数理解析研究所講究録
第1957巻 2015年 55-67

長時間積分における丸め誤差の影響とその改良
Round-off error analysis in long-term integration and an improved implementation

小澤 一文
KAZUFUMI OZAWA

秋田県立大学 システム科学技術学部
Faculty of Systems Science and Technology, Akita Prefectural University

1 はじめに

1990年代初め頃からどの計算機にもFPUが内蔵されるようになり，IEEE754規格 [11] の倍精度演算が単精度計算と同程度のコストで行えるようになった。そのため倍精度演算が日常的に使われるようになり，丸め誤差の累積に配慮することなく数値計算が行われるようになった。一方，最近の計算機の計算速度の向上は目覚しいものがあり，デスクトップ機でも数10GFlopsにも及び，最高速のスーパーコンピュータでは数10PFlopsのピーク性能を誇るようになった [15]。そうなると一昔前までは事実上不可能であった長時間におよぶ計算が現実的な時間で実行可能になり，倍精度計算でも丸め誤差の累積が無視できない時代が到来しつつあると言える。

最近では，微分方程式系の数値計算において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を用いたとしても，長時間積分においては，丸め誤差のため必ずしもこれらの量が十分に保存されない例が報告されている [3, 14]。本研究報告では，まず，長時間における丸め誤差の振る舞いについて研究し，その後，シンプレクティック Runge–Kutta 法を用いた 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プログラミング手法を提案する。

2 長時間計算における丸め誤差の累積と不変量

まず初めに漸化式

$$z_{n+1} = z_n + \delta_n, \qquad n = 0, 1, 2, \ldots \tag{1}$$

の計算を考える。この式から生成される数列 $\{z_n\}$ は不変量 $I(z)$ を持つものとする。すなわち

$$I(z_n) = I(z_0), \qquad n = 1, 2, \ldots$$

とする。漸化式 (1) の計算において

$$|z_n| \gg |\delta_n|$$

を仮定する。この仮定より実際の計算では δ_n の下位桁は z_n に足されないで誤差となる。式 (1) によって実際に計算される値を $\tilde{z}_n$ とすれば，$\tilde{z}_n$ は

$$\tilde{z}_{n+1} = \tilde{z}_n + \delta_n + \lambda_n, \qquad \tilde{z}_0 = z_0 \tag{2}$$

という変化をすることになる。ここで λ_n は第 n ステップの加算で生じる丸め誤差である。この式より，$\tilde{z}_n$ と z_n の関係は

$$\tilde{z}_n = z_n + \sum_{i=0}^{n-1} \lambda_i \tag{3}$$

となる。

以下では Henrici[5] が行ったように局所丸め誤差 λ_i を確率変数とみなし統計的な解析を行う。まず以下の仮定を設ける：

$$\begin{aligned} &\mathbb{E}[\lambda_i] = \mu, \quad i = 0, 1, \dots \\ &\mathbb{E}[(\lambda_i - \mu)(\lambda_j - \mu)] = \sigma^2 \delta_{ij}, \quad i, j = 0, 1, \dots, \end{aligned} \tag{4}$$

上式において $\mathbb{E}[\cdot]$ は期待値を表し，δ_{ij} はクロネッカーデルタである。この仮定より，第 n ステップでの誤差を $e_n (= \tilde{z}_n - z_n)$ とすると

$$e_n = \sum_{i=0}^{n-1} \lambda_i$$

となるので，

$$\mathbb{E}[e_n] = n\mu, \qquad \mathrm{Var}[e_n] = n\sigma^2 \tag{5}$$

である。これより不変量 $I(z)$ に関して

$$I(\tilde{z}_n) \approx I(z_0) + (n\mu + \chi) I_z, \tag{6}$$

を得る。ここで $I_z = \nabla_z I$ である。また χ は平均値からの変動を表し，仮定 (4) より中心極限定理が適用され，e_n の分布はほぼ正規分布とみなせる。よって，かなり高い確率で $\chi \in (-\sqrt{n}\sigma, \sqrt{n}\sigma)$ となる。これより $\mu = 0$ の場合は不変量 $I(z)$

$$I(\tilde{z}_n) = I(z_0) + \mathrm{O}(\sqrt{n}\sigma), \qquad n \to \infty \tag{7}$$

となり，$\mu \neq 0$ かつ $\sqrt{n} \ll n$ と仮定すれば

$$I(\tilde{z}_n) = I(z_0) + \mathrm{O}(n\mu), \qquad n \to \infty \tag{8}$$

という漸近的な振る舞いを示すことになる。Brouwer [2] は Hamilton 系の長時間積分において一定であるべき Hamiltonian の値が $\sqrt{n}$ に比例して増大していくことを予測している。これは以上の解析より $\mu \approx 0$ の場合に相当することがわかる。

次に，いま示した誤差の漸近的な振る舞いを確認するため単位円上の点の回転の問題を考える：

$$\begin{aligned} x_{n+1} &= c\,x_n - s\,y_n, \quad x_0 = 1, \\ y_{n+1} &= s\,x_n + c\,y_n, \quad y_0 = 0 \end{aligned}, \qquad n = 0, 1, 2, \cdots \tag{9}$$

ここで $c = \cos\alpha$, $s = \sin\alpha$ と置いた。式 (9) の計算では，明らかに

$$I(x_n, y_n) := x_n^2 + y_n^2, \qquad n = 0, 1, 2, \dots$$

が不変量であり $I(x_0, y_0) = I(x_1, y_1) = \cdots = 1$ である。いま $0 < \alpha \ll 1$ を仮定する。この仮定より，$c \approx 1$ かつ $s \approx \alpha$ となる。x_n, y_n の計算値を $\tilde{x}_n, \tilde{y}_n$ とすると，これら計算値は

$$\begin{aligned} \tilde{x}_{n+1} &= \tilde{c}\tilde{x}_n - \tilde{s}\tilde{y}_n + u_n, \qquad \tilde{x}_0 = x_0 \\ \tilde{y}_{n+1} &= \tilde{s}\tilde{x}_n + \tilde{c}\tilde{y}_n + v_n, \qquad \tilde{y}_0 = y_0 \end{aligned} \tag{10}$$

を満たす。この式で u_n と v_n は加算における丸め誤差であり，$\tilde{c}$ と $\tilde{s}$ はそれぞれ $\cos\alpha$ と $\sin\alpha$ の計算値である。$x_n = \mathrm{O}(1)$, $y_n = \mathrm{O}(1)$ なので

$$u_n = O(\varepsilon_M), \qquad v_n = \mathrm{O}(\varepsilon_M)$$

と仮定できる。ここで ε_M はマシンエプシロンとする。IEEE754 規格の倍精度では $\varepsilon_M \approx 2.22 \times 10^{-16}$ であり，また同規格では対称な丸めが行われているので

$$\mathbb{E}[u_n] = \mathbb{E}[v_n] = 0$$

と仮定できる。また c と s における誤差を $e_c := \tilde{c} - c$ および $e_s := \tilde{s} - s$ とすれば，仮定 $0 < \alpha \ll 1$ より

$$e_c = \mathrm{O}(\varepsilon_M), \qquad e_s = \mathrm{O}(\alpha\,\varepsilon_M)$$

を得る。I_n の計算値を $\tilde{I}_n$ とすると，以上の結果より

$$\begin{aligned} \tilde{I}_{n+1} &= (\tilde{c}^2 + \tilde{s}^2)\,\tilde{I}_n + 2(\tilde{c}\tilde{x}_n - \tilde{s}\tilde{y}_n)u_n + 2(\tilde{s}\tilde{x}_n + \tilde{c}\tilde{y}_n)v_n + \mathrm{O}(\varepsilon_M^2), \\ \tilde{I}_0 &= I_0 = 1 \end{aligned} \tag{11}$$

を得る。ここで

$$\begin{aligned} \tilde{c}^2 + \tilde{s}^2 &= c^2 + s^2 + 2(c\,e_c + s\,e_s) + \mathrm{O}(\varepsilon_M^2) = 1 + R + \mathrm{O}(\varepsilon_M^2), \\ R &:= 2(c\,e_c + s\,e_s) = \mathrm{O}(\varepsilon_M) \end{aligned}$$

である。式 (11) の右辺第 2 項と第 3 項の期待値は 0 とみなせるので

$$\tilde{I}_n \approx (1+R)^n \tilde{I}_0 \approx 1 + n\,R$$

なので，

$$\mathbb{E}\,[\tilde{I}_n] \approx 1 + 2\,n\,\varepsilon_M \tag{12}$$

を得る。ここで $\alpha = 10^{-4}$ として $n = 1, 2, \ldots, 10^{12}$ まで計算したものを図 1 に示す。この図より不変量 I の誤差はほぼ $n\varepsilon_M$ に比例して大きくなっていることがわかる。これは誤差がバイアスを持っているからである。

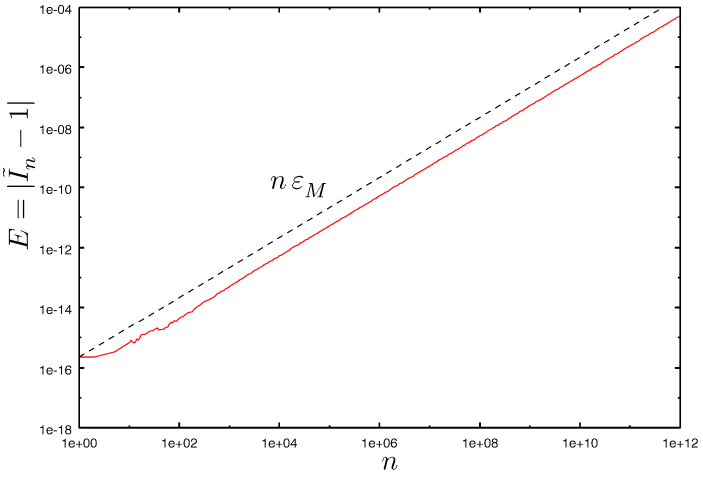

図 1: 式 (9) における不変量 $\tilde{I}_n = \tilde{x}_n^2 + \tilde{y}_n^2$ の誤差

以下ではバイアスを消去することを考える。バイアスの原因は，誤差を含んだ $\cos\alpha$ および $\sin\alpha$ の計算値 $\tilde{c}$ と $\tilde{s}$ を使い続けるからである。$0 < \alpha \ll 1$ なので，それぞれの誤差は

$$e_c = \tilde{c} - \cos\alpha = \mathrm{O}(\varepsilon_M), \qquad e_s = \tilde{s} - \sin\alpha = \mathrm{O}(\alpha\varepsilon_M) \tag{13}$$

と見積もれる。以下では，パラメータの計算における誤差を上に示したオーダよりも小さいオーダで計算し，バイアスを実質的に消去することを考える。

まず，式 (9) を

$$\begin{aligned} x_{n+1} &= x_n + (d\,x_n - s\,y_n), \\ y_{n+1} &= y_n + (s\,x_n + d\,y_n) \end{aligned} \tag{14}$$

と変形する。ここで $d = c - 1\,(= \cos\alpha - 1)$ である。d の計算を

$$d = -\frac{\sin^2\alpha}{\cos\alpha + 1} \tag{15}$$

という形で計算すれば，d の計算における丸め誤差はおおよそ $-\sin^2\alpha \approx -\alpha^2$ に ε_M を掛けた程度の大きさになる。すなわち

$$e_d := \tilde{d} - d = \mathrm{O}(\alpha^2\varepsilon_M) \tag{16}$$

となる。式 (14) は，実際は

$$\begin{aligned} \tilde{x}_{n+1} &= \tilde{x}_n + (\tilde{d}\,\tilde{x}_n - \tilde{s}\,\tilde{y}_n) + u_n, \\ \tilde{y}_{n+1} &= \tilde{y}_n + (\tilde{s}\,\tilde{x}_n + \tilde{d}\,\tilde{y}_n) + v_n \end{aligned} \tag{17}$$

という式で計算したことになる。ここで u_n, v_n は n ステップの加算における丸め誤差である。これにより I_n の計算値 $\tilde{I}_n$ は

$$\tilde{I}_{n+1} = \left(1 + (\tilde{d}^2 + \tilde{s}^2 + 2\tilde{d})\right)\tilde{I}_n + 2\,(\tilde{x}_n u_n + \tilde{y}_n v_n) + \mathrm{O}(\varepsilon_M^2) \tag{18}$$

となる。ここで式 (13), (16) より

$$\begin{aligned} \tilde{d}^2 + \tilde{s}^2 + 2\tilde{d} &= d^2 + s^2 + 2\,d + 2(de_d + se_s + e_d) = 2(de_d + se_s + e_d) \\ &= \mathrm{O}(\alpha^2\varepsilon_M) \end{aligned}$$

である。したがって

$$\tilde{I}_n \approx \left(1 + \mathrm{O}(\alpha^2\varepsilon_M)\right)^n \tilde{I}_0 + \text{式 (18) 右辺第 2 項の和}$$

となる。ここで式 (18) 右辺第 2 項の和の期待値は 0 と仮定できるので

$$\tilde{I}_n = 1 + \mathrm{O}(n\,\alpha^2\,\varepsilon_M) + \mathrm{O}(\sqrt{n}\,\varepsilon_M) \tag{19}$$

を得る。この式において n に関して線形に増加する項は依然として存在するが，式 (12) と比べ，その係数は $n\varepsilon_M$ から $n\alpha^2\varepsilon_M$ に減少していることがわかる。前の実験と同様 $n = 10^{12}$ および $\alpha = 10^{-4}$ とすれば

$$n\,\alpha^2 < \sqrt{n}.$$

となるので，誤差はかなり長い時間 $\mathrm{O}(\sqrt{n}\,\varepsilon_M)$ という振る舞いをすることが予想される。実際に上記のパラメータで実験した結果を図 2 に示す。図 2 より式 (14) および (15) による計算では Brouwer の法則が達成されていることが分かる。

上の実験ではパラメータの計算精度を向上させ，誤差のバイアスをほぼ消去し Brouwer の法則を達成させた。しかし式 (17) に示すように増分を足すときに誤差 u_n, v_n が依然として発生する。この加算における誤差を Kahan のアルゴリズム (Kahan's compensated summation)[6, 8]

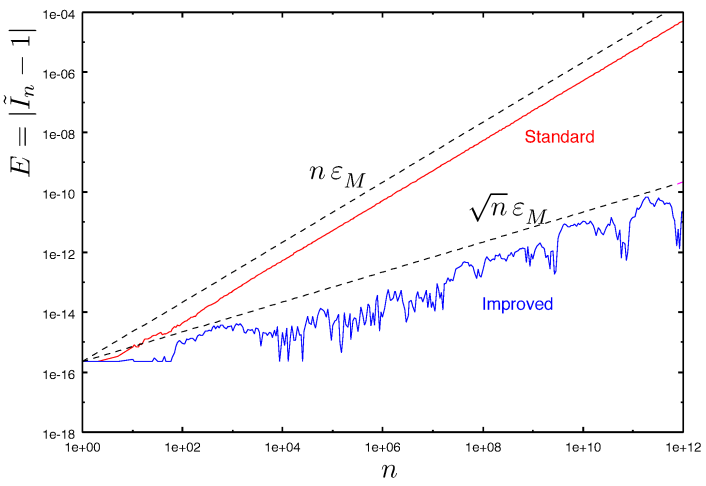

図 2: 式 (14) における不変量 $\tilde{I}_n = \tilde{x}_n^2 + \tilde{y}_n^2$ の変化

を用いて相殺することを考える。加算における誤差が消去されると，$\tilde{I}_n$ の漸近的な振る舞いは，式 (19) から $\mathrm{O}(\sqrt{n}\,\varepsilon_M)$ の項が消え

$$\tilde{I}_n = 1 + \mathrm{O}(n\,\alpha^2\,\varepsilon_M), \qquad n \to \infty \tag{20}$$

となるはずである。実際に実験した結果を図 3 に示す。この図より理論で予測される通りの振る舞いをしていることがわかる。

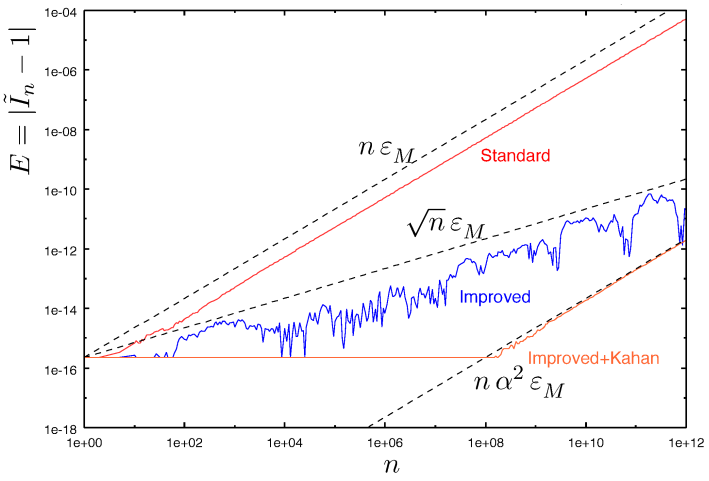

図 3: Kahan のアルゴリズムを用いたときの不変量 $\tilde{I}_n = \tilde{x}_n^2 + \tilde{y}_n^2$ の変化

3 Hamilton 系の長時間積分への応用

前節の実験より，長時間計算では，不変量 I の計算値 $\tilde{I}_n$ は

$$\tilde{I}_n = I + \mathrm{O}(c_1\sqrt{n}) + \mathrm{O}(c_2\,n), \qquad n \to \infty$$

という振る舞いを示すことがわかった。ここで c_1, c_2 は n に依存しない定数である。仮に $0 < c_2 \ll c_1$ であればかなり長い時間誤差が抑制され Brouwer の法則が達成されることになる。また Kahan の計算法を用いれば加算における丸め誤差も相殺することができ，誤差は $\mathrm{O}(c_2 n)$ の項だけになり，かなり長い時間 I の値を限界精度で計算できることがわかった。

以上の知見を Hamilton 系の数値積分に応用する。ここで次の微分方程式系を考える。

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = J\nabla H(\boldsymbol{z}), \qquad \boldsymbol{z} \in \mathbb{R}^{2d}, \quad J = \begin{pmatrix} 0 & I_d \\ -I_d & 0 \end{pmatrix} \tag{21}$$

ここで $\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{p})^T$ であり，$H(\boldsymbol{z})$ は Hamiltonian である。よく知られているように Hamiltonian H の値は解軌道上で一定である [13]。Hamilton 系を数値積分するときは Symplectic 法を用いれば Hamiltonian の値は $\mathrm{O}(h^p)$ の精度で保存される [4]。ここで h は解法の刻み幅であり，p は解法の次数である。しかし Symplectic Runge–Kutta 法の通常の実装では Hamiltonian H の値は時間 t に比例して増大する [3]。文献 [3] では Gauss 型 Runge–Kutta 法の計算の一部を倍精度演算から有理演算に置き換えることによって $t \approx 10^7$ 程度まで Brouwer の法則を達成した例が報告されている。本報告では Hairer 等 [3] とは別な方法で Gauss 型 Runge–Kutta 法を用いて $t \approx 10^8$ 程度まで Brouwer の法則を達成することを目的とする。

4 シンプレクティック Runge–Kutta 法の誤差

まず，Hamilton 系 (21) を通常の正規形に書き換える：

$$\frac{\mathrm{d}\boldsymbol{z}}{\mathrm{d}t} = \boldsymbol{f}(\boldsymbol{z}(t)) \tag{22}$$

この方程式に対する Runge–Kutta 法は

$$\begin{cases} \boldsymbol{z}_{n+1} = \boldsymbol{z}_n + h \displaystyle\sum_{i=1}^{s} b_i \boldsymbol{f}(\boldsymbol{Z}_i), \\ \boldsymbol{Z}_i = \boldsymbol{z}_n + h \displaystyle\sum_{j=1}^{s} a_{ij} \boldsymbol{f}(\boldsymbol{Z}_j), \quad i = 1, \ldots, s \end{cases} \tag{23}$$

である。ここでは Runge–Kutta 法 (23) 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 [3]：

- 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，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 (23) 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

$$\boldsymbol{z}_{n+1} = \boldsymbol{z}_n + \delta_n, \qquad \delta_n = h \sum_{i=1}^{s} b_i \boldsymbol{f}(\boldsymbol{Z}_i) \tag{24}$$

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統計的に独立で平均 0 のノイズとして考える。すなわち

$$\mathbb{E}[\mu_l] \approx 0, \qquad \mathbb{E}[\mu_l \mu_k] \approx \varepsilon_B^2 \delta_{lk} \tag{25}$$

とする。ここで ε_B は加算 (24) 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 [3, 4] によって解析すると，時刻 $t(= n h)$ では

$$E_B \sim \sqrt{n}\,\varepsilon_B = \sqrt{t/h}\,\varepsilon_B \tag{26}$$

となる [3]。

次に 3 番目の誤差 E_L について考える。方程式 (22) に対して対称行列 $C \in \mathbb{R}^{2d \times 2d}$ が存在して

$$\boldsymbol{z}^T C \boldsymbol{f}(\boldsymbol{z}) = 0, \quad \boldsymbol{z} \in \mathbb{R}^{2d} \tag{27}$$

ならば，

$$Q(t) = \boldsymbol{z}^T C \boldsymbol{z} \tag{28}$$

によって定義される量は，方程式 (22) 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

$$\begin{aligned} Q_1 &= Q_0 + 2h \sum_{i=1}^{s} b_i \boldsymbol{Z}_i^T C \boldsymbol{f}(\boldsymbol{Z}_i) + h^2 \sum_{i,j=1}^{s} \boldsymbol{f}(\boldsymbol{Z}_i)^T m_{ij} C \boldsymbol{f}(\boldsymbol{Z}_j), \\ m_{ij} &= b_i b_j - b_i a_{ij} - b_j a_{ji} \end{aligned} \tag{29}$$

を満たす [4]。上式において右辺第 2 項は式 (27) 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 [4]。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, 4]。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

$$|E_L| = |Q_n - Q_0| < K_L n h^2 \varepsilon_L = K_L t h \varepsilon_L \tag{30}$$

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

$$|E_B| \gg |E_L|, \qquad |E_B| \gg |E_C| \tag{31}$$

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。

5 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 3 つの誤差の抑制法を実装する。

5.1 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 $h = 2^{-3} \sim 2^{-7}$ の範囲とする。また内部反復は不動点反復法

$$\boldsymbol{V}_i^{(\nu+1)} = h \sum_{j=1}^{s} a_{ij} \boldsymbol{f}(\boldsymbol{V}_j^{(\nu)} + \boldsymbol{z}_n), \quad \nu = 0, 1, \ldots \tag{32}$$

を用い，反復停止条件を

$$D^{(\nu)} = 0, \quad \text{or} \quad D^{(\nu)} \leq D_n^{(\nu+1)} \tag{33}$$

とする。ここで

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

である。すなわち Runge–Kutta 法の式 (23) が有限桁の範囲で正確に成り立つまで内部反復を行う。

5.2 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 (24) の加算にて Kahan のアルゴリズム [6, 8] を用いる。具体的には以下のような計算を行う：

Kahan's compensated summation algorithm

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 4: 式 $\boldsymbol{z}_{n+1} = \boldsymbol{z}_n + \delta_n$ の計算における Kahan のアルゴリズム

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

$$\varepsilon_B \sim h\varepsilon_M \tag{34}$$

となる。したがって，式 (26) より

$$E_B \sim \sqrt{t\,h}\,\varepsilon_M \tag{35}$$

である。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。

5.3 E_L とその制御法

条件 (31) が成立するためには，すなわち

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

であるためには

$$\varepsilon_L \ll \frac{\varepsilon_M}{\sqrt{t\,h}} \tag{36}$$

としなけれならない。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。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 [10, 11]，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 (36) 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

$$\boldsymbol{Z}_i = \boldsymbol{z}_n + h \sum_{i=1}^{s} a_{ij}\, \boldsymbol{f}(\boldsymbol{Z}_j) \tag{37}$$

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 [7, 9])，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 $\boldsymbol{f}(\boldsymbol{Z}_i)$ を Vertkamp の分割法 [10] を用いて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 $\boldsymbol{f}$ 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：

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

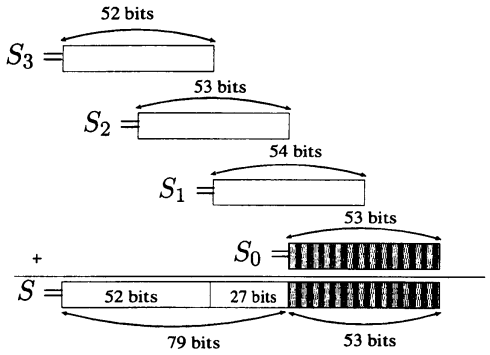

図 5: 三倍精度を用いた $\sum_{j=1}^{s} a_{ij} f_j$ の計算

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 5 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 Kahan のアルゴリズムで加え h を掛けた後に，やはり Kahan のアルゴリズムで $\boldsymbol{z}_n$ に加え $\boldsymbol{z}_{n+1}$ を求める。

6 数値実験

前節で提案した誤差の低減法を組み合わせた以下の 5 つの実装法を試す。ここでは Kepler 問題

表 1: s 段ガウス RK 法の 5 つの実装法

	update formula	iteration	evaluations 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. (33), t: triple precision

$$\begin{cases} H(\boldsymbol{q},\boldsymbol{p}) = -\dfrac{1}{\|\boldsymbol{q}\|} + \dfrac{1}{2}\boldsymbol{p}^T\boldsymbol{p} \qquad \boldsymbol{q},\boldsymbol{p} \in \mathbb{R}^2, \\ \boldsymbol{q}(0) = (1-e,0)^T, \quad \boldsymbol{p}(0) = \left(0, \sqrt{(1+e)/(1-e)}\right)^T, \\ 0 \le e < 1. \end{cases} \tag{38}$$

を解く。ここで離心率 e を 0.6 とする。5 段 Gauss 型 RK 法で解いた結果を図 6 に示す。この実験より，`rkg54` 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 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表 3 に示す。

表 2: 計算環境

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

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 7)。そこで `rkgs4` にてステップサイズを 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を表 4 に示す。なお，ここで解いた問題は Kepler 問題で時刻 t は $0 \leq t \leq 10^6$ の範囲である。表 4 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。

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。なお，Runge–Kutta–Nyström 法への実装ならびに他の方程式系への適用した詳細なデータは [12] に掲載されている。今後は，より厳密な誤差解析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] Hairer, E., Mclachlan, R.I. and Razakarivony, A.: Achieving Brouwer's Law with Implicit Runge–Kutta Methods, *BIT*, Vol. 48, pp.231–243 (2008).

[4] Hairer, E., Lubich, C. and Wanner, G.:*Geometric Numerical Integration*, Springer, Berlin (2001).

[5] Henrici, P.:*Discrete Variable Methods in Ordinary Differential Equations*, John Wiley & Sons, Inc. New York (1961).

[6] Higham, N.J.: *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia (1996).

[7] Ikebe, Y. : Note on Triple-Precision Floating-Point Arithmetic with 132 bit Numbers, *Commu ACM* Vol. 8, pp.175–177 (1965).

[8] Kahan, W.: Further remarks on reducing truncation errors, Commun. ACM 8 (1965), p.40.

[9] 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.

[10] Muller, J. et al. : *Handbook of Floating-point Arithmetic*, Birkäuser, Boston (2010).

[11] Overton, M. L. :*Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia (2001).

[12] Ozawa, K. : 長時間数値積分において保存量を保存するプログラミングテクニック,「第 12 回 常微分方程式の数値解法とその周辺」報告集, Mar. 13–16, 2012.

[13] Taylor, J. R.:*Classical Mechanics*, University science Books, Sauslito, California (2005).

[14] Vilmart, G: Reducing Round-Off Errors in Rigid Body Dynamics, Journal of Computational Physics, Vol. 227, pp.7083–7088 (2008).

[15] TOP 500 site: `http://www.top500.org/`

表 3: 5 段ガウス型 RK 法の CPU 時間の比較 ($h = 2^{-6}$, $0 \le t \le 10^6$)

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)	—

`rkg50(q)` は `rkg50` の 4 倍精度版

表 4: rkg$s$4 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

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

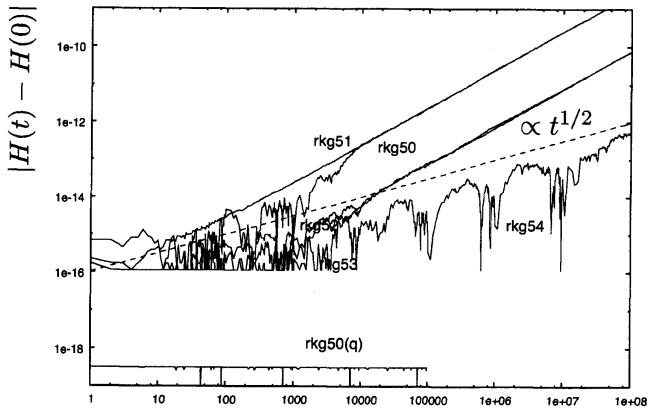

図 6: 5 段ガウス型 RK 法の 5 つの実装法の比較：Kepler 問題における $H(\boldsymbol{q}, \boldsymbol{p})$ の変化

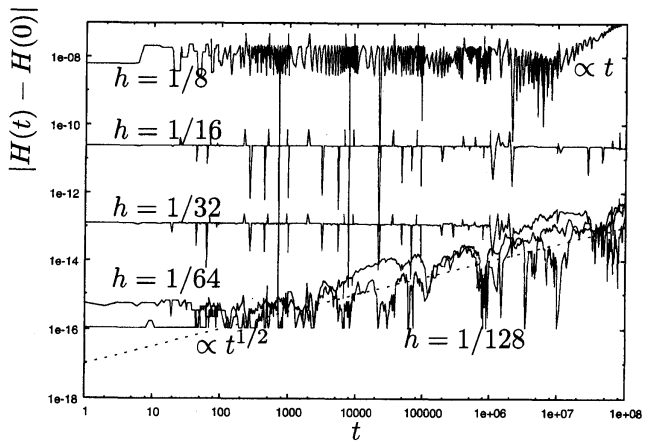

図 7: 4 段ガウス型 RK 法の実装法 (`rkg44`) におけるステップサイズ h と H の変化の関係