ハイパフォーマンスコンピューティング 60－11
（１９９６．３．４）

C++言語による逆関数の級数展開

平山 弘　田口靖之

神奈川工科大学　機械システム工学科

C++言語を使うとTaylor級数の四則演算を容易に定義することができる。同様な方法で関数も容易に定義できる。この演算を利用するといろいろな関数を容易にTaylor級数に展開することができる。常微分方程式の初期値問題解もPicardの方法を使えば、容易にTaylor展開できること、関数の逆関数は簡単な常微分方程式を満たすことから、関数の逆関数のTaylor展開も容易にできる。この展開式は、容易にPadé展開できる。

この逆関数の展開式を利用すると非線形方程式の効率的な計算法を与える。このことを数値例によって示した。

Taylor Expansion of Inverse Function by C++ Language

Hiroshi Hirayama and Yasuyuki Taguchi

Kanagawa Institute of Technology

An arithmetic system for Taylor series can be defined by C++ language. Elementary mathematical functions for Taylor series can be also defined. Using these operations, many function can be expand in Taylor series. We can easily get a solution of an initial value problem in ordinary differential equations in Taylor series with Picard method. An inverse function can be expanded in Taylor series because it can be expressed with the simple ordinary differential equation.

When Taylor series of this inverse function is used, the effective way of computing non-linear equation is given. These are illustrated by numerical examples.

1. はじめに

計算機によるべき級数やTaylor級数の計算は、比較的簡単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。これらの研究については、G. Corliss and Y.F.Chang[3]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、大部分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、通常の数値計算では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である。この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、Taylor級数を利用するプログラムは限られたものになっている。この問題点に関しては、このようなプログラムの作成者も理解しており、高級言語で書かれたプログラムをべき級数の計算に変換するためのプリ・プロセッサを準備しているのもある。

プリ・プロセッサを準備しないでも、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++[4]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、汎用の言語上に作成しているため、この言語で作成されたプログラムを簡単に利用できるため、いろいろな応用プログラムの作成が容易にできる。

非線形方程式の実用的な解法には、Newton法や連続変化法[7]などがあり、代数方程式に限ると、平野法や連立法がある。この他、収束次数を自由に調整できる方法に逆関数のTaylor級数を使って求める解法がある。この方法は、手回し計算機が使われていた頃の数値解析の文献[6]の中に説明があるが、最近の文献には載らなくなった。これは、この文献にも述べられているが、関数の微分係数が容易に求められるときに有効な方法であり、数値微分では高階まで精度良く求まらないため、現在、ほとんど使われていない。逆関数のTaylor級数の各係数を求めることが複雑であることも原因の一つである。

逆関数は、微分方程式で記述すると簡単な形で表されることを利用して、常微分方程式の解法であるPicard[2]の方法を利用して、常微分方程式を解き、逆関数のTaylor級数を求められる。

使用するTaylor級数ライブラリにも微積分の機能があるため、この展開を得るためのプログラム作成は、容易にできる。また、求めた逆関数のTaylor級数の精度を高め、収束範囲などを広げるために、Padé 近似することもできる。これを使って、いろいろな方程式を解いてみた。

代数方程式の解法で、収束次数を任意に選べる方法としては、方程式をPadé 近似で有理関数に近似して、分子=0として根を求める方法もある。この場合、分子の次数は1次か2次程度にすると容易に近似根が求まる。この方法も併せて計算してみる。

この研究で使用したC++言語は、主にBorland C++ 4.5, 4.5Jである。計算機は、主にパソコンのGATEWAY2000を使用した。

2. 逆関数のTaylor級数の計算法

谷本[6]の方法では関数$f(x)$の微分を利用して、その逆関数の1階微分から2階、3階,...と逐次求めて逆関数のTaylor級数を求めている。

すなわち、$y=f(x)$の逆関数を$x=f(y)$すなわち$y=g(x)$とする。元の関数の展開位置をx_0とすると、

(2.1) $y_0=f(x_0)$、 $x_0=g(y_0)$

これを使って、$x=y_0$で逆関数をTaylor展開すると、

$$y=g(x)$$

(2.2)
$$=g(y_0)+\frac{g'(y_0)}{1!}(x-y_0)+\frac{g''(y_0)}{2!}(x-y_0)^2+\frac{g''(y_0)}{3!}(x-y_0)^3+\cdots$$

ここで、

$$g(y_{0-})=x_0$$

$$g'(y_0)=\frac{dg(x)}{dx}=\frac{dy}{dx}=\frac{1}{\frac{dx}{dy}}=\frac{1}{f'(y)}$$

(2.3)
$$g''(y_0)=\frac{d^2y}{dx^2}=\frac{dy}{dx}\frac{d}{dy}\left(\frac{1}{f'(y)}\right)=\frac{-f''(y)}{\{f'(y)\}^3}$$

$$g'''(y_0)=\frac{d^3y}{dx^3}=\frac{dy}{dx}\frac{d}{dy}\left(\frac{-f''(y)}{\{f'(y)\}^3}\right)=\frac{3\{f''(y)\}^2-f'(y)f'''(y)}{\{f'(y)\}^5}$$

このようにして逆関数のTaylor級数を求めている。

非線形方程式の根を求めるには、逆関数の式(2.2)に$x=0$を代入した次式を使う。

(2.4)
$$y=x_0-\frac{1}{f'(y_0)}y_0-\frac{f''(y_0)}{2\{f'(y_0)\}^3}(y_0)^2+\frac{f'(y_0)f'''(y_0)-3\{f''(y_0)\}^2}{6\{f'(y_0)\}^5}(y_0)^3+\cdots$$

しかし、この方法では一般項がどのようになるのか推定するのは困難である。また、簡単な関数でない限り、高階の微分係数を求めるのは難しいので、この方法を実際の問題に適用することはほとんど使われなかった。

そこで、本論文では、Picard の方法による逐次近似法で逆関数の Taylor 展開する方法を提案する。この方法は、$y=f(x)$の逆関数$x=f(y)$は、次の微分方程式

(2.5)
$$y'=\frac{dy}{dx}=\frac{1}{\frac{dx}{dy}}=\frac{1}{f'(y)}$$

を満たすことが簡単にわかるので、この方程式を初期条件$x=y_0$のとき$y=x_0$として級数解を求めれば、逆関数の Taylor 展開が得られる。この級数の計算に、Picard の方法を利用する。

ここでは、(2.5)の方程式を初期条件$x=x_0$のとき$y=y_0$として、Picardの方法で計算する方法を説明する。第一近似として、

(2.6) $y=y_0$

として、(2.5)式の右辺に代入して、両辺を積分する。すなわち、

(2.7) $$y = y_0 + \int_{x_0}^{x} \frac{1}{f'(y_0)} dx$$

を計算して、次の近似式を得る。(2.7)の式を計算すると、

(2.8) $$y = y_0 + \frac{1}{f'(y_0)}(x - x_0)$$

となり、一次までの展開式が得られる。この式を、さらに、(2.5)式の左辺に代入し、積分する。このとき、左辺の計算は、x の１次まで計算し、２次以上を無視するように計算する。これを積分して、２次の近似値を得る。このため、本来の Picard の方法とは若干異なる。一般に、i 次の逆関数の Taylor 級数を y_i として、(2.5)式の右辺に代入して、両辺を積分する。すなわち、

(2.9) $$y = y_0 + \int_{x_0}^{x} \frac{1}{f'(y_i)} dx$$

を計算する。積分の中は、I次で計算し、積分することによって、$i+1$次の展開式を得る。この計算法は、多項式を代入し、係数を決定する方法とほぼ同じ方法である。この手順をプログラムにすると次の様になる。

```
#include "taylor.h"                    // Taylor展開の定義
taylor inverse( const taylor& f )      // 引数として、逆関数を求める関数のTaylor展開
{                                      //    を与える。
  taylor y, fd ;
  int DEG = degree(f) ;
  fd = 1/diff(f) ;                     // 1/f'(y)を計算する
  for( int i=0 ; i<=DEG ; i++ )
  {
    set_degree(i) ;                    // 計算する次数を設定
    y = integrate(eval(fd, y)) ;       // 積分して、近似式を求める。
  }
  return y ;
}
```

多項式を表現する構造体には、展開位置の情報が含まれていないので、定数項（(2.9)式で y_0 に相当する。）は常に 0 になっている。逆関数をこの関数で計算する場合には、定数部分を修正する必要がある。

3. 逆関数のTaylor級数の数値例

ここでは、簡単な例を使って、逆関数の計算法を示す。以下のプログラムでは、e^x の原点における展開式を計算し、その逆関数である $\log(x)$ の展開式を計算している。

```
#include "taylor.h"                    // Taylor展開の定義
main()
{
  taylor x, y, z ;
  x[0]=0 ;
  x[1]=1 ;
  y = exp(x) ;                         // exp(x)の展開式を計算する。
  z = inverse( y ) ;                   // exp(x)の逆関数を計算する。
```

```
    cout << y << endl ;
    cout << z << endl ;
}
```

出力は、次のようになる。計算次数は、既定値の 20 次で計算されるが、ここでは、6 次まで示す。

```
exp(x)=1+x+0.5*x^2+0.166667*x^3+0.0416667*x^4+0.00833333*x^5+0.00138889*x^6
log(x)=x-0.5*x^2+0.333333*x^3-0.25*x^4+0.2*x^5-0.166667*x^6
```

log(x)の展開式の x は、$x-1$ を意味する。

4. Padé展開の計算方法

Taylor 級数を分子、分母を l、m 次の多項式の商である有理関数 $P_l(x)/Q_m(x)$（Padé 展開）に展開することは、しばしば、元の Taylor 級数の収束範囲を広げ、収束を良くすることがある。このような展開は、次のような式を満たすことを要求することによって、決定される。

$$f(x)=\sum_{i=0}^{\infty}c_i x^i=\frac{a_0+a_1x+a_2x^2+\cdots+a_lx^l}{b_0+b_1x+b_2x^2+\cdots+b_mx^m}+O(x^{l+m+1}) \tag{4.1}$$

ただし、一般性を失うことなく、$b_0=1$ と置くことができる。(4.1)の両辺に右辺の分母を掛けると

$$\begin{aligned}(b_0+b_1x+b_2x^2+\cdots+b_mx^m)(c_0+c_1x+c_2x^2+\cdots)\\ =a_0+a_1x+a_2x^2+\cdots+a_lx^l+O(x^{l+m+1})\end{aligned} \tag{4.2}$$

となる。(4.2)式において、$x^{l+1},x^{l+2},\cdots,x^{l+m}$ の係数を比較することによって、$b_1,b_2,\cdots,b_m$ に関する次の連立一次方程式が得られる。

$$\begin{cases}c_{l-m+1}b_m+c_{l-m+2}b_{m-1}+c_{l-m+3}b_{m-2}+\cdots+c_lb_1 &= -c_{l+1}\\ c_{l-m+2}b_m+c_{l-m+3}b_{m-1}+c_{l-m+4}b_{m-2}+\cdots+c_{l+1}b_1 &= -c_{l+2}\\ \cdots\\ c_lb_m+c_{l+1}b_{m-1}+c_{l+2}b_{m-2}+\cdots+c_{l+m-1}b_1 &= -c_{l+m}\end{cases} \tag{4.3}$$

この方程式を解くことによって、(4.1)の分母 $b_1,b_2,\cdots,b_m$ が決定される。この連立一次方程式は、Toeplitz 行列なので、縁取り法を使えば高速に解くことができる。分子の係数 $a_0,a_1,\cdots,a_l$ は、$x^0,x^1,x^2,\cdots,x^m$ の係数を比較したときに得られる関係式

$$\begin{aligned}a_0&=c_0\\ a_1&=c_1+b_1c_0\\ a_2&=c_2+b_1c_1+b_2c_0\\ &\cdots\\ a_l&=c_l+\sum_{i=1}^{l}b_ic_{l-i}\end{aligned} \tag{4.4}$$

から得られる。

5. 応用例

ここでは、4 次の逆関数およびその逆関数を分子、分母共に 2 次に Padé展開したものを利用して、(1) $(x-1)(x-2)=0$ (2) $e^{-x}\cos x=0$ を初期値 0 として解いた様子を表 1 に示

す。 Padé展開した方が収束が速いことがわかる。

表１　反復計算による方程式の収束状況

方程式	４次の逆関数	[2/2]の Padé展開
$(x-1)(x-2)=0$	0.917238225880201	0.917238225880201
	0.999974517713750	0.999999641771778
	1	1
$e^{-x}\cos x=0$	1.166666666666667	1.166666666666667
	1.561528532874529	1.568030254066273
	1.570796326321629	1.570796326794897
	1.570796326794897	1.570796326794897

6. 終わりに

C++言語を利用すると、Taylor級数の計算を簡単にできる。これを利用すると任意の次数で常微分方程式の初期値問題を解くことができる。これを利用すると、逆関数のTaylor展開、さらに、 Padé展開を得ることができる。これを利用すると、非線形方程式の収束の良い計算法が得られる。

参考文献

[1] Abramowitz M. and Stegun I.A., Handbook of Mathmatical Functions, Dover, New York, 1970

[2] 赤坂隆、数値計算、コロナ社、1967

[3] Corliss G. and Chang Y. F., Solving Ordinary Didderential Equations Using Taylor Series, ACM Trans. Math. Soft. Vol. 8, No. 2, 1982

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

[5] 三井、数値解析入門、朝倉書店、1985

[6] 谷本勉之助、実用数値計算法、森北出版, 1958.

[7] 杉原正顯、室田一雄、数値計算法の数理、岩波書店, 1994