Chapter 4 Numerical Integration Problem Statement

4.1 Numerical integration problem statement

Given a function $f(x)$, approximate the integral of $f(x)$ over the total **interval**, [a,b]. Fig. 4.1 illustrates this area. To accomplish this goal, we assume that the interval has been discretized into a numeral grid, *x*, consisting of $n+1$ points with spacing, $h = \frac{b-a}{n}$. Here, we denote each point in *x* by x_i, where $x_0 = a$ and $x_n = b$. Note that there are n + 1 grid points because the count starts at x0. Assume that the function, $f(x)$, can be computed for any of the grid points, or that we have been given the function implicitly as $f(x_i)$. The interval $[x_i,\ x_{i+1}]$ is referred to as a **subinterval**.

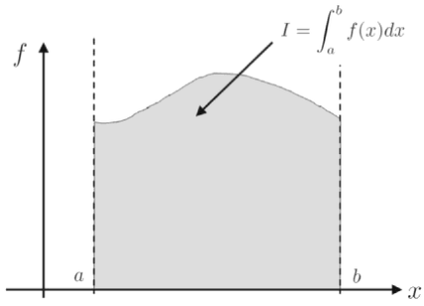


FIGURE 4.1

Illustration of the integral. The integral from *a* to *b* of the function *f* is the area below the curve (shaded in grey).

The following sections give some of the most common entry level methods of approximating. $\int_a^b f(x)dx$. Each method approximates the area under *f(x)* for each subinterval by a shape for which it is easy to compute the exact area, and then sums the area contributions of every subinterval.

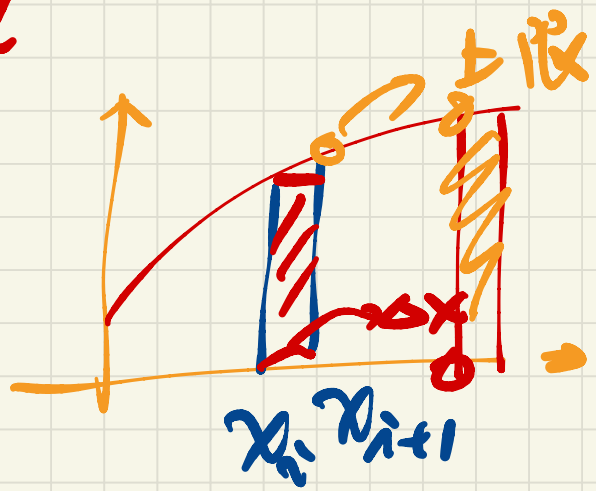
4.2 Riemann integral

The simplest method for approximating integrals is by summing the area of rectangles that are defined for each subinterval. The width of the rectangle is $x_{i+1} - x_i = h$ and the height is defined by a function value $f(x)$ for some *x* in the subinterval. An obvious choice for the height is the function value at the left endpoint, x_i, or the right endpoint, x_{i+1}, because these values can be used even if the function itself is not known. This method gives the **Riemann integral** approximation, which is

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} hf(x_i)$$

or

① Integral



$$A_i = f(x_i) \cdot \Delta x$$

$$A = \sum_{i=1}^{n} f(x_i)\Delta x$$

$$A = \lim_{\Delta x \to 0} \sum_{i=1}^{n} f(x_i)\Delta x \rightarrow \text{下限}$$

$$A=\lim_{\Delta x\to 0}\sum_{i=1}^{n} f(x_{i+1})\Delta x \to \text{上限}$$

梯度积分(梯形)

$$A=\lim_{\Delta x\to 0}\sum_{i=1}^{n}\frac{f(x_{i+1})+f(x_i)}{2}\Delta x$$

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} hf(x_i)$$

depending on whether the left or right endpoint is chosen.

As with numerical differentiation, we want to characterize how the accuracy improves as *h* decreases. To determine this characterizing, we first rewrite the integral of *f (x)* over an arbitrary subinterval in terms of the Taylor series. The Taylor series of *f (x)* around $a = x_i$ is

$$f(x) = f(x_i) + f'(x_i)(x - xi) + \cdots.$$

Thus

$$\int_{x_i}^{x_{i+1}} f(x_i)\,dx = \int_{x_i}^{x_{i+1}} (f(x_i) + f'(x_i)(x - x_i) + \cdots)\,dx$$

by substitution of the Taylor series for the function. Since the integral distributes, we can rearrange the right-hand side into the following form:

$$\int_{x_i}^{x_{i+1}} f(x_i)\,dx + \int_{x_i}^{x_{i+1}} f'(x_i)(x - x_i)\,dx + \cdots$$

Computing each integral separately results in the approximation

$$\int_{x_i}^{x_{i+1}} f(x)dx = hf(x_i) + \frac{h^2}{2}f'(x_i) + O(h^3)$$

which is

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = hf(x_i) + O(h^2)$$

Since the $hf(x_i)$ term is our Riemann integral approximation for a single subinterval, the Riemann integral approximation over a single interval is $O(h^2)$.

If we sum the $O(h^2)$ error over the entire Riemann sum, we obtain $nO(h^2)$.The relationship between *n* and *h* is

$$h = \frac{b-a}{n}$$

and so our total error becomes $\frac{b-a}{n}O(h^2) = O(h)$ over the whole interval. Thus the overall accuracy is $O(h)$.

The **Midpoint Rule** takes the rectangle height of the rectangle at each subinterval to be the function value at the midpoint between x_i and x_{i+1}, which for compactness we denote by $y_i = \frac{x_{i+1}+x_i}{2}$.

The midpoint rule says

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} hf(y_i)$$

Similarly, to the Riemann integral, we take the Taylor series of $f(x)$ around y_i, which is

$$f(x) = f(y_i) + f'(y_i)(x - y_i) + \frac{f''(y_i)(x - y_i)^2}{2!} + \cdots$$

Then the integral over a subinterval is

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = \int_{x_i}^{x_{i+1}} \left(f(y_i) + f'(y_i)(x - y_i) + \frac{f''(y_i)(x - y_i)^2}{2!} + \cdots \right) dx$$

which distributes to

$$\int_{x_i}^{x_{i+1}} f(x)dx = \int_{x_i}^{x_{i+1}} f(y_i)dx + \int_{x_i}^{x_{i+1}} f'(y_i)(x - y_i)dx + \int_{x_i}^{x_{i+1}} \frac{f''(y_i)(x - y_i)^2}{2!} dx + \cdots$$

Recognizing that x_i and x_{i+1} are symmetric around y_i, we get $\int_{x_i}^{x_{i+1}} f'(y_i)(x - y_i)dx = 0$. This is true for the integral of $(x - y_i)^p$ for any odd p. For the integral of $(x - y_i)^p$ and with p even, $\int_{x_i}^{x_{i+1}} (x - y_i)^p dx = \int_{-\frac{h}{2}}^{\frac{h}{2}} x^p dx$, which will result in some multiple of h^{p+1} with no lower order powers of h.

Utilizing these facts reduces the expression for the integral of $f(x)$ to

$$\int_{x_i}^{x_{i+1}} f(x)dx = hf(y_i) + O(h^3).$$

Since $hf(y_i)$ is the approximation of the integral over the subinterval, the midpoint rule is $O(h^3)$ for one subinterval; using similar arguments as those used for the Riemann integral, we get $O(h^2)$ over the whole interval. Since the midpoint rule requires the same number of calculations as the Riemann integral, we essentially get an extra order of accuracy for free; however, if $f(x_i)$ is given in the form of data points, then we will not be able to compute $f(y_i)$ for this integration scheme.

Problem4.1 Use the left and right Riemann integral, as well as midpoint rule, to approximate $\int_0^{\pi} \sin(x)dx$ with 11 evenly spaced grid points over the whole interval. Compare this value to the exact value of 2.

Ans:

```
import numpy as np

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_riemannL = h * sum(f[:n-1])
err_riemannL = 2 - I_riemannL

I_riemannR = h * sum(f[1::])
err_riemannR = 2 - I_riemannR

I_mid = h * sum(np.sin((x[:n-1] + x[1:])/2))
err_mid = 2 - I_mid

print(I_riemannL)
print(err_riemannL)

print(I_riemannR)
print(err_riemannR)

print(I_mid)
print(err_mid)
```

4.3 Trapezoid rule

The **Trapezoid Rule** fits a trapezoid into each subinterval and sums the areas of the trapezoids to approximate the total integral. This approximation for the integral to an arbitrary function is show in Fig. 4.2. For each subinterval, the trapezoid rule computes the area of a trapezoid with corners at $(x_i, 0)$, $(x_{i+1}, 0)$, $(x_i,\ f(x_i))$, and $(x_{i+1},\ f(x_{i+1}))$, which is $h\frac{f(x_i)+f(x_{i+1})}{2}$. Thus, the trapezoid rule approximates integrals according to the expression

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} h\frac{f(x_i)+f(x_{i+1})}{2}$$

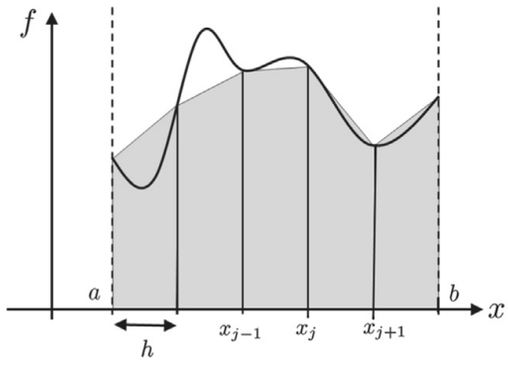


FIGURE 4.2

Illustration of the trapezoid integral procedure. The area below the curve is approximated by a sum of areas of trapezoids that approximate the function.

Problem 4.2 You may notice that the trapezoid rule "double-counts" most of the terms in the series. To illustrate this fact, consider the expansion of the trapezoid rule:

$$\sum_{t=0}^{n-1} h\frac{f(x_i)+f(x_{i+1})}{2} = \frac{h}{2}\left[f(x_0) + f(x_1) + \left(f(x_1) + f(x_2)\right) + \left(f(x_2) + f(x_3)\right) + \cdots + \left(f(x_{n-1}) + f(x_n)\right)\right].$$

Computationally, this is many extra additions ant=0d calls to *f (x)* than are really necessary. We can be made more computationally efficient using the following expression:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right)$$

Ans:

收斂性依 $O(h^2)$

Problem 4.3 Use the trapezoid rule to approximate $\int_0^{\pi} \sin(x)dx$ with 11 evenly spaced grid points over the whole interval. Compare this value to the exact value of 2.

Ans:

```
import numpy as np

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trap = (h/2)*(f[0] + \
        2 * sum(f[1:n-1]) + f[n-1])
err_trap = 2 - I_trap

print(I_trap)
print(err_trap)
```

4.4 Simpson's rule

Consider *two* consecutive subintervals,$[x_{i-1}, x_i]$and $[x_i, x_{i+1}]$.**Simpson's Rule** approximates the area under $f(x)$ over these two subintervals by fitting a quadratic polynomial through the points $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, and $(x_{i+1}, f(x_{i+1}))$, which is a unique polynomial, and then integrates the quadratic exactly. Fig. 4.3 shows this integral approximation for an arbitrary function.

First, we construct the quadratic polynomial approximation of the function over the two subintervals. The easiest way to do this is to use Lagrange polynomials, which was discussed in Chapter 17. By applying the formula for constructing Lagrange polynomials, we obtain

$$P_i(x) = f(x_{i-1})\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + f(x_i)\frac{(x-x_i)(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} + f(x_{i+1})\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)},$$

and with substitutions for h results in

$$P_i(x) = \frac{f(x_{i-1})}{2h^2}(x-x_i)(x-x_{i+1}) - \frac{f(x_i)}{h^2}(x-x_{i-1})(x-x_{i+1}) + \frac{f(x_{i+1})}{2h^2}(x-x_{i-1})(x-x_{i+1}).$$

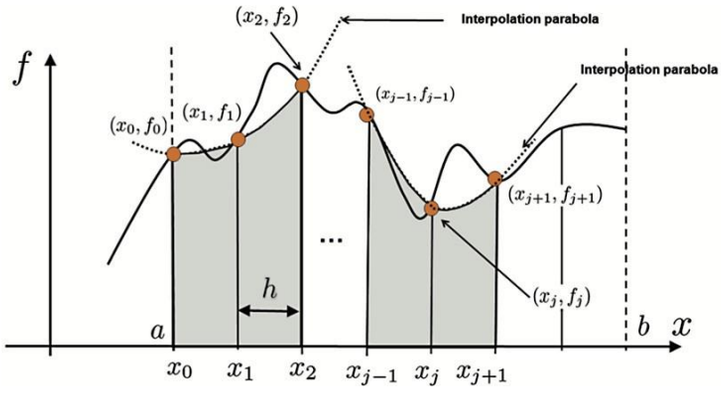


FIGURE 4.3

Illustration of the Simpson integral formula. Discretization points are grouped by three, and a parabola is fit between the three points. This can be done by a typical interpolation polynomial. The area under the curve is approximated by the area under the parabola.

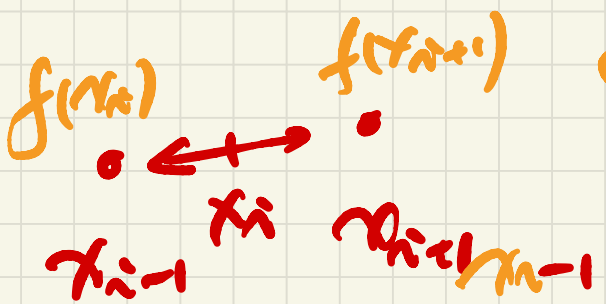
You can confirm that the polynomial curve intersects the desired points. With some algebra and manipulation, the integral of $P_i(x)$ over the two subintervals is

$$\int_{x_{i-1}}^{x_{i+1}} P_i(x)dx = \frac{h}{3}\left(f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right)$$

To approximate the integral over (*a, b*), we must sum the integrals of $P_i(x)$ over all *pairs* of subintervals since $P_i(x)$ spans two subintervals. Substituting $\frac{h}{3}\left(f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right)$ for the integral of $P_i(x)$ and regrouping the terms for efficiency leads to the formula

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 4\sum_{i=1,i\ odd}^{n-1} f(x_i) + 2\sum_{i=1,i\ even}^{n-2} f(x_i) + f(x_n)\right]$$

This regrouping is illustrated in Fig. 4.4:



① 两端点. x_{i-1}, x_i x_{i+1}

$$L(x) = \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i+1}-x_i)}$$

$$f(x) = f(x_{i-1}) \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}$$

$$+ f(x_i) \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}$$

$x = x_{i+1}$ $x = x_{i-1}$ shape function

$$+ f(x_{i+1}) \frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}$$

$$\int f(x) dx.$$

WARNING! Note that to use Simpson's rule, you **must** have an even number of intervals and, therefore, an odd number of grid points.

To compute the accuracy of the Simpson's rule, we take the Taylor series approximation of $f(x)$ as around x_i, which is

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{f''(x_i)(x - x_i)^2}{2!} + \frac{f'''(x_i)(x - x_i)^3}{3!} + \frac{f''''(x_i)(x - x_i)^4}{4!} + \cdots$$

Computing the Taylor series with x_{i-1} and x_{i+1}, and substituting for h where appropriate, gives the expressions

$$f(x_{i-1}) = f(x_i) - hf'(x_i) + \frac{h^2 f''(x_i)}{2!} - \frac{h^3 f'''(x_i)}{3!} + \frac{h^4 f''''(x_i)}{4!} - \cdots$$

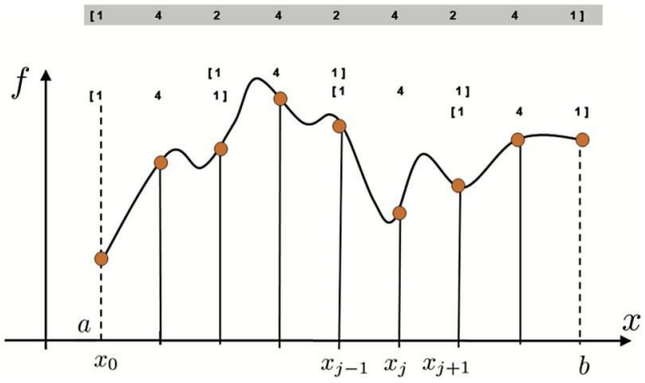


FIGURE 4.4
Illustration of the accounting procedure to approximate the function f by the Simpson rule for the entire interval $[a, b]$.

and

$$(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2 f''(x_i)}{2!} + \frac{h^3 f'''(x_i)}{3!} + \frac{h^4 f''''(x_i)}{4!} + \cdots$$

Consider the expression $\frac{f(x_{i-1})+4f(x_i)+f(x_{i+1})}{6}$. Substituting the Taylor series for the respective numerator values produces the equation

$$\frac{f(x_{i-1}) + 4f(x_i) + f(x_{i+1})}{6} = f(x_i) + \frac{h^2}{6} f''(x_i) + \frac{h^4}{72} f''''(x_i) + \cdots$$

Note that the odd terms cancel out, implying

$$f(x_i) = \frac{f(x_{i-1}) + 4f(x_i) + f(x_{i+1})}{6} - \frac{h^2}{6} f''(x_i) + O(h^4)$$

By substitution of the Taylor series for $f(x)$, the integral of $f(x)$ over two subintervals is then

$$\int_{x_{i-1}}^{x_{i+1}} f(x_i)dx = \int_{x_{i-1}}^{x_{i+1}} \left(f(x_i) + f'(x_i)(x - x_i) + \frac{f''(x_i)(x - x_i)^2}{2!} + \frac{f'''(x_i)(x - x_i)^3}{3!} + \frac{f''''(x_i)(x - x_i)^4}{4!} + \cdots \right) dx$$

Again, we distribute the integral and, without showing it, drop the integrals of terms with odd powers because they are zero to obtain

$$\int_{x_{i-1}}^{x_{i+1}} f(x_i)dx = \int_{x_{i-1}}^{x_{i+1}} f(x_i)dx + \int_{x_{i-1}}^{x_{i+1}} \frac{f''(x_i)(x - x_i)^2}{2!} dx + \int_{x_{i-1}}^{x_{i+1}} \frac{f''''(x_i)(x - x_i)^4}{4!} dx + \cdots$$

at which point we perform the integrations. As will soon be clear, computing the integral of the second term exactly has benefits. The resulting equation is

$$\int_{x_{i-1}}^{x_{i+1}} f(x_i)dx = 2hf(x_i) + \frac{h^3}{3} f''(x_i) + O(h^5)$$

Substituting the expression for $f(x_i)$ derived earlier, the right-hand side becomes

$$2h\left(\frac{f(x_{i-1}) + 4f(x_i) + f(x_{i+1})}{6} - \frac{h^2}{6} f''(x_i) + O(h^4)\right) + \frac{h^3}{3} f''(x_i) + O(h^5)$$

which can be rearranged to

$$\left[\frac{h}{3}\left(f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right) - \frac{h^3}{3} f''(x_i) + O(h^5)\right] + \frac{h^3}{3} f''(x_i) + O(h^5)$$

Canceling and combining the appropriate terms results in the integral expression

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \frac{h}{3}\left(f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right) + O(h^5)$$

Recognizing that $\frac{h}{3}\left(f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right)$ is exactly the Simpson's rule approximation for the integral over this subinterval, this equation implies that Simpson's rule is $O(h^5)$ over a subinterval and $O(h^4)$ over the whole interval. Because the h^3 terms cancel out exactly, Simpson's rule gains another *two* orders of accuracy!

① accuracy

② 收斂性 $O(h^4)$ ...

③ stability.

Problem4.4 Use Simpson's rule to approximate $\int_0^{\pi} \sin(x)dx$ with 11 evenly spaced grid points over the whole interval. Compare this value to the exact value of 2.

Ans:

```
import numpy as np

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_simp = (h/3) * (f[0] + 2*sum(f[:n-2:2]) \
         + 4*sum(f[1:n-1:2]) + f[n-1])
err_simp = 2 - I_simp

print(I_simp)
print(err_simp)
```

4.5 Computing integrals in python

The `scipy.integrate` subpackage has several functions for computing integrals. The `trapz` takes as input arguments an array of function values f computed on a numerical grid x.

Problem4.5 Use the `trapz` function to approximate $\int_0^{\pi} \sin(x)dx$ for 11 equally spaced points over the whole interval. Compare this value to the one computed in the earlier example using the trapezoid rule

Ans:

```
import numpy as np
from scipy.integrate import trapz

a = 0
b = np.pi
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)
f = np.sin(x)

I_trapz = trapz(f,x)
I_trap = (h/2)*(f[0] + 2 * sum(f[1:n-1]) + f[n-1])

print(I_trapz)
print(I_trap)
```

Problem4.6 Use the `cumtrapz` function to approximate the cumulative integral of $f(x) = \sin(x)$ from 0 to π, with a discretization step of 0.01. The exact solution of this integral is $F(x) = \sin(x)$. Plot the results.

Ans:

```
from scipy.integrate import cumtrapz
import matplotlib.pyplot as plt

%matplotlib inline
plt.style.use('seaborn-poster')

x = np.arange(0, np.pi, 0.01)
F_exact = -np.cos(x)
F_approx = cumtrapz(np.sin(x), x)

plt.figure(figsize = (10,6))
plt.plot(x, F_exact)
plt.plot(x[1::], F_approx)
plt.grid()
plt.tight_layout()
plt.title('$F(x) = \int_0^{x} sin(y) dy$')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.legend(['Exact with Offset', 'Approx'])
plt.show()
```

Problem4.7 Use the integrate.quad function to compute $\int_0^{\pi} \sin(x)dx$. Compare your answer with the correct answer of 2

Ans:

```
from scipy.integrate import quad

I_quad, est_err_quad = \
        quad(np.sin, 0, np.pi)
print(I_quad)
err_quad = 2 - I_quad
print(est_err_quad, err_quad)
```

4.6 Summary

1. Explicit integration of functions is often impossible or inconvenient, and numerical approaches must be used instead.
2. The Riemann integral, trapezoid rule, and Simpson's rule are common methods of approximating integrals.
3. Each method has an order of accuracy that depends on the approximation of the area below the function.