Vol.2013-ARC-207 No.3
Vol.2013-HPC-142 No.3
2013/12/16

Taylor 展開法による常微分方程式の高次並列計算

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ことを示した。

計算次数は任意に選べるので、問題の計算精度に合った次数で計算が可能である。次数が高くなると安定性が良くなるので、高次の公式を使えば、ある程度 stiff な問題も効率的に計算出来る。計算結果は Taylor 展開式の形で得られるので、計算の誤差も容易に評価できる。

高次数高精度の問題を Taylor 展開法と陰的 Runge-Kutta 法による計算結果の比較を行い、Taylor 展開法が効率的な計算法であることを示した。Taylor 展開法は、計算は単純なので、並列化も容易である。

Higher Order Parallel Computation of Ordinary Differential Equations by Taylor Series

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.

Since the degree of the computation can be chosen arbitrarily, it is calculable by the degree suitable for the problem. If we use the higher order numerical formula for the ordinary differential equations, some stiff problem can be solved efficiently. Since we can obtaine the Taylor series as computation results, the error of computation can also be evaluated easily.

The results were computed ill-conditioned problems in hih order and high precision by the Taylor series method was compared, and it is shown that Taylor series method is an efficient algorithm.

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数値計算法として、次の陽的 Runge-Kutta 法がよく使われている。

$$\begin{cases} \mathbf{k}_1 &= \mathbf{f}(x_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= \mathbf{f}(x_n + c_2 h, \mathbf{y}_n + a_{21} h \mathbf{k}_1) \\ \mathbf{k}_3 &= \mathbf{f}(x_n + c_2 h, \mathbf{y}_n + a_{31} h \mathbf{k}_1 + a_{32} h \mathbf{k}_2) \\ &\vdots \\ \mathbf{k}_s &= \mathbf{f}(x_n + c_s h, \mathbf{y}_n + a_{s1} h \mathbf{k}_1 + a_{s2} h \mathbf{k}_2 + \cdots + a_{s,s-1} h \mathbf{k}_{s-1}) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \sum_{i=1}^{s} b_i \mathbf{k}_i \end{cases} \tag{2}$$

この式では、$x = x_n$ 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。

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、出来るだけ簡単な分数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。たとえば、25 段 12 次の公式を作成する場合、条件式が 7813 個の非線形方程式[7] を解き、計算式の係数を決定しなければならない。

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。

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

Vol.2013-ARC-207 No.3
Vol.2013-HPC-142 No.3
2013/12/16

微分方程式の数値解法でよく使われている次のような 4 段 4 次の公式がある。

$$\begin{cases} \mathbf{k}_1 &= \mathbf{f}(x_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= \mathbf{f}(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1) \\ \mathbf{k}_3 &= \mathbf{f}(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2) \\ \mathbf{k}_4 &= \mathbf{f}(x_n + h, \mathbf{y}_n + h\mathbf{k}_3) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{5}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \end{cases} \tag{3}$$

この公式では、関数 $\mathbf{f}(x, \mathbf{y})$ を 4 回計算して、4 次の精度が得られる。上に挙げた 25 段の公式では、関数計算を 25 回行っても、計算精度は 12 次である。4 次の公式までは、関数計算の回数と同じ次数の公式が得られるが、5 次以上の Runge-Kutta 公式では、関数計算回数と同じ次数が存在しないことが知られている。

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。

$$\begin{cases} \mathbf{k}_i &= \mathbf{f}(x_n + c_i h, \mathbf{y}_n + h\sum_{j=1}^{s} a_{ij}\mathbf{k}_j) \quad i = 1, \cdots, s \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\sum_{i=1}^{s} b_i \mathbf{k}_i \end{cases} \tag{4}$$

$a_{ij} = 0 \quad (j \geq i)$ 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、(4) 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。

また、Taylor 展開を使った解法[3] が知られている。この方法は、手計算で常微分方程式を級数展開する方法と同様である。教科書では、Taylor 展開の係数が簡単な規則性を持つ場合だけを扱っているが、規則性を持たない場合でも計算可能である。この計算方法では何次までででも計算出来るから、任意の次数の計算が可能である。

本論文では、IRK で計算された問題[5] を Taylor 展開法で解きその性能およびその特徴の比較を行った。

Taylor 展開法で使用した計算機環境は Intel i7-3930K, 3.2GHz(6core)、Windows 8(64 ビット)、MS Visual C++ 2012, 多倍長計算プログラムとして自作の 10^8 進数多倍長プログラムを使用した。IRK 法の計算機環境は Intel i7-3820, 3.6GHz(4core), Scientific Linux 6.3(64 ビット), Intel C++ 13.0.1, 多倍長プログラム MPFR 3.1.1/GMP 5.1.1, BNC-pack 0.8 である。

2. 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[2] 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。

$$\begin{cases} y_1' &= -1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007 \\ y_2' &= 1.71y_1 - 8.75y_2 \\ y_3' &= -10.03y_1 + 0.43y_4 + 0.035y_5 \\ y_4' &= 8.32y_2 + 1.71y_3 - 1.12y_4 \\ y_5' &= -1.745y_5 + 0.43y_6 + 0.43y_7 \\ y_6' &= -280y_6y_8 + 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7 \\ y_7' &= 280y_6y_8 - 1.81y_7 \\ y_8' &= -280y_6y_8 + 1.81y_7 \end{cases} \tag{5}$$

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。

2.1 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。

Vol.2013-ARC-207 No.3
Vol.2013-HPC-142 No.3
2013/12/16

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)6)] でもある。

実際の計算には、Taylor 展開のライブラリー[4)] 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。

2.2 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。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。

$$y(t) = y_0 + y_1(t-t_0) + y_2(t-t_0)^2 + \cdots + y_n(t-t_0)^n \tag{6}$$

この式を利用すると、絶対的誤差は $y_n(t-t_0)^n$ 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。

$$|y_n h^n| \leq \epsilon_{abs} \tag{7}$$

相対誤差は、$y_0 \neq 0$ のとき、y_0 は $y_n h^n$ と比較して非常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。

$$|\frac{y_n}{y_0} h^n| \leq \epsilon_{rel} \tag{8}$$

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の刻み幅とする。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	874.00	1244404	6.04e-3	4.28e-12
4	45.41	61444	3.73e-2	3.06e-8
5	13.06	16254	1.63e-1	2.85e-6
6	9.40	10980	2.17e-1	4.75e-5
7	8.42	9179	4.15e-1	3.03e-4
8	7.69	8200	4.85e-1	8.96e-4
9	7.51	7445	5.00e-1	1.81e-3
10	7.32	6870	5.33e-1	3.48e-3
11	7.08	6371	5.73e-1	6.0ee-3
12	6.90	5951	6.24e-1	9.74e-3
13	6.89	5583	6.31e-1	1.46e-2
14	6.78	5261	7.12e-1	1.97e-2
15	6.77	4974	7.28e-1	2.33e-2
16	6.59	4718	7.65e-1	2.85e-2
17	6.59	4487	8.14e-1	3.46e-2
18	6.59	4277	8.78e-1	3.79e-2
19	6.65	4088	8.73e-1	3.97e-2
20	6.59	3914	9.00e-1	4.15e-2
25	6.59	3228	1.11e-1	5.03e-2
30	6.59	2749	1.26e-1	5.91e-2
35	6.71	2395	1.22e-1	6.79e-2

3. 陰的 Runge-Kutta 法との比較

ここでは陰的 Runge-Kutta 法との比較を行う。陰的 Runge-Kutta 法は、任意次数の計算が可能であることが以前から知られていたが、計算途中で非線型方程式を解く必要があるため、実際に高次の計算が行われることは、ほとんどなかった。

最近このような計算[5)] が行われので、その結果と Taylor 展開法との比較を行った。

Vol.2013-ARC-207 No.3
Vol.2013-HPC-142 No.3
2013/12/16

3.1 Lorenz モデル

まず簡単な問題として、Lorenz モデルを扱う。このモデルは次の常微分方程式で表されるようにかなり単純な 3 元連立の微分方程式である。

$$\begin{cases} \dfrac{dy_1}{dx} &= \sigma(-y_1 + y_2) \\ \dfrac{dy_2}{dx} &= -y_1 y_3 + r y_1 - y_2 \\ \dfrac{dy_3}{dx} &= y_1 y_2 - b y_3 \end{cases} \tag{9}$$

初期条件及び定数は、以下の様にする。

$$y_1(0) = 0,\ y_2(0) = 1,\ y_3(0) = 0$$
$$\sigma = 10,\ r = \frac{470}{19},\ b = \frac{8}{3}$$

この問題を区間 $[0, 50]$ で計算精度 200 桁の多倍長浮動小数点数で計算する。

この問題はカオス的ふるまいを示す非線型方程式の一つとして知られている。倍精度計算では、桁落ちのため、精度良い結果が得られない問題としても知られている。ここでは、十分な精度の 200 桁で計算し桁落ちの影響が少なくして計算する。

Taylor 展開法[3] では、出来るだけプログラム作成を簡単にするため、計算に無駄があるが、ここでは、それらの無駄をなくしできるだけ効率的なプログラムを作成した。

方程式 (9) 最上段の式は、Taylor 級数の係数の計算と見ると、Taylor 級数 y_2 の i 次の係数 $y_2[i]$ から Taylor 級数 y_1 の i 次の係数 $y_1[i]$ を引いて、σ 倍し、それを $(i+1)$ で割ることによって積分を行い $i+1$ 次の係数 $y_1[i+1]$ を求めることを意味する。方程式 (9) の 2 段目の式は非線型項 $y_1 y_3$ を含んでいる。

この項の i 次の係数は、$\sum_{j=0}^{i} y_1[j] y_3[i-j]$ として計算できるから、プログラムは以下のようになる。上の方程式の計算のカーネルの部分は以下のような 7 行と非常に短いプログラムとなる。

```
1: y1[i+1]=10*(y2[i]-y1[i])/(i+1) ;
2: y1y3=0 ;
3: for(j=0;j<=i;j++) y1y3+=y1[j]*y3[i-j];
4: y2[i+1]=(470*y1[i]/19-y2[i]-y1y3)/(i+1) ;
5: y1y2=0 ;
6: for(j=0;j<=i;j++) y1y2+=y1[j]*y2[i-j];
7: y3[i+1]=(y1y2-8*y3[i]/3)/(i+1) ;
```

定数項 (0 次の項) は初期条件で与え、1 次以上の係数はこのカーネル部分を使って計算する。この計算の反復計算の回数によって、その Taylor 展開の次数が決まる。たとえば、10 回反復計算をすれば、10 次の Taylor 展開式が得られる。

このプログラムを利用して、要求精度が 10^{-120} と 10^{-170} の場合、計算精度が 160 次、100 次、120 次の場合について計算した。その結果を表 2 と表 3 に示す。幸谷[5] による IRK 法の結果も一緒に示す。

IRK 法では、要求精度が 10^{-120} の場合、計算次数が 80 段 160 次の場合が最も計算時間が少なく、要求精度が 10^{-170} の場合、計算次数が 120 段 240 次の場合が最も計算時間が少ない。Taylor 展開法では、要求精度が 10^{-120}、10^{-170} の両方の場合、計算次数が 160 次の場合が最も計算時間が少ない。

計算機、コンパイラ、多倍長計算プログラムが異なるため、比較は困難であるが、単順に比較すると Taylor 展開法が IRK 法と比べると約 40 倍程度高速になっていることがわかる。

計算のカーネルの 3 行目は、積和の形になっているので、OpenMP の積和の並列化の指示句を指定しても、並列化ができない。計算が多倍長の計算の場合、このような積和の部分でも並列化は出来ないようである。言語が持っている数値でないと並列化しないようである。

並列化のため、二つの多倍長の数値の積を、一時的に変数に入れ、その変数の和を計算する形にし、並列化を行った。

3.2 1次元 Brusselator 問題

少し大きな問題として、1 次元 Brusselator 問題を扱う。この常微分方程式は、次の偏微分方程式

$$\begin{cases} \dfrac{\partial u}{\partial t} &= 1 + u^2 v - 4 + 0.02 \dfrac{\partial^2 u}{\partial x^2} \\ \dfrac{\partial v}{\partial t} &= 3u - u^2 v + 0.02 \dfrac{\partial^2 v}{\partial x^2} \end{cases} \tag{10}$$

を空間方向に $N+1$ 等分 $(N = 500)$ し、常微分方程式化したものである。境界条件は $u(x=0,t) = 0, u(x=1,t) = 0, v(x=0,t) = 3, v(x=0,t) = 3$ である。初期条件は $u(x, t=0) = 1 + \sin(2\pi x), v(x, t=0) = 3$ である。

Vol.2013-ARC-207 No.3
Vol.2013-HPC-142 No.3
2013/12/16

表 2 Lorenz モデルの 200 桁精度の計算結果 1

要求精度	Taylor 展開法 (10^{-120})		
次数	160	200	240
計算時間 (秒)	42.5	46.1	81.7
4 Threads	16.8	17.9	20.4
speedup ratio	2.5	2.6	4.0
ステップ数	1005	706	557
誤差	1.0e-110	2.7e-110	2.3e-110
要求精度	陰的 Runge-Kutta 法 (10^{-120})		
段数	80	100	120
計算時間 (秒)	1991.4	2317.4	2555.0
4 Threads	659.3	762.1	2215.4
speedup ratio	3.0	3.0	3.0
ステップ数	1661	863	563
誤差	6.5e-110	1.3e-109	2.3e-109

表 3 Lorenz モデルの 200 桁精度の計算結果 2

要求精度	Taylor 展開法 (10^{-170})		
次数	160	200	240
計算時間 (秒)	87.6	82.0	132.3
4 Threads	34.4	32.2	32.8
speedup ratio	2.6	2.7	4.0
ステップ数	2066	1257	902
誤差	5.0e-161	1.2e-160	5.5e-160
要求精度	陰的 Runge-Kutta 法 (10^{-170})		
段数	80	100	120
計算時間 (秒)	8233.6	5761.3	5285.9
4 Threads	2858.1	2089.4	1826.0
speedup ratio	3.0	2.8	2.9
ステップ数	6879	2603	1370
誤差	1.0e-161	9.0e-160	7.2e-160

$$\begin{cases} \dfrac{du_i}{dt} &= 1 + u_i^2 v_i - 4 + 0.02\dfrac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \\ \dfrac{dv_i}{dt} &= 3u_i - u_i^2 v_i - 4 + 0.02\dfrac{v_{i+1} - 2v_i + v_{i-1}}{(\Delta x)^2} \\ & (i = 0, 1, \cdots, N+1) \end{cases} \tag{11}$$

表 4 1 次元 Brusselator 問題の 70 桁精度の計算結果 1

要求精度	Taylor 展開法 (10^{-50})		
次数	40	60	80
計算時間 (秒)	6538.5	9159.9	11875.6
6 Threads	1144.6	1475.7	1889.6
speedup ratio	5.7	6.2	6.3
ステップ数	12313	8449	6436
誤差	3.9e-53	1.1e-55	1.1e-55

要求精度	陰的 Runge-Kutta 法 (10^{-50})		
段数	20	30	40
計算時間 (秒)	8866.0	9271.5	10171.5
4 Threads	4195.4	3978.8	4159.4
speedup ratio	2.1	2.3	2.4
ステップ数	1629	860	553
誤差	2.9e-41	2.1e-38	9.0e-35

表 5 1 次元 Brusselator 問題の 70 桁精度の計算結果 2

要求精度	Taylor 展開法 (10^{-60})		
次数	40	60	80
計算時間 (秒)	6518.2	9131.3	11797.6
6 Threads	1144.6	1552.5	1894.9
speedup ratio	5.7	5.9	6.2
ステップ数	12313	8454	6436
誤差	2.9e-63	8.6e-65	1.2e-65

要求精度	陰的 Runge-Kutta 法 (10^{-60})		
段数	20	30	40
計算時間 (秒)	19712.0	11377.8	13667.6
4 Threads	8966.9	4784.8	5000,6
speedup ratio	2.2	3.3	2.7
ステップ数	3249	890	630
誤差	4.8e-53	1.1e-43	1.7e-44

これを、積分区間 $[0, 10]$ で計算する。計算は、10^8 進数 10 桁（10 進数で約 80 桁）の多倍長数を利用して計算した。使用した多倍長精度プログラムでは、10 進数で 8 桁単位で指定しなければならない仕様になっている。幸谷では 70 桁の精度で計算しているので、72 桁で計算するのが一番近い精度であるが 8 桁の丸め処理があるので、確実に 70 桁の精度を確保するために 80 桁の精度で計算した。

IRK 法では、計算次数が段数の 2 倍になるので、それに合わせて、次数が 40 次、60 次、80 次の Taylor 展開を利用して計算した。要求精度が 10^{-50} と 10^{-60} の 2 つの場合を計算した。この計算結果を、表 4、表 5 に示す。

この方程式の係数は、すべて桁数の小さい整数か桁数の小さい小数なので、計算に利用した多倍長数は、10^8 進数なので、有効桁数の小さい多倍長数で表すことができるため、計算を速くすることができた。

方程式には、2 乗の計算を含んでいる。この 2 乗部分には、乗算の約半分の時間で計算できる計算法を使った。

この問題では、Taylor 展開法と IRK 法の計算時間は、あまり差がないが IRK 法による計算結果は、Taylor 展開法の計算結果と比べ、10 桁程度計算精度が悪くなっていることが

Vol.2013-ARC-207 No.3
Vol.2013-HPC-142 No.3
2013/12/16

表 6 刻み幅を制限した場合の 1 次元 Brusselator 問題の 70 桁精度の計算結果

要求精度 (10^{-60})	max $h_k \leq 0.005$		max $h_k \leq 0.002$	
段数	30	40	30	40
計算時間 (秒)	21834.0	32983.4	43723.7	74230.3
4 Threads	9286.8	12421.8	18641.4	28060.2
speedup ratio	2.4	2.7	2.3	2.6
ステップ数	1864	1748	3856	4156
誤差	1.1e-49	7.4e-49	3.4e-53	2.9e-53

わかる。幸谷は計算精度は刻み幅を制限することによって、この点の改善をはかっている。表 6 には、要求精度 (10^{-60}) の計算を刻み幅を 0.005 以下、0.002 以下制限することによって、計算精度を改善できていることがわかる。この改善された結果と Taylor 展開法の結果と比較すると、計算精度は少し Taylor 展開法が良い結果を与えるがほぼ同じ程度の結果になっている。また、この刻み幅の制限により、計算時間がかなり増加することがわかる。刻み幅を 0.002 以下に制限すると 4.7〜7.3 倍程度の時間を必要とすることがわかる。Taylor 展開を利用した場合、要求精度が 10^{-50}、計算次数が 40 次の場合、刻み幅は計算開始時に 3.97×10^{-5} と最小になり、時間が 2.31×10^{-2} のとき、9.82×10^{-4} と最大になり、多くは 8.13×10^{-4} 程度の幅であった。要求精度が同じならば、次数が大きいほど、刻み幅は大きくとることができる。

Taylor 展開法の方が IRK 法に比べ、ステップ数が 10 倍前後多くなっている。Taylor 展開法は、陽的計算法なので、刻み幅を非常に小さくする必要があることがわかる。

IRK 法では、要求精度を指定しても、得られた結果がその精度近くの結果にならない場合が多いようである。その点、Taylor 展開法は、要求精度を指定すれば指定精度近くの計算結果が得られる。

Taylor 展開式をパデ展開すれば、A 安定な計算法に変換できるが、今回の計算では、使用しなかった。パデ展開を使うと経験的には、精度の良い結果になるが、どの程度の精度であるか推定するのが困難であるためである。

Taylor 展開法は単純であるためか、並列化が容易であることがわかる。計算次数が大きいほと並列化が効果的であることがわかる。

表 4 と表 5 に示した Taylor 展開法の誤差は、計算精度、計算次数を上げて計算した結果が正しい結果として誤差を計算したものである。

4. 終わりに

計算環境が異なるために、優劣を論ずることは無理があるが、コンパイラー、多倍長演算プログラムの性能がほぼ同じとあると仮定すると、性質の良い問題では、Taylor 展開法は IRK 法より約 40 倍程度高速に、Stiff な問題でも数倍程度高速に計算できた。

Stiff な問題を含め、多くの常微分方程式が Taylor 展開法を使い、高次の計算法を使えば、高速で高精度な計算が期待できる。

Taylor 展開法は計算手順が簡単なので、並列化が容易である。特に次数が高い計算ほど並列化が効果的である。時間のかかる高次の計算は並列化によって高速化がはかれると思われる。

参 考 文 献

1) Abramowitz, M. & Stegun, I.A. Handbook of mathematical functions. Washington, DC: National Bureau of Standards. (1964)
2) E. Hairer, G. Wanner, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Springer-Verlag, (1993)
3) 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
4) 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
5) 幸谷 智紀, 列化した多倍長陰的 Runge-Kutta 法の性能分析、情報処理学会研究報告, Vol. 2013-HPC-139(2013), No. 18, 1–8
6) 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)
7) 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
8) Rall,L. B. , Automatic Differentiation-Technique and Applications, Lecture Notes in Computer Science, Vol.120, Springer-Verlag, Berlin-Heidelberg-New York(1981)