数理解析研究所講究録 1335 巻 2003 年 49-56

微分代数方程式の解のべき級数展開法

平山　弘 (Hiroshi Hirayama)
神奈川工科大学*

1　はじめに

関数をべき級数展開（Taylor 級数展開）する方法として、単純な式のべき級数展開式を与え、それを計算することによって、複雑な式をべき級数展開する方法がある。この方法を使うと、多くの関数を容易にべき級数展開することができる。ここで使うべき級数は、係数を浮動小数点数で表現するもので非常に高速にできる。この方法は、数式処理的な方法であるが、厳密性を追求する数式処理とは異なり、通常の数値計算法と同程度の速さで計算できる。

このような計算には、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機能、C++言語で言うオペレータ・オーバロード（operator overload）を使うのが便利である。

この方法を使うと常微分方程式の解を、任意の次数までべき級数展開できる。数学的には、常微分方程式の級数展開による解法と同じであり、Corliss and Chang[1] によって、任意の次数のべき級数展開式を利用した常微分方程式を解くためのパッケージ・ソフトウエアが発表されている。この計算方法は、A 安定化することもでき、その計算速度が通常の Runge-Kutta 法と同程度である [6] ことが示された。

この高速性を利用すると、べき級数を使い数値積分を効率的に計算することができる。次のような積分の計算することを考える。

$$I = \int_a^b f(x)dx \tag{1}$$

この積分計算は、微分方程式

$$\frac{dy}{dx} = f(x) \qquad y(a) = 0 \tag{2}$$

の初期値問題を数値的に解き、$y(b)$ を求める問題に相当する。この計算法として、べき級数を使った計算法を適用するとガウス型数値積分法や二重指数型数値積分法等に代表される通常の効率的な数値積分法と同程度の時間で計算できる。

特に、宮広、野田 [7] によって指摘されている、通常の数値積分法 [3]（シンプソン、ロンバーグ、ガウス型、二重指数型、適応型ニュートンコーツ）ではうまく計算できない積分

$$I = \int_0^1 \frac{-1}{x^5 - x^4 - 0.75x^3 + x^2 - 0.25x - 10^{-6}} dx \tag{3}$$

も容易に計算できることを示すことができる。

本論文では、この計算法を拡張し、未定係数を含むべき級数の演算を定義し、それを利用して微分代数方程式の解も容易にべき級数展開できること示した。微分代数方程式のこの種の計算法については、Chang

*hirayama@sd.kanagawa-it.ac.jp

and Corliss[2] によっても与えられているが、本方法は、より一般的な方程式を解くことができる。べき級数展開された解を、Padé 展開式に変形すれば、任意次数の A 安定な数値計算方法を与えることを示す。

本計算では、C++言語（Borland C++ Builder Ver.6) と Fortran95（富士通製）を使用した。実行時間測定には、2.0GHz の Pentium4 を使用した。

2 べき級数の計算

この節では、関数をべき級数展開するための基本的な考え方を説明し、その計算方法について論じる。詳しくは、Rall[8] や平山 [5] などに述べられている。

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。

$$f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + \cdots \tag{4}$$

$$g(x) = g_0 + g_1 x + g_2 x^2 + g_3 x^3 + \cdots \tag{5}$$

$$h(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3 + \cdots \tag{6}$$

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少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べき級数の次数である。

2.1 和差積 $h(x) = f(x) \pm g(x)$ $h(x) = f(x)g(x)$

べき級数の和差の係数および積の係数はそれぞれ次のように計算される。

$$h_n = f_n \pm g_n, \qquad h_n = \sum_{k=0}^{n} f_k g_{n-k} \qquad (n = 0, \cdots, m) \tag{7}$$

2.2 除算 $h(x) = \frac{f(x)}{g(x)}$

べき級数の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。係数は次のように計算される。

$$h_0 = \frac{f_0}{g_0} \qquad h_n = \frac{1}{g_0}(f_n - \sum_{k=0}^{n-1} h_k g_{n-k}) \qquad (n = 1, \cdots, m) \tag{8}$$

上のような計算だけでなく、べき級数と定数との計算が定義されているが、ここでは省略する。

3 べき級数の関数および微積分

べき級数の関数計算は、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。ここでは、指数関数の計算法を示す。

3.1 指数関数 $h(x) = e^{f(x)}$

この式は、次の微分方程式を満たす。

$$\frac{h(x)}{dx} = h(x)\frac{df(x)}{dx} \tag{9}$$

この式の両辺に、(4)、(5)、(6) の式を代入して、各次数の係数を比較して、次の関係式が得られる。

$$h_0 = e^{f_0}, \qquad h_n = \frac{1}{n}\sum_{k=1}^{n} k h_{n-k} f_k \quad (n = 1, \cdots, m) \tag{10}$$

3.2 微分 $h(x) = \frac{df(x)}{dx}$

この定義式から、次のような関係式が得られる。

$$h_m = 0, \qquad h_n = (n+1) f_{n+1} \quad (n = 0, \cdots, m-1) \tag{11}$$

3.3 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。

$$h_0 = 0, \qquad h_n = \frac{1}{n} f_{n-1} \quad (n = 1, \cdots, m) \tag{12}$$

定数項 h_0 は、積分定数なので、任意で良いが、ここでは、0 と定義する。

4 べき級数プログラムを利用した計算例

ここでは、C++言語で作成したべき級数のプログラムの使用例を示す。
簡単な例として、次の関数の $x = 2$ におけるべき級数を計算する。

$$f(x) = \sqrt{x}e^x + \sin x \tag{13}$$

この計算は簡単で、次のようなプログラムで計算される。

```
 1: #include "power.h"      // このファイルでべき級数を定義する。
 2: void main()
 3: {
 4:    set_degree( 5 );          // 計算を 5 次に指定
 5:    power f, x;               // f と x のべき級数変数を宣言
 6:    x[0]=2 ;                    // x をべき級数に展開   x[0] は定数項
 7:    x[1]=1 ;                    //    x[1] は 1 次の係数
 8:    f=sqrt(x)*exp(x)+sin(x);     // 計算を実行
 9:    cout << "f=" << f << "\n" ;  // 計算結果出力
10: }
```

1 行目では、べき級数を定義するヘッダーファイル (power.h) 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。5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。6、7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、$x=2$であるから$x=2+(x-2)$と展開される。すなわち、定数項が2、1次の係数が1である。この結果を6, 7行目で与えている。8行目で、関数$f(x)$のべき級数を計算する。9行目で関数のべき級数を出力する。その計算結果は

```
f=11.359+12.646*x+7.05608*x^2+2.87227*x^3+0.801546*x^4+0.162275*x^5
```

となる。このように、簡単に計算することができる。べき級数を表現する変数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。出力形式は、原点での展開形式になるので、xを$x-2$と読み替える必要がある。

この数値的べき級数展開の計算時間は、100万回ループさせて測定すると、1回あたり0.0033msecであった。同じ計算を数式処理言語(Mathmatica 4.1)を使って計算すると、実行時間は1000回ループさせ測定すると、1回あたり1.7msecであった。この時間は、C++言語を使った場合の約515倍である。

数式処理的なアルゴリズムを使っても、浮動小数点演算を使えば、高速に計算できることがわかる。

5 微分代数方程式の解のべき級数展開

べき級数展開の方法を利用して、常微分方程式の解を任意の次数までべき級数展開する方法を示す。べき級数展開された解は、任意次数の一段公式として、Runge-Kutta公式の代わりとして利用できる。また、解の許容誤差を与えたとき、その範囲で最良の刻み幅を容易に決めることもできる。

代数微分方程式は、次の形式を持つものと仮定する。

$$F(x,y_1,y_1',\cdots y_1^{(n_1)},y_2,y_2',\cdots y_2^{(n_2)},\cdots,y_m,y_m',\cdots y_m^{(n_m)})=0 \tag{14}$$

初期条件は、次のように与えられているものとする。

$$\begin{aligned} &y_1(x_0)=y_{1,0},\ y_1'(x_0)=y_{1,1},\ \cdots,\ y_1^{(n_1-1)}(x_0)=y_{1,n_1-1},\\ &y_2(x_0)=y_{2,0},\ y_2'(x_0)=y_{2,1},\ \cdots,\ y_2^{(n_2-1)}(x_0)=y_{2,n_2-1},\\ &\qquad\cdots\\ &y_m(x_0)=y_{m,0},\ y_m'(x_0)=y_{m,1},\ \cdots,\ y_m^{(n_m-1)}(x_0)=y_{m,n_m-1} \end{aligned} \tag{15}$$

ここで、$y_k^{(j)}$はy_kのj階微分を意味する。Fはm次のベクトルで、十分なめらかで必要な回数だけ微分可能とする。(14)の解は、次のような解を持つと仮定する。

$$y_k=y_{k,0}+y_{k,1}x+\cdots+y_{k,n_k-1}x^{n_k-1}+e_kx^{n_k}\quad(k=1,\cdots,m) \tag{16}$$

ここで、e_kはこれから決定すべき未定係数である。(16)を(14)に代入し、未定係数を含む項よりxについて高次の項を省略しながら計算し、その計算結果を0と置くことによって、e_kに対する連立方程式が得られる。この場合、最初に得られる連立方程式は一般に非線形方程式であるが、多くの場合、連立一次方程式である。この方程式を解くことによってe_kを決定することができる。この係数をy_{k,n_m}と置くと、(14)の解の次の次数の解を、さらに次のように仮定することができる。

$$y_k=y_{k,0}+y_{k,1}x+\cdots+y_{k,n_k-1}x^{n_k-1}+y_{k,n_k}x^{n_k}+e_kx^{n_k+1}\quad(k=1,\cdots,m) \tag{17}$$

ここで、e_kはこれから決定すべき未定係数である。(16)と同じ記号e_kを用いているが一般に異なる係数である。x^n次の時と同じように、(17)を(14)に代入し、未定係数を含む項よりxについて高次の項を省略しながら計算し、計算結果を0と置くと、e_kに対するm元連立一次方程式が得られる。

$y_{k,n}$ を決定する最初の連立方程式は一般に非線形連立方程式であるが、それより高次の係数を決定するために解く必要がある連立方程式は m 元連立一次方程式である。次の次数の係数を決定するために、解を (17) の式のように仮定して、(14) 式に代入し未定係数を含む項より高次の項を省略しながら計算し、その計算結果を 0 と置くことによって、e_k すなわち解の高次の係数を決定することができる。これを必要な回数繰り返すことによって、(14) の解のべき級数を任意次数まで計算することができる。

上のような計算を行うために、次のような最高次数の係数に未定係数 e_1、e_2、$\cdots$、e_m を含むべき級数の演算を定義した。

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + (f_n + p_1 e_1 + \cdots + p_m e_m) x^n \tag{18}$$

この級数は、x^k の係数 $f_k (k = 1, \cdots, n)$ を表す n 個の配列と e_k の係数 $p_k (k = 1, \cdots, m)$ を表す m 個の配列からなる。この級数の演算は、前のべき級数と類似の方法で定義できる。以下にその計算法を示す。

6 未定係数を含むべき級数展開式の計算

べき級数と類似の方法で未定係数を含むべき級数展開式の計算を定義できる。この計算では、未定係数を含む項より高次の項は省略している。代表的な公式を次に示す。

6.1 未定係数を含む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。通常のべき級数と同じように平行移動によって、展開位置を原点へ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。

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + (f_i + p_1 e_1 + \cdots + p_m e_m) x^i \tag{19}$$

$$g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + (g_j + q_1 e_1 + \cdots + q_m e_m) x^j \tag{20}$$

$$h(x) = h_0 + h_1 x + h_2 x^2 + \cdots + (h_k + r_1 e_1 + \cdots + r_m e_m) x^k \tag{21}$$

6.1.1 加減算

$h(x)$ が $f(x)$ と $g(x)$ の和差のとき、f, g および h の係数は、次のような関係になる

$$h(x) = f(x) \pm g(x) \qquad h_n = f_n \pm g_n \qquad (n = 0, \cdots, k) \tag{22}$$

ここで、$k = min(i, j)$ である。未定係数は、$i = j$ ならば、

$$r_n = p_n + q_n \qquad (n = 1, \cdots, m) \tag{23}$$

となる。もし、$i > j$ および $i < j$ ならば、それぞれ次のようになる。

$$r_n = q_n, \qquad r_n = p_n \qquad (n = 1, \cdots, m) \tag{24}$$

6.1.2 乗算

$h(x)$ が $f(x)$ と $g(x)$ の積のとき、f, g および h の係数は、次のような関係になる

$$h(x) = f(x)g(x) \qquad h_n = \sum_{s=0}^{n} f_s g_{n-s} \qquad (n = 0, \cdots, k) \tag{25}$$

ここで、$k = min(i, j)$ である。未定係数は、$i = j$ ならば、

$$r_n = g_0 p_n + f_0 q_n \qquad (n = 1, \cdots, m) \tag{26}$$

となる。もし、$i > j$ および $i < j$ ならば、それぞれ次のようになる。

$$r_n = f_0 q_n, \qquad r_n = g_0 p_n \qquad (n = 1, \cdots, m) \tag{27}$$

6.1.3 除算

$h(x)$ が $f(x)$ の逆数のとき、すなわち

$$h(x) = \frac{1}{f(x)} \tag{28}$$

であるとき、$f(x)$ と $h(x)$ は同じ次数（$k = i$）になり、係数には、以下のような関係が成り立つ。

$$h_0 = \frac{1}{f_0} \qquad h_n = -\frac{1}{f_0} \sum_{s=0}^{n-1} h_s f_{n-s} \qquad (n = 0, \cdots, k) \tag{29}$$

未定係数は、

$$r_n = \frac{h_0}{f_0} p_n \qquad (n = 1, \cdots, m) \tag{30}$$

となる。除算は、この逆数を掛けることによって計算する。

6.2 未定係数を含むべき級数の関数演算

べき級数の関数計算も通常のべき級数のように定義できる。ここでは指数関数の計算法を示す。

6.2.1 指数関数

$h(x) = e^{f(x)}$ とおくと

$$\frac{dh}{dx} = h\frac{df}{dx} \tag{31}$$

が成り立つ。この微分方程式から、次のような関係が得られる。

$$h_0 = e^{f_0} \qquad h_n = \frac{1}{n} \sum_{s=1}^{n} s h_{n-s} f_s \qquad (n = 0, \cdots, i) \tag{32}$$

未定係数部分は、次のようになる。

$$r_n = h_0 p_n \qquad (n = 1, \cdots, m) \tag{33}$$

7 簡単な数値例

簡単な例として、Chang and Corliss[2] によって扱われている振り子の方程式を解く。ここで、未知関数は、x 、y 、λ であり、時間 t の関数である。

$$\begin{cases} \frac{d^2x}{dt^2} = -\lambda x \\ \frac{d^2y}{dt^2} = -\lambda y + 9.8 \\ x^2 + y^2 = 1 \end{cases} \tag{34}$$

初期条件は

$$\begin{cases} x(0) = \sin(1.2) & y(0) = -\cos(1.2) \\ x'(0) = 0 & y'(0) = 0 \end{cases} \tag{35}$$

この方程式の解を (16) に従って、初期条件から、つぎのように仮定できる。

$$\begin{cases} x(t) &= \sin(1.2) + e_1 t^2 \\ y(t) &= -\cos(1.2) + e_2 t^2 \\ \lambda(t) &= e_3 \end{cases} \tag{36}$$

ここで、e_1 、e_2 、e_3 は、これから決定する定数である。(36) 式を (34) に代入すると

$$\begin{cases} 2e_1 + 0.932039e_3 = 0 \\ 2e_2 - 0.362358e_3 - 9.8 = 0 \\ 1.86408e_1 - 0.724716e_2 = 0 \end{cases} \tag{37}$$

この方程式を解くことによって

$$\begin{cases} x(t) = 0.932039 + 1.65488\mathrm{t}^2 \\ y(t) = -0.362358 + 4.25661\mathrm{t}^2 \\ \lambda(t) = -3.55111 \end{cases} \tag{38}$$

が得られる。これから、次の次数の解を次のように仮定する。

$$\begin{cases} x(t) = 0.932039 + 1.65488\mathrm{t}^2 + e_1 t^3 \\ y(t) = -0.362358 + 4.25661\mathrm{t}^2 + e_2 t^3 \\ \lambda(t) = -3.55111 + \mathrm{e}_3 t \end{cases} \tag{39}$$

この (39) 式を (34) に代入することによって、e_1 、e_2 、e_3 を決定できる。これを繰り返すことによって、任意の次数の解が得られる。たとえば、x 、y の 8 次および λ の 6 次までの展開式は

$$\begin{cases} x(t) = 0.932039 + 1.65488\mathrm{t}^2 - 9.23024\mathrm{t}^4 - 12.5981\mathrm{t}^6 + 22.9718\mathrm{t}^8 \\ y(t) = -0.362358 + 4.25661\mathrm{t}^2 + 5.03856\mathrm{t}^4 - 15.3707\mathrm{t}^6 - 26.4184\mathrm{t}^8 \\ \lambda(t) = -3.55111 + 125.144\mathrm{t}^2 + 148.134\mathrm{t}^4 - 451.899\mathrm{t}^6 \end{cases} \tag{40}$$

となる。(40) の式の t に時間刻み幅 Δt を代入することによって、次のステップにおける x、y を与える。これを初期条件として、x、y および λ の $t = \Delta t$ におけるべき級数展開式を同じ手順で求める。これを繰り返すことによって、(34) の方程式を解くことができる。

多くの数値計算法では、(34) のような方程式は、1 階連立微分方程式に変形して解かなければならない。 ここで示した方法でも、同じように変形し解くことが可能であるが、このように変形すると、(37) に相当する方程式は、5 元連立一次方程式となり計算量が増え計算速度が遅くなる。1 階連立微分方程式に変形しないで、2 階ののままで高速に解けることが本手法の特徴の一つである。このため、この例でも 1 階連立微分方程式に変形しないで、2 階微分方程式のまま解いている。

8 まとめ

C++言語を使うと、その関数の数値計算用プログラムの宣言部分の変更程度で、容易にべき級数展開できる。この計算は通常の数値計算と同等の速さで計算できる。

これを利用すれば、常微分方程式の解を任意の次数まで、べき級数展開できる。解がべき級数展開であるため、その扱いは非常に容易であり、計算結果の誤差評価、許容誤差を与えたときの最良の刻み幅の決定などに利用できる。さらに、このべき級数展開を Padé 展開することによって、常微分方程式の初期値問題を解く、任意次数の A 安定な公式 [4] を得ることができる。

さらに未定係数を含むべき級数の演算を定義し、それを利用すれば、微分代数方程式の解を任意の次数まで求めることができる。

本方法は、見かけは数式処理的であるが、計算効率は通常の数値計算程度である。通常の数値計算と比較すると得手不得手な計算があるので、どちらが効率的とは言えないがどちらも同じ程度の速度で計算できる。計算次数が任意に取れるので、この性質を利用すると多くの場合、通常の数値計算より効率的に計算できる場合が多い。

参 考 文 献

[1] Corliss G. and Chang Y. F., Solving Ordinary Differential Equations Using Taylor Series, ACM Trans. Math. Soft., Vol. 8, 114–144(1982)

[2] Chang Y. F. and Corliss G., ATOMFT: Solving ODEs and DAE Using Taylor Series, Computers Math. Applic., Vol. 28, 209–233(1994)

[3] P.J.Davis P.Rabinwitz(森 正武訳), 計算機による数値積分法, 日本コンピュータ協会, 1981

[4] Hairer E., Wanner G., Solving Ordinary Differential Equations II, Springer-Verlag, 1991

[5] 平山弘, C++言語によるべき型特異点をもつ関数の数値積分, 日本応用数理学会論文誌, vol. 5, pp. 257–266 (1995)

[6] 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, vol.12, pp. 1–8(2002)

[7] 宮広, 野田, 新しい有理関数近似によるハイブリッド積分の拡張について, 日本応用数理学会学会論文誌, Vol. 2, pp. 193–206 (1992)

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