並列型合成法による常微分方程式の高次数値積分

石森　勇次
(工学部教養教育)

常微分方程式に対する高次の数値積分法として，２次の対称なスキームの直列合成と並列合成によるスキームを考え，具体的な数値計算を通して数値としての計算精度や計算コストを調べる。

キーワード：並列型合成法，常微分方程式，数値積分法，高次の精度，計算コスト

１．はじめに

常微分方程式に対する高次の数値積分法として，対称な２次のスキームを直列に連結する方法（直列型合成法：図１）がある [1-3]。一方，最近筆者により対称な２次のスキームを並列に連結する方法（並列型合成法：図２）も提案された [4]。

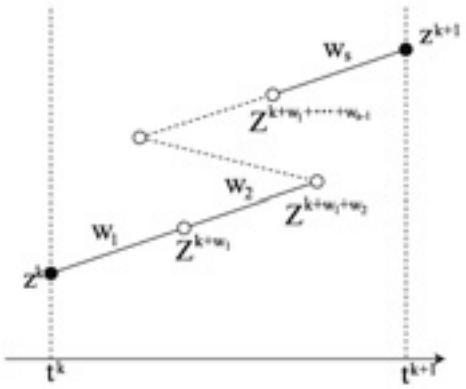

図１　直列型合成法

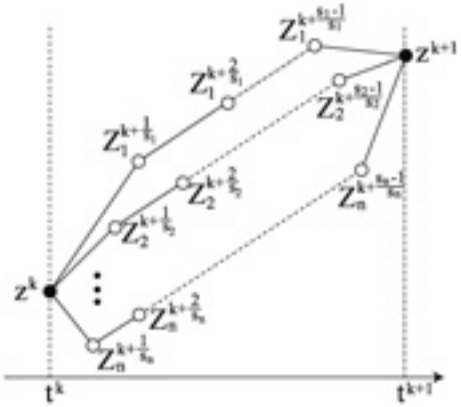

図２　並列型合成法

これらの合成法はいわゆる幾何学的数値積分法 [1-4] の研究の中で提案されてきたものであるが，一般の常微分方程式を単に数値的に解くために使用することも可能である。本論文では，幾何学的数値積分という視点ではなく，単に数値積分法として合成法がどのような特徴を持つのかについて，特に数値としての精度や計算コストについて議論する。

２．合成法

2.1 微分方程式の積分表示

独立変数 t の N 個の未知関数

$$z = (z_1(t), z_2(t), \ldots, z_N(t)) \tag{1}$$

に対する一般的な形の連立微分方程式

$$\frac{dz}{dt} = f(z,t) \tag{2}$$

を考える。ここで，

$$f(z,t) = (f_1(z,t), f_2(z,t), \ldots, f_N(z,t)) \tag{3}$$

である。刻み幅を Δt とし，離散時間

$$t^k = k\Delta t \ (k = 0, 1, 2, \ldots) \tag{4}$$

での $z(t^k)$ の数値解を z^k と表す。微分方程式 (2) を区間 $[t^k,\ t^{k+1}]$ で積分すると，積分表示

$$z(t^{k+1}) = z(t^k) + \int_{t^k}^{t^{k+1}} f(z(t),t)dt \tag{5}$$

を得る。積分表示 (5) の右辺の積分をどのように近似するのかによって，さまざまな数値積分法を構成することができる。

本論文で議論する合成法では，積分

$$\int_{t^{k+a}}^{t^{k+b}} f(z(t),t)dt$$

の２次近似を

$$I^{b,a} = (b-a)\Delta t f^{b,a}(z^k, t^k) \tag{6}$$

と表す。ここで，$f^{b,a}(z^k,t^k)$ は対称性

$$f^{b,a}(z^k,t^k) = f^{a,b}(z^k,t^k) \tag{7}$$

を持つものとする。具体的には，台形公式

$$f^{b,a}(z^k,t^k) = \frac{f(z^{k+a},t^{k+a}) + f(z^{k+b},t^{k+b})}{2} \tag{8}$$

および陰的中点公式

$$f^{b,a}(z^k,t^k) = f\left(\frac{z^{k+a}+z^{k+b}}{2}, \frac{t^{k+a}+t^{k+b}}{2}\right) \tag{9}$$

を考える。積分の２次近似 (6) の合成によって，高次の数値積分法が構成される。

2.2 直列型合成法

直列型合成法では，s 個の非整数ステップ

$$w_1,\ w_2, \ldots,\ w_s \tag{10}$$

$$\sum_{m=1}^{s} w_m = 1 \tag{11}$$

の２次のスキームを直列に合成した計算スキーム

$$\begin{cases} Z^{k+w_1+\cdots+w_m} = Z^{k+w_1+\cdots+w_{m-1}} \\ +I^{w_1+\cdots+w_m, w_1+\cdots+w_{m-1}} \\ (m=1,2,\ldots,s) \\ \\ Z^k = z^k,\ Z^{k+1} = z^{k+1},\ w_0 = 0 \end{cases} \tag{12}$$

を考える（図１）。ここで，Z^{k+a} は中間変数であり (12) は s 段のスキームとなっている。また

$$I^{b,a} = (b-a)\Delta t f^{b,a}(Z^k, t^k) \tag{13}$$

である。非整数ステップ $w_1,\ldots,w_s$ を適当に選べば高次のスキームとなる。ここで扱う具体的な数値は以下の通りである。

４次のスキーム (McLachlan [5])：

$$\begin{cases} s = 5 \\ w_1 = w_5 = 0.28 \\ w_2 = w_4 = 0.62546426846767004501 \\ w_3 = 1 - 2\sum_{j=1}^{2} w_j \end{cases} \tag{14}$$

６次のスキーム (Yoshida [6])：

$$\begin{cases} s = 7 \\ w_1 = w_7 = 0.78451361047755726382 \\ w_2 = w_6 = 0.23557321335935813368 \\ w_3 = w_5 = -1.17767998417887100695 \\ w_4 = 1 - 2\sum_{j=1}^{3} w_j \end{cases} \tag{15}$$

８次のスキーム (McLachlan [5])：

$$\begin{cases} s = 15 \\ w_1 = w_{15} = 0.74167036435061295345 \\ w_2 = w_{14} = -0.40910082580003159400 \\ w_3 = w_{13} = 0.19075471029623837995 \\ w_4 = w_{12} = -0.57382471116082266666 \\ w_5 = w_{11} = 0.29906418130365592384 \\ w_6 = w_{10} = 0.33462491824529818378 \\ w_7 = w_9 = 0.31529309239676659663 \\ w_8 = 1 - 2\sum_{j=1}^{7} w_j \end{cases} \tag{16}$$

2.3 並列型合成法

並列型合成法では，n 個の多段２次のスキームを並列に合成する。それぞれ段数を

$$s_1 < s_2 < \cdots < s_n \tag{17}$$

となる任意の正の整数として，中間変数を区間 $[t^k,\ t^{k+1}]$ を s_j $(j=1,\ldots,n)$ 等分した時間での値として

$$Z_j^{k+\frac{m}{s_j}} \tag{18}$$

$$(m=1,\ldots,s_j-1\,;j=1,\ldots,n) \tag{19}$$

で表す。s_j 段２次のスキームを重み c_j で並列に合成した計算スキーム

$$z^{k+1} = z^k + \sum_{j=1}^{n} c_j \sum_{m=1}^{s_j} I_j^{\frac{m}{s_j}, \frac{m-1}{s_j}} \tag{20}$$

$$Z_j^{k+\frac{m}{s_j}} = \frac{s_j - m}{s_j}\left(z^k + \sum_{l=1}^{m} I_j^{\frac{l}{s_j}, \frac{l-1}{s_j}}\right) + \frac{m}{s_j}\left(z^{k+1} - \sum_{l=m+1}^{s_j} I_j^{\frac{l}{s_j}, \frac{l-1}{s_j}}\right) \tag{21}$$

$$Z_j^k = z^k,\ Z_j^{k+1} = z^{k+1} \tag{22}$$

$$(j=1,2,\cdots,n\,;m=1,2,\cdots,s_j-1) \tag{23}$$

を考える（図２）。ここで

$$I_j^{b,a} = (b-a)\Delta t f^{b,a}(Z_j^k, t^k) \tag{24}$$

である。重み $c_1, c_2, \cdots, c_n$ を

$$c_j = \frac{s_j^{2n-2}}{\prod_{l=1(\neq j)}^{n} (s_j^2 - s_l^2)} \tag{25}$$

$$(j = 1, 2, \cdots, n\ ;\ n \geqq 2) \tag{26}$$

のように選ぶと，数値積分の誤差は $O(\Delta t^{2n+1})$ となり，$2n$ 次のスキームとなる。

3．数値計算例

3.1 例1

線形微分方程式

$$\frac{dz_1}{dt} = z_1 + e^t \tag{27}$$

$$z_1(0) = 1 \tag{28}$$

を考える。厳密解は

$$z_1(t) = (t+1)e^t \tag{29}$$

である。

並列型合成法での段数は

$$s_j = j\ (j = 1, 2, \ldots, n) \tag{30}$$

とし，計算時間 T_c は

$$T_c = 1 \tag{31}$$

とする。このとき，計算回数 K_c は $K_c = T_c/\Delta t$ で定まる。数値解の誤差 $e(t^k)$ は

$$e(t^k) = |z_1(t^k) - z_1^k| \tag{32}$$

で与えられる。

表1に $\Delta t = 0.1$ のときの $z_1^{K_c}$ の計算値を示す。省略記号として，2次のスキームとして台形公式を選んだときの $2n$ 次の直列型合成法を ST_{2n}，並列型合成法を PT_{2n} と表し，2次のスキームとして陰的中点公式を選んだときの $2n$ 次の直列型合成法を SM_{2n}，並列型合成法を PM_{2n} とした。直列型，並列型どちらの場合も陰的中点公式を用いる方が台形公式を用いるより誤差が若干小さい。また，並列型の方が直列型より誤差は小さい。図3に ST_2 の場合の誤差 $e(t^k)$ を示す。他の計算法でも同様であるが，この系では誤差は単調に時間とともに増大する。

表1　$\Delta t = 0.1$ のときの z_1^{10} の計算値

スキーム	数値 z_1^{10}	誤差 $e(1)$
正確な値	5.436563656918090	0
ST_2=PT_2	5.446777771185877	1.02×10^{-2}
SM_2=PM_2	5.443373534408262	6.80×10^{-3}
ST_4	5.436561093579508	2.56×10^{-6}
PT_4	5.436561673517383	1.98×10^{-6}
SM_4	5.436561866992457	1.78×10^{-6}
PM_4	5.436562204745151	1.45×10^{-6}
ST_6	5.436563684543017	2.76×10^{-8}
PT_6	5.436563657227880	3.09×10^{-10}
SM_6	5.436563676572398	1.96×10^{-8}
PM_6	5.436563657147549	2.29×10^{-10}
ST_8	5.436563656917681	4.09×10^{-13}
PT_8	5.436563656918058	3.26×10^{-14}
SM_8	5.436563656917815	2.75×10^{-13}
PM_8	5.436563656918066	2.45×10^{-14}

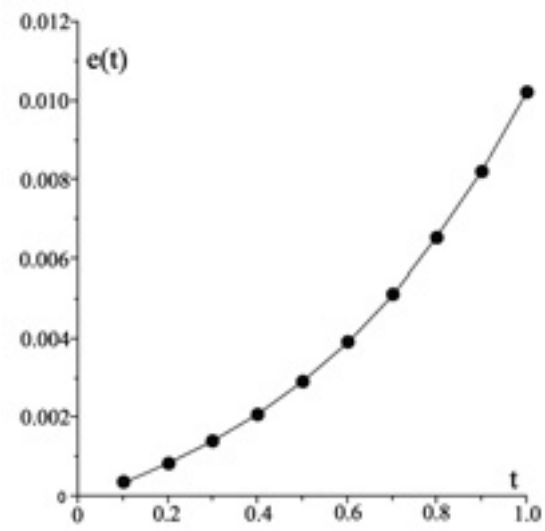

図3　誤差の時間変化 (ST_2)

図4と図5にそれぞれ2次のスキームとして台形公式を選んだ場合と陰的中点公式を選んだ場合の誤差 $e(T_c)$ と刻み幅 Δt のグラフを示す。破線は ST_{2n}，PT_{2n}，SM_{2n}，PM_{2n} に対して傾き $2n$ の直線である。グラフから $e(T_c) = O(\Delta t^{2n})$ であることがわかる。ST_{2n}，PT_{2n} のグラフと SM_{2n}，PM_{2n} のグラフが異なるので読みづらいが，表1と同様に SM_{2n}，PM_{2n} の方が ST_{2n}，PT_{2n} より誤差は若干小さい。即ち，誤差としては2次のスキームとして陰的中点公式を用いた方がよい。また，並列型の方が直列型より誤差は小さい。

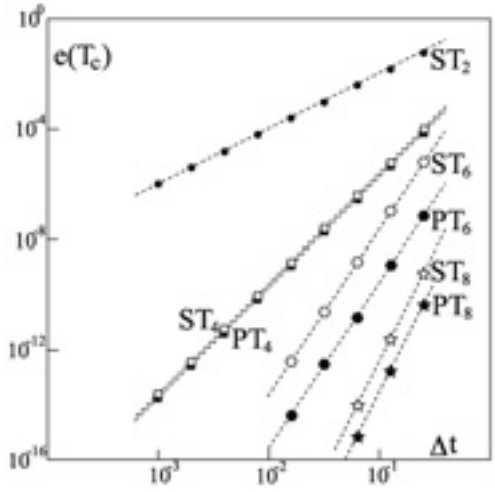

図４　誤差と刻み幅（台形公式の合成）

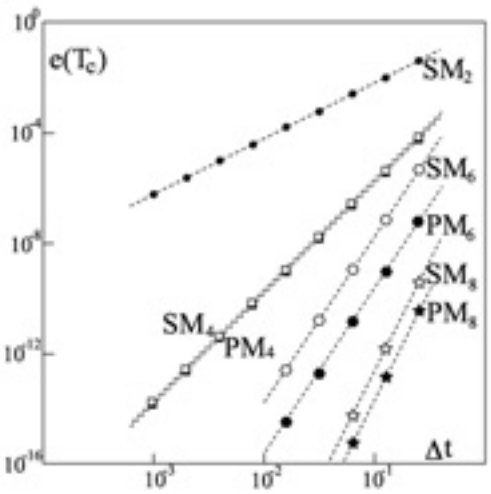

図５　誤差と刻み幅（陰的中点公式の合成）

計算コストは使用しているハードウエアーやソフトウエア，計算法の具体的なアルゴリズムやプログラムの書き方など様々な要素から決まるので，正確な分析は困難であるが，大まかな分析は可能である。ここでは前処理や後処理を除いた K_c ステップのくり返し計算の実行時間を比較し，実行時間の値そのものより相対的な傾向に焦点を合わせる。

図６に陰的中点公式を用いたときの計算実行時間と計算ステップのグラフを示す。どのスキームでも計算実行時間は概ね計算回数に比例して長くなっている。また，どの次数でも並列型合成法は直列型合成法より計算実行時間が長くなっており計算コストは高い。しかし，並列型では次数 $2n$ の増加によってあまり計算実行時間は長くなっていない。これは，大きな次数では，直列型の計算実行時間が並列型と大きな差がない可能性を示唆している。

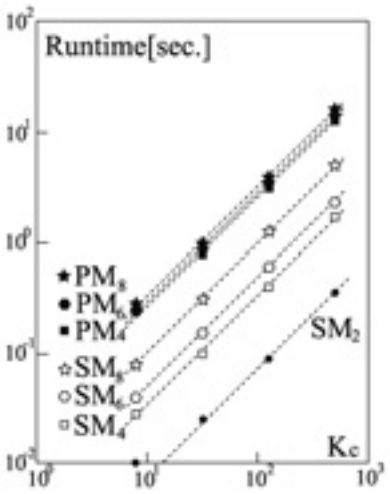

図６　計算実行時間と計算回数（陰的中点公式の合成）

図７に陰的中点公式の直列型合成法における $K_c = 128$ の場合の計算実行時間と段数 s のグラフを示す。計算実行時間が段数に比例して増加しているのがわかる。これは直感的な予想と矛盾しない。残念ながら，段数 s と次数 $2n$ との関係が明確ではないので，次数と計算実行時間との関係は読み取れない。なお，台形公式でもほとんど値は変わらない。

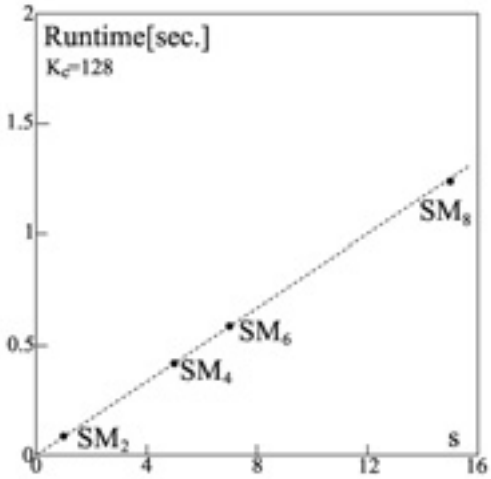

図７　計算実行時間と段数（直列型合成法）

図８に並列型合成法における $K_c = 128$ の場合の計算実行時間と次数 $2n$ のグラフを示す。次数が大きくなれば計算時間も長くなることがわかるが，かなり高次のスキームでも非常に長くなることはない。また，次数が小さくても数秒程度の固定的な計算コストが存在することがわかる。２次の場合でも，境界での中間変数 Z_1^k, Z_1^{k+1} を考えるのでその分無駄な計算をすることが原因と考えられる。

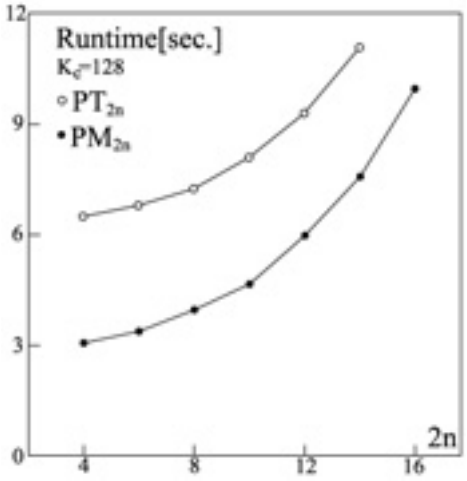

図８　計算実行時間と次数 (並列型合成法)

3.2 例２

非線形微分方程式

$$\frac{dz_1}{dt} = z_1(1 - z_1) \tag{33}$$

$$z_1(0) = 0.5 \tag{34}$$

を考える。厳密解は

$$z_1(t) = \frac{1}{1 + e^{-t}} \tag{35}$$

である。

並列型合成法での段数は

$$s_j = j \ (j = 1, 2, \ldots, n) \tag{36}$$

とし，計算時間 T_c は

$$T_c = 2 \tag{37}$$

とする。

表２に $\Delta t = 0.25$ のときの $z_1^{K_c}$ の計算値を示す。例１の場合と異なり，台形公式と陰的中点公式どちらの誤差が小さいかは次数によって様々である。また，ST_4，PT_4 を除けば並列型の方が直列型より誤差は小さい。いずれにしても，台形公式，陰的中点公式，直列型，並列型，どの方法を用いるのがよいのかは一概にいえない。

図９および図１０にそれぞれ台形公式および陰的中点公式を用いた場合の誤差 $e(t^k)$ を示す。誤差は単調に時間とともに増大するわけではなく，減少することもある。PT_4 は始めに ST_4 より誤差が小さいが，すぐに逆転して誤差が大きくなっている。一方，PM_4 は全般に SM_4 より誤差が大きく最終的に逆転して誤差が小さくなっている。

表２　$\Delta t = 0.25$ のときの z_1^8 の計算値

スキーム	数値 z_1^{10}	誤差 $e(2)$
正確な値	0.880797077977882	0
ST_2=PT_2	0.880640369817541	1.56×10^{-4}
SM_2=PM_2	0.881266949451895	4.69×10^{-4}
ST_4	0.880797058679045	1.92×10^{-8}
PT_4	0.880797338826003	2.60×10^{-7}
SM_4	0.880796882326922	1.95×10^{-7}
PM_4	0.880797181192899	1.03×10^{-7}
ST_6	0.880797080359314	2.38×10^{-9}
PT_6	0.880797077847340	1.30×10^{-10}
SM_6	0.880797081877165	3.89×10^{-9}
PM_6	0.880797077930136	4.77×10^{-11}
ST_8	0.880797077976391	1.49×10^{-12}
PT_8	0.880797077977881	1.38×10^{-15}
SM_8	0.880797077977803	7.91×10^{-14}
PM_8	0.880797077977914	3.18×10^{-14}

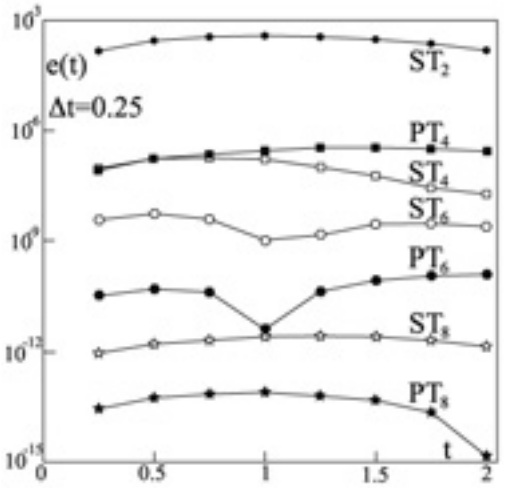

図９　誤差の時間変化 (台形公式の合成)

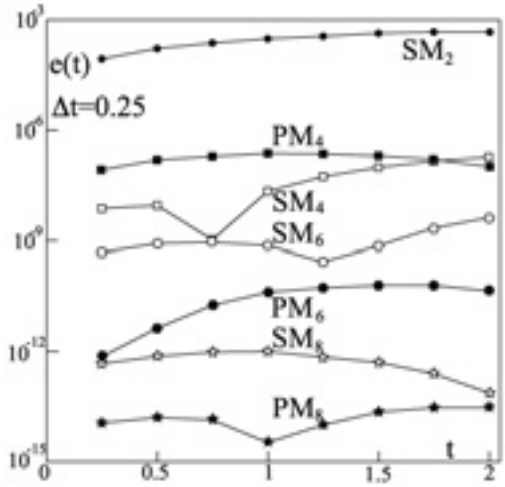

図１０　誤差の時間変化 (陰的中点公式の合成)

図１１と図１２にそれぞれ２次のスキームとして台形公式を選んだ場合と陰的中点公式を選んだ場合の誤差 $e(T_c)$ と刻み幅 Δt のグラフを示す。破線は例１と同様に ST_{2n}，PT_{2n}，SM_{2n}，PM_{2n} に対して傾き $2n$ の直線であり，$e(T_c) = O(\Delta t^{2n})$ である。台形公式の４次の合成を除いて，並列型合成法の方が誤差は小さい。

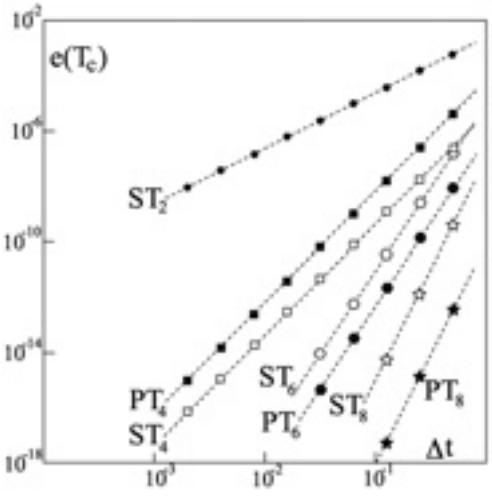

図１１　誤差と刻み幅 (台形公式の合成)

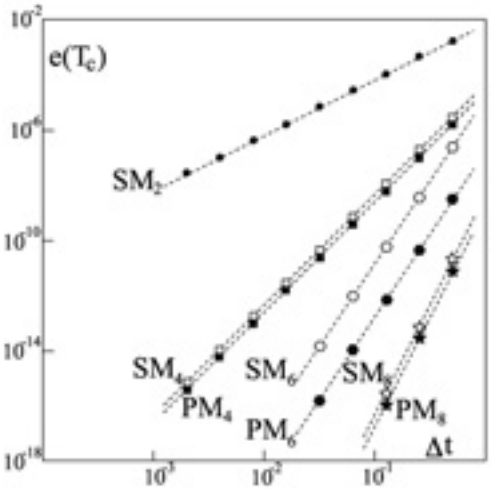

図１２　誤差と刻み幅 (陰的中点公式の合成)

図１３に台形公式の直列型合成法における $K_c = 128$ の場合の計算実行時間と段数 s のグラフを示す。例１と同様に計算実行時間が段数に比例して増加しているのがわかる。なお陰的中点公式でも値はほとんど変わらない。

図１４に並列型合成法における $K_c = 128$ の場合の計算実行時間と次数 $2n$ のグラフを示す。破線は $\mathrm{const.}2^n$ の直線を表す。例１の線形の系と異なり，次数が増大するとともにほぼ指数関数的に計算実行時間が増大することがわかる。したがってより高次になると並列型合成法の計算コストは非常に高くなる。

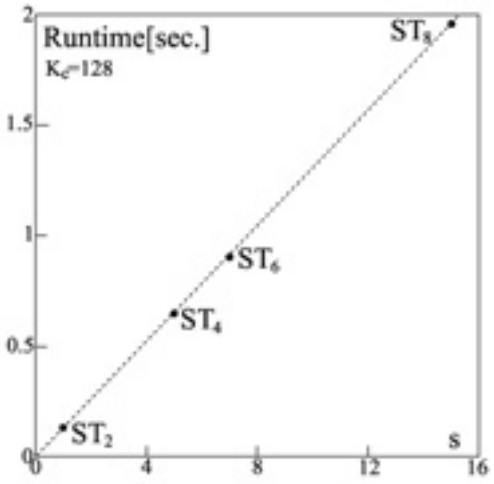

図１３　計算実行時間と段数 (直列型合成法)

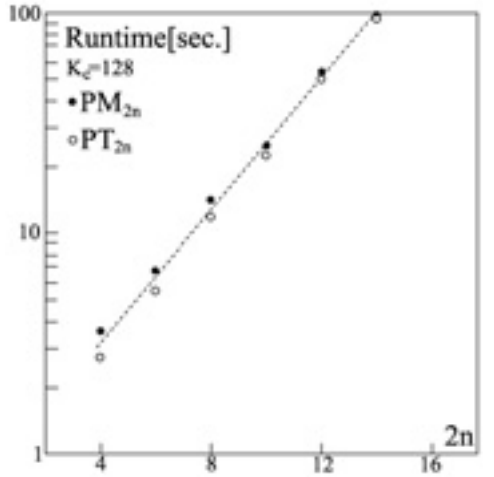

図１４　計算実行時間と次数 (並列型合成法)

４．おわりに

本研究では，常微分方程式に対する高次の数値積分法として，直列型および並列型の合成法について数値計算を通じた議論をした。精度については，概ね並列型の方が直列型より精度がよかった。計算コストについては，直列型の方が並列型より計算実行時間が短かかった。直列型では，ほぼ段数に比例して計算実行時間が長くなった。並列型では，線形系の場合，次数の増大とともに実行時間はゆるやかに増大した。しかし，非線形系の場合，次数の増大とともに指数関数的に実行時間が増大し，より高次の計算には多大な計算コストが必要であることがわかった。

参考文献

[1] E. Hairer, C. Lubich and G. Wanner: *Geometric Numerical Integration, Structure-Preserving Algorithms for Ordinary Differential Equations* (Springer, Berlin, 2006)2nd ed..

[2] B. Leimkuler and S. Reich: *Simulating Hamiltonian Dynamics* (Cambridge University Press, Cambridge, 2004).

[3] J.M. Sanz-Serna and M.P. Calvo: *Numerical Hamiltonian Problems* (Chapman & Hall, London, 1994).

[4] Y. Ishimori：A high-order energy-conserving integration scheme for Hamiltonian systems, Phys. Lett. A, **372** (2008) 1562-1573.

[5] R.I. McLachlan: On the numerical integration of ordinary differential equations by symmetric composition methods, SIAM J. Sci. Comput., **16** (1995) 151-168.

[6] H. Yoshida: Construction of higher order symplectic integrators, Phys. Lett. A, **150** (1990) 262-268.

On Computational Accuracy and Cost in Numerical Integrations of ODEs by the Parallel Composition Method

Yuji ISHIMORI

Department of Liberal Arts and Sciences, Faculty of Engineering

Summary

We consider the parallel composition scheme as a high-order numerical integration method for ordinary differential equations and study its computational accuracy and cost by doing numerical experiments.

Key Words: parallel composition, ordinary differential equations, numerical integration, high-order scheme, computational cost