 9: Smoothed data from Figure 8 and an improved estimate of the derivative.

2. Using the least-squares method, fit an analytical solution to the data (which has some fitting errors that then propagate to the derivative) and then take the derivative of the function.
3. Smooth the data without fitting a function and then take the numerical derivative. The smoothing can be done with a running average or with Fourier transforms (Chapter 7). Figure 9 shows the noisy data in Figure 8 now smoothed (green) with its derivative (black).

5.11: Interpolation

Although interpolation and extrapolation are not related to integrals and derivatives, it is worthwhile to discuss here before moving onto more difficult material because it's widely used in computational physics. Suppose you are given the value of a function $f(x)$ at just two points $x = a$ and b, and we want to know the value of $f(x)$ between these two points. We will be looking at *linear interpolation*, where we approximate the function with a linear function. Figure 10 shows the basic geometry of interpolation, and the fundamental formula

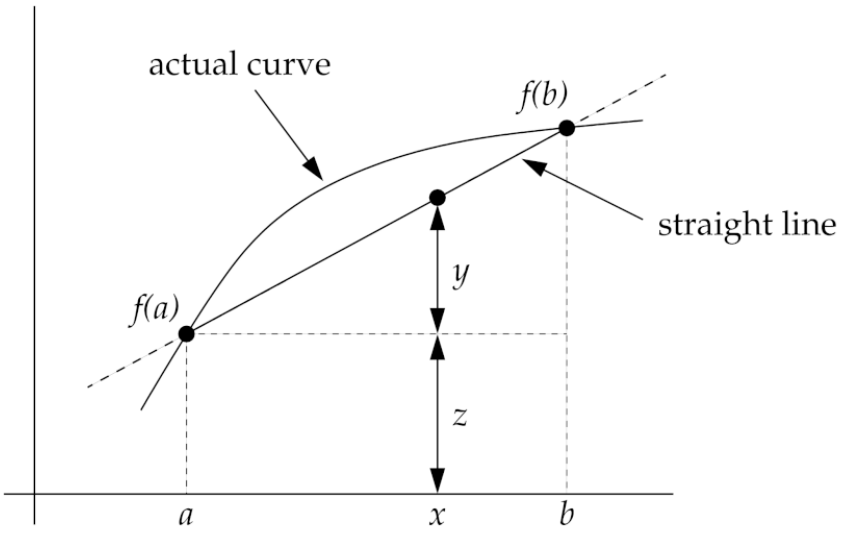

Figure 10: Linear interpolation. The value of $f(x)$ in between the two known points at $x = a$ and $x = b$ is estimated by assuming a straight line from $f(a)$ to $f(b)$.

for linear interpolation is

$$f(x) \simeq y + z = \frac{f(b) - f(a)}{b - a}(x - a) + f(a) \tag{98}$$

$$\boxed{f(x) \simeq \frac{(b - x)f(a) + (x - a)f(b)}{b - a}} \tag{99}$$

Its approximation error is determined by taking two Taylor expansions around a and b:

$$f(a) = f(x) + (a - x)f'(x) + \tfrac{1}{2}(a - x)^2 f''(x) + \ldots \tag{100}$$

$$f(b) = f(x) + (b - x)f'(x) + \tfrac{1}{2}(b - x)^2 f''(x) + \ldots \tag{101}$$

$$\tag{102}$$

and substituting them into Equation (99) to find the approximation error to be $(a - x)(b - x)f''(x)$. One can see the error is the largest at the midpoint and has a magnitude of $\frac{1}{4}h^2|f''(x)|$.

There exist higher order approximations for interpolation and extrapolation that fit parabolic, cubic, and quartic (and so on) functions to the points, and these are known as *Lagrange interpolation* methods. Another highly used interpolation method is known as *cubic splines*. However again, we will only use linear interpolation in this course.