<table>
<thead>
<tr>
<th>Title</th>
<th>A Fast Verified Automatic Integration Algorithm using Double Exponential Formula (Numerical Analysis: Theory, Methods and Applications)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Yamanaka, Naoya; Okayama, Tomoaki; Oishi, Shin’ichi; Ogita, Takeshi</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (2009), 1638: 146-158</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2009-04</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/140531">http://hdl.handle.net/2433/140531</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>publisher</td>
</tr>
</tbody>
</table>
A Fast Verified Automatic Integration Algorithm using Double Exponential Formula

Abstract

A fast verified automatic integration algorithm of calculating univariate integrals over finite interval using numerical computations is proposed. The proposed algorithm is applicable to the double exponential formula for numerical integration proposed by H. Takahashi and M. Mori. To get highly accurate integral value using the formula, how to decide the two parameters $h$ and $n$ is critical. In this paper, we present a theorem to get an adequate pair $h$ and $n$ for a given tolerance. Furthermore, we propose a fast verified automatic integration algorithm based on the theorem with a priori error algorithm for rounding error. Numerical results are presented showing the performance of the proposed algorithm.

1 Introduction

We are concerned with verified computation of an integral

$$I = \int_{a}^{b} f(x) dx.$$

We assume that $f(x)$ may have an integrable singularity at the end-points $x = a$ and/or $x = b$. We adopt the double exponential formula for numerical integration proposed by H. Takahashi and M. Mori [2] in 1974. The double exponential transformation $\varphi$ for the integral over finite interval $(a, b)$ is defined by

$$x = \varphi(t) = \frac{b-a}{2} \tanh \left( \frac{\pi}{2} \sinh(t) \right) + \frac{a+b}{2},$$

and then the double exponential formula can be written as follows:

$$I_{h,n} := h \sum_{k=-n}^{n} f(\varphi(kh)) \varphi'(kh),$$
where $h$ and $n$ denote a mesh size and the number of points, respectively. Here, the error $E(h, n)$ can be described as

$$I = I_{h,n} + E(h, n)$$

$$E(h, n) = \tilde{C}(E_D(h) + E_T(h, n)),$$

which $E_D(h)$ and $E_T(h, n)$ denote the discretization error and the truncation error with a constant $\tilde{C}$. In order that the formula works accurately and efficiently, $E_D$ and $E_T$ should be of the same order in magnitude, so that we attempt to select the pair $h$ and $n$ adequately.

Several authors have proposed the error analysis of the double exponential formula [3, 4, 5]. In particular, for verified computation, a theorem (Theorem 1), which is proposed by Okayama et al. [6], is useful to get an upper bound of $E(h, n)$. We think that this theorem is much important because it shows some constants of the upper bound of the formula explicitly, so that it can easily be applied for verified integration algorithms. Using the theorem, however, these errors $E_D$ and $E_T$ are sometimes not of the same order since the pair $h$ and $n$ is not suitably selected.

To solve the problem, in this paper, we present a theorem (Theorem 3). Using this theorem, we can easily get an adequate pair satisfying

$$E(h, n) \leq \varepsilon_{abs},$$

which $\varepsilon_{abs}$ denotes a given absolute tolerance.

For developing verified algorithm using the double exponential formula, all rounding errors that occur throughout the algorithm must be taken into account. An upper bound of rounding error for verified computations can be calculated by interval arithmetic, but it is much slower than pure floating-point arithmetic because it calculates the interval of evaluation $\text{res}_i$ and the upper bound of rounding error $\varepsilon_i$ for every $x_i$ s.t.

$$|\text{res}_i - f(x_i)| \leq \varepsilon_i.$$  

Furthermore, to get the upper bound, whole calculations by interval arithmetic must be completed. To avoid these problems, we adopt an algorithm of calculating a priori error bounds of function evaluations using floating-point computations (Algorithm 1). This algorithm calculates a global constant $\varepsilon$ for any floating point numbers $a \leq x \leq b$ s.t.

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

which $\text{res}$ denotes the approximate value of $f(x)$.

Automatic integration in this paper means that the user inputs an integration and a relative tolerance $\varepsilon_{rel}$, and automatic integration algorithm outputs an interval $\tilde{I}$ s.t.

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

where $\text{rad}(\tilde{I})$ denotes the radius of $\tilde{I}$.

In this paper, we propose a fast verified automatic integration algorithm based on Theorem 3 and Algorithm 1. Using this algorithm, we can calculate $\tilde{I}$ as fast as the widely-used approximation software developed by Ooura [10] for integration problems using the double exponential formula.
2 Errors of The Double Exponential Formula

2.1 The Double Exponential Formula

Now consider again the integral
\[ I = \int_{a}^{b} f(x)dx. \]  \hspace{1cm} (1)

We assume that \( f(x) \) may have an integrable singularity at the end-points \( x = a \) and/or \( x = b \). We apply a variable transformation
\[ x = \varphi(u) \]
to (1), where \( \varphi(u) \) is an analytic increasing function without singularities on the real axis except infinity satisfying
\[ a = \varphi(-\infty), \quad b = \varphi(+\infty). \]

Then we have
\[ I = \int_{-\infty}^{\infty} f(\varphi(u))\varphi'(u)du, \]  \hspace{1cm} (2)
in which \( f(\varphi(u))\varphi'(u) \) has no singularities on the real axis except infinity. We apply the trapezoidal rule to (2) with a mesh size \( h \),
\[ I_{h} := h \sum_{k=-\infty}^{\infty} f(\varphi(kh))\varphi'(kh). \]

Obviously, the infinite sum must be appropriately truncated in actual application; e.g.
\[ I_{h,n} := h \sum_{k=-n}^{n} f(\varphi(kh))\varphi'(kh). \]

In order that the formula above works accurately, the transformed function by the double exponential transformation should be analytic and bounded on some strip domain,
\[ \mathcal{D}_{d} = \{ z \in \mathbb{C} : |\text{Im} z| < d \}, \]
for a positive constant \( d \). More specifically, the function before the transformation is subject to be non-singular on the following domain:
\[
\varphi(\mathcal{D}_{d}) = \left\{ z \in \mathbb{C} : \varphi^{-1}(z) \in \mathcal{D}_{d} \right\}
= \left\{ z \in \mathbb{C} : \left| \frac{1}{\pi} \log \left( \frac{z - a}{b - z} \right) + \sqrt{1 + \frac{1}{\pi} \log \left( \frac{z - a}{b - z} \right) \frac{1}{\pi} \log \left( \frac{z - a}{b - z} \right)^{2}} \right| < d \right\}.
\]

To be more specific, we define the following function space:

**Definition 1**

Let \( K, \alpha, \beta \) be positive constants. Then \( L_{K,\alpha,\beta}(\varphi(\mathcal{D}_{d})) \) denotes the family of all functions \( f \) that are holomorphic on \( \varphi(\mathcal{D}_{d}) \) for \( d \) with \( 0 < d < \pi/2 \), and satisfy the condition that
\[ |f(z)| \leq K |z - a|^{|\alpha-1}|b - z|^{|\beta-1}|, \]  \hspace{1cm} (3)
for all \( z \in \varphi(\mathcal{D}_{d}) \).

We can calculate the upper bound of \( K \) using circle interval arithmetic. See Eiermann [7].
2.2 Error Analysis of The Double Exponential Formula

For developing verified algorithm using the double exponential formula, all errors of the formula must be taken into account. Since the trapezoidal rule in the double exponential formula is used for transformed infinite interval, we have to consider the following two errors as the errors of the double exponential formula:

1. The Discretization error $E_D(h)$
2. The Truncation error $E_T(h, n)$

For estimating the discretization error, a lemma proposed by Okayama et al. can be used.

Lemma 1 (Okayama et al. [6, Lemma 4.16])
Let $f \in L_{K, \alpha, \beta}(\varphi(\mathcal{D}_d))$ and $\mu = \min \{\alpha, \beta\}$. Then

\[
|E_D| = \left| \int_{a}^{b} f(x)dx - h \sum_{j=-\infty}^{\infty} f(\varphi(jh)) \varphi'(jh) \right|
\leq C_1 C_2 \frac{e^{-2\pi d/h}}{1 - e^{-2\pi d/h}},
\]

where the constants $C_1$ and $C_2$ are defined by

\[
C_1 = \frac{2K(b-a)^{\alpha+\beta-1}}{\mu} \quad \text{(4)}
\]

\[
C_2 = \frac{2}{\cos^{\alpha+\beta}(\frac{\pi}{2} \sin d) \cos d} \quad \text{(5)}
\]

Next, the following lemma proposed by Okayama et al. can be used for the estimation of the truncation error:

Lemma 2 (Okayama et al. [6, Lemma 4.17])
Assume that the assumptions of Lemma 1 are fulfilled. Furthermore let $\nu = \max \{\alpha, \beta\}$, $n$ be a positive integer, and $M$ and $N$ be positive integers defined by

\[
\begin{cases}
M = n, & N = n - \lfloor \log(\beta/\alpha)/h \rfloor \quad \text{(if } \mu = \alpha) \\
N = n, & M = n - \lfloor \log(\alpha/\beta)/h \rfloor \quad \text{(if } \mu = \beta)
\end{cases}
\]

Then, it follows that

\[
|E_T| \leq \left| h \sum_{j=-\infty}^{M-1} f(\varphi(jh)) \varphi'(jh) \right| + \left| h \sum_{j=N+1}^{\infty} f(\varphi(jh)) \varphi'(jh) \right|
\leq e^{\pi \nu} C_1 e^{-\mu \exp(nh)},
\]

where the constant $C_1$ is defined by (4).

Using the above two lemmas, Okayama et al. has proposed the following theorem:
Theorem 1 (Okayama et al. [6, Theorem 2.11])
Let $f \in L_{K, \alpha, \beta}(\varphi(\mathcal{D}_d))$, $\mu = \min\{\alpha, \beta\}$, $\nu = \max\{\alpha, \beta\}$, $n$ be a positive integer with $n \geq (\nu e)/(4d)$, and $h$ be selected by
$$h = p(n) = \frac{\log(4dn/\mu)}{n}.$$
(7)
Furthermore, let $M$ and $N$ be positive integers defined by (6). Then it follows that
$$\left| \int_a^b f(x)dx - h \sum_{k=-M}^N f(\varphi(kh)) \varphi'(kh) \right| \leq |E_D| + |E_T|$$
$$\leq C_1 \left[ \frac{C_2}{1 - e^{-\frac{2\pi d}{\nu \mu}}} + e^{\frac{\nu}{2}} \right] e^{-\frac{2\pi dn}{\log(4dn/\mu)}},$$
where the constants $C_1$ and $C_2$ are defined by (4) and (5).

We think that this theorem is much important because it shows some constants of the upper bound of the formula explicitly, so that it can easily be applied for verified integration algorithms. Using the theorem, however, these errors $E_D$ and $E_T$ are sometimes not of the same order since the pair $h$ and $n$ is not suitably selected.

In addition, though Theorem 1 requires $n$ first to determine $h$ from $n$, it does not describe how to select $n$.

To solve these problems, we present two theorems as follows:

Theorem 2
Let $f \in L_{K, \alpha, \beta}(\varphi(\mathcal{D}_d))$, $\mu = \min\{\alpha, \beta\}$, $\nu = \max\{\alpha, \beta\}$, $h$ be a positive number, and $n$ be selected by
$$n = q(h) = \left\lceil \frac{1}{h} \log \left( \frac{4d}{\mu h} - \frac{2}{\pi \mu} \log e \left( \frac{C_2}{e^{\frac{\nu}{2}}} \right) \right) \right\rceil.$$
(8)
Furthermore, let $M$ and $N$ be positive integers defined by (6). Then it follows that
$$\left| \int_a^b f(x)dx - h \sum_{k=-M}^N f(\varphi(kh)) \varphi'(kh) \right| \leq |E_D| + |E_T|$$
$$\leq 2C_1C_2 \frac{e^{-2\pi d/h}}{1 - e^{-2\pi d/h}},$$
where the constants $C_1$ and $C_2$ are defined by (4) and (5).

Proof.
Clearly it follows by Lemma 1 and 2 that
$$|E_D| + |E_T| \leq C_1 \left( C_2 \frac{e^{-2\pi d/h}}{1 - e^{-2\pi d/h}} + e^{\frac{\nu}{2}} e^{-\frac{\nu}{2} \mu \exp(nh)} \right) \leq C_1 (\hat{E}_D + \hat{E}_T),$$
where $\hat{E}_D$ and $\hat{E}_T$ are defined by
$$\hat{E}_D = C_2 \frac{e^{-2\pi d/h}}{1 - e^{-2\pi d/h}}$$
$$\hat{E}_T = e^{\frac{\nu}{2}} e^{-\frac{\nu}{2} \mu \exp(nh)}.$$
(9)
For decreasing $|E_D|$ and $|E_T|$ at the same order, using $e^{-2\pi d/h} \ll 1$ when $h$ is small, suppose
\[ C_2 e^{-2\pi d/h} = \bar{E}_T. \tag{11} \]

(11) can be rewritten by $n$ as follows:
\[
e^{-\frac{\pi}{2}\mu \exp(nh)} = \frac{C_2}{e^{\frac{\mu}{2}\nu}} e^{-2\pi d/h}
\]
\[
\iff -\frac{\pi}{2\mu} \exp(nh) = \log_e \left( \frac{C_2}{e^{\frac{\mu}{2}\nu}} e^{-2\pi d/h} \right)
\]
\[
= \log_e \left( \frac{C_2}{e^{\frac{\mu}{2}\nu}} \right) - \frac{2\pi d}{h}
\]
\[
\iff n = \frac{1}{h} \log \left( \frac{4d}{\mu h} - \frac{2}{\pi\mu} \log_e \left( \frac{C_2}{e^{\frac{\mu}{2}\nu}} \right) \right)
\]
\[
\leq \left\lfloor \frac{1}{h} \log \left( \frac{4d}{\mu h} - \frac{2}{\pi\mu} \log_e \left( \frac{C_2}{e^{\frac{\mu}{2}\nu}} \right) \right) \right\rfloor.
\]

Using (11), the upper bound of the double exponential formula can be written as follows:
\[
|E_D| + |E_T| \leq C_1 \left( \bar{E}_D + \bar{E}_T \right)
\]
\[
= C_1 C_2 \left( \frac{e^{-2\pi d/h} + e^{-2\pi d/h}}{1 - e^{-2\pi d/h}} \right)
\]
\[
= C_1 C_2 e^{-2\pi d/h} \left( \frac{2 - e^{-2\pi d/h}}{1 - e^{-2\pi d/h}} \right)
\]
\[
\leq 2C_1 C_2 \frac{e^{-2\pi d/h}}{1 - e^{-2\pi d/h}}
\]

Theorem 2 shows that when $E_D \approx E_T$, the upper bound of the double exponential formula depends only on $h$. The following theorem shows the condition of the pair $h$ and $n$ when a tolerance $\epsilon_{abs}$ is inputted:

**Theorem 3**

Let $f \in L_{K,\alpha,\beta}(\varphi(\mathcal{D}_d))$, $\mu = \min\{\alpha, \beta\}$ and $\nu = \max\{\alpha, \beta\}$. Furthermore, let $M$ and $N$ be positive integers defined by (6), and the constants $C_1$ and $C_2$ be defined by (4) and (5). If a positive number $h$ and a positive integer $n$ are selected by
\[
h = \frac{2\pi d}{\log_e \left( 1 + \frac{2C_2}{\epsilon_{abs}} \right)},
\]
\[
n = \left\lfloor \frac{1}{h} \log \left( \frac{2}{\pi\mu} \log_e \left( \frac{2e^{\frac{\nu}{2}}}{\epsilon_{abs}} \right) \right) \right\rfloor,
\]
then
\[
\left| \int_a^b f(x)dx - h \sum_{k=-M}^{N} f(\varphi(kh)) \varphi'(kh) \right| \leq C_1 \epsilon_{abs}
\]
holds.
Proof.
It follows by Lemma 1 and 2 that

\[ |E_D(h)| + |E_T(h, n)| \leq C_1 \left( \tilde{E}_D + \tilde{E}_T \right). \]

For conforming \(|E_D|\) to \(|E_T|\), assign \(\varepsilon_{\text{abs}}\) as follows:

\begin{align*}
\tilde{E}_D(h) &\leq \frac{\varepsilon_{\text{abs}}}{2}, \quad \text{(12)} \\
\tilde{E}_T(h, n) &\leq \frac{\varepsilon_{\text{abs}}}{2}. \quad \text{(13)}
\end{align*}

First, since \(\tilde{E}_D(h)\) depends on only \(h\), from (12) we have

\[ C_2 \frac{e^{-2\pi d/h}}{1 - e^{-2\pi d/h}} \leq \frac{\varepsilon_{\text{abs}}}{2} \]

\[ \iff e^{-2\pi d/h} \leq \frac{\varepsilon_{\text{abs}}}{2C_2 + \varepsilon_{\text{abs}}} \]

\[ \iff h \leq \frac{2\pi d}{\log_e \left( 1 + \frac{2C_2}{\varepsilon_{\text{abs}}} \right)} =: h_m. \]

Next, though \(\tilde{E}_T(h, n)\) depends on \(h\) and \(n\), we can choose adequate \(n\) from (13) using maximum \(h (= h_m)\) as follows:

\[ e^{\frac{2}{3} \nu} e^{-\frac{\pi}{2} \mu \exp(nh_m)} \leq \frac{\varepsilon_{\text{abs}}}{2} \]

\[ \iff -\frac{\pi}{2} \mu \exp(nh_m) \leq \log_e \left( \frac{2e^{\frac{2}{3} \nu}}{\varepsilon_{\text{abs}}} \right) \]

\[ \iff n \geq \frac{1}{h_m} \log_e \left( \frac{2}{\pi \mu} \log_e \left( \frac{2e^{\frac{2}{3} \nu}}{\varepsilon_{\text{abs}}} \right) \right) \]

Since the right-hand side is monotone decreasing for \(h\), we have

\[ n = \left[ \frac{1}{h_m} \log_e \left( \frac{2}{\pi \mu} \log_e \left( \frac{2e^{\frac{2}{3} \nu}}{\varepsilon_{\text{abs}}} \right) \right) \right]. \]

\[ \square \]

3 Fast Verified Automatic Integration Algorithm

3.1 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 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, which is proposed by M. Kashiwagi [8]. This algorithm calculates a global constant \(\varepsilon\) for any \(a \leq x \leq b\) s.t.

\[ \max_{a \leq x \leq b} |\text{res} - f(x)| \leq \varepsilon, \]
which \( \text{res} \) 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:

\[
|\tilde{z} - g(\tilde{x}, \tilde{y})| \leq |g(\tilde{x}, \tilde{y})|\varepsilon_M.
\] (14)

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

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

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

For the single operation \( z = g(x) \), we can show the upper bound of rounding error by the almost same way. Denote \( \tilde{x} \) in the intervals \( I_x \) by approximate values of \( x \) in \( I_x \). Suppose

\[
|x - \tilde{x}| \leq \varepsilon_x
\]

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

\[
|\tilde{z} - g(\tilde{x})| \leq |g(\tilde{x})|\varepsilon_M.
\] (15)

Then, the following inequality holds for \( z \in I_x \):

\[
|z - \tilde{z}| \leq |D_x|\varepsilon_x + |I_x|\varepsilon_M.
\]

Here, let us suppose the interval \( I_x \) holds

\[
I_x \supset \{g(x) \mid x \in I_x\},
\]

and intervals \( D_x \) hold

\[
D_x \supset \{g'(x) \mid x \in I_x\}.
\]

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

\[
I : \text{An input interval into the operation}
\]

\[
\varepsilon : \text{Collected errors until the operation},
\]
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 "." 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 \( \varepsilon_y \) 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 (14) and (15) 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 [9] 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.

### 3.2 Proposed Algorithm

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

**Algorithm 2**

*A verified automatic integration algorithm outputs an interval \( \tilde{I} \) satisfying*

\[ \left| \frac{I - \tilde{I}}{I} \right| \leq \varepsilon_{rel} \]

*when user inputs the integral (1) and relative tolerance \( \varepsilon_{rel} \).*

**Step 1** Choose \( d \) and calculate \( K \).
Step 2 Get the order of the true value by verified computation.

Step 3 Rewrite inputted relative tolerance $\varepsilon_{rel}$ to absolute tolerance $\varepsilon_{abs}$.

Step 4 Calculate an upper bound of rounding error $|E_r|$ using Algorithm 1.

Step 5 If $|E_r| < \varepsilon_{abs}$, get $h$ and $n$ for $\varepsilon = \varepsilon_{abs} - |E_r|$ by Theorem 3.

Step 6 Calculate the integral value $s$ by the double exponential formula using $h$ and $n$ with pure floating-point numbers.

Step 7 Output the interval $[s - |E_r|, s + |E_r|]$.

4 Numerical Results

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 is used to satisfy (14) and (15).)

4.1 Comparison for Estimating Rounding Error

We shall present numerical results that Algorithm 1 and interval arithmetic are applied for the following two examples for comparison in terms of the execution time and the upper bound of rounding error.

Example 1

$$f_1(x) = x^{25} + x^{24} + \cdots + x + 1$$

Example 2

$$f_2(x) = \sin(\exp(x))$$

First, we present a comparison of the execution time of Algorithm 1 and interval arithmetic for $f_1(x)$ and $f_2(x)$ when the number of points has increased.

Left figure of Figure 1 shows the ratios of the execution time $t_i/t_p$, which $t_p$ and $t_i$ denote the execution time of Algorithm 1 and that of interval arithmetic.

This figure shows that because $t_i$ depends on the number of points and $t_p$ does not depend on that, the ratios between them have increased exponentially.

Next, we show a comparison of the upper bound of rounding error when the number of points has increased. Right figure of Figure 1 shows that the ratios of the number of points $e_p/e_i$, which $e_p$ and $e_i$ denote the upper bound of rounding error calculated by Algorithm 1 and that by interval arithmetic, respectively.

This figure shows that $e_p$ is about 30 times larger than $e_i$ for $f_1(x)$, and $e_p$ is almost the same as $e_i$ for $f_2(x)$.
4.2 Comparison between Two Theorems

In this section, we also present numerical results that the comparison between Theorem 1 and 2.

The main difference between two theorems is the relation between $n$ and $h$: Theorem 1 uses (7) and Theorem 2 uses (8).

In this numerical experiment, we'd like to compare the ratio of the order between $\tilde{E}_D$ and $\tilde{E}_T$ defined by (9) and (10), when the user input an absolute tolerance $\varepsilon_{abs}$. Figure 2 shows the ratio of the order of each errors as

$$\log_{10} \left( \frac{\tilde{E}_D}{\tilde{E}_T} \right),$$

when $d = 1.0$ and $\alpha = \beta = 1$.

This figure displays the errors of Theorem 1 don't decrease at the same order in magnitude, but that of Theorem 2 keep in step.

4.3 Comparison on Automatic Integration

We compare the following three algorithms on automatic integration:
(A) (Verified) Proposed algorithm (Algorithm 2)

(B) (Verified) An algorithm consists of interval arithmetic and Theorem 2

(C) (Approximate) Automatic integration software developed by T. Ooura [10]:

Example 1

\[ I_1 = \int_{-1}^{1} f_1(x) \, dx \]

Example 2

\[ I_2 = \int_{0}^{1} \frac{f_2(x)}{\sqrt{x}} \, dx \]

We show a comparison of the execution time of (A)-(C) for \( I_1 \) and \( I_2 \) when the relative tolerance has become tighter gradually on left figure and right figure of Figure 3 respectively. These figures show that (A) is 3-10 times faster than (B) and (A) can calculate at almost the same speed compared with (B).

![Figure 3: Ratio of the execution time](image)

5 Conclusion

We presented verified automatic integration algorithms using the double exponential formula. The proposed algorithm is designed for the discretization error and the truncation error to decrease at the same order in magnitude. From the numerical results, we confirm that the proposed algorithm tends to be as fast as the widely-used approximation software developed by Ooura.

References


[9] Correctly Rounded mathematical library
http://lipforge.ens-lyon.fr/www/crlibm/

[10] Ooura's Mathematical Software Packages
http://www.kurims.kyoto-u.ac.jp/~ooura/intde.html