Unsuccessful attempt to speed-up numerical integration of functions with peaks and some related topics: *Monte Carlo “area” integration *Speed-up of a real uniform random number generator

Andrej Liptaj[*]

Abstract

An (unsuccessful) attempt to use “damping” functions for sharp peak integration is made. Some related comments about Monte Carlo “area” integration and speeding a uniform random number generator are made.

Introductory note: This text was previously published on Scribd[1].

1 Introduction

Reading about numerical integration methods, one notices that sharp peaks are difficult to integrate. My idea was to trade a sharp peak integration for several flat-function integrations. The idea is wrong, still I write this text to keep memory of what I have done. A clever person might probably see from the beginning that the method does not speed up the integration, unfortunately it was not my case.

2 Damping method

Method is simple: use a damping function together with Taylor expansion to integrate a sharp-peaked function. By damping function I understand a function that makes large numbers small, for example logarithm. Let g be a damping function and G its inverse $G\left[g\left(x\right)\right]=g\left[G\left(x\right)\right]=x$ and suppose g is bijective. The damping integration approach uses the Maclaurin expansion of the inverse damping, $G\left(y\right)=\sum_{i=0}^{\infty}c_i y^i$ and some standard numerical integration method that performs well for flat functions, let me note it $\int_{Std}$. The method then looks like:

$$\begin{aligned}\int f\left(x\right)dx &= \int G\left\{g\left[f\left(x\right)\right]\right\}dx \\ &= \int\left(\sum_{i=0}^{\infty}c_i\left\{g\left[f\left(x\right)\right]\right\}^i\right)dx, \\ &= \sum_{i=0}^{\infty}c_i\int_{Std}\left\{g\left[f\left(x\right)\right]\right\}^i dx.\end{aligned}$$

For the expression in the last line I will use the label “integral series”. In the Table 1 I present my choice of different damping functions together with related properties. Note please: *the statements in the table ale based only on numerical observations!* The presented method does not need to be much more slowly then the “standard method”: the value of $g\left[f\left(x\right)\right]$ needs to be calculated only once, then recursively multiplied. The integral summation (within the “standard method”) for every integration in the series needs, of course, to be done separately.

[*]Institute of Physics, Bratislava, Slovak Academy of Sciences, andrej.liptaj@savba.sk
I am willing to publish any of my ideas presented through free-publishing services in a journal, if someone (an editor) judges them interesting enough. Journals in the “*Current Contents*” database are strongly preferred.

[1]*https://www.scribd.com/document/261720474/Unsuccessful-attempt-to-speed-up-numerical-integration-of-functions-with-peaks-and-some-related-topics-Monte-Carlo-area-integration-Speed-up-of-a*

Damping g	$\tanh(x)$	$\frac{x}{\sqrt{x^2+1}}$	$\operatorname{asinh}(x)$	$\frac{2}{\pi}\arctan(x)$	$\ln(x+1)$	$\operatorname{erf}(x)$
$G = \operatorname{inv}(g)$	$\operatorname{atanh}(x)$	$\frac{x}{\sqrt{1-x^2}}$	$\sinh(x)$	$\tan\left(\frac{\pi}{2}x\right)$ *[1]*	$e^x - 1$	$\operatorname{erf}^{-1}(x)$
ith odd Maclaurin coefficient of G (even coefficients are zero for odd functions)	$a_i = \frac{1}{2i+1}$	$b_0 = 1$ $b_i = b_{i-1} \times (2i+1) \times (2i-1)$ $a_i = \frac{b_i}{(2i+1)!}$	$a_i = \frac{1}{(2i+1)!}$	$b_0 = 1$ $b_i = \frac{1}{2i+1} \times \sum_{k=0}^{i-1}(b_k \times b_{i-1-k})$ $a_i = \left(\frac{\pi}{2}\right)^{2i+1} b_i$	$a_0 = 0$ $a_{n>0} = \frac{1}{n!}$ *For all coefficients (even included)*	$b_0 = 1$ $b_i = \sum_{k=0}^{i-1}[b_k \times b_{i-1-k} \times \frac{1}{(k+1)} \times \frac{1}{(2k+1)}]$ $a_i = \left(\frac{\sqrt{\pi}}{2}\right)^{2i+1} \times \frac{b_i}{2i+1}$
Convergence of the Maclaurin series	*Converges*	*Converges*	*Converges*	*Converges*	*Converges*	*Converges*
Convergence of the "integral series"	*Diverges*	*Diverges*	*Converges rapidly*	*Converges slowly*	*Converges rapidly*	*Diverges*

Table 1: *Studied damping functions and related issues (unproven statements based on numerical observations).*

3 Results and conclusion

I used two test peak functions (rational and exponential) on the $0-1$ interval:

$$f_1(x) = \frac{100}{[100(x-0.1415926)]^4+1}, \quad \int_0^1 f_1(x) = 2.221323519290568,$$

$$f_2(x) = 10\exp\left[-1000(x-0.71828182)^2\right], \quad \int_0^1 f_2(x) = 0.560499121639793.$$

I tested two standard methods: Simpson integration rule and Monte Carlo method. It turns out that for the Simpson rule the damping method gives the same result as the standard one, only it is expressed in series. The damping procedure and Simpson approach obviously commute:

$$\int_{Simpson} f(x)\,dx = \sum_{i=0}^{\infty} c_i \int_{Simpson} \{g[f(x)]\}^i\,dx.$$

The damping method is useless - it produces the same results with additional computations. I assume the situation is analogical for all other methods based on summation over function values.

In case of the Monte Carlo approach I base my conclusions on numerical observations. The value of the integral is the product of the average function value Avg on an interval and the integral length:

$$\begin{aligned} I &= (b-a) \times Avg, \\ Avg &= \frac{1}{N}\sum_{i=1}^{N} f(x_i^{rnd.}), \end{aligned}$$

where $x^{rnd.}$ stands for a random number obtained from a uniform distribution over the interval (a,b). I let run both methods for same time and I observe that both methods are competitive, without possibility to claim that damping method is better. Let me notice that for all integrals in the series I used only one random number, so the integrals are in some way correlated. The results suggest the damping method commutes with the MC integration procedure and thus is of no use.

4 Some related ideas

4.1 MC "area" integration

I suppose that mostly for pedagogical reasons the "area" MC integration method is sometimes presented in an introduction to MC integration (for positive functions). It differs from the standard method described in the Section 3, it requires an

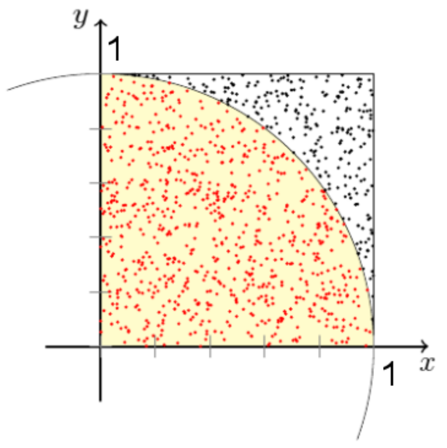

Figure 1: MC "area" integration to estimate the integral $\int_0^1 f(x)\,dx$, $f(x) = \sqrt{1-x^2}$.

additional random number to obtain the coordinate of a random point in the "value" direction. The method relates the area (volume) under the integrand to the number of randomly generated points there and estimates the integral using the fraction of this number to the total number of points in a rectangle, which contains the whole function graph, see the Figure 1. The formula looks like:

$$\begin{aligned} I &= \frac{\#\,(points\ under)}{\#\,(all\ points)} \times (rectangle\ area)\,, \\ &= \frac{\#\,(points\ under)}{\#\,(all\ points)} \times (b-a) \times (rectangle\ height)\,. \end{aligned}$$

This method is less effective then the "standard" MC integration and honestly I see no reason why it might be preferred. However, if for any reason someone sticks to this method, he encounters a problem: the maximal value of the function is, in general, unknown, so one cannot tune the rectangle height universally and write down a general code that would work with any function. In this very specific situation the damping method helps - one simply chooses a damping function bounded from above by one, for example $\frac{2}{\pi}\arctan(x)$. Then the rectangle height can be universally fixed to one and the value of the integral calculated using integral series, where "standard" method is the area MC method (in each term the value remains under one, since $0 < x < 1 \Rightarrow x^n < 1$).

4.2 Method of almost doubling the speed of a real uniform random number generator

This paragraph is almost unrelated to the topic of the article, however the idea came to me when considering the additional random number needed in case of the area MC integration. Suppose you have some random number generator that provides you with a random number. A new one could be quickly obtained from a previous one by inverting the order of its digits. Let me be more precise: imagine a random number (in some numeral system) from uniform distribution between 0 and 1 written in the following way:

$$x_{RND} = 0.a_1a_2a_3\ldots a_{N-1}a_N$$

where a_i represents a digit and the number of digits N is limited because of practical reasons (length of a computer register). Then a second number

$$y = 0.a_Na_{N-1}\ldots a_3a_2a_1$$

is very few correlated with the first one. The argument is simple: a random number means random digits. Since random number is mostly used by its value (size, magnitude) and not its digits, only first few digits play a role in the use. When the order of digits is inverted then some other random digits, before negligible, play a role. The method of course supposes a sufficient number of digits for a random number.

I did an internet search and it seems that processors do not have order-inversion of bits in a register as basic instruction, thus this would need to be programmed on a higher level. Such an instruction should be however easy to hard-wire into a processor.

References

[1] Raymond Manzoni, http://math.stackexchange.com/questions/286529/how-to-expand-tan-x-in-taylor-order-to-ox6