

Taylor級数の演算ライブラリの開発

平山 弘

機械システム工学科

Development of Arithmetic Package for Taylor series

Hiroshi HIRAYAMA

Abstract

An arithmetic package for Taylor series

$$f(x) = f_0 + f_1(x-c) + f_2(x-c)^2 + f_3(x-c)^3 + \dots$$

is developed by overloading in C++ language. When we use this program, this series can be treated as intrinsic numbers in this language and we can easily calculate power series of the functions for numerical computation. As Taylor series can be got easily, many analytical methods can be used for solving numerical problems. It gives new methods for numerical analysis. Numerical examples are shown to illustrate the performance of the present package.

Key Words: Taylor series, C++ language, numerical analysis, power series

1. はじめに

計算機によるべき級数やTaylor級数（以下では、両級数をTaylor級数と言う。）の計算は、容易に行うことができる。すでに多くの研究がなされ、いろいろな数値計算に利用されている。これまでの研究については、G. Corliss and Y.F.Chang¹⁾やその参考文献を見ていただきたい。

このように多くの研究が行われているにもかかわらず、多くの数値計算関係のテキストが、Taylor級数を使った計算方法については全く述べられていない。この事実からみると、実際の計算には、ほとんど利用されていないと思われる。

この最大の原因は、Taylor級数でいろいろな計算をするためには、FORTRANなどのプログラム言語で単純な式で表現できるものが、数行にわたるサブルーチンの呼び出し列になることにありと思われる。高級言語で作られたプログラムをTaylor級数で計算できるようにする作業は単純であるが、ある程度以上の大きなプログラムでは、かなりの作業となる。このような作業が発生するためか、Taylor級数を利用する数値計算は限られたものになっている。

このような変換作業をしなくても、Taylor級数の計算を行うことができるプログラミング言語C++²⁾やFORTRAN 90が、一般に使われるようになって来

たのを機会に、Taylor級数のプログラムをC++言語で作成した。C++言語は、現在の主要プログラミング言語の一つであるC言語をサブセットとして持ち、C言語と非常に相性の良い言語である。

C++言語で作成されたTaylor級数計算プログラムは、言語仕様上は、C++言語と同じになるため、問題が発生する可能性が少なくなるだけでなく、問題が起きたとしても対処が容易である。また、汎用の言語で記述しているため、この言語で作成されたプログラムを簡単に利用できるなど、いろいろな応用プログラムの作成が容易にできる。

このようなプログラムを利用した数値計算は、これまでの数値計算とはかなり違ったものになっている。これまで数式処理とか数式処理と数値計算のハイブリットな計算とか呼ばれることが多いが、ここでは、この計算方法を数値計算法の一種として提案したいので、特別な呼び方はしない。

Taylor級数のプログラムは、比較的簡単に作成できるが、その応用は、数値計算の広範囲にわたる。たとえば、関数計算、極限計算、数値微分、数値積分、常微分方程式、方程式の解法などの広い範囲の応用が考えられる。

本論文では、Taylor級数のプログラムの作成方法について述べる。応用例についても、概要を述べる。応用例については、このプログラムの有効性、可能性を論ずるのが目的であり、応用の限界など

の詳細に関しては、次の機会に述べるつもりである。

この研究で使用したC++言語は、主にBorland C++ 3.1, 4.5Jである。計算機は、主にNEC PC-9821AP2を使用した。

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

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

で表現するとき、この中の m 個を取ったもの

$$f_0, f_1, f_2, f_3, f_4, \dots, f_m \quad (2.2)$$

を配列として表現する。展開位置は、簡単化のために、Taylor級数の表現の中には含めていない。

Taylor級数は、C言語の構造体 (struct) にあたるclassを使って定義される。係数配列の大きさは、プログラムの中で最初に指定するようになっている。指定されない場合には、既定値が自動的に設定されるようになっている。以下で示すプログラムでは、その既定値は20である。この値はメモリ管理を省略して、プログラムの構造を簡単にするために、プログラム実行中に変更できないようになっている。最初のTaylor級数の変数を宣言する前に、一度だけ設定することができる。この値は計算可能Taylor級数の最大次数に1を加えた値に設定することになる。この次数以下ならば、Taylor級数の次数は計算途中でも自由に変更することができる。

以下で示すTaylor級数の演算は、配列と配列の計算として定義している。以下で説明したもの以外にも、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。

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

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

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

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。

$$(1) \text{ 和差 } h(x) = f(x) \pm g(x)$$

このとき、係数は次の式によって計算することができる。

$$h_n = f_n \pm g_n \quad (n = 0, \dots, m) \quad (2.6)$$

$$(2) \text{ 乗算 } h(x) = f(x)g(x)$$

このとき、係数は次の式によって計算することができる。

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

$$(3) \text{ 除算 } h(x) = \frac{f(x)}{g(x)}$$

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、 $f_0 = g_0 = 0$ 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。

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

この公式は、 $g(x)h(x) = f(x)$ において、(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。

$$(4) \text{ 逆数 } h(x) = \text{invers}(f(x)) = \frac{1}{f(x)}$$

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。

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

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

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。

$$h_0 = f_0^2$$

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

$$(n = 1, \dots, m) \quad (2.10)$$

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となっている。

$$(6) \text{ べき乗} \quad h(x) = f(x)^a \quad (\alpha \text{ は定数})$$

このとき、この関数は、つぎの微分方程式を満たす。

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

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。

$$\begin{aligned} h_0 &= f_0^a, \\ h_n &= \frac{1}{n f_0} \sum_{k=1}^n \{(\alpha+1)k - n\} f_k h_{n-k} \\ (n &= 1, \dots, m) \end{aligned} \quad (2.12)$$

$a = \frac{1}{2}$ 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、 $f_k = 0$ ($k < p$)、 $f_p = 0$ 、 $a > 0$ 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、

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

がある。プログラムでは、このような場合でも計算できるようになっている。

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

このとき、この関数は、次の微分方程式を満たす。

$$\frac{dh(x)}{dx} = h(x) \frac{df(x)}{dx} \quad (2.14)$$

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。

$$\begin{aligned} h_0 &= e^{f_0}, \quad h_n = \frac{1}{n} \sum_{k=1}^n k h_{n-k} f_k \\ (n &= 1, \dots, m) \end{aligned} \quad (2.15)$$

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

このとき、この関数は、次の微分方程式を満たす。

$$f(x) \frac{dh(x)}{dx} = \frac{df(x)}{dx} \quad (2.16)$$

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。

$$\begin{aligned} h_0 &= \log f_0, \quad h_n = \frac{1}{n f_0} \left(n f_n - \sum_{k=1}^{n-1} k h_k f_{n-k} \right) \\ (n &= 1, \dots, m) \end{aligned} \quad (2.17)$$

(9) 三角関数

$$g(x) = \sin f(x), \quad h(x) = \cos f(x)$$

このとき、この関数は、次の微分方程式を満たす。

$$\begin{aligned} \frac{dg(x)}{dx} &= h(x) \frac{df(x)}{dx}, \\ \frac{dh(x)}{dx} &= -g(x) \frac{df(x)}{dx} \end{aligned} \quad (2.18)$$

この式から、係数に対する次のような関係式が得られる。

$$\begin{aligned} g_0 &= \sin f_0, \quad h_0 = \cos f_0 \\ g_n &= \frac{1}{n} \sum_{k=1}^n k h_{n-k} f_k, \quad h_n = -\frac{1}{n} \sum_{k=1}^n k g_{n-k} f_k \\ (n &= 1, \dots, m) \end{aligned} \quad (2.19)$$

三角関数は、 \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。この事情は、 \sinh と \cosh の場合も同様である。

(10) 双曲線関数

$$g(x) = \sinh f(x), \quad h(x) = \cosh f(x)$$

このとき、この関数は、次の微分方程式を満たす。

$$\begin{aligned} \frac{dg(x)}{dx} &= h(x) \frac{df(x)}{dx}, \\ \frac{dh(x)}{dx} &= g(x) \frac{df(x)}{dx} \end{aligned} \quad (2.20)$$

この式から、係数に対する次のような関係式が得られる。

$$\begin{aligned} g_0 &= \sinh f_0, \quad h_0 = \cosh f_0 \\ g_n &= \frac{1}{n} \sum_{k=1}^n k h_{n-k} f_k, \quad h_n = \frac{1}{n} \sum_{k=1}^n k g_{n-k} f_k \\ (n &= 1, \dots, m) \end{aligned} \quad (2.21)$$

$$(11) \text{ 微分} \quad h(x) = \frac{df(x)}{dx}$$

この定義式から、次のような関係式が得られる。

$$\begin{aligned} f_m &= 0, \quad h_n = (n+1) f_{n+1} \\ (n &= 0, \dots, m-1) \end{aligned} \quad (2.22)$$

このように、最高次数の係数 f_m は、0 となる。

$$(12) \text{ 積分} \quad h(x) = \int_0^x f(t) dt$$

この定義式から、次のような関係式が得られる。

$$h_0 = 0, \quad h_n = \frac{1}{n} f_{n-1} \quad (n = 1, \dots, m) \quad (2.23)$$

定数項は、積分定数なので、任意で良いが、ここでは、0としてある。

2.3 その他のTaylor級数の関数

上記以外の関数で、C++言語で標準で定義されている関数を、上の関数を利用して、次のように定義することができる。

$$(13) \text{ べき乗} \quad h(x) = f(x)^{g(x)}$$

この関数は、

$$h(x) = e^{g(x) \log f(x)} \quad (2.24)$$

として計算することができる。

$$(14) \text{ 逆三角関数}$$

これらの関数は、

$$\sin^{-1} f(x) = \sin^{-1} f_0 + \int_0^x \frac{f'(t)}{\sqrt{1-f(t)^2}} dt \quad (2.25)$$

$$\cos^{-1} f(x) = \cos^{-1} f_0 - \int_0^x \frac{f'(t)}{\sqrt{1-f(t)^2}} dt \quad (2.26)$$

$$\tan^{-1} f(x) = \tan^{-1} f_0 + \int_0^x \frac{f'(t)}{1+f(t)^2} dt \quad (2.27)$$

よって計算することができる。

2.4 Taylor級数の比較

Taylor級数の比較には、いろいろな定義が考えられるが、ここでは、数値計算の流れと同様な手順になるように定義する。Taylor級数の計算は、なるべく数値計算の自然な拡張にするためである。このような拡張を行えば、既に開発されている多く数値計算用のプログラムを、Taylor級数の計算に利用することができるからである。

このようにするには、Taylor級数の展開位置の近傍で、Taylor級数の大きさを比較すればよいことがわかる。すなわち、

$$s = \lim_{x \rightarrow 0} \{f(x) - g(x)\} \quad (2.28)$$

の値によって、大きさを決める。これによって、

$$s > 0 \quad \text{ならば} \quad f(x) > g(x) \quad (2.29)$$

$$s = 0 \quad \text{ならば} \quad f(x) = g(x) \quad (2.30)$$

$$s < 0 \quad \text{ならば} \quad f(x) < g(x) \quad (2.31)$$

と判断する。このように定義すると、定数項の比較だけで判定できるので、単純で大変効率的であり、また、数値計算用のプログラムを簡単に利用できる。しかしながら、この定義では、常微分方

程式の解法のように、まず定数項を決定し、次に1次の係数を決定し、次々に次数を上げていくような計算では、収束の判定がうまくできないなどの不都合も生じる。このような場合には、定数項以外の項を含めた、比較を定義する必要がある。

3. Taylor級数プログラムの簡単な使用例

ここでは、Taylor級数のプログラムの使用例を示し、このプログラムの簡単な使用方法を示す。この使用法を通して上記で説明しにくい機能などを説明する。

Taylor級数は、次のように宣言される。

```
taylor a,b;
```

この場合、aとbのdouble型の配列が宣言される。このとき、Taylor級数は0に初期化される。これらのTaylor級数の係数を設定したり、読み込んだりするには、

```
a[2] = 3; // 2次の係数を3にする。
```

```
p = b[1]; // bの1次の係数を読み込み
```

```
// double型変数pに代入する。
```

のように使う。このようにして設定した、Taylor級数は、

```
a += b; // Taylor級数aとbを加え、
```

```
// Taylor級数aに入れる。
```

```
a = exp(b); // Taylor級数bの指数関数を計
```

```
// 算し、Taylor級数aに入れる。
```

のように、使うことができる。C++言語で使える演算子は、ほぼすべて使えるように定義されている。

Taylor級数に値を代入したい場合、

```
p = a(2.3); // Taylor級数aの変数に2.3を
```

```
// 代入し、値を計算する。
```

と書く。このとき、注意しなければならないことは、Taylor級数はすべて原点で展開していると仮定して計算するので、もし展開位置が原点でない場合には、利用者が位置を調節しなければならないことである。上の例で、もし展開位置が $x=1$ ならば、 $x=2.3$ における関数値を計算するには、

```
p = a(1.3); // 展開位置を考慮して代入
```

```
//する値を決めなければならない。
```

とすることがある。

簡単な例として、次の関数の $x=1$ における次の関数 $f(x)$ のTaylor級数を計算する。

$$f(x) = \sqrt{x}e^x + \sin x \quad (3.1)$$

この計算は簡単で、次のようになる。

```
1: #include "taylor.h" // このファイルで
```

```
2: main() // taylorを定義している。
```

```
3: {
```

```
4:     taylor f, x; //fとxのtaylor変数を宣言
```

```
5:     set_degree( 5 ); // 計算を5次に指定
```

```
6:     x[0]=1; // xのTaylor級数の
```

```
7:     x[1]=1; // 係数を設定
```

```

8:   f=sqrt(x)*exp(x)+sin(x); //計算を実行
9:   cout << "f=" << f << "\n" ; //計算結
10:   return 0 ;                // 果出力
11: }

```

1 行目では、Taylor 級数 Taylor を定義するヘッダーファイルを読み込んでいる。このファイルは、Taylor 級数を使うプログラムでは必ず取り込まなければならない。4 行目では、Taylor 級数を宣言している。宣言した段階では、すべての係数が 0 になっている。5 行目で何次まで計算するかを指定する。この例では、5 次まで計算することを指定している。6 行目で変数 x の Taylor 級数展開を与える。上の問題では、展開位置は、 $x=1$ であるから

$$x = 1 + (x-1) \quad (3.2)$$

と展開される。すなわち、定数項が 1、1 次の係数が 1 である。7 行目で、関数 $f(x)$ の Taylor 級数を計算する。8 行目で関数 $f(x)$ を出力する。その出力は

```

f=3.55975+4.61773*x+1.95776*x^2
+0.872674*x^3+0.268664*x^4+0.0802463*x^5

```

となる。変数 f の中には展開位置の情報が入っていないので、原点で展開した形で出力される。このように、非常に簡単に計算することができる。出力形式は、原点での展開形式であるが、 x を $x+1$ と読み替える必要がある。

上に示したプログラムを実行するには、このプログラムをコンパイルしたものと Taylor 級数のライブラリをコンパイルしたものをリンクしなければならない。

4. Taylor 級数プログラムの数値計算への応用

これまで説明した Taylor 級数 power の数値計算への応用について述べる。ここでは、細かな部分は省略し、その応用可能性を重点的にのべる。

4.1 いろいろな関数の Taylor 級数展開

Taylor 級数を求める計算は、現在の数値計算の一部にはなっていない。これは、Taylor 級数を求める一般的な計算方法が複雑で実用的でないと見なされていたからではないかと思われる。しかしながら、Taylor 級数は数値計算の特に関数計算ではよく使われる方法である。

ここでは、gamma 関数の Taylor 級数を計算する。この関数の級数には、公式集や最近の研究に載っているものもある。これらの Taylor 級数を計算する。

計算のための公式は、簡単で通常の数値計算方法と同様に、スターリングの公式

$$\Gamma(x) \approx e^{-x} x^x \sqrt{\frac{2\pi}{x}} \left(1 + \frac{1}{12x} + \frac{1}{288x^2} + \dots \right) \quad (4.1)$$

を使う。数値計算では、この公式は、漸近展開式

なので、 x が十分に大きくないと、精度のよい計算ができないので、 x が小さな数値の場合、整数を加えて 20 より大きな値にし、その値の関数値を (4.1) の公式を使って精度よく計算する。得られた計算値を、公式

$$\Gamma(x) = \frac{\Gamma(x+1)}{x} \quad (4.2)$$

を何回か使って、もとの x に戻し、目的の値の関数値を計算する。この計算法を Taylor 級数に適用すれば、gamma 関数の Taylor 級数が得られる。

4.2. Riemann の zeta 関数 $\zeta(x)$ の計算

拡張された Euler-Maclaurin の公式の中に現れる Riemann の zeta 関数 $\zeta(x)$ は、 $x > 1$ のとき次のような公式²⁾によって計算できる。

$$\begin{aligned} \zeta(x) = & \frac{1}{1^x} + \frac{1}{2^x} + \dots + \frac{1}{N^x} + \frac{1}{(x-1)N^{x-1}} \\ & - \frac{1}{2N^x} + \frac{B_2 x}{2! N^{x+1}} + \frac{B_4 x(x+1)(x+2)}{4! N^{x+3}} \\ & + \frac{B_6 x(x+1)(x+2)(x+3)(x+4)}{6! N^{x+5}} + \dots \end{aligned} \quad (4.3)$$

ここで、 B_n はベルヌーイの定数である。 N は大きいほど精度高く計算できるが計算量は多くなる。本論文では、倍精度 (10 進 15 桁) の精度を得るために、 $N=8$ として計算している。倍精度の値を得るほぼ最小の数値である。 $x < 0$ の場合、(4.3) の公式は、このままでは使えないので、公式³⁾

$$\zeta(x) = 2^x \pi^{x-1} \sin\left(\frac{\pi}{2} x\right) \Gamma(1-x) \zeta(1-x) \quad (4.4)$$

を使って、(4.3) の公式を使えるように変形することによって行う。ここで、 $\Gamma(x)$ は gamma 関数である。この方法で計算した $x=2$ における 9 次までの展開式を以下に示す。

```

1.64493-0.523451*x+0.933425*x^2-
0.826243*x^3+0.935938*x^4-0.894696*x^5
+0.912799*x^6-0.903747*x^7+0.906043*x^8-
0.904715*x^9

```

ここで、 x は、 $x-2$ を意味する。

4.3. 特異点近くでの関数値の計算

Taylor 級数の計算方法を利用すれば、つぎの関数の $x=0$ における関数値の計算のように、見かけの特異点における関数値を桁落ちなしで容易に計算することができる。

$$f(x) = \frac{x}{e^x - 1} \quad (4.5)$$

この関数の $x=0$ における値は、通常の数値方法

では、精度良く計算することができない。 $x=0$ を代入すると、ゼロによる割り算が起こるので、0の近似値で代用して計算するが、精度の良い値は期待できない。

関数値を計算する位置 ($x=0$) で分子及び分母をTaylor展開すれば、分子と分母の定数項が0となるので、分子及び分母を x で割れば、関数値を桁落ちなしで計算できる。このように、関数の解析的な性質を使うと、大きな桁落ちが生じることなしに、計算することができる。このような問題は、数値解析にはよく現れる計算で、非常に重要である。

5. おわりに

1変数の関数を数値的に、Taylor展開する方法を述べた。この方法は、C++言語を使うと、宣言部分の変更程度で、関数を簡単にTaylor展開できる。

これを利用すれば、関数の任意の階数微分係数を高精度で計算できる。このことを利用すれば、微分を含むいろいろな公式を数値計算に利用できる。

参考文献

- [1] Corliss G. and Chang Y. F., Solving Ordinary Differential Equations Using Taylor Series, ACM Trans. Math. Soft., 8(1982), 114-144
- [2] Ellis M. A. and Stroustrup B., The Annotated C++ Reference Manual, Addison-Wesley, New York, 1990
- [3] 奥村晴彦, C言語によるアルゴリズム事典、技術評論社、1991
- [4] Abramowitz M. and Stegun I.A., Handbook of Mathematical Functions, Dover, New York, 1970
- [5] Henrici P., Applied and Computational Complex Analysis, Vol. 1, Chap. 1, John Wiley & Sons, New York, 1974
- [6] Lyness J. N. and Ninham B. W., Numerical Quadrature and Asymptotic Expansions, Math. Comp., 21(1967), 162-178
- [7] Rall, L. B., Automatic Differentiation Technique and Applications, Lecture Notes in Computer Science, Vol. 120, Springer Verlag, Berlin-Heidelberg-New York, 1981