A Verified Automatic Contour Integration Algorithm

Naoya Yamanaka*, Shin'ichi Oishi† and Takeshi Ogita‡
*Graduate School of Science and Engineering, Waseda University
†Faculty of Science and Engineering, Waseda University
‡Department of Mathematics, Tokyo Woman's Christian University
*naoya_yamanaka@suou.waseda.jp

Abstract. A verified automatic integration algorithm is proposed for calculating contour integration over complex field using numerical computations. The proposed algorithm is based on trapezoidal rule for angle. The error analysis of the method have been presented by several authors, however, these investigations are done basically for examining the rates of convergence, and several constants in these error formula were left unevaluated. In order to construct verified numerical integrator using the algorithm, the error formula is presented. To construct efficient verified numerical integrator, an efficient a priori method of evaluating function calculation errors is adopted. Combining these, a verified automatic integration algorithm is proposed. Numerical examples are presented for illustrating effectiveness of the proposed algorithm.

Keywords: numerical integration, verification, contour integraion

1. Introduction

This paper concerns with a verified numerical computation of the line integral over complex field. The purpose of this paper is to present an efficient automatic inclusion algorithm for a certain class of contour integrals. A type of integrals considered in this paper are following:

$$I = \int_C f(z)dz, \tag{1}$$

$$C : |z - z_0| = r. \tag{2}$$

Here, let us assume that $f(z)$ can be Laurent expansion such that

$$f(z) = \sum_{j=0}^{\infty} a_j (z - z_0)^j + \sum_{j=1}^{\infty} b_j (z - z_0)^{-j},$$

here, a_j and b_j denote

$$a_j = \frac{1}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{j+1}} dz \tag{3}$$

$$b_j = \frac{1}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{1-j}} dz. \tag{4}$$

4th International Workshop on Reliable Engineering Computing (REC 2010)
Edited by Michael Beer, Rafi L. Muhanna and Robert L. Mullen

ISBN: 978-981-08-5118-7. *Published by* Research Publishing Services.
doi:10.3850/978-981-08-5118-7_045

To get an approximate value of this type of integrals, it is well known that trapezoidal rule for angle of the circle Eq. (2) is one of the most efficient method (*e.g.* (Hirayama, 1994)). The idea of this method is to transform a given problem into an integral over $[0, 2\pi]$ via a change of variable:

$$I = \int_C f(z)\,dz \tag{5}$$

$$= \int_0^{2\pi} ir\exp(i\theta)\, f(z_0 + r\exp(i\theta))\,d\theta \tag{6}$$

and then apply the n-point trapezoidal formula with the width

$$h = \frac{2\pi}{n}$$

to the transformed integral. Thus, the trapezoidal rule is explicitly written like

$$I = \int_C f(z)\,dz \tag{7}$$

$$\simeq irh\sum_{k=0}^{n-1} \exp(ihk)\, f(z_0 + r\exp(ihk)) =: I_n. \tag{8}$$

The error analysis of the trapezoidal rule over complex field have been presented by several authors (Rabinowitz, 1968; Hirayama, 1994). These investigations are done basically for examining the rates of convergence, and several constants in these error formula were left unevaluated.

In this paper, we assume that a tolerance of numerical integration error is given. Then, we present a theorem which gives an upper bound of the error of the method. Then, based on this theorem, we propose an automatic inclusion algorithm for Eq. (1). Here, the automatic inclusion algorithm means an algorithm calculating an interval $\tilde{I}$ s.t.

$$\left|\frac{I - \tilde{I}}{I}\right| \leqq \frac{\mathrm{rad}(\tilde{I})}{|I|} \leqq \varepsilon_{rel},$$

where ε_{rel} is a given relative tolerance and $\mathrm{rad}(\tilde{I})$ denotes the radius of $\tilde{I}$.

By numerical experiments it is shown that the proposed algorithm is efficient, *i.e.* we will present numerical examples for a certain class which show inclusions for several contour integrals by our proposed algorithm can be obtained with almost the same computational time than that for obtaining approximate values of the integrals by standard trapezoidal rule provided that the same relative tolerance is posed.

2. Error Analysis and A Priori Error Algorithm for Function References

2.1. Preliminary

In this paper, we are concerned with a problem of calculating an inclusion of the contour integral defined by Eq.(1) within a given tolerance. Here, let us assume that $f(z)$ can be Laurent expansion such that

$$f(z) = \sum_{j=0}^{\infty} a_j (z - z_0)^j + \sum_{j=1}^{\infty} b_j (z - z_0)^{-j}, \tag{9}$$

here, a_j and b_j denote

$$a_j = \frac{1}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{j+1}} dz \tag{10}$$

$$b_j = \frac{1}{2\pi i} \int_C \frac{f(z)}{(z - z_0)^{1-j}} dz. \tag{11}$$

We apply a variable transformation

$$z = z_0 + r \exp(i\theta), \qquad 0 \le \theta \le 2\pi$$

to Eq. (1), then we have

$$I = \int_0^{2\pi} ir \exp(i\theta) f(z_0 + r \exp(i\theta)) d\theta. \tag{12}$$

If we apply the trapezoidal rule to Eq. (12) with a mesh size h, we have an approximation of the definite integral defined by Eq.(1) as

$$I_n = irh \sum_{k=0}^{n-1} \exp(ihk) f(z_0 + r \exp(ihk)). \tag{13}$$

Here, n is a suitable positive integers and

$$h = \frac{2\pi}{n}.$$

Let r_1, r_2 be a positive constant satisfying $r_1 < r < r_2$. We define an annulus D by

$$D = \{z \in \mathbb{C} : r - r_1 < |z - z_0| < r_2 - r\}.$$

Then, we define the following function space:

Definition 1
Let K, α, β be positive constants. Then $\mathbf{L}(D)$ denotes the family of all functions f that are holomorphic on D, and satisfy the condition that

$$\max\left(\max_{z \in D_1} |f(z)|, \max_{z \in D_2} |f(z)|\right) \leqq K, \tag{14}$$

where D_1 and D_2 are defined by

$$D_1 = \{z \in \mathbb{C} : |z - z_0| = r_1\} \tag{15}$$

$$D_2 = \{z \in \mathbb{C} : |z - z_0| = r_2\}. \tag{16}$$

Now, we consider to estimate an error

$$E_n(f) := |I_n - I|.$$

Clearly it follows from Eq.(9) that

$$E_n(f) = \left| irh \sum_{k=0}^{n-1} \exp(ihk) \left(\sum_{j=0}^{\infty} a_j r^j \exp(ijhk) + \sum_{j=1}^{\infty} b_j r^{-j} \exp(-ijhk) \right) - I \right| \tag{17}$$

$$= \left| irh \left(a_0 \sum_{k=0}^{n-1} e^{ihk} + \sum_{j=1}^{\infty} a_j r^j \sum_{k=0}^{n-1} e^{ih(j+1)k} + \frac{b_1 n}{r} + \sum_{j=2}^{\infty} b_j r^{-j} \sum_{k=0}^{n-1} e^{-ih(j-1)k} \right) - I \right|. \tag{18}$$

Since $\sum_{k=0}^{n-1} e^{ihk}$ is the sum of geometric series, we have

$$\sum_{k=0}^{n-1} \exp(ihk) = \frac{1 - \exp(ihn)}{1 - \exp(ih)} = \frac{1 - \exp(i2\pi)}{1 - \exp(ih)} = 0,$$

and from the definition of b_1, it is seen that $ihnb_1$ is equivalent to I as follows:

$$ihnb_1 = 2\pi i \left(\frac{1}{2\pi i} \int_C f(z)\, dz \right) = \int_C f(z)\, dz = I.$$

Then we have

$$E_n(f) = \left| irh \left(\sum_{j=1}^{\infty} a_j r^j \sum_{k=0}^{n-1} \exp(i(j+1)hk) + \sum_{j=2}^{\infty} b_j r^{-j} \sum_{k=0}^{n-1} \exp(-i(j+1)hk) \right) \right|. \tag{19}$$

It is easily verified that

$$\sum_{k=0}^{n-1} \exp(i(j+1)hk) = \sum_{k=0}^{n-1} \exp(-i(j+1)hk) = \begin{cases} n & : \ n \mid j+1 \\ 0 & : \ \text{otherwise.} \end{cases} \tag{20}$$

Hence, we have

$$E_n(f) = \left| irh \left(\sum_{j=1}^{\infty} a_{jn} r^{jn} n + \sum_{j=2}^{\infty} b_{jn} r^{-jn} n \right) \right| \tag{21}$$

$$= \left| 2\pi i r \left(\sum_{j=1}^{\infty} a_{jn} r^{jn} + \sum_{j=2}^{\infty} b_{jn} r^{-jn} \right) \right|. \tag{22}$$

2.2. Main Theorem

We now consider the problem of constructing an inclusion algorithm by calculating the error term reviewed in the previous subsection. More concretely, we will consider an upper bound of the error term $|E_n(f)|$, and the number of points n satisfying that $|E_n(f)|$ is below the tolerance.

Here, we present a theorem which shows the upper bound of $E_n(f)$:

$$E_n(f) = |I_n - I| \tag{23}$$

$$= \left| irh \sum_{k=0}^{n-1} \exp(ihk) f(z_0 + r\exp(ihk)) - \int_C f(z)dz \right|. \tag{24}$$

Theorem 1
Let r, r_1, r_2 be positive constant satisfying $r_1 < r < r_2$ and let $f \in \mathbf{L}(D)$. Furthermore, $g(w)$ denote

$$g(w) = f(z_0 + \log(w - i\theta)\exp(i\theta)) = f(\varphi^{-1}(w)). \tag{25}$$

Here $\varphi^{-1}(w)$ denotes

$$\varphi^{-1}(w) = z_0 + \log(w - i\theta)\exp(i\theta).$$

Then

$$E_n(f) = |I_n - I| \tag{26}$$

$$= 2\pi\exp(r)K\left[\left(\frac{\exp(-n(r_2 - r))}{1 - \exp(-n(r_2 - r))}\right) + \left(\frac{\exp(-2n(r - r_1))}{1 - \exp(-n(r - r_1))}\right)\right] \tag{27}$$

holds.

Proof.

First of all, we can see from (25) that the annulus D is transformed to an annulus $\varphi(D)$:

$$\varphi(D) = \{z \in \mathbb{C} : \exp(r_1) < |w| < \exp(r_2)\}.$$

Let us $g(z)$ be Laurent expansion such that

$$g(w) = \sum_{j=0}^{\infty} a_j w^j + \sum_{j=1}^{\infty} b_j w^{-j},$$

then g is regular and single valued in an annulus A which contains $|w| = \exp(r)$ in its interior. Moreover

$$a_k = \frac{1}{2\pi i}\int_{\varphi(D)} \frac{g(w)}{w^{k+1}}dw, \qquad b_k = \frac{1}{2\pi i}\int_{\varphi(D)} \frac{g(w)}{w^{1-k}}dw,$$

where D lies in A and contains $|w| = \exp(r)$ in its interior.

$$E_n(f) = 2\pi i r \left(\sum_{j=1}^{\infty} a_{jn} r^{jn} + \sum_{j=2}^{\infty} b_{jn} r^{-jn} \right) \tag{28}$$

$$= 2\pi i \exp(r) \left(\sum_{j=1}^{\infty} \left(\frac{1}{2\pi i} \int_{\varphi(D)} \frac{g(w)}{w^{jn+1}} dw \right) \exp(r)^{jn} + \sum_{j=2}^{\infty} \left(\frac{1}{2\pi i} \int_{\varphi(D)} \frac{g(w)}{w^{1-jn}} dw \right) \exp(r)^{-jn} \right) \tag{29}$$

$$= \int_{\varphi(D)} g(w) \sum_{j=1}^{\infty} \left(\frac{\exp(r)}{w} \right)^{jn+1} dw + \int_{\varphi(D)} g(w) \sum_{j=2}^{\infty} \left(\frac{\exp(r)}{w} \right)^{1-jn} dw \tag{30}$$

$$= \int_{\varphi(D_2)} g(w) \sum_{j=1}^{\infty} \left(\frac{\exp(r)}{w} \right)^{jn+1} dw + \int_{\varphi(D_1)} g(w) \sum_{j=2}^{\infty} \left(\frac{\exp(r)}{w} \right)^{1-jn} dw, \tag{31}$$

where D_1 and D_2 are defined by Eq. (15) and Eq. (16) respectively.

Here, if w on $\varphi(D_2)$ is chosen as

$$w = \exp(r_2)$$

then the first term of (31) can be written as

$$\left| \int_{\varphi(D_2)} g(w) \sum_{j=1}^{\infty} \left(\frac{\exp(r)}{w} \right)^{jn+1} dw \right| = 2\pi \exp(r_2) \max_{w \in \varphi(D_2)} |g(w)| \left(\frac{\exp(r - r_2)^{n+1}}{1 - \exp(r - r_2)^n} \right) \tag{32}$$

$$= 2\pi \exp(r) \max_{z \in D_2} |f(z)| \left(\frac{\exp(-n(r_2 - r))}{1 - \exp(-n(r_2 - r))} \right). \tag{33}$$

In the same way, if w on $\varphi(D_1)$ is chosen as

$$w = \exp(r_1)$$

then the second term of (31) can be written as

$$\left| \int_{\varphi(D_1)} g(w) \sum_{j=2}^{\infty} \left(\frac{\exp(r)}{w} \right)^{1-jn} dw \right| = 2\pi \exp(r_1) \max_{w \in \varphi(D_1)} |g(w)| \left(\frac{\exp(r_1 - r)^{2n-1}}{1 - \exp(r_1 - r)^n} \right) \tag{34}$$

$$= 2\pi \exp(r) \max_{z \in D_1} |f(z)| \left(\frac{\exp(-2n(r - r_1))}{1 - \exp(-n(r - r_1))} \right). \tag{35}$$

Combining these results, it turns out that the required statement is proved. □

2.3. A Priori Error Algorithm for Function References

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

To avoid these problems, we adopt an algorithm of calculating a priori error bounds of function evaluations using floating-point computations.This algorithm calculates a global constant ε for any $a \leq x \leq b$ s.t.

$$\max_{a \leq x \leq b} |\mathrm{res} - f(x)| \leq \varepsilon,$$

which res denotes the approximate value of $f(x)$. In the case that some numerical algorithm computes the same function with a number of different points, we can expect the algorithm with the a priori error algorithm to become faster than that with interval arithmetic, because the evaluations of the function are executed by pure floating-point operations.

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

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

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

$$|\tilde{z} - g(\tilde{x}, \tilde{y})| \leq |g(\tilde{x}, \tilde{y})| \, \varepsilon_M. \tag{36}$$

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

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

Here, let us suppose the interval I_z holds

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

and intervals D_x, D_y hold

$$D_x \supset \left\{ \frac{\partial g}{\partial x}(x, y) \mid x \in I_x, y \in I_y \right\} \tag{37}$$

$$D_y \supset \left\{ \frac{\partial g}{\partial y}(x, y) \mid x \in I_x, y \in I_y \right\}. \tag{38}$$

We make the pair (I, ε) as

$$I : \text{An input interval into the operation} \tag{39}$$

$$\varepsilon : \text{Collected errors until the operation,} \tag{40}$$

and define every operation for the pair.

For example, the addition operator "+" for the pair is defined by

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

To similar, the multiplication operator "$\cdot$" is defined by

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

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 1
Computation of an a priori error algorithm of rounding errors when evaluating $f(\xi)$ in floating-point arithmetic ($a \leq \xi \leq b,\ \xi \in \mathbb{F}$).

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

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.*

Remark 1
In IEEE standard 754 double precision with rounding-to-nearest mode,

$$\varepsilon_x = \varepsilon_y = 2^{-53},$$

and for the operators of addition, subtraction, multiplication, division and square root,

$$\varepsilon_M = 2^{-53}.$$

*When we build a program based on this algorithm, we have to use a software satisfying (36)for all inputted floating point numbers on every function. Unfortunately, some free mathematical libraries do NOT satisfy these inequality for all inputted floating point numbers on every function. CRlibm software (*CRlibm, 2006*) developed by J.Muller, F.Dinechin and others is designed to satisfy these inequalities, so that in numerical results of this paper we use this library.*

3. Verified Automatic Contour Integration Algorithm

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

Algorithm 2
Assume that a relative tolerance of numerical integration error ε_{rel} is given. A verified automatic integration algorithm outputs an interval $\tilde{I}$ satisfying

$$\left| \frac{I - \tilde{I}}{I} \right| \leqq \varepsilon_{rel}.$$

Step 1 *Get the order of the true value by verified computation.*

Step 2 *Rewrite inputted relative tolerance ε_{rel} to absolute tolerance ε_{abs}.*

Step 3 *Calculate K.*

Step 4 *Based on Theorem 1, calculate n satisfying $|E_n| < \varepsilon_{abs}$.*

Step 5 *Calculate an upper bound of function reference error $|E_r|$ using Algorithm 1.*

Step 6 *Calculate the integral value s by the trapezoidal rule using and n with pure floating-point numbers.*

Step 7 *Output the interval* $[Re(s) - |E_r|, Re(s) + |E_r|] + [Im(s) - |E_r|, Im(s) + |E_r|]i$.

4. Numerical Result

In this section, we present the numerical experiments. These experiments have been done under the following computer environment: Linux (Fedora8), Memory 8GB, Intel Core 2 Extreme 3.0GHz (Use 1 Core Only), GCC 4.1.2 with CRlibm 1.0 beta(CRlibm, 2006).

Here, we compare the following two algorithms on automatic integration:

(A) (Approximate) Automatic integration algorithm based on trapezoidal rule.

(B) (Verified) Proposed algorithm (Algorithm 2)

Example 1

$$I_1 = \int_{|z|=2} \frac{z\exp(z)}{(z-0.5i)^2(z+0.5)^2}dz, \quad r_1 = 0.8, r_2 = 3.2$$

Example 2

$$I_2 = \int_{|z-1|=4} \frac{z\exp(z)}{(z-0.5i)^2(z+0.5)^3(z+i)^2}dz, \quad r_1 = 3, r_2 = 5$$

We present a comparison of the execution time of (A) and (B) for I_1 and I_2 when the tolerance has become tighter gradually on Figure 1 and 2 respectively.

It can be seen form these figures that the execution time of (A) are almost the same compared with that of (B). From the results, we found that in both examples our verified algorithm is in fact comparable to the approximation algorithm in terms of the computational time.

References

Takahashi, H. and Mori, M. Double Exponential Formulas for Numerical Integration. Publ. RIMS, Kyoto Univ, 9 (1974), 721–741.

Eiermann, M. C. Automatic, Guaranteed integration of analytic functions. BIT 29 (1989) 270–282.

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

Rabinowitz, P. Practical error coefficients in the integration of periodic analytic functions by the trapezoidal rule. In *Communications of the ACM*, v.11 n.11, pp.764-765, Nov. 1968.

Hirayama, H. Numerical calculation of contour integration on complex field. JPSJ 38, pp 55–56. (Japanese)

Correctly Rounded mathematical library : `http://lipforge.ens-lyon.fr/www/crlibm/`

R. Piessens et. al., QUADPACK: A Subroutine Package for Automatic Integration, Springer, 1983.

N. Yamanaka, T. Okayama, S. Oishi, T. Ogita: A fast verified automatic integration algorithm using double exponential formula, RIMS Kokyuroku, No.1638, 146-158, 2009.

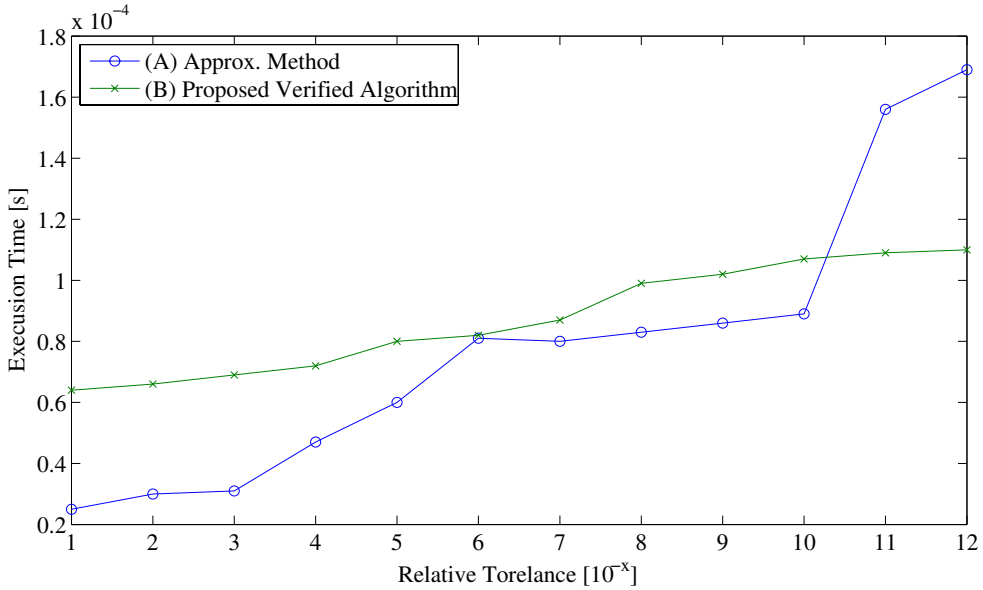

Figure 1. Numerical result of Example 1.

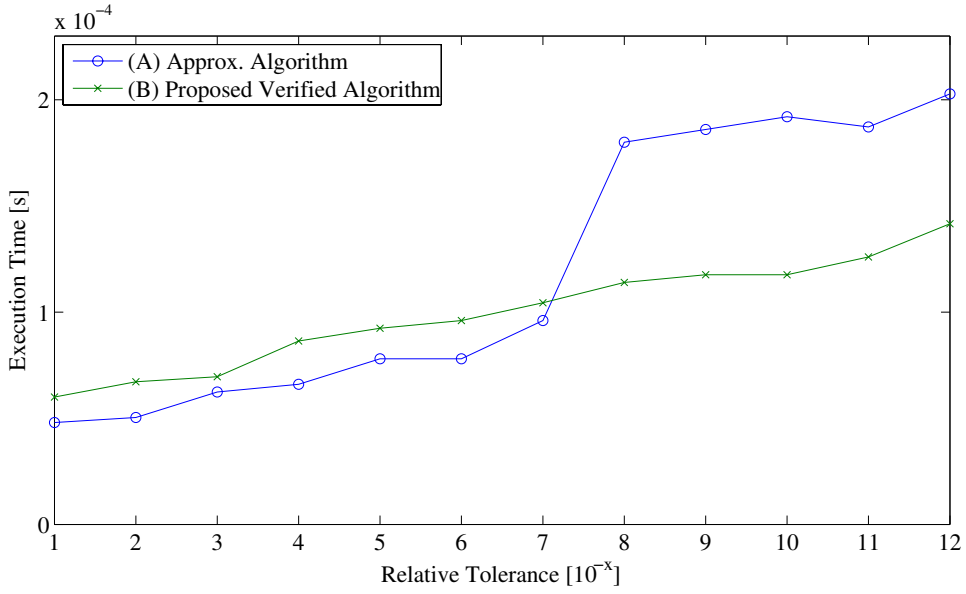

Figure 2. Numerical result of Example 2.