

# MA5233 Computational Mathematics

## Lecture 5: Quadrature

Simon Etter



Semester I, AY 2020/2021

# Quadrature

## Problem statement

Compute

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

assuming the only thing we can do with  $f : [a, b] \rightarrow \mathbb{R}$  is to evaluate it at a finite number of points  $x_k \in [a, b]$ .

# Quadrature

## Quadrature vs. pattern matching and Riemann integrals

The pen-and-paper technique for computing integrals is to match  $f(x)$  against a table of functions whose integrals are known and then use these known integrals to assemble the integral of  $f(x)$ .

This technique has two drawbacks:

- ▶ It assumes that we have a formula for  $f(x)$  which we can use for pattern-matching.
- ▶ Not every formula has a match in the table of known derivatives and thus this technique may fail for some  $f(x)$ .

Quadrature overcomes these drawbacks by assuming only that we can evaluate  $f(x)$  at any input argument  $x \in [a, b]$ . Quadrature is hence fairly close to the definition of integrals as the limits of Riemann sums, but quadrature differs from Riemann integrals in that quadrature does not allow to take limits.

Prohibiting limits is of course done to ensure that quadrature algorithms can be evaluated in finite time and implies that quadrature algorithms are generally only convergent but not exact.

# Quadrature

## Why is this problem called “quadrature”?

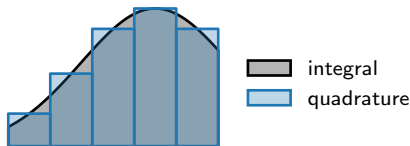
Integrals measure the area under a curve, and the easiest way to describe an area is to specify how many squares of a fixed side length we can fit into this area.

Example: The area of a football pitch is  $7140\text{m}^2$ , which means that we can fit  $7140$   $1\text{m} \times 1\text{m}$  squares onto a pitch.

Computing integrals can hence be interpreted as dividing an arbitrary two-dimensional shape into many squares, and many early quadrature algorithms followed this idea fairly literally (see picture below). We therefore call this operation “quadrature”.

Square and quadrature are indeed derived from the same Latin root even though this shared origin is not very clear in English.

In modern parlance, “numerical integration” might be a more natural term to describe the above problem, but this term is sometimes used to describe numerical ODE solvers and should therefore be used with care.



# Quadrature

## Quadrature algorithms

It turns out that all algorithms  $\tilde{Q}(f)$  approximating the integral map

$$Q(f) = \int_a^b f(x) dx$$

are of the form

$$\tilde{Q}(f) = \sum_{k=1}^n f(x_k) w_k \quad \text{for some } (x_k \in \mathbb{R}, w_k \in \mathbb{R})_{k=1}^n,$$

and most practically relevant  $\tilde{Q}(f)$  determine the parameters  $(x_k, w_k)_k$  such that  $\tilde{Q}(p)$  is exact for all polynomials  $p$  up to some maximal degree  $d \in \mathbb{N}$ , i.e. we have

$$\tilde{Q}(p) = Q(p) \quad \text{for all } p \in \mathcal{P}_d.$$

The next slide will look into why this is the case.

# Quadrature

## Quadrature algorithms (continued)

Quadrature algorithms follow the functional form

$$\tilde{Q}(f) = \sum_{k=1}^n f(x_k) w_k \quad \text{for some } (x_k \in \mathbb{R}, w_k \in \mathbb{R})_{k=1}^n, \quad (1)$$

for a good reason: the exact integral map  $Q(f)$  is a linear function, i.e. we have

$$Q(c_1 f_1 + c_2 f_2) = c_1 Q(f_1) + c_2 Q(f_2)$$

for all  $f_1, f_2 : [a, b] \rightarrow \mathbb{R}$  and all  $c_1, c_2 \in \mathbb{R}$ , so it makes sense to require that  $\tilde{Q}(f)$  is a linear function as well. Combined with the finite evaluation requirement, this yields that  $\tilde{Q}(f)$  must be of the form (1).

On the other hand, it is not necessary that the parameters  $(x_k, w_k)_k$  in (1) are chosen such that  $\tilde{Q}(f)$  is exact on polynomials. In fact, there are algorithms which pursue a different approach, e.g. Monte Carlo methods, but I will not discuss such method in this lecture. The reason for requiring polynomial exactness will become clear at a later point.

# Quadrature

## Outlook

The above discussion raises two questions.

1. How do we determine quadrature rules exact for polynomials up to a certain degree?
2. Can we predict the rate of convergence of such methods?

I will address these questions shortly, but first let me formally write down some terminology related to quadrature techniques.

## Def: Quadrature rule

A formula of the form

$$\sum_{k=1}^n f(x_k) w_k \approx \int_a^b f(x) dx$$

is called a *quadrature rule*. Moreover,

- ▶  $x_k$  are called *quadrature points* or *quadrature nodes*.
- ▶  $w_k$  are called *quadrature weights*.

# Quadrature

## Def: Degree of polynomial exactness

A quadrature rule  $(x_k, w_k)_{k=1}^n$  is said to be exact on  $\mathcal{P}_d$  and  $d$  is called its *degree of polynomial exactness* if

$$\sum_{k=1}^n p(x_k) w_k = \int_a^b p(x) dx \quad \text{for all } p \in \mathcal{P}_d.$$

$d$  is sometimes also called precision, order or degree of accuracy.

## Thm: Quadrature rules exact on $\mathcal{P}_{n-1}$

Let  $(x_k \in \mathbb{R})_{k=1}^n$  be  $n$  distinct points and denote by  $\ell_k(x)$  the associated Lagrange polynomials (cf. Lecture 4). Then, the quadrature rule

$$(x_k, w_k)_{k=1}^n \quad \text{with} \quad w_k = \int_a^b \ell_k(x) dx \quad \text{is exact on } \mathcal{P}_{n-1}.$$

*Proof.* It follows from the uniqueness of polynomial interpolants (cf. Lecture 4) and the above choice of quadrature weights that

$$\int_a^b p(x) dx = \int_a^b \sum_{k=1}^n p(x_k) \ell_k(x) dx = \sum_{k=1}^n p(x_k) w_k.$$



# Quadrature

## Thm: Quadrature error estimate

Assume the quadrature rule  $(x_k, w_k)_{k=1}^n$  is exact on  $\mathcal{P}_{n-1}$ . Then,

$$\left| \int_a^b f(x) dx - \sum_{k=1}^n f(x_k) w_k \right| \leq (b-a) \|f - p\|_{[a,b]},$$

where  $p \in \mathcal{P}_{n-1}$  denotes the polynomial interpolant to  $f(x)$  in  $(x_k)_{k=1}^n$ .

*Proof.*

$$\begin{aligned} \left| \int_a^b f(x) dx - \sum_{k=1}^n f(x_k) w_k \right| &= \dots \\ &= \left| \int_a^b f(x) dx - \sum_{k=1}^n p(x_k) w_k \right| && p(x) \text{ interpolates } f(x) \text{ in } (x_k)_k \\ &= \left| \int_a^b f(x) dx - \int_a^b p(x) dx \right| && (x_k, w_k)_k \text{ is exact for } p \in \mathcal{P}_{n-1} \\ &\leq \int_a^b |f(x) - p(x)| dx \leq (b-a) \|f - p\|_{[a,b]}. \end{aligned}$$

# Quadrature

## Discussion

The result on the last slide allows us to bound quadrature errors in terms of interpolation errors, and it explains why it is useful to require quadrature rules to be exact on  $\mathcal{P}_n$ .

In Lecture 4, we have seen that interpolation in equispaced points may fail to converge but interpolation in Chebyshev zeroes is guaranteed to converge as long as  $f(x)$  is continuous.

This motivates the following definition.

## Def: Clenshaw-Curtis quadrature rules

Quadrature rules  $(x_k, w_k)_{k=1}^n$  where  $(x_k)_{k=1}^n$  are the  $n$  zeroes of the degree- $n$  Chebyshev polynomial and  $(w_k)_{k=1}^n$  are chosen such that  $(x_k, w_k)_k$  is exact on  $\mathcal{P}_{n-1}$  (cf. the theorem on slide 8) are known as *Clenshaw-Curtis quadrature rules*.

Simply put:

Clenshaw-Curtis quadrature  $\longleftrightarrow (x_k)_k = \text{Chebyshev zeroes}$

# Quadrature

**Example: Clenshaw-Curtis applied to  $|x|^k$**

`clenshaw_curtis_abs()` applies Clenshaw-Curtis (CC) quadrature to the integrals

$$\int_{-1}^1 |x| dx \quad \text{and} \quad \int_{-1}^1 |x|^3 dx.$$

These integrands have  $k = 1$  and  $k = 3$  derivatives, respectively; hence the theory of Chebyshev interpolants predicts that CC applied to  $|x|^3$  should converge two powers of  $n$  faster than CC applied to  $|x|$ , and this is indeed what we observe.

# Quadrature

## Example: Clenshaw-Curtis applied to $|x|^k$ (continued)

However, the Chebyshev theory also predicts that the error in CC applied to  $|x|^k$  should be  $O(n^{-k})$ , but we numerically observe that the error is in fact  $O(n^{-(k+1)})$ .

This extra power of  $n$  is qualitatively explained in `chebyshev_error_abs()`: we observe that the interpolation error oscillates and is heavily concentrated around  $x = 0$ , which means that both of the bounds

$$\left| \int_a^b (f(x) - p(x)) dx \right| \leq \int_a^b |f(x) - p(x)| dx \leq (b - a) \|f - p\|_{[a,b]}$$

employed on slide 9 are quite loose.

I am not aware of any result which turns this qualitative argument into a rigorous proof.

# Quadrature

**Example: Clenshaw-Curtis applied to  $\frac{1}{1+ax^2}$**

`clenshaw_curtis_runge()` applies Clenshaw-Curtis (CC) quadrature to the integrals

$$\int_{-1}^1 \frac{1}{1+4x^2} dx \quad \text{and} \quad \int_{-1}^1 \frac{1}{1+25x^2} dx.$$

We have seen in Lecture 4 that Chebyshev interpolation converges faster when applied to  $\frac{1}{1+4x^2}$  than when applied to  $\frac{1}{1+25x^2}$ , and this phenomenon is also visible in the CC convergence plot.

However, we also observe that the convergence is slightly better than predicted by the Chebyshev interpolation theory for small  $n$ . This “kink” in the convergence plot is explained in Weidemann & Trefethen (2007), *The kink phenomenon in Fejér and Clenshaw–Curtis quadrature*, but this explanation is beyond the scope of this module.

## Conclusion

Quadrature convergence theory = Interpolation convergence theory  
+ some extra quirks specific to quadrature.

# Quadrature

## **Remark: Determining quadrature weights**

The theorem on slide 8 provides the formula

$$w_k = \int_a^b \ell_k(x) dx$$

for determining weights  $(w_k)_k$  such that  $(x_k, w_k)_{k=1}^n$  is exact on  $\mathcal{P}_{n-1}$ .

This formula is important from a theoretical perspective because it shows that such weights always exist, but it is rarely used in practice because it is difficult to evaluate.

Instead, many quadrature rules come with specialised algorithms for constructing the quadrature points and weights (have a look at `clenshaw_curtis()` for an example of such an algorithm).

I will not discuss such algorithms because you do not need to know about them to understand the theory of quadrature, and you can easily look them up in the literature or find a software package which computes them for you once the need arises.

# Quadrature

## Discussion

Clenshaw-Curtis quadrature is not used very often because it is shadowed by its competition.

- ▶ Newton-Cotes quadrature generally performs worse than Clenshaw-Curtis but is much easier to understand and implement.
- ▶ Gauss quadrature is at a similar level of conceptual complexity as Clenshaw-Curtis but performs better.

I will next discuss both of these competitors.

## Def: Newton-Cotes quadrature rules

Quadrature rules  $(x_k, w_k)_{k=1}^n$  where  $(x_k)_{k=1}^n$  are the  $n$  equispaced points

$$x_k = a + (b - a) \frac{k-1}{n-1} \quad \Longleftrightarrow \quad \begin{array}{c} a \qquad \qquad \qquad b \\ \hline | \quad | \quad | \quad | \quad | \\ x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \end{array}$$

and  $(w_k)_{k=1}^n$  are chosen such that  $(x_k, w_k)_k$  is exact on  $\mathcal{P}_{n-1}$  (cf. the theorem on slide 8) are known as *Newton-Cotes quadrature rules*.

Simply put:

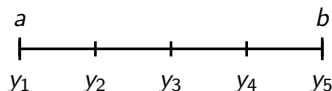
$$\text{Newton-Cotes quadrature} \quad \longleftrightarrow \quad (x_k)_k = \text{equispaced points}$$

# Quadrature

## Discussion

We have seen in Lecture 4 that interpolation in equispaced points may diverge and so Newton-Cotes rules are generally unreliable for large  $n$ . Newton-Cotes rules are therefore mainly used as building blocks for composite methods.

Recall from Lecture 4 that composite methods proceed by splitting a large interval  $[a, b]$  into many small intervals  $[y_k, y_{k+1}]$  and then applying a nested method on each small interval.



I demonstrated this idea in Lecture 4 at the example of composite interpolation, and it is clear that this idea can also be applied to quadrature. The next slides presents a few well-known examples.

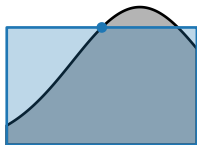


# Quadrature

## Special Newton-Cotes rules ( $m = \frac{a+b}{2}$ )

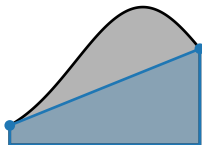
Midpoint rule

$$(b-a)f(m)$$



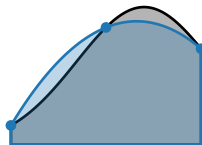
Trapezoidal rule

$$\frac{b-a}{2} (f(a) + f(b))$$

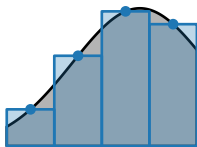


Simpson rule

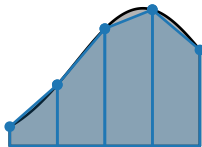
$$\frac{b-a}{6} (f(a) + 4f(m) + f(b))$$



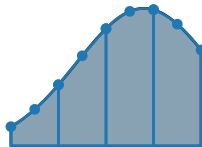
Composite midpoint



Composite trapezoidal



Composite Simpson



# Quadrature

## Implementation of composite Newton-Cotes rules

See `midpoint()` and `trapezoidal()`.

## Discussion

The convergence theory for composite Newton-Cotes methods involves the same steps as the theory for Clenshaw–Curtis rules:

1. Determine the degree of polynomial exactness.
2. Bound the quadrature error in terms of the interpolation error.
3. Deduce rates of convergence.

However, we shall see that composite quadrature rules introduce a few extra twists to this story.

For starters, the result on the next slide shows that Newton-Cotes rules with an odd number of points  $n$  have a degree of exactness which is one higher than you might expect.

# Quadrature

## Thm: Degree of exactness of Newton-Cotes rules

The  $n$ -point Newton-Cotes rule is exact on  $\begin{cases} \mathcal{P}_{n-1} & \text{if } n \text{ is even,} \\ \mathcal{P}_n & \text{if } n \text{ is odd.} \end{cases}$

*Proof.* It follows immediately from the choice of quadrature weights  $(w_k)_k$  that the  $n$ -point Newton-Cotes rule is exact on  $\mathcal{P}_{n-1}$ .

The only nontrivial part of this proof is therefore to show that Newton-Cotes rules are exact on  $\mathcal{P}_n$  if  $n$  is odd.

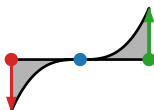
This will be tackled on the next slide.

# Quadrature

*Proof (continued).* We note that any  $p \in \mathcal{P}_n$  can be written as

$$p(x) = c(x - m)^n + q(x)$$

for some  $c \in \mathbb{R}$ ,  $q \in \mathcal{P}_{n-1}$  and  $m = \frac{b+a}{2}$ , and since  $(x - m)^n$  is odd about  $m$  and the quadrature points  $(x_k)_k$  are symmetric about  $m$ , we have

$$\int_a^b (x - m)^n dx = 0 = \sum_{k=1}^n (x_k - m)^n w_k.$$


Combining this result with the exactness of  $(x_k, w_k)_k$  on  $\mathcal{P}_{n-1}$ , we obtain

$$\begin{aligned} \int_a^b p(x) dx &= \int_a^b c(x - m)^n dx + \int_a^b q(x) dx \\ &= \sum_{k=1}^n c(x_k - m)^n w_k + \sum_{k=1}^n q(x_k) w_k = \sum_{k=1}^n w_k p(x_k). \end{aligned}$$

# Quadrature

## Discussion

The extra degree of polynomial exactness of Newton-Cotes methods requires us to sharpen the quadrature error estimate from slide 9 as follows.

### Thm: Improved quadrature error estimate

Assume the quadrature rule  $(x_k, w_k)_{k=1}^n$  is exact on  $\mathcal{P}_d$  with  $d \geq n$ . Then,

$$\left| \int_a^b f(x) dx - \sum_{k=1}^n f(x_k) w_k \right| \leq (b-a) \min_{p \in I_n \mathcal{P}_d} \|f - p\|_{[a,b]},$$

where  $I_n \mathcal{P}_d$  denotes the set of all polynomials  $p \in \mathcal{P}_d$  which interpolate  $f(x)$  in the points  $(x_k)_{k=1}^n$ .

*Proof.* Analogous to the one on slide 9 except that the larger degree of exactness  $d \geq n$  gives us some freedom for choosing  $p \in I_n \mathcal{P}_d$  which we can then minimise over.

# Quadrature

## Discussion

The final step in the convergence theory of composite Newton-Cotes methods is to translate the above error bound into a convergence estimate. I need the following terminology to do so.

## Terminology: Local quadrature rules

The quadrature rule which is applied on each small interval  $[y_k, y_{k+1}]$  is called a *nested* or *local* quadrature rule.

Example: The midpoint rule is the local quadrature rule employed in the composite midpoint method.

## Thm: Convergence of composite quadrature

Consider a composite quadrature rule which splits the global interval  $[a, b]$  into  $m$  intervals  $[y_k, y_{k+1}]$  of equal lengths and whose local quadrature rule is exact on  $\mathcal{P}_d$ .

Then, the error incurred by this quadrature rule when applied to a  $d + 1$  times differentiable function  $f(x)$  satisfies

$$\text{error} = O(m^{-(d+1)}).$$

# Quadrature

*Proof.* Let us denote the local quadrature rule applied to the interval  $[y_k, y_{k+1}]$  by  $(x_\ell^{(k)}, w_\ell^{(k)})_{\ell=1}^n$ . Since this quadrature rule is exact on  $\mathcal{P}_d$ , we have

$$\left| \int_{y_k}^{y_{k+1}} f(x) dx - \sum_{\ell=1}^n f(x_k) w_k \right| \leq (y_{k+1} - y_k) \|f - p\|_{[y_k, y_{k+1}]}$$

where  $p \in \mathcal{P}_d$  is the interpolant to  $f(x)$  in the  $n$  quadrature points  $(x_\ell)_{\ell=1}^n$  and any arbitrary  $d+1-n$  additional points  $(x_\ell)_{\ell=n+1}^{d+1}$ .

The above is an application of the improved quadrature error estimate from slide 21.

According to the generalised Taylor's theorem from Lecture 4, we have

$$\|f - p\|_{[y_k, y_{k+1}]} \leq O(|y_{k+1} - y_k|^{d+1}) = O(m^{-d-1})$$

and thus the global error is given by

$$\begin{aligned} \text{error} &= \sum_{k=1}^m \left| \int_{y_k}^{y_{k+1}} f(x) dx - \sum_{\ell=1}^n f(x_k) w_k \right| \\ &= \sum_{k=1}^m O(m^{-d-2}) = O(m^{-d-1}). \end{aligned}$$

# Quadrature

## Discussion

The following corollary demonstrates how to apply the above result to the composite Newton-Cotes methods from slide 17.

## Corollary

Assume  $f(x)$  is two times differentiable. Then, the errors incurred by the composite midpoint and trapezoidal rules are

$$\text{error} = O(m^{-2}).$$

*Proof.*

- ▶ The midpoint rule is the Newton-Cotes rule for  $n = 1$  points. This number is odd, so the midpoint rule is exact for polynomials up to degree  $d = n = 1$  and its order of convergence is  $d + 1 = 2$ .
- ▶ The trapezoidal rule is the Newton-Cotes rule for  $n = 2$  points. This number is even, so the trapezoidal rule is exact for polynomials up to degree  $d = n - 1 = 1$  and its order of convergence is  $d + 1 = 2$ .

## Numerical demonstration

See `newton_cotes_convergence()`.



# Quadrature

## Remark

The result from slide 19 regarding the extra degree of exactness of  $n$ -point Newton-Cotes rules with odd  $n$  also applies to Clenshaw–Curtis quadrature rules.

However, this extra degree does not matter for Clenshaw–Curtis rules because there it makes the difference between

$$O(n^{-k}) \text{ and } O((n+1)^{-k}) \text{ for differentiable } f(x)$$

and

$$O(r^n) \text{ and } O(r^{-(n+1)}) \text{ for analytic } f(x).$$

# Quadrature

## Discussion

Newton-Cotes rules with odd  $n$  demonstrate that it is possible for an  $n$ -point quadrature rule to achieve a degree of polynomial exactness  $d$  larger than the  $d \geq n - 1$  lower bound introduced by the theorem on slide 8, and we have just seen that this can have a significant impact on the speed of convergence of composite quadrature rules.

This raises the question whether it is possible to achieve even higher degrees of polynomial exactness and hence even faster convergence.

The answer is yes, but only up to the upper bound presented next.

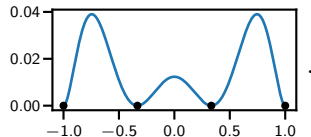
# Quadrature

## Thm: Maximal degree of exactness

No  $n$ -point quadrature rule can be exact on  $\mathcal{P}_{2n}$ .

*Proof.* Consider a generic  $n$ -point quadrature rule  $(x_k, w_k)_{k=1}^n$  and the polynomial

$$p(x) = \prod_{k=1}^n (x - x_k)^2 \in \mathcal{P}_{2n} \quad \Longleftrightarrow$$



Since  $p(x) > 0$  for all  $x \in [a, b] \setminus \{x_k \mid k \in \{1, \dots, n\}\}$  but  $p(x_k) = 0$  for all  $k \in \{1, \dots, n\}$ , we have

$$\int_a^b p(x) dx > 0 = \sum_{k=1}^n p(x_k) w_k;$$

hence  $(x_k, w_k)_{k=1}^n$  is not exact on  $\mathcal{P}_{2n}$ .

# Quadrature

## Outlook

We will next show that the upper bound  $d \leq 2n - 1$  on the degree of polynomial exactness  $d$  can indeed be achieved by an  $n$ -point quadrature rule. Doing so will require the following definitions.

## Def: $L^2$ inner product

The  $L^2$  inner product of two functions  $f, g : [a, b] \rightarrow \mathbb{R}$  is given by

$$\langle f, g \rangle_{L^2([a,b])} = \int_a^b f(x) g(x) dx.$$

## Theorem

$L^2$  inner products and norms are related by

$$\|f\|_{L^2([a,b])} = \sqrt{\langle f, f \rangle_{L^2([a,b])}}.$$

# Quadrature

## Thm: Existence of orthogonal polynomials

There exists a sequence of polynomials  $(L_n \in \mathcal{P}_n)_{n=0}^{\infty}$  which are pairwise orthogonal in the  $L^2([-1, 1])$  norm, i.e.

$$\langle L_m, L_n \rangle_{L^2([-1, 1])} = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{otherwise.} \end{cases}$$

*Proof.* Apply the Gram-Schmidt orthogonalisation procedure

$$\tilde{L}_n = p_n - \sum_{k=0}^{n-1} \langle L_k, p_n \rangle_{L^2([-1, 1])} L_k, \quad L_n = \frac{\tilde{L}_n}{\|\tilde{L}_n\|_{[-1, 1]}}$$

to the sequence  $p_n(x) = x^n$ . This process will not break down (i.e.  $\tilde{L}_n \neq 0$  for all  $n$ ) because the monomials  $x^n$  are linearly independent.

## Def: Legendre polynomials

The above polynomials  $L_n$  are called *Legendre polynomials*.

# Quadrature

## Example

The first three Legendre polynomials are

$$L_0(x) = \frac{1}{\sqrt{2}}, \quad L_1(x) = \sqrt{\frac{3}{2}} x, \quad L_2(x) = \sqrt{\frac{5}{8}} (3x^2 - 1).$$

*Proof.* I verify only some of the inner products  $\langle L_m, L_n \rangle_{L^2([-1,1])}$  since the computations are straightforward but tedious.

$$\langle L_0, L_0 \rangle_{L^2([-1,1])} = \int_{-1}^1 \frac{1}{2} dx = 1$$

$$\langle L_0, L_1 \rangle_{L^2([-1,1])} = \int_{-1}^1 \frac{\sqrt{3}}{2} x dx = 0$$

$$\langle L_0, L_2 \rangle_{L^2([-1,1])} = \int_{-1}^1 \frac{\sqrt{5}}{4} (3x^2 - 1) dx = \frac{\sqrt{5}}{4} (2 - 2) = 0$$

$$\langle L_1, L_2 \rangle_{L^2([-1,1])} = \int_{-1}^1 \frac{\sqrt{15}}{4} x (3x^2 - 1) dx = 0 \quad (\text{integrand is odd})$$

# Quadrature

The following result is an immediate consequence of the definition of Legendre polynomials, but it is nevertheless useful to write it down explicitly.

## Theorem

The  $n$ th Legendre polynomial is  $L^2([-1, 1])$ -orthogonal to  $\mathcal{P}_{n-1}$ , i.e.

$$\langle p, L_n \rangle_{L^2([-1,1])} = \int_{-1}^1 p(x) L_n(x) dx = 0 \quad \text{for all } p \in \mathcal{P}_{n-1}.$$

*Proof.* Undergraduate material.

# Quadrature

## Remark 1

You must have heard of Gram-Schmidt orthogonalisation in your undergraduate linear algebra course, but do not worry too much in case you forgot some of the details.

I will discuss Gram-Schmidt orthogonalisation (and a better algorithm for computing Legendre polynomials) in the lecture on Krylov methods.

## Remark 2

We have now seen three families of polynomials with fairly non-descriptive names. Let me briefly summarise them for your convenience.

- ▶ Lagrange polynomials  $\ell_k \in \mathcal{P}_n$ :  $\ell_k(x_\ell) = \delta_{k,\ell}$ .
- ▶ Chebyshev polynomials  $T_n \in \mathcal{P}_n$ : equioscillating on  $[-1, 1]$ .
- ▶ Legendre polynomials  $L_n \in \mathcal{P}_n$ : orthogonal in  $L^2([-1, 1])$ .



# Quadrature

## Def: Gauss quadrature rules

Quadrature rules  $(x_k, w_k)_{k=1}^n$  where  $(x_k)_{k=1}^n$  are the  $n$  zeroes of the degree- $n$  Legendre polynomial and  $(w_k)_{k=1}^n$  are chosen such that  $(x_k, w_k)_k$  is exact on  $\mathcal{P}_{n-1}$  (cf. the theorem on slide 8) are known as *Gauss quadrature rules*.

Simply put:

$$\text{Gauss quadrature} \longleftrightarrow (x_k)_k = \text{Legendre zeroes}$$

## Thm: Degree of polynomial exactness of Gauss quadrature

The  $n$ -point Gauss quadrature rule achieves the largest possible degree of polynomial exactness  $2n - 1$ .

*Proof.* See next slide.

# Quadrature

*Proof.* According to fundamental results from algebra, any  $p \in \mathcal{P}_{2n-1}$  can be written as

$$p(x) = q_1(x) L_n(x) + q_2(x) \quad \text{for some } q_1, q_2 \in \mathcal{P}_{n-1}.$$

See e.g. [https://en.wikipedia.org/wiki/Polynomial\\_long\\_division](https://en.wikipedia.org/wiki/Polynomial_long_division)

According to the result from slide 31 and since the quadrature points  $(x_k)_{k=1}^n$  are chosen such that  $(L_n(x_k) = 0)_{k=1}^{n-1}$ , we have

$$\int_{-1}^1 q_1(x) L_n(x) dx = 0 = \sum_{k=1}^n q_1(x_k) L_n(x_k) w_k.$$

Combining this result with the exactness of  $(x_k, w_k)_{k=1}^n$  on  $\mathcal{P}_{n-1}$ , we obtain

$$\begin{aligned} \int_{-1}^1 p(x) dx &= \int_{-1}^1 q_1(x) L_n(x) dx + \int_{-1}^1 q_2(x) dx \\ &= \sum_{k=1}^n q_1(x_k) L_n(x_k) w_k + \sum_{k=1}^n q_2(x_k) w_k = \sum_{k=1}^n p(x_k) w_k. \end{aligned}$$

# Quadrature

## Corollary 1: Convergence of Gauss quadrature

The error incurred by the  $n$ -point Gauss rule applied to a function  $f(x)$  is

- ▶  $O(n^{-k})$  if  $f(x)$  has  $k$  derivatives, and
- ▶  $O(r^{-2n})$  if  $f(x)$  is analytic on the Bernstein ellipse with radius  $r > 1$ .

*Proof sketch.* The improved quadrature error estimate from slide 21 yields the bound

$$\left| \int_a^b f(x) dx - \sum_{k=1}^n f(x_k) w_k \right| \leq (b-a) \min_{p \in I_n \mathcal{P}_{2n-1}} \|f - p\|_{[a,b]},$$

where  $I_n \mathcal{P}_{2n-1}$  denotes the set of all polynomials  $p \in \mathcal{P}_{2n-1}$  which interpolate  $f(x)$  in the points  $(x_k)_{k=1}^n$ .

We would then have to show that

$$\min_{p \in I_n \mathcal{P}_{2n-1}} \|f - p\|_{[a,b]} \lesssim \|f - p_{\text{Cheb}}\|_{[a,b]}$$

where  $p_{\text{Cheb}}(x)$  denotes the Chebyshev interpolant to  $f(x)$  in  $2n$  points. This can be done but requires tools beyond the scope of this module.

# Quadrature

## Numerical demonstration

- ▶ For differentiable functions, see `gauss_abs()`.

We again observe that the convergence in the case of differentiable functions is one power of  $n$  better than predicted by theory.

- ▶ For analytic functions, see `gauss_runge()`.

The theoretical and empirical rates of convergence match perfectly.

`clenshaw_curtis_vs_gauss()` further reveals that Gauss quadrature indeed performs better than Clenshaw–Curtis, but only for large enough  $n$  and not by a factor of 2.

# Quadrature

## **Corollary 2: Composite Gauss quad. for differentiable functions**

Consider a composite quadrature rule which applies the  $n$ -point Gauss rule over  $m$  equispaced intervals to a function  $f(x)$  with  $2n$  derivatives. The error incurred by this quadrature rule is

$$\text{error} = O(m^{-2n}).$$

*Proof.* The claim follows immediately from the degree of polynomial exactness  $d = 2n - 1$  of Gauss rules shown on slide 33 and the convergence theorem for composite quadrature rules from slide 22.

## **Numerical demonstration**

See Assignment 2.

# Quadrature

## Summary

- ▶ Clenshaw-Curtis quadrature: Interpolate in  $n$  Chebyshev zeroes.

$$\text{error} \leq (b - a) \|f - p\|_{[a,b]}$$

where  $p \in \mathcal{P}_{n-1}$  denotes the Chebyshev interpolant to  $f(x)$ .

Rarely used in practice.

- ▶ Newton-Cotes quadrature: Interpolate in  $n$  equispaced points. Almost exclusively used in composite methods.

$$\text{error} = O(m^{-n-1+\text{mod}(n,2)}) \quad \text{where} \quad m = \# \text{ intervals.}$$

Famous examples: midpoint, trapezoidal and Simpson rules.

- ▶ Gauss quadrature: Interpolate in  $n$  Legendre zeroes to achieve exactness on  $\mathcal{P}_{2n-1}$ , which is best possible.

$$\text{error} \lesssim (b - a) \|f - p\|_{[a,b]},$$

where  $p \in \mathcal{P}_{2n-1}$  denotes the Chebyshev interpolant to  $f(x)$ .