Calculation of π by trapezoidal integration

Sangwook Ryu

I present trapezoidal integration in `C/C++` with `OpenMP` parallelization and implement to calculate value of π. The question is how we can evaluate integration

$$\mathcal{I} = \int_{x_{\min}}^{x_{\max}} dx\, f(x) \tag{1}$$

of an integrand $f(x)$. In many cases, it is not feasible to perform the integration analytically and one viable option is getting an approximate value. If the integrand is regular, we have a Riemann integration.

$$\begin{aligned} \mathcal{I} &= \lim_{N\to\infty} \overline{\mathcal{I}}_{[N]} \qquad (2) \\ \overline{\mathcal{I}}_{[N]} &= \Delta_{[N]}x \sum_{i=0}^{N-1} \frac{1}{2}\left[f(x_i) + f(x_{i+1})\right] \qquad (3) \end{aligned}$$

$$\Delta_{[N]}x = \frac{x_{\max} - x_{\min}}{N} \quad \text{and} \quad x_i = x_{\min} + i\,\Delta_{[N]}x$$

In practice, when the value of $\overline{\mathcal{I}}_{[N]}$ does not change much as we keep increasing N, we can consider it converging. Quantitatively, we can look whether the following condition is met.

$$\left|\overline{\mathcal{I}}_{[2N]} - \overline{\mathcal{I}}_{[N]}\right| < \frac{\epsilon}{2}\left|\overline{\mathcal{I}}_{[2N]} + \overline{\mathcal{I}}_{[N]}\right| \tag{4}$$

for a small number ϵ, whose value is determined based on the desired accuracy. In addition, we can utilize the following recursive relation to reduce number of evaluation of the integrand.

$$\overline{\mathcal{I}}_{[2N]} = \frac{1}{2}\overline{\mathcal{I}}_{[N]} + \Delta_{[2N]}x \sum_{i=0}^{N-1} f(x_{2i+1}) \qquad \text{where} \qquad x_i = x_{\min} + i\,\Delta_{[2N]}x \tag{5}$$

One option to implement parallel computing for integration is to evenly split the interval of integration according to number of threads. In this way, we have the number of evaluation (of the integrand) per thread is equally distributed.

As an example, I demonstrate how to calculate π with numerical integration. In Euclidean geometry, π is an irrational number which relates area and circumference of a circle with its radius.

$$\pi = \frac{\text{circumference}}{2 \times \text{radius}} = \frac{\text{area}}{\text{radius} \times \text{radius}} \tag{6}$$

It is also possible to express π in terms of arctan (arctangent) function and its derivative. We can take advantage of the fact that the value of arctan function with certain arguments is given in terms of π, and derivative of arctan is rational polynomial function. In this example, we implement the following integration to obtain the value of π.

$$\begin{aligned} \pi &= 4\arctan(1) \qquad (7) \\ &= 4\int_0^1 du\, \frac{1}{1+u^2} \qquad (8) \end{aligned}$$

Also, note that there are many different integral representations. One can use arcsin (arcsine) or arccos (arccosine) and their derivative, but those methods require evaluation of square-root and can be computationally more expensive.

The trapezoidal rule is implemented in a function `get_integral_trapezoidal` in `ODESolve` namespace. The prototype and definition of `get_integral_trapezoidal` function are contained in `IntTrapezoid.h` and `IntTrapezoid.cpp`, respectively. The `main` function to calculate π is defined in `test_IntTrapezoid_pi.cpp` source file. When `OpenMP` is used, the number of threads can be specified with an environmental variable `OMP_NUM_THREADS`.