Section 5: Numerical integration

April 14, 2019

In this section we will discuss some of basics principles behind numerical integration. For many common integrals that appear in practice, it is simply not possible to get straightforward closed-form expressions. They must approximated numerically, meaning in an approximative sense. Like we have mentioned before, our knowledge of polynomial approximation will come in very handy in the study of numerical integration. In these notes, we will restrict ourselves to integrals that are defined on some interval of the real line.

1 Basic methods of numerical integration

Let us start with some of the basics methods which you are probably already very familiar with. Suppose we want to evaluate the integral:

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

where $f(x)$ is (for now) assumed to be a continuous function. The first thing which we may do is to break up the interval $[a, b]$ into equally sized segments of length $(b-a)/n$ and compute the left-hand Riemann sum (or the right-hand Riemann sum for that matter):

$$I_n = \sum_{i=1}^{n} f(x_{i-1})(x_i - x_{i-1}) = (b-a)/n \sum_{i=1}^{n} f(x_{i-1})$$

where;

$$x_i = a + \frac{b-a}{n} i.$$

Naturally, I_n will converge to I in the limit. But the question is, how fast and effective is this approach? We may use this result to establish the following bound:

$$\begin{aligned}
I - I_n &= \int_a^b f(x)dx - \frac{b-a}{n}\sum_{i=1}^{n} f(x_{i-1}) \\
&= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f(x)dx - \frac{b-a}{n}\sum_{i=1}^{n} f(x_{i-1}) \\
&= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f(x_{i-1}) + f^{'}(\xi(x))(x - x_{i-1})dx - \frac{b-a}{n}\sum_{i=1}^{n} f(x_{i-1}) \\
&= \sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} f^{'}(\xi(x))(x - x_{i-1})dx \\
|I - I_n| &\leq \sum_{i=1}^{n}\left(\max_{x\in[x_{i-1},x_i]} f^{'}(x)\right)\int_{x_{i-1}}^{x_i} |x - x_{i-1}|dx \\
&\leq \left(\max_{x\in[a,b]} f^{'}(x)\right)\sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} |x - x_{i-1}|dx \\
&\leq \left(\max_{x\in[a,b]} f^{'}(x)\right)\sum_{i=1}^{n}\frac{1}{2}\frac{(b-a)^2}{n^2} \\
&\leq K\frac{(b-a)^2}{2n}, \quad K := \max_{x\in[a,b]} f^{'}(x)
\end{aligned}$$

Notice how we made use of the mean-value theorem in our analysis. The error is inverse proprotional with the number of segments in used in the integration. With some simple adjustments to integration scheme, we can do a lot better than this. For example, the mid-point rule:

$$I_n = \frac{(b-a)}{n}\sum_{i=1}^{n} f\left(\hat{x}_i\right), \quad \hat{x}_i = \frac{x_{i-1} + x_i}{2}$$

does already a lot better. Indeed,

$$
\begin{aligned}
I - I_n &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx - \frac{(b-a)}{n} \sum_{i=1}^{n} f\left(\hat{x}_i\right) \\
&= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f\left(\hat{x}_i\right) + f^{'}\left(\hat{x}_i\right)\left(x - x_i\right) + \frac{f^{''}\left(\xi(x)\right)}{2}(x-\hat{x}_i)^2 dx - \frac{(b-a)}{n} \sum_{i=1}^{n} f\left(\hat{x}_i\right) \\
&= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f^{'}\left(\hat{x}_i\right)\left(x - x_i\right)dx + \int_{x_{i-1}}^{x_i} \frac{f^{''}\left(\xi(x)\right)}{2}(x-\hat{x}_i)^2 dx \\
&= \sum_{i=1}^{n} 0 + \int_{x_{i-1}}^{x_i} \frac{f^{''}\left(\xi(x)\right)}{2}(x-\hat{x}_i)^2 dx \\
|I - I_n| &\leq \left(\max_{x \in [a,b]} f^{''}\left(x\right)\right) \sum_{i=1}^{n} \frac{1}{2} \int_{x_{i-1}}^{x_i} (x-\hat{x}_i)^2 dx \\
&\leq \left(\max_{x \in [a,b]} f^{''}\left(x\right)\right) \sum_{i=1}^{n} \frac{(b-a)^3}{24n^3} \\
&\leq K\frac{(b-a)^3}{24n^2}, \quad K := \max_{x \in [a,b]} f^{''}\left(x\right)
\end{aligned}
$$

In the analysis above, we made use of Taylor's remainder theorem which you probably have learned in your calculus classes:

Theorem 1. *Let $f(x)$ be k+1 times differentiable on the inverval $[a,b]$ and let $c \in (a,b)$. Then,*

$$
f(x) = f(c) + f^{'}(c) + \ldots + \frac{f^{(k)}(c)}{k!}(x-c)^k + \frac{f^{(k+1)}(\xi)}{(k+1)!}(x-c)^k
$$

for some $\xi \in [c,x]$ if $x > c$ and $\xi \in [x,c]$ if $x < c$.

Notice that the error decays quadratically with the number of segments. Another method which has a similar performance is the trapezoidal method:

$$
I_n = \frac{(b-a)}{n} \sum_{i=1}^{n} \frac{f\left(x_i\right) + f(x_{i-1})}{2}
$$

To summarize:

- Left-hand Riemann sums:

$$|I - I_n| \leq K\frac{(b-a)^2}{2n}, \quad K := \max_{x\in[a,b]} f^{'}(x)$$

.

- Mid-point rule:

$$|I - I_n| \leq K\frac{(b-a)^3}{24n^2}, \quad K := \max_{x\in[a,b]} f^{''}(x)$$

- Trapezoidal rule:

$$|I - I_n| \leq K\frac{(b-a)^3}{12n^2}, \quad K := \max_{x\in[a,b]} f^{''}(x)$$

2 Simpon's rule: integration using quadratic interpolation

The previously constructed integration schemes relied on breaking up to domain $[a, b]$ into smaller segments. On each segment, the function is approximated by simple function which consequtively easy to integrate. In the case of left hand Riemann sums and midpoint rules, we used constant functions, whereas for the trapezoidal rule we use linear approximations. Nothing is stopping us from using quadrating interpolants within these segments. A higher order approximation may yield better results, and this gives rise to the Simpson's rule for integration.

To derive simpson's rule, let approximate the function $f(x)$ by quadratic interpolation on $[a, b]$. That is,

$$f(x) \approx f(a)\frac{(x-c)(x-b)}{(a-c)(a-b)} + f(c)\frac{(x-a)(x-b)}{(c-a)(c-b)} + f(b)\frac{(x-a)(x-c)}{(b-a)(b-c)}$$

where:

$$c = (a+b)/2.$$

Integrating yields:

$$\begin{aligned}
I &= \int_a^b f(x)dx \\
&\approx \int_a^b f(a)\frac{(x-c)(x-b)}{(a-c)(a-b)} + f(c)\frac{(x-a)(x-b)}{(c-a)(c-b)} + f(b)\frac{(x-a)(x-c)}{(b-a)(b-c)}dx \\
&\approx \frac{(b-a)}{6}\left(f(a) + 4f(c) + f(b)\right)
\end{aligned}$$

We may do this for every segment, resulting in the (composite) simpson rule

$$I_n = \frac{(b-a)}{6n}\sum_{i=1}^{n} f(x_{i-1}) + 4f(\hat{x}_i) + f(x_i), \quad \hat{x}_i = \frac{x_{i-1}+x_i}{2}$$

To establish an error bound. Write:

$$f(x) = f(\hat{x}_i) + f^{'}(\hat{x}_i)(x-\hat{x}_i) + \frac{f^{''}(\hat{x}_i)}{2}(x-\hat{x}_i)^2 + \frac{f^{'''}(\hat{x}_i)}{6}(x-\hat{x}_i)^3 + \frac{f^{(4)}(\xi)}{24}(x-\hat{x}_i)^4$$

Since:

$$f(x_{i+1}) = f(\hat{x}_i) + f^{'}(\hat{x}_i)\left(\frac{b-a}{2n}\right) + \frac{f^{''}(\hat{x}_i)}{2}\left(\frac{b-a}{2n}\right)^2 + \frac{f^{'''}(\hat{x}_i)}{6}\left(\frac{b-a}{2n}\right)^3 + \frac{f^{(4)}(\xi)}{24}\left(\frac{b-a}{2n}\right)^4$$

$$f(x_i) = f(\hat{x}_i) - f^{'}(\hat{x}_i)\left(\frac{b-a}{2n}\right) + \frac{f^{''}(\hat{x}_i)}{2}\left(\frac{b-a}{2n}\right)^2 - \frac{f^{'''}(\hat{x}_i)}{6}\left(\frac{b-a}{2n}\right)^3 + \frac{f^{(4)}(\xi)}{24}\left(\frac{b-a}{2n}\right)^4$$

we may obtain:

$$f^{''}(\hat{x}_i) = \left(f(x_{i+1}) - 2f(\hat{x}_i) + f(x_i)\right)\left(\frac{b-a}{2n}\right)^{-2} - \frac{f^{(4)}(\xi)}{12}\left(\frac{b-a}{2n}\right)^2$$

Now:

$$
\begin{aligned}
I - I_n &= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)dx - \frac{(b-a)}{6n}\left(f(x_{i-1}) + 4f(\hat{x}_i) + f(x_i)\right) \\
&= \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(\hat{x}_i) + f^{'}(\hat{x}_i)(x-\hat{x}_i) + \frac{f^{''}(\hat{x}_i)}{2}(x-\hat{x}_i)^2 + \frac{f^{'''}(\hat{x}_i)}{6}(x-\hat{x}_i)^3 + \frac{f^{(4)}(\xi)}{24}(x-\hat{x}_i) \\
&\quad - \frac{(b-a)}{6n}\left(f(x_{i-1}) + 4f(\hat{x}_i) + f(x_i)\right) \\
&= \sum_{i=1}^{n} f(\hat{x}_i)\left(\frac{b-a}{n}\right) + \frac{f^{''}(\hat{x}_i)}{3}\left(\frac{b-a}{2n}\right)^3 + \frac{f^{(4)}(\xi)}{60}\left(\frac{b-a}{2n}\right)^5 \\
&\quad - \frac{(b-a)}{6n}\left(f(x_{i-1}) + 4f(\hat{x}_i) + f(x_i)\right) \\
&= \sum_{i=1}^{n} f(\hat{x}_i)\left(\frac{b-a}{n}\right) + \left(f(x_{i+1}) - 2f(\hat{x}_i) + f(x_i)\right)\left(\frac{b-a}{6n}\right) \\
&\quad - \frac{f^{(4)}(\xi)}{24}\left(\frac{b-a}{2n}\right)^5 + \frac{f^{(4)}(\xi)}{60}\left(\frac{b-a}{2n}\right)^5 - \frac{(b-a)}{6n}\left(f(x_{i-1}) + 4f(\hat{x}_i) + f(x_i)\right) \\
&= \sum_{i=1}^{n} \frac{f^{(4)}(\xi)}{40}\left(\frac{b-a}{2n}\right)^5 \\
&= \sum_{i=1}^{n} f^{(4)}(\xi)\frac{(b-a)^5}{720n^5} \\
|I - I_n| &\leq K\frac{(b-a)^5}{1280n^4}, \quad K := \max_{x\in[a,b]} f^{(4)}(\xi)
\end{aligned}
$$

3 Quadrature formulas

In the previous sections we have derived some methods of numerical integration. All of them aimed at approximating the integral by the sum:

$$I = \int_a^b f(x)dx \approx \sum_i^n w_i f(x_i)$$

The right hand sidea above is called a quadrature sum. The numbers $x_i \in [a, b]$ are referred to as the quadrature nodes, while the numbers $w_i \in \mathbb{R}$ are called the weights. Quadratures formulas may be constructed in two ways. The first way is to approximate the function by an interpolating polynomial

$$f(x) \approx \sum_{i=1}^{n} f(x_i) L_{n-1,i}(x), \quad L_{n-1,i}(x) = \prod_{j=1, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}.$$

There is freedom in how one should pick the quadrature nodes $x_i \in [a, b]$. Once they have been chosen, the weights are found by:

$$w_i = \int_a^b L_{n-1,i}(x) dx$$

The second way of forming quadratures is to construct lower order interpolating polynomials on subinterval of $[a, b]$. These are called comoposite rules. The methods which we have derived so far were all composite rules.

4 Gaussian quadratures: Maximizing degree of precision

We have now introduced the concept of quadratures, and now big the question is how should we pick our integration nodes. For that, let us introduce a metric to quantify the performance of the quadrature rule. We observe that if use a degree n interpolating polynomial to construct a quadrature rule, then the integration will be performed exactly if the integrand is a polynomial of degree at most n. This motivates the following definition:

> A quadrature rule has degree of precision n if the rule integrates all polynomials exactly upto degree n.

Clearly, if we have n degree interpolation polynomial for our quadrature rule, the degree should also at least be n. However, we have seen that the simpson method has the error term:

$$|I - I_n| \leq K \frac{(b-a)^5}{1280 n^4}, \qquad K := \max_{x \in [a,b]} f^{(4)}(\xi).$$

Since the fourth derivative of all cubic polynomials are all equal to zero, it follows that the simpson rule has degree of precision 3. This is more than 2. Remember, simpson rule used only a quadratic function, which shows that the degree of precision can be bigger than degree of interpolating polynomial used. In fact, with a clever choice of nodes, it possible to get a degree precision of $2n-1$, where n denotes the number of nodes used in the quadrature formula.

So what should those nodes be to get those nice properties? Let me first reveal what it should not be: equispaced nodes. Equispaced nodes give rise to the Newton-cotes formulas. For the closed Newton-cotes formula (where endpoints are included in the quadrature formua), the degree of precision $n+1$. Aside from that, the method is unstable for higher order polynomial and with Runge's phenomenon on equispaced grids thing can easily turn into a disaster. Instead we should pick something different.

Let us look at the case of $n=2$. In this we case we would expect precision $2n-1 = 2 \cdot 2 - 1 = 3$. Hence the quadrature formula should be able to integrate the integrand:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

Restricting ourselves to the integration domain $[-1, 1]$, we obtain: Since:

$$w_1 f(x_1) + w_2 f(x_2) = \int_{-1}^{1} f(x) dx$$

This yield a set of 4 nonlinear equations with four unknowns:

$$\begin{aligned} w_1 1 + w_2 1 &= \int_{-1}^{1} 1 dx = 2 \\ w_1 x_1 + w_2 x_2 &= \int_{-1}^{1} x dx = 0 \\ w_1 x_1^2 + w_2 x_2^2 &= \int_{-1}^{1} x^2 dx = 2/3 \\ w_1 x_1^3 + w_2 x_2^3 &= \int_{-1}^{1} x^3 dx = 0 \end{aligned}$$

it has a unique solution:

$$w_1 = 1, \quad w_2 = 1, \quad x_1 = -\frac{\sqrt{3}}{3}, \quad x_2 = \frac{\sqrt{3}}{3}$$

It turns out that the nodes of the above solution are the Legendre polynomials of degree 2. This is no co-incidence! Recall that $l_k(x)$ for $k = 0, 1, 2, 3, \ldots$ denote the Legendre polynomials which are obtained from the Gram-schmidt process involving the monomials x^k for $k = 0, 1, 2, 3, \ldots$. Subsequently,

$$\left\langle x^j, l_k(x) \right\rangle = 0, \quad 0 = j < k.$$

We have the following theorem:

Theorem 2. *A n-point quadrature rule has degree of precision $2n - 1$ if the nodes are selected to be the roots on the n-th legendre polynomial.*

Proof. Let $P_{2n-1}(x)$ be a polynomial of degree $2n - 1$. We may factor this polynomial by:

$$P_{2n-1}(x) = Q(x)l_n(x) + r(x).$$

where $Q(x)$ is a degree $n - 1$ polynomial, $l_n(x)$ is the n-th legendre polynomials (hence degree is n), and $r(x)$ is a polynomial of degree strictly less than $n - 1$. By the orthogonality property we know that:

$$\int_{-1}^{1} Q(x)l_n(x)dx = 0.$$

Hence,

$$\int_{-1}^{1} P_{2n-1}(x)dx = \int_{-1}^{1} r(x)dx$$

Now let us apply our n point quadrature rule:

$$\sum_{i=1}^{n} w_i P_{2n-1}(x_i) = \sum_{i=1}^{n} w_i (Q(x_i)l_n(x_i) + r(x_i))$$

If we choose our nodes to be the roots of the n-the legendre polynomial, we have: $l_n(x_i) = 0$. Therefore:

$$\sum_{i=1}^{n} w_i P_{2n-1}(x_i) = \sum_{i=1}^{n} w_i r(x_i)$$

Recall that $r(x)$ was of degree less than $n-1$, which means that:

$$\sum_{i=1}^{n} w_i P_{2n-1}(x_i) = \sum_{i=1}^{n} w_i r(x_i) = \int_{-1}^{1} r(x)dx = \int_{-1}^{1} P_{2n-1}(x)dx.$$

$\square$

To find the *nodes* and *weights* for Gauss-Legendre quadrature formulas, the following can be done. Define:

$$\beta_k := \frac{1}{2\sqrt{1-\frac{1}{(2k)^2}}}, \quad k = 1, 2, \ldots, n.$$

It can be shown that the Legendre polynomials satisfy the relations:

$$x \begin{bmatrix} l_0(x) \\ l_1(x) \\ l_2(x) \\ \vdots \\ l_{n-2}(x) \\ l_{n-1}(x) \end{bmatrix} = \begin{bmatrix} 0 & \beta_1 & & & & \\ \beta_1 & 0 & \beta_2 & & & \\ & \beta_2 & 0 & \ddots & & \\ & & \beta_3 & \ddots & \beta_{n-2} & \\ & & & \ddots & 0 & \beta_{n-1} \\ & & & & \beta_{n-1} & 0 \end{bmatrix} \begin{bmatrix} l_0(x) \\ l_1(x) \\ l_2(x) \\ \vdots \\ l_{n-2}(x) \\ l_{n-1}(x) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \beta_n l_n(x) \end{bmatrix}$$

If $x_1, x_2, \ldots, x_n$ denote the roots of $l_n(x)$, we see that $x_1, x_2, \ldots, x_n$ are the eigenvalues of the matrix above. That is:

$$x_i \vec{v}_i = \begin{bmatrix} 0 & \beta_1 & & & & \\ \beta_1 & 0 & \beta_2 & & & \\ & \beta_2 & 0 & \ddots & & \\ & & \beta_3 & \ddots & \beta_n & \\ & & & \ddots & 0 & \beta_{n-1} \\ & & & & \beta_{n-1} & 0 \end{bmatrix} \vec{v}_i, \qquad i = 1, 2, \ldots, n$$

In other words, solving for the eigenvalues will find us the nodes. As for the weights, it turns out that these can be computed from the first entries of the eigenvectors. In particular,

$$w_i = 2([\vec{v}_i]_1)^2.$$