Numerical Integration

Anne Kværnø, André Massing

May 1, 2020

The Python codes for this note are given in `quadrature.py`.

1 Introduction

Imagine you want to compute the finite integral

$$I[f](a,b) = \int_a^b f(x)\,\mathrm{d}x.$$

The "usual" way is to find a primitive function F (also known as the indefinite integral of f) satisfying $F'(x) = f(x)$ and then to compute

$$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a).$$

While there are many analytical integration techniques and extensive tables to determine definite integral for many integrands, more often than not it may not feasible or possible to compute the integral. For instance, what about

$$f(x) = \frac{\log(2+\sin(1/2-\sqrt(x))^6)}{\log(\pi+\arctan(\sqrt{1-\exp(-2x-\sin(x))}))}?$$

Finding the corresponding primitive is highly likely a hopeless endeavor. And sometimes there even innocent looking functions like e^{-x^2} for which there is not primitive functions which can expressed as a composition of standard functions such as $\sin, \cos$. etc.

A **numerical quadrature** or a **quadrature rule** is a formula for approximating such definite integrals $I[f](a,b)$. Quadrature rules are usually of the form

$$Q[f](a,b) = \sum_{i=0}^{n} w_i f(x_i),$$

where x_i, w_i for $i = 0, 1, \ldots, n$ are respectively the *nodes/points* and the *weights* of the quadrature rule. To emphasize that a quadrature rule is defined by some given quadrature points $\{x_i\}_{i=0}^n$ and weights $\{w_i\}_{i=0}^n$, we sometimes might write

$$Q[f](\{x_i\}_{i=0}^n, \{w_i\}_{i=0}^n) = \sum_{i=0}^{n} w_i f(x_i).$$

If the function f is given from the context, we will for simplicity denote the integral and the quadrature simply as $I(a,b)$ and $Q(a,b)$.

Example 1.1. *Quadrature rules from previous math courses.*

The trapezoidal rule, the midpoint rule and Simpson's rule known from previous courses are all examples of numerical quadratures, and we quickly review them here:

- **Midpoint rule** is the simplest possible quadrature rule defined by

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

The node is given by the midpoint, $x_0 = \frac{a+b}{2}$ with the corresponding weight $w_0 = b - a$.

- **Trapezoidal rule** is given by

$$Q[f](a,b) := (b-a)\left(\frac{f(a)+f(b)}{2}\right)$$

and thus the nodes are defined by $x_0 = a$, $x_1 = b$ with corresponding weights $w_0 = w_1 = \frac{1}{2}$.

- Finally, **Simpson's rule** which you know from M1, is defined as follows:

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

which we identify as quadrature rule with 3 points $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$ and corresponding weights $w_0 = w_2 = \frac{b-a}{6}$ and $w_1 = \frac{4(b-a)}{6}$.

In this note we will see how quadrature rules can be constructed from integration of interpolation polynomials. We will demonstrate how to do error analysis and how to find error estimates. In the sequel, we will use material from *Preliminaries*, section 3.2, 4 and 5.

2 Quadrature based on polynomial interpolation.

This section relies on the content of the note on polynomial interpolation, in particular the section on Lagrange polynomials.

Choose $n+1$ distinct nodes x_i, $i = 0, \ldots, n$ in the interval $[a,b]$, and let $p_n(x)$ be the interpolation polynomial satisfying the interpolation condition

$$p_n(x_i) = f(x_i), \qquad i = 0, 1, \ldots n.$$

We will then use $\int_a^b p_n(x)\,\mathrm{d}x$ as an approximation to $\int_a^b f(x)\,\mathrm{d}x$. By using the Lagrange form of the polynomial

$$p_n(x) = \sum_{i=0}^{n} f(x_i)\ell_i(x)$$

with the cardinal functions $\ell_i(x)$ given by

$$\ell_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j},$$

the following quadrature formula is obtained

$$I[f](a,b) \approx Q[f](a,b) = \int_a^b p_n(x)\,\mathrm{d}x = \sum_{i=0}^{n} f(x_i)\int_a^b \ell_i(x)\,\mathrm{d}x = \sum_{i=0}^{n} w_i f(x_i) = Q(a,b),$$

where the weights in the quadrature are simply the integral of the cardinal functions over the interval

$$w_i = \int_a^b \ell_i(x)\,\mathrm{d}x \quad \text{for } i = 0, \ldots, n.$$

Let us derive three schemes for integration over the interval $[0,1]$, which we will finally apply to the integral

$$I(0,1) = \int_0^1 \cos\left(\frac{\pi}{2}x\right) = \frac{2}{\pi} = 0.636619\ldots.$$

Example 2.1. *The trapezoidal rule revisited.*

Let $x_0 = 0$ and $x_1 = 1$. The cardinal functions and thus the weights are given by

$$\ell_0(x) = 1 - x, \qquad w_0 = \int_0^1 (1-x)\,\mathrm{d}x = 1/2$$
$$\ell_1(x) = x, \qquad w_1 = \int_0^1 x\,\mathrm{d}x = 1/2$$

and the corresponding quadrature rule is the trapezoidal rule (usually denoted by T) recalled in Example 1.1 with $[a, b] = [0, 1]$:

$$T(0,1) = \frac{1}{2}\left[f(0) + f(1)\right].$$

Example 2.2. *Gauß-Legendre quadrature for $n = 2$.*

Let $x_0 = 1/2 + \sqrt{3}/6$ and $x_1 = 1/2 - \sqrt{3}/6$. Then

$$\ell_0(x) = -\sqrt{3}x + \frac{1+\sqrt{3}}{2}, \qquad w_0 = \int_0^1 \ell_0(x)\,\mathrm{d}x = 1/2,$$
$$\ell_1(x) = \sqrt{3}x + \frac{1-\sqrt{3}}{2}, \qquad w_1 = \int_0^1 \ell_1(x)\,\mathrm{d}x = 1/2.$$

The quadrature rule is

$$Q(0,1) = \frac{1}{2}\left[f\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right) + f\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)\right].$$

Example 2.3. *Simpson's rule revisited.*

We construct Simpson's rule on the interval $[0, 1]$ by choosing the nodes $x_0 = 0$, $x_1 = 0.5$ and $x_2 = 1$. The corresponding cardinal functions are

$$\ell_0 = 2(x - 0.5)(x - 1) \qquad \ell_1(x) = 4x(1 - x) \qquad \ell_2(x) = 2x(x - 0.5)$$

which gives the weights

$$w_0 = \int_0^1 \ell_0(x)\,\mathrm{d}x = \frac{1}{6}, \qquad w_1 = \int_0^1 \ell_1(x)\,\mathrm{d}x = \frac{4}{6}, \qquad w_2 = \int_0^1 \ell_2(x)\,\mathrm{d}x = \frac{1}{6}$$

such that

$$\int_0^1 f(x)\,\mathrm{d}x \approx \int_0^1 p_2(x)\,\mathrm{d}x = \sum_{i=0}^{2} w_i f(x_i) = \frac{1}{6}\left[\, f(0) + 4f(0.5) + f(1)\,\right].$$

Exercise 1: Accuracy of some quadrature rules

Use the `QR` function

```
def QR(f, xq, wq):
    """ Computes an approximation of the integral f
    for a given quadrature rule.

    Input:
        f:  integrand
        xq: quadrature nodes
        wq: quadrature weights
    """
    n = len(xq)
    if (n != len(wq)):
        raise RuntimeError("Error: Need same number of quadrature nodes and weights!")
    return np.array(wq)@f(np.array(xq))
```

to compute an approximate value of integral

$$I(0,1) = \int_0^1 \cos\left(\frac{\pi}{2}x\right) = \frac{2}{\pi} = 0.636619\ldots .$$

using the quadrature rules from Example 2.1-2.3. Tabulate the corresponding quadrature errors $I(0,1) - Q(0,1)$.

Solution. The trapezoidal rule applied to the function $f(x) = \cos(\pi x/2)$ gives

$$T(0,1) = \frac{1}{2}\left[\cos(0) + \cos\left(\frac{\pi}{2}\right)\right] = \frac{1}{2},$$

and the error is

$$I(0,1) - T(0,1) = \frac{2}{\pi} - \frac{1}{2} = 0.138\ldots$$

The Gauß-Legendre quadrature with two points applied to $f(x) = \cos(\pi x/2)$ is given by

$$Q(0,1) = \frac{1}{2}\left[\cos\left(\frac{\pi}{2}x_0\right) + \cos\left(\frac{\pi}{2}x_1\right)\right] = 0.635647\ldots$$

with an error

$$I(0,1) - Q(0,1) = 9.72\ldots \cdot 10^{-4}.$$

Simpson's rule applied to $f(x) = \cos(\pi x/2)$ gives

$$Q(0,1) = 0.6380712\ldots$$

with an error

$$I(0,1) - Q(0,1) = -1.45\ldots \cdot 10^{-3}.$$

Remarks. We observe that with the same number of quadrature points, the Gauß-Legendre quadrature gives a much more accurate answer then the trapezoidal rule. So the choice of nodes clearly matters. Simpon's rule gives very similar results to Gauß-Legendre quadrature, but it uses 3 instead of 2 quadrature nodes. The quadrature rules which based on polynomial interpolation and *equidistributed quadrature nodes* go under the name **Newton Cotes formulas**.

We observe that with the same number of quadrature points, the Gauß-Legendre quadrature gives a much more accurate answer then the trapezoidal rule. So the choice of nodes clearly matters. Simpon's rule gives very similar results to Gauß-Legendre quadrature, but it uses 3 instead of 2 quadrature nodes.

3 Degree of exactness and an estimate of the quadrature error

Motivated by the previous examples, we know take a closer look at how assess the quality of a method. We start with the following definition.

Definition 3.1. *The degree of exactness.*

A numerical quadrature has degree of exactness d if $Q[p](a,b) = I[p](a,b)$ for all $p \in \mathbb{P}_d$ and there is at least one $p \in \mathbb{P}_{d+1}$ such that $Q[p](a,b) \neq I[p](a,b)$.

Since both integrals and quadratures are linear in the integrand f, the degree of exactness is d if

$$\begin{aligned} I[x^j](a,b) &= Q[x^j](a,b), \qquad j = 0,1,\ldots,d, \\ I[x^{d+1}](a,b) &\neq Q[x^{d+1}](a,b). \end{aligned}$$

Observation: All quadratures constructed from Lagrange interpolation polynomials in $n+1$ distinct nodes will automatically have a degree of exactness of least n. This follows immediately from the fact the interpolation polynomial $p_n \in \mathbb{P}_n$ of any polynomial $q \in \mathbb{P}_n$ is just the original polynomial q itself.

Exercise 2: Degree of exactness for some quadrature rules

- Show that the trapezoidal and midpoint rule from Example 1.1 is of precision 1
- Show that the Gauß-Legendre quadrature for 2 points from Example 2.2 is of precision 3.
- Show that Simpson's rule from Example 2.3 is also of precision 3.

Hint. You can do this either using pen and paper (boring!) or numerically (more fun!), using the code from Example 1

3.1 Error estimates

Theorem 3.1. *Error estimate for quadrature rule with degree of exactness n.*

Assume that $f \in C^{n+1}(a,b)$ and let $Q[\cdot](\{x_i\}_{i=0}^n, \{w_i\}_{i=0}^n)$ be a quadrature rule which has degree of exactness n. Then the quadrature error $|I[f] - Q[f]|$ can be estimated by

$$|I[f] - Q[f]| \leqslant \frac{M}{(n+1)!} \int_a^b \prod_{i=0}^n |x - x_i| \, \mathrm{d}x$$

where $M = \max_{\xi \in [a,b]} |f^{n+1}(\xi)|$.

Proof. Let $p_n \in \mathbb{P}_n$ be the interpolation polynomial satisfying $p_n(x_i) = f(x_i)$ for $i = 0, \ldots, n$. Thanks to the error estimate for the interpolation error, we know that

$$f(x) - p_n(x) = \frac{f^{n+1}(\xi(x))}{(n+1)!} \prod_{k=0}^n (x - x_k).$$

for some $\xi(x) \in (a,b)$. Since $Q(a,b)$ has degree of exactness n we have $I[p_n] = Q[p_n] = Q[f]$ and thus

$$\begin{aligned} |I[f] - Q[f]| = |I[f] - I[p_n]| &\leqslant \int_a^b |f(x) - p_n(x)| \, \mathrm{d}x \\ &= \int_a^b \left| \frac{f^{n+1}(\xi(x))}{(n+1)!} \prod_{k=0}^n (x - x_k) \right| \mathrm{d}x \leqslant \frac{M}{(n+1)!} \int_a^b \prod_{k=0}^n |(x - x_k)| \, \mathrm{d}x, \end{aligned}$$

which concludes the proof.

The previous theorem often The advantage of the previous theorem is that it is easy to prove. On downside is that the provided estimate can be rather crude, and often sharper estimates can be established. We give two examples here of some sharper estimates (but without proof).

Theorem 3.2. *Error estimate for the trapezoidal rule.*

For the trapezoidal rule, there is a $\xi \in (a,b)$ such that

$$I[f] - Q[f] = \frac{(b-a)^3}{12} f''(\xi).$$

Theorem 3.3. *Error estimate for Simpson's rule.*

For Simpson's rule, there is a $\xi \in (a,b)$ such that

$$I[f] - Q[f] = -\frac{(b-a)^5}{2880} f^4(\xi).$$

4 General construction of quadrature rules

In the following, you will learn the steps on how to construct realistic algorithms for numerical integration, similar to those used in software like Matlab of SciPy. The steps are:

Construction.

1. Choose $n+1$ distinct nodes on a standard interval I, often chosen to be $I = [-1, 1]$.
2. Let $p_n(x)$ be the polynomial interpolating some general function f in the nodes, and let the $Q[f](-1, 1) = I[p_n](-1, 1)$.
3. Transfer the formula Q from $[-1, 1]$ to some interval $[a, b]$.
4. Design a composite formula, by dividing the interval $[a, b]$ into subintervals and applying the quadrature formula on each subinterval.
5. Find an expression for the error $E[f](a, b) = I[f](a, b) - Q[f](a, b)$.
6. Find an expression for an estimate of the error, and use this to create an adaptive algorithm.

In this course, we will not have the time to cover the last two steps.

5 Constructing quadrature rules on a single interval

We have already seen in the previous Lecture how quadrature rules on a given interval $[a, b]$ can be constructed using polynomial interpolation.

For $n+1$ quadrature points $\{x_i\}_{i=0}^n \subset [a, b]$, we compute weights by

$$w_i = \int_a^b \ell_i(x)\,\mathrm{d}x \quad \text{for } i = 0, \ldots, n.$$

where $\ell_i(x)$ are the cardinal functions associated with $\{x_i\}_{i=0}^n$ satisfying $\ell_i(x_j) = \delta_{ij}$ for $i, j = 0, 1, \ldots, n$. The resulting quadrature rule has (at least) degree of exactness equal to n.

But how to you proceed if you know want to compute an integral on a different interval, say $[c, d]$? Do we have to reconstruct all the cardinal functions and recompute the weights?

The answer is NO! One can easily transfer quadrature points and weights from one interval to another. One typically choose the simple **reference interval** $\widehat{I} = [-1, 1]$. Then you determine some $n+1$ quadrature points $\{\widehat{x}_i\}_{i=0}^n \subset [-1, 1]$ and quadrature weights $\{\widehat{w}_i\}_{i=0}^n$ to define a quadrature rule $Q(\widehat{I})$

The quadrature points can then be transferred to an arbitrary interval $[a, b]$ to define a quadrature rule $Q(a, b)$ using the transformation

$$x = \frac{b-a}{2}\widehat{x} + \frac{b+a}{2}, \quad \text{so} \quad \mathrm{d}x = \frac{b-a}{2}\,\mathrm{d}\widehat{x},$$

and thus we define $\{x_i\}_{i=0}^n$ and $\{w_i\}_{i=0}^n$ by

$$x_i = \frac{b-a}{2}\widehat{x}_i + \frac{b+a}{2}, \quad w_i = \frac{b-a}{2}\widehat{w}_i \quad \text{for } i = 0, \ldots n.$$

Example 5.1. *Simpson's rule.*

- Choose standard interval $[-1, 1]$. For Simpson's rule, choose the nodes $t_0 = -1$, $t_1 = 0$ and $t_2 = 1$. The corresponding cardinal functions are

$$\ell_0 = \frac{1}{2}(t^2 - t), \qquad \ell_1(t) = 1 - t^2, \qquad \ell_2(t) = \frac{1}{2}(t^2 + t).$$

which gives the weights

$$w_0 = \int_{-1}^{1} \ell_0(t)dt = \frac{1}{3}, \qquad w_1 = \int_{-1}^{1} \ell_1(t)dt = \frac{4}{3}, \qquad w_2 = \int_{-1}^{1} \ell_2(t)dt = \frac{1}{3}$$

such that

$$\int_{-1}^{1} f(t)dt \approx \int_{-1}^{1} p_2(t)dt = \sum_{i=0}^{2} w_i f(t_i) = \frac{1}{3}\left[\, f(-1) + 4f(0) + f(1) \,\right].$$

- After transferring the nodes and weights, Simpson's rule over the interval $[a, b]$ becomes

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

6 Composite quadrature rules

To generate more accurate quadrature rule $Q(a, b)$ we have in principle two possibilities:

- Increase the order of the interpolation polynomial used to construct the quadrature rule.
- Subdivide the interval $[a, b]$ into smaller subintervals and apply a quadrature rule on each of the subintervals, leading to **Composite Quadrature Rules** which we will consider next.

Select $m \geqslant 2$ and divide $[a, b]$ into m equally spaced subintervals $[x_{i-1}, x_i]$ defined by $x_i = a + ih$ with $h = (b - a)/m$ for $i = 1, \ldots, m$. Then for a given quadrature rule $\mathrm{Q}[\cdot](x_{i-1}, x_i)$ the corresponding composite quadrature rule $\mathrm{CQ}[\cdot]([x_{i-1}, x_i]_{i=1}^m)$ is given by

$$\int_a^b f \,\mathrm{d}x \approx \mathrm{CQ}[f]([x_{i-1}, x_i]_{i=1}^m) = \sum_{i=1}^{m} \mathrm{Q}[f](x_{i-1}, x_i). \tag{1}$$

6.1 Composite trapezoidal rule

Using the trapezoidal rule $\mathrm{T}[f](x_{i-1}, x_i) = \frac{h}{2} f(x_{i-1}) + \frac{h}{2} f(x_i)$ the resulting composite trapezoidal rule is given by

$$\int_a^b f \,\mathrm{d}x \approx \mathrm{CT}[f]([x_i, x_{i+1}]_{i=1}^m) = h\left[\tfrac{1}{2} f(x_0) + f(x_1) + \ldots + f(x_{m-1}) + \tfrac{1}{2} f(x_m)\right]$$

Exercise 3: Testing the accuracy of the composite trapezoidal rule

Have a look at the `CT` function

```
def CT(f, a, b, m):
    """ Computes an approximation of the integral f
    using the composite trapezoidal rule.
    Input:
        f:  integrand
        a:  left interval endpoint
        b:  right interval endpoint
        m:  number of subintervals
    """
    x = np.linspace(a,b,m+1)
    h = float(b - a)/m
    fx = f(x[1:-1])
    ct = h*(np.sum(fx) + 0.5*(f(x[0]) + f(x[-1])))
    return ct
```

implementing the composite trapezoidal rule.

Use this function to compute an approximate value of integral

$$I(0,1) = \int_0^1 \cos\left(\frac{\pi}{2}x\right) = \frac{2}{\pi} = 0.636619\ldots .$$

for $m = 4, 8, 16, 32, 64$ corresponding to $h = 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}$. Tabulate the corresponding quadrature errors $I(0,1) - Q(0,1)$. What do you observe?

See `exercise_ctr()` function in `quadrature.py`.

Remarks. We observe that for each *doubling* of the number of subintervals we decrease the error by a *fourth*. That means that if we look at the quadrature error $I[f] - \mathrm{CT}[f]$ as a function of the number of subintervals m (or equivalently as a function of h), then $|I[f] - \mathrm{CT}[f]| \approx \frac{C}{m^2} = Ch^2$.

6.2 Error estimate for the composite trapezoidal rule

We will now theoretically explain the experimentally observed convergence rate in the previous Exercise 3.

First we have to recall the error estimate for for the trapezoidal rule on a single interval $[a, b]$. If $f \in C^2(a, b)$, then there is a $\xi \in (a, b)$ such that

$$I[f] - \mathrm{T}[f] = \frac{(b-a)^3}{12} f''(\xi).$$

Theorem 6.1. *Quadrature error estimate for composite trapezoidal rule.*

Let $f \in C^2(a, b)$, then the quadrature error $I[f] - \mathrm{CT}[f]$ for the composite trapezoidal rule can be estimated by

$$|I[f] - \mathrm{CT}[f]| \leqslant \frac{M_2}{12}\frac{(b-a)^3}{m^2} = \frac{M_2}{12}h^2(b-a) \tag{2}$$

where $M_2 = \max_{\xi\in[a,b]} |f''(\xi)|$.

Proof.

$$\begin{aligned}
|I[f] - \mathrm{CT}[f]| &= \left|\sum_{i=1}^{m}\Big[\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x - \big(\tfrac{h}{2}f(x_{i-1}) + \tfrac{h}{2}f(x_i)\big)\Big]\right| \\
&\leqslant \sum_{i=1}^{m}\frac{h^3}{12}|f''(\xi_i)| \leqslant M_2\sum_{i=1}^{m}\frac{(h)^3}{12} \\
&= M_2\frac{h^3}{12}\underbrace{\frac{m}{(b-a)}}_{\frac{(b-a)}{h}} = \frac{M_2}{12}h^2(b-a) = \frac{M_2}{12}\frac{(b-a)^3}{m^2}
\end{aligned}$$

.

Exercise 4: Composite Simpson's rule

We can now play the exact same game to construct and analyze a **Composite Simpson's rule**. In this set of exercises you are ask to develop the code and theory for the composite Simpson's rule. *This will be part of the next homework assignment.*

a) Start from Simpson's rule

$$\mathrm{S}[f](x_{i-1}, x_i) = \frac{h}{6}\left(f(x_{x_{i-1}}) + 4f(x_{i-1/2}) + f(x_i)\right),$$

where $x_{i-1/2} = \frac{x_{i-1}+x_i}{2}$ is the midpoint of the interval $[x_{i-1}, x_i]$.

Show that the resulting composite Simpson's rule is given by

$$\begin{aligned}
\int_a^b f\,\mathrm{d}x \approx \mathrm{CSR}[f]([x_i, x_{i+1}]_{i=1}^m) = \tfrac{h}{6}[f(x_0) + 4f(x_{x_{1/2}}) + 2f(x_1) + 4f(x_{3/2}) + 2f(x_2) + \ldots \\
+ 2f(x_{m-1}) + 4f(x_{x_{m-1/2}}) + f(x_m)].
\end{aligned}$$

b) Implement the composite Simpson's rule. Use this function to compute an approximate value of integral

$$I(0,1) = \int_0^1 \cos\left(\frac{\pi}{2}x\right) = \frac{2}{\pi} = 0.636619\ldots .$$

for $m = 4, 8, 16, 32, 64$ corresponding to $h = 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}$. Tabulate the corresponding quadrature errors $I(0,1) - Q(0,1)$. What do you observe? How does it compare to the composite trapezoidal rule?

c) Recall the error estimate for Simpson's rule on a *single interval*, stating that there is a $\xi \in (a,b)$ such that

$$I[f] - Q[f] = -\frac{(b-a)^5}{2880} f^4(\xi).$$

whenever $f \in C^4(a,b)$. Start from this estimate and prove the following theorem.

Theorem 6.2. *Quadrature error estimate for composite Simpon's rule.*

Let $f \in C^4(a,b)$, then the quadrature error $I[f] - \mathrm{CT}[f]$ for the composite trapezoidal rule can be estimated by

$$|I[f] - \mathrm{CSR}[f]| \leqslant \frac{M_4}{2880}\frac{(b-a)^5}{m^4} = \frac{M_4}{2880} h^4 (b-a) \tag{3}$$

where $M_4 = \max_{\xi\in[a,b]} |f^{(4)}(\xi)|$.

Proof. *Insert your proof here.*

7 Newton-Cotes formulas

We have already seen that *given* $n+1$ distinct but otherwise arbitrary quadrature nodes $\{x_i\}_{i=0}^n \subset [a,b]$, we can construct a quadrature rule $\mathrm{Q}[\cdot](\{x_i\}_{i=0}^{n+1}, \{w_i\}_{i=0}^{n+1})$ based on polynomial interpolation which has degree of exactness equals to n.

An classical example was the trapezoidal rule, which are based on the two quadrature points $x_0 = a$ and $x_1 = b$ and which has degree of exactness equal to 1.

The trapezoidal is the simplest example of a quadrature formula which belongs to the so-called **Newton Cotes formulas**.

By definition, **Newton-Cotes formulas** are quadrature rules which are based on **equidistributed nodes** $\{x_i\}_{i=0}^n \subset [a,b]$ and have degree of exactness equals to n.

But Newton-Cotes formula for $n \geqslant 8$ contains negative weights which is highly undesired property in a quadrature rule.

The reason is that for a positive function $f(x) \geqslant 0$ we have that the integral $I[f](a,b) \geqslant 0$ But for a quadrature rule with negative weights we have not necessarily that $Q[f](a,b) \geqslant 0$!

8 Gauß quadrature

Last lecture, when comparing the trapezoidal rule with Gauß-Legendre quadrature rule, both based on two quadrature nodes, we observed that

- the Gauß-Legendre quadrature was much more accurate than the trapezoidal rule,
- the Gauß-Legendre quadrature *has degree of exactness equal to* 3 *and not only* 1.

So obviously the position of the nodes matters!

Questions:

- Is there a general approach to construct quadrature rules $Q[\cdot](\{x_i\}_{i=0}^n, \{w_i\}_{i=0}^n)$ based on $n+1$ nodes with a degree of exactness $> n$?
- What is the maximal degree of exactness we can achieve?

Intuition: If we don't predefine the quadrature nodes, we have $2n+2$ parameters ($n+1$ nodes and $n+1$ weights) in total.

With $2n+2$ parameters, we might hope that we can construct quadrature rules which are exact for $p \in \mathbb{P}_{2n+1}$.

Definition 8.1. *Gaussian quadrature.*

A quadrature rule $Q[\cdot](\{x_i\}_{i=0}^n, \{w_i\}_{i=0}^n)$ based on $n+1$ nodes which has degree of exactness equals to $2n+1$ is called a **Gaussian (Legendre) quadrature** (GQ).

8.1 Orthogonal polynomials

To construct Gaussian quadrature rule, we need to briefly review the concept of orthogonality, which we introduced when we learned about Fourier series.

Two functions $f, g : [a,b] \to \mathbb{R}$ are orthogonal if

$$\langle f, g \rangle := \int_a^b f(x)g(x)\,\mathrm{d}x = 0.$$

Usually, it will be clear from the context which interval $[a,b]$ we picked.

Theorem 8.1. *Orthogonal polynomials on* $[a,b]$.

There is a sequence of $\{p_k\}_{k=1}^\infty$ of polynomials satisfying

$$p_0(x) = 1, \tag{4}$$
$$p_k(x) = x^k + r_{k-1}(x) \quad \text{for } k = 1, 2, \ldots \tag{5}$$

with $r_{k-1} \in \mathbb{P}_{k-1}$ and satisfying the *orthogonality property*

$$\langle p_k, p_l \rangle = \int_a^b p_k(x)p_l(x)dx = 0 \quad \text{for } k \neq l, \tag{6}$$

and that every polynomial $q_n \in \mathbb{P}_n$ can be written as a linear combination of those orthogonal polynomials up to order n. In other words

$$\mathbb{P}_n = \mathrm{Span}\{p_0, \ldots, p_n\}$$

Proof. We start from the sequence $\{\phi_k\}_{k=0}^\infty$ of monomials $\phi_k(x) = x^k$ and apply the Gram-Schmidt orthogonalization procedure:

$$\begin{aligned}
\widetilde{p}_0 &:= 1 \\
\widetilde{p}_1 &:= \phi_1 - \frac{\langle \phi_1, \widetilde{p}_0 \rangle}{\|\widetilde{p}_0\|^2}\widetilde{p}_0 \\
\widetilde{p}_2 &:= \phi_2 - \frac{\langle \phi_2, \widetilde{p}_0 \rangle}{\|\widetilde{p}_0\|^2}\widetilde{p}_0 - \frac{\langle \phi_2, \widetilde{p}_1 \rangle}{\|\widetilde{p}_1\|^2}\widetilde{p}_1 \\
&\ldots \\
\widetilde{p}_k &= \phi_k - \sum_{j=0}^{k-1} \frac{\langle \phi_k, \widetilde{p}_j \rangle}{\|\widetilde{p}_j\|^2}\widetilde{p}_j
\end{aligned}$$

By construction, $\widetilde{p}_n \in \mathbb{P}_n$ and $\langle p_k, p_l \rangle = 0$ for $k \neq l$. Since $\widetilde{p}_k(x) = a_k x^k + a_{k-1}x^{k-1} + \ldots a_0$, we simply define $p_k(x) = \widetilde{p}_k / a_k$ to satisfy (5).

Theorem 8.2. *Roots of orthogonal polynomials.*

Each of the polynomials p_n defined in Theorem 8.1 has n **distinct real roots**.

Proof. Without proof, will be added later for the curious among you.

Theorem 8.3. *Construction of Gaussian quadrature.*

Let $p_{n+1} \in \mathbb{P}_{n+1}$ be a polynomial on $[a, b]$ satisfying

$$\langle p_{n+1}, q \rangle = 0 \quad \forall\, q \in \mathbb{P}_n.$$

Set $\{x_i\}_{i=0}^n$ to be the $n+1$ real roots of p_{n+1} and define the weights $\{w_i\}_{i=0}^n$ by

$$w_i = \int_a^b \ell_i(x)\,\mathrm{d}x.$$

where $\{\ell_i\}_{i=0}^n$ are the $n+1$ cardinal functions associated with $\{x_i\}_{i=0}^n$. The resulting quadrature rule is a Gaussian quadrature.

Proof. Without proof, will be added later for the curious among you.

Recipe 1 to construct a Gaussian quadrature.

To construct a Gaussian formula on $[a, b]$ based on $n+1$ nodes you proceed as follows

1. Construct a polynomial $p_{n+1} \in \mathbb{P}_{n+1}$ on the interval $[a, b]$ which satisfies

$$\int_a^b p_{n+1}(x) q(x)\,\mathrm{d}x \quad \forall\, q \in \mathbb{P}_n.$$

You can start from the monomials $\{1, x, x^2, \ldots, x^{n+1}\}$ and use Gram-Schmidt to orthogonalize them.

Determine the $n+1$ **real** roots $\{x_i\}_{i=0}^n$ of p_{n+1} which serve then as quadrature nodes.

Calculate the cardinal functions $\ell_i(x)$ associated with $n+1$ nodes $\{x_i\}_{i=0}^n$ and then the weights are given by $w_i = \int_a^b \ell_i(x)\,\mathrm{d}x$.

This is the recipe you are asked to use in Exercise set 10. Alternatively one can start from a reference interval, leading to

Recipe 2 to construct a Gaussian quadrature.

To construct a Gaussian formula on $[a, b]$ based on $n+1$ nodes you proceed as follows

1. Construct a polynomial $p_{n+1} \in \mathbb{P}_{n+1}$ on the reference interval $[-1, 1]$ which satisfies

$$\int_{-1}^1 p_{n+1}(x) q(x)\,\mathrm{d}x \quad \forall\, q \in \mathbb{P}_n.$$

1. You determine the $n+1$ **real** roots $\{x_i\}_{i=0}^n$ of p_{n+1} which serve then as quadrature nodes.
2. Calculate the cardinal functions $\ell_i(x)$ associated with $n+1$ nodes $\{x_i\}_{i=0}^n$ and then the weights are given by $w_i = \int_a^b \ell_i(x)\,\mathrm{d}x$.
3. Finally, transform the resulting Gauß quadrature formula to the desired interval $[a, b]$ via

$$x_i = \frac{b-a}{2}\widehat{x}_i + \frac{b+a}{2}, \quad w_i = \frac{b-a}{2}\widehat{w}_i \quad \text{for } i = 0, \ldots n.$$

8.2 Example: Revisiting Gauß-Legendre quadrature with 2 nodes

We will now derive the Gauß-Legendre quadrature with 2 nodes we encountered in the previous lectures

Today we will use the `sympy` quite a bit, and start with the snippets

```
from sympy.abc import x # Denote our integration variable x
from sympy import integrate
```

Spend a minute and have look at integrate submodule.

First we construct the first 3 orthogonal polynomials (order 0, 1, 2) on $[0, 1]$. Spend 2 minutes to understand the code below:

```
# Interval
a, b = 0, 1

# Define scalar product
def scp(p,q):
    return integrate(p*q, (x, a, b))

# Define monomials up to order 2
mono = lambda x,m: x**m
def mono(x,m):
    return x**m

phis = [ mono(x,m) for m in range(6)]
print(phis)
```

Construct orthogonal polynomials (not normalized)

```
ps = []
for phi in phis:
    ps.append(phi)
    for p in ps[:-1]:
        ps[-1] = ps[-1] - scp(p, ps[-1])/scp(p, p)*p

print("ps")
print(ps)
```

Now write a code snippet to check whether they are actually orthogonal.

```
for p in ps:
    for q in ps:
        int_p_q = scp(p,q)
        print("int_p_q = {}".format(int_p_q))
```

Compute the roots of the second order polynomial. Of course you can do it by hand but le'ts us `sympy` for it. Spend a minute a have a look at "solve":"(https://docs.sympy.org/latest/modules/solvers/solvers.html)" submodule.

```
from sympy.solvers import solve

print(ps[-1])
xqs = solve(ps[-1])
print(xqs)
```

Next construct the cardinal functions ℓ_0 and ℓ_1 associated with the 2 roots.

```
# Non-normalized version
L_01 = (x-xqs[1])
print(L_01)
print(L_01.subs(x, xqs[1]))
print(L_01.subs(x, xqs[0]))
# Normalize
L_01 /= L_01.subs(x, xqs[0])
print(L_01.subs(x, xqs[0]))

# Non-normalized version
L_11 = (x-xqs[0])
```

```
# Normalize
L_11 /= L_11.subs(x, xqs[1])

Ls = [L_01, L_11]
print(Ls)
```

Finally, compute the weights.

```
ws = [integrate(L, (x, a, b)) for L in Ls ]
print(ws)
```