Vol.2012-ARC-202 No.20
Vol.2012-HPC-137 No.20
2012/12/14

Euler-Maclaurinの総和公式を利用した数値積分の性能

平　山　　弘[†1]

Euler-Maclaurin の総和公式[1] とは、積分とその積分を台形公式を使って計算した値との誤差評価式と見なせる。この誤差評価式には関数の微分係数が含まれているため、これまでこの公式を使って、数値積分計算が行われることはほとんどなかった。この微分係数の問題を解決するため自動微分の一種である Taylor 展開法を使うと、精度良く微分係数を計算出来る。これを利用すれば、有効な数値積分公式を導くことができると期待できる。

本論文では、微分係数を精度良く出来る Taylor 展開法を利用して、Euler-Maclaurin の総和公式を利用して数値積分を行うと、他の有力な数値積分法と同等程度の数値積分法となることを示す。Taylor 展開法は、見かけ上の特異性を持つ関数の特異点での関数値を精度良く計算出来るので、見かけ上の特異点をもつ関数に対する数値積分を精度良く計算出来る特徴を持つ。

Performance of the Numerical Integration by Euler-Maclaurin summation formula

Hiroshi Hirayama[†1]

Euler-Maclaurin summation formula[1] can be regarded as error evaluation with the value of integration and the value of the numerical integration by the trapezoidal rule. Since the differential coefficients of the function were contained in the error evaluation formula, numerical integration was not performed until now using this.

If the Taylor series method which is a kind of automatic differentiation is used in order to solve this problem, the differential coefficients are calculable with sufficient accuracy. If this method is used, it can be expected that an effective numerical-integration formula can be given.

In this paper, it is shown that Euler-Maclaurin summation formula with the Taylor series method which give the accuracy differential coefficients becomes an effective numerical integration method as same as other leading numerical integration and equivalent grades. Since the Taylor series method can calculate the value of a function in the singular point of a function with the singularity on appearance with sufficient accuracy, it has the feature which can calculate the numerical integration to a function with the apparent singular with sufficient accuracy.

1. はじめに

Euler-Maclaurin の総和公式は、周期関数の一周期に渡る積分が高精度で計算できることや、積分の積分区間の両端で関数値や微分係数が 0 になる場合、台形公式を使うと高精度で計算できることを説明するためによく使われる公式である。

この公式には、被積分関数の高階微分係数を含むため、実際の数値積分に使われることは今までほとんどなかった。高階微分係数を差分法で計算すると、桁落ちが生じ、精度の良く計算ができない。このため、これらの公式が積分計算に使われることがほとんどなかった。

関数の微分係数を計算するために差分を使わない方法として自動微分法[12] が知られている。この方法と原理的には同じ方法である Taylor 展開法もある。これを使えば、高精度で微分係数が計算できるので、Euler-Maclaurin の総和公式を使えば、数値積分を容易に計算できる。

逆に、台形公式の計算部分を級数と見なし、積分部分が解析的にできるか何等かの方法で簡単に計算できる場合積分の値を微分を含む部分で補正することによって、収束の遅い無限級数が計算できる。このような研究は長田[11] 等によって行われている。この場合、微分係数の計算に、数式処理システムが使われている。微分係数の計算に数式処理システムを使う場合、途中に人間が介入することになるので、少し大きな問題になると、かなり大きな作業になる。また途中で人的な誤りが入る可能性が生じる。計算の状況に応じて、微分係数等を計算するような場合は、実際上不可能になる。

2. Euler-Maclaurin の総和公式

数値積分や級数和の計算でよく使われる公式に Euler-Maclaurin の総和公式がある。この公式はいろいろな文献[2][4][13][14] 等で紹介されている。ここでは主に長田[10] を参考にし、記号を一部同じにした。

†1 神奈川工科大学
Kanagawa Institute of Technology

Vol.2012-ARC-202 No.20
Vol.2012-HPC-137 No.20
2012/12/14

関数 $f(x)$ は区間 $[a.b]$ で連続で微分可能とする。積分区間 $[a,b]$ を n 等分して、台形公式を適用した計算値と厳密な積分値との差を微分係数を含む m 項で近似すると、次の式が成り立つ。

$$h\left\{\frac{1}{2}f(a)+\sum_{k=1}^{n-1}f(a+kh)+\frac{1}{2}f(b)\right\}-\int_a^b f(x)dx \tag{1}$$
$$=\sum_{k=1}^{m}\frac{B_{2k}}{(2k)!}h^{2k}\left\{f^{(2k-1)}(b)-f^{(2k-1)}(a)\right\}+R_m$$

ここで、B_n は Bernoulli 数 (Bernoulli Number)、$h=\dfrac{b-a}{n}$ であり、

$$|R_m|\le\frac{h^{2m+2}\,|B_{2m+2}|}{(2m+2)!}\int_a^b\left|f^{(2m+2)}(t)\right|dt \tag{2}$$

である。

2.1 Bernoulli の多項式と Bernoulli の数

Bernoulli の多項式 $B_n(t)$ を次のように定義する。

$$\frac{xe^{tx}}{e^x-1}=\sum_{k=0}^{\infty}B_k(t)\frac{x^k}{k!} \tag{3}$$

(3) の両辺に e^x-1 の Taylor 展開式を掛け、展開すると

$$\sum_{k=1}^{\infty}\frac{t^{k-1}x^k}{(k-1)!}=\left(\sum_{k=1}^{\infty}\frac{x^k}{k!}\right)\left(\sum_{k=0}^{\infty}B_k(t)\frac{x^k}{k!}\right)$$
$$=\sum_{k=1}^{\infty}\left(\sum_{j=0}^{k-1}\left(\begin{array}{c}k\\j\end{array}\right)B_j(t)\right)\frac{x^k}{k!} \tag{4}$$

ここで、$\left(\begin{array}{c}k\\j\end{array}\right)=\dfrac{k!}{j!(k-j)!}$ である。(4) の式の x^k の係数を等しいと置くと

$$kt^{k-1}=\sum_{j=0}^{k-1}\left(\begin{array}{c}k\\j\end{array}\right)B_j(t) \tag{5}$$

(5) に $k=1$ を代入すると、$B_0(t)=1$、$k=2$ を代入すると

$2t=B_0(t)+2B_1(t)$

したがって、$B_1(t)=t-\frac{1}{2}$ となる。この操作を繰り返すと

$$B_2(t)=t^2-t+\frac{1}{6},\qquad B_3(t)=t^3-\frac{3}{2}t^2+\frac{1}{2}t,\qquad\cdots$$

が得られる。Bernoulli の多項式の定数部分を B_n と記述し、Bernoulli 数と呼ぶ。すなわち $B_n=B_n(0)$ である。$t=0$ を (5) に代入すると、次の Bernoulli 数の漸化式が得られる。

$$B_0=1,\qquad\sum_{j=0}^{k-1}\left(\begin{array}{c}k\\j\end{array}\right)B_j=0 \tag{6}$$

この式を使って、浮動小数点数で Bernoulli 数を計算すると、桁落ちが生じ、精度良い計算ができない悪条件の計算であることが知られている。

(3) に $t=0$ を代入すると Bernoulli 数 B_n を次のように定義することもできる。

$$\frac{x}{e^x-1}=\sum_{k=0}^{\infty}B_k\frac{x^k}{k!} \tag{7}$$

上の (7) の式に $\frac{x}{2}$ を加え、x に $-x$ を代入すると

$$\frac{(-x)}{e^{(-x)}-1}+\frac{(-x)}{2}=\frac{xe^x+x}{2(e^x-1)}=\frac{2x+x(e^x-1)}{2(e^x-1)}=\frac{x}{e^x-1}+\frac{x}{2} \tag{8}$$

となり、(8) の式は偶関数であることがわかる。したがって、x の奇数次の係数はゼロとなるから

$$\frac{x}{e^x-1}+\frac{x}{2}=\sum_{k=0}^{\infty}B_{2k}\frac{x^{2k}}{(2k)!} \tag{9}$$

となる。このことから、n が 3 以上の奇数のとき、$B_n=0$ である。

(3) の式に $t=1$ を代入すると

$$\frac{xe^x}{e^x-1}=\sum_{k=0}^{\infty}B_k(1)\frac{x^k}{k!} \tag{10}$$

となる。(10) 式は、(7) から、次のように変形できる。

$$\frac{xe^x}{e^x-1}=\frac{x}{1-e^{-x}}=\frac{(-x)}{e^{(-x)}-1}=\sum_{k=0}^{\infty}(-1)^kB_k\frac{x^k}{k!} \tag{11}$$

(10) と (11) の x の係数を比較すると、次の関係式が得られる。

Vol.2012-ARC-202 No.20
Vol.2012-HPC-137 No.20
2012/12/14

$$B_k(1) = (-1)^k B_k \tag{12}$$

B_k は k が奇数なら、$k = 1$ 以外ではゼロであり、偶数なら $B_k(1)$ と等しくなる。すなわち

$$B_{2k}(1) = B_{2k}(0) = B_{2k} \qquad (k = 0, 1, \cdots) \tag{13}$$
$$B_{2k-1}(1) = B_{2k-1}(0) = 0 \qquad (k \neq 1)$$

となる。

(3) の式の両辺を、t で微分すると

$$\frac{x^2 e^{tx}}{e^x - 1} = \sum_{k=0}^{\infty} \frac{B_k'(t)}{k!} x^k \tag{14}$$

(3) と (14) を比較することによって、次の関係式が得られる。

$$B_n'(t) = nB_{n-1}(t) \tag{15}$$

微分の公式 (15) を使うと、Bernoulli の多項式 $B_{2n}(x)$ を区間 $[0, 1]$ で、次の様に Fourier 級数に展開できる。

$$B_{2n}(x) = \frac{(-1)^{n-1} 2(2n)!}{(2\pi)^{2n}} \sum_{k=1}^{\infty} \frac{\cos 2k\pi x}{k^{2n}} \tag{16}$$

この Fourier 級数に $x = 0$ を代入すると

$$B_{2n} = \frac{(-1)^{n-1} 2(2n)!}{(2\pi)^{2n}} \sum_{k=1}^{\infty} \frac{1}{k^{2n}} \tag{17}$$

となる。$n > 1$ のとき、級数部分は 2 より小さいから、次の関係式が得られる。

$$|B_{2n}| < \frac{4(2n)!}{(2\pi)^{2n}} \tag{18}$$

(16) の Fourier 級数から、区間 $[0, 1]$ では、$x = 0$ の時が $B_n(x)$ の絶対値が最大値になることがわかる。すなわち

$$|B_{2n}(x)| \leq |B_{2n}| \qquad (0 \leq x \leq 1) \tag{19}$$

2.2 Euler-Maclaurin の総和公式の証明

次の積分 $I_{j,k}$ を定義する。

$$I_{j,k} = \frac{1}{(2k)!} \int_0^h B_{2k}\left(\frac{t}{h}\right) f^{(2k)}(x_j + t)dt$$

この式で $k = 1$ の場合を考える。この式は部分積分を使って次のように変換できる。

$$\begin{aligned} I_{j,1} &= \frac{1}{2!} \int_0^h B_2\left(\frac{t}{h}\right) f''(x_j + t)dt \\ &= \frac{1}{2!} \int_0^h \left(\frac{t^2}{h^2} - \frac{t}{h} + B_2\right) f''(x_j + t)dt \\ &= \frac{B_2}{2!} \left[f'(x_{j+1}) - f'(x_j)\right] - \frac{1}{2h} \left[f(x_{j+1}) + f(x_j)\right] \\ &+ \frac{1}{h^2} \int_{x_j}^{x_{j+1}} f(t)dt \end{aligned} \tag{20}$$

上の式 (21) を $j = 0, \cdots, n-1$ について加えると、次の式が得られる。

$$\sum_{j=0}^{n-1} I_{j,1} = \frac{B_2}{2!} \left[f'(b) - f'(a)\right] - \frac{1}{h} \left[\frac{1}{2} f(a) + \sum_{k=1}^{n-1} f(a + kh) + \frac{1}{2} f(b)\right] + \frac{1}{h^2} \int_a^b f(t)dt \tag{21}$$

(13) と (15) の関係式を利用すると

$$\begin{aligned} I_{j,k} &= \frac{1}{(2k)!} \int_0^h B_{2k}\left(\frac{t}{h}\right) f^{(2k)}(x_j + t)dt \\ &= \frac{1}{(2k)!} \left[B_{2k}\left(\frac{t}{h}\right) f^{(2k-1)}(x_j + t)\right]_0^h - \frac{1}{(2k-1)!h} \int_0^h B_{2k-1}\left(\frac{t}{h}\right) f^{(2k-1)}(x_j + t)dt \\ &= \frac{B_{2k}}{(2k)!} \left[f^{(2k-1)}(x_{j+1}) - f^{(2k-1)}(x_j)\right] + \frac{1}{h^2} I_{j,k-1} \end{aligned} \tag{22}$$

よって、次のようになる。

$$I_{j,k-1} = h^2 I_{j,k} - \frac{B_{2k} h^2}{(2k)!} \left[f^{(2k-1)}(x_{j+1}) - f^{(2k-1)}(x_j)\right] \tag{23}$$

(21) の $I_{j,1}$ に (23) の関係式を代入すると

$$\begin{aligned} &h \left[\frac{1}{2} f(a) + \sum_{k=1}^{n-1} f(a + kh) + \frac{1}{2} f(b)\right] - \int_a^b f(x)dx \\ &= \sum_{k=1}^{m+1} \frac{B_{2k} h^{2k}}{(2k)!} \left[f^{(2k-1)}(b) - f^{(2k-1)}(a)\right] + R_m \end{aligned} \tag{24}$$

を得る。ここで、

Vol.2012-ARC-202 No.20
Vol.2012-HPC-137 No.20
2012/12/14

$$R_m = -h^{2m+2}\sum_{j=0}^{n-1} I_{j,m+1}$$
$$= -\frac{h^{2m+2}}{(2m+2)!}\int_0^h B_{2m+2}\left(\frac{t}{h}\right)\sum_{j=0}^{n-1} f^{(2m+2)}(x_j+t)dt \quad (25)$$

である。(19) を使い上の式の R_m を評価すると以下のようになる。

$$|R_m| \le \frac{h^{2m+2}\,|B_{2m+2}|}{(2m+2)!}\int_a^b \left|f^{(2m+2)}(t)\right| dt \quad (26)$$

さらに (15) 式を適用すると、

$$|R_m| \le 4\left(\frac{h}{2\pi}\right)^{2m+2}\int_a^b \left|f^{(2m+2)}(t)\right| dt \quad (27)$$

ここで、等式が成り立つのは、右辺の積分が 0 のときである。多くの Euler-Maclaurin の総和公式の誤差評価は、(26) の式で与えているが、Bernulli 数の大きさがわかないため直感的にその大きさがわかりにくい。(27) の式は少し過大評価であるが、直感的に分かりやすくなっている。

この式からたとえ周期関数の一周期に渡る積分でも、微分係数が非常に大きくなる関数は、効率的に求められない事になる。たとえば

$$I = \int_0^{2\pi} \cos(50\sin x)dx \quad (28)$$

の場合、1 回微分する毎にだいたい 50 倍位の数値になる。このような場合、(27) の誤差評価からあまり急速に収束するようにはならないと推定できる。分割数 90 でほぼ 15 桁の精度が得られる。もし、積分が

$$I = \int_0^{2\pi} \cos(\sin x)dx \quad (29)$$

だったら、分割数 16 で 15 桁の精度が得られる。台形公式を使った場合、分割数を 2, 4, 8, 16 と増やして計算する場合が多いので分割数が偶数になることが多い。このためあまり気づかれていないが、分割数を奇数にすると原因は分からないが非常に高精度の結果が得られる。(28) の問題の場合、分割数が 9 でも 16 の場合よりも高精度の結果が得られる。分割数が 7 の場合でも 16 の場合とほぼ同精度の結果が得られる。(28) の問題の場合でも同様な結果が得られる。分割数が 90 の結果より 45 の場合が良い結果が得られる。

3. Euler-Maclaurin の総和公式による数値積分

Euler-Maclaurin の総和公式は次のように書ける。

$$\int_a^b f(x)dx = h\left\{\frac{1}{2}f(a) + \sum_{k=1}^{n-1} f(a+kh) + \frac{1}{2}f(b)\right\} \quad (30)$$
$$-\sum_{k=1}^{m}\frac{B_{2k}}{(2k)!}h^{2k}\left\{f^{(2k-1)}(b) - f^{(2k-1)}(a)\right\} - R_m$$

最初に、積分区間の端点において、Taylor 展開を計算する。この Taylor 展開を計算するプログラムは、通常の関数値の計算と宣言部分を除いてほぼ同じになる。この Taylor 展開を行うためのテンプレート・プログラムも公開されている[6] のその計算は容易である。テンプレート機能[3] を使えば Taylor 展開と関数値を計算するプログラムは多くの場合 1 個のプログラムで記述できる。実際この論文で示した計算例は、C++言語のテンプレート機能を使って 1 個の関数プログラムの形で記述した。たとえば、$0.92\cosh x - \cos x$ をプログラムで記述するには、次の様に書く。

```
template<typename T>
T func( const T& x )
{
      T  s ;
      s =0.92*cosh(x)-cos(x) ;
      return s ;
}
```

このように 1 個関数プログラムを書けば、関数計算や Taylor 展開の計算に使える。

次に分割数 n を決める。分割数がある程度推定できる場合、その値にする。何もなければ最小の 2 にする。この分割数を利用して、台形公式を利用して、積分の近似値を計算する。近似値を次の項を利用して、補正する。この級数は漸近級数なので漸近級数の手法を使って計算する。

$$c(k) = \frac{B_{2k}}{(2k)!}h^{2k}\left\{f^{(2k-1)}(b) - f^{(2k-1)}(a)\right\} \quad (31)$$

この補正項 $c(k)$ の絶対値が要求精度以下になるまで、補正項を減算して行く。補正項の絶対値が要求精度より小さくなったら、その項を使って補正し計算を止める。ここまでの計算

Vol.2012-ARC-202 No.20
Vol.2012-HPC-137 No.20
2012/12/14

値が求める積分値になる。補正項の絶対値が要求精度より小さくならない場合や逆に大きくなった場合、補正するのを止め、分割数 n を 2 倍にして、台形公式による積分の計算に戻る。分割数を 2 倍にすると、その前の台形公式計算値が利用し易くするためである。この手順を繰り返す事によって積分値を計算する。

3.1 簡単な計算例

簡単な例として、次の積分を計算する。これを要求精度 10^{-9} で計算する。

$$I = \int_0^1 \frac{1}{1+x} dx = \log 2 \tag{32}$$

最初に、積分区間の両端点で Taylor 展開をする。展開は 20 次まで行った。その計算結果を以下に示す。$x = 0$ における Taylor 展開を 15 次まで表示すると次の様になる。

$1 - x + x^2 - x^3 + x^4 - x^5 + x^6 - x^7 + x^8 - x^9 + x^{10} - x^{11} + x^{12} - x^{13} + x^{14} - x^{15}$

$x = 1$ における Taylor 展開を 5 次まで表示すると次の様になる。

$0.5 - 0.25(x-1) + 0.125(x-1)^2 - 0.0625(x-1)^3 + 0.03125(x-1)^4 - 0.015625(x-1)^5$

$n = 2$ として、台形公式で数値積分すると、0.708333333333333 となる。これを補正する。この場合の補正項は収束が非常に遅く要求精度を満たすことができない。$k = 8$ で逆に絶対値が増加し始める。このため分割数を 2 倍の $n = 4$ として、台形公式を使って数値積分を行う。このときの積分値は、0.697023809523809 となる。この結果を再度補正する。ここでのは補正項も十分速く小さくなるとは言えないが、$k = 7$ で補正項が 3.1×10^{-9} となり要求精度より小さくなった。そこまで計算値が積分値となる。積分値は 0.69314718048630297 18 となった。要求精度通りの結果が得られた。このときの関数計算回数は、5 回でその内 2 回が Taylor 展開を計算するための関数計算であった。

分割数 n を増加させると、h は小さくなるので補正項 $c(k)$ はすばやく小さくなる。このため、高次の微分係数はあまり必要としなくなる。Taylor 展開は低次数でも Euler-Maclaurin の総和公式を利用できることになる。逆に Taylor 展開が高次数であるならば、分割数 n を小さく出来る。上の計算では 20 次の Taylor 展開を利用したが、それが適切かどうか考慮する必要がある。

3.2 数 値 例

Euler-Maclaurin の総和公式を利用した計算法の性能を評価するために、Kahaner の問題の中から両端点で特異性を持つため Taylor 展開できない問題等を除いた 13 問題について計算を行った。計算した問題を表 2 に示す。番号は Kahaner の問題の番号である。それを利用して計算した結果を示す。

実行結果を表 2 に示す。N は分割数すなわち関数の計算回数、error はその積分ルーチンが出力した誤差である。分割数の内 2 回は Taylor 展開の計算である。EM は Euler-Maclaurin の総和公式を利用した計算、DE は二重指数型積分公式[15](Double Exponential formula)、DAQN9 は、適応型ニュートン・コーツ法[8] の結果はその参考文献による結果である。IBM 型 64 ビットの浮動小数点演算による結果である。DE のプログラムとして、ネットで公開されている大浦[9] の C 言語用 DE プログラムを C++言語に変換し使用した。

計算機として、AMD Phenom(tm) □ X6 1090T 3.2GHz を使った自作パソコンを利用した。コンパイラーとして Microsoft Visual C++ 2012 を使用した。

3.3 結　　論

Euler-Maclaurin の総和公式を利用して、数値積分する方法を提案した。この方法は、単純でわかり易い計算法であるにも関わらず、多くの場合、有力な数値積分法と同等程度またはそれ以上の性能を発揮する。

この計算を行うには、被積分関数の微分係数を精度良く計算する必要がある。この方法として、自動微分法があるが、一般のユーザーにそれほど普及していない。この方法が使われるためにも自動微分の普及することを期待したい。

参 考 文 献

1) Abramowitz M. and Stegun I. A., Handbook of Mathmatical Functions, Dover,(1972)
2) Davis P.J., Rabinwitz P.(森 正武訳), 計算機による数値積分法, 日本コンピュータ協会, (1981)
3) David Abrahams, Aleksey Gurtovoy, "C++ Template Metaprogramming", Addison Wesley, (2005)
4) Gisela Engeln-Müllges, Frank Uhlig, Numerical Algorithms with Fortran, Springer, 1996
5) Henrici P., Applied and Computational Complex Analysis, Vol. 1, John Wiley & Sons, New York, (1974)
6) 平山, 舘野, 浅野, 川口, Taylor 級数演算ライブラリの使用法, 東北大学情報シナジーセンター大規模科学計算システム広報 SENAC, 40(2007) 29–68
7) 平山、小宮、佐藤、Taylor 級数法による常微分方程式の解法, 日本応用数理学会論文誌、12(2002), pp.1–8
8) 二宮市三, 適応型ニュートン・コーツ積分法の改良, 情報処理学会誌,21(1980),504–513
9) 大浦 拓哉, Ooura's Mathematical Software Packages, http://www.kurims.kyoto-u.ac.jp/ooura/index-j.html

Vol.2012-ARC-202 No.20
Vol.2012-HPC-137 No.20
2012/12/14

表 1 Test Problems

No.	Integral
1	$\int_0^1 e^x dx = 1.7182818285$
4	$\int_{-1}^1 0.92\cosh x - \cos x dx = 0.47942822669$
5	$\int_{-1}^1 \frac{1}{x^4+x^2+0.9} dx = 1.5822329637$
8	$\int_0^1 \frac{1}{x^4+1} dx = 0.86697298734$
9	$\int_0^1 \frac{2}{2+\sin(31.4159x)} dx = 1.1547006690$
10	$\int_0^1 \frac{1}{1+x} dx = 0.69314718056$
11	$\int_0^1 \frac{1}{e^x+1} dx = 0.37988549304$
12	$\int_0^1 \frac{x}{e^x-1} dx = 0.77750463411$
13	$\int_{0.1}^{10} \frac{\sin(314.159x)}{3.14159x} dx = 0.0090986452566$
16	$\int_0^{10} \frac{50}{3.14159(2500x^2+1)} dx = 0.49936380287$
17	$\int_{0.01}^1 (\frac{\sin(50\times x)}{50\times 3.14159x})^2 \times 50 dx = 0.11213956963$
18	$\int_0^\pi \cos(\cos x + 3\sin x + 2\cos 2x + 3\sin 2x + 3\cos 3x) dx = 0.83867632338$
20	$\int_{-1}^1 \frac{1}{x^2+1.005} dx = 1.5643964441$

10) 長田直樹, 対数収束級数の漸近展開と加速法, 情報処理学会論文誌 29(1988), 256–261
11) 長田直樹, お話:数値解析第 3 回 オイラー・マクローリンの公式, http://www.cis.twcu.ac.jp/osada/rikei/rikei2008-7.pdf
12) Rall,L. B., Automatic Differentiation Technique and Applications, Lecture Notes in Computer Science, Vol. 120, Springer Verlag, Berlin-Heidelberg-New York, (1981)
13) 篠原能材, 数値解析の基礎, 7 章, 日新出版, (1978)
14) 杉原, 室田, 数値計算法の数理, 12 章, 岩波書店, (2007)

表 2 Comparison of Performance of Quadrature Routines($\epsilon = 10^{-9}$)

	EM		DE		DAQN9	
No.	N	error	N	error	N	error
1	3	3.5e-11	33	3.4e-9	25	5.2e-18
4	3	2.5e-10	67	2.2e-9	25	3.5e-17
5	9	2.2e-10	65	6.3e-9	61	1.8e-11
8	9	1.7e-10	65	3.5e-9	91	8.9e-12
9	33	5.4e-10	519	3.7e-9	81	1.0e-5
10	5	3.1e-10	33	1.4e-9	21	3.9e-13
11	3	2.6e-10	33	7.6e-11	21	3.3e-18
12	3	7.3e-11	33	1.6e-9	21	2.2e-18
13	129	7.9e-10	461	1.2e-8	321	6.2e-7
16	5	3.2e-10	141	4.2e-9	91	5.2e-7
17	129	1.9e-10	555	5.7e-9	101	7.4e-4
18	33	1.1e-10	131	1.7e-8	51	5.4e-6
20	17	3.1e-13	65	6.3e-9	21	1.1e-8

15) Takahasi, H. and Mori, M., Double exponential formula for numerical integration, *Publ. RIMS*, Kyoto Univ.,9(1974),121–141