2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

Fast Verified Automatic Integration Algorithm using Complex Analysis

Naoya YAMANAKA[†,♯], Takeshi OGITA[‡], Masahide KASHIWAGI[♮] and Shin'ichi OISHI[♮,♯]

†Graduate School of Science and Engineering, Waseda University
3-4-1 Okubo Shinjuku, Tokyo, 169-8555 Japan
‡Department of Mathematics, Tokyo Woman's Christian University
2-6-1 Zenpukuji, Suginami, Tokyo, 167-8585 Japan
♮ Faculty of Science and Engineering, Waseda University
3-4-1 Okubo Shinjuku, Tokyo, 169-8555 Japan
♯ CREST, Japan Science and Technology Agency (JST)
Email: naoya_yamanaka@suou.waseda.jp

Abstract—This paper is concerned with verified automatic integration of a univariate function. For this purpose, interval arithmetic and automatic differentiation have widely been used to calculate the error of integral. In this paper, we propose fast verified automatic integration algorithm using an a priori error algorithm for rounding errors in floating-point arithmetic and new algorithm for truncation error which is based on complex analysis. Numerical results are illustrated that proposed verified algorithm is from 5 to 15 times faster than conventional verified method.

1. Introduction

Assume $f(x) \in C^{\infty}([a,b])$. We consider the following two types of integrals with respect to x from a to b:

$$I_1 = \int_a^b f(x)\,dx$$

$$I_2 = \int_a^b \frac{f(x)}{\sqrt{1-x^2}}dx.$$

To verify these types of integrals using numerical computations, the evaluations of rounding error and truncation error are needed.

Rounding error for verified computations is more or less calculated by interval arithmetic, but the method is much slower than pure floating-point arithmetic. Furthermore, to get the upper bound, whole calculation by interval arithmetic must be completed. To solve these problems, we stress that Kashiwagi's method, which is an a priori error algorithm to calculate the upper bound of rounding errors in floating-point arithmetic, is useful. After running some algorithm with Kashiwagi's method once, we only need to execute pure floating-point arithmetic, so that verified computation is expected to get much faster.

The evaluation of truncation error depends on the quadrature in use. In this paper we adopt Gauss-Legendre quadrature and Clenshaw-Curtis quadrature for I_1 and Gauss-Chebyshev quadrature for I_2 respectively. Truncation error of these algorithms includes differential of high order depending on the number of points n. Let us define n_{opt} be the minimum n satisfying the following equation:

$$\left|E^{(n_{\text{opt}})}\right| < \varepsilon,$$

where $E^{(n_{\text{opt}})}$ denotes truncation error of some quadrature and ε does an absolute tolerance. For some tolerance to calculate differential of high order, there are two possibilities; automatic differentiation and the algorithm based on complex analysis.

Automatic differentiation has been widely used for this problem [2, 3]. It requires much computational effort when the number of derivative order become large, because it must use interval arithmetic throughout the computation.

Recent years, Petras has proposed an efficient verified algorithm for these quadratures using complex analysis [4, 5]. Execution time of his algorithm depends slightly on the number of derivative order n so that when n is large, it is relatively faster than automatic differentiation. Furthermore, his algorithm can calculate the upper bounds for "difficult" problems. However, using his algorithm the number of points n sometimes becomes bigger than n_{opt} and it takes a lot of time.

In this paper, we propose a new algorithm to get the upper bound of truncation error using complex analysis. Using this algorithm, we can frequently obtain smaller n than Petras's algorithm, so that the proposed algorithm is expect to be faster.

2. Framework of Verified Automatic Integration Algorithm

We show the framework of verified automatic integration algorithm as follows:

Algorithm 1
Framework of verified automatic integration algorithm when user inputs the integral and relative tolerance.

Step 1 *Get the order of the true value and estimate rounding errors.*

Step 2 *Rewrite inputted relative tolerance to absolute tolerance.*

Step 3 *Calculate the number of points n satisfying rewritten absolute tolerance.*

Step 4 *Calculate the integral using a n-point quadrature with rounding error.*

3. A Priori Error Algorithm for Rounding Error

In verified numerical computations, all rounding errors that occur throughout the algorithm must be taken into account. Although rounding error can be counted by interval arithmetic, it is much slower than pure floating-point arithmetic. Moreover it is not until all calculations have done in interval arithmetic that we could get the upper bound of rounding errors.

To solve these problems, we emphasis that Kashiwagi's method, which is an a priori error algorithm to calculate the upper bound of rounding errors in floating-point arithmetic, is useful for verified algorithm. In the case that some numerical algorithm computes the same function with a number of different points, with Kashiwagi's method we can expect the algorithm to become faster than that with interval arithmetic, because the evaluations of the function are executed by pure floating-point operations. Let us show the detail of Kashiwagi's method.

Consider the binary operation $\tilde{z} = g(\tilde{x}, \tilde{y})$. Denote $\tilde{x}$ and $\tilde{y}$ in the intervals I_x and I_y by approximate values of x and y in I_x and I_y, respectively. Suppose

$$|x - \tilde{x}| \le \varepsilon_x, \quad |y - \tilde{y}| \le \varepsilon_y$$

hold. In addition, assume the following inequality is satisfied:

$$|\tilde{z} - g(\tilde{x}, \tilde{y})| \le |g(\tilde{x}, \tilde{y})| \varepsilon_M. \tag{1}$$

Then, the following inequality holds for $z \in I_z$:

$$|z - \tilde{z}| \le |D_x| \varepsilon_x + |D_y| \varepsilon_y + |I_z| \varepsilon_M.$$

Here, let us suppose the interval I_z holds

$$I_z \supset \left\{ g(x,y) \mid x \in I_x, y \in I_y \right\},$$

and intervals D_x, D_y hold

$$D_x \supset \left\{ \frac{\partial g}{\partial x}(x,y) \mid x \in I_x, y \in I_y \right\}$$

$$D_y \supset \left\{ \frac{\partial g}{\partial y}(x,y) \mid x \in I_x, y \in I_y \right\}.$$

The single operation $z = g(x)$ is similar to the binary operation.

We make the pair $(I,\ \varepsilon)$ as

I : An input interval into the operation

ε : Collected errors until the operation,

and define every operation for the pair.

For examples, "+" operator is defined by

$$(I_x, \varepsilon_x) + \left(I_y, \varepsilon_y\right) = \left(I_x + I_y,\ \varepsilon_x + \varepsilon_y + \left|I_x + I_y\right| \varepsilon_M\right).$$

To similar, "·" operator is defined by

$$(I_x, \varepsilon_x) \cdot \left(I_y, \varepsilon_y\right) = \left(I_x \cdot I_y,\ \varepsilon_x \cdot \varepsilon_y + \left|I_x \cdot I_y\right| \varepsilon_M\right).$$

With bottom-up calculation by recursive use of the defined operation, we can get an upper bound of rounding errors when evaluating a point of a function in floating-point arithmetic.

Algorithm 2
Computation of an a priori error algorithm of rounding errors when evaluating $f(\xi)$ in floating-point arithmetic ($a \le \xi \le b,\ \xi \in \mathbb{F}$).

Step 1 *Set $I = [a, b]$.*

Step 2 *Make a pair $x = (I, 0)$.*

Step 3 *Calculate $y = f(x)$ with the pair.*

Step 4 *Output the second value ε_y of y.*

4. Error Analysis of Truncation Error

4.1. Error Term of Some Quadratures

In this paper we adopt Gauss-Legendre quadrature and Clenshaw-Curtis quadrature for I_1 and Gauss-Chebyshev quadrature for I_2. Truncation error of these algorithms are as follows:

- n point Gauss-Legendre quadrature

$$\left|E_{gl}^{(n)}\right| \le \frac{(b-a)^{2n+1}(n!)^4}{(2n+1)\left[(2n)!\right]^3} \max_{x \in [a,b]} \left|f^{(2n)}(x)\right|$$

- n point Clenshaw-Curtis quadrature

$$\left|E_{cc}^{(n)}\right| \le \frac{8}{(n+1)!} \left(\frac{b-a}{4}\right)^{n+2} \max_{x \in [a,b]} \left|f^{(n+1)}(x)\right|$$

- n point Gauss-Chebyshev quadrature

$$\left|E_{gc}^{(n)}\right| \le \frac{4\pi}{(2n)!} \left(\frac{b-a}{4}\right)^{2n+1} \max_{x \in [a,b]} \left|f^{(2n)}(x)\right|$$

4.2. Error Analysis of High Order Differential

Consider the simply connected domain on complex plane D. Suppose that $f(z)$ is analytic on D and z_0 denotes the point on D. In addition, C denotes a simply closed curve around z_0. Then Goursat's theorem is as follows (e.g. [1]):

$$f^{(n)}(z_0) = \frac{n!}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{n+1}} dz$$

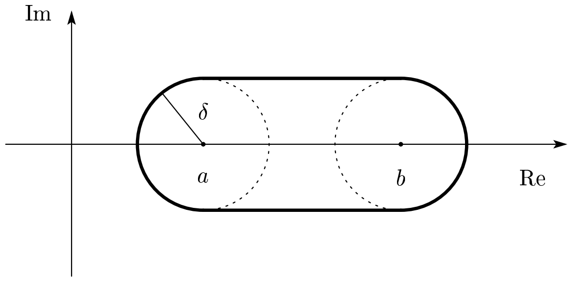

Figure 1: Figure of suitable D.

Next, suppose the line segment on real axis $a \le x \le b$ in C, then

$$\left|f^{(n)}(x)\right| \le \frac{n!}{2\pi}\int_C \frac{|f(z)|\,|dz|}{|z-x|^{n+1}} \le \frac{n!LM}{2\pi\delta^{n+1}},$$

which implies,

$$\max_{x\in[a,b]}\left|f^{(n)}(x)\right| \le \frac{n!LM}{2\pi\delta^{n+1}}.$$

Here, L denotes the length of C, M does $\max_{z\in C}|f(z)|$ and δ does the minimum distance to the point of $a \le x \le b$.

A suitable domain of D and a suitable shape of C for estimating differential of high order are as follows:

$$D: \quad |z-t| \le \delta \qquad \text{for } \forall t \quad \text{s.t.} \quad a \le t \le b$$
$$C: \quad \text{Closed curve on the edge of } D.$$

Then δ become the radius of the circles whose centers are a and b on Figure 1. Because M depends only on δ, let us rewrite M as

$$M_\delta := \max_{z\in C}|f(z)|.$$

Since the circular disk $|z-t| \le \delta$ is contained on D for the fixed t in $[a,b]$,

$$\left|f^{(n)}(x)\right| \le \frac{n!2\pi\delta M_\delta}{2\pi\delta^{n+1}} = \frac{n!M_\delta}{\delta^n}$$

holds, and therefore

$$\max_{x\in[a,b]}\left|f^{(n)}(x)\right| \le \frac{n!M_\delta}{\delta^n}$$

holds [1]D

We can rewrite truncation error of each quadrature as follows:

- n point Gauss-Legendre quadrature

$$\left|E_{gl}^{(n)}\right| \le \frac{5}{4}(b-a)M_\delta\left(\frac{b-a}{\sqrt{15}\delta}\right)^{2n}$$

- n point Clenshaw-Curtis quadrature

$$\left|E_{cc}^{(n)}\right| \le 8(b-a)M_\delta\left(\frac{b-a}{4\delta}\right)^{n+1}$$

- n point Gauss-Chebyshev quadrature

$$\left|E_{gc}^{(n)}\right| \le \pi(b-a)M_\delta\left(\frac{b-a}{4\delta}\right)^{2n}$$

4.3. New Algorithm using Complex Analysis

Recent years, Petras has proposed a verification algorithm for these quadratures [4, 5]. Execution time of his algorithm depends slightly on n so that when n is large, it is relatively faster than automatic differentiation. Furthermore, his algorithm can treat sensitive problems.

Using his algorithm, however, δ is fixed so that n sometimes becomes much bigger than n_{opt} and it takes a lot of time. To overcome them, we propose the algorithm that calculates some variable δ.

Algorithm 3
Computation of the algorithm outputs adequate n when user inputs the integral and absolute tolerance.

Step 1 *Set N such that execution time of N point quadrature is not exceed to that of calculate M_δ.*

Step 2 *For a certain δ, calculate M_δ. In addition, calculate the number of n that satisfy inputted absolute error using δ and M_δ. If $n < N$, outputs n.*

Step 3 *For some δ, do Step 2.*

Step 4 *Calculate function $\tilde{M}(\delta)$ by interpolation using calculated some M_δ.*

Step 5 *Calculate minimal point of $\tilde{M}(\delta)$ as δ^*.*

Step 6 *Calculate M_{δ^*} and the number of n, and output n.*

Using this algorithm, we can frequently calculate faster than Petras's algorithm.

5. The Proposed Algorithm

Summarizing the above mentioned discussions, we propose the following algorithm.

Algorithm 4
Computation of proposed verified automatic integration algorithm when user inputs the integral and relative tolerance.

Step 1 *Calculate the upper bound of rounding error using Kashiwagi's method.*

Step 2 *Get the order of the true value.*

Step 3 *Rewrite inputted relative tolerance to absolute tolerance.*

Step 4 *Calculate the number of points n satisfying rewritten absolute tolerance and the upper bound of rounding errors.*

Step 5 *Calculate the integral using a n-point quadrature with pure floating-point numbers.*

In this algorithm, Algorithm 2 is used in Step 1, and Algorithm 3 is used in Step 2 and Step 4.

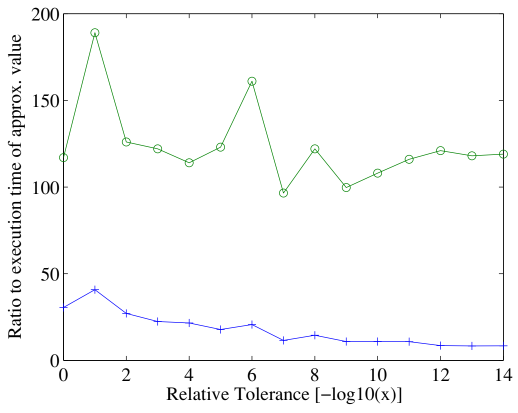

Figure 2: Ratio of execution time using I_1, Algorithm 5 normed to 1.

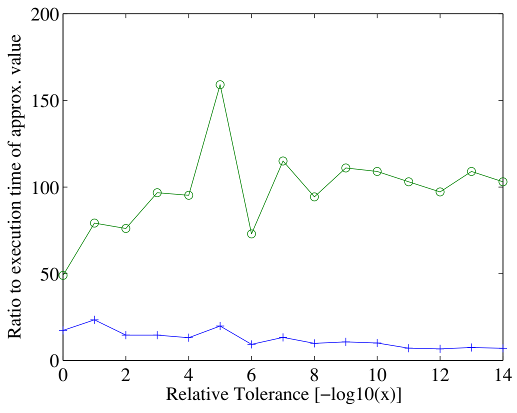

Figure 3: Ratio of execution time using I_2, Algorithm 5 normed to 1.

6. Numerical Results

- Linux (Fedora8), Memory 8GB, Intel Core 2 Extreme 3.0GHz (Use 1 Core Only), GCC with CRlibm. (CRlibm is used to satisfy (1).)
- The Conventional Method :
  - Rounding error : Interval arithmetic
  - Truncation error : Petras's algorithm [4] Unfortunately, because we could not find C file of Petras's algorithm, we did our best to make it based on the algorithm of chapter 3 in [4], equally to the proposed algorithm as much as possible.

We compare the proposed algorithm and the conventional method with an approximated automatic integration algorithm as follows:

Algorithm 5
Automatic integration algorithm for approximate value when user inputs the integral and relative tolerance $\tilde{\varepsilon}$.

```
Step 1  Set the number of points $n_0$.
Step 2  Calculate integral $s_0$ using a quadrature with $n_0$.
Step 3  Set $k = 1$.
Step 4  Set $n_k = 2n_{k-1}$.
Step 5  Calculate integral $s_k$ using a quadrature with $n_k$.
Step 6  Check
            $\left|\frac{s_k - s_{k-1}}{s_k}\right| < \tilde{\varepsilon}.  \quad (s_k \neq 0)$
        If satisfy, output $s_k$.
        If not, reset $k = k + 1$ and go back to Step 4.
```

Examples

$$I_1 = \int_{-1}^{1} \sin(\exp(x))dx = 1.455\cdots$$
$$I_2 = \int_{-1}^{1} \frac{x\exp(x)}{\sqrt{1-x^2}}dx = 1.775\cdots$$

Figure 2 and 3 display the ratio of execution time for the verification algorithms to that for Algorithm 5 for I_1 with Gauss-Legendre quadrature and I_2 with Gauss-Chebyshev quadrature, respectively. A glance at Figures 2 and 3 will reveal that the execution time of the conventional method are about 100 times slower than Algorithm 5. On the other hand, the proposed algorithm is from 5 to 25 times slower. It means that the proposed algorithm is from 5 to 15 times faster than the conventional method.

References

[1] P. J. Davis and P. Rabinowitz: Methods in Numerical Integration. Academic Press,. New York, 1975.

[2] G. F. Corliss, Computing Narrow Inclusions for Definite Integrals, in: Scientific Computation and Programming Languages, Teubner, Stuttgart, 1987, pp. 150-169.

[3] G.F. Corliss, L.B. Rall, Adaptive, Self-Validating Numerical Quadrature, SIAM J. Sci. Statist. Comput. 8 (1987) 831-847.

[4] K. Petras: Self-Validating Integration and Approximation of Piecewise Analytic Functions, J. Comput. Appl. Math. 145 (2002) 345-359.

[5] K. Petras: Principles of Verified Numerical Integration, J. Comput. Appl. Math. 199 (2007) 317-328.