社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－177 (4)
2008－HPC－114 (4)
2008／3／5

最良近似式を利用した区間演算における初等関数の計算

平山弘　舘野裕文

神奈川工科大学工学部

区間演算のプログラムはネット上で多数紹介されている。しかしその多くは四則演算・平方根の区間演算に留まり、基本的な超越関数である初等関数まで扱えるものはほとんど見受けられない。本論文では最良近似式を利用して、初等関数の区間演算プログラムを作成した。その計算方法を示す。

Calculation of the Elementary Functions for the Interval Arithmetic System

Hiroshi Hirayama and Hirofumi Tateno

Kanagawa Institute of Technology

There are many programs for the interval arithmetic in the net. However, almost of them can treat only the interval arithmetic for basic operations and the square root, and they cannot compute even the elementary function i.e. a basic transcendental function. In this paper, it is shown that the interval arithmetic program of the elementary function by using the mini-max approximation and the computational method.

1. はじめに

区間演算は精度保障計算で重要な役割[5-6]を果たすものである。区間演算プログラムは検索エンジンを利用すると複数のプログラムが見つかる。その多くは四則演算、平方根までの区間演算はできるが、指数関数、対数関数などの初等関数が計算できるプログラムはほとんどない。

初等関数の計算法についても大石[5]に少し紹介されている程度である。

区間演算で関数値を計算するには、剰余項を持つ Taylor 展開式を利用して計算する。たとえば指数関数e^xでは、Taylor 展開[1]は

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + R_n$$

$$R_n = e^{\theta x}\frac{x^{n+1}}{(n+1)!} \quad 0 \le \theta \le 1 \qquad (1.1)$$

となる。剰余項R_nが十分小さくなったとき、この Taylor 展開式を評価することによって指数関数の区間を計算する。指数関数e^xを最良近似式の計算を行う。計算は厳密ではないが十分な精度で計算を行う。このような計算をおこなうと

$$e^x = 0.9999999992481 + 1.0000001065372x + 0.49999758x^2 + \cdots + 3.248653\times10^{-4}x^7 + R_n$$

$$|R_n| \le 7.5187\times10^{-10} \quad 0 \le x \le 1 \quad (1.2)$$

の結果が得られる。(1.1)式を評価する方法と同様に(1.2)式を評価することによって、指数関数e^xを精度良く計算できると期待できる。

本論文ではこのような方法で初等関数の区間演算を行う方法を述べる。

2. 最良近似式

区間[a,b]において、与えられた関数$f(x)$を、n 次の多項式$p_n(x)$で近似することを考える。このとき、$f(x)$と$p_n(x)$の距離 L を

$$L(f,p_n) \equiv \sup\left\{|f(x)-p_n(x)|; a \le x \le b\right\} \quad (2.1)$$

と定義する。$p_n(x)$の次数 n を一定に保つとき、$L(f,p_n)$を最小にする関数を、最良近似多項式という。

関数$f(x)$を近似する多項式$p_n(x)$の差を$e_n(x)$とする。すなわち

$$e_n(x) = f(x) - p_n(x) \quad (2.2)$$

と定義する。$p_n(x)$が最良近似式であるとき、$e_n(x)$は次の性質[7]をもつことが知られている。

1) $e_n(x)$の相隣る極大極小値は符号が異なる。
2) $e_n(x)$の極値は、区間内で少なくとも(n+2)個ある。
3) $e_n(x)$の極大極小値の絶対値はすべて等しい。

この性質から、$p_n(x)$を求めることができる。このような性質は、多項式の最良近似式だけでなく、有理関数の最良近似式も同様な性質をもつ。$e_n(x)$の極値は、最良近似式を求めると自動的に求まるので、その値が最良近似式の最大誤差となる。

最良近似プログラムとして、山下[8]が作成したものを C++言語に変更し一部変更したプログラムを使用した。初期値として最良近似したい関数の Chebyshev 展開を求め、Newton 法を使用して高精度化するプログラムである。このとき使用した高精度計算プログラムとして、MPPack[4]を改良したものを使用した。計算精度は 10000 進数 32 桁（10 進数約 128 桁）で計算を行った。

最良近似式についてはその計算方法やそのプログラムについては浜田[2]に詳しい説明がある。具体的な最良近似式は Hart[3]に多数収録されている。数は少ないが、Abramowitz and . Stegun[1]にも近似式が記載されている。

2. 初等関数の区間演算

ここでは、区間演算として 8 バイトの倍精度浮動小数点数で上限下限を指定する区間を考える。区間幅 0 の厳密な数値を与えることができたとしても関数値は途中で丸め処理等が入るために一般に区間幅が 0 でない区間数になる。

作成したプログラムでは、区間[a,b]に含まれる 1 点で与えられた区間[c,c]を意味する倍精度浮動小数点数一個を引数とする関数を準備した。この関数の中で最良近似式を評価し、関数値が求められる。関数値は最良近似式が区間数を係数とする式であり、途中計算に丸め処理が入るため区間数となる。

倍精度浮動小数点数一個を引数とする関数を呼び出すことによって、一般の区間演算用の関数を作成した。

2.1 指数関数

指数関数e^xは次のように求められる。計算機の内部では 2 進数を使っているので、まず

$$e^x = 2^y \qquad y = \log_2 e = \frac{1}{\log 2} \quad (2.3)$$

を満たすyを求める。yを整数部分と小数部分に分ける。

$$y = n + m \qquad 0 \le m \le 1 \quad (2.4)$$

ここでnは整数部分、mは小数部分である。mの指数関数が求まれば、その後はスケーリングだけで指数関数の値が計算できる。

作成したプログラムではmの[0,1]区間を4等分した[0,0.25]区間で関数$\frac{2^m-1}{m}$を最良近似した式を作成し使用した。このように式変形することによって、定数項に区間幅（誤差）が全く無くなるので区間幅の狭い計算が可能となると期待できる。7次式を使うと最大相対誤差が7.073532×10^{-17}となる最良近似式が得られる。以下にその係数を示す。

次数	係数
0	6.93147180559945260387240e-01
1	2.40226506959126022256534e-01
2	5.55041086626832027222974e-02
3	9.61812917632008441734408e-03
4	1.33335473303907635571939e-03
5	1.54044532151211535042559e-04
6	1.52092622443431096617022e-05
7	1.42740017909915102109237e-06

この式は、係数の絶対値がすべて1より小さいので、これらの定数をかけることによって区間幅が小さくなるので、計算結果の区間幅を小さくするのに役立っている。

次数が上がることによって、係数が徐々に小さくなっているため、Honer の方法で計算する場合、加算に対しても区間幅が定数が持っている区間幅程度となることを意味する。

最良近似多項式の係数がすべて正数で、代入する値も区間[0.0,0.25]内の正数であるので、区間数で表現された係数の上限を使うことによって関数値の上限を計算できる。係数の下限を使えば下限を計算できる。このように上限下限の計算を通常の区間数の計算からみるとかなり簡単にできる。これによって計算の高速化が図れる。

7次の最良近似式の0次の定数は、倍精度浮動小数点では厳密には表現できないため、区間数となる。10進数でおよそ

0.6931471805599451752

0.6931471805599452862

の間にある数値である。この二つの数値は倍精度浮動小数点で表現できる数値である。これらの数値のビット表現をみると1ビットだけの違いである。

これらの数値（数値文字列）を直接プログラムに書き込めば計算機の内部表現でこれらの数値に最も近いものに変換されるのであれば、1ビットだけ違う隣り合う倍精度浮動小数点に変換されるはずである。これが保障されるのであれば、このような数値をプログラムに書き込むことによって区間数を係数とする最良近似式を定義でき、容易に計算することができる。

しかしながらこのように最後の1ビットまで正確に内部表現に変換することをプログラム言語は保証していない。このため、本プログラムでは16進数表現を使って正確に入れるようにした。具体的には

```
union edouble
{
    unsigned int ix[2];
    double       x;
};
```

と定義して、

```
edouble p = { 0xfefa39ee, 0x3fe62e42 };
edouble q = { 0xfefa39ef, 0x3fe62e42 };
```

とした。言語の文法書では確認できなかったが union の宣言で最初に宣言された変数に対して初期化ができるようであった。倍精度浮動小数点として使うには、たとえば上のpについてはp.xと記述して、edoubleに代入された倍精度浮動小数点数を使うことができる。

このように、記述した場合、Visual C++とBorland C++では期待どおり動作をしたが文法上正しいのかどうかは不明である。また実際のプログラムでは、上のp,qは区間の上限下

限を示すように区間を初期化し、その配列を作成したかったがうまく動作しなかった。

以下に作成した指数関数の計算例を示す。

x =[1.0, 3.0] のとき exp(x)は

[2.718281828451554194, 20.085536923187692790]

x =[2.0, 3.0] のとき exp(x)は

[7.38905609893064863, 20.085536923187692790]

2.2 対数関数

対数関数 $\log x$ は次のように求められる。計算機の内部では２進数を使っているので、まず

$$x = 2^e \times m \qquad 1 \le m \le 2 \qquad (2.5)$$

を満たす e と m を求める。関数 frexp を使えば容易にこのように分けることができる。従って対数は、

$$\log x = e \log 2 + \log m \qquad (2.6)$$

と計算できる。$\log m$ の計算は、最良近似式を使って計算する。関数 $\dfrac{\log(1+x)}{x}$ を最良近似した式を求めた。単純な多項式で近似したかったが、多項式の次数が高くなるため、分子６次、分母６次の有理関数を使った最良近似式を求めた。このときの最大相対誤差は 5.95×10^{-18} であった。区間数のプログラム記述に対しては、指数関数と同様な方法を使った。対数関数の数値計算例を以下に示す。

x=[2.0, 3.0]のとき log(x)は

[0.693147180559945286 3,1.098612288668110005]

となる。この数値の下限は高精度計算結果よりも小さくなり上限は大きくなっていることがわかる。

2.3 三角関数

三角関数 $\sin x$ は、次のように計算する。まず、三角関数の周期性を利用するために、次の値を計算する。

$$x\frac{2}{\pi} = n + m \quad 0 \le m \le 1 \qquad (2.7)$$

ここで、nは整数、mは小数部である。

n の値によって次のように計算する。

1) $n = 4k$ (k は整数)のとき、

$$\sin x = \sin\frac{\pi}{2}m \qquad (2.8)$$

2) $n = 4k+1$ (k は整数)のとき、

$$\sin x = \sin\frac{\pi}{2}(1-m) \qquad (2.9)$$

3) $n = 4k+2$ (k は整数)のとき、

$$\sin x = -\sin\frac{\pi}{2}m \qquad (2.10)$$

4) $n = 4k+3$ (k は整数)のとき、

$$\sin x = -\sin\frac{\pi}{2}(1-m) \qquad (2.11)$$

として計算する。$\sin\dfrac{\pi}{2}m$ を区間[0,1]で 13 次式を使って最良近似する。そのときの最大相対誤差は 2.73×10^{-18} である。

三角関数の区間演算は、これまでの関数とはまた違った性質を示す。区間幅が 2π を超える場合、区間の上限下限がどんな数値でも区間幅は[-1, 1]となる。

上限と下限から式（2.7）を使うことによってそれぞれの n の値を計算する。下限の n と上限の n を使って区間を求める。

$\cos x$ の計算は、

$$\cos x = \sin\left(\frac{\pi}{2} + x\right) \qquad (2.12)$$

によって計算している。$\sin x$ の式で n を 1 増加させれば $\cos x$ の計算式になることがわかる。

$\tan x$ の計算は

$$\tan x = \frac{\sin x}{\cos x} \qquad (2.13)$$

によって計算している。$\sin x$ と $\cos x$ の両方計算する必要があるので、最初から $\tan x$ を計算した方が高速になる可能性がある。

2.4 逆三角関数

逆三角関数として $\tan^{-1} x$ を計算しそれをもとに $\sin^{-1} x$ や $\cos^{-1} x$ を計算した。逆正接関数 $\tan^{-1} x$ は奇関数であることから次の形に展開できる。

$$\tan^{-1} x = \frac{xP(x^2)}{Q(x^2)} \qquad (2.14)$$

このような形に展開するために、次のような関数を有理関数展開した。

$$\frac{\tan^{-1}\sqrt{x}}{\sqrt{x}} = \frac{p(x)}{Q(x)} \qquad (2.15)$$

区間[0,1]で分子 6 次、分母 6 次で展開すると最大相対誤差は、5.8088×10^{-18} となる。

奇関数であることを利用しない場合、展開式の係数は大きく変化し次数の割にはあまり精度の良くない式になる。

逆正弦関数 $\sin^{-1} x$ の計算は

$$\sin^{-1} x = \tan^{-1}\frac{x}{\sqrt{1-x^2}} \qquad (2.16)$$

によって計算している。(2.16)式では $x=1$ の場合計算ができないので、$x=1$ の場合、$\frac{\pi}{2}$ を返すようにしなければならない。

逆正弦関数 $\cos^{-1} x$ の計算は

$$\cos^{-1} x = \tan^{-1}\frac{\sqrt{1-x^2}}{x} \qquad (2.17)$$

によって計算している。$\sin^{-1} x$ の場合と同様に、$x=0$ の場合、$\frac{\pi}{2}$ を返すようにしなければならない。

3. 数値例

次の計算を行う。計算例は、文献[5]で使用されているものを使った。

3.1 例1

次の対数を計算する。

log1.001

この計算を本プログラム利用して計算するプログラムは、

```
1:  #include “interval.h”
2:  void main()
3:  {
4:      interval x ;
5:      x = interval( 1.001 ) ;
6:      cout << x << "   " << log(x) << endl ;
7:  }
```

となる。このプログラムを計算実行すると下限と上限は

0.00099950033308342235

0.00099950033308342343

となる。この結果は参考文献の結果

0.0009995003330830

0.0009995003330838

とは一致する。しかしながら,高精度計算の結果では

log(1.001)=

0.000999500333083533166809398920 5·

となる。この結果は本プログラムの結果とは一致しない。これは、1.001 の数値が倍精度浮動小数点数は厳密に表現できないことからきている。

プログラムでは

log（1.001）

と書かれているが、1.001 が厳密に表現できないため、プログラムでは

log(1.0009999999999988990)

と解釈される。高精度計算でこの式を計算すると、

log(1.0009999999999988990)=

0.000999500333083423176799408910 5

となり、区間演算結果と一致する。

このような問題点を避けるもっと簡単な方法は、厳密に表現できる 1001 と 1000 を区間数に変換し、割り算を行う。このようにして計算すると、プログラムは

```
1:  #include "interval.h"
2:  void main()
3:  {
4:      interval x ;
```

```
5:      x = interval( 1001 );
6:      x /= 1000;
7:      cout << x << "   " << log(x) << endl;
8:  }
```

となり、計算結果の結果の上限と下限は

0.00099950033308320074

0.00099950033308364526

となる。

区間の計算を行うには、定数の精度を考慮して計算する必要がある。区間演算は厳密な結果を出すので、特に注意しなければならない。

3.2 例2

次の三角関数を計算する。

$\sin 0.02$

この計算を本プログラム利用して、

```
sin(0.02)
```

と記述して計算すると下限と上限は

0.01999866669333306979

0.01999866669333308714

となる。この結果は参考文献の結果

0.00199986666933329

0.00199986666933334

とは一致する。高精度計算結果

0.00199986666933330793664902946 9

とも一致する。厳密に表現できる 1001 と 1000 を区間数に変換し、割り算を行う。このようにして計算すると、結果の上限と下限は

0.01999866669333306979

0.01999866669333308714

4. まとめ

区間演算のための初等関数を作成した。以前のバージョンは、初等関数を高精度計算し、それを区間に直す方法をとっていたが、その場合、区間の演算を行う場合必ず高精度計算ライブラリをリンクする必要があり、あまり使い勝手が良いとは言えなかった。このプログラムを使うことによって、高精度ルーチンをリンクする必要性がなくなりかなり使いやすくなった。

高精度ルーチンを使わないので計算が高速化されたが、少し計算精度が落ちることになった。

ガンマ関数および誤差関数についてはほぼ完成しているので、C99 に準備されている実数値関数はほぼ満たしている。Bessel 関数のように二つ以上引数を持つ関数は、最良近似式を使う方法はあまり良い方法ではないと考えている。

複素区間数についてはまだ四則演算も作っていないが、複素 template を使えば簡単に使えるようになると考えている。

参考文献

[1] M. Abramowitz and I. A. Stegun, Handbook of Mathmatical Functions, Dover(1972)

[2] 浜田 穂積, 近似式のプログラミング, 培風館. (1995)

[3] Hart et al, Computer Approxmation, John Wiley & Sons, (1968)

[4] 平山 弘, C++言語による高精度計算パッケージの開発,日本応用数理学会, Vol. 5, No.3, pp. 123-134, (1995)

[5] 大石進一, 精度保証付き数値計算、コロナ社、(2001)

[6] 大石進一, 非線形解析入門、コロナ社、(2001)

[7] 宇野利雄："数値計算", 朝倉書店、(1963)

[8] 山下眞一郎, 最良近似式を求めるプログラム, 情報処理、Vol.10,6(1969) ,pp.442-446