Rössler Model の数値的性質について

幸谷智紀* tkouya@cs.sist.ac.jp

2000 年 6 月 7 日 (水)–9 日 (金)

1 初めに

Rössler Model は Chaos 現象が見られる比較的簡単な 3 次元力学系の一つである [6]。

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -(y+z) \\ x+\alpha y \\ \beta + z(x-\mu) \end{bmatrix} \tag{1}$$

ここでは $\alpha = \beta = 1/5$ と固定して考える。μ を 3, 4, 5 としていくと，この常微分方程式の解の周期が増加し，その運動は Chaos になることが知られている [6]。

Rössler Model に限らず，非線型の離散・連続力学系における Chaos 現象の視覚化には，数値解法が不可欠である。何故ならば，解析解が導入できない上，解の挙動が複雑で予測することが困難であるからである。それ故に，数値解の精度や，数値的性質には十分に配慮しなければならない。極めて初等的でありながら，これは欠かせない作業である。

本講演では，Rössler Model の数値的挙動について，現時点で把握できている事柄を述べる。

2 数値解の挙動

初期値を $[1\ 0\ 0]^T$ とし，積分区間を $[0, 500]$ に設定して，6 次の陽的 Runge-Kutta，陰的 Runge-Kutta 法を使用して (1) を計算した。このとき，刻み幅 h を $h = 1/8192$ とした時の結果を示す。Fig.1 が $\mu = 4$ の時の図，Fig.2 が $\mu = 5$ の時の図である。

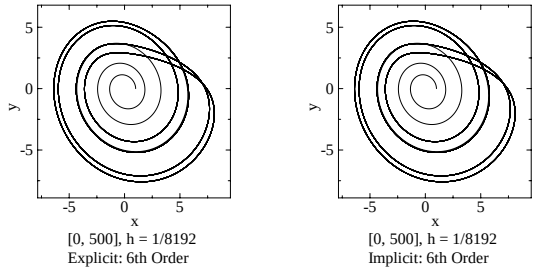

Figure 1: Explicit and Implicit: 6th Order

*静岡理工科大学

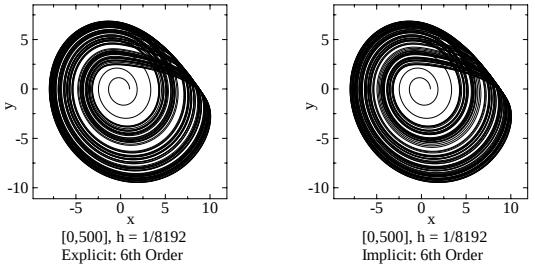

Figure 2: Explicit and Implicit: 6th Order

前述の通り，Rössler Model は μ が 3, 4, 5,... となるにつれて軌道周期が増加し，Chaos になる。数値解もそれに歩調を合わせて，同じ刻み幅であっても次第に精度が悪くなる。実際，陽的・陰的 Runge-Kutta 法のどちらも同じ軌道を再現できていない。

有限桁の浮動小数点数を使用した数値計算では，数値解の誤差は打ち切り誤差（理論誤差）と丸め誤差のトータルで考える必要がある。以下，そのことについて考察する。

3 常微分方程式の初期値問題の数値解

真の解が不明である常微分方程式の初期値問題に対して，どこまでが丸め誤差でどこまでが打切り誤差であるかを調べるには，次のようにして考えなければならない。

1. 積分区間を固定し，刻み幅を小さくしていき (例えば，$1/2, 1/2^2, 1/2^3$... とする)，それぞれ数値解を求める。

2. 刻み幅が小さくなるにつれて，数値解の上の桁から数値が一致していく時，これを真の解として採用する。

3. 刻み幅に合わせて，一致する桁が増えてきているときには，不一致桁の上位が打切り誤差を現している。

4. ある程度を超えると，刻み幅を小さくしても一致する桁数が増えないか，一致する桁数が減少し出す。このとき，不一致桁の上位が丸め誤差を表わしている。

以上は数値計算の常識である。今回のケースでは一致している桁を得ることすら出来ていないため，このままでは丸め誤差と打ち切り誤差の切り分けが不可能である。よってまず切り分けすることを考える。

4 数値実験

丸め誤差の測定は厳密さを問わないのであれば比較的簡単に行える [3]。そこでまず丸め誤差について調べてみる。もし丸め誤差が増大してきているようであれば，多倍長計算を行って丸め誤差を末尾に追いやり，次に前述の数値計算の常道に従って打ち切り誤差の order を見ていく。

4.1 IEEE754 浮動小数点計算

まず，$t = 500$ の時の x の数値解を，7 段 6 次の陽的 Runge-Kutta 法と 3 段 6 次陰的 Runge-Kutta 法を使用して (1) を IEEE754 倍精度浮動小数点数を使用して計算した。その結果を以下に示す。

Explicit : 7 step (6th order), $\mu = 5$

	$h = 1/4096$	$h = 1/8192$
RN	$-4.8295189\mathrm{e}+00$	$-4.6293495\mathrm{e}+00$
RZ	$-6.0714505\mathrm{e}+00$	$-5.2090559\mathrm{e}+00$
RP	$-4.2363258\mathrm{e}+00$	$-3.5081142\mathrm{e}+00$
RM	$-5.1634064\mathrm{e}+00$	$-3.8688926\mathrm{e}+00$

Implicit : 3 step (6th order), $\mu = 5$

	$h = 1/4096$	$h = 1/8192$
RN	$-3.8470654\mathrm{e}+00$	$-3.5698751\mathrm{e}+00$
RZ	$-6.1574318\mathrm{e}+00$	$-5.2671008\mathrm{e}+00$
RP	$-3.4452750\mathrm{e}+00$	$-4.6235796\mathrm{e}+00$
RM	$-3.4044998\mathrm{e}+00$	$-3.5246481\mathrm{e}+00$

上の表は $\mu = 5$ のときの y_1 の値を，RN, RM, RP, RZ モードでそれぞれ計算したものある。浮動小数点演算過程での丸めのモードを変更することによって，丸め誤差の大きさを精度を増やすことなく示すことができる [3]。全ての桁が異なっており，明らかに丸め誤差が数値解を覆い尽くしていることが分かる。

また刻み幅を縮小させても，丸め誤差の order には殆ど変化が無いことも同時に分かる。実際，もっと小さい刻み幅で計算された数値解でも，order には明らかな変化は見られなかった。従って，ここで現れた丸め誤差は，単なる計算量に比例する蓄積によるものでは無く，元の常微分方程式 (1) の性質に依存するものである，と言える。このように，この積分区間 $[0, 500]$ における Rössler Model の数値計算においては，IEEE754 浮動小数点数倍精度の桁数では不足である。Chaos になると言われている $\mu = 5.7$ の場合も同様に，丸め誤差の影響が大きく，精度が必要であれば多倍長計算を行う必要がある。

4.2 多倍長計算ライブラリの選択

本稿で示される数値実験は全て，次の計算機環境下で行ったものである。

1. 多倍長計算ライブラリ: gmp 2.0.2
2. 本体: (CPU)AMD K6-2 300MHz, (RAM)64MB, (OS)Laser5 Linux 6.0
3. コンパイラ: gcc egcs-2.91.66 19990314/Linux (egcs-1.1.2 release)2.7

多倍長計算ライブラリに gmp[2] を選んだ理由は，桁数を自在に伸縮できることと，他のライブラリと比較して四則演算の性能が良かったことにある。以下にその結果を示す。なお，gmp は 2 進の桁数 (bit 数) を指定する。参考までに，IEEE754 浮動小数点数の結果も載せておく。表の数字は全て MFLOPS である。

ベンチマークテストには自作の BASE Bench プログラムを使用した。これは乱数を発生させて二つの配列に格納し四則演算を行い，もう一つの配列に演算結果を代入するという過程を繰り返し，四則演算の MFLOPS 値を計算するという極めて単純なプログラムである。メモリキャッシュの効果などについては特段考慮していない。

	IEEE		**gmp**				
bits	24	53	64	128	256	512	1024
ADD	5.12	5.46	1.60	1.44	1.30	1.00	0.69
SUB	4.56	4.82	1.82	1.64	1.32	0.93	0.59
MUL	5.12	5.46	0.72	0.46	0.21	0.08	0.02
DIV	2.28	2.28	0.36	0.24	0.12	0.05	0.02

gmp は C プログラムのパッケージであり，version 3.0.1 が出た現在でも初等関数も殆ど用意されていない。そのためかなり使い勝手の悪いものではあるが，反面，全ての四則演算はアセンブラライクなスタイルを遵守しなければならないため，うまくコーディングすればかなり最適化したプログラムを作ることができる。

4.3 gmp を使用した多倍長計算

IEEE 浮動小数点数を用いた計算では，丸め誤差が全ての桁を覆い尽くしてしまい，その刻み幅で打ち切り誤差が十分に小さくなっているかどうかは判然としない。よって，多倍長計算で丸め誤差を追いやった後は，打ち切り誤差について考慮しなくてはならない。

$\mu = 5.7$ とし，先程と同じ 7 段 6 次陽的 Runge-Kutta 法を使用して，$t = 500$ での $[x\ y\ z]^T$ の値を計算した結果を以下の表に示す。下線部は丸め誤差と推定される部分を示す。

Explicit Runge-Kutta(7 steps, 6th order), 1/4096

	bits	
x	64	0.42002162655255609 36e1
	128	0.37797640861961057138 5290835153 61628517e1
	256	0.37797640861961057136 7320738859 43094458944368613029228307889161 870091831326861e1
	512	0.37797640861961057136 7320738859 43094458944368613029228307883875 82336351152028201398447393794661 53733881715630110980996720453677 50871589300081703851108862 09e1

	bits	
y	64	0.42060695656418666 05e1
	128	0.38428172611464481745 3441735228 07126894e1
	256	0.38428172611464481743 9737345962 27788033050490108557463268231946 733832416882709e1
	512	0.38428172611464481743 9737345962 27788033050490108557463268227915 47874863034813779978274762266601 92661660747701918571439176927429 49993796360462256269452241 33e1

	bits	
z	64	0.88810894100767647 88e1
	128	0.10439089987024904918 2852521796 15762755e2
	256	0.10439089987024904918 8155037197 04162295035368654361370659942702 246460836084238e2
	512	0.10439089987024904918 8155037197 04162295035368654361370659944262 02353313690460282662773677378369 07818894538423727642416612911396 96131511337627617705928526 07e2

Explicit Runge-Kutta(7 steps, 6th order), 1/8192

	bits	
x	64	0.89709208814224654 92e0
	128	0.37797640552000112697 6329829788 31388409e1
	256	0.37797640552000112694 0489649563 25782522446700452040632311437858 062608729623888e1
	512	0.37797640552000112694 0489649563 25782522446700452040632311427345 52931089301283674221682661851212 13624691869440035120827640115549 07468266990926456079363984 73e1

	bits	
y	64	0.19161317029869553 54e1
	128	0.38428172375081472738 5024559808 15434243e1
	256	0.38428172375081472735 7692052970 53884245646566454781243764789038 175785599114272e1
	512	0.38428172375081472735 7692052970 53884245646566454781243764781021 08752210586541297215637430589581 62055401694085866144932911781715 84447506200206924074157066 1e1

	bits	
z	64	0.12282320611561214 29e2
	128	0.10439090078486444085 3094962449 74233149e2
	256	0.10439090078486444086 3670481963 06277723442213310665437088642699 707110051298398e2
	512	0.10439090078486444086 3670481963 06277723442213310665437088645801 68651773236312522813427674896625 74642700185083490165294408946976 72961239725272697220372327 69e2

上の表から，次のことが分かる。

1. 丸め誤差は 10 進 18〜19 桁程度の大きさである。故に，64bit 計算の結果は全く信用できない。
2. 128bit 以上の計算結果のうち，同じ計算桁数で刻み幅を小さくしていった結果の不一致桁の上位は打ち切り誤差を示している。刻み幅を 1/2 にすると 10 進 2 桁ずつ減少していく。

従って，128bit 以上の計算結果のうち刻み幅が 1/8192 のものも，打ち切り誤差は更に縮小させることができると予想される。しかし，陽的 Runge-Kutta 法で更に次数を上げるためにはより多くの段数を必要とし，段数と次数の差は拡大していく。そのため，多倍長計算においてはあまり効率の良い方法ではない。アルゴリズムが単純で，可変次数の公式が望まれる。

そのために補外法を使用する [5]。以下の計算は 128bit の浮動小数点数を使用して行われたものである。補外表の大きさは 4 段とした。この補外には Romberg 数列を使用しているため，4 段では 8 次程度の公式に相当する [5]。この表では，下線部が一致している桁を示す。

Extrapolation 4 Stages, 128bit

	Stepsize	
x	1/4096	0.37797640547076089 4482920610 00266069882e1
	1/8192	0.37797640547076090 5493547015 61087091259e1
y	1/4096	0.38428172371326304 7040226732 50611715412e1
	1/8192	0.38428172371326305 5437171714 79686423255e1
z	1/4096	0.10439090079939397 2686850428 212526679e2
	1/8192	0.10439090079939396 9437896498 84268313596e2

以上の結果より，一致している桁は上から 16〜17 桁である。全 38 桁のうち，下位桁 18〜19 桁は丸め誤差であることが示されており，桁の一致している上位桁は正確であると考えて良い。従って，ほぼ IEEE754 倍精度程度の精度を得ていると言える。参考までに z の相対丸め誤差と $\mu = 5.7$ の時の z を描いたグラフを Fig.3 に示す。μ の値によって丸め誤差の変動が著しく異なっていることが分かる。$\mu = 5, 5.7$ の時と $\mu = 4$ の時とを比較してみると，単なる丸め誤差の蓄積以上に増大が激しく，微少な桁落ちが随所で起こり，それが丸め誤差の増大を引き起こしていることを予想させる。

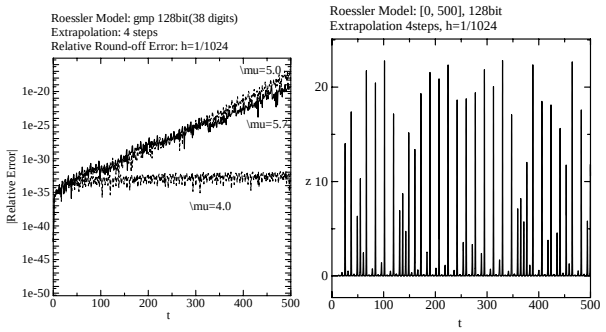

Figure 3: Relative errors of z and z at $\mu = 5.7$

最後に計算時間を示す。同次数での比較のため，3 段の補外法の計算時間も載せておく。

Computation Time (sec)

	Runge-Kutta	Extrapolation	
Stepsize: 1/1024			
	7-6	3 stage	4 stage
gmp 64 bits	146.69	167.08	364.3
gmp 128 bits	194.53	204.82	444.4
gmp 256 bits	328.71	308.55	664.8
Stepsize: 1/2048			
gmp 64 bits	293.32	334.13	727.1
gmp 128 bits	388.35	409.62	888.8
gmp 256 bits	657.34	617.09	1330.0

4 段の補外法は 7 段 6 次の陽的 Runge-Kutta 法の倍近い時間が必要であるが，同次数の 3 段の場合はほぼ同時間で済む。

5 まとめ

Røssler Model の数値的性質として，離散化スキームとして陽的・陰的 Runge-Kutta 法を使用した場合，次のことが判明した。

1. μ の増加に伴い，収束性が悪くなる
2. μ の増加に伴い，計算途上での丸め誤差による影響が大きくなる
3. 今回用いた倍精度計算では，$\mu = 5$ とした場合，$t = 500$ あたりが精度限界である。

今後の課題としては以下のものが挙げられる。

- 丸め誤差の影響の原因を，より初等的な観点で追及する。恐らくは，Lorenz Model についても同様のことが言えるのではないかと予想されるので，そこでも同様の考察を行う。
- 本研究で使用した多倍長計算用 ODE solver をライブラリとして整備する。

謝辞

本研究を進めるにあたり，面倒な議論におつき合いいただいた永坂先生に感謝いたします。

参考文献

[1] 幸谷智紀, 常微分方程式の初期値問題における多倍長計算の必要性についての一考察, 研究集会「工学における微分方程式の数値解析」口頭発表, 2000.

[2] T.Grandlund, TMG Datakonsult, GNU MP, The GNU Multiple Precision Arithmetic Library, Manuscript, 1996.

[3] 幸谷智紀・永坂秀子, IEEE754 規格を用いた丸め誤差測定法について，日本応用数理学会論文誌, Vol.7, No.1, 1997.

[4] 幸谷智紀，常微分方程式および固有値問題の高精度計算法の研究，博士論文 (日本大学), 1997.

[5] 室伏誠, 有限桁計算における Richardson の補外法による丸め誤差評価の研究, 博士論文 (日本大学), 1998.

[6] 下條隆嗣, カオス力学入門, 近代科学社, 1992.