MATH3511/6111: Scientific Computing

07. Quadrature

Lindon Roberts

Semester 1, 2022

Office: Hanna Neumann Building #145, Room 4.87
Email: lindon.roberts@anu.edu.au
Based on lecture notes written by S. Roberts, L. Stals, Q. Jin, M. Hegland, K. Duru.

Quadrature

The goal of this section is to approximate integrals:

$$\int_a^b f(x)dx,$$

using only evaluations of $f(x)$ at particular points.

This process is called numerical integration or quadrature ("calculating areas").

Of course, we can calculate some integrals in closed form, if $f(x)$ has a simple form (which we know). Often in practice this is not the case.

Applications of Quadrature

Quadrature is a very important task in

- Geometry: areas & volumes of regions
- Physics: mass from density, energy from energy density, flow from flux, etc.
- Probability: expectations, averages, covariances, etc.
- Finance: option pricing, risk measures
- Solving differential equations

Riemann Sums

Like we did for differentiation, let's start with the definition using Riemann sums (Bernhard Riemann, 1854):

$$\int_a^b f(x)dx = \lim_{\Delta x \to 0} \sum_{i=0}^{n-1} (x_{i+1} - x_i) f(\widetilde{x}_i),$$

for some subdivision $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ and some $\widetilde{x}_i \in [x_i, x_{i+1}]$. The limit exists whenever f is continuous on $[a, b]$.

A simple approximation, therefore, is to use a fixed subdivision $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ to get

$$\boxed{\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} h_i f(\widetilde{x}_i)}$$

where $h_i = x_{i+1} - x_i$ and $\widetilde{x}_i \in [x_i, x_{i+1}]$ (e.g. $\widetilde{x}_i = x_i$ or $\widetilde{x}_i = x_{i+1}$).

Riemann Sums

Using equally spaced points and $\widetilde{x}_i = x_i$ (left-hand endpoint), this looks like:

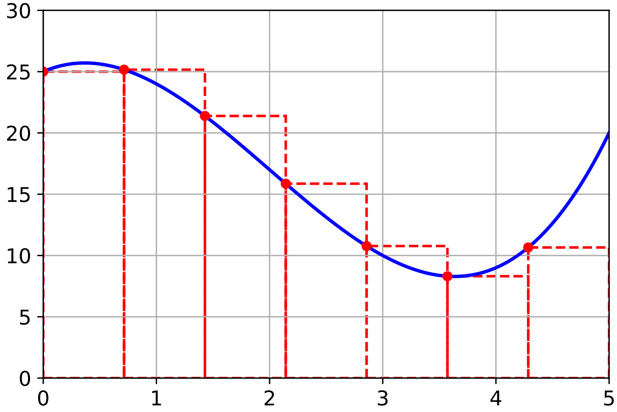


Figure 1. Riemann Sum example.

Riemann Sums

- As long as $h = \max_i h_i$ converges to zero, this approximation will converge to the true value of the integral (as long as f is continuous).
- However, taking h small requires evaluating $f(x)$ at many points $f(x_i)$.
  - We will look at ways to reduce the number of evaluation points, while keeping high accuracy approximations.
- Unlike differentiation, there is no subtraction of similar values or division by small numbers, so rounding error is not usually a problem.

Naming: for quadrature, usually the name of the rule refers to the approximation over one interval $[x_i, x_{i+1}]$. The composite rule is the sum of these to do integration over the whole interval $[a, b]$.

Riemann Sums: Analysis

Let's analyse the error in our approximation.

First, let's just look at the approximation for a single term in the sum:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx (x_{i+1} - x_i)f(\widetilde{x}_i).$$

To make the analysis easy, we will assume f is differentiable, so we have the Taylor polynomial

$$f(x) = f(\widetilde{x}_i) + f'(\xi(x))(x - \widetilde{x}_i),$$

for some $\xi(x)$ between $\widetilde{x}_i$ and x.

Riemann Sums: Analysis

Our error term can be written as

$$\begin{aligned}\int_{x_i}^{x_{i+1}} f(x)dx - (x_{i+1}-x_i)f(\widetilde{x}_i) &= \int_{x_i}^{x_{i+1}} [f(x) - f(\widetilde{x}_i)]dx, \\ &= \int_{x_i}^{x_{i+1}} f'(\xi(x))(x-\widetilde{x}_i)dx.\end{aligned}$$

Defining $M := \max_{x\in[a,b]} |f'(x)|$, and noting $|x-\widetilde{x}_i| \le x_{i+1}-x_i$ for all $x\in[x_i,x_{i+1}]$, we get

$$\left|\int_{x_i}^{x_{i+1}} f(x)dx - (x_{i+1}-x_i)f(\widetilde{x}_i)\right| \le \int_{x_i}^{x_{i+1}} Mh_i\, dx = Mh_i^2 \le Mh^2.$$

To make h as small as possible (for a fixed number of intervals n), we use equally spaced points:

$$x_i = a + ih, \qquad h = h_i = \frac{b-a}{n}.$$

Riemann Sums: Analysis

Now, we get our overall error by summing the error from each term:

$$\begin{aligned}\left|\int_a^b f(x)dx - \sum_{i=0}^{n-1} h_i f(\widetilde{x}_i)\right| &= \left|\sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(x)dx - \sum_{i=0}^{n-1} h_i f(\widetilde{x}_i)\right|, \\ &\leq \sum_{i=0}^{n-1}\left|\int_{x_i}^{x_{i+1}} f(x)dx - h_i f(\widetilde{x}_i)\right|, \\ &\leq Mh^2 n, \\ &= M\frac{(b-a)^2}{n} = M(b-a)h,\end{aligned}$$

where the last line is true if we have equally spaced points.

This is not a good result: to get 10 significant figures of accuracy, we need to evaluate $f(x)$ at $n \approx 10^{10}$ points!

Note: if $\widetilde{x}_i = (x_i + x_{i+1})/2$ (the "midpoint rule"), then we can get a better error — more on this later.

Riemann Sums

Using Riemann sums to estimate $\int_0^1 2x\sin x + x^2\cos x \, dx$, we get:

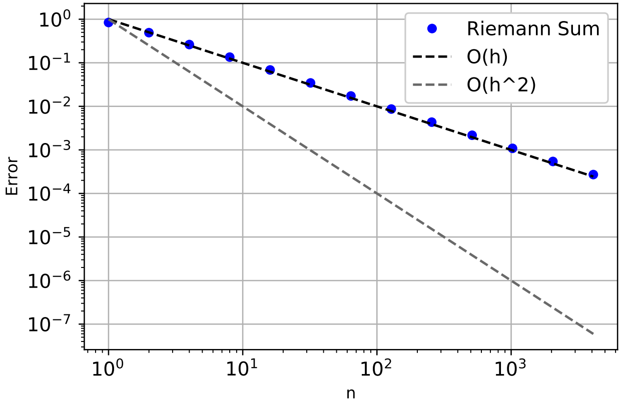


Figure 2. Riemann sum errors for increasing *n* (equally spaced points).

Trapezoidal Rule

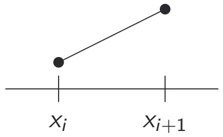
The Riemann sum formula comes from approximating $f(x)$ with functions that are constant on $[x_i, x_{i+1}]$. How can we improve on this?

The trapezoidal rule approximates $f(x)$ on $[x_i, x_{i+1}]$ by the linear function passing through $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$:

$$f(x) \approx p_1(x) = f(x_i)\frac{x_{i+1} - x}{x_{i+1} - x_i} + f(x_{i+1})\frac{x - x_i}{x_{i+1} - x_i}.$$

We get

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \int_{x_i}^{x_{i+1}} p_1(x)dx = \frac{f(x_i) + f(x_{i+1})}{2(x_{i+1} - x_i)} = \frac{f(x_i) + f(x_{i+1})}{2} h_i.$$



Trapezoidal Rule

Summing this approximation over the full interval, we get the composite trapezoidal rule

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

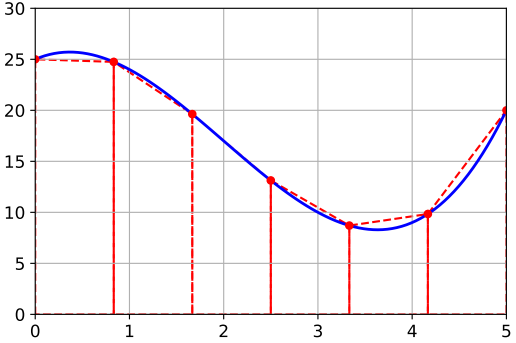


Figure 3. Composite trapezoidal rule example.

Trapezoidal Rule: Analysis

We can analyse the trapezoidal rule in a similar way to Riemann sums (but we need to use a second-order Taylor series approximation). Our analysis will be a bit different.

Theorem (Mean Value Theorem, integral form)

If $f : [a, b] \to \mathbb{R}$ and $g : [a, b] \to \mathbb{R}$ are continuous and $g(x) \geq 0$, then there exists $\xi \in [a, b]$ such that

$$\int_a^b f(x)g(x)dx = f(\xi) \int_a^b g(x)dx.$$

For example, if $g(x) = 1$ we get the standard integral MVT: there exists $\xi \in [a, b]$ such that

$$\int_a^b f(x)dx = f(\xi)(b - a).$$

Trapezoidal Rule: Analysis

Since we approximate the integral of $f(x)$ with the integral of the linear interpolant $p_1(x)$, we start with the error formula for interpolation and then integrate. We assume f is twice continuously differentiable.

The error for the linear interpolant $p_1(x)$ on $x \in [x_i, x_{i+1}]$ is (see interpolation lectures)

$$f(x) - p_1(x) = \frac{f''(\xi(x))}{2}(x - x_i)(x - x_{i+1}) = -\frac{f''(\xi(x))}{2}(x - x_i)(x_{i+1} - x),$$

for some $\xi \in [x_i, x_{i+1}]$. We now integrate and use the MVT with $g(x) = (x - x_i)(x_{i+1} - x) \geq 0$:

$$\begin{aligned}
\int_{x_i}^{x_{i+1}} f(x) - p_1(x) dx &= -\frac{1}{2}\int_{x_i}^{x_{i+1}} f''(\xi(x))(x - x_i)(x_{i+1} - x) dx, \\
&= -\frac{f''(\xi_i)}{2}\int_{x_i}^{x_{i+1}} (x - x_i)(x_{i+1} - x) dx, \\
&= -\frac{1}{12} f''(\xi_i) h_i^3,
\end{aligned}$$

for some $\xi_i \in [x_i, x_{i+1}]$.

Trapezoidal Rule: Analysis

To get a composite error bound, it is easiest to assume equally spaced points, $h = h_i = (b-a)/n$.

$$\int_a^b f(x)dx - \sum_{i=0}^{n-1} \frac{f(x_i)+f(x_{i+1})}{2} h = -\frac{h^3}{12} \sum_{i=0}^{n-1} f''(\xi_i).$$

Since $f''(\xi)$ is continuous, it takes all values between the largest and smallest $f''(\xi_i)$. Therefore $f''(\xi) = \frac{1}{n}\sum_{i=0}^{n-1} f''(\xi_i)$ for some $\xi \in [a,b]$ (intermediate value theorem). Hence

$$\int_a^b f(x)dx - \sum_{i=0}^{n-1} \frac{f(x_i)+f(x_{i+1})}{2} h = -\frac{1}{12}(b-a)f''(\xi)h^2, \qquad \text{some } \xi \in [a,b].$$

If $|f''(\xi)| \le M$ for all $\xi \in [a,b]$, we get

$$\left| \int_a^b f(x)dx - \sum_{i=0}^{n-1} \frac{f(x_i)+f(x_{i+1})}{2} h \right| \le \frac{M}{12}(b-a)h^2.$$

For non-equally spaced points, the error formula is the same, but with $h = \max_i h_i$.

Trapezoidal Rule: Analysis

We can also rearrange this formula to get constants that are easier to calculate than M (an upper bound on $f''(\xi)$). We get

$$\int_a^b f(x)dx - \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} h = -\frac{h^2}{12}\left[\sum_{i=0}^{n-1} hf''(\xi_i)\right].$$

The term in brackets is the Riemann sum approximation of $\int_a^b f''(x)dx$, so using that error formula we get

$$\int_a^b f(x)dx - \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} h = -\frac{h^2}{12}\left[f'(b) - f'(a) + \mathcal{O}(h)\right],$$

$$\left|\int_a^b f(x)dx - \sum_{i=0}^{n-1} \frac{f(x_i) + f(x_{i+1})}{2} h\right| \le \frac{h^2}{12}|f'(b) - f'(a)| + \mathcal{O}(h^3).$$

It is easier to calculate $f'(b) - f'(a)$ than M (but we introduce a small $\mathcal{O}(h^3)$ error).

Trapezoidal Rule: Example

Again, estimating $\int_0^1 2x\sin x + x^2\cos x\, dx$ we get:

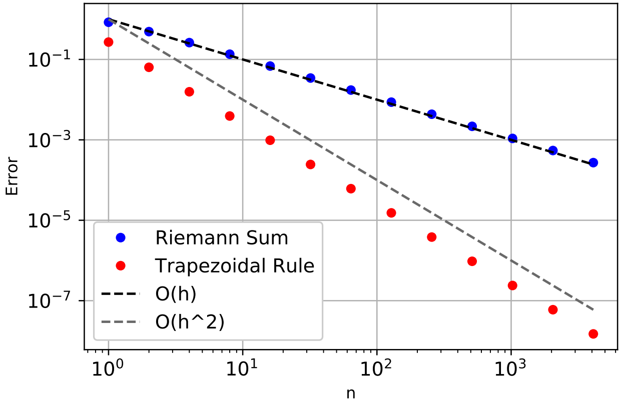


Figure 4. Composite trapezoidal rule errors for increasing n (equally spaced points).

Euler-Maclaurin Formula

Just like for numerical differentiation, we want to try and use Richardson extrapolation to build higher-order methods. To do this, we need a rule that has an error term that obeys a power law error.

Let I be the true integral and $T(f, h)$ be the result of the composite trapezoidal rule with grid size h. We want

$$I = T(f, h) + C_1 h + C_2 h^2 + \cdots ,$$

where C_i don't depend on h. Fortunately this is true (and we only have even powers of h).

Theorem (Euler-Maclaurin Formula)

If f and its derivatives up to order $2r+2$ are continuous on $[a, b]$, then

$$I = T(f, h) + C_2 h^2 + C_4 h^4 + \cdots + C_{2r} h^{2r} + \mathcal{O}(h^{2r+2}),$$

where $C_2 = [f'(b) - f'(a)]/12$ and $C_{2m} = c_{2m}[f^{(2m-1)}(b) - f^{(2m-1)}(a)]$ in general.

Romberg Integration

Now we are in a position to do Richardson extrapolation. When applied to the composite trapezoidal rule, this is called Romberg integration (Werner Romberg, 1955).

Recall, we apply the composite trapezoidal rule with different values of h:

$$\begin{aligned} I &= T(f,h) + C_2 h^2 + C_4 h^4 + \mathcal{O}(h^6), \\ &= T(f,h/2) + \frac{1}{4} C_2 h^2 + \frac{1}{16} C_4 h^4 + \mathcal{O}(h^6). \end{aligned}$$

We can cancel the $\mathcal{O}(h^2)$ terms using:

$$I = \underbrace{\frac{4}{3} T(f,h/2) - \frac{1}{3} T(f,h)}_{=T_2(f,h/2)} - \frac{1}{4} C_4 h^4 + \mathcal{O}(h^6)$$

Then build $T_3(f,h/4)$ from $T_2(f,h/2)$ and $T_2(f,h/4)$ to cancel the $\mathcal{O}(h^4)$ terms, etc.

Romberg Integration

Suppose we want the highest accuracy approximation $T_{m+1}(f, h/2^m)$. Working backwards, we need to calculate $T_m(f, h/2^{m-1})$ and $T_m(f, h/2^m)$, etc., and ultimately $T(f,h), \ldots, T(f, h/2^m)$.

We can do this easily by starting with $h = b - a$:

$$T(f, b-a) = \frac{f(a)+f(b)}{2}(b-a).$$

Then we try grid size $h/2 = (b-a)/2$:

$$\begin{aligned} T(f, h/2) &= \frac{f(a)+f(a+h/2)}{2} \cdot \frac{h}{2} + \frac{f(a+h/2)+f(b)}{2} \cdot \frac{h}{2}, \\ &= \frac{1}{2} T(f,h) + \frac{h}{2} f(a+h/2), \end{aligned}$$

by noticing that all the terms in $T(f,h)$ also appear in $T(f, h/2)$.

Romberg Integration

Next, for $h/4 = (b-a)/4$:

$$\begin{aligned}
T(f, h/4) &= \frac{f(a) + f(a+h/4)}{2} \cdot \frac{h}{4} + \frac{f(a+h/4) + f(a+h/2)}{2} \cdot \frac{h}{4} \\
&\qquad + \frac{f(a+h/2) + f(a+3h/4)}{2} \cdot \frac{h}{4} + \frac{f(a+3h/4) + f(b)}{2} \cdot \frac{h}{4}, \\
&= \frac{1}{2} T(f, h/2) + \frac{h}{4} \left[f(a+h/4) + f(a+3h/4) \right].
\end{aligned}$$

In general, it is easy to generate $T(f, h/2^{i+1})$ from $T(f, h/2^i)$:

$$T(f, h/2^{i+1}) = \frac{1}{2} T(f, h/2^i) + \frac{h}{2^{i+1}} \sum_{k=0}^{2^i - 1} f\left(a + (1+2k)h/2^{i+1}\right)$$

Romberg Integration

All together, we end up building the hierarchy of approximations:

$$\begin{tikzcd}
T(f,h) \arrow[dr] & & & & & \\
T(f,h/2) \arrow[r] \arrow[dr] & T_2(f,h/2) \arrow[dr] & & & & \\
T(f,h/4) \arrow[r] & T_2(f,h/4) \arrow[r] & T_3(f,h/4) & & & \\
\vdots \arrow[dr] & \vdots \arrow[dr] & \vdots & \ddots & & \\
T(f,h/2^{m-1}) \arrow[r] \arrow[dr] & T_2(f,h/2^{m-1}) \arrow[r] \arrow[dr] & T_3(f,h/2^{m-1}) & \cdots & T_m(f,h/2^{m-1}) \arrow[dr] & \\
T(f,h/2^m) \arrow[r] & T_2(f,h/2^m) \arrow[r] & T_3(f,h/2^m) & \cdots & T_m(f,h/2^m) \arrow[r] & T_{m+1}(f,h/2^m)
\end{tikzcd}$$

We build the first column from top to bottom using the formula on the previous page. Going left-to-right between columns, we use Richardson extrapolation:

$$T_{j+1}(f,h/2^{i+1}) = \frac{4^j T_j(f,h/2^{i+1}) - T_j(f,h/2^i)}{4^j - 1}$$

Romberg Interpolation: Convergence

The Euler-Maclaurin formula gives some guarantees of convergence: if f has $2m+2$ continuous derivatives, the errors in T_{m+1} go to zero as the grid size gets small. That is,

$$\lim_{i\to\infty} T_{m+1}(f, h/2^i) = I.$$

Unfortunately, we have no guarantees that the 'best' approximation $T_{m+1}(f, h/2^m)$ converges to I as $m \to \infty$ (although in practice is usually does, and faster than the above sequence).

Romberg Interpolation: Implementation

In Python, this looks like:

```python
import numpy as np

def romberg(f, a, b, m):
    h = b - a
    # Store all the values in a matrix R, where R[i,j] = T_{i+1}(f, h / 2**(-j))
    R = np.zeros((m+1,m+1))
    R[0,0] = (f(a) + f(b)) * h/2
    for i in range(1, m+1):
        # Create the first column of R recursively (top to bottom)
        R[i,0] = 0.5 * R[i-1,0]
        for k in range(2**(i-1)):
            R[i,0] = R[i,0] + h/2**i * f(a + (1+2*k)*h/2**i)
    # Fill each column recursively
    for j in range(1, m+1):
        for i in range(j, m+1):
            R[i,j] = (4**j * R[i,j-1] - R[i-1,j-1])/(4**j-1)
    return R[m,m] # return value T_{m+1}(f, h/2**m)
```

Romberg Interpolation: Implementation

Again trying to calculate $\int_0^1 2x\sin x + x^2\cos x\,dx$, with $m=4$ we get

```
[[1.111622137742 0.             0.             0.              0.            ]
 [0.905221658409 0.836421498632 0.             0.              0.            ]
 [0.857183862895 0.84117126439  0.841487915441 0.              0.            ]
 [0.84538533285  0.841452489502 0.841471237842 0.841470973119  0.            ]
 [0.8424487076   0.841469832517 0.841470988718 0.841470984764  0.84147098481 ]]
```

The errors in these approximations are:

```
[[ 0.270151152934                                                               ]
 [ 0.063750673601 -0.005049486176                                               ]
 [ 0.015712878087 -0.000299720418  0.000016930633                               ]
 [ 0.003914348042 -0.000018495306  0.000000253034                               ]
 [ 0.000977722793 -0.000001152291  0.000000003910 -0.000000000044   0.000000000002]]
```

Our final approximation has 11 correct significant figures (absolute error 1.8×10^{-12}), but we only used 17 evaluations of $f(x)$!

Newton-Cotes Formulae

The trapezoidal rule (and Romberg integration) is based on linear interpolation of $f(x)$ on each interval $[x_i, x_{i+1}]$ (or a piecewise linear approximation on $[a, b]$).

What happens if we use higher-order interpolation on subintervals?

For equally spaced points, these are called Newton-Cotes formulae (Isaac Newton and Roger Cotes, around 1700).

In the case of quadratic approximation, this is called Simpson's rule (Thomas Simpson, mid-1700s).

Simpson's Rule

For quadratic interpolation, we need 3 interpolation points rather than 2. So, let's consider the double-sized interval $[x_0, x_2]$, and interpolate to x_0, x_1 and x_2.

First, we can compute the Lagrange polynomials for our nodes:

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \quad L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}, \quad L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)},$$

so the interpolating quadratic for $f(x)$ is

$$p_2(x) = \sum_{i=0}^{2} f(x_i)L_i(x).$$

Our quadrature rule on $[x_0, x_2]$ is given by integrating $p_2(x)$:

$$\int_{x_0}^{x_2} f(x)dx \approx \int_{x_0}^{x_2} p_2(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)],$$

for equally spaced points with spacing $h = x_1 - x_0 = x_2 - x_1$.

Composite Simpson's Rule

To get the composite Simpson's rule, we apply Simpson's rule to $[x_0, x_2]$, $[x_2, x_4]$, etc. (assuming we have $h = (b-a)/n$ for n even):

$$\int_a^b f(x)dx \approx \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] + \frac{h}{3}[f(x_2) + 4f(x_3) + f(x_4)] + \cdots + \frac{h}{3}[f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)],$$

and so we get

$$\boxed{\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]}$$

for equally spaced points $x_i = a + ih$ and $h = (b-a)/n$ for n even.

Method of Undetermined Coefficients

For higher-order Newton-Cotes formulae we use larger subintervals. Alternatively, we can derive these rules using the method of undetermined coefficients.

Key idea: quadratic interpolation is an exact approximation (zero error) if $f(x)$ is linear or quadratic. Therefore, the error in the quadrature rule will also be zero.

For Simpson's rule, we want a method

$$\int_{x_0}^{x_2} f(x)dx \approx a_0 f(x_0) + a_1 f(x_1) + a_2 f(x_2),$$

which is exact for all $f(x)$ which are polynomials of degree ≤ 2.

We can pick any basis for these polynomials (e.g. $f(x) = 1, x, x^2$). A choice which is easier to manipulate is $f(x) = 1,\ x - x_1,\ (x - x_1)^2$.

Method of Undetermined Coefficients

Substitute each $f(x)$ and require equality in

$$\int_{x_0}^{x_2} f(x)dx \approx a_0 f(x_0) + a_1 f(x_1) + a_2 f(x_2).$$

For equally spaced points with grid size h, we get the equations

$$\begin{aligned} 2h &= a_0 + a_1 + a_2, && \text{from } f(x) = 1, \\ 0 &= -a_0 h + a_2 h, && \text{from } f(x) = x - x_1, \\ \frac{2}{3}h^3 &= a_0 h^2 + a_2 h^2, && \text{from } f(x) = (x - x_1)^2. \end{aligned}$$

The solution of this system is $a_0 = h/3$, $a_1 = 4h/3$ and $a_2 = h/3$, which gives Simpson's rule.

Simpson's Rule: Exmaple

Again, estimating $\int_0^1 2x\sin x + x^2\cos x \, dx$ we get:

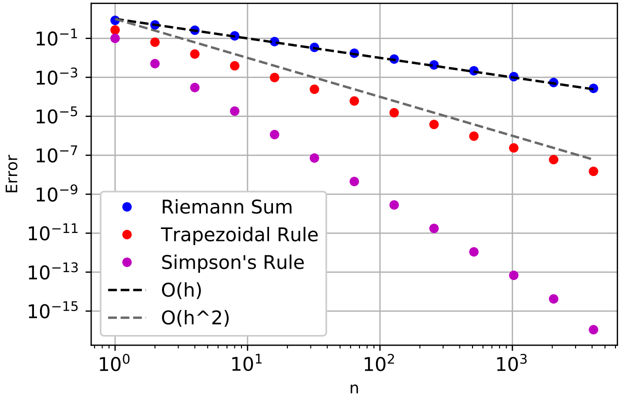


Figure 5. Composite Simpson's rule errors for increasing *n* (equally spaced points).

Simpson's Rule

A surprise!

Try Simpson's rule for $f(x) = (x - x_1)^3$. The true integral is

$$\int_{x_0}^{x_2} f(x)dx = 0.$$

Simpson's rule gives

$$\int_{x_0}^{x_2} f(x)dx \approx \frac{h}{3}[-h^3 + 0 + h^3] = 0.$$

So Simpson's rule is also exact for cubics — we get an extra order of accuracy for free!

Simpson's Rule: Analysis

The analysis of Simpson's rule is similar to the trapezoidal rule (but harder).

For a single interval $[x_0, x_2]$, we have

$$\int_{x_0}^{x_2} f(x)dx - \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] = -\frac{1}{90}f^{(4)}(\xi)h^5,$$

for some $\xi \in [x_0, x_2]$. We sum this over all subintervals in $[a, b]$, and we get the composite Simpson's rule error

$$|\text{error}| \leq \frac{b-a}{180}h^4 f^{(4)}(\xi),$$

for some $\xi \in [a, b]$. This is two orders of magnitude better than the trapezoidal rule $\mathcal{O}(h^2)$.

Since we went from linear to quadratic approximations, we would expect only $\mathcal{O}(h^3)$, but we got an extra order of accuracy for free.

Newton-Cotes Formulae: Analysis

In general for Newton-Cotes formulae, the error bounds depend on the degree of polynomials for which the formula is exact.

If the formula is accurate for all polynomials of degree $\leq n$, the error bound for a single interval will be of size $Cf^{(n+1)}(\xi)h^{n+1}$ or better.

The associated composite rule will have an error of size $\mathcal{O}(h^n)$.

Newton-Cotes Formulae

General comments:

- High-order formulae are not used often, as high-degree polynomial interpolation with equally spaced points loses accuracy (Runge phenomenon; see interpolation lectures)
  - Better to use a low-degree method like Simpson's rule, with smaller h.
- Polynomial interpolation is more accurate with Chebyshev points — can we use these?
  - Yes — Fejér or Clenshaw-Curtis rules use Chebyshev points or points where $T_n(x)$ is maximised/minimised (Lipót Fejér, 1933; Charles Clenshaw and A. R. Curtis, 1960).
  - These methods can be implemented quite efficiently for large n and can be very accurate.
  - The next method we study is more accurate in theory, and more widely used.

In general, Newton-Cotes methods are for situations where we can't choose the nodes x_i (e.g. only have collected data at certain points). We will see that being allowed to choose the x_i can give us more accurate methods.

Gaussian Quadrature

Previously, we have worked with quadrature formulae of the form

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

for equally spaced points x_i, and some coefficients c_i.

For the smallest error order (in terms of grid size h), we want our approximation to be exact for all polynomials of the highest degree possible. What is the best we can do?

Theorem

A quadrature rule with $n+1$ nodes cannot be exact for all polynomials of degree $\leq 2n+2$.

Proof: Consider the degree-$(2n+2)$ polynomial $p(x) = (x-x_0)^2(x-x_1)^2\cdots(x-x_n)^2$. Since $p(x) \geq 0$, we have $\int_a^b p(x)dx > 0$. However, each $p(x_i) = 0$, so for any weights c_i we have

$$\int_a^b p(x)dx > 0 = \sum_{i=0}^{n} c_i f(x_i).$$

Gaussian Quadrature

This theorem tells us we cannot be exact for all polynomials of degree $\leq 2n+2$. So, the best we can hope for is to be exact for degree $\leq 2n+1$.

The Newton-Cotes formula with $n+1$ points are exact for all polynomials of degree $\leq n$ (if n is odd) or $\leq n+1$ (if n is even, like Simpson's rule).

Question

Is there a rule that is exact for all polynomials of degree $\leq 2n+1$?

Yes, there is exactly one method that achieves this!

This is called Gaussian quadrature (Carl Friedrich Gauss, 1815).

Gaussian Quadrature: $n = 0$

Let's try to build the simplest Gaussian quadrature rule, with $n = 0$:

$$\int_a^b f(x)dx \approx c_0 f(x_0),$$

which should be exact for all polynomials of degree ≤ 1.

Using the method of undetermined coefficients, trying $f(x) = 1$ and $f(x) = x$ (a basis for all such polynomials), we get:

$$b - a = c_0, \qquad \text{and} \qquad \frac{b^2}{2} - \frac{a^2}{2} = c_0 x_0.$$

This gives us $c_0 = b - a$, so substituting this in and solving for x_0, we get $x_0 = (a+b)/2$.

So, the first Gaussian quadrature rule is

$$\int_a^b f(x)dx \approx (b-a) f\left(\frac{a+b}{2}\right).$$

This is just the midpoint rule! (see the section on Riemann sums)

Gaussian Quadrature: $n \geq 1$

Using the method of undetermined coefficients for $n \geq 1$ is much harder.

For example, for $n = 1$ with $f(x) = 1, x, x^2, x^3$, we get a system of 4 <u>nonlinear</u> equations:

$$\begin{aligned} b - a &= c_0 + c_1, \\ \frac{b^2}{2} - \frac{a^2}{2} &= c_0 x_0 + c_1 x_1, \\ \frac{b^3}{3} - \frac{a^3}{3} &= c_0 x_0^2 + c_1 x_1^2, \\ \frac{b^4}{4} - \frac{a^4}{4} &= c_0 x_0^3 + c_1 x_1^3. \end{aligned}$$

We get systems of polynomial equations in the c_i and x_i, which are usually hard to solve (algebraic geometry).

Fortunately, there is an easier way to derive the Gaussian quadrature rules!

Orthogonal Polynomials

We can construct Gaussian quadrature rules using orthogonal polynomials. For simplicity, we consider quadrature only on the interval $[-1, 1]$ (we can shift and scale to integrate over any other interval).

Definition

Two polynomials $p(x)$ and $q(x)$ are orthogonal on $[-1, 1]$ if

$$\int_{-1}^{1} p(x)q(x)dx = 0.$$

Remember, $\int_{-1}^{1} f(x)g(x)dx$ is an inner product on the vector space of continuous functions (or the vector space of polynomials of a given degree).

Legendre Polynomials

Remember for polynomial interpolation, we constructed an optimal collection of points using Chebyshev polynomials. They are not orthogonal (with the standard inner product).

Instead, we will use Legendre polynomials (Adrien-Marie Legendre, 1785). The n-th Legendre polynomial, $q_n(x)$, is a degree-n polynomial such that:

- The leading coefficient of $q_n(x)$ is 1 ("q_n is monic").
- $q_n(x)$ is orthogonal on $[-1,1]$ to all $q_0, \ldots, q_{n-1}$.

The first few Legendre polynomials are:

$$q_0(x) = 1, \qquad q_3(x) = x^3 - \frac{3}{5}x,$$

$$q_1(x) = x, \qquad q_4(x) = x^4 - \frac{6}{7}x^2 + \frac{3}{35},$$

$$q_2(x) = x^2 - \frac{1}{3}, \qquad \ldots$$

Constructing Legendre Polynomials

Every polynomial of degree $\leq n-1$ can be written as a linear combination of $\{q_0, \ldots, q_{n-1}\}$. So since each q_n is monic, we can use this basis to write:

$$\boxed{q_n(x) = x^n + c_{n-1}q_{n-1}(x) + c_{n-2}q_{n-2}(x) + \cdots c_1 q_1(x) + c_0 q_0(x)}$$

Suppose we know all $q_0, \ldots, q_{n-1}$ already. Then the orthogonality requirement is

$$\begin{aligned} 0 &= \int_{-1}^{1} q_n(x)q_i(x)dx, \\ &= \int_{-1}^{1} x^n q_i(x)dx + c_{n-1}\int_{-1}^{1} q_{n-1}(x)q_i(x)dx + \cdots + c_0\int_{-1}^{1} q_0(x)q_i(x)dx, \\ &= \int_{-1}^{1} x^n q_i(x)dx + c_i\int_{-1}^{1} q_i(x)^2 dx. \end{aligned}$$

Therefore we have $\boxed{c_i = -\left(\int_{-1}^{1} x^n q_i(x)dx\right) / \left(\int_{-1}^{1} q_i(x)^2 dx\right)}$.

Constructing Legendre Polynomials

$\boxed{n = 0}$ we must have $q_0(x) = 1$ (monic of degree zero).

$\boxed{n = 1}$ we write

$$q_1(x) = x + c_0 q_0(x),$$

where

$$c_0 = -\frac{\int_{-1}^{1} x q_0(x) dx}{\int_{-1}^{1} q_0(x)^2 dx} = -\frac{\int_{-1}^{1} x dx}{\int_{-1}^{1} 1 dx} = 0.$$

Therefore

$$q_1(x) = x.$$

Constructing Legendre Polynomials

$\boxed{n = 2}$ we write

$$q_2(x) = x^2 + c_1 q_1(x) + c_0 q_0(x),$$

where

$$c_0 = -\frac{\int_{-1}^{1} x^2 q_0(x) dx}{\int_{-1}^{1} q_0(x)^2 dx} = -\frac{\int_{-1}^{1} x^2 dx}{\int_{-1}^{1} 1 dx} = -\frac{1}{3},$$
$$c_1 = -\frac{\int_{-1}^{1} x^2 q_1(x) dx}{\int_{-1}^{1} q_1(x)^2 dx} = -\frac{\int_{-1}^{1} x^3 dx}{\int_{-1}^{1} x^2 dx} = 0.$$

Therefore

$$q_2(x) = x^2 - \frac{1}{3} q_0(x) = x^2 - \frac{1}{3}.$$

(and so on for larger n...)

Gaussian Quadrature

We construct the n-th Gaussian quadrature rule using the following:

Theorem

The Legendre polynomial $q_n(x)$ has n distinct roots in $[-1, 1]$.

Recipe:

- Set $x_0, \dots, x_n$ to be the $n+1$ distinct roots of q_{n+1} in $[-1, 1]$.
- Find the weights c_i using the method of undetermined coefficients for $f(x) = 1, x, \dots, x^n$.

Even though we only ensure exactness for degree $\leq n$ (in step 2), we get more degrees of exactness for free!

Theorem

The Gaussian quadrature rule with $n+1$ nodes is exact for all polynomials of degree $\leq 2n+1$.

Gaussian Quadrature

Let's derive the rules for small n.

$\boxed{n = 0}$ here, we take x_0 to be the root of $q_1(x) = x$. So, $x_0 = 0$.

Now apply the method of undetermined coefficients for $f(x) = 1$:

$$\int_{-1}^{1} f(x)dx = c_0 f(0),$$
$$2 = c_0,$$

so we get the Gaussian quadrature rule on $[-1, 1]$:

$$\boxed{\int_{-1}^{1} f(x)dx \approx 2f(0)}$$

which is the midpoint rule, as expected.

Gaussian Quadrature

$\boxed{n=1}$ we take x_0 and x_1 to be the roots of $q_2(x) = x^2 - 1/3$. Therefore $x_0 = -1/\sqrt{3}$ and $x_1 = 1/\sqrt{3}$.

Now apply the method of undetermined coefficients for $f(x) = 1, x$. We get

$$\begin{aligned} 2 &= \int_{-1}^{1} 1\, dx = c_0 + c_1, \\ 0 &= \int_{-1}^{1} x\, dx = -\frac{1}{\sqrt{3}} c_0 + \frac{1}{\sqrt{3}} c_1, \end{aligned}$$

we get $c_0 = c_1 = 1$.

Therefore the Gaussian quadrature rule on $[-1, 1]$ is

$$\boxed{\int_{-1}^{1} f(x) dx \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)}$$

Gaussian Quadrature

$\boxed{n=1}$ We have already checked it is exact for $f(x)=1, x$.

It should be exact for $f(x)=x^2, x^3$ also (up to degree $\leq 2n+1=3$):

$$\int_{-1}^{1} x^2 dx = \left[\frac{x^3}{3}\right]_{-1}^{1} = \frac{2}{3} = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right),$$
$$\int_{-1}^{1} x^3 dx = \left[\frac{x^4}{4}\right]_{-1}^{1} = 0 = f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right),$$

as expected.

$\boxed{n=2}$ with a bit of work, we can derive the rule

$$\boxed{\int_{-1}^{1} f(x)dx \approx \frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}f(0) + \frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right)}$$

This rule is exact for all polynomials of degree ≤ 5.

Gaussian Quadrature: General Intervals

This process works for quadrature rules on $[-1,1]$. What if we want to integrate on $[a,b]$?

We just need to shift and scale the nodes and weights:

$$x_i^{\mathrm{new}} = \frac{b-a}{2}x_i + \frac{a+b}{2}, \qquad \text{and} \qquad c_i^{\mathrm{new}} = \frac{b-a}{2}c_i.$$

Gaussian Quadrature: Composite Rule

How do we build a composite rule for Gaussian quadrature?

- Subdivide $[a, b]$ into $[x_0, x_1]$, $[x_1, x_2]$, ..., $[x_{n-1}, x_n]$.
- Apply the Gaussian quadrature rule on each subinterval separately, and sum the results.

Note that this means we need to evaluate f at $n+1$ points in <u>each</u> subinterval. This is different to Newton-Cotes, where we only evaluate f at the end-points of each subinterval.

Gaussian Quadrature: Analysis

Theorem

The Gaussian quadrature rule has error

$$\int_a^b f(x)dx - \sum_{i=0}^{n} c_i f(x_i) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}\int_a^b \prod_{i=0}^{n}(x-x_i)^2 dx,$$

for some $\xi \in [a,b]$.

Therefore the composite Gaussian quadrature rule has error $\mathcal{O}(h^{2n+2})$.

However, we also have a nice consistency result, that we don't have for Newton-Cotes or Romberg integration. This is for the regular rule, not the composite rule.

Theorem

If f is continuous, the n-point Gaussian quadrature rules converge to the true integral as $n \to \infty$.

In practice, however, it is better to use the composite rule rather than take n very large.

Gaussian Quadrature: Results

Again, let's estimate $\int_0^1 2x\sin x + x^2\cos x \, dx$. We compare the composite trapezoidal rule, composite Simpson's rule and Romberg integration against (non-composite) Gaussian quadrature, measuring the absolute error vs. number of evaluations of the integrand.

n	Evals	Comp. Trapezoidal	Comp. Simpson's	Romberg	Gaussian Quad.
0	1	—	—	—	0.14264 98057 31100
1	2	0.27015 11529 34070	—	0.27015 11529 34070	0.00338 13318 86391
2	3	0.06375 06736 01485	0.00504 94861 76044	0.00504 94861 76044	0.00001 62883 97267
4	5	0.01571 28780 86970	0.00029 97204 17869	0.00001 69306 32676	0.00000 00000 35651
8	9	0.00391 43480 41958	0.00001 84953 06380	0.00000 00116 89396	0.00000 00000 00000
16	17	0.00097 77227 92553	0.00000 11522 90582	0.00000 00000 01792	0.00000 00000 00000

The higher accuracy of Gaussian quadrature is clear, as is the improvement of extrapolation in Romberg integration (compared to composite trapezoidal rule).

Other Gaussian Quadrature Rules

Gaussian quadrature is based on roots of Legendre polynomials, which are orthogonal via

$$(f,g) = \int_{-1}^{1} f(x)g(x)dx.$$

In general, if we want to integrate

$$\int_a^b f(x)w(x)dx,$$

for some known weight function $w(x) > 0$, then we choose nodes to be the roots of polynomials which are orthogonal with $(f,g)_w = \int_a^b f(x)g(x)w(x)dx$.

- The "Gauss-Legendre quadrature" rules above are for $w(x) = 1$, $(a,b) = (-1,1)$.
- If $w(x) = 1/\sqrt{1-x^2}$, $(a,b) = (-1,1)$, then we get Chebyshev polynomials ("Gauss-Chebyshev").
- If $w(x) = e^{-x^2}$, $(a,b) = (-\infty,\infty)$, we need "Hermite polynomials" (not discussed in this course) — this is useful in probability (integrals related to normal distributions).

Multiple Integrals

What can we do if we want to calculate multiple integrals (i.e. integrals over higher dimensional regions), such as

$$\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dx\,dy,$$

or perhaps even more complicated regions? There are two main options:

- Use a 1D quadrature rule to approximate the inner integral ($\int\ dx$) in terms of evaluations of $f(x,y)$: e.g. with the midpoint rule

$$\int_{-1}^{1}\left(\int_{-1}^{1} f(x,y)\,dx\right)\,dy \approx \int_{-1}^{1} 2f(0,y)dy.$$

  Then use a 1D quadrature rule on this integral (with respect to y).
- Divide the higher-dimensional region (e.g. the box $[-1,1]^2 \subset \mathbb{R}^2$) into regions (e.g. boxes, triangles) and approximate $f(x,y)$ by simple polynomials in each region. A common choice is bilinear approximation: $f(x,y) \approx c_0 + c_1 x + c_2 y + c_3 xy$.

Curse of Dimensionality

Regardless of the method, high-dimensional integration suffers from the curse of dimensionality:

To do quadrature with grid size h in d dimensions, we need to evaluate the integrand $\mathcal{O}(h^{-d})$ times. That is, the effort grows exponentially with problem dimension.

This is ok if d is small (e.g. $d \leq 5$), but rapidly becomes a problem.

e.g. a grid size of $h = 0.1$ on $[-1, 1]^d$ with $d = 10$ requires $\approx 10^{13}$ evaluations!

Monte Carlo Integration

Another integration technique comes from probability/statistics: Monte Carlo integration. This is often used as a way of calculating expected values of random variables (e.g. mathematical finance). Recipe:

- Generate N random values x_i, uniformly distributed inside the domain of integration (e.g. uniform in $[a, b]$).
- Approximate the integral by the scaled sample mean: $I \approx Q_N(f) = \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$.

The approximation $Q_N(f)$ is random, so rerunning multiple times gives different answers.

Theorem

The expected value of $Q_N(f)$ is the true integral I. The average error is of size $\mathcal{O}(1/\sqrt{N})$.

So, using larger N is better (increasing N by $100\times$ decreases error by $10\times$, on average).

Importantly, the error is independent of the problem dimension!

Monte Carlo Integration

One drawback: random points tend to be 'clumped':

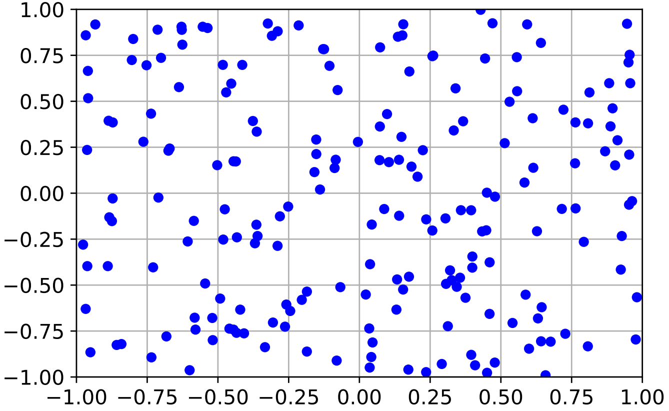


Figure 6. 200 randomly generated points on $[-1, 1]^2$.

Quasi Monte Carlo Integration

An alternative approach is Quasi Monte Carlo integration.

Instead of selecting truly random points (actually "pseudorandom" on a computer), we pick quasi-random points that are: deterministic, don't clump, and "look random".

Theorem

For quasi-Monte Carlo, the expected value of $Q_N(f)$ is the true integral I. The average error is approximately of size $\mathcal{O}(\log(N)^d/N)$.

- Advantage: error decreases much faster than $\mathcal{O}(1/\sqrt{N})$, almost dimension-independent.
- Disadvantage: hard to build good quasi random sequences.
  - Common choices include Sobol sequences (Ilya Sobol, 1967) and lattice rules.

Monte Carlo Integration

Quasi-random points avoid 'clumping' (every grid square has 3–4 points):

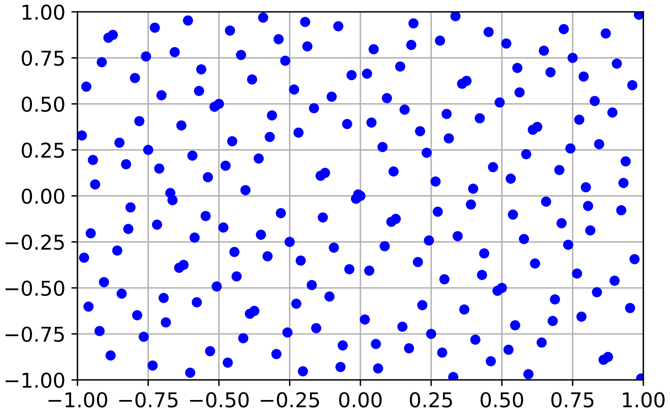


Figure 7. 200 Sobol points on $[-1, 1]^2$.