Topics covered

• Quadrature based on interpolation
  ○ Basic formulas; error analysis
  ○ Newton-Cotes formulas
  ○ Composite rules
  ○ Numerical stability of integration formulas
  ○ The method of undetermined coefficients

• Other variants
  ○ Gaussian quadrature
  ○ Romberg integration
  ○ Adaptive quadrature
1 Numerical quadrature

In this section we derive formulas to approximate a definite integral

\[ \int_{a}^{b} f(x) \, dx \]

for a continuous function \( f \) using its values at some finite set of nodes. This problem is called \textit{quadrature} (somewhat confusingly, ‘numerical integration’ refers to solving ODEs numerically).

1.1 Newton-Cotes formulas

First, we derive a basic set of rules that follow the same strategy that we used for differentiation. Given nodes \( x_0, x_1, \ldots, x_n \) in the interval \((a,b)\), construct the interpolating polynomial:

\[ f(x) = p_n(x) + E(x) = \sum_{k=0}^{n} f(x_k)\ell_k(x) + E(x). \]

Then

\[ \int_{a}^{b} f(x) \, dx = \int_{a}^{b} p_n(x) \, dx + \int_{a}^{b} E(x) \, dx. \]

Since \( p_n \) is a linear combination of the function values \( f(x_k) \) at the nodes, we end up with a formula

\[ \int_{a}^{b} f(x) \, dx \approx \sum_{k=0}^{n} c_k f(x_k) \]

for constants \( c_k \). There is a trick to evaluating the \( c_k \) (which involves integrating the Lagrange basis polynomials), which we will get to shortly. For now, let us derive a few standard formulas by brute force.

1.2 Degree of accuracy

Let \( \Pi_n \) be the space of polynomials of degree \( \leq n \). One desirable property of an approximation \( I[f] \) (e.g. an interpolant, derivative, integral) is that

\[ I[f] \text{ is exact for all } \Pi_n. \]

This property is sometimes called the degree of accuracy of the method. The order and degree of accuracy are closely related for methods based on interpolation, since the degree \( n \) interpolant is (trivially) exact for all \( f \in \Pi_n \).

This property can be used to derive relations for coefficients (see HW) or other facts, and can also be used to derive methods in the first place (see: undetermined coefficients later in these notes).
1.2.1 Trapzeoidal rule

We use two points $x_0 = a$ and $x_1 = b$. Let $h = b - a$. The Lagrange basis polynomials are

$$\ell_0 = \frac{-1}{h}(x - b), \quad \ell_1 = \frac{1}{h}(x - a).$$

We have

$$f(x) = f(a)\ell_0(x) + f(b)\ell_1(x) + \frac{f''(\xi_x)}{2}(x - a)(x - b).$$

Integrating,

$$\int_a^b f(x) \, dx = c_0 f(a) + c_1 f(b) + \int_a^b \frac{f''(\xi_x)}{2}(x - a)(x - b) \, dx$$

where

$$c_0 = \int_a^b \ell_0(x) \, dx = \left. \frac{-1}{2h}(x - b)^2 \right|_a^b = h/2, \quad c_1 = \int_a^b \ell_1(x) \, dx = h/2.$$

For the error term, use the mean value theorem for integrals. The function $(x - a)(x - b)$ is negative on $(a, b)$, so

$$\int_a^b \frac{f''(\xi_x)}{2}(x - a)(x - b) \, dx = \frac{f''(\eta)}{2} \int_a^b (x - a)(x - b) \, dx = -\frac{f''(\eta)}{12} h^3$$

for some $\eta \in (a, b)$. This yields the **trapezoidal rule**

$$\int_a^b f(x) \, dx = \frac{h}{2}(f(a) + f(b)) - \frac{f''(\eta)}{12} h^3.$$

Note the extra power of $h$ in the error; this is because as $h \to 0$ the size of the interval goes to zero. Roughly,

$$\int_a^b f(x) + \text{error} \, dx \approx \int_a^b f(x) \, dx + (b - a)(\text{error})$$

so integration over an interval should contribute a factor of $b - a$ to the error from approximating $f$. Importantly, the formula is exact for all $f \in \Pi_1$ (up to degree 1 polynomials).

1.2.2 Simpson’s rule

Let $x_0 = a$, $x_1 = (a + b)/2$ and $x_2 = b$ (three equally spaced points including the endpoints). After some work, we find that

$$\int_a^b f(x) \, dx = \frac{b - a}{6} \left( f(a) + 4f(\frac{a + b}{2}) + f(b) \right) + \text{error}.$$

In terms of the spacing $h = (b - a)/2$ between consecutive points,

$$\int_a^b f(x) \, dx \approx \frac{h}{3} \left( f(a) + 4f(a + h) + f(b) \right).$$
The error should have the form

$$\text{error} = \frac{1}{6} \int_a^b f^{(3)}(\xi)w(x) \, dx, \quad w(x) := (x - a)(x - x_1)(x - b)$$

However, $w(x)$ changes sign in $(a, b)$ so we cannot simply apply the mean value theorem. Moreover, since $w(x)$ has an odd symmetry around the midpoint $x_1$,

$$\int_a^b w(x) \, dx = 0.$$

This means that if $f$ is a cubic polynomial, the error should be zero (even though we interpolated a quadratic!) - which is indeed the case. Some more work is required (omitted here) to derive the formula with the true error term:

$$\int_a^b f(x) \, dx = \frac{b - a}{6} \left( f(a) + 4f\left(\frac{a + b}{2}\right) + f(b) \right) - \frac{1}{90} h^5 f^{(4)}(\eta).$$

Due to a symmetry, we have gained an extra order (and degree!) of accuracy.

### 1.2.3 More points!

For contrast, let’s add one more point, and let

$$x_0 = a, \ x_1 = a + h, \ x_2 = a + 2h, \ x_3 = b.$$

The result (tedious but straightforward to derive) is Simpson’s 3/8-ths rule:

$$\int_a^b f(x) \, dx = \frac{3h}{8} \left( f(a) + 3f(x_1) + 3f(x_2) + f(b) \right) - \frac{3}{80} h^5 f^{(4)}(\eta).$$

There is no symmetry to exploit here, so the error just has the expected form - which means that adding a point has not improved the error (in fact, coincidentally, the constant is even worse!).

### 1.2.4 Omitting endpoints: Midpoint rule

We do not need to use the endpoints in the interpolant. Take $x_0 = (a + b)/2$ (only one point). Then

$$\int_a^b f(x) \, dx = (b - a)f(x_0) + \int_a^b f'(\xi)(x - x_0) \, dx.$$

As with Simpson’s rule, $\int_a^b (x - x_0) \, dx = 0$ so the formula is exact up to degree 1 (despite using only one point). The result is that

$$\int_a^b f(x) \, dx = (b - a)f(x_0) - \frac{f''(\eta)}{24}(b - a)^3.$$

Surprisingly, the midpoint rule has a better error (by a factor of 2) than the trapezoidal rule!
1.3 Open and closed Newton Cotes formulas

In general, the **closed Newton-Cotes formulas** for \( \int_a^b f(x) \, dx \) are the formulas resulting from using equally spaced points

\[
x_k = a + kh, \quad k = 0, \ldots, n, \quad h = \frac{b-a}{n+1}.
\]

The **open Newton-Cotes formulas** use equally spaced points but exclude the endpoints (e.g. the midpoint rule).

\[
x_k = a + (k + 1)h, \quad k = 0, \ldots, n, \quad h = \frac{b-a}{n+2}.
\]

The errors follow the same pattern as the examples we just derived. For the \( n \)-th closed formula,

\[
\text{error} = \begin{cases} 
C_n f^{(n+1)}(\eta) h^{n+2} & n \text{ odd} \\
C_n f^{(n+2)}(\eta) h^{n+3} & n \text{ even}
\end{cases}.
\]

If \( n \) is odd the error is what we expect (one power of \( h \) better than the interpolant because the interval is size \( O(h) \) and exact for polynomials up to degree \( n \)). If \( n \) is even then by a symmetry, there is an extra order/degree of accuracy.

The open formulas have the same property.

However, Newton-Cotes formulas have a deficiency for large \( n \). Recall that high-degree, equally spaced interpolants can cause problems. The coefficients in the formula are eventually of mixed signs. This can lead to subtractive cancellation, which is not good!

For these reasons, high-order Newton-Cotes (about \( n = 9 \) and above) are not often used.

2 Composite rules

Suppose we want a good estimate of

\[
\int_a^b f(x) \, dx
\]

using points

\[
a = x_0 < x_1 < \cdots x_n = b.
\]

To avoid high order interpolation, we apply a simple rule to sub-intervals and add them up.

2.1 Composite trapezoidal rule

Use the trapezoidal rule in each interval \([x_k, x_{k+1}]\):

\[
\int_a^b f(x) \, dx = \sum_{k=0}^{n-1} \int_{x_k}^{x_{k+1}} f(x) \, dx \approx \sum_{k=0}^{n-1} \frac{1}{2} (x_{k+1} - x_k) (f(x_k) + f(x_{k+1})).
\]
Suppose that the points are equally spaced and let $h = x_{k+1} - x_k$. Then, including the error term derived earlier,

$$
\int_a^b f(x) \, dx = \sum_{k=0}^{n-1} \frac{h}{2} (f(x_k) + f(x_{k+1})) + E_T
$$

where

$$
E_T = -\sum_{k=0}^{n-1} \frac{f^{(2)}(\eta_k)}{12} h^3.
$$

where $\eta_k \in [x_k, x_{k+1}]$. We use the IVT trick to simplify, writing the sum as an average of $f^{(2)}$ values times something:

$$
E_T = -\sum_{k=0}^{n-1} \frac{f^{(2)}(\eta_k)}{12} h^3 = \frac{n h^3}{12} \left( \frac{1}{n} \sum_{k=0}^{n-1} f^{(2)}(\eta_k) \right) = \frac{n h^3}{12} f^{(2)}(\eta).
$$

Now note that $h = (b - a)/n$. Also suppose that

$$
|f^{(2)}(x)| \leq M \text{ for } x \in [a, b].
$$

It is convenient to have the error expressed in terms of $n$ only or $h$ only. The two forms are

$$
|E_T| \leq \frac{M(b - a)^3}{12n^2}, \quad |E_T| \leq \frac{M(b - a)}{12} h^2.
$$

Thus the error is $O(1/n^2)$ as $n \to \infty$ or alternatively $O(h^2)$ as $h \to 0$. Note that our rule of thumb still applies here: $f$ is approximated by linear functions with an $O(h^2)$ error, so the error in the integral should look like $(b - a)$ times something that is $O(h^2)$.

Note that for equally spaced points, the Trapezoidal rule can be simplified a bit to be

$$
\int_a^b f(x) \, dx \approx \frac{h}{2} f(a) + hf(x_1) + \cdots + hf(x_{n-1}) + \frac{h}{2} f(b).
$$

### 2.2 Composite Simpson’s rule

Assume there are an even number of sub-intervals ($n$ is even). Then we can use Simpson’s rule on the intervals

$$
[x_{2k}, x_{2k+2}], \quad k = 0, \cdots, n/2 - 1.
$$

The result is a formula with error

$$
E_S = -\frac{1}{180}(b - a)h^4 f^{(4)}(\eta).
$$

where $h = x_{k+1} - x_k$. 


3 Some additional details

3.1 Numerical stability

Suppose we have a formula

$$\int_a^b f(x) \, dx \approx A(f) = \sum_{i=0}^{n} c_i f(x_i)$$

but each function evaluation has some error when the formula is used:

$$\tilde{f}_i = f(x_i) + \delta_i, \quad |\delta_i| < \epsilon.$$

When does such a formula have good numerical stability, i.e. the $\delta_i$’s do not accumulate too much?

Integration, it turns out, is quite stable so long as the coefficients $c_i$ are all positive. Note that if the formula is at least exact for degree zero polynomials, then

$$b - a = \sum_{i=0}^{n} c_i.$$  

We then have (denoting the approx. with errors in $f$ by $A(\tilde{f})$)

$$A(\tilde{f}) = \sum_{i=0}^{n} c_i (f(x_i) + \delta_i) = A(f) + \sum_{i=0}^{n} c_i \delta_i.$$  

If the $c_i$’s are positive then

$$\left| \sum_{i=0}^{n} c_i \delta_i \right| \leq \epsilon \sum_{i=0}^{n} c_i = \epsilon (b - a).$$  

Thus errors in $f$ grow by at worst a factor of $b - a$, independent of the number of points. For this reason, integration is considered a ‘good’ numerical operation - the opposite of differentiation.

Note that if the $c_i$’s are not all positive then the bound is instead

$$\epsilon \sum_{i=0}^{n} |c_i|$$

which might be much larger than $b - a$ (this is the case. e.g. for high order Newton-Cotes formulas).
3.2 the method of undetermined coefficients

Suppose we want a formula for $\int_a^b f(x) \, dx$ that looks like

$$A[f] = \sum_{i=0}^{m} c_i f(x_i) \approx \int_a^b f(x) \, dx.$$ 

One way to derive the formula is to impose the degree of accuracy condition:

$A[f]$ is exact for $f \in \Pi_n$

where $n$ is chosen to be as large as possible. It suffices to require this for a basis of $\Pi_n$, typically $1, x-a, (x-a)^2, \cdots (x-a)^n$ or just $1, x, x^2, \cdots x^n$ (why?). This provides $n + 1$ equations (although really we just keep increasing $n$ until there are no coefficients left to choose).

For example, consider trying to find a formula

$$\int_0^1 f(x) \, dx \approx c_0 f(0) + c_1 f(1/2) + c_2 f(1).$$

We have three coefficients, so we can try to make this formula exact for polynomials up to degree $n \leq 2$. Plugging in $f = 1, x, x^2$, the formula is exact in $\Pi_2$ if and only if

$$1 = c_0 + c_1 + c_2$$
$$1/2 = c_1/2 + c_2$$
$$1/3 = c_1/4 + c_2$$

which can be solved to obtain $c_0 = c_2 = 1/6$ and $c_1 = 2/3$.

Now we check if the formula happens to be exact for the next degree:

$$1/4 = c_1/8 + c_2.$$ 

This is true, so we get an extra degree of accuracy for free (it is easy to check that $x^4$ is not integrated exactly). The result is just Simpson’s rule for the interval $[0, 1]$.

The benefit of the method is that it works for more exotic arrangements of formulas, as long as they are linear in the function input, e.g.

$$\int_0^\infty f(x)e^{-x} \, dx \approx c_0 f(0) + c_1 f(10).$$

When the points are free to be chosen, there is an elegant theory that can be employed to avoid the ‘brute force’ solving of a system of equations (see Gaussian Quadrature).
3.3 Change of interval

By a change of variables, we can transform any interval \([a, b]\) to another without changing the integral much. This means that formulas can be expressed in terms of some fixed interval (this is often the case). For example, suppose we have the rule above for \([0, 1]\) and want to derive it for a general interval. Let \(x = \tau(t)\) be a linear function that maps \(t = 0\) to \(x = a\) and \(t = 1\) to \(x = b\) (so as \(x\) goes from \(a\) to \(b\), \(t\) goes from 0 to 1). This is, explicitly,

\[
\tau(t) = a + (b - a)t.
\]

Make the change of variables \(x = \tau(t)\) (with \(dx = \tau'(t)dt = (b - a)dt\))

\[
\int_a^b f(x) \, dx = \int_0^1 f(\tau(t)) \, dt = (b - a) \int_0^1 f(\tau(t)) \, dt.
\]

Now we can apply the rule on the given interval to \(f \circ \tau\):

\[
\int_0^1 f(\tau(t)) \, dt \approx c_0 f(\tau(0)) + c_1 f(\tau(1/2)) + c_2 f(\tau(1)).
\]

The result here will be Simpson’s rule for \([a, b]\). Since composing \(f\) with \(\tau\) does not change the degree when \(f\) is a polynomial, the degree of accuracy is the same.

The benefit is that we can, without loss of generality, derive definite integral formulas for some convenient interval only.

4 A few other integration methods

4.1 Gaussian quadrature (briefly)

Suppose we are given an interval - say, \([-1, 1]\) - on which we need to integrate a collection of functions quickly and to a desired accuracy. Rather than use equally spaced points, it may make sense to choose an ‘optimal’ arrangement of points.

Let \(w(x) > 0\) be a weight function (often just \(w(x) = 1\)) and let \([a, b]\) be an interval. Gaussian quadrature is concerned with finding \(n + 1\) nodes \(x_i\) and coefficients \(c_i\) such that

\[
\int_a^b f(x)w(x) \, dx \approx \sum_{i=0}^{n} c_i f(x_i)
\]

has the highest possible degree of accuracy. From before, we know what the \(c_i\)'s are \((\int_a^b \ell_i(x) \, dx)\) but now the nodes are free to be chosen as well. Because we have \(2n + 2\) variables at our disposal, we expect to get a formula exact for polynomials with degree \(\leq 2n + 1\), which is indeed the case.

It turns out there is a rather elegant way of determining these nodes (see textbook), which we will not address here. Unfortunately, some numerical work is required to computed them,
so the values of $x_i$’s and $c_i$’s are generally tabulated in advance.

As a simple example without using the theory, let us consider $w(x) = 1$ and $n = 1$ and the interval $[-1, 1]$: 

$$
\int_{-1}^{1} f(x) \, dx \approx c_0 f(x_0) + c_1 f(x_1).
$$

The formula should be the same if we transform $x \to -x$, so $x_1 = -x_0$ and $c_0 = c_1$. This gives 

$$
\int_{-1}^{1} f(x) \, dx \approx c_0(f(-x_0) + f(x_0)).
$$

Now $x^n$ for any odd degree is exactly integrated (to zero). Requiring that the formula be exact for 1 and $x^2$, we find that 

$$
2 = 2c_0, \quad \frac{2}{3} = 2x_0^2
$$

so $x_0 = 1/\sqrt{3}$ and $c_0 = 1$. Thus the ‘best’ choice of two nodes in $[-1, 1]$ is $\pm 1/\sqrt{3}$, which gives a formula exact for polynomials up to degree 3.

This brute force method stops being feasible with more points (note that the equations are not linear!), necessitating the use of the theory.

### 4.2 Romberg integration

Richardson extrapolation can be applied to the trapezoidal rule. However, to do so, we need an asymptotic error formula. Let 

$$
x_i = a + ih, \quad i = 0, \cdots, n.
$$

The Euler-Maclaurin formula relates a definite integral to a sum, giving the error as an asymptotic series. In a suitable form for this discussion, the formula states that 

$$
\frac{h}{2} \sum_{i=0}^{n} (f(x_i) + f(x_{i+1})) = \int_a^b f(x) \, dx + \sum_{k=1}^{m-1} A_{2k} h^{2k} (f^{(2k-1)}(b) - f^{(2k-1)}(a)) + A_{2m} (b-a) h^{2m} f^{(2m)}(\xi)
$$

for some $\xi \in (a, b)$. Here $A_{2k}$ is a particular sequence of numbers (the Bernoulli numbers). Note that $m$ can be any integer $\geq 2$. The formula gives an infinite series 

$$
T(n) = \int_a^b f(x) \, dx + c_2 h^2 + c_4 h^4 + c_6 h^6 + \cdots
$$

for the composite trapezoidal rule $T(n)$ with $n + 1$ equally spaced points. Note that the coefficients actually diverge as $m \to \infty$, so the series does not converge; to be precise, we would need to only work with finitely many terms at a time.

Richardson extrapolation can now be applied by computing the composite trapezoidal rule for $2, 4, 8, \cdots$ points. This is the method of Romberg integration. As a benefit, $T(2n)$ can be efficiently computed given $T(n)$ since most of the sum can be recycled. For details, see the textbook.
4.3 Adaptive integration

Let \( I = \int_a^b f(x) \, dx \) and suppose we want to create an ‘adaptive’ scheme that takes in \( f \), the interval \([a, b]\) and a value \( \epsilon \) and returns an approximation to \( I \) with error at most \( \epsilon (b - a) \). The idea is that the scheme should be able to assess its own accuracy and do more work as needed without help from the user.

One trick for estimating error - as in Richardson extrapolation - is to compute two approximations with error terms of the same form and then ‘solve’ for the error. For example, let \( S(a, b) \) denote Simpson’s rule in \((a, b)\):

\[
S(a, b) := \frac{b - a}{6} (f(a) + 4 f\left(\frac{a + b}{2}\right) + f(b)).
\]

A more accurate approximation can be computed by splitting the interval in half and applying Simpson’s rule to both halves. Let \( h = (b - a)/2 \) and \( c = (a + b)/2 \). Then

\[
I = S(a, b) - \frac{f^{(4)}(\xi_1)}{90} h^5
\]

\[
I = S(a, c) + S(c, b) - \frac{f^{(4)}(\xi_2)}{90} (h/2)^5 - \frac{f^{(4)}(\xi_3)}{90} (h/2)^5.
\]

We want to estimate the error \( E = S(a, b) - I \). By assuming that the \( \xi \)'s are equal, we get two (approximate) equations with two unknowns, \( I \) and \( E \):

\[
I = S(a, b) - E,
\]

\[
I \approx S(a, c) + S(c, b) - \frac{1}{16} E.
\]

Solving for \( E \), we get

\[
\frac{|E|}{15} \approx |S(a, c) + S(c, b) - S(a, b)|. \tag{1}
\]

The algorithm can then proceed recursively as follows, given \( \epsilon, a, b \) and \( f \):

- Compute the approximate integrals \( S_1 = S(a, b) \) and \( S_2 = S(a, c) + S(c, b) \)
- Estimate the error \( E \) in \( S_1 \) using (1).
- If \( |E| < \epsilon (b - a) \), return \( S_2 \) (since it is likely more accurate than \( S_1 \)).
- If \( |E| > \epsilon (b - a) \) then apply the algorithm in \([a, c]\) and \([c, b]\) (recursively).

The algorithm recursively splits the interval in half until a sub-interval is accepted. The result is a a collection of subintervals \([a_j, b_j]\) for which

\[
\text{error in } \int_{a_j}^{b_j} f(x) \, dx < \epsilon (b_j - a_j)
\]
Once added up, the total error will be bounded by $\epsilon(b - a)$.

Note that it is not an absolute guarantee, since our error estimate relied on some rough simplification; however, if $f^{(4)}$ does not vary too much the error estimate should be good enough.

It is usually most efficient to avoid writing the algorithm recursively in practice, instead using a while loop and some careful bookkeeping. Also, the algorithm should have a maximum number of ’levels’ of recursion in case the error tolerance is impossible to achieve.