ハイパフォーマンス
コンピューティング 55－3
（１９９５．３．９）

C++言語による常微分方程式の解法

平山　弘

神奈川工科大学　機械システム工学科

C++言語を使うとべき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べき級数に展開することができる。常微分方程式の解の存在を証明するときに使われる Picard の方法を利用することによって、常微分方程式の初期値問題の解をベキ級数の形で得ることができる。

各ステップ毎に、その点における解のべき級数を計算し、最良の刻み幅と誤差評価を行うことができる。さらに得られたべき級数を連分数展開や Pade 展開を行うと、A 安定な常微分方程式の解法が得られる。

Solving Ordinary Differential Equations using C++ Language

Hiroshi Hirayama

Kanagawa Institute of Technology

An arithmetic system for power series can be defined by C++ language. Elementary mathematical functions for power series can be also defined. Using these operations, many function can be expand in power series. We can get a solution of an initial value problem in ordinary differential equations in power series with Picard method that is a method to proof existence of solution for it.

At each step of the integration, the program generates the series of the solution and analyzes that series to determine the optimal step and to estimate truncation error. When we expand these series in continued fraction or Pade series, we can get A-stable method for an ordinary differential equation.

1. はじめに

計算機によるべき級数やTaylor級数（以下では特に述べない限り、両級数をべき級数と言う。）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[2]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、べき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、べき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、べき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・プロセッサが準備されているとしても、そのプリ・プロセッサが十分に一般的で使い易いものでなければ、それほど使われることはないであろう。プリ・プロセッサに精通していなければ、プリ・プロセッサによる変換作業中に起こるいろいろな問題に対処出来ないようでは、一般に使われることはないと思われる。また、そのプリ・プロセッサが定義する言語が、オペレーティング・システムや他のアプリケーションと調和していなければ、それほど使われることはないであろう。べき級数による計算が数値計算で使われなかった理由は、この点にあったのではないかと思われる。

このような観点から、プリ・プロセッサを準備しないでも、べき級数の計算を行うことができるプログラミング言語 C++[3]が一般に使われるようになって来たのを機会に、べき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++言語で作成されたべき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。

このプログラムを利用した数値計算は、今までの数値計算とはかなり毛色の違ったものになっている。このような場合、今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。

べき級数のプログラムは短く簡単に作成できるが、その応用は非常に数値計算の広範囲にわたる。たとえば、関数計算、極限計算、数値積分、常微分方程式、方程式の解法ような広い範囲の応用が考えられる。さらに、研究を進めれば、他の応用がいろいろ考えられると思われる。

本論文では、べき級数を使って常微分方程式の初期値問題について述べ、その有効性について述べる。この方法では、初期条件から関数をその時点のべき級数に展開する。そのべき級数に数値を代入し、次の時点の関数の値を計算する。その値を初期条件として何回も続ければ常微分方程式の解法になることを示す。さらに、得られたべき級数を連分数展開をすれば、A安定な常微分方程式の解法になることを示す。この方法は、A安定な解法である陰的ルンゲ・クッタ法に比べ、A安定の計算方法であるにもかかわらず、計算の途

中に非線形連立方程式を解く必要がないので、非常に効率的である。また、その考え方も非常に単純なので大変使いやすいものである。

この研究で使用したC++言語は、主にBorland C++ 3.1, 4.0Jである。計算機は、NEC PC-9821AP2を使用した。

2. べき級数の計算

べき級数は、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、べき級数を$x=a$で展開したときの式を

(2.1) $\mathrm{f}(x)=\mathrm{f}_0+\mathrm{f}_1(x-a)+\mathrm{f}_2(x-a)^2+\mathrm{f}_3(x-a)^3+\mathrm{f}_4(x-a)^4+\cdots$

で表現するとき、この中のm個を取ったもの

(2.2) $\mathrm{f}_0,\ \mathrm{f}_1,\ \mathrm{f}_2,\ \mathrm{f}_3,\ \mathrm{f}_4,\ \cdots,\ \mathrm{f}_m$

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

べき級数は、C言語の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。最初のべき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。この値は計算可能べき級数の最大次数に1を加えた値になる。この次数以下ならば、べき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。

以下で示すべき級数の演算は、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、べき級数の定数倍なども定義しているが、容易に導けるので省略してある。

2.1 べき級数の四則演算

べき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。

(2.3) $\mathrm{f}(x)=\mathrm{f}_0+\mathrm{f}_1 x+\mathrm{f}_2 x^2+\mathrm{f}_3 x^3+\mathrm{f}_4 x^4+\cdots$

(2.4) $\mathrm{g}(x)=\mathrm{g}_0+\mathrm{g}_1 x+\mathrm{g}_2 x^2+\mathrm{g}_3 x^3+\mathrm{g}_4 x^4+\cdots$

(2.5) $\mathrm{h}(x)=\mathrm{h}_0+\mathrm{h}_1 x+\mathrm{h}_2 x^2+\mathrm{h}_3 x^3+\mathrm{h}_4 x^4+\cdots$

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。mは、演算の対象となっているべき級数の次数である。

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

このとき、係数は次の式によって計算することができる。

(2.6) $\mathrm{h}_n=\mathrm{f}_n\pm\mathrm{g}_n \qquad (n=0,\cdots,m)$

（2）乗算　$\mathrm{h}(x)=\mathrm{f}(x)\mathrm{g}(x)$

このとき、係数は次の式によって計算することができる。

(2.7) $\mathrm{h}_n=\sum_{k=0}^{n}\mathrm{f}_k\,\mathrm{g}_{n-k} \qquad (n=0,\cdots,m)$

（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、$f_0=0$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。

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

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。

2.2 べき級数の関数計算

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べき級数による解法の例となっている。

（4）ベキ乗　$h(x)=f(x)^{\alpha}$　ここで、αは定数である。

このとき、この関数は、つぎの微分方程式を満たす。

(2.9)　$$f(x)\frac{dh(x)}{dx}=a\,h(x)$$

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。

(2.10)　$$h_0=f_0^{\alpha},\quad h_n=\frac{1}{n f_0}\sum_{k=1}^{n}\{(\alpha+1)k-n\}f_k\,h_{n-k}\qquad(n=1,\cdots,m)$$

$a=\dfrac{1}{2}$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、計算結果はべき級数になる。たとえば、

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

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。

（5）指数関数　$h(x)=e^{f(x)}$

このとき、この関数は、次の微分方程式を満たす。

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

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。

(2.13)　$$h_0=e^{f_0},\quad h_n=\frac{1}{n}\sum_{k=1}^{n}k\,h_{n-k}\,f_k\qquad(n=1,\cdots,m)$$

（6）対数関数　$h(x)=\log f(x)$

このとき、この関数は、次の微分方程式を満たす。

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

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。

(2.15) $$h_0=\log f_0,\qquad h_n=\frac{1}{nf_0}\left(nf_n-\sum_{k=1}^{n-1}k\,h_k\,f_{n-k}\right)\qquad(n=1,\cdots,m)$$

（7）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。

(2.16) $$\frac{dg(x)}{dx}=h(x),\quad \frac{dh(x)}{dx}=-g(x)$$

この式から、係数に対する次のような関係式が得られる。

$$g_0=\sin f_0,\qquad h_0=\cos f_0$$

(2.17) $$g_n=\frac{1}{n}\sum_{k=1}^{n}k\,h_{n-k}f_k,\qquad h_n=-\frac{1}{n}\sum_{k=1}^{n}k\,g_{n-k}f_k\qquad(n=1,\cdots,m)$$

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。

（8）微分　$h(x)=\dfrac{df(x)}{dx}$

この定義式から、次のような関係式が得られる。

(2.18) $$f_m=0,\qquad h_n=(n+1)f_{n+1}\quad(n=0,\cdots,m-1)$$

このように、最高次数の係数f_mは、0となる。

（9）積分　$h(x)=\int_0^x f(t)dt$

この定義式から、次のような関係式が得られる。

(2.19) $$h_0=0,\qquad h_n=\frac{1}{n}f_{n-1}\qquad(n=1,\cdots,m)$$

3. べき級数プログラムの簡単な使用例

ここでは、べき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。

べき級数は、次のように宣言される。

```
power   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、べき級数は 0 に初期化される。これらのべき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、べき級数は、

```
a +=  b ;   // べき級数 a と b を加え、べき級数 a にいれる。
a = exp(b); // べき級数 b の指数関数を計算し、べき級数 a にいれる。
```

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。べき級数に値を代入したい場合、

```
p = a(2.3); // べき級数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、べき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。
```

とする必要がある。

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

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

この計算は簡単で、次のようになる。

```
 1: #include "power.h"          このファイルでpowerを定義している。
 2: main()
 3: {
 4:     power f, x ;                    f とx のpower 変数を宣言
 5:     set_degree( 5 ) ;               計算を5次に指定
 6:     x[0]=1 ; x[1]=1 ;               x のべき級数の係数を設定
 7:     f=sqrt(x)*exp(x)+sin(x) ;       計算を実行
 8:     cout << "f=" << f << "\n" ;     計算結果出力
 9:     return 0 ;
10: }
```

1行目では、べき級数 power 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、べき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、べき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のべき級数展開を与える。上の問題では、展開位置は、$x=1$であるから

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

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$f(x)$のベキ級数を計算する。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。

4. 常微分方程式の計算

常微分方程式として、次のような初期値問題を考える。

(4.1) $$\frac{dy}{dx}=f(x,y) \quad x=0 \text{のとき} y=y_0$$

このとき、yを(2.3)のように展開できるとして、(4.1)に代入することによって、yの展開係数を決定できる。この展開式に適当な刻み幅hを代入することによって、x=hにおけるyが決まる。この値を初期値として、もう一度同じ操作を繰り返せば、x=2hのときの値が求められる。この操作を繰り返せば、常微分方程式を解くことができる。

この方法は、常微分方程式の解の存在を証明するときに使われる Picard の方法とほぼ同じになる。Picard の方法は、(4.1)に対して

(4.2) $$y=y_0$$

として、漸化式

(4.3) $$y_{n+1}=y_0+\int_0^x f(t,y_n)dt$$

によって、次々と精度を上げる方法である。この方法にべき級数を代入して計算すると、1回の操作で1次の精度が上がる。計算は、何次式でも計算できるが、計算量を少なくするために、通常得られる精度の次数のべき級数で計算する。ただし、次のように問題に特異性がある場合には、得られる精度より高次の級数で計算する必要がある。たとえば、

$$\text{(4.4)}\qquad \begin{cases} \dfrac{dy}{dx} = z \\ \dfrac{dz}{dx} = -\dfrac{z}{x} - y \end{cases} \qquad x = 0 \text{ のとき } y = 1, z = 0$$

この方程式の解 y は、0 次のベッセル関数 $J_0(x)$ である。この問題は、通常の微分方程式の数値解法である Runge-Kutta 法では解けない問題である。このように C++言語使ってべき級数を使って解く方法は、任意の次数で広い範囲の問題を解くことができる。

5. A安定な計算方法

常微分方程式の線形安定理論では、方程式

$$\text{(5.1)}\qquad \frac{dy}{dx} = \lambda y \qquad \Re\lambda < 0$$

を解いたとき、どのような刻み幅を入れて計算を進めても絶対値が 1 を越えない計算スキームを A 安定であると言う。

これは、級数展開した式にどんな刻み幅を入れても絶対値が 1 より小さいことを意味する。この条件はべき級数では満たされないので、ここでは、連分数展開を行う。(5.1)を解くと

$$\text{(5.2)}\qquad y = e^s \qquad \text{ここで} s = \lambda x$$

この式を連分数展開[1]すると、

$$\text{(5.3)}\qquad e^s = \cfrac{1}{1 - \cfrac{s}{1 + \cfrac{s}{2 - \cfrac{s}{3 + \cfrac{s}{2 - \cfrac{s}{5 + \ddots}}}}}}$$

を利用する。この式を途中で打ち切り、簡単化すると Pade 展開式になる。この展開式に一致するように、4 で計算されたべき級数を変換しするには、連立 1 次方程式を解けば簡単に得られる。(5.3)式を使った計算方法が A 安定であることを証明するには、$\Re s < 0$ で(5.3)式の絶対値を 1 より小さいことを証明しなければならない。三井[4, 7 章]によれば、(5.3)式を Pade 展開したとき、分子と分母が同じ次数の場合すなわち偶数次の場合には、A 安定の条件を満たすことが証明されているから、ここでは、分母が分子より 1 次だけ大きい場合を調べればよい。s の絶対値が十分大きい場合は明らかであるから、原点の近くでの振る舞いを調べる。ここでは、数値的方法を使って調べた。この結果 23 次まで、A 安定の条件を満たすことが確かめることができた。プログラムの制限上 23 次までしか調べられなかったが、奇数次においてもさらに高い次数で A 安定の条件を満たしていると思われる。この計算法を利用すれば、A 安定な高次の計算方法が得られたことになる。

6. 終わりに

C++言語を利用すると、べき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。さらに解はべき級数の形で得られるので、それを連分数展開などを行えば数値的に安定な計算方法が得られる。

本方法は、常微分方程式の解法である Runge-Kutta 法と比較すると、次数を同じにした場合、問題にもよるが、計算時間はほぼ同程度である。メモリーは多量に使うので非常に大きな問題を除けば、常微分方程式の有力な計算方法となる。

参考文献

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

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

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

[4] 三井、数値解析入門、朝倉書店、1985