Vol.2012-HPC-134 No.4
2012/6/1

長時間積分における誤差とその抑制法

小澤 一文[1,a)]

概要: Hamilton 系を数値積分するとき，解そのものの精度だけでなく，理論上一定である第一積分の値も精度よく保存されていることが望ましい．シンプレクティック解法はほぼこの条件を満たしている．しかし，シンプレクティック Runge–Kutta 法の場合，長時間積分においては各種誤差の累積のため，Hamiltonian の値が時刻 t に比例して増大していくことが報告されている．一方，シンプレクティックな線形多段解法では，この誤差は $t^{1/2}$ という Runge–Kutta 法よりは緩やかな速さで増大する (Brouwer's law) ため，長時間積分においては Runge–Kutta 法より有利である．本研究報告では，Runge–Kutta 法に部分的に 3 倍精度演算を組込み，Hamiltonian の誤差の増大を $t^{1/2}$ に比例した速さで増大する方法を提案する．

キーワード: シンプレクティック数値積分法，ルンゲ・クッタ法，長時間積分，Brouwer の法則，3 倍精度演算，第一積分

A Control Method of Errors in Long-Term Integration

Ozawa Kazufumi[1,a)]

Abstract: When solving the Hamilton canonical equation by numerical methods, it is important not only to solve the equation accurately, but also to preserve the values of the first integrals. Symplectic Runge–Kutta methods are shown to preserve these values very accurately. In practice, however, for standard implementations of the Runge–Kutta methods the Hamiltonian error behaves like $\mathrm{O}(t)$, where t is the time. This report proposes an implementation of Gauss Runge–Kutta methods, the most commonly used symplectic method, which suppresses the error growth up to $\mathrm{O}(t^{1/2})$ (which is called Brouwer's law) by using triple-precision arithmetic.

Keywords: symplectic numerical integration, Runge–Kutta method, long-term integration, Brouwer's law, triple-precision arithmetic, first integral

1. はじめに

常微分方程式系の数値積分において発生する丸め誤差については Henrici[6] によって十分に解析されてきた．その後，IEEE754[12] 規格の倍精度演算がハードウェアとして実装され日常的に用いられるようになったため，丸め誤差解析の研究は廃れてきた．しかし，最近の計算機の処理能力の飛躍的な向上に伴い，大規模な計算が頻繁に行われるようになり，倍精度演算を用いても誤差の累積が無視できないような例も報告されている．例えば微分方程式系の数値計算においては，理論上一定となるべき値，すなわち保存量 (第一積分) の保存が保証されている数値解法においても，長時間の積分においては必ずしも十分に保存されていないことがある [4], [15].

本研究報告では，シンプレクティック Runge–Kutta 法による Hamilton 系の数値積分において，誤差の増大をできるだけ緩やかにするような実装法を提案する．

2. Hamilton 系とシンプレクティック数値解法

次の Hamilton 正準方程式を考える：

[1] 秋田県立大学 システム科学技術学部
Yuri-Honjo, Akita, 015-0055, Japan
[a)] ozawa@akita-pu.ac.jp

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} &= \ \nabla_p H(\boldsymbol{q},\boldsymbol{p}),\\ \frac{\mathrm{d}\boldsymbol{p}}{\mathrm{d}t} &= -\nabla_q H(\boldsymbol{q},\boldsymbol{p}),\end{aligned}\qquad \boldsymbol{q},\boldsymbol{p}\in\mathbb{R}^d. \tag{1}$$

ここで，$\boldsymbol{q}$ は一般化座標，$\boldsymbol{p}$ はそれに対応した一般化運動量，$H(\boldsymbol{q},\boldsymbol{p})$ は全エネルギー Hamiltonian である．$H(\boldsymbol{q},\boldsymbol{p})$ の値は方程式 (1) の任意の解軌道上で一定になることが知られている．したがってそのうな性質を持った数値解法を用いることが望ましい．シンプレクティック解法はほぼこの性質を満たしている [5], [9]．具体的には，h を時間方向のステップサイズ，p を解法の次数としたとき，シンプレクティック解法では，Hamiltonian の誤差は $\mathrm{O}(h^p)$ であり時刻 t に対して一定である．一方，通常の解法では $\mathrm{O}(t\,h^p)$ となり，t に比例して増大していく．だが，シンプレクティック Runge–Kutta 法に対する標準的な実装では，通常の解法と同様に $\mathrm{O}(t\,h^p)$ という速さで誤差が増大していくこと報告されている [4].

ここでシンプレクティック Runge–Kutta 法における誤差について考える．

3. シンプレクティック Runge–Kutta 法の誤差

まず，Hamilton 系 (1) を

$$\boldsymbol{z}=\begin{pmatrix}\boldsymbol{q}\\ \boldsymbol{p}\end{pmatrix},\quad f(\boldsymbol{z})=\begin{pmatrix}\nabla_p H(\boldsymbol{q},\boldsymbol{p})\\ -\nabla_q H(\boldsymbol{q},\boldsymbol{p})\end{pmatrix},\quad \boldsymbol{z},\,f\in\mathbb{R}^{2d}$$

とし，通常の正規形に書き換える：

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t}=f(\boldsymbol{z}(t)). \tag{2}$$

この方程式に対する Runge–Kutta 法は

$$\begin{cases}\boldsymbol{z}_{n+1}=\boldsymbol{z}_n+h\displaystyle\sum_{i=1}^{s}b_i\,f(\boldsymbol{Z}_i),\\ \boldsymbol{Z}_i=\boldsymbol{z}_n+h\displaystyle\sum_{j=1}^{s}a_{ij}\,f(\boldsymbol{Z}_j),\quad i=1,\ldots,s\end{cases} \tag{3}$$

である．ここでは Runge–Kutta 法 (3) がシンプレクティックであるとする．

シンプレクティック Runge–Kutta 法を用いて Hamilton 系を数値積分するとき，Hamiltonian $H(\boldsymbol{q},\boldsymbol{p})$ 関して，以下の 3 種類の誤差が存在する [4]:

- t に依存しない誤差 — E_C で表す
- 速さ $\mathrm{O}(t^{1/2})$ で成長する誤差 — E_B で表す
- 速さ $\mathrm{O}(t)$ で成長する誤差 — E_L で表す

E_C は離散化誤差そのものであり，大きさは，解法の次数を p，ステップサイズを h とすれば

$$E_C=\mathrm{O}(h^p)$$

となる．Runge–Kutta 法の公式 (3) が正確に計算されていれば E_C のみとなる．

一方，E_B は Runge–Kutta 法における加算

$$\begin{aligned}\boldsymbol{z}_{n+1}&=\boldsymbol{z}_n+\delta_n,\\ \delta_n&=h\sum_{i=1}^{s}b_i\,f(\boldsymbol{Z}_i)\end{aligned} \tag{4}$$

において，一般に $|\,\boldsymbol{z}_n\,|\gg|\,\delta_n\,|$ であるために生じる丸め誤差 (μ_n とする) が原因となるものである．この誤差 μ_n を各ステップにおいて独立で平均 0 のノイズとして考える．すなわち

$$E[\mu_l]\approx 0,\quad E[\mu_l\,\mu_k]\ \approx\ \varepsilon_B^2\,\delta_{lk} \tag{5}$$

とする．ここで ε_B は加算 (4) における相対精度である．この統計的モデルとシンプレクティック積分法の後退誤差解析 [4], [5] によって解析すると，時刻 $t\,(=n\,h)$ では

$$E_B\sim\sqrt{n}\,\varepsilon_B=\sqrt{t/h}\,\varepsilon_B \tag{6}$$

となる [4].

次に 3 番目の誤差 E_L について考える．方程式 (2) に対して対称行列 $C\in\mathbb{R}^{2d\times 2d}$ が存在して

$$\boldsymbol{z}^T\,C\,f(\boldsymbol{z})=0,\quad \boldsymbol{z}\in\mathbb{R}^{2d} \tag{7}$$

ならば，

$$Q(t)=\boldsymbol{z}^T\,C\,\boldsymbol{z} \tag{8}$$

によって定義される $Q(t)$ は，方程式 (2) の 2 次の保存量になる．ここで $Q_0=Q(t_0)$ とおけば，Runge–Kutta 法による $t_1(=t_0+h)$ における $Q(t)$ の近似値 Q_1 は

$$\begin{aligned}Q_1=Q_0&+2\,h\sum_{i=1}^{s}b_i\,\boldsymbol{Z}_i^T\,C\,f(\boldsymbol{Z}_i)\\ &+h^2\sum_{i,j=1}^{s}f(\boldsymbol{Z}_i)^T\,m_{ij}\,C\,f(\boldsymbol{Z}_j),\\ m_{ij}&=b_i\,b_j-b_i\,a_{ij}-b_j\,a_{ji}\end{aligned} \tag{9}$$

を満たす [5]．上式において右辺第 2 項は式 (7) より 0 になる．また，シンプレクティック Runge–Kutta 法においては第 3 項の係数 m_{ij} はすべて 0 になる [5], [13]．これより 2 次の保存量だけでなくシンプレクティック構造も保存される [1], [5]．係数 a_{ij}, b_i を相対精度 ε_L で計算したとすると，第 n ステップにおいては，ある定数 $K_L>0$ が存在して

$$|\,E_L\,|=|Q_n-Q_0|<K_L\,n\,h^2\,\varepsilon_L=K_L\,t\,h\,\varepsilon_L \tag{10}$$

と見積もられる．

全誤差はこれら 3 つの誤差の和になると考えられるから，t が大きくなっていけばやがては E_L が支配的になり計算が破綻する．ここでは，IEEE754 規格の倍精度計算に

おいてできるだけ長時間

$$|E_B| \gg |E_L|, \qquad |E_B| \gg |E_C| \tag{11}$$

となるような，すなわち Brouwer の法則 [2] が成り立っているような実装法を考える．そのためには，離散化誤差が丸め誤差のレベルまで小さくなるようなステップサイズ h を選び，さらに

$$\varepsilon_B \gg \varepsilon_L$$

とする必要がある．

4. 誤差の増大を抑制した Runge–Kutta 法の実装

次に，シンプレクティック Runge–Kutta 法における 3 つの誤差の抑制法とその実装について考える．

4.1 E_C とその制御

離散化誤差 E_C を小さくするためには，高次の解法を用い h をある程度小さくしなければならない．ここでは 3 ∼ 10 段の Gauss 型 Runge–Kutta 法を用い，ステップサイズは $h = 2^{-4} \sim 2^{-7}$ の範囲とする．また内部反復は不動点反復法

$$\boldsymbol{Z}_i^{(\nu+1)} = \boldsymbol{z}_n + h \sum_{j=1}^{s} a_{ij}\, f(\boldsymbol{Z}_j^{(\nu)}), \quad \nu = 0, 1, \ldots \tag{12}$$

を用い，反復停止条件を

$$D^{(\nu)} = 0, \quad \text{or} \quad D^{(\nu)} \le D_n^{(\nu+1)} \tag{13}$$

とする．ここで

$$D^{(\nu)} = \max_i \| \boldsymbol{Z}_i^{(\nu)} - \boldsymbol{Z}_i^{(\nu-1)} \|$$

である．すなわち Runge–Kutta 法の式 (3) が有限桁の範囲で正確に成り立つまで内部反復を行う．

4.2 E_B とその制御

まず E_B を小さくするためには ε_B を小さくする必要がある．そのためには式 (4) の加算にて compensated summation[7] を用いる．具体的には以下のような計算を行う：

この方法によって加算を行えば，$\mathrm{O}(h)$ の大きさである δ_n の下位桁まで補償されるので

$$\varepsilon_B \sim h\,\varepsilon_M \tag{14}$$

となる．ここで ε_M は倍精度計算におけるマシン・エプシロンで

$$\varepsilon_M \approx 2.22 \times 10^{-16}$$

である．したがって，式 (6) より

$$E_B \sim \sqrt{t\,h}\,\varepsilon_M \tag{15}$$

となる．

Compensated Summation

1: $\varepsilon_0 = 0$
2: **for** $n = 0$ to $\cdots$ **do**
3: $\quad t_n = \delta_n \oplus \varepsilon_n$
4: $\quad \boldsymbol{z}_{n+1} = \boldsymbol{z}_n \oplus t_n$
5: $\quad \varepsilon_{n+1} = t_n \ominus (\boldsymbol{z}_{n+1} \ominus \boldsymbol{z}_n)$
6: **end for**

図 1 加算 $\boldsymbol{z}_{n+1} = \boldsymbol{z}_n + \delta_n$ における compensated summation.
Fig. 1 Compensated summation in the addition $\boldsymbol{z}_{n+1} = \boldsymbol{z}_n + \delta_n$.

4.3 E_L とその制御法

条件 (11) が成立するためには，すなわち

$$\sqrt{t\,h}\,\varepsilon_M \gg t\,h\,\varepsilon_L$$

であるためには

$$\varepsilon_L \ll \frac{\varepsilon_M}{\sqrt{t\,h}} \tag{16}$$

としなければならない．いま，$t = 10^8$, $h = 10^{-2}$ においてこの式を成り立たせるためには

$$\varepsilon_L \ll 2.2 \times 10^{-19} \approx 2^{-62}$$

となる．IEEE754 規格で推奨され，実際に x86 系のプロセッサに実装されている拡張倍精度では，仮数部長が 64 bit なので [11], [12]，この条件を満たすにはやや不足していると思われる．また，ほとんどの Fortran コンパイラでは 4 倍精度演算が実装されている．4 倍精度演算では，通常，仮数部長が 112 bit なので条件 (16) は十分に満たしている．しかし 4 倍精度演算はソフトウェアによる実装のため，その計算コストは非常に大きい (倍精度の 50∼100 倍)．そこで，ステージの計算

$$\boldsymbol{Z}_i = \boldsymbol{z}_n + h \sum_{i=1}^{s} a_{ij}\, f(\boldsymbol{Z}_j) \tag{17}$$

を 3 倍精度で行うことにする．3 倍精度演算に関してはいくつか方法が提案されているが (例えば [8], [10])，ここでは IEEE754 規格の倍精度に合った計算法を用いる．

まず Runge–Kutta 法の係数 a_{ij}, b_i と関数値 $f(\boldsymbol{Z}_i)$ を以下のように分割する：

$$a_{ij} = a_{ij}^{(2)} + a_{ij}^{(1)} + a_{ij}^{(0)}, \qquad b_i = b_i^{(2)} + b_i^{(1)} + b_i^{(0)},$$
$$f_i = f_i^{(1)} + f_i^{(0)}.$$

ここで各々の長さは

$$a_{ij}^{(2)} : 26\text{bits}, \quad a_{ij}^{(1)} : 27\text{bits}, \quad a_{ij}^{(0)} : 26\text{bits},$$
$$b_i^{(2)} : 26\text{bits}, \quad b_i^{(1)} : 27\text{bits}, \quad b_i^{(0)} : 26\text{bits},$$
$$f_i^{(1)} : 26\text{bits}, \quad f_i^{(0)} : 27\text{bits}$$

とし，1 つの倍精度変数に格納する．すなわち 3 つの倍精

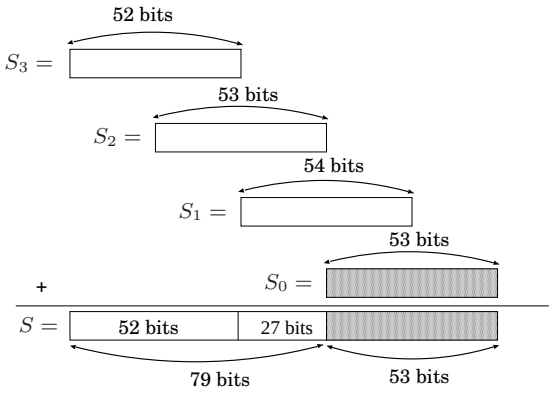

図 2　3 倍精度演算による $\sum_{j=1}^{s} a_{ij} f_j$ の計算.

Fig. 2　Computation of $\sum_{j=1}^{s} a_{ij} f_j$ by triple-precision arithmetic

度変数によって 3 倍精度変数 1 つを実現する．係数の分割は計算が始まる前に行っておくが，関数 f はステップ毎に倍精度演算で評価し 上位と 下位に分割する．このように分割した後に，$S = \sum_{j=1}^{s} a_{ij} f_j$ の計算を 以下のように部分和に分割して行う：

$$
\begin{aligned}
S = \sum_{j=1}^{s} a_{ij} f_j = \sum_{j=1}^{s} a_{ij}^{(2)} f_j^{(1)} &:= S_3 \\
&+ \left(\sum_{j=1}^{s} a_{ij}^{(2)} f_j^{(0)} + \sum_{j=1}^{s} a_{ij}^{(1)} f_j^{(1)} \right) := S_2 \\
&+ \left(\sum_{j=1}^{s} a_{ij}^{(1)} f_j^{(0)} + \sum_{j=1}^{s} a_{ij}^{(0)} f_j^{(1)} \right) := S_1 \\
&+ \sum_{j=1}^{s} a_{ij}^{(0)} f_j^{(0)} := S_0
\end{aligned}
$$

ここで各部分和の長さは，S_3 は 53 bits，S_2 は 52 bits，S_1 は 54 bits，S_0 は 53 bits となる．図 2 より，3 倍精度，すなわち倍精度の 3/2 倍 (79.5bits) の精度を得るためには S_0 の計算は不要であることがわかる．そこで S_0 は計算せず，下から順に S_1, S_2, S_3 を compensated summation で計算し h を掛けた後に，この 3 つをやはり compensated summation で加え $\boldsymbol{z}_n$ に加え $\boldsymbol{z}_{n+1}$ を求める．

5. 数値実験

前節で提案した誤差の低減法を組み合わせた 5 つの実装法で実験する．具体的には，以下の 3 つの問題に対して表 1 に示した 5 つの実装法を試し，保存量の変動を調べる．計算機環境は 2 の通りである．

(1) Kepler 問題

$$
H(\boldsymbol{q}, \boldsymbol{p}) = -\frac{1}{\|\boldsymbol{q}\|} + \frac{1}{2} \boldsymbol{p}^T \boldsymbol{p} \qquad \boldsymbol{q}, \boldsymbol{p} \in \mathbb{R}^2, \quad (18)
$$

表 1　s 段 Gauss 型 Runge–Kutta 法の 5 つの実装法

Table 1　5 implementations of the s-stage Gauss Runge–Kutta methods

	update formula	iteration	evaluation of f
rkg$s$0			d
rkg$s$1	c		d
rkg$s$2	c	z	d
rkg$s$3	c+t	z	d
rkg$s$4	c+t	z+t (only once)	d

s: number of the stages of the Runge–Kutta method
c: compensated summation,
d: double precision, t: triple precision
z: zero tolerance itr. by eq. (13), t: triple precision

表 2　計算機環境

Table 2　Computing environment

OS	Linux ver. 2.6.18 (Red Hat)
Compiler	ifort 11.1 (Opt 2)
CPU	Xeon X3450 (2.67GHz)

$$
\begin{aligned}
\boldsymbol{q}(0) &= (1 - e, 0)^T, \\
\boldsymbol{p}(0) &= \left(0, \sqrt{(1+e)/(1-e)}\right)^T, \\
0 &\le e < 1.
\end{aligned}
$$

ここで離心率 e を 0.6 とする．5 段と 10 段の Runge–Kutta 法で解いた結果を図 3, 4 に示す．

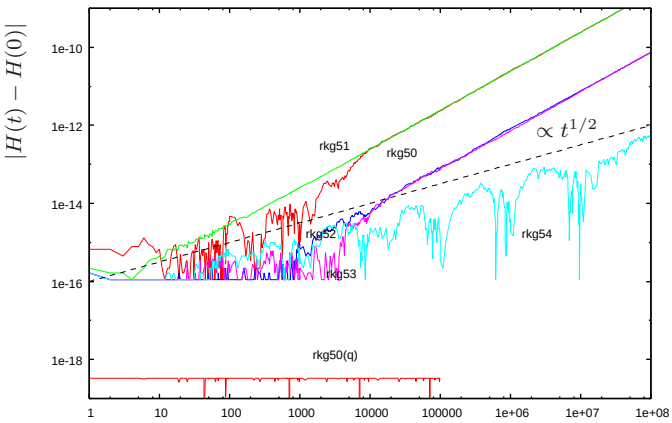

図 3　5 段 Gauss Runge–Kutta 法の 5 つの実装によって Kepler 問題を解いたときの $H(\boldsymbol{q}, \boldsymbol{p})$ の変動.

Fig. 3　Long-term behaviours of $H(\boldsymbol{q}, \boldsymbol{p})$ when solving the Kepler problem by the five implementations of 5-stage Gauss Runge–Kutta method.

(2) Hénon–Heiles 問題

$$
\begin{aligned}
H(\boldsymbol{q}, \boldsymbol{p}) &= \frac{1}{2}(q_1^2 + q_2^2) + q_1^2 q_2 - \frac{1}{3} q_2^3 + \frac{1}{2}(p_1^2 + p_2^2), \\
\boldsymbol{q} &= (q_1, q_2)^T, \quad \boldsymbol{p} = (p_1, p_2)^T.
\end{aligned} \qquad (19)
$$

ここでは $H(\boldsymbol{q}, \boldsymbol{p}) = 0.8$ となるような $\boldsymbol{q}, \boldsymbol{p}$ の初期値を選ぶ．結果を図 5，6 に示す．

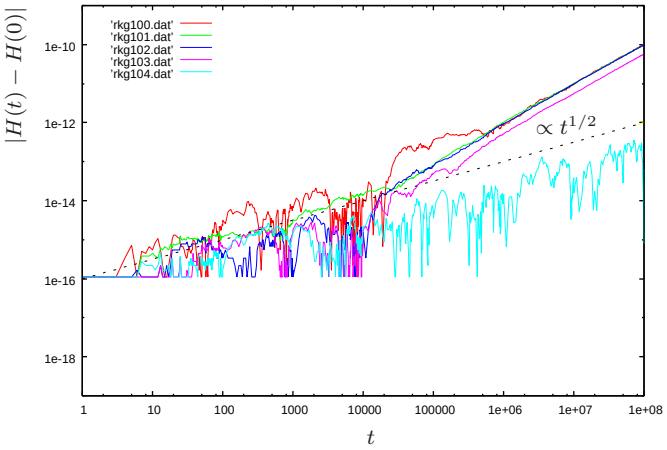

図 4　10 段 Gauss Runge–Kutta 法の 5 つの実装によって Kepler 問題を解いたときの $H(\boldsymbol{q}, \boldsymbol{p})$ の変動.

Fig. 4　Long-term behaviours of $H(\boldsymbol{q}, \boldsymbol{p})$ when solving the Kepler problem by the five implementations of 10-stage Gauss Runge–Kutta method.

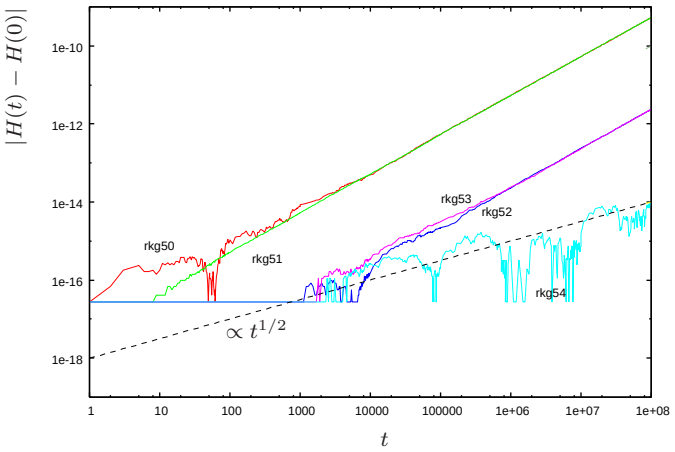

図 5　5 段 Gauss Runge–Kutta 法の 5 つの実装によって Hénon–Heiles 問題を解いたときの $H(\boldsymbol{q}, \boldsymbol{p})$ の長時間における振る舞い.

Fig. 5　Long-term behaviours of $H(\boldsymbol{q}, \boldsymbol{p})$ when solving the Hénon–Heiles problem by the five implementations of 5-stage Gauss Runge–Kutta method.

(3) 剛体問題

トルクフリーの状態で回転している剛体の運動を考える (図 7). 質量中心を原点とする 3 つの主軸のまわりの慣性モーメントを I_1, I_2, I_3，角運動量を z_1, z_2, z_3 としたとき，z_1, z_2, z_3 は次式で与えられ運動方程式 (Euler 方程式) を満たす [14]:

$$\begin{cases} \dot{z_1}(t) = a_1\, z_2(t)\, z_3(t), \\ \dot{z_2}(t) = a_2\, z_3(t)\, z_1(t), \\ \dot{z_3}(t) = a_3\, z_1(t)\, z_2(t). \end{cases} \tag{20}$$

ここで

$$a_1 = (I_2 - I_3)/I_1,\ \ a_2 = (I_3 - I_1)/I_2,\ \ a_3 = (I_1 - I_2)/I_3.$$

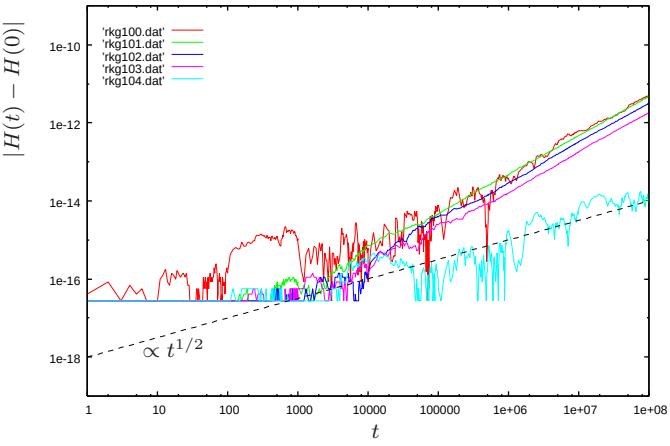

図 6　10 段 Gauss Runge–Kutta 法の 5 つの実装によって Hénon–Heiles 問題を解いたときの $H(\boldsymbol{q}, \boldsymbol{p})$ の長時間における振る舞い.

Fig. 6　Long-term behaviours of $H(\boldsymbol{q}, \boldsymbol{p})$ when solving the Hénon–Heiles problem by the five implementations of 10-stage Gauss Runge–Kutta method.

である．この方程式は以下の2 つの保存量

$$Q_1(t) = z_1^2 + z_2^2 + z_3^2,$$
$$Q_2(t) = \frac{1}{2}\,(I_1^{-1}\, z_1^2 + I_2^{-1}\, z_2^2 + I_3^{-1}\, z_3^2).$$

を持っている [14]．初期値は $z_1(0) = 0, z_2(0) = 1, z_3(0) = 1$ とする．計算結果を図 8，9 に示す．

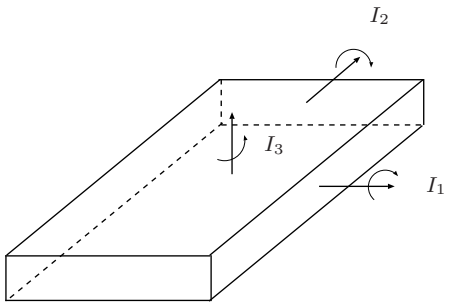

図 7　トルクフリーの状態で回転する剛体.

Fig. 7　Torque free motion of a rigid body.

以上の実験より，`rkg54`，`rkg104` などによって Brouwer の法則が達成されていることがわかる．

次に 5 段と 10 段の Gauss 型 Runge–Kutta 法の 5 つの実装法の計算時間を比較する．ここでは，Kepler 問題を $0 \le t \le 10^6$ の範囲で解いた場合の結果を 3, 4 に示す．

これまでの考察および実験結果より，Brouwer の法則を達成するにはできるだけステップサイズを小さくすることが望ましいことがわかる．一方，計算コストを考えれば当然ステップサイズは大きい方が望ましい．だが，ステップサイズを大きくすれば，E_C, E_L などが E_B を凌駕し同法則は成り立たなくなる (図 10)．そこで `rkg`s`4` にてステップサ

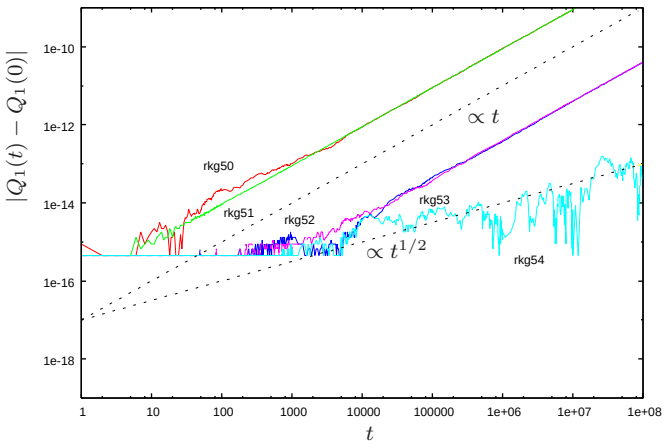

図 8　5 段 Gauss Runge–Kutta 法の 5 つの実装によって剛体問題 (20) を解いたときの保存量 Q_1 の変動.

Fig. 8　Long-term behaviours of the invariant Q_1 when solving the rigid-body problem (20) by various implementations of 5-stage Runge–Kutta method.

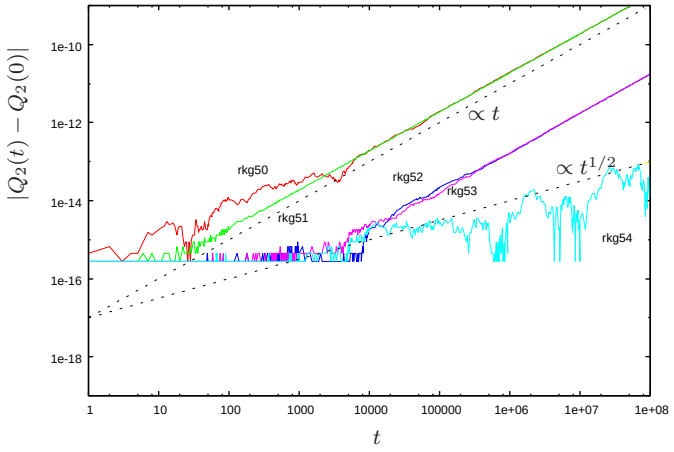

図 9　5 段 Gauss Runge–Kutta 法の 5 つの実装によって剛体問題 (20) を解いたときの保存量 Q_2 の変動.

Fig. 9　Long-term behaviours of the invariant Q_2 when solving the rigid-body problem (20) by various implementations of 5-stage Runge–Kutta method.

表 3　5 段 Gauss Runge–Kutta 法の各種実装法と計算時間の比較

Table 3　CPU times of the various implementations of 5-stage Gauss Runge–Kutta method ($h = 2^{-6}$, $0 \le t \le 10^6$)

Implementations	sec (ratio)	Error growth in H
`rkg50`	74.9 (1.00)	O (t)
`rkg51`	77.8 (1.04)	O (t)
`rkg52`	110 (1.47)	O (t)
`rkg53`	130 (1.74)	O (t)
`rkg54`	215 (2.87)	O ($t^{1/2}$)
`rkg50(q)`	8830 (118)	—

`rkg500(q)` は `rkg100` の 4 倍精度版

イズを h を $2^{-8}, 2^{-7}, \ldots, 2^{-3}$ と大きくしていき，Brouwer の法則を達成する最大のステップサイズを数値実験から求めることにする．その限界のステップサイズでの計算時間

表 4　10 段 Gauss Runge–Kutta 法の各種実装法と計算時間の比較 ($h = 2^{-5}$, $0 \le t \le 10^6$).

Table 4　CPU times of the various implementations of 10-stage Gauss Runge–Kutta method ($h = 2^{-5}$, $0 \le t \le 10^6$).

Implementations	sec (ratio)	Error growth in H
`rkg100`	112 (1.00)	O (t)
`rkg101`	113 (1.01)	O (t)
`rkg102`	143 (1.28)	O (t)
`rkg103`	163 (1.46)	O (t)
`rkg104`	317 (2.83)	O ($t^{1/2}$)
`rkg100(q)`	15550 (139)	—

`rkg100(q)` は `rkg100` の 4 倍精度版

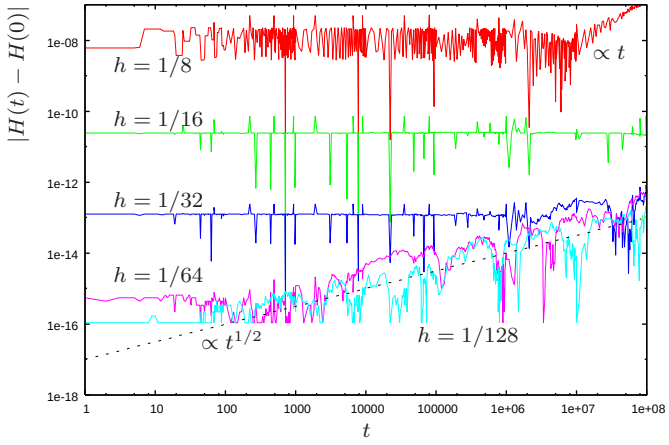

図 10　4 段 Gauss Runge–Kutta 法の実装 rkg44 における $H(\boldsymbol{q}, \boldsymbol{p})$ の長時間における変動.

Fig. 10　Long-term behaviours of Hamiltonian $H(\boldsymbol{q}, \boldsymbol{p})$ with rkg44 implementation of the 4-stage GaussRunge–Kutta .

を表 5 に示す．なお，ここで解いた問題は Kepler 問題で時刻 t は $0 \le t \le 10^6$ の範囲である．

表 5　rkg$s$4 (s は段数で $s = 3, \ldots, 10$) において Brouwer の法則を達成する限界のステップサイズ h と計算時間

Table 5　Maximum step-size and CPU times achieving Brouwer's law with rkg$s$4 implementations, where $s\,(= 3, \ldots, 10)$ is the number of stages.

stage s	h	cpu time [sec]
3	1/256	4.23×10^2
4	1/64	1.58×10^2
5	1/32	1.14×10^2
6	1/16	7.95×10
7	1/16	9.96×10
8	1/16	1.19×10^2
9	1/16	1.44×10^2
10	1/8	8.94×10

表 5 に示されている結果より段数 s は 6 位がちょうどよいようである．

6. Runge–Kutta–Nyström 法について

方程式 (1) が，式

$$\ddot{\boldsymbol{q}} = g(\boldsymbol{q}) \tag{21}$$

のように書き表されるとき，Runge–Kutta 法よりも Runge–Kutta–Nyström 法

$$\begin{cases} \boldsymbol{q}_{n+1} = \boldsymbol{q}_n + h\,\boldsymbol{p}_n + h^2 \sum_{i=1}^{s} \bar{b}_i\, g(\boldsymbol{Q}_i), \\ \boldsymbol{p}_{n+1} = \boldsymbol{p}_n + h \sum_{i=1}^{s} b_i\, g(\boldsymbol{Q}_i), \\ \boldsymbol{Q}_i = \boldsymbol{q}_n + c_i\, h\,\boldsymbol{p}_n + h^2 \sum_{j=1}^{s} \bar{a}_{ij}\, g(\boldsymbol{Q}_j). \end{cases} \tag{22}$$

を用いた方が効率がよい．Runge–Kutta–Nyström 法にも Runge–Kutta 法と同じ誤差の制御法を適用し，Kepler 問題を解いた結果を図 11 に示す．また，CPU time の比較を表 7 に示しておく．

ここで注目すべきは，Runge–Kutta–Nyström 法では内部反復に 3 倍精度演算を行わない実装法 (rkng$s$3) においても Brouwer の法則を達成しているということである．標準的な実装 (rkng$s$0) のおおよそ 2 倍の時間で Brouwer の法則が達成されていることが表 7 からわかる．Runge–Kutta–Nyström 法でにおいてより効率的に Brouwer の法則を達成できる理由は，式 (22) のステージ $\boldsymbol{Q}_i$ を計算する式に（ h ではなく）h^2 が掛っていることであると思われるが，これに関してはより厳密な理論解析が必要である．

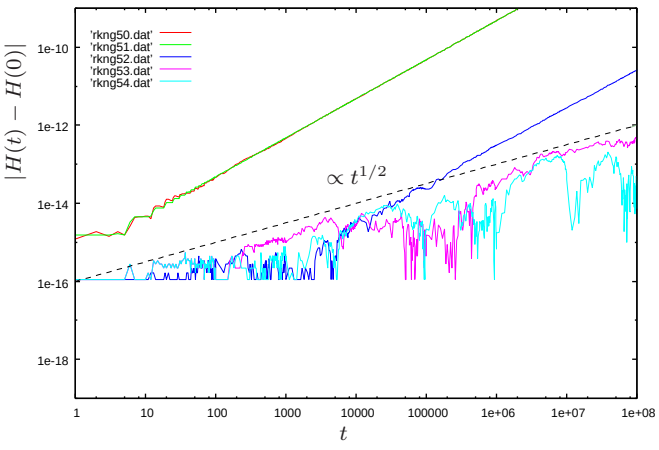

図 11　5 段 Gauss Runge–Kutta–Nyström 法の 5 つの実装によって Kepler 問題を解いたときの $H(\boldsymbol{q}, \boldsymbol{p})$ の変動.

Fig. 11　Long-term behaviours of $H(\boldsymbol{q}, \boldsymbol{p})$ when solving the Kepler problem by the five implementations of 5-stage Gauss Runge–Kutta–Nyström method.

以上の結果より Runge–Kutta–Nyström 法ではより少ないコストで Brouwer の法則が達成できることがわかった．

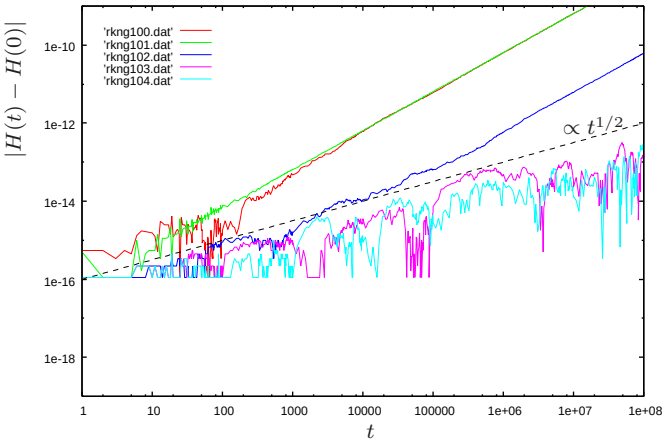

図 12　10 段 Gauss Runge–Kutta–Nyström 法の 5 つの実装によって Kepler 問題を解いたときの $H(\boldsymbol{q}, \boldsymbol{p})$ の変動.

Fig. 12　Long-term behaviours of $H(\boldsymbol{q}, \boldsymbol{p})$ when solving the Kepler problem by the five implementations of 10-stage Gauss Runge–Kutta–Nyström method.

表 6　5 段 Gauss 型 Runge–Kutta–Nyström 法における CPU time の比較 ($h = 2^{-6}$, $0 \le t \le 10^6$).

Table 6　CPU times of the 5-stage Gauss Runge–Kutta–Nyström method ($h = 2^{-6}$, $0 \le t \le 10^6$).

Implementations	sec (ratio)	Error growth in H
rkng50	39.0 (1.00)	O (t)
rkng51	43.0 (1.10)	O (t)
rkng52	58.1 (1.49)	O (t)
rkng53	86.0 (2.20)	O ($t^{1/2}$)
rkng54	136 (3.49)	O ($t^{1/2}$)
rkng50(q)	3814 (97.8)	—

表 7　10 段 Gauss 型 Runge–Kutta–Nyström 法における CPU time の比較 ($h = 2^{-6}$, $0 \le t \le 10^6$).

Table 7　CPU times of the 10-stage Gauss Runge–Kutta–Nyström method ($h = 2^{-6}$, $0 \le t \le 10^6$).

Implementations	sec (ratio)	Error growth in H
rkng100	44.3 (1.00)	O ()(t)
rkng101	46.4 (1.05)	O ()(t)
rkng102	70.2 (1.58)	O ()(t)
rkng103	97.4 (2.20	O ()($t^{1/2}$)
rkng104	180 (4.06)	O ()($t^{1/2}$)
rkng100(q)	5630 (127)	—

7. まとめ

本研究報告では主に Hamilton 系の長時間積分において生ずる誤差を解析し，Hamiltonian の誤差の増大をできるだけ緩やかにする，すなわち $\mathrm{O}(t)$ から $\mathrm{O}(t^{1/2})$ にするシンプレクティック Runge–Kutta 法の実装を提案した．また，同じ実装法を Runge–Kutta–Nyström 法にも適用しよりよい結果を得た．

今後は，より厳密な誤差解析（特に Runge–Kutta–Nyström 法について）および大規模問題の並列計算機への実装などが考えられる．

参考文献

[1] Cooper, G.J.: Stability of Runge–Kutta Methods for Trajectory Problems, IMA J. Numer. Anal. Vol. 7, pp.1–13, (1987).
[2] Brouwer, D.: On the Accumulation of Errors in Numerical Integration, Astron. J., Vol. 46, pp.149–153 (1937).
[3] Fukushima, T.: Reduction of Round-off Error in Symplectic Integrators, Astron. J., Vol. 121, pp.1768–1775 (2001).
[4] Hairer, E. Mclachlan, R.I. and Razakarivony, A.: Achieving Brouwer's Law with Implicit Runge–Kutta Methods, *BIT*, Vol. 48, pp.231–243 (2008).
[5] Hairer, E., Lubich, C. and Wanner, G.:*Geometric Numerical Integration*, Springer, Berlin (2001).
[6] Henrici, P.:*Discrete Variable Methods in Ordinary Differential Equations*, John Wiley & Sons, Inc. New York (1961).
[7] Higham, N.J.: *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia (1996).
[8] Ikebe, Y. : Note on Triple-Precision Floating-Point Arithmetic with 132 bit Numbers, *Commu ACM* Vol. 8, pp.175–177 (1965).
[9] Leimkuhler, B. and Reich, S.:*Simulating Hamiltonian Dynamics*, Cambridge University Press, Cambridge (2004).
[10] 椋木大地，高橋大介: GPU による 3 倍精度浮動小数点演算の検討, 情報処理学会研究報告，Vol. 2011-HPC-132, pp.1–9.
[11] Muller, J. et al. : *Handbook of Floating-point Arithmetic*, Birkäuser, Boston (2010).
[12] Overton, M. L. :*Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia (2001).
[13] Sanz-Serna, J. M. and Calvo, M. P.: *Numerical Hamiltonian Problems*, Chapman & Hall, London (1993).
[14] Taylor, J. R.:*Classical Mechanics*, University science Books, Sauslito, California (2005).
[15] Vilmart, G: Reducing Round-Off Errors in Rigid Body Dynamics, Journal of Computational Physics, Vol. 227, pp.7083–7088 (2008).