Vol.2016-HPC-154 No.10
2016/4/25

Taylor 展開法による常微分方程式の解法における計算ステップ幅

平　山　　　弘[†1]

本論文では、次のような常微分方程式の初期値問題について考える。

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \qquad \mathbf{y}(x_0) = \mathbf{y}_0$$

この問題を Taylor 級数の係数を浮動小数点数で計算する数値的 Taylor 展開法を使うと高速に計算出来る。

得られた Taylor 展開式を Padé 展開してから計算を行うと A 安定な数値計算法になる。A 安定な公式は一般に、計算量が多いので、計算法を軽減するために、計算ステップ幅を大きくとる方法を考える。ある条件の下で、計算ステップ幅を Taylor 展開法の 2 倍程度にとると、計算時間と精度は Taylor 展開法とほぼ同じで、A 安定な計算法が得られる。

Step Size in Solutions of Ordinary Differential Equations by Taylor Series Method

Hiroshi Hirayama[†1]

In this paper we consider the following intial values problem of the ordinary differential equations.

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \qquad \mathbf{y}(x_0) = \mathbf{y}_0$$

When the numerical Taylor series method which computes the coefficient of Taylor series for this problem by the floating point number is used, it is shown that it is computable at high speed.

It'll be A-stable numerical calculation method after the obtained Taylor series are expaned into Padé expansion. The calculation is difficult because there is a lot of calculated amount for the A-stable formula. I get large calculation step size to reduce the calculated amount. When calculation step size is taken for about 2 times of the Taylor expansion method under some condition, the computing time and the precision are same as Taylor series method mostly, and it is A-stable calculation method.

1. はじめに

次のような常微分方程式の初期値問題の数値解法を考える。

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \tag{1}$$

このような常微分方程式の一段法の数値計算法として、現在 Runge-Kutta 法がよく使われている。その他に Taylor 級数法[2] が知られている。

Runge-Kutta 法による解法には、次のような特徴がある。この計算方法には、多くの公式が存在するが、計算次数毎に計算公式の係数を計算する必要がある。この計算は、かなり計算量の大きい計算となるため簡単には計算できない。このため、高次の公式の係数を計算するのは非常に困難である。このため、10 次以上の公式はあまり知られてないし、15 次以上の公式はほとんど知られていない。常微分方程式の研究書である Hairer[1] 等の書籍にも 10 次以上の公式は載っていない。たとえば、25 段 12 次の公式の係数を計算するには 7813 個の非線形方程式[6] を解かなければならない。

各次数の公式は独立性が高く、1 次だけ次数が高くなるだけでも、共通部分がほとんどない場合が多い。一部共通部分がある公式を埋め込み型と呼ばれている。これらの公式では、(1) の右辺の計算を追加計算することなしに、低次の計算ができる公式である。本来の次数の計算と低次の計算が出来るので、その差を利用して、計算誤差が推定できる公式としてよく使われる。このような型の公式として 6 段 5 次の Runge-Kutta-Fehlberg 法[4] がよく知られている。

これに対し Taylor 展開法は、1 個の公式（プログラム）から任意の次数の計算可能である。展開式は途中の次数まで共通であるから、すべての次数の公式が誤差評価可能でその計算の誤差推定も容易である。

Taylor 展開法は多くの長所持つが、Taylor 展開法を解説している単行本がないためか、その利用はあまり多くない。Taylor 展開法は簡単で使いやすいため、常微分方程式の精度保証にも使われている。Taylor 展開法では、Taylor 展開式の四則演算や関数計算を扱う。このためにはそのオペレーター・オーバーロードやオーバーロード機能を持つプログラミング言語である C++言語や Fortran 90 を使用しなければならない。数値計算では、今でも C 言語や FORTRAN 77 の機能を使うのが主流であり、C++言語や Fortran 90 を使って

†1 神奈川工科大学
Kanagawa Institute of Technology

も、これらの機能を利用していないことが多い。

Taylor 展開式の計算を行う場合、プログラミングの作り易さは、C 言語と C++言語では、高級言語とアセンブラ程度の差が生じる。このため、C 言語を使って、Taylor 展開式の計算を行うことは、実際上、不可能と思われる。

このような状況のためか、Taylor 級数を使った微分方程式を数値計算は一部[10] を除いてあまり使われているとはいえない。

通常の Runge-Kutta 法を使った場合、A 安定ではないため、stiff な微分方程式は効率的に解くことが出来ない。Taylor 展開法は、その展開式を Padé 展開することによって A 安定な計算法が得られる。Runge-Kutta 法では、陰的 Runge-Kutta 法を使えば A 安定な公式が得られるが、陰的 Runge-Kutta 法では、計算を 1 ステップ進める毎に一般に非線形の連立方程式を解く必要があり、格段に計算量が増える。陰的公式では、Fehlberg の公式のように簡単に誤差が推定できる公式がないため、精度を確認するには、非常に困難な作業となる。

Taylor 展開法では、高次の項の大きさを評価することによって、誤差を推定できるが、A 安定な計算法では、Padé 展開された式から誤差評価しなければならないが、このような推定法は知られていない。元の Taylor 展開式から誤差を推定することが一番簡単な方法であるが、この方法を使うと A 安定な計算法を使うメリットがなくなる。

本論文では、Taylor 展開式から得られるステップ幅何倍かし、それを Padé 展開式の計算に利用する方法を提案する。

2. 常微分方程式の解の Taylor 展開

高階の常微分方程式は一般性を失うことなしに 1 階微分方程式に書けるので、次の形を持つものと仮定する。

$$\frac{d\mathbf{y}}{dx} = \mathbf{f}(x, \mathbf{y}(x))$$

初期条件は、次のように与えられているものとする。

$$\mathbf{y}(x_0) = \mathbf{y}_0$$

ここで、$\mathbf{f}$ 、$\mathbf{y}$ は、一般にベクトル関数で、十分なめらかで必要な回数だけ微分可能とする。初期条件の $\mathbf{y}_0$ は定数ベクトルである。このような微分方程式は、次の Picard の逐次近似法[9] によって解くことができる。

$$\mathbf{y}_0(x) = \mathbf{y}_0$$
$$\mathbf{y}_{k+1}(x) = \mathbf{y}_0 + \int_{x_0}^{x} \mathbf{f}(t, \mathbf{y}_k(t))dt$$

Taylor 展開式を上の式の被積分関数に代入し、被積分関数を Taylor 展開する。Taylor 級数は、k 回目の反復計算の場合、k 次まで Taylor 展開する。その k 次の Taylor 展開を積分し、定数項 $\mathbf{y}_0$ を加えて $k+1$ 次の解を計算する。1 回の計算で、最低 1 次次数の高い解が得られる。

例として次の簡単な常微分方程式を解く。

$$\frac{dy}{dx} = 1 + \sqrt{y} \qquad y(0) = 1$$

初期条件から $y_0(x) = 1$ であるから、これを picard の逐次近似式に代入して

$$y_1(x) = 1 + \int_0^x (1 + \sqrt{1})dt = 1 + 2x$$

となる。被積分関数は 0 次の定数となり、最終計算結果は 1 次式になる。さらにこの結果を、Picard の逐次近似式に代入して計算する。被積分関数は展開し 1 次式まで取る。$1 + 2x$ となる。これを積分して、計算すると

$$y_2(x) = 1 + \int_0^x (1 + \sqrt{1 + 2t})dt = 1 + 2x + 0.5x^2 + O(x^3)$$

となる。このような計算を 2 回繰り返すと、次のように 4 次までの解が得られる。

$$y_3(x) = 1 + \int_0^x (1 + \sqrt{1 + 2t + 0.5t^2})dt = 1 + 2x + 0.5x^2 - 0.08333333x^3 + O(x^4)$$
$$y_4(x) = 1 + \int_0^x (1 + \sqrt{1 + 2t + 0.5t^2 - 0.0833333t^3})dt$$
$$= 1 + 2x + 0.5x^2 - 0.08333333x^3 + 0.0520833x^4 + O(x^5)$$

この計算を必要な回数行えば、任意の次数の Taylor 展開式が得られる。

この Taylor 展開式を利用して、次のステップにおける関数値を計算する。次のステップの幅を h とすると、$y(h)$ を計算する。この値を、次のステップの初期値として、これまでの方法と同様にしてさらに次のステップの関数値を求める。これを繰り返すことで微分方程式を解く。

ここでは、Picard の逐次近似法を使って計算したが、Taylor 展開式の係数を計算する方法としては、Picard の逐次近似法は、級数展開法と同じ計算になる。

2.1 Taylor 級数の平方根の計算

前の例題で Taylor 展開の平方根の Taylor 展開式を計算している。この計算は、次の微分

方程式を級数展開法で解くことによって計算できる。ここでは一般に α 乗することを考える。$\alpha = \frac{1}{2}$ の場合、平方根になる。$f(x)$ と $g(x)$ をそれぞれ次のような Taylor 展開とする。

$$f(x) = f_0 + f_1 x + \cdots + f_n x^n + \cdots \qquad g(x) = g_0 + g_1 x + \cdots + g_n x^n + \cdots \tag{2}$$

$f(x)$ の α 乗を $g(x)$ とすると次の関係式が得られる。

$$g(x) = f(x)^\alpha$$

微分することによって、次の式が得られる。

$$g'(x) = \alpha f(x)^{\alpha-1} f'(x)$$

上の式の両辺に $f(x)$ を掛け、$g(x) = f(x)^\alpha$ であることを使うと、次の式が得られる。

$$f(x)g'(x) = \alpha g(x) f'(x) \tag{3}$$

(2) の Taylor 展開式を (3) に代入すると、次のようになる。

$$\begin{aligned}&(f_0 + f_1 x + f_2 x^2 + \cdots)(g_1 + 2g_2 x + 3g_3 x^2 + \cdots)\\ &\quad = \alpha(g_0 + g_1 x + g_2 x^2 + \cdots)(f_1 + 2f_2 x + 3f_3 x^2 + \cdots)\end{aligned}$$

x の $n-1$ 次の項を比較すると、次の式が得られる。

$$\sum_{i=0}^{n} \{(n-i) - \alpha i\} f_i g_{n-i} = 0$$

上の式で、$i = 0$ の場合とそれ以外に分けると

$$n f_0 g_n + \sum_{i=1}^{n} \{(n-i) - \alpha i\} f_i g_{n-i} = 0$$

これから

$$g_n = \frac{1}{n f_0} \sum_{i=1}^{n} \{(\alpha+1)i - n\} f_i g_{n-i}$$

$g_0 = f_0^\alpha$ として、上の漸化式から係数を求める。Taylor 展開式の加減乗は簡単に計算でき、上と同様な方法で Taylor 展開式のべき乗、指数対数、三角関数などの計算が可能である。

3. A 安 定

次のようなテスト常微分方程式の初期値問題を考える。

$$y'(x) = -\lambda y(x) \qquad y(0) = 1 \tag{4}$$

この微分方程式を何らかの数値解法で、$x = x_n$ における y の値 y_n が計算されているとき、h だけ離れた $x_{n+1} = x_n + h$ に y の値を y_{n+1} の計算を行う。

$$y_{n+1} = r(z) y_n \qquad z = \lambda h \tag{5}$$

このとき、$h > 0$ をどのようにとっても、次の条件を満たす時、A 安定な計算法と呼ぶ。

$$|r(z)| \le 1 \tag{6}$$

(6) の条件を満たす、z を安定領域と言う。A 安定でない公式では、安定領域が広い公式が使いやすい良い公式と言われる。

常微分方程式の数値計算では、次の陽的 Runge-Kutta 法がよく使われている。この数値計算法は、A 安定でないことを説明する。

つぎのような微分方程式を考える。

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \tag{7}$$

s 段の陽的 Runge-Kutta 法は次のように書ける。

$$\left\{\begin{array}{lcl}\mathbf{k}_1 &=& \mathbf{f}(x_n, \mathbf{y}_n)\\ \mathbf{k}_2 &=& \mathbf{f}(x_n + c_2 h, \mathbf{y}_n + a_{21} h \mathbf{k}_1)\\ \mathbf{k}_3 &=& \mathbf{f}(x_n + c_3 h, \mathbf{y}_n + a_{31} h \mathbf{k}_1 + a_{32} h \mathbf{k}_2)\\ &\vdots&\\ \mathbf{k}_s &=& \mathbf{f}(x_n + c_s h, \mathbf{y}_n + a_{s1} h \mathbf{k}_1 + a_{s2} h \mathbf{k}_2 + \cdots + a_{s,s-1} h \mathbf{k}_{s-1})\\ \mathbf{y}_{n+1} &=& \mathbf{y}_n + \sum_{i=1}^{s} b_i \mathbf{k}_i\end{array}\right. \tag{8}$$

この式は、$x = x_n$ における $\mathbf{y}$ の値 $\mathbf{y}_n$ が与えられているとき、$x_{n+1} = x_n + h$ における $\mathbf{y}$ の値 $\mathbf{y}_{n+1}$ を計算する式である。

$a_{ij}(1 \le j < i \le s)$、$b_i(i = 1, \cdots, s)$、$c_i(i = 2, \cdots, s)$ は定数である。これらの定数は、この公式が h について、Taylor 展開したとき、p 次まで一致するように選ぶ。このような公式を s 段 p 次の Runge-Kutta 公式と呼ぶ。陽的 Runge-Kutta 法の作成方法から、

$$r(z) = r_0 + r_1 z + r_2 z^2 + \cdots + r_n z^n$$

と考えられる。このとき、十分大きな z を入れれば、$|r(z)| > 1$ となり A 安定とはならないことがわかる。すなわち

$$\lim_{z \to \infty} |r_0 + r_1 z + r_2 z^2 + \cdots + r_n z^n| = \infty$$

となり、A 安定でないことがわかる。

(5) の式からわかるように、$r(z) = e^{-z}$ である。e^{-z} の Taylor 展開式は、収束半径が無限大であるから、次数を十分高くすれば、安定領域を広くすることができる。これは、高次の公式を使えば、広い安定領域を持つことを意味する。

Taylor 展開式と一致するための独立な条件式の数は、定数 (a, b, c) の数より少ないため、これらの定数を一意に決定することはできない。このため、これらの定数は出来るだけ零に選び、零でない定数は、出来るだけ簡単な数値で表現できるものを選ぶ。さらに桁落ちが生じない、安定領域ができるだけ広くなるように選ぶ。

これら定数を決定する作業は、次数が高くなるにつれて非常に難しい計算になる。高次の

Runge-Kutta の公式を作るためには、大規模な非線形方程式を解かなければならない。このため、現在使える Runge-Kutta の公式は 12 次程度までで、15 次とか 20 次とか計算次数を自由に選ぶことができない。

このような欠点を解決するために、次のような陰的 Runge-Kutta 法 (IRK 法) がある。

$$\begin{cases} \mathbf{k}_i = \mathbf{f}(x_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j) \quad i = 1, \cdots, s \\ \mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i \end{cases} \tag{9}$$

$a_{ij} = 0 \quad (j \geq i)$ の場合は陽的 Runge-Kutta であり、それ以外の場合、陰的 Runge-kutta 法と呼ぶ。陰的 Runge-Kutta 法は、計算次数を自由に選ぶことができ、さらに通常の Runge-Kutta 法とは異なり A 安定であるという特徴を持つ。また s 段の陰的 Runge-Kutta の公式では、$2s$ 次の公式が存在する。この公式を使うには、(9) の最上段の $\mathbf{k}_i \quad (i = 1, \cdots, s)$ の連立方程式を解く必要がある。一般にこの方程式は非線形の連立方程式で各計算ステップ毎に解かなければならない。非線形連立方程式を解くことを厭わないならば、多くの問題をこの方法で解くことが可能である。

3.1 A 安定な Taylor 展開法よる解法

Taylor 展開法で解く場合、陽的 Runge-Kutta 法と同様に A 安定ではない。この場合、Taylor 展開式を Padé 展開すれば、A 安定である。

Padé 展開とは、以下のように Taylor 級数を、有理関数に変形したものである。

$$a_0 + a_1 x + \cdots = \frac{p_0 + p_1 x + \cdots + p_M x^M}{1 + q_1 x + \cdots + q_L x^L} \tag{10}$$

(10) 式の両辺に右辺の分母を掛け、$M + L$ 次の係数まで一致するように、有理関数の係数を決定することによって得られる。このとき、解かなければならない一次方程式の係数は、Toeplitz 行列と呼ばれる特別な行列になる。この場合の一次方程式の計算量は、未知数の数を n としたとき、通常 n^3 に比例するのに対し、n^2 に比例する手間で計算できる高速な計算法が知られている。ここでの計算は、n^2 に比例する Levinson-Durbin の方法[7] によって行った。

常微分方程式の計算ステップ幅を大きくしても、(10) の右辺からわかるように、無限大に発散するすることはない。$L = M$、$L = M + 1$、$L = M + 2$ のとき、A 安定であることが知られている。L が M より小さい場合、(10) の式は x が大きくなるとゼロに近づく。このとき L 安定と呼ばれている。

4. 高次公式による常微分方程式の解法

ここでは、計算公式の次数を変えることによって、計算効率がどのようになるかを述べる。扱う問題は HIRES と呼ばれる常微分方程式である。テスト問題としてよく引用される方程式で、多くの書籍[1] や Web 上で扱われている。生理学（physiology）の形態形成（形態発生, morphogenesis）における光との関係を表す方程式である。

以下のようなに未知関数が 8 個の問題である。

$$\begin{cases} y_1' = -1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007 \\ y_2' = 1.71y_1 - 8.75y_2 \\ y_3' = -10.03y_1 + 0.43y_4 + 0.035y_5 \\ y_4' = 8.32y_2 + 1.71y_3 - 1.12y_4 \\ y_5' = -1.745y_5 + 0.43y_6 + 0.43y_7 \\ y_6' = -280y_6y_8 + 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7 \\ y_7' = 280y_6y_8 - 1.81y_7 \\ y_8' = -280y_6y_8 + 1.81y_7 \end{cases} \tag{11}$$

初期条件は $y_1(0) = 1, y_2(0) = y_3(0) = y_4(0) = y_5(0) = y_6(0) = y_7(0) = 0, y_8(0) = 0.0057$

積分区間は $0 \leq t \leq 321.8122$ である。

4.1 計算プログラム

プログラムは非常に簡単で、次のように簡単に書ける。Taylor 展開式の係数を配列で表わすとする。すなわち、y_m の n 次の係数を配列の要素 y[m][n] で表わす。ここでは、y_4 を計算する式のみについて書く。他の関数についても同様にできる。まず 0 次の定数項を初期値を利用して決定する。

```
y[4][0] = 0 ;
```

Taylor 展開の $i+1$ 次の係数を決定するには、常微分方程式に Taylor 展開式を代入し、t^i の係数を比較する。左辺は (i+1)y[4][i+1] となるので、次の式が得られる。

```
y[4][i+1] = (8.32 * y[2][i] + 1.71 * y[3][i]-1.12 * y[4][i])/(i+1) ;
```

この式を反復利用することによって、任意次数の Taylor 展開式の係数が得られる。

y_6 以降の式には、非線型項 y_6y_8 がある。この項は、次のようにして計算できる。以下のプログラムではその計算された非線型項を利用して、y_6 の計算も行っている。

```
y6y8[i] = 0 ;
for( int j=0 ; j<= i ; j++ ) y6y8[i] += y[5][j] * y[7][i-j] ;
```

```
    y[6][i+1] = (-280y6y8[i]+0.69*y[4][i]
              + 1.71*y[5][i]-0.43*y[6][i]+0.69*y[7][i])/(i+1) ;
```

当然ながら、ここで計算された非線型項は、y_7、y_8 の計算にも利用できる。この非線型項の計算は、一種の自動微分法[8)5)] でもある。

実際の計算には、Taylor 展開のライブラリー[3)] を使用した。この例題では並列化は行わなかったが、上のような書き方をすると、右辺と左辺に同じ名前の変数が出るためか OpenMP による並列化ができないコンパイラーがあった。一旦別の変数に入れ、改めて代入すると問題なく並列化ができる。

4.2 計算結果

この方程式を次数 3 から 20 次までおよび 25,30,35 次の Taylor 展開式を使って解いた。使用した計算機は Intel i7-4770 3.5GHz, コンパイラーとして、Windows OS 上で、Visual Studio 2013 C++ (オプション /EHsc /Ox) を使用した。その結果を表 1 に示す。

計算途中で得られた Taylor 展開式を次の形式になったとする。

$$y(t) = y_0 + y_1(t-t_0) + y_2(t-t_0)^2 + \cdots + y_n(t-t_0)^n \tag{12}$$

この Taylor 展開式から、絶対的誤差は $y_n(t-t_0)^n$ 程度と推定できる。刻み幅 h とすると絶対誤差 ϵ_{abs} を満たす刻み幅 h は、次のように書くことができる。

$$|y_n h^n| \le \epsilon_{abs} \tag{13}$$

相対誤差は、$y_0 \ne 0$ で、y_0 が $y_n h^n$ と比較して十分に大きいと仮定すると、刻み幅 h は、次の式を満たすことがわかる。

$$|\frac{y_n}{y_0} h^n| \le \epsilon_{rel} \tag{14}$$

これらの条件式から刻み幅 h を、各式毎に計算する。得られた計算結果の最小値をそのステップ幅とした。この刻み幅を利用する適応型数値解法を使って計算を行った。今回の計算では、$\epsilon_{abs} = \epsilon_{rel} = 10^{-14}$ として計算した。

計算結果は、3 次の公式を使ったとき丸め誤差のためか、12 桁程度、4 次の公式では 13 桁程度、それ以上ではほぼ要求精度の 14 桁以上の精度が得られた。

計算次数が低いとき、計算の次数が 1 次上がる毎に急速に計算時間が減少していることがわかる。このことは、3 次の計算のとき、刻み幅の最大値と最小値がそれぞれ 6.04×10^{-3}、4.28×10^{-12} であるのに対し、12 次の計算のとき、6.24×10^{-1}、9.74×10^{-3} であることからもわかる。次数が低い場合、要求精度を満たすには、刻み幅が非常に小さくする必要があることがわかる。この問題では、15 次程度以上の次数からは計算時間がほとんど変わらないことがわかる。要求精度が厳しくなければ、低次な公式でも刻み幅を大きくとれるので、比較的高速に計算出来る。高精度の計算を行うためには、高次の計算法は必要不可欠なものになる。

表 1　Numerical results of HIRES

order	comp. time(msec)	No. of steps	max. step size	min. step size
3	740.00	1244404	6.04e-3	4.28e-12
4	38.09	61385	1.10e-1	3.06e-8
5	10.88	16255	2.19e-1	2.85e-6
6	7.90	10980	2.17e-1	4.75e-5
7	6.87	9179	4.15e-1	3.03e-4
8	6.21	8200	4.85e-1	8.96e-4
9	5.97	7445	5.00e-1	1.81e-3
10	5.77	6869	5.33e-1	3.48e-3
11	5.70	6371	5.73e-1	6.03e-3
12	5.50	5951	6.24e-1	9.74e-3
13	5.37	5583	6.31e-1	1.46e-2
14	5.30	5261	7.12e-1	1.97e-2
15	5.18	4974	7.28e-1	2.33e-2
16	5.01	4718	7.65e-1	2.85e-2
17	5.02	4487	8.14e-1	3.46e-2
18	4.99	4277	8.78e-1	3.79e-2
19	4.96	4088	8.73e-1	3.97e-2
20	4.89	3914	9.00e-1	4.15e-2
25	4.87	3228	1.11e-1	5.03e-2
30	4.89	2749	1.26e-1	5.91e-2
35	4.94	2395	1.22e-1	6.79e-2

5. Padé 展開式を利用した A 安定な数値解法

Padé 展開を使って常微分方程式を計算するには、次の手順で計算する。

(1) 常微分方程式の解の Taylor 展開式を計算する。

(2) Taylor 展開式を Padé 展開する。このとき連立一次方程式を解く必要がある。Padé 展開式の分母の次数を M とすると、M 元連立一次方程式を解かなければならない。
この連立一次方程式は、特別な形をした方程式なので、M^2 に比例する Levinson-Durbin の方法[7)] によって行うことができる。

(3) Taylor 展開式の最高次項が要求精度以下になるように、ステップ幅を選ぶ。

(4) 得られた、Padé 展開式に次のステップの位置を代入して次のステップの関数値を計算する。次のステップ幅 h は、Taylor 展開法と同様な方法を使う。

このような方法を使うと、Padé 展開するために時間が使われるため、微分方程式の関数が単純で計算時間がかからない場合、相対的に Padé 展開するための時間が大きくなる。

例題に示した方程式も単純で計算が簡単なため、要求精度 $\epsilon = 10^{-14}$ の倍精度計算で、Padé 展開を使うと 4〜5 倍の計算時間がかかった。この計算条件の場合、Taylor 展開法でも倍精度数の限界までの精度が得られるので、計算時間がかかるだけ Padé 展開法が不利である。

要求精度 $\epsilon = 10^{-6}$ の倍精度計算で、Padé 展開を使うと Taylor 展開する回数が半減し、全体で 2〜3 割増の計算時間となった。この計算条件の場合、Taylor 展開法と Padé 展開法の計算精度は、20 次以下の低次数では Taylor 展開法が良いが高次になると Padé 展開法がやや良いが、ほぼ同程度といえる。

Padé 展開法では、大きなステップ幅をとっても、安定なので、Taylor 展開法から得られるステップ幅を 2 倍にして計算した。要求精度 $\epsilon = 10^{-6}$ の倍精度計算の計算に、この方法を適用した。計算速度は Taylor 展開法と比較して 1 割程度速くなったが計算精度は、1 桁程度低下した。

6. 終わりに

Taylor 展開を使えば、高次の常微分方程式が簡単に得られる。この Taylor 展開を Padé 展開すれば、A 安定な計算法が得られる。A 安定な計算は一般に計算量が多いため、これまであまり使われることはなかった。

これを改善するために、ステップ幅を大きく取る方法を提案した。計算要求精度が計算精度の半分程度の場合、Taylor 展開法で得られたステップ幅を 2 倍にする方法である。このよう取ると計算時間がそれまでの計算時間の 4 分の 1 程度になり、計算時間、計算精度も Taylor 展開法と同程度になる。

参考文献

1) E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Springer-Verlag, (1993)
2) 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), 1–8
3) 平山, 舘野, 浅野, 川口, Taylor 級数演算ライブラリの使用法, 東北大学情報シナジーセンター大規模科学計算システム広報 SENAC, 40(2007) 29–68
4) Gisela Engeln-Müllges, Frank Uhlig, Numerical Algorithms with Fortran, Springer, 1996
5) 久保田光一, 伊理正夫, アルゴリズムの自動微分と応用、コロナ社、(1998)
6) 大野博, 25 段 12 次陽的ルンゲ·クッタ法構成の試み, 日本応用数理学会論文誌, 16(2006), 177–186
7) Press W.H., Flannery B.P., Teukolsky S.A., Vetterling W.T., Numerical Recipes in C, Cambridge University Press, Cambridge(1988)
8) Rall,L. B. , Automatic Differentiation-Technique and Applications, Lecture Notes in Computer Science, Vol.120, Springer-Verlag, Berlin-Heidelberg-New York(1981)
9) 佐野理, キーポイント微分方程式, 岩波書店, 東京, (1993)
10) Shiraishi, F., Tomita, T., Iwata, M., Berrada, A. Aziz., Hirayama, H., A reliable Taylor series-based computational method for the calculation of dynamic sensitivities in large-scale metabolic reaction systems: Algorithm and software evaluation, Mathmatical Bioscience, **222** (2008), 73–85.