Numerical integration II: Composite quadrature rules

Anne Kværnø, André Massing

Sep 8, 2021

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

1 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.

2 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

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

where $\ell_i(x)$ are the cardinal functions associated with $\{x_i\}_{i=0}^n$ satisfying $\ell_i(x_j) = \delta_{ij}$ for $i, j = 0, 1, \dots, n$. 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 2.1. *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

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

3 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

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

3.1 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 1: Testing the accuracy of the composite trapezoidal rule

Have a look at the `CT` function

```python
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 $|I[f] - \mathrm{CT}[f]| \approx \frac{C}{m^2} = Ch^2$.

3.2 Error estimate for the composite trapezoidal rule

We will now theoretically explain the experimentally observed convergence rate in the previous Exercise 1.

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 3.1. *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.

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

.

3.3 Interlude: Convergence of h-dependent approximations

Let X be the exact solution, and $X(h)$ some numerical solution depending on a parameter h, and let $e(h)$ be the norm of the error, so $e(h) = \|X - X(h)\|$. The numerical approximation $X(h)$ converges to X if $e(h) \to 0$ as $h \to 0$. The order of the approximation is p if there exists a positive constant M such that

$$e(h) \leq Mh^p$$

The Big $\mathcal{O}$-notation we can simply write

$$e(h) = \mathcal{O}(h^p) \qquad \text{as } h \to 0.$$

This is often used when we are not directly interested in any expression for the constant M, we only need to know it exists.

Again, we see that a higher approximation order p leads for small values of h to a better approximation of the solution. Thus we are generally interested in approximations of higher order.

Numerical verification. The following is based on the assumption that $e(h) \approx Ch^p$ for some unknown constant C. This assumption is often reasonable for sufficiently small h.

Choose a test problem for which the exact solution is known and compute the error for a decreasing sequence of h_k's, for instance $h_k = H/2^k$, $k = 0, 1, 2, \ldots$. The procedure is then quite similar to what was done for iterative processes.

$$\begin{array}{rcl} e(h_{k+1}) & \approx & Ch_{k+1}^p \\ e(h_k) & \approx & Ch_k^p \end{array} \quad \Rightarrow \quad \frac{e(h_{k+1})}{e(h_k)} \approx \left(\frac{h_{k+1}}{h_k}\right)^p \quad \Rightarrow \quad p \approx \frac{\log\left(e(h_{k+1})/e(h_k)\right)}{\log\left(h_{k+1}/h_k\right)}$$

For one refinement step where one passes from $h_k \to h_{k+1}$, the number

$$EOC(k) \approx \frac{\log\left(e(h_{k+1})/e(h_k)\right)}{\log\left(h_{k+1}/h_k\right)}$$

is often called the "Experimental order of convergence at refinement level k"

Since

$$e(h) \approx Ch^p \quad \Rightarrow \quad \underbrace{\log e(h)}_{y} \approx \underbrace{\log C}_{a} + p\underbrace{\log h}_{x}$$

a plot of $e(h)$ as a function of h using a logarithmic scale on both axes (a log-log plot) will be a straight line with slope p. Such a plot is referred to as an *error plot* or a *convergence plot*.

time	velocity	acceleration
0.0	1.4186	-5.01
2.0	1.376512	11.919
4.0	1.1E+1	14.717624

Level	Error	EOC
0	0.008202	inf
1	0.002047	2.002789
2	0.000511	2.000696
3	0.000128	2.000174
4	0.000032	2.000043

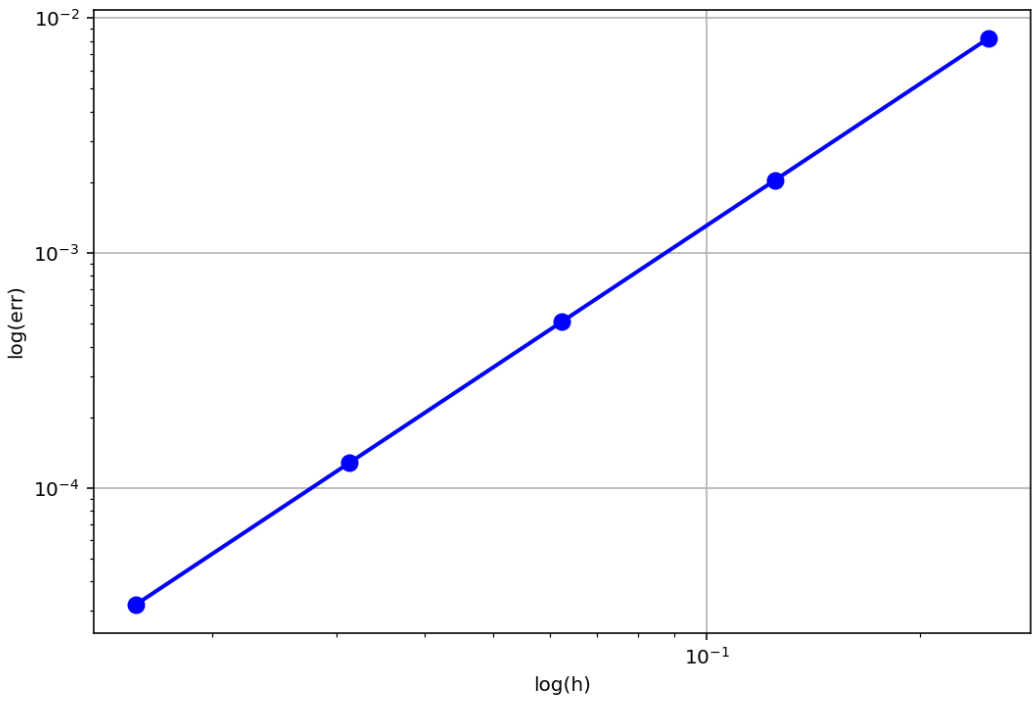

Figure 1: $\log(err) - \log(h)$ plot for the composite trapezoidal rule showing a slope of 2.

3.4 Error estimate for the composite Simpson rule

Theorem 3.2. *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.*