

J の簡易反復法による関数のプログラミング

SHIMURA Masato
JCD02773@nifty.ne.jp

2009 年 7 月 30 日

目次

1	線形反復 (マトリクス)	2
1.1	サブレ	2
1.1.1	フィボナッチ数	3
1.1.2	シュメール人の $\sqrt{2}$	4
1.2	折り込みパイ	5
1.2.1	QR 法で固有値を求める	6
1.3	串団子	8
1.3.1	A^n を求める	8
1.3.2	ケーリー・ハミルトン	8
1.4	ビスケット	11
1.4.1	レオンティエフ逆行列と連鎖	11
2	非線形反復 (関数系)	14
2.1	クレム・キャラメル	14
2.1.1	ニュートン法	14
2.1.2	連立方程式とニュートン法	15
2.1.3	Broyden 法	17
2.1.4	ニュートン最適化	21
2.2	バウムクーヘン	22
2.2.1	オイラー法	23
2.2.2	ルンゲ・クッター法	25
3	非線形の行列風演算	28
3.1	練り込みパイ	28

3.1.1	エノンのカオス	28
4	非線形反復 (差分系)	32
4.1	ジェラート	32
4.1.1	ロレンツ	32
4.1.2	Rössler attracta	32
4.1.3	ロトカ・ヴォルテラ	34
5	確率推移行列の反復計算	38
5.1	びろしき	38
5.1.1	マルコフ連鎖	38
5.1.2	レスリー行列	39
6	References	45

初めに

J の Tacit (関数型定義) の接続詞に反復 (*power conjunction*($\hat{:}n$)) があり、関数と反復手続きを分けることができる。この分離により簡潔なスクリプトが作成でき、マトリクスや関数の構造がより理解しやすくなる。

さらに非線形へ拡張したり、関数風行列も作成できる。*Art of Computing Science* への一歩である。

ハンドミキサーのようにループを廻してマトリクスや関数を実際に計算してみよう。永年作り続けられてきた各種のケーキのレシピは辛抱強い泡立てが基本である。

A 先生 スクリプトと計算は M さんに一任

M さん 美しいスクリプトを目指す

N さん *Novice* だが質問は鋭い

1 線形反復 (マトリクス)

1.1 サブレ

x		y
マトリクス	内積	初期値

A シンプルな演算用マトリクスと内積を用いて同じパターンで打ち出す。初期値は最初の 1 回のみ用いる。

1.1.1 フィボナッチ数

A フィボナッチ (Leonardo of Pisa) は 13C のイタリアはピサの人。父はピサの商人で今のアルジェリアの領事を務め、レオナルドはアラビアで教育を受け、後にエジプト、シリア、ギリシャ、プロバンスなどへ行って様々な数学を身につけた。1202 に著書 (Liber abaci) で最新のインド、アラビア数学を未開のヨーロッパに伝えた。この本は評判でコピーや模造が多く出た。

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{if } n > 1 \end{cases} \quad \text{Matrix form:}$$

$$\begin{bmatrix} F_{k+1} \\ F_{k+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} F_k \\ F_{k+1} \end{bmatrix}$$

M スクリプトは見事なまでに簡潔

```
0 1
1 1
```

```
(0 1, :1 1)&(+/ . *) ^:(i.16) 1 1
```

N 条件式はどこに行ったか

M マトリクスに巧妙に取り込まれている。マトリクスの上の行が条件式の最初の 2 行を、下の行が条件式の最終行を表現している。条件式をマトリクスで極めて簡潔に表現することができる。

N ^:(16) と ^:(i.16) とは何が違うか

M ^:(16) はループが終わった最終解が、 ^:(i.16) の方は経過の一覧が打出される。

Worked Example

M マトリクスも初期値も決まっている紋切り方だ。

A マトリクスが生成する二つの値を割ると黄金比に近づく。また、マトリクス $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ の固有値にも黄金比が出る。そして $k \rightarrow \infty$, のとき $\frac{f_{k+1}}{f_k}$ は黄金比に近づく。

*1

*1 黄金比はオームの法則のオームが 1835 年に唱えたもので、古代ギリシャやローマでは用いられていない。パイオリニストのクライスラーは自作の小品を一昔前の架空の作曲家の曲としていた。

a=(0 1,:1 1)&(+/.*)^(i.16)1 1

a,./"1|.(^1)a

			34	55	1.61765
			55	89	1.61818
1	1	1	89	144	1.61798
1	2	2	144	233	1.61806
2	3	1.5	233	377	1.61803
3	5	1.66667	377	610	1.61804
5	8	1.6	610	987	1.61803
8	13	1.625	987	1597	1.61803
13	21	1.61538			
21	34	1.61905			

char_lf 0 1,: 1 1

++++-----+-----+

|1|1.61803 _0.618034|_1 _1 1|

++++-----+-----+

(固 有 値) f=_1_x+x^2

1.1.2 シュメール人の $\sqrt{2}$

A 本項はシャトランによる。

紀元前 2000 - 3000 年にシュメール人が用いていた。スミルナの *Théon* が 2 世紀に再発見したという。

$$\begin{cases} x \leftarrow x + 2y \\ y \leftarrow x + y \end{cases}$$

反復により $\frac{x^2}{y^2}$ が 2 に近づく

$$\begin{cases} u_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ u_k = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix} u_{k-1}, \quad k \geq 1, \quad \text{ここで} \quad u_k = \begin{bmatrix} x^k \\ y^k \end{bmatrix} \end{cases}$$

Worked Example

M マトリクスを $\begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}$ とし、内積演算を反復する。初期値 1,1 は最初のみ用いる。

(1 2,:1 1)&(+/.*)^(i.10) 1 1

M 2列の数の商が $\sqrt{2}$ に近づく。

```
(%/"1 a),.~ a=(1 2,:1 1)&(+/ . *)^(i.10) 1 1
  1      1      1
  3      2      1.5
  7      5      1.4
 17     12 1.41667
 41     29 1.41379
 99     70 1.41429
239    169 1.4142
577    408 1.41422
1393   985 1.41421
3363  2378 1.41421
```

M マトリクスの 2 を他の数を変更すれば他の平方根を求められる

```
(%/"1 a),.~ a=(1 3 ,:1 1)&(+/ . *)^(i.10) 1 1
  1      1      1
  4      2      2
 10     6 1.66667
 28    16      1.75
 76    44 1.72727
208   120 1.73333
568   328 1.73171
1552  896 1.73214
4240 2448 1.73203
11584 6688 1.73206
```

1.2 折り込みパイ

x		y
⊗	内積 QR 分解	マトリクス

A 折り込みパイ型は計算の軌跡は必要としないタイプ。

1.2.1 QR 法で固有値を求める

$A^0 \rightarrow QR$ 分解 $\rightarrow (R$ 内積計算 $Q) \rightarrow A^1 \rightarrow QR$ 分解 $\rightarrow (R$ 内積計算 $Q) \rightarrow A^2 \rightarrow QR$ 分解 $\rightarrow \dots A^n$

A J のフォーラムで話題になったときに参加したメモリアルが <http://www.jsoftware.com/> の ESSAY/Eigenvalues にある。R.Hui は K.E.Iverson と J 言語を開発した人。ここまでスマートになるとは思いもつかなかた。

NB. Iterative QR algorithm

```
rheval=:+/ .*>/@|.@(128!:0) NB. Roger Hui  
rh=: (<0 1) |: rheval^:_
```

```
mseval=:(>&(1&{) +/ .* >&(0&{))&(128!:0) NB. M. Shimura  
ms=: (<0 1) |: mseval^:_
```

N 128!:0 とは何か?

M J はシステム関数を外部接続詞と読んで区別している。m!:n を用いる。ここにテスト中の関数が載ることもある。128!:0 は QR 分解の関数で永い間載っている。早く正規関数にして欲しい。

Worked Example

A 調理実習ですよ。M さん

M 128!:0 で QR 分解を行って、結果を Q R と左右に分離して内積計算を行う。これを収束するまで繰り返すところが折り込みパイ捏ねのようだ。

```
EX0=:4 3 0, 3 1 _1.: 0 _1 3
```

```
4 3 0  
3 1 _1  
0 _1 3
```

```
128!:0 EX0
```

```
+-----+-----+  
|0.8 0.424264 0.424264|5      3      _0.6|  
|0.6 _0.565685 _0.565685|0 1.41421 _1.55563|
```

```
| 0 0.707107 0.707107|0      0 2.68701|
+-----+-----+
      Q                      R
```

N ところで&とは

M 数値を動詞（関数）と繋ぐ時には&を用いる。動詞と動詞の時は&と@がある。単項では同じ働きだが、両項では作用が異なる。

N QR の内積計算で元に戻ってしまいますよ

```
(>{. a) +/ . * >{: a=.128!;0 EX0 NB. Q +/ . * R
```

```
4 3 0
3 1 _1
0 _1 3
```

M RQ としなければならない。

```
(>{: a) +/ . * >{. a NB. R +/ . * Q
```

```
5.8 0.848528 _1.83187e_15
0.848528      0.3      _1.9
      0      _1.9      1.9
```

M この EX0 は収束が悪い。ms EX0 では当分解が帰ってこない。^:_を用いるのは勇気が要る。^:100 などでテストした方がよい。

対角に固有値が並び、他が 0 ならば収束している。clean は numeric.ijs に入っている微少塵取り関数

```
clean mseval^:(100) EX0
5.95126      0      0
      0 3.08397      0
      0      0 _1.03523
```

N ところで QR 分解はグラム・シュミットの直交化を用いるようですが

M Q は A をグラム・シュミットの直交化したもので、 $R = Q^t A$ となる。

```
clean (|: >{. a) +/ . * EX0
5      3      _0.6
0 1.41421 _1.55563
0      0 2.68701
```

1.3 串団子

x		y
マトリクス	内積	マトリクス

N 団子 3 兄弟のモデルは築地の場外の団子屋ですが。

1.3.1 A^n を求める

M データとして受けたマトリクスの内積計算で $A^2 \cdot A^3 \cdots A^n$ を求める最も単純な計算。まあ、串に刺した団子でも思い浮かべて。

$$A(\text{内積})A(\text{内積})A \cdots (\text{内積})A$$

Worked Example

M データマトリクスを反復用マトリクスとしても用いるため、両項タイプとし、内積計算のみを用いる。

$$EX2 = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 2 \end{bmatrix}$$

<"2 EX2 (+/ . *) ^:(i.3) EX2

```

+-----+-----+-----+
| 2 _1 1| 6 _5 3|20 _19 7|
|_1 2 1|_3 4 3|_7 8 7|
| 1 _1 2| 5 _5 4|19 _19 8|
+-----+-----+-----+
A      A^2      A^3

```

N <"2 とは

M ランク 2("&2) でボックス化する。マトリクス毎にボックスに入る

1.3.2 ケーリー・ハミルトン

A A^n に A の固有値 λ を掛けるとケーリー・ハミルトンの定理が計算できるマトリクスの特性方程式 $\phi(\lambda)$ の λ に A を代入すると $\phi(A) = 0_n$ となる。

(ケーリー・ハミルトンの定理)

$$I + \lambda_1 A + \lambda_2 A^2 + \lambda_3 A^3 = 0_3$$

Worked Example

M 固有値はルベリエ・ファディエフ法で求める。

```

ch_sub1 EX2
      I      A      A^2      A^3
+-----+-----+-----+-----+
| 1 0 0 | 2 _1 1 | 6 _5 3 | 20 _19 7|
| 0 1 0 | _1 2 1 | _3 4 3 | _7 8 7| NB. A^n
| 0 0 1 | 1 _1 2 | 5 _5 4 | 19 _19 8|
+-----+-----+-----+-----+
|_6      | 11      | _6      | 1      | NB. Lamda
+-----+-----+-----+-----+
|_6 0 0| 22 _11 11|_36 30 _18|20 _19 7|
| 0 _6 0|_11 22 11| 18 _24 _18|_7 8 7| NB. A^n * Lamda
| 0 0 _6| 11 _11 22|_30 30 _24|19 _19 8|
+-----+-----+-----+-----+

```

```

cayley_hamilton EX2
+-----+-----+-----+-----+-----+
|_6 0 0| 22 _11 11|_36 30 _18|20 _19 7|=|0 0 0|
| 0 _6 0|_11 22 11| 18 _24 _18|_7 8 7| |0 0 0|
| 0 0 _6| 11 _11 22|_30 30 _24|19 _19 8| |0 0 0|
+-----+-----+-----+-----+-----+

```

$$I + \lambda_1 A + \lambda_2 A^2 + \lambda_3 A^3 = 0_3$$

M ここで box の間に内積演算 (+/ . *) を挿入するには一度ボックスを開いて原型に戻した方が計算しやすい

A 特性方程式は次のようにも表すことができる。この表現はルベリエ・ファディエフ法で用いられている。

$$\det A - (\text{trace} A)t + t^2 = 0$$

成分では

$$(ad - bc) - (a + d)t + t^2 = 0$$

ケーリー・ハミルトンの等式では

$$(ad - bc)I - (a + d)A + A^2 = \det(A)I - \text{trace}(A)A + A^2 = 0$$

Script

```
%power_matrix=: 4 : 'x&mp y' NB. A^n
%NB. Usage: A power_matrix ^:(i.3) A
%
NB. -Cayley_Hamilton Inverse Matrix-----
ch_sub0=: 3 : '};{:char_lf y ' NB. find lamda
NB. mk A^n
ch_sub1=: 3 : 0
TMP0=.(<=/~i.# y), <"2 y power_matrix ^:(i. # y) y
TMP0,TMP1,: TMP0 * L:0 TMP1=. {@> ch_sub0 y
)
cayley_hamilton=:3 : 0
TMP=. ch_sub1 y
({: TMP),(<'='),< +/ > {: TMP
)
```

1.4 ビスケット

x		y
投入産出マトリクス	内積	投入産出マトリクス

1.4.1 レオンティエフ逆行列と連鎖

A レオンティエフ逆行列はハーバードの真空管時代のコンピュータ MARKII で開発され、プログラムは *K.E.Iverson* がサポートした。(1949)

500 本の方程式を 42 本に縮約して、 42×42 の逆行列を用いる方程式を 52 時間で解いた。コンピュータを用いた大規模経済モデルの嚆矢である。

レオンティエフはペテルブルグ生まれの経済学者。ロシア革命後の 1925 年出国を許され、ドイツ滞在後 1931 年アメリカに渡った。産業連関表の研究で 1973 年ノーベル経済学賞受賞。

	<i>demand</i>	<i>Inputs</i>
<i>Final demand</i>	d	Cd
(<i>Intermediate demand</i>)		
1st round	Cd	$C(Cd) = C^2d$
2nd round	C^2d	$C(C^2d) = C^3d$
3rd round	C^3d	$C(C^3d) = C^4d$
	\vdots	\vdots

$$x = d + Cd + C^2d + C^3d + \dots = (I + C + C^2 + C^3 + \dots)d$$

$$(I - C)(I + C + C^2 + C^3 + \dots + C^m) = I - C^{m+1}$$

$$(I - C)^{-1} \approx I + C + C^2 + C^3 + \dots + C^m, (C < 1)$$

A この $(I - C)^{-1}$ が Leontief 逆行列で、 $(I - C)^{-1} \approx I + C + C^2 + C^3 + \dots + C^m$ と生産の波及効果を示している。

Worked Example

<i>Purchased from</i>	<i>Manufac turing</i>	<i>Agriculture</i>	<i>Service</i>
<i>Manufacturing.</i>	0.5	0.4	0.2
<i>Agriculture</i>	0.2	0.3	0.1
<i>Service</i>	0.1	0.1	0.3

consumption matrix

C0		
0.5	0.4	0.2
0.2	0.3	0.1
0.1	0.1	0.3

M $C + C^2 + C^3 + \dots + C^m$ は
 +/> C0&(+/.*)^(i.24) C0
 で計算できる。24 反復で \approx らしくなる。

A レオンティエフ逆行列のパワーは偉大だ。

M

%. (=/~i.3)-C0			
(I - C) ⁻¹	2.96296	1.85185	1.11111
	0.925926	2.03704	0.555556
	0.555556	0.555556	1.66667

(=/~i.3)+ +/> C0&(+/.*)^(i.24) C0			
$I + C + C^2 + C^3 + \dots + C^m$	2.95994	1.84882	1.10916
	0.924412	2.03552	0.554578
	0.554578	0.554578	1.66604

N (= i.3)/は何をしているのか
 3 × 3 の単位行列 (I) を生成している

M 最終需要を工業 50, 農業 30, サービス 20 ユニットと与えた場合の各部門の必要産出量がレ
 オンチエフ逆行列を用いて求められる。

(%. (=/~i.3)-C0) +/. * 50 30 20			
$x = (I - C)^{-1}d$	225.926	118.519	77.7778
final demand(m. a. s.)			
= 50 30 20			

N これは拡大係数行列を用いて逆行列解法 (クラメル法) でも解ける

```
cr=: %."1
```

拡大係数行列

```
((=/~i.3)- C0),. 50 30 20
```

```
0.5 _0.4 _0.2 50
```

```
_0.2 0.7 _0.1 30
```

```
_0.1 _0.1 0.7 20
```

```
clean cr((=/~i.3)- C0),.50 30 20
```

```
1 0 0 225.926
```

```
0 1 0 118.519
```

```
0 0 1 77.7778
```

途中経過 (5 回)

```
,. <"2 C0&(+/ . *)^(i.5) C0
```

```
+-----+
```

```
|0.5 0.4 0.2 |
```

```
|0.2 0.3 0.1 |
```

```
|0.1 0.1 0.3 |
```

```
+-----+
```

```
|0.35 0.34 0.2 |
```

```
|0.17 0.18 0.1 |
```

```
| 0.1 0.1 0.12 |
```

```
+-----+
```

```
|0.263 0.262 0.164 |
```

```
|0.131 0.132 0.082 |
```

```
|0.082 0.082 0.066 |
```

```
+-----+
```

```
|0.2003 0.2002 0.128 |
```

```
|0.1001 0.1002 0.064 |
```

```
| 0.064 0.064 0.0444 |
```

```
+-----+
```

```
|0.15299 0.15298 0.09848|
```

```
|0.07649 0.0765 0.04924|
```

```
|0.04924 0.04924 0.03252|
```

```
+-----+
```

2 非線形反復 (関数系)

2.1 クレム・キャラメル

x		y
関数	微分	初期値(ベクトル)

M 底のキャラメルを X 軸と見よう。

2.1.1 ニュートン法

A ニュートン法またはニュートン・ラフソン法と呼ばれる非線形方程式の解法がある。

ニュートンは唯一つ次の式の解法のメモを残した。

$$y^3 - 2y - 5 = 0$$

ニュートンは第 4 近似で 2.09455147 の解を得ている。ラフソンはニュートンの弟子でニュートン法を今の形に纏め、普及させた。

ニュートン法は、

$$x - \frac{f(x)}{f'(x)}$$

の反復計算を行うものである。

M 次のスクリプトは J の定番である。ニュートン法は差分算子 (D.) を用いてシンプルに定義できる。動詞 (関数) を左パラメーターに取るので副詞 (1:0) で定義する。

ランクはベクトルを引数に取ることができるように ("0) とする。 (^:_) は収束まで計算するようにしているが (^:100) 程度で打ち切っても良い。

NB. Newton method

```
new_1=: 1 : ' ] - x % x D.1' (^:_)("0)
```

Worked Example

M $y = -5 - 2x + x^3$ をニュートン法で解いてみよう。

多項式は p. を用いて定義し初期値はベクトルで与える。

```
_5 _2 0 1&p. new_1 i:3  
2.09455 2.09455 2.09455 2.09455 2.09455 2.09455 2.09455
```

N $y = x^3 - 2x - 5$ が _5 _2 0 1&p. と反対に向いてるが。

- A Jの多項式関数 p. は $y = a_0 + a_1x + a_2x^2 \dots$ と高次の項を右側に持ってくる形を採用している。回帰分析の時も $y = a_0 + a_1x$ と解を打ち出す。
 $y = x^3 - 2x - 5$ と数式通りにしたいのであれば `f=:#.&1 0 _2 _5` と基底を使う方法もある。ニュートン法はどちらも受け入れる。

```
(#.&1 0 _2 _5) new_1 i:3
2.09455 2.09455 2.09455 2.09455 2.09455 2.09455 2.09455
```

- N ニュートン法は初期値に敏感だと言うが?
M グラフを描いてみれば良い。Jのファンクションプロットの機能を使うと X 軸の指定も一度にできる。このJのスク립トは初期値をベクトルで与え、各初期値毎の収束する値を返すので、初期値を広く取れば解が変化するポイントも見つけられる。

```
plot _5 5;' _5 _2 0 1&p.' NB. function plot (nln2 ; '')
```

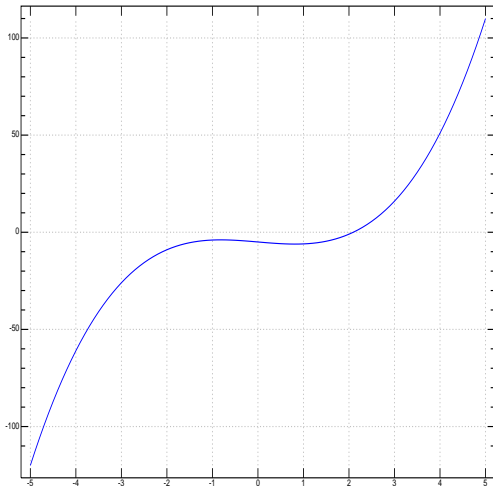


図 1 newton method

2.1.2 連立方程式とニュートン法

- M Jは差分のプリミティブ (D.) を備えている。N.Thomson が 1994 年にこの D. を用いる簡潔な多変数のニュートン法を発表している。

NB. N.Thomson

```
new_2=: 1 : ' ] - x (%. |: ) x D.1' (^:17) ("1)
```

(^{^:17}) は反復回数を指定する任意の整数で、例として 17 が与えられている。

$$2 \text{ 変数 } \begin{cases} f_0 = e^x + xy - 1 \\ g_0 = \sin xy + x + y + 2 \end{cases}$$

Tacit と Explicit の表現。どちらを用いても良い。

Tacit	Explicit
<pre>f0=: -&1@(*/) + ^@{. g0=: +&2@(+/) + 1&o.@*/</pre>	<pre>f0=: 3 : ' (^{.y}) + (* / y) - 1 ' g0=: 3 : ' 2 + (+ / y) + 1 &o. * / y '</pre>

M 連立では (f0,g0) とカンマで動詞を連結して用いる。

```
(f0,g0) new_2 1 1
_9.4112e_6 _2
```

$$3 \text{ 変数の例 } \begin{cases} h_1(x,y,z) = 16x^4 + 16y^4 + z^4 - 16 \\ h_2(x,y,z) = x^2 + y^2 + z^2 - 3 \\ h_3(x,y,z) = x^3 - y \end{cases}$$

```
h1=:3 : ' (16 16 1 +/ . * y ^4) - 16 '
```

```
h2=:3 : ' 3 - +/ y ^2 '
```

```
h3=:3 : ' (1{y}) - ~(( {. y ) ^3) '
```

```
(h1,h2,h3) new_2 1 1 1
0.877966 0.676757 1.33086
```

```
(h1,(h2,h3)) new_2 1 1 1
0.877966 0.676757 1.33086
```

M (h1,(h2,h3)) としても結果は同じである。h1,h2,h3,h4... と続けていけば、多変数に適用できる。

2.1.3 Broyden 法

Newton 法の別の解法として Broyden 法がある。Broyden 法の原典として次のような文献が紹介されている。

C.G.Broyden [Quasi Newton Method and their application to function minimization] 'Mathematics of Computing 19' 1967

ブロイデン法は最初に手で一階と 2 回の偏微分 (ヘッシアン) を行う。ニュートン法に見られる自動差分の脆弱性に陥らない堅牢な方法である。

$$A_k(x^{(k)} - x^{(k-1)}) = F(x^{(k)}) - F(x^{(k-1)})$$

$$f'(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

$$A_k = A_{k-1} - \frac{y - A_{k-1}\Delta}{\Delta^T \Delta} \Delta^T$$

$$y = F(x^k) - F(x^{k-1})$$

$$\Delta = x^k - x^{k-1}$$

$$A_k^{-1} = \left(A_{k-1} \frac{y - A_{k-1}\Delta}{\Delta^T \Delta} \Delta^T \right)^{-1}$$

$$= A_{k-1}^{-1} + \frac{(\Delta - A_{k-1}^{-1}y)\Delta^T A_{k-1}^{-1}}{\Delta^T A_{k-1}^{-1}y} \leftarrow \text{この式を用いる}$$

Worked Example
$$\begin{cases} x_1^3 - 2x_2 - 2 = 0 \\ x_1^3 - 5x_3^2 - 7 = 0 \\ x_2x_3^2 - 1 = 0 \end{cases}$$

$f_1(x_1, x_2, x_3) = x_1^3 - 2x_2 - 2 = 0$ $f_1(x_1, x_2, x_3) = x_1^3 - 5x_3^2 - 7 = 0$ $f_1(x_1, x_2, x_3) = x_2x_3^2 - 1 = 0$	$F(x) = \begin{bmatrix} f_1(x_1, x_2, x_3) \\ f_1(x_1, x_2, x_3) \\ f_1(x_1, x_2, x_3) \end{bmatrix} = \begin{bmatrix} x_1^3 - 2x_2 - 2 \\ x_1^3 - 5x_3^2 - 7 \\ x_2x_3^2 - 1 \end{bmatrix}$
Starting from initial vector $x^{(0)} = [1 \ 1 \ 1]$	初期値 x_0
Hessian Matrix $J(x) = \begin{bmatrix} 3x_1^2 & -2 & 0 \\ 3x_1^2 & 0 & -10x_3 \\ 0 & x_3^2 & 2x_2x_3 \end{bmatrix}$	

M 関数のスクリプトは1本ずつ作成して確認した方がよい。

NB. Bradie EX 3.1.7

```
fbr0=: 3 : '+/(&3 {y),(_2*1{y),_2'
```

```
fbr1=: 3 : '+/(&3 {y),(*&5 (^&2) 2{y),7'
```

```
fbr2=: 3 : '<: */ (1 2 {y)^ 1 2'
```

```
fx=: fbr0,fbr1,fbr2
```

NB. -----

```
jbr0=: 3 : '( 3* ({y)^2),_2,0'
```

```
jbr1=: 3 : '( 3 * ({y)^2),0,_10 * {y'
```

```
jbr2=: 3 : '0,({y)^2),(* / 2 1 * 1 2 {y)'
```

```
jx=: jbr0,jbr1,:jbr2
```

$F(x^{(0)})$	<pre>fx=: fbr0,fbr1,fbr2 fx 1 1 1 _3 3 0</pre>
$A_0 = J(x^{(0)})$	<pre>jx=: jbr0,jbr1,:jbr2 jx 1 1 1 3 _2 0 3 0 _10 0 1 2</pre>
A_0^{-1}	<pre>1 x: %.@jx 1 1 1 5r21 2r21 10r21 _1r7 1r7 5r7 1r14 _1r14 1r7</pre>

$v^{(0)} = -A_0^{-1}F(x^{(0)})$	<pre> 1 x: br_v 1 1 1 3r7 _6r7 3r7 </pre>
$x^{(1)} = x^{(0)} + v^{(0)}$	<pre> 1 x: br_x 1 1 1 10r7 1r7 10r7 </pre>

M 最初の v_0, x_0 を得る

```

br_v=: [: - fx %. jx      NB. same br0
br_x=: ] + br_v         NB. x(n)

```

M プロイデン法のエンジン A_k^{-1}

$$A_k^{-1} = A_{k-1}^{-1} + \frac{(\Delta - A_{k-1}^{-1}y)\Delta^T A_{k-1}^{-1}}{\Delta^T A_{k-1}^{-1}y}$$

Δ は v_{k-1}

```
br_sub=: (%.@jx )+((br_v - br_d1)+/ . * br_d2)% br_d3
```

$F(x^{(1)})$	<pre> fx br_x 1 1 1 0.629738 _0.28863 _0.708455 </pre>
$y = F(x^{(1)}) - F(x^{(0)})$	<pre> (fx@br_x - fx) 1 1 1 3.62974 _3.28863 _0.708455 </pre>
$A_0^{-1}y$	<pre> br_d1 1 1 1 0.213661 _1.49438 0.392961 </pre>

$\Delta^T A_0^{-1}$	br_d2 1 1 1 0.255102 _0.112245 _0.346939
$\Delta = v^{(0)}$ $\Delta^T A_{(0)}y$	br_d3 1 1 1 1.54088
$\frac{A_k^{-1}(\Delta - A_{k-1}^{-1}y)\Delta^T A_{k-1}^{-1}}{\Delta^T A_{k-1}^{-1}y} = A_{k-1}^{-1} +$	br_sub 1 1 1 0.219238 0.0763808 0.457333 _0.161714 0.124 0.695428 0.0525713 _0.0902859 0.124

```

br_y=: fx@br_x - fx      NB. y
br_d1=: br_y %. jx
br_d2=:br_v +/ . * %.@jx NB. not equal(br_v %. jx)
br_d3=: (br_v +/ . * %.@jx) +/ . * br_y
NB. ----broyden Engine-----
br_sub=(%.@jx )+((br_v - br_d1)+/ . * br_d2)% br_d3
NB. broyden main
broyden=: 3 : 0
NB. Usage: broyden 1 1 1
XN=. br_x y      NB. present x(n)
AINV=. br_sub y  NB. inverse An
DELTA=. - AINV +/ . * fx XN NB. add minus
XN + DELTA      NB. new x(n)
)

broyden ^:(i.7) 1 1 1
      1      1      1
1.63655 0.773164 1.45725
1.44611 0.499603 1.41262
1.44225      0.5 1.41421
1.44225      0.5 1.41421
1.44225      0.5 1.41421

```

1.44225 0.5 1.41421

2.1.4 ニュートン最適化

- A 制約なし最適化の解法にニュートン法がある。テーラー展開を利用して目的関数を 2 次近似する。ニュートンの名を弟子が用いたものではないか。

$$f(x+d) \simeq f(x) + \nabla f(x)^T d + \frac{1}{2} d^T H f(x) d$$

最小化関数 $d = -H f(x)^{-1} \nabla f(x)$

Worked Example (田村・村松 EX3.10)

A $f(x) = x_1^2 - 2x_1x_2 + \frac{1}{4}x_2^4 - \frac{1}{3}x_2^3$

のとき 1 階偏微分は

$$\nabla f(x) = \begin{pmatrix} 2x_1 & -2x_2 \\ -2x_1 & x_2^3 - x_2^2 \end{pmatrix}$$

2 階偏微分 (ヘッセ行列) は

$$\nabla H f(x) = \begin{pmatrix} 2 & -2 \\ -2 & 3x_2^2 - 2x_2 \end{pmatrix}$$

である。

$d^k = -H f(x^k)^{-1} \nabla f(x^k)$ を収束するまで反復計算する。

- M 関数のプログラムであるが、関数が $f(x), H f(x)$ と 2 つあり、 u で呼び出すと混乱しそうなので 3 : 0 を用いる。従って $f0, h0$ の名前が固定されてしまう。

1 階微分は $f0$, ヘッセは $h0$ で定義する。

$d^k = -H f(x^k)^{-1} \nabla f(x^k)$ は `newton_ml` に組み込む。

NB. Newton maximum Likelihood

NB. Tamura&Muramatsu EX.3.5

```
na0=: 3 : '+/ 2 _2 &* y'
```

```
na1=: 3 : '+/ (_2&{* . y), _1 1 &* ({:y)^2 3'
```

NB. -----

```
nb0=: 2:
```

```
nb1=: nb2=: -&2:
```

```
nb3=: 3 : '+/_2 3 * ({:y )^1 2'
```

NB. -----

```
f0=: na0;na1
```

```
h0=: (nb0;nb1),.(nb2;nb3)
```

```
newton_ml=: 3 : 0
```

NB. `newton_ml` 3 3

```
y + (- %.;('1) h0 L:0 y) +/ . * ;f0 L:0 y
)
```

N 数値を入れて見てみましょう

$$\nabla f(x^0) = \begin{pmatrix} 0 \\ 12 \end{pmatrix}, \quad Hf(x^0) = \begin{pmatrix} 2 & -2 \\ -2 & 21 \end{pmatrix}$$

```
- %. (2 _2, : _2 21)
_0.552632 _0.0526316
_0.0526316 _0.0526316
```

$$d^0 = -Hf(x^0)^{-1} \nabla f(x^0)$$

```
( - %. (2 _2, : _2 21)) +/ . * 0 12
_0.631579 _0.631579
```

$$x^1 = x^0 + d^0$$

```
3 3 + ( - %. (2 _2, : _2 21)) +/ . * 0 12
2.36842 2.36842
```

M 7回の反復で収束している。

```
newton_ml ^:(i.7) 3 3
3 3
2.36842 2.36842
2.07716 2.07716
2.00452 2.00452
2.00002 2.00002
2 2
2 2
```

2.2 バウムクーヘン

x		y
関数	反復計算関数	初期値(ベクトル)

N 屋外で焼く本式のバウムクーヘンには滅多にお目にかかれないが、焼き跡が年輪のように積み重なるところがおいしそうですね。

2.2.1 オイラー法

西川「Jによる微分方程式のグラフィック・アプローチ（その1）」(JAPLA 2006/10) は数式とおり入力し、正規表現でJの関数に変換し、解のグラフとベクトル場を表示する本格的なものである。この中にオイラー法からルンゲ・クッタ法までのスクリプトが入っている。この4本のScriptを取り出し多少修正した。

Worked Examples .

$$\frac{dt}{dx} = 1 + \frac{x}{t}$$

NB. Bradie EX.7.5

f8=:3 : '>:({:y)%{.y}'

A オイラー法は言われているほど精度は悪くない。

$$\frac{dt}{dx} = 1 + \frac{x}{t}$$

	0.5	f8 euler0 ^:(i.12) 1 1
x(1) = 1	1	1
w ₀ = 1	1.5	2
w _{i+1} = w _i + h(1 + $\frac{w_i}{t_i}$)	2	3.16667
w ₁ = w ₀ + h(1 + $\frac{w_0}{t_0}$) = 1 + 0.5($\frac{1}{1}$) = 2	2.5	4.45833
w ₂ = w ₁ + h(1 + $\frac{w_1}{t_1}$) = 2 + 0.5($\frac{2}{1.5}$) = 3.16...	3	5.85
	3.5	7.325
	4	8.87143
	4.5	10.4804
NB.written by Toshio Nishikawa	5	12.1448
NB.slightly modified by M.Shimura	5.5	13.8593
NB. 0.1 (u=f) rk^:(i.11) 0 0	6	15.6193
euler0 =: 1 : 0	6.5	17.4209
:		
h0 =: x NB. 0.01		
'X0 Y0' =: y		
k0 =: u X0, Y0		
2{.(X0+h0), (Y0 + h0*k0)		
)		

N どうしてオイラーの式が出てくるのかよく分からない

A テーラー展開

$$y(t) = y_i + (t - t_i)y'_i + \frac{1}{2}(t - t_i)^2 y''(\xi)$$

$t = t_{i+1}$ をテーラー展開に当てはめ y_{i+1} を求める式を作成する。

$$y_{i+1} = y_i + hf(t - t_i) + \frac{1}{2}h^2 y''(\xi)$$

オイラー法は誤差を少なくしながら次の計算を繰り返す

$$w_0 = \alpha$$

$$w_{i+1} = w_i + hf(t_i, w_i) \quad i = 0, 1, 2, \dots, N-1.$$

オイラー法は次の微分を行っている

$$y'(ti) = \frac{y_{i+1} - y_i}{h} - \frac{h}{2} y''(\xi)$$

$$\frac{y_{i+1} - y_i}{h} - \frac{h}{2} y''(\xi) = f(t_i, y_i)$$

$$y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(t, y) dt$$

M 修正オイラー法

$$\tilde{w} = w_i + \frac{h}{2} f(t_i, w_i)$$

$$w_{i+1} = w_i + hf(t_i + \frac{h}{2}, \tilde{w})$$

$$\frac{dt}{dx} = 1 + \frac{x}{t}$$

NB. written by T.Nishikawa

eulerx =: 1 : 0

:

h0 =. x

'X0 Y0' =: y

k0 =: u X0, Y0

k1 =: Y0 + (-:h0) * k0

2{. (X0+h0), Y0 + h0* u (X0 + -:h0), k1

)

```
0.5 f8 eulerx ^:(i.12) 1 1
1      1
1.5    2.1
2      3.37143
2.5    4.76984
3      6.26926
3.5    7.8526
4      9.50774
4.5    11.2256
5      12.9992
5.5    14.823
6      16.6922
6.5    18.6033
```


2.2.2 ルンゲ・クッター法

M ドイツ生まれのルンゲ・クッター法。100年以上使い込まれた有名な手法だ。

Classical Fourth Order scheme などと呼ばれる。

N バウムクーヘンの年輪を感じる。

$$w_{i+1} = w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(t_i, w_i) \quad 0.5 \quad f8 \quad rk \quad ^:(i.12) \quad 1 \quad 1$$

$$k_2 = hf(t_i + \frac{h}{2}, w_i + \frac{k_1}{2}) \quad 1 \quad 1$$

$$k_3 = hf(t_i + \frac{h}{2}, w_i + \frac{k_2}{2}) \quad 1.5 \quad 2.10778$$

$$k_4 = hf(t_i + h, w_i + k_3) \quad 2 \quad 3.38563$$

$$2.5 \quad 4.78985$$

$$\frac{dt}{dx} = 1 + \frac{x}{t} \quad 3 \quad 6.29477$$

$$t_0 = w_0 = 1 \quad 3.5 \quad 7.88341$$

$$k_1 = hf(t_0, w_0) = f(1, 1) = 2 \quad 4 \quad 9.54374$$

$$k_2 = hf(t_0 + \frac{h}{2}, w_0 + \frac{k_1}{2}) = f(1.5, 2) = 2.3333 \quad 4.5 \quad 11.2667$$

$$k_3 = hf(t_0 + \frac{h}{2}, w_0 + \frac{k_2}{2}) = f(1.5, 2.1666) = \quad 5 \quad 13.0454$$

$$2.444 \quad 5.5 \quad 14.8741$$

$$k_4 = hf(t_0 + h, w_0 + k_3) = f(2, 3.444) = 2.722 \quad 6 \quad 16.7484$$

$$w_1 = 1 + \frac{1}{6}(2 + 2(2.33) + 2(2.44) + 2.72) = 3.379 \quad 6.5 \quad 18.6644$$

rk =: 1 : 0

NB. written by T.Nishikawa

:

h0 =. x

'X0 Y0' =: y

k1 =: u (X0), (Y0)

k2 =: u (X0 + h0%2), (Y0 + h0*k1%2)

k3 =: u (X0 + h0%2), (Y0 + h0*k2%2)

k4 =: u (X0 + h0), (Y0 + h0*k3)

(X0+h0), (Y0 + h0*(k1 + (2*k2) + (2*k3) + k4)%6)

)

2階微分方程式 2階微分方程式： $\frac{d^2y}{dx^2} + 4\frac{dy}{dx} + 4y = 0$

1階連立微分方程式に書き直す。

$$\begin{cases} \frac{dy}{dx} = z \\ \frac{dz}{dx} = -4(y+z) \end{cases}$$

初期値 $x = 0, y = 1, \frac{dy}{dx} = 0 (= z), h = 0.2$

連立方程式の関数定義。 ; で 2 の関数を繋ぎ L:0 で同時に計算する。

f10=:3 : '({: y); _4* +/ }. y'

(出典 佐藤 中村)

```
0.2 f10 rk_2nd ^:(i.6) 0 1 0 NB. repeat 6 times
x      y      z
-----
0      1      0
0.2 0.938133 _0.535467
0.4 0.808413 _0.717954
0.6 0.662289 _0.721974
0.8 0.524667 _0.645349
1 0.405817 _0.540802
```

2階微分方程式： $\frac{d^2y}{dx^2} + \frac{dy}{dx} + 8y = 0$

1階連立微分方程式に書き直す。

$$\begin{cases} \frac{dy}{dx} = z \\ \frac{dz}{dx} = -8y - z \end{cases}$$

初期値 $x = 0, y = 0.5, \frac{dy}{dx} = 1 (= z), h = 0.1$

```
plot {2{. |: 0.1 f11 rk_2nd ^:(i.50) 0 0.5 1
```

Van der Pole's equation： $\frac{d^2y}{dx^2} + \mu(y^2 - 1)\frac{dy}{dx} + y = 0$

1階連立微分方程式に書き直す。

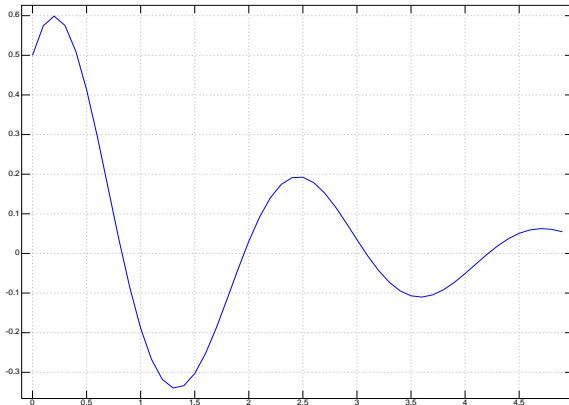


图 2

$$\begin{cases} \frac{dy}{dx} = z \\ \frac{dz}{dx} = \mu(1-y^2)z - y \end{cases}$$

初期值 $x = 0, y = 1, \frac{dy}{dx} = 0 (= z), h = 0.1$

Script .

```
rk_2nd =: 1 : 0
NB. Usage: 0.2 f10 rk_2nd ^:(i.6) 0 1 0
:
hh=. -: h0 =. x
'X0 Y0 Z0' =: y
'K1 L1' =: u L:0 X0,Y0,Z0
'K2 L2' =: u L:0 (X0 + hh), (Y0 + hh*K1), Z0+hh*L1
'K3 L3' =: u L:0 (X0 + hh), (Y0 + hh*K2), Z0+hh*L2
'K4 L4' =: u L:0 (X0 + h0), (Y0 + h0*K3), Z0+h0*L3
YY=. (Y0 + (h0 * +/ K1, (+: K2,K3),K4)%6)
ZZ=. (Z0 + (h0 * +/ L1, (+: L2,L3),L4)%6)
(X0+h0), YY, ZZ
)
```

3 非線形の行列風演算

3.1 練り込みパイ

x		y
パラメータ a, b	疑似非線形関数の 行列風演算	初期値(ベクトル)

3.1.1 エノンのカオス

$$\begin{cases} x_1 = 1 - ax_0^2 + y_0 \\ y_1 = bx_0 \end{cases} \quad \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} -a * sqre & 1 \\ b & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

西川のエノンのスクリプト 2007年夏の蓼科での研究会で披露された西川のエノンのカオスのスクリプト。

- A 非線形を差分関数の(擬)行列に組み上げて線形の振りをしながら反復系で計算してしまうというものかな
- M 非線形の差分化の教科書として練り込みパイを連想しながら眺めた。

```
NB. original Script                                henon ^:(i.10) 1 1
NB. written by Toshio Nishikawa                    1          1
HA=: 1.5                                           0.5          0.25
HB=: 0.25                                          0.875        0.125
dia=: i.@#} NB. extractdiagnal terms              _0.0234375   0.21875
henon=: 3 : 0                                     1.21793 _0.00585938
'x0 y0'=. y                                        _1.23088    0.304482
x1=. +/ dia (1:- (HA"_ **:))' ]' : (0) x0,y0      _0.968099   _0.307719
y1=. +/ dia (HB"_ *])' 0: ' : (0) x0,y0          _0.713541   _0.242025
x1,y1                                              _0.00573661 _0.178385
)
```

- M evoke gerund' : (n) の 0 形を用いている。
- N $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ はどこにあるのかな
- M $1 - a$ で整理されている。厳密な数式でなく手続きを分かりやすく表示していると考えよう。

$$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 - a * sqre(x_0) & 1 \\ b & 0 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

- . は $1 - a$ になる。] は何も演算しないでそのまま (右) 引数の値を返す。^{*2}

西川のスク립トを目を凝らして解読しパラメーターを取り込んで簡約化を探った

```
henon3=: 4 : 0
NB. Usage:(1.5 0.25)& henon3 ^:(i.10) 1 1
'HA HB'=: x
'x0 y0'=: y
(+ / (- . HA * *: x0), y0), HB*x0
)
```

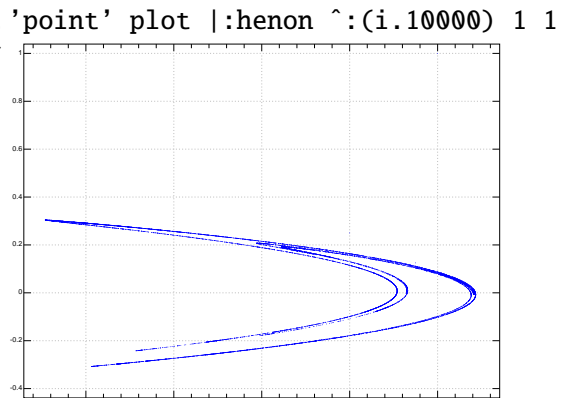


図3 Caos of Henon

A このスク립トは $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ を全部強引に関数で記述してる。0:のタイプの動詞は9:までしかないので12等が入る場合は更に工夫が要る。

M *henon3* は簡略化の過程で行列風の形は崩れている。

非線形・関数行列風の反復計算

A 関数行列風にするにはもう一工夫できるのではないか。ボックスは数字や文字列を格納するだけではない。関数も入れられる。

M 次の Example が参考になる。関数は (t0;t1) のように別枠で定義した方が安全だ。

```
NB. ----example (testpiece)--- (t0;t1) test 1 2 3;4 5 6
t0=: + +-----+-----+
t1=: * |5 7 9|4 10 18|
+-----+-----+

test=: 1 : 0
NB. test 1 2 3;4 5 6 (+;*) test 1 2 3;4 5 6
'A B' =. y +-----+-----+
A u B |5 7 9|4 10 18|
) +-----+-----+
```

M これで *henon* をやってみよう。解が分かっているので安心できる。

^{*2} 1&*@] は通らない

```

NB. henon matrix like
henon5=: 1 : 0
NB. f5 henon5 ^:(i.10) 1 1
'x0 y0'=. y
+/"1 ;("1) tmp=. u x0,y0
)
      1      1
      0.5      0.25
      0.875      0.125
      _0.0234375      0.21875
      1.21793 _0.00585938
      _1.23088      0.304482
      _0.968099 _0.307719
      _0.713541 _0.242025
      _0.00573661 _0.178385
      0.821565 _0.00143415
f5=: (ha5;hb5),:(hc5;hd5)

```

```

NB. 'point' plot |: f5 henon5 ^:(i.10000) 1 1

```

N f5 はマトリクスの形になっていない

```

f5
(ha5 ; hb5) ,: hc5 ; hd5

```

M 関数は見かけは 1 行だが次の計算をして数値マトリクスの結果を返している。厳密に x, y を分離するため Explicit 型で作成した。

$$\frac{ha =: [: -.HA * *: | hb =:]}{hc =: HB* | hd =: 0 :}$$

M 更に左パラメータ (x) まで取り込むと次のようになる

```

ha6=: 4 : ' -. ({.x) * *: {. y'
hb6=: 4 : '{: y'
hc6=: 4 : ' ({:x) * {. y'
hd6=: 0:

f6=: (ha6;hb6),:(hc6;hd6)
henon6=: 1 : '+/"1 ;("1) tmp=. u y'

```

```
'point' plot { |: (1.5 0.25)&f6 henon6 ^:(i.10000) 1 1
```

N tmp=. u y で x を引いていないが

M 4:0 のところで引いている。組み込む個別関数は Explicit の場合は全部を 3:0 か 4:0 のどちらかに揃えなければならない。

4 非線形反復 (差分系)

4.1 ジェラート

x		y
		初期値(ベクトル)

4.1.1 ロレンツ

ロレンツの方程式 (1963) を離散化し、漸加法で構成する山下, 西川によるスクリプトが JAPLA で公表されている。^{*3}

M 10 50 8r3&calc_lorenz plot_lorenz init でロレンツアトラクタを描いてみよう。

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$

σ is Prandtl number

ρ is Rayleigh number

$\sigma, \rho, \beta > 0$

usually $\sigma = 10, \beta = 8r3, \rho = 28$

```
dt=: 0.005 NB. or 0.01
NB.10 50 8r3&calc_lorenz plot_lorenz init

\textbf{init} is 5 8 10 or 1 1 1

lz=:4 : 0
NB. Lorenz attracta
's r b'=. x NB. parameter
'xx yy zz'=: y NB. initial 5 8 10 or 1 1 1
X=: xx + dt*s*(yy-xx)
Y=: yy + dt*((r*xx)-(yy+xx*zz))
Z=: zz + dt*((xx*yy)-b*zz)
X,Y,Z
)
```

10 50 8r3&calc_lorenz plot_lorenz init

4.1.2 Rössler attracta

Otto Rössler(1940- ,Berlin, biochemist)designed simpler Lorenz like attracta in 1976.

A ロレンツアトラクタを単純化したアトラクタはドイツのロスラーが 1976 年に発表した。山下の漸加法のスクリプトをロスラー用に改良した。

^{*3} パラメーターと初期値を分離するため若干改良した。

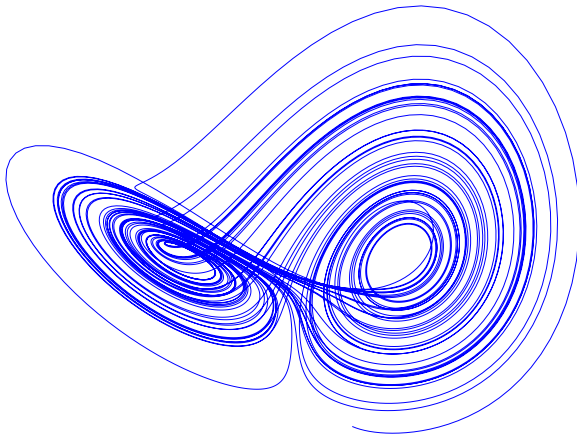


図 4 Lorenz

*4

```
M      0.2 0.2 14&calc_rossler plot_lorenz 1 1 1
```

```
dt=: 0.005
```

$$\begin{aligned} \frac{dx}{dt} &= -y - z \\ \frac{dy}{dt} &= x + ay \\ \frac{dz}{dt} &= b + (x - c)z \end{aligned}$$

```
rsr=:4 : 0
```

```
NB. Rossler attracta
```

```
'a b c'=. x NB. 0.2 0.2 14
```

```
'xx yy zz'=: y
```

```
X=: xx + dt*-(yy+zz)
```

```
Y=: yy + dt* xx + a*yy
```

```
Z=: zz + dt* (b+ (xx*zz)-c*zz)
```

```
X,Y,Z
```

```
)
```

```
parameter a = b = 0.2, c = 5.7 (Rössler)
```

```
a = b = 0.1 c = 14 more common
```

```
0.2 0.2 14&calc_rs plot_lorenz 1 1 1
```

```
pd 'eps /temp/rossler0.eps'
```

Script

```
calc_lorenz=:4 : ' <"1 |: x lz ^:(i.10000) y'
```

```
NB. 10 50 8r3&calc_lorenz plot_lorenz init
```

*4 何種類かあるようだがここに記載したものが有名である。

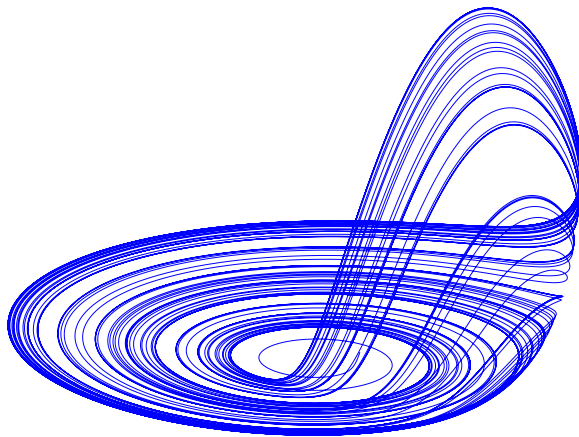


図5 a,b,c is 0.2 0.2 5.7

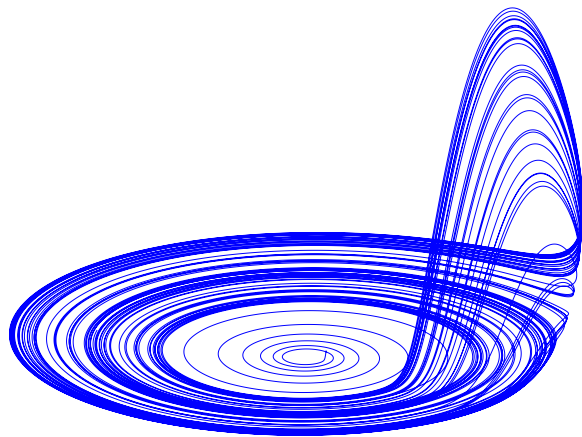


図6 a,b,c is 0.1 0.1 14

```
calc_rossler=:4 : ' <"1 |: x rsr ^:(i.100000) y'
NB. 0.2 0.2 14&calc_rossler plot_lorenz 1 1 1
```

```
plot_lorenz=:1 : 0
NB. 10 50 8r3&calc_lorenz plot_lorenz init
pd 'reset'
pd 'noaxes'
pd 'viewpoint 1.6 _2.4 _1.3'
pd u y
pd 'show'
)
```

4.1.3 ロトカ・ヴォルテラ

Alfred.J.Lotka(1880-1949) アメリカ人の両親の下にオーストリー帝国(現ウクライナ)のレムベルク生まれ。少年時代をフランスで過ごし、バーミンガム大学(英国)とライプチヒ大学(ドイツ)で化学、物理を学ぶ。1902年アメリカに渡り、General Chemicalで働く。1920年から4年間フェローシップを得てジョン・ホプキンス大学で研究し、「物理的生物学」を完成させる。メトロポリタン生命に勤務。アメリカ人口学会と統計学会の会長を勤める。

Vito Volterra(1860-1940) イタリアのアンコナの貧しいユダヤ系の生まれ。早くから数学の才を示しピサ大学で学ぶ。トリノやローマ大学の教授となり、上院議員も務める。第一次大戦には50才台で従軍しヘリウムを用いた飛行船などの設計に携わる。大戦後生物学の研究を精力的に行った。ムッソリーニへの忠誠を拒否した1%の教授(12人)の一人で全ての公職を追われ海外移り住んだが亡くなる前にローマに戻った。

A 2種の生物の個体群に関するロトカ・ボルテラモデルで記述される、捕食者と生け贄の遅延を伴う生態モデルである。

第一次大戦でアドリア海が物騒になり漁師も兵役に取られたりで漁が出来なかったのも、大戦後魚が増えていると思ったところ捕食者の鮫ばかりが増えていたと言うものである。

M plot { |: PAR vtr ^:(i.8000) FST

```

vtr=: 4 : 0
NB. Lotka-Volterra Model
NB. x is 2r3 1r300 1r2 1r2
'a b c d'=: x
'xx yy'=: y
X=: xx + dt * -xx*(a - b*yy) NB. enemy
Y=: yy + dt * yy*(c - d*xx) NB. butterfly
X,.Y
)

dt=: 0.01
PAR=:2r3 1r300 1r2 1r2
NB. parameter enemy and
FST=: 1 100 NB. 10000 1000000
NB. first individual

```

$$\frac{dx}{dt} = x - (a - by)x$$

$$\frac{dy}{dt} = y - (c - dx)y$$

M ロトカ・ヴォルテラの式も2次ではあるがロレンツとよく似た形をしており、右項には微分記号がない。これを漸加法で計算しplotする。dt=0.01とする。

N 金融工学に応用した人がいると聞きましたが

Example-1 蝶と天敵とする。

初期値 1 (天敵) 100 (蝶)

天敵の増減 a=2r3 (蝶が居ない場合の一期当たり減少) 依存率が高い

b=1r300 (蝶に合わせ増える天敵の数)

蝶の増減 (c=1r2 (天敵が居ない場合の蝶の増加))

d=1r2 (天敵に捕食されて減少する蝶の減少))

*5

*5 2r3 is $\frac{2}{3}$

```
'key enemy butterfly' plot |: 1 0.01 *"1 PAR vtr ^: (i.8000) FST
pd 'eps /temp/vorterra_2.eps'
```

蝶（赤）の個体数は 1/100 で図示している。

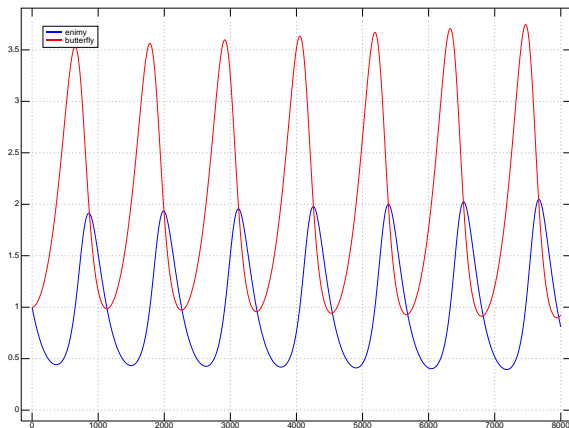


図7 $x=t$, 蝶（赤）の Y 値は 1/100

$dt=0.01$ で原点 (1,100) から時計回りに進み、概ね $1130t$ で一周する。

```
'key enemy butterfly' plot
{|:1 0.01 *"1 PAR vtr ^:(i.1130)FST
```

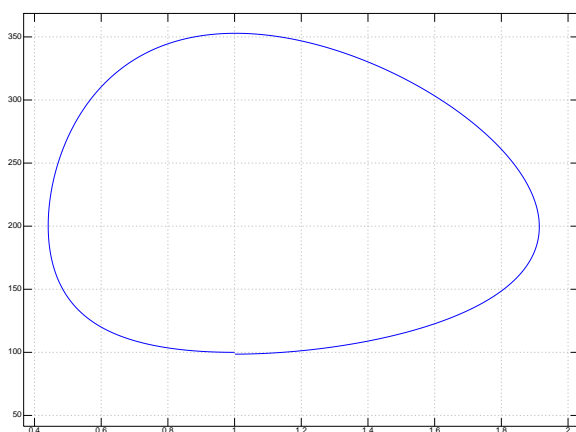


図8 Lotka-Vortera

$^:(i.50000)$ での状態。少しずつ外縁が広がってゆく。

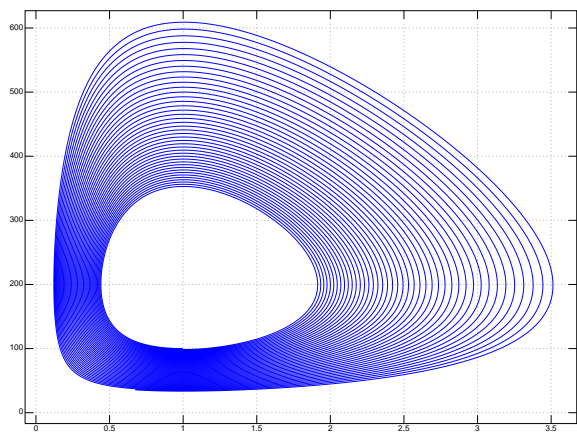


図9 Lotka-Vortera

OpenGL Oleg Kobchenko が Lorenz の OpenGL 版を作成している。
<http://www.jsoftware.com/jwiki/Essays/Lorenz%20Attractor>
の中の Lorenz Attractor で見られ、ソースコードも公開されている。
Rossler を Oleg のソースコードを少し改良して OpenGL で表示できるようにした。
<http://jsoftware.com/jwiki/Essays> \rightarrow [rosslarattractor](#) にも掲載

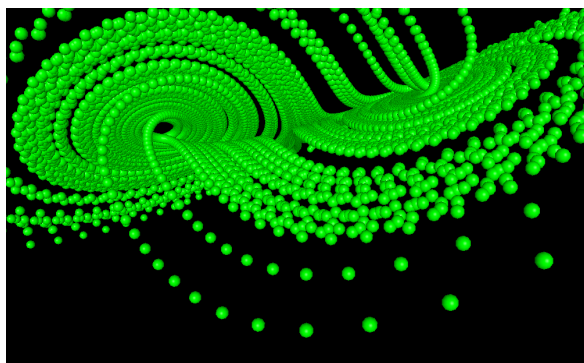


図 10 Lorenz 3d

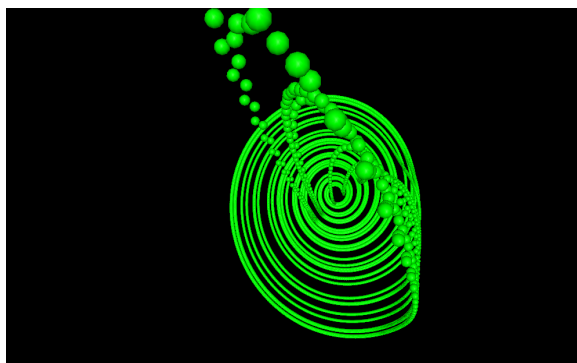


図 11 rossler 3d

5 確率推移行列の反復計算

5.1 ぴろしき

x		y
確率推移行列	内積	確率推移行列

5.1.1 マルコフ連鎖

A マルコフがプーシキンの「スペードの女王」を読んでいて閃いたと言われる。明日は昨日の続きではない。

確率過程の次の状態 (X_{n+1}) がどうなるかは、現在の状態のみに依存し、確率的に決まり、過去の履歴 ($X_0, X_1, X_2, \dots, X_{n-1}$) には無関係であることをマルコフ性と言う。

ある程度のリピート（期間）の後に推移確率行列は既約、正再帰的、非周期的な場合に定常分布に収束する。

$$\pi = \pi P$$

Worked Example ビールの銘柄選択

あるビールの銘柄。ABCD を買った人が次にどの銘柄を買うか

```

DAT1
  A   B   C   D
A  0.9 0.05 0.03 0.02
B  0.1 0.8 0.05 0.05
C  0.08 0.1 0.8 0.02
D  0.1 0.1 0.1 0.7
    
```

`., <"2 DAT1&(+/ . *) ^:(i.30) DAT1` で趨勢が分かる。

12 回程度の反復で収束し始める。初期値に関係なく顧客の定着が良い銘柄がシェアを確保する。(DAT1 の各銘柄のリピータは対角行列である) 強い銘柄に顧客をとられると吸い込まれる。

```

DAT1&(+/ . *) ^:(30) DAT1 NB. repeat 30 times
0.482836 0.252654 0.17837 0.0861406
0.481615 0.25323 0.178873 0.0862821
0.481269 0.253361 0.179072 0.0862978
0.48151 0.253256 0.178952 0.0862823
    
```

A 推移確率行列と固有値の関係は、固有値に単根の 1 がでる。他に絶対値が 1 となるものがあると次のようになり極限值に収束しないで周期解がでる。

$$\lambda = e^{i\theta} = \cos\theta + i\sin\theta$$

```

char_lf DAT1
++++-----+-----+
|1|1 0.80865 0.717559 0.673791|0.39097 _1.99957 3.8086 _3.2 1|
++++-----+-----+
( 固 有 値 ) a + bx + cx^2 + dx^3 + x^4

```

5.1.2 レスリー行列

P.H.Leslie (1900-1974)

Oxford に学び、Oxford の *Bureau of Animal Population* に在籍

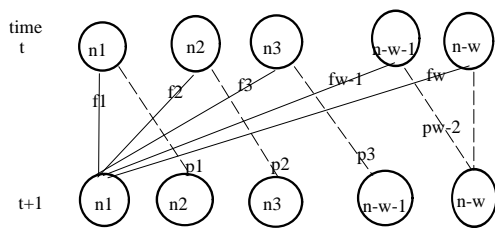
1940 年代のはじめに *Bxernardelli*(1941),*Lewis*(1942),*Leslie*(1945) がマトリクスで人口や動物の生殖のモデル化に成功した。

- A レスリー行列は出生率と生存率から x^{k+1} 年後の生存数を求めることができる。
 寿命の長い種は慣性 (モメンタム) が働くので、生じた変化が人口数に現れてくるのに時間を要する。

$$x^{(k+1)} = Lx^{(k)}$$

birth $a_i, i = 1, 2, 3, \dots, n$

death $b_i, i = 1, 2, 3, \dots, n - 1$



(Leslie matrix)

$$L = \begin{bmatrix} a_1 & a_2 & a_3 & \cdot & \cdot & \cdot & a_n \\ b_1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & b_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \dots & & & & & & \dots \\ \dots & & & & & & \dots \\ \dots & & & & & & \dots \\ 0 & \dots & \dots & \dots & 0 & b_{n-1} & 0 \end{bmatrix}$$

図 12 mreg

Worked Example SIRMON

• $A = \begin{pmatrix} 0 & 4 & 5 \\ 0.53 & 0 & 0 \\ 0 & 0.22 & 0 \end{pmatrix}$ SIRMON2

- 最大固有値 1.5778 は毎年 57.78% の増加率を示す

```

char_lf SIRMON2
++++-----+-----+

```

```
|1|1.57781 _1.29178 _0.286039|_0.583 _2.12 0 1|
+-----+-----+
```

- 固有ベクトル

```
}... dgeev_jlapack_ SIRMON2
+-----+
|_1.29178 1.57781 _0.286039 |
+-----+
| 0.923231 0.947016 0.393319|
|_0.378791 0.31811 _0.728777|
|0.0645112 0.0443551 0.560521|
+-----+
```

- $n_0 = 12 \ 12 \ 12$

各年魚 12 匹づつ 36 匹を投入。

```
SIRMON2&(+ / . *) ^:(i.16) 3#12
12      12      12
108     6.36    2.64
38.64   57.24   1.3992
235.956 20.4792 12.5928
144.881 125.057 4.50542
522.754 76.7868 27.5125
444.71  277.06  16.8931
1192.7  235.696 60.9531
1247.55 632.133 51.8531
2787.8  661.201 139.069
3340.15 1477.53 145.464
6637.45 1770.28 325.057
8706.41 3517.85 389.462
16018.7 4614.4 773.927
22327.2 8489.91 1015.17
39035.5 11833.4 1867.78
```

Worked Example Sheep の寿命の計算

Input DATA: *SHEEP*

雌の羊の各年齢の出生率と生存率 (食用は考えない)

A *Sheep* の実データが *Bradie* で紹介されていた。これを用いてスクリプトを検証しながら *Sheep* の平均寿命を計算してみよう。

```
(i.12),. SHEEP
age(1)ai 出生 bi 生存
```

```
-----
0 1      0      1
1 1 0.045 0.845
2 1 0.391 0.824
3 1 0.472 0.795
4 1 0.484 0.755
5 1 0.546 0.699
6 1 0.543 0.626
7 1 0.502 0.532
8 1 0.468 0.418
9 1 0.459 0.289
10 1 0.433 0.162
11 1 0.421      0
```

(issue *Bradie* //Caughley , data collected by Hicky)

(最初の 1 の列は初期値として用いる)

$x_0 = [1 1 1 1 1 1 1 1 1 1 1 1]$

(1) は生存数の初期値。1 の場合は比率となる。

これを *SHEEP* の (1) の列に入れた。

- レスリー行列

```
]a=.}:}: "1 leslie_mat{1 2{ |: SHEEP NB. drop lastnull raw and line
```

```
0 0.045 0.391 0.472 0.484 0.546 0.543 0.502 0.468 0.459 0.433 0.421
1      0      0      0      0      0      0      0      0      0      0      0
0 0.845      0      0      0      0      0      0      0      0      0      0
0      0 0.824      0      0      0      0      0      0      0      0      0
0      0      0 0.795      0      0      0      0      0      0      0      0
0      0      0      0 0.755      0      0      0      0      0      0      0
0      0      0      0      0 0.699      0      0      0      0      0      0
0      0      0      0      0      0 0.626      0      0      0      0      0
0      0      0      0      0      0      0 0.532      0      0      0      0
0      0      0      0      0      0      0      0 0.418      0      0      0
```

```

0      0      0      0      0      0      0      0      0 0.289      0      0
0      0      0      0      0      0      0      0      0      0 0.162      0

```

- 固有値

実数は 2 個. 最大固有値 1.0899 から年増加率は 9%

```

6 2 $ > 1{ 6 2 ;1{ char_lf a
          1.08999   0.395417j0.520533
0.395417j_0.520533  _0.174379j0.59078
_0.174379j_0.59078  0.0932632j0.533933
0.0932632j_0.533933      _0.486984
_0.380423j0.232537 _0.380423j_0.232537
_0.235381j0.322598 _0.235381j_0.322598

```

- 固有ベクトル

最大固有値 1.08999 の固有ベクトルとそのパーセント比
年齢別の分布が求められる。

```

(tmp=(i. # v0),.v0,. v0 % +/ v0),+ tmp

0      0.586481      0.23937 NB. 0-1 age
1      0.53806      0.219607 NB. 1-2 age
2      0.417124      0.170248 NB. 2-3 age
3      0.315333      0.128702 NB. 3-4 age
4      0.229992      0.0938705
5      0.159308      0.0650211
6      0.102163      0.0416975
7      0.0586737      0.0239475
8      0.0286373      0.0116882
9      0.0109821      0.00448231
10     0.0029118      0.00118844
11     0.000432766    0.000176632
total  2.4501          1
      (固有ベクトル)(構成比)

```

A 各才の生存率と死亡率で構成したを確率推移行列から平均寿命が求められる。この計算はレ
オンテフ逆行列と同様、逆行列

$$(I - Q)^n = (I - Q)^{-1}$$

を求める。

- 確率推移行列

```

age_mat_sub0 1- 0 2{"1 SHEEP
  1 0      0      0      0      0      0      0      0      0      0      0
0.155 0 0.845      0      0      0      0      0      0      0      0
0.176 0      0 0.824      0      0      0      0      0      0      0
0.205 0      0      0 0.795      0      0      0      0      0      0
0.245 0      0      0      0 0.755      0      0      0      0      0
0.301 0      0      0      0      0 0.699      0      0      0      0
0.374 0      0      0      0      0      0 0.626      0      0      0
0.468 0      0      0      0      0      0      0 0.532      0      0
0.582 0      0      0      0      0      0      0      0 0.418      0
0.711 0      0      0      0      0      0      0      0      0 0.289      0
0.838 0      0      0      0      0      0      0      0      0      0 0.162
  1 0      0      0      0      0      0      0      0      0      0      0

```

- 平均余命の計算

$$(I - Q)^n = (I - Q)^{-1}$$

レオンティエフ逆行列と同じである。

右端列は合計で、当歳の平均余命であり、0才の余命(4.13)が *Sheep* の平均寿命である。

```

(i. # tmp),. " 7j3 ":(+/"1 tmp),.~ (tmp=. %. i_minus }.}."1 a)
 0 1 0.845 0.696 0.554 0.418 0.292 0.183 0.097 0.041 0.012 0.002 4.139
 1 0      1 0.824 0.655 0.495 0.346 0.216 0.115 0.048 0.014 0.002 3.715
 2 0      0      1 0.795  0.6  0.42 0.263 0.14 0.058 0.017 0.003 3.295
 3 0      0      0      1 0.755 0.528 0.33 0.176 0.073 0.021 0.003 2.887
 4 0      0      0      0      1 0.699 0.438 0.233 0.097 0.028 0.005 2.499
 5 0      0      0      0      0      1 0.626 0.333 0.139 0.04 0.007 2.145
 6 0      0      0      0      0      0      1 0.532 0.222 0.064 0.01 1.829
 7 0      0      0      0      0      0      0      1 0.418 0.121 0.02 1.558
 8 0      0      0      0      0      0      0      0      1 0.289 0.047 1.336
 9 0      0      0      0      0      0      0      0      0      1 0.162 1.162
10 0      0      0      0      0      0      0      0      0      0      1      1

```

(少数点下3桁まで表示)

M 先のレオンティエフ逆行列は次のようであった。

$$(I - C)^{-1} = I + C + C^2 + C^3 + \dots + C^m$$

これは次のようになり、逆行列で C^m が求められる。収束 (m) を気にしなくとも良い。

$$(I - C)^{-1} - I = C + C^2 + C^3 + \dots + C^m$$

```

C + C^2 + C^3 + ... + C^m
+ / C 0 & (+ / . *) ^ : (i.100) C 0
1.96296 1.85185 1.11111
0.925926 1.03704 0.555556
0.555556 0.555556 0.666667

(I - C)^-1 - I
(% . I 0 - C 0) - I 0 = . = / ~ i.3
1.96296 1.85185 1.11111
0.925926 1.03704 0.555556
0.555556 0.555556 0.666667

```

N 固有値の絶対値が 1 より小さい場合が具合が良さそうですが。

```

char_lf (= / ~ i.3) - C 0
++-----+-----+
|1|0.9 0.764575 0.235425|_0.162 1.08 _1.9 1|
++-----+-----+
( 固 有 値 ) f=a+bx+cx^2+x^3

```

レスリー行列による将来頭数推計 推計年数は ^ : (i.n) で指定する。

```

(i. # tmp),. ". 7j3": tmp=. a & (+ / . *) ^ : (i.26) 12#1
0 1 1 1 1 1 1 1 1 1 1 1 1
1 4.764 1 0.845 0.824 0.795 0.755 0.699 0.626 0.532 0.418 0.289 0.162
2 2.889 4.764 0.845 0.696 0.655 0.6 0.528 0.438 0.333 0.222 0.121 0.047
3 2.354 2.889 4.026 0.696 0.554 0.495 0.42 0.33 0.233 0.139 0.064 0.02
4 3.173 2.354 2.441 3.317 0.554 0.418 0.346 0.263 0.176 0.097 0.04 0.01
5 3.591 3.173 1.989 2.012 2.637 0.418 0.292 0.216 0.14 0.073 0.028 0.007
6 3.756 3.591 2.681 1.639 1.599 1.991 0.292 0.183 0.115 0.058 0.021 0.005

```

7	4.187	3.756	3.034	2.209	1.303	1.208	1.392	0.183	0.097	0.048	0.017	0.003
8	4.612	4.187	3.174	2.5	1.756	0.984	0.844	0.871	0.097	0.041	0.014	0.003
9	4.964	4.612	3.538	2.615	1.988	1.326	0.688	0.528	0.463	0.041	0.012	0.002
10	5.392	4.964	3.897	2.915	2.079	1.501	0.927	0.431	0.281	0.194	0.012	0.002

(10年分表示)

9%の増加率 (1.09^{30}) は相当パワーがある

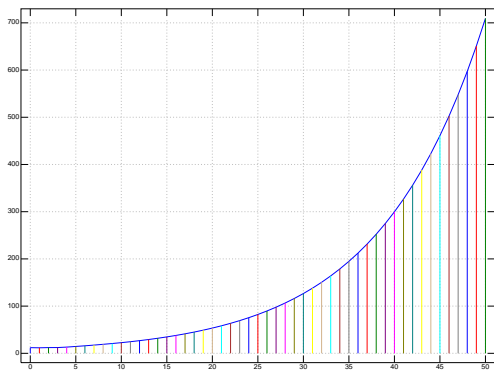


図 13 Sheep

6 References

- Michael F. Barnsley [Fractals Everywhere 2nd Edition] 1993 Morgan-Kaufmann
 David C. Lay [Linear Algebra and its Applications] 2nd.ed. Addison-Wesley 2000
 F. シャトラン/伊理正夫、由美訳「行列の固有値」 Springer/Tokyo 1993/2003
 石村貞夫・石村園子「ブラックショールズ微分方程式」 1999 東京図書
 田村明久・村松正和「最適化法」 共立出版 2002
 服部雄一 [Fortran による数値計算] 1992 森北出版
 平下幸男「数理科学のレッスン」 産業図書 1992
 伏見正則「確率と確率過程」 朝倉書店 2004
 三井・小藤・斉藤 [微分方程式による計算科学入門] 2004 共立出版
 J602: <http://www.jssoftware.com>
 Script: http://homepage3.nifty.com/asagaya_avenue