2005－HPC－104（7）
2005／10／7

Taylor 展開法を使った AGM 法による指数対数関数計算の高速化

平山　弘、西川　瞬、山村　礼

神奈川工科大学

Taylor 級数の四則演算や関数計算は、C++言語や Fortran を使うと容易に定義できる。これを利用すると多くの種類の関数を Taylor 級数に展開できる。AGM(相加相乗平均法、Arithmetic and Geometric Means)で対数関数や指数関数値を計算するには、非線形方程式を解かなければならない。解くべき方程式を Taylor 級数に展開し、その逆関数の Taylor 展開式を求め計算する。この方法を使うと Brent の方法と比較し約２倍程度の高速化がはかれる。

Effective Computing Method for Exponetial and Logarithmic Functions by AGM Algorithm using Taylor series

Hiroshi Hirayama, Shun Nishikawa, Akira Yamamura

Kanagawa Institute of Technology

Arithmetic operations and functions of Taylor series can be defined easily by FORTRAN 90 and C++ program language. Using this, it is shown that many types of functions can be expanded in the Taylor seeies. The non-linear equations must be solved to calculate logarithmic and exponential functions by AGM algorithm. Expands the non-linear equations in Taylor series and compute the Taylor series of the invers functions of the equations, then the values of these functions can be computed about two times faster than Brent algorithm.

1. はじめに

プログラムでよく使われる演算子（+,-,*,/など）を、被演算の型が異なる場合、別の意味を与えることができる C++言語[2]の機能（operator overload）を使い、有限項で打ち切った Taylor 級数間の四則演算、Taylor 級数の関数演算[3,4,6]を定義することができる。この機能を使うと、プログラムの形で与えられた任意の関数を Taylor 級数展開することができる。

これを使うと常微分方程式の初期値問題の解は Picard の逐次近似法[4,5,7]を使うことによって、任意次数の Taylor 級数展開の形で得ることができる。関数 $y=f(x)$ の逆関数 $y=f^{-1}(x)$ はつぎのような常微分方程式を満たすことが知られている。

$$\frac{dy}{dx}=\frac{1}{f'(y)} \qquad (1.1)$$

この方程式は、初期値 $y(0)=a_0$ のとき、つぎのような Picard 逐次近似法

$$y_0=a_0 \text{、} \quad y_n=a_0+\int_0^x f(x,y_{n-1})dx \qquad (1.2)$$

によって、Taylor 級数解を任意次数で計算することができる。これを利用すれば任意の関数の逆関数の Taylor 級数を計算できる。

本論文では、この方法を AGM 法[1]を利用した指数対数関数の計算に利用した。

AGM 法によって指数関数や対数関数を計算するには、ＡＧＭ法で高速に計算できる関数$U(m)$、$T(m)$があり、

$$U(m) = \log T(m) \qquad (1.3)$$

の関係がある。もしある数xの指数関数を計算したい場合、非線形方程式

$$U(m) = x \qquad (1.4)$$

を解き、mを求め、$T(m)$を計算すれば、指数関数を計算できる。逆に、

$$T(m) = x \qquad (1.5)$$

を解き、$U(m)$を計算すれば、対数関数を計算したことになる。Brent の論文では、差分法で近似した微分係数を利用した Newton 法で計算することを提案している。関数$U(m)$、$T(m)$はつぎのように定義される関数である。ここでは倍精度で計算するＣ言語プログラムで示す。

```
double u( const double m )
{
  double a, b, c, s ;
  const double pi=3.141592653589793238 ;
   a = 1 ; b = sqrt(1-m) ;
   while( a-b > 1.0e-7  )
   {
       c = (a+b)/2 ;   b = sqrt(a*b) ;  a = c ;
   }
   a = pi/(a+b) ;
   s = sqrt(m) ;
   while( 1-s>1.0e-7 )
   {
       a = a*(1+s)/2 ;  s = 2*sqrt(s)/(1+s) ;
   }
   return a*(1+s)/2 ;
}
double t( const double m )
{
   double s, v, w ;
   v = 1 ; s = sqrt(m) ;
   while( 1-s > 1.0e-7 )
   {
       w = 2*s*v/(1+v*v) ;
       w = w/(1+sqrt(1-w*w)) ;
       w = (v+w)/(1-v*w) ;
       v = w/(1+sqrt(1+w*w)) ;
       s=2*sqrt(s)/(1+s) ;
   }
   return (1+v)/(1-v) ;
}
```

この計算法は、非常に精度が高いとき有用で、特に乗算を行うのに高速フーリエ変換を使わなければならない場合、多くの計算機では 1000 桁程度以上の高精度計算の場合に有用であることが知られている。

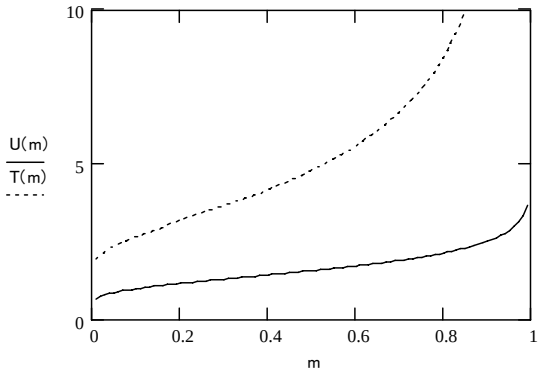

図 1

U(m)、T(m)のグラフを図 1 に示す。図は$0.01 \le m \le 0.99$の範囲のグラフで、両端 0.01 は描かれていないので注意する必要がある。この両端部分では、関数は急激に増加減少する。このため、計算は$m = 0.5$付近で行うようにスケール変換を行って計算する。

2. Taylor 級数プログラムの作成

Taylor 級数は、プログラムとしては、係数の配列として表現する。すなわち、Taylor 級数を$x = a$で展開したときの式を

$$f(x) = f_0 + f_1(x-a) + f_2(x-a)^2 + \cdots \qquad (2.1)$$

と表現する。この中の低次の係数ｍ個を取り、ｍ次を越える高次係数を省略する。係数

$$f_0, f_1, f_2, f_3, f_4, \cdots, f_m \qquad (2.2)$$

を配列として表現する。展開位置は、原点に固定することにして、Taylor 級数の表現の中には含めない。関数は平行移動によって展開位置を原点にすることができるので、このことによって、一般性を失うことはない。

Taylor 級数は、C＋＋言語及び FORTRAN90 の構造体を使って定義される。計算できる最大次数は、係数配列の大きさになる。

以下で示す Taylor 級数の演算は、すべて配列と配列の演算として定義している。以下で説明するもの以外にも、Taylor 級数の定数倍なども定義しているが、容易に導けるので、ここでは省略してある。

2.1 Taylor 級数の四則演算

Taylor 級数の四則計算のプログラムは、以下のように簡単に作ることができる。平行移動によって、展開位置を原点移すことができるので一般性を失うことなしに、原点で展開した式だけを扱うことができる。この級数を次のように定義する。

$$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots \quad (2.3)$$

$$g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots \quad (2.4)$$

$$h(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \cdots \quad (2.5)$$

このとき、四則演算は、以下のように定義できる。これらの公式は簡単なものであるがまとめて、記載されている文献があまりないので以下に記載する。ここで、mは、演算の対象となっている Taylor 級数の次数である。

（１）和差　$h(x) = f(x) \pm g(x)$

このとき、係数は次の式によって計算することができる。

$$h_n = f_n \pm g_n \quad (2.6)$$

（２）乗算　$h(x) = f(x)g(x)$

このとき、係数は次の式によって計算することができる。

$$h_n = \sum_{k=0}^{n} f_k g_{n-k} \quad (2.7)$$

（３）除算　$h(x) = \dfrac{f(x)}{g(x)}$

このとき、係数は次の式によって計算することができる。式からわかるように、$g_0 = 0$ のときは、計算することはできない。ただし、$f_0 = \mathbf{g}_0 = 0$ の場合は、分子と分母をxで割る操作を行う。この操作で、$g_0 \neq 0$ になれば、以下の式で除算を行うことができる。

$$h_n = \frac{1}{g_0}\left(f_n - \sum_{k=0}^{n-1} h_k g_{n-k} \right) \quad (2.8)$$

この公式は、$g(x)h(x) = f(x)$ とおいて、(2.3)、(2.4)、(2.5)の式を代入して、展開し、各次数の係数が等しいと置いて得られる。

（４）逆数　$h(x) = \mathrm{invers}(f(x)) = \dfrac{1}{f(x)}$

このとき、係数は次の式によって計算することができる。除算と同じ方法で得られる。除算と同じように、$g_0 = 0$ のときは、計算することはできない。

$$h_0 = \frac{1}{f_0},\ h_n = -\frac{1}{f_0}\sum_{k=0}^{n-1} h_k f_{n-k} \quad (2.9)$$

（５）二乗　$h(x) = \mathrm{square}(f(x)) = f(x)^2$

このとき、係数には次のような関係式が成り立つ。この式によって、二乗の計算を約２倍の速さで計算することができる。

$$h_0 = f_0^2$$

$$h_n = \begin{cases} 2\sum_{k=0}^{(n/2(} f_k f_{n-k} & n : odd \\ (f_{n/2})^2 + 2\sum_{k=0}^{\left(\frac{n-1}{2}\right(} f_k f_{n-k} & n : even \end{cases} \quad (2.10)$$

ただし、記号$\lfloor x \rfloor$はxを超えない最大の整数を示す。

2.2 Taylor 級数の関数計算

関数の計算は、常微分方程式を級数法で解くアルゴリズムを使って計算することができる。この計算方法は、簡単な常微分方程式の Taylor 級数による解法の例になっている。

（６）べき乗 $h(x) = f(x)^{\alpha}$ （αは定数）

このとき、この関数は、つぎの微分方程式を満たす。

$$f(x)\frac{d\,h(x)}{dx} = a\,h(x) \quad (2.11)$$

この式の両辺に、(2.3)、(2.4)、(2.5)の式を代入して、各次数のxの係数を等しいと置いて、次の関係式が得られる。

$$h_0 = f_0^{\alpha} \quad 、$$

$$h_n = \frac{1}{n f_0}\sum_{k=1}^{n}\{(\alpha+1)k - n\} f_k h_{n-k} \quad (2.12)$$

$a = \dfrac{1}{2}$ とおけば、平方根を計算するためのプログラムになる。上の式を単純に計算すると、$f_0 = 0$ のとき、計算ができなくなるが、$f_0 = 0$ であっても、$f_k = 0 (k < p)$、$f_p \neq 0$、$a > 0$ で ap が整数ならば、計算可能で、計算結果は Taylor 級数になる。たとえば、

$$\sqrt{x^2 + x^3} = x + \frac{1}{2}x^2 - \frac{1}{8}x^3 + \cdots \quad (2.13)$$

となる。プログラムでは、このような場合でも問題なく計算できるようになっている。

（７）指数関数　$\mathrm{h}(x)=e^{\mathrm{f}(x)}$

このとき、この関数は、次の微分方程式を満たす。

$$\frac{d\,\mathrm{h}(x)}{dx}=\mathrm{h}(x)\frac{d\,\mathrm{f}(x)}{dx} \tag{2.14}$$

この式から、べき乗計算の場合と同様な方法で、次のような関係式が得られる。

$$\mathrm{h}_0=e^{\mathrm{f}_0},\ \mathrm{h}_n=\frac{1}{n}\sum_{k=1}^{n}k\,\mathrm{h}_{n-k}\mathrm{f}_k \tag{2.15}$$

（８）対数関数　$\mathrm{h}(x)=\log\mathrm{f}(x)$

このとき、この関数は、次の微分方程式を満たす。

$$\mathrm{f}(x)\frac{d\,\mathrm{h}(x)}{dx}=\frac{d\,\mathrm{f}(x)}{dx} \tag{2.16}$$

この式から、べき乗計算の場合と同様な方法で、次のような関係式が得られる。

$$\mathrm{h}_0=\log\mathrm{f}_0,\ \mathrm{h}_n=\frac{1}{n\,\mathrm{f}_0}\left(n\,\mathrm{f}_n-\sum_{k=1}^{n-1}k\,\mathrm{h}_k\mathrm{f}_{n-k}\right) \tag{2.17}$$

（９）三角関数

$$\mathrm{g}(x)=\sin\ \mathrm{f}(x),\ \mathrm{h}(x)=\cos\ \mathrm{f}(x)$$

このとき、この関数は、次の微分方程式を満たす。

$$\frac{d\,\mathrm{g}(x)}{dx}=\mathrm{h}(x)\frac{df(x)}{dx},\qquad \frac{d\,\mathrm{h}(x)}{dx}=-\mathrm{g}(x)\frac{df(x)}{dx} \tag{2.18}$$

この式から、係数に対する次のような関係式が得られる。

$$\mathrm{g}_0=\sin\ \mathrm{f}_0,\ \mathrm{h}_0=\cos\ \mathrm{f}_0$$

$$\mathrm{g}_n=\frac{1}{n}\sum_{k=1}^{n}k\,\mathrm{h}_{n-k}\mathrm{f}_k,\ \mathrm{h}_n=-\frac{1}{n}\sum_{k=1}^{n}k\,\mathrm{g}_{n-k}\mathrm{f}_k \tag{2.19}$$

三角関数は、このようにsinとcosを同時に計算すると、計算式が単純で見易い公式となる。tanはこのようにして得られたsinとcosの Taylor 級数をわり算することによって得る。この事情は、sinhとcoshの場合も同様である。

3. 逆関数の Taylor 級数を使った方程式の解法

方程式を逆関数の Taylor 展開式を利用して解く方法を示す。

次のような方程式を考える。

$$f(x)=a \tag{3.1}$$

この方程式を解くために、x の近似値 x_0 を推定する。この点での関数 $f(x)$ の値を a_0 とする。すなわち

$$f(x_0)=a_0 \tag{3.2}$$

とする。$x=x_0$ で関数 $f(x)$ を Taylor 級数展開すると

$$f(x)=a_0+a_1(x-x_0)+a_2(x-x_0)^2+\cdots \tag{3.3}$$

となる。ここで、a_1、a_2、…は関数を展開して得られる係数である。式(3.3)を利用して、$f(x)$ の逆関数の Taylor 展開式を求める。展開式は、$x=a_0$ で行うと

$$f^{-1}(x)=x_0+b_1(x-a_0)+b_2(x-a_0)^2+\cdots \tag{3.4}$$

のようになる。ここで、b_1、b_2、…は(1.2)式などを利用して逆関数を展開して得られる係数である。

式(3.4)に $x=a$ を代入することによって方程式(3.1)の解を計算することができる。式（3.4）が収束し、収束が速いならばこの計算によって、解を十分な精度で計算できる。しかしながら、実際に利用できる展開式は有限項で打ち切った近似式であるから、その値も近似値になる。精度が不十分ならば、その近似値を使ってもう一度その点における式(3.3)に相当する Taylor 展開式を求め、それから、式(3.4)に相当する逆関数の Taylor 展開式を求め、十分な精度が得られるまで計算を繰り返す事になる。

数値例として、1節において C 言語で定義された $T(x)$ を Taylor 展開し、その逆関数を求める。

次の方程式を考える。

$$T(x)=0.5 \tag{3.5}$$

$T(x)$ を $x=0.5$ で Tayor 展開する。7 次まで展開すると

```
4.81048+6.90563(x-0.5)+8.11216(x-0.5)^2
+15.2492(x-0.5)^3+24.0906(x-0.5)^4
 +46.7273(x-0.5)^5+80.0641(x-0.5)^6
 +157.153(x-0.5)^7
```

の式が得られる。この計算のように計算をある程度計算し、収束したものとして途中で計算を打ち切るような計算では、定数項は、通常の数値計算と同じなので、その精度は問題ないが、１次、２次のなどの高次の係数はどの程度一致するかは、これからの検討課題である。一般に、高次の係数はそれほどの精度を必要としないが、十分な精度が得られてい

るかどうか検討する必要がある。

この関数$T(x)$の逆関数を$x=4.81048$で Taylor 展開すると

0.5+0.144809(x-4.81048)-0.0246335
(x-4.81048)^2+0.00167526(x-4.81048)^3+0.00
0605233(x-4.81048)^4-0.00031087(x-4.81048)
^5+8.78442e-05
(x-4.81048)^6-1.79875e-05(x-4.81048)^7

となる。この式に$x=0.5$を代入すると、方程式(2.5)の解が得られる。真の解が$x=0.526572$と 0.5 に近いため逆関数の収束が速くなっているため、かなり精度の良い解が得られる。

4. 計算法の改良

Taylor 展開を使って、方程式(3.5)のような計算を行う利点は、Taylor 展開の方が関数値を計算するに比べて、高速であることが期待できる場合である。すなわち、関数値を２回、３回と計算する計算量と Taylor 級数を１次の項、２次の項まで計算する計算量を比較し、Taylor 級数の計算が小さくなれば Taylor 展開が有利である。

通常の加減の演算では演算量は、同じになるのでそれ以外の計算が重要である。乗算は、Taylor 展開が不利である。Taylor 展開式の係数は式(2.7)によって計算することができる。これからわかるように、次数が高くなると計算量が増え、Taylor 展開法が不利になる。

除算は Taylor 展開が低次の場合に有利であるが次数が上がるつれて不利になる。何次の Tayor 級数を計算する場合でも、除算は、一回しか必要ないので、低次の場合特に有用である。

平方根の計算も何次の Taylor 展開式を計算しても、平方根の計算は１回しか必要としないので、除算以上に有利である。

これらを総合すると、Taylor 展開法は低次のとき有用で、ある程度高次になると、あまり有用とは言えなくことがわかる。

今回の計算では、計算に平方根の計算があるので、２～３倍程度の高速化が可能であると推定し、計算を行った。

単純に n 次の Taylor 級数を計算するプログラムを適用しただけでは、高速化を行うことが出来なかった。

このため、１次から４次まで Taylor 展開専用のプログラムを作成し、高速化をはかった。. たとえば、１次式の平方根を計算するにはつぎのような計算を直接書いて、計算のオーバーヘッドを少なくした。

$$\sqrt{a_0+a_1x}=\sqrt{a_0}+\frac{a_1}{2\sqrt{a_0}}x \tag{4.1}$$

１次式では、完全にこのように記述にした。２次以上の式もなるべくこのような記述にしたが、式が複雑になるため、完全には行うことができなかった。

Taylor 級数の逆数の計算では、通常の Newton 法と同様に

$$x_{n+1}=x_n+x_n(1-ax_n) \tag{4.2}$$

を使い計算した。最後の括弧の中は、x_nがm次まで正しい展開式ならば、m次までゼロになるので、実質的に次数が半分となるため、かなり計算量を減らせる。

Taylor 級数の平方根の逆数（$1/\sqrt{x}$）の計算でも、逆数の計算と同様な性質を持つ式

$$x_{n+1}=x_n+\frac{x_n(1-ax_n^2)}{2} \tag{4.3}$$

が得られるので、これを利用したプログラムを作成した。これによって計算量を減らすことができる。

通常平方根の計算では、平方根を計算しないで、平方根の逆数を計算する。平方根を計算するには、得られた式に元の式を掛ければ簡単に得られる。この方法は、高精度計算ではもちろんのこと、最近開発された多くの CPU で平方根を計算するために利用されている。

5. 数値例

ここでは、

$\log\pi=1.14472988584940017414342735 13\cdots$

の計算を行ってその性能評価を行った。

計算は、１次式から３次式を利用して、最も速く計算できたものをその結果としている。

表１　計算時間（単位秒）

計算桁数	Taylor 法	Brent 法	倍率
1000	0.188	0.359	1.91
2000	0.594	1.109	1.87
10000	6.297	8.969	1.42

実際の計算では、計算機のメモリとかの関係か、２次式が最良の時間を出す場合もあったが多くの場合１次式であった。ここで示した結果はすべて１次式を利用した場合の結果である。

高次の式による計算があまり高速にならなかった理由として、高次の式を使うためには数値を高精度で計算しなければならない点が

あげることができる。通常の数値計算では、高次の公式を使った場合でも、計算は同じ精度で計算する。高精度計算では、その次数に合った計算精度で計算する必要がある。最終的に得られる精度で計算する必要がある。たとえば、３次の式を使う場合、精度は４倍の結果が得られる。このため、この公式を有効に使うためには、計算を４倍の精度で計算しなければならない。３次式を使うことは、１次の式を２回使うのと同じ計算となるが、１次式を使った場合には、１回目の計算は計算精度は２倍で済み、計算の高速化がはかれる。

高精度の計算が、あまり高速にならない問題点として考えられる事は、前節でいろいろな改良を行っているが、基本的には、すべての精度で計算できるようにプログラムが作成されている点にもある。この点は、計算精度を変更するパラメターを細かく設定することによって 1000 桁と同程度の高速化がはかれると思われる。

使用した高精度計算ルーチンは、１語に１０進４桁を入れるようにした自作の高精度浮動小数点計算プログラムである。このプログラムでは、約 1200 桁を超えると乗算には基数 8 の FFT を利用して乗算を行う。約 1500 桁を超えた場合、2 乗計算でもこの FFT を利用して計算するようになっている。

使用したコンパイラは Borland C++ Ver.6、コンピュータは Pentium D 3.2GHz, OS は Windows XP Pro. X64 を使った。計算は 1 個の CPU を使用して行った。

6. 結論

AGM による指数対数関数の計算法に Taylor 級数を適用し、高速化の可能性を調べた。高次の公式を使うためには、途中の計算も高精度にする必要があり、精度を低く抑えて高速化をはかるという高精度計算の特有の高速化が難しいという問題点があることがわかった。また、高次の公式を使うと、計算精度が４倍、５倍と変化するため、計算したい精度とうまく整合する場合は効率的であるが、整合しない場合、あまり効率的とはいえない状況になることがあった。

このため、5 次を超えるような公式では、あまり良い結果は得られなかったが、１次、２次、３次の公式を使った場合、２倍程度の高速化が行えた。

今回組み込むことができなかったが、高次の公式を利用するとき現れる高次の多項式を計算する場合、FFT のサブルーチンを呼び出す回数をかなり減らすことができる。これを利用すればかなり改善できると思われる。

Brent の計算法より高速に計算はできたが、AGM を利用しない、Taylor 展開式や連分数展開式を利用した計算法から比べれば、それほど速くないので、さらなる改良が必要であると思われる。

参考文献

[1] Brent R.P., “Fast Multiple-Precision Evaluation of Elementary Functions” J. Assoc. Comp. Machi. Vol.23, 1976, pp.242-251

[2] Ellis M. A. and Stroustrup B., The Annotated C++ Reference Manual, Addison-Wesley,1990

[3] Henrici, P., Applied Computational Complex Analysis, Vol. 1, John Wiely & Sons, New York,Chap. 1, 1974

[4] 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

[5] Hirayama H., Numerical Technique for Solving an Ordinary Differrential Equation by Picard's Methods, Integral Methods in Science and Engineering/Editor P. Schiavone, C. Constanda, A. Mioduchowski, Birkhauser, Berlin(2002)

[6] Rall,L. B. , Automatic Differentiation-Technique and Applications, Lecture Notes in Computer Science, Vol.120, Springer-Verlag, Berlin-Heidelberg-New York(1981)

[7] 佐野理, キーポイント微分方程式, 岩波書店, 東京(1993)