そるぶ（１）

Masato Shimura
JCD02773@nifty.ne.jp

2004/12/11

目次

ijs	*sorubu.ijs*
tex	sorubu.tex

1 関数の定義とグラフ

1.1 多項式

Example $y = x^3 + x - 1$

多項式 (p.) に渡す関数を定義するときは、高次の方が右に来る。

steps は nimeric のファイルに入っている関数で、グラフ用に間隔を滑らかにしてくれる。

i : は主にグラフ用 0 の両側に展開するので、X 軸の指定に有用である。

```
 load 'plot numeric trig'

    i:5
_5 _4 _3 _2 _1 0 1 2 3 4 5

 f0=.  _1 1 0 1 &p.

_1 1 0 1&p. i:5
_131 _69 _31 _11 _3 _1 1 9 29 67 129

plot _1 1 0 1&p. steps _5 5 100

 pd 'plot_f0.eps'
```

Example $x^4 - 10x^3 + 36x^2 - 58x + 35 = 0$

```
plot  35 _58 36 _10 1& p.  steps _100 100  1000
```

Example $x^4 + y^6$

```
f3=.3 : '(({. y.)^4) + ({: y.)^6'
```

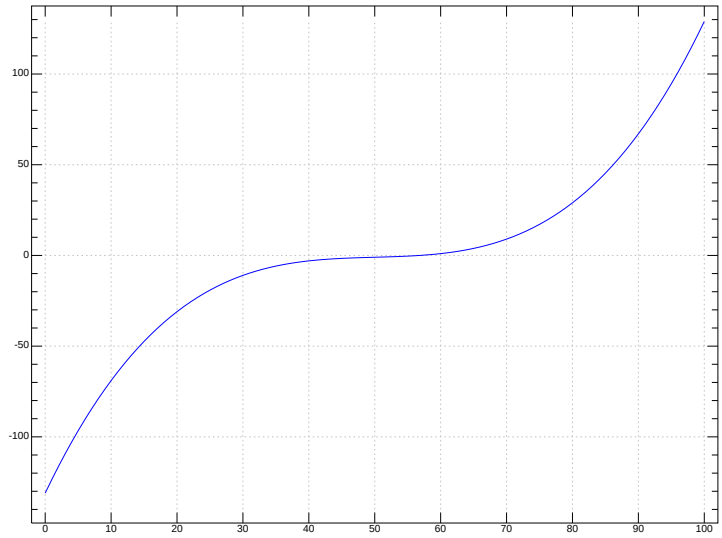


図 1　plot_f0

```
'wire' plot > f3 L:0 {(steps _1 1 100);steps _1 1 100
```

steps の 0 のとき関数に ^_n が入ると無限大がでて plot しないことがある

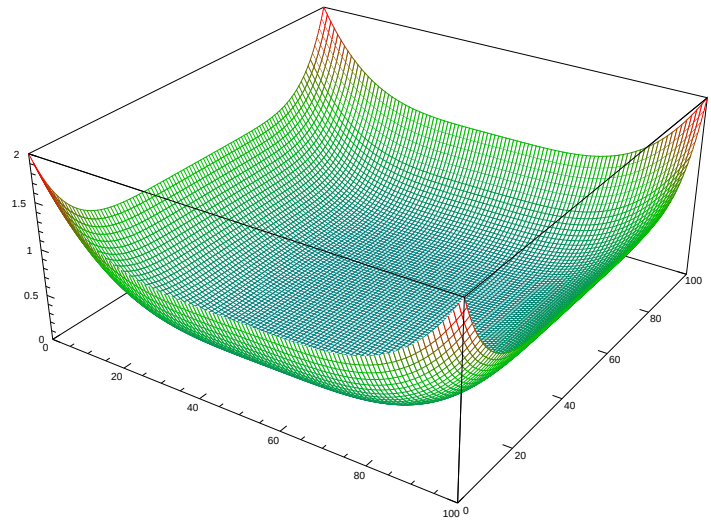


図 2　plot_f3

2　非線形方程式の解法

2.1　多項式

多項式は J の強力な polinominal(*p.*) の機能で、いきなり解ける。

Example　$x^4 - 10x^3 + 36x^2 - 58x + 35 = 0$

解 4.41421 2 ± 1*j* 1.58579

```
  p. 35 _58 36 _10 1
+-+----------------------+
|1|4.41421 2j1 2j_1 1.58579|
+-+----------------------+
```

2.1.1　固有値を用いた解法

上記の Example を固有値を用いた対角化によって解く。

$p_n x = x^n + a_1 x^{n-1} + \cdots + a_1 x + a_n$

多項式の係数から次の行列を作る。

$$A = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

対角行列にくる固有値 λ は $p_n(x) = 0$ の根となり、残余は固有ベクトルとなる。

```
   a2
10 _36 58 _35
 1   0  0   0
 0   1  0   0
 0   0  1   0

 ] a3=.>{: zgeev_jlapack_ a2
 0.974005             0.895144            0.895144 0.785996
```

```
 0.220652   0.358057j_0.179029   0.358057j0.179029 0.495651
0.0499867   0.107417j_0.143223   0.107417j0.143223 0.312558
 0.011324 0.0143223j_0.0787726 0.0143223j0.0787726   0.1971

  (<0 1)&|: (%. a3) +/ . * a2 +/ . * a3
4.41421j_2.36526e_12 2j1 2j_1 1.58579j5.32047e_13
```

Example $8x^3 + 12x^2 + 14x + 9$

```
  p. 9 14 12 8
+-+-----------------------------------------+
|8|_0.288073j1.06524 _0.288073j_1.06524 _0.923854|
+-+-----------------------------------------+
```

8 **は反復回数**

```
   a2
_8 _12 _14 _9
 1   0   0  0
 0   1   0  0
 0   0   1  0

 ] a3=.>{: zgeev_jlapack_ a2
   0.987873            0.620862 _0.5            0.620862
  _0.153381 _0.124263j_0.510305  0.5  _0.124263j0.510305
  0.0238146  _0.394564j0.204271 _0.5 _0.394564j_0.204271
_0.00369755    0.246867j0.28342  0.5   0.246867j_0.28342

  (<0 1)&|: (%. a3) +/ . * a2 +/ . * a3
_6.44064 _0.279679j1.14855 _1 _0.279679j_1.14855
```

(<0 1)&|:) は対角行列を取り出す定跡

最高次の_6.44064 は解ではない。 精度は p. より悪い。

```
plot(steps _10 10 100) ; 9 14 12 8&p. steps _10 10 100
```

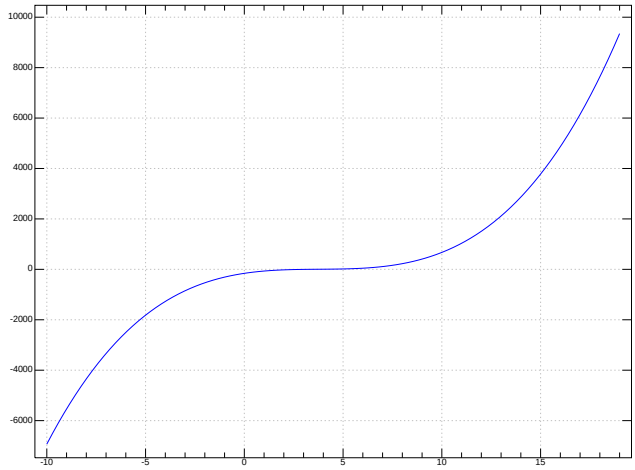


図 3　多項式

Example　$1 + 2x + x^2$ の $x = 2$ における値

```
  1 2 1 &p. 2
9
```

2.2　Newton 法（一次）

非線形方程式の解法で、ニュートン法が有名である。J でのニュートン方については、N.Thomson の芸術的なスクリプトが次の文献に載っている。

[Applying Matrix Divide in APL and J] (APL Quote Quad 1994)

*1

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。J は微分演算子 D. があるので APL より容易にスクリプトが作れる。

ニュートン法は微分演算子を用いてシンプルに定義できる。動詞を左パラメーターに取

*1 N.Thomson のスクリプトは旧版の J6.2（1·62 に相当）なので、バージョンの経過によりにより仕様が変っている部分は補正した。

るので副詞で定義しなければならない。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。

Example $y = x^3 + x - 1$ をニュートン法で解いてみよう。

fn で　$x^3 + x - 1$ を定義する。初期値を関数のグラフを書いた後、ベクトルでレンジを指定するため、ランクを (”0) にしてある。newton は関数をパラメーターにとる作用素（副詞）であってベクトル計算のため、同じくランクを (”0) にしてある。

```
   _1 1 0 1&p. new_1 i:3
0.682328 0.682328 0.682328 0.682328 0.682328 0.682328 0.682328
```

Example $8x^3 + 12x^2 + 14x + 9$

先の多項式を Newton 法で解いてみる。実根のみ求まる。

```
   9 14 12 8&p. new_1 i:3
_0.923854 _0.923854 _0.923854 _0.923854 _0.923854 _0.923854 _0.923854
```

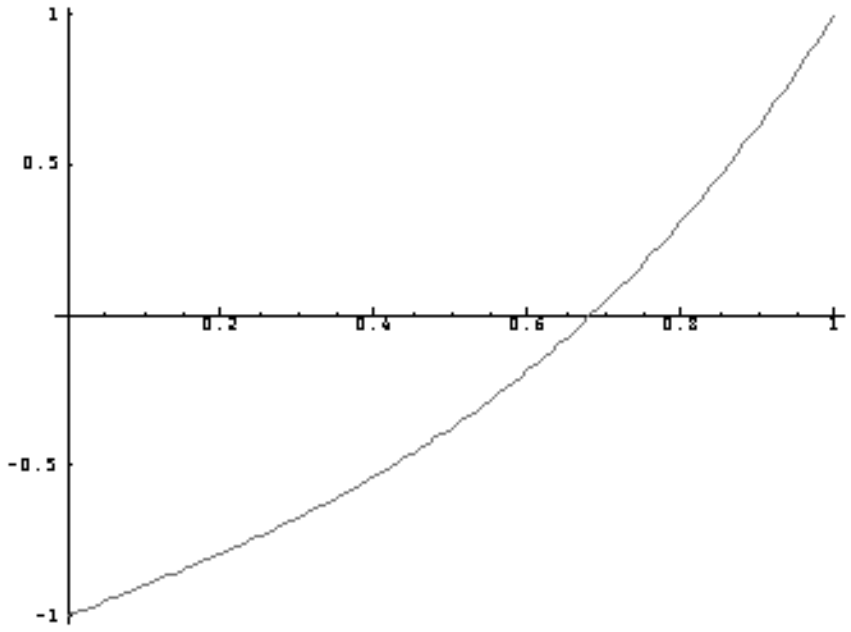
解の検証もかねて、詳細のグラフを書いてみよう　plot の steps はX軸を正しく生成させるため、Newton に与えた引数と同じとした。

```
plot (steps 0 1 100); _1 1 0 1&p. steps 0 1 100
```

Newton 法といわれる $x_{k+1} = x_k - f(x_k)/f'(x_k)$ という公式にまとめたのはラプソンで、1690 年にロンドンで出版されている。テーラー展開の高次の項を無視して, 局所的な接線の 1 次式で $f(x)$ を近似し, その 0 の点を次の近似値とする方法は Newton により使われたものである。

2.2.1　Reference

NormanThomson　[Applying Matrix Divide in APL and J] (APL Quote Quad 1994)
小国 力「新数値計算法」 サイエンス社 1997



3　線形連立方程式

3.1　逆行列を用いる

3.2　クラメール法

線形一次連立方程式は、配列計算言語の得意とするところであり、逆行列を用いて直ちに解が求まる。

$Ax = b$
$x = \frac{b}{A}$
これは　$b\,\%.\,A$ である

$\begin{cases} x_1 + 2x_2 - x_3 = 2 \\ 3x_2 + 4x_4 = 18 \\ x_1 - x_3 = -2 \end{cases}$	S1 1 2 _1 2 0 3 4 18 1 0 _1 _2

係数行列の逆行列を用いる方法

$$\begin{bmatrix} 2 \\ 18 \\ 2 \end{bmatrix} \%. \begin{bmatrix} 1 & 2 & -1 \\ & 3 & 4 \\ 1 & & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

クラメール法は次のような行列の構造を残した行列除算であり、すっきり解けた場合は単位行列が現れる。（場合によっては、0に限りなく近い微少な数が残る。）

$$\begin{bmatrix} 1 & 2 & -1 & 2 \\ & 3 & 4 & 18 \\ 1 & & -1 & -2 \end{bmatrix} \%. \begin{bmatrix} 1 & 2 & -1 \\ & 3 & 4 \\ 1 & & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

3.2.1 script and worked examples

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

```
   S1
1 2 _1  2
0 3  4 18
1 0 _1 _2
```

```
   cr S1
1 0 0 1
0 1 0 2
0 0 1 3
```

3.3 掃き出し法

3.3.1 Gauss-jordan 法

Math package of J include gauss-jordan method

```
   load 'system\packages\math\linear.ijs'


   gauss_jordan S1
1 0 0 1
0 1 0 2
0 0 1 3
```

3.4 対角化を用いる

固有値と固有ベクトルを求めて相似変換により対角化して連立方程式を分解する方法

対称行列と固有値が異なる非対称行列に用いることができる。簡単なスクリプトを添付するので、実際の対角化と連立方程式の

最初に addons の LAPACK から *jlapack.ijs* , *dgeev.ijs* の 2 つのファイルをロードする。

```
  S3
 6  4 _2
 4 12 _4
_2 _4 13


  a=: dgeev_jlapack_  S3


   1 2 {a
+------+------------------------------+
|4 18 9|   0.894427  0.333333 0.298142|
|      |  _0.447214  0.666667 0.596285|
```

```
|      |1.49233e_16 _0.666667 0.745356|
+------+------------------------------+

  a2=. (%.>2{a) +/ . * S3 +/ . * >2{a

           4 1.04835e_16 _9.99201e_16
_2.25555e_15          18  5.32907e_15
  4.4475e_16           0            9


taikaku=: 3 : 0
NB. solve with disg
TMP0=: dgeev_jlapack_ y.
TMP1=: (%.>{: TMP0) +/ . * y. +/ . * >{:TMP0
)

taikaku2=: 4 : 0
NB. x. is y_n
NB. seimal regression
TMP0=: dgeev_jlapack_ y.
TMP1=: (%.>{: TMP0) +/ . * y. +/ . * >{:TMP0
TMP2=: ( |: >{: TMP0) +/ . * x.
TMP3=: TMP2 %. TMP1
(> {: TMP0) +/ . * TMP3
)
```

4 非線形同時方程式

4.1 多変数 Newton 法

4.1.1 2 変数

$f_0 = e^x + xy - 1$

$g_0 = sinxy + x + y + 2$

これを Tacit と Explicit で定義する。Explicit では動詞で定義する。右引数のｘ、ｙは

リストで与える。（x、yの順）

```
f0=: -&1@(*/)+^@{.
g0=: +&2@(+/)+1&o.@*/

f00=: 3 : '(^{.y.)+(*/y.)-1'
g00=: 3 :  '2+(+/ y.)+1&o.*/ y.'
```

この Multiple Newton は次のように定義される。 Divide(%) **に代えて** Matrix Divide(%.) が用いられている。

Multiple Newton では D.1 を用いるがヤコビアン jac=: 1 : 'x. D.1' を意識しなくとも良い。

(^:17) は任意の整数であって、例として 17 が与えられている。

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

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

   (f00,g00) new_2 1 1
_9.41097e_6 _2
```

(f0,g0) はフォークである。

```
9.41097e_6 _2 は 0 _2
```

である。

4.1.2 3 変数の例

$h_1(x,y,z) = 16x^4 + 16y^4 + z^4 - 16$

$h_2(x,y,z) = x^2 + y^2 + z^2 - 3$

$h_3(x,y,z) = x^3 - y$

```
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
```

(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。

4.1.3 Reference

NormanThomson [Applying Matrix Divide in APL and J] (APL Quote Quad 1994)

5 e^{At} の計算

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 \cdots$$

5.1 スペクトル分解

行列を射影の一次結合で表すことをスペクトル分解という。

固有値が重根の場合には, 部分分数展開が必要。

行列 A がスペクトル分解可能のとき、その射影行列は $P_1, \cdots .P_r$ は次により求められる。

$$P_k = \frac{(A-\lambda_1 I)\cdots \underbrace{k} \cdots (A-\lambda_r I)}{(\lambda_k-\lambda_{1)}\cdots \underbrace{k} \cdots (\lambda_k-\lambda_r)}$$

$$\frac{1}{5\cdot 6} = \frac{(\frac{1}{6}-\frac{5}{6})}{5} + \frac{1}{6}$$

```
   a7
 1  1  _1
```

```
 1 1 _1
_1 1  1


  dgeev_jlapack_ a7  NB. compute eigenvalues and eigenvectors
+---------------------------------+----------------+
|   0.707107    _0.707107 _0.57735|_6.66134e_16 2 1|
|  _0.707107 _7.05086e_16  0.57735|                |
|1.66533e_16     0.707107  0.57735|                |
+---------------------------------+----------------+

 as=. 0 1 2  NB. _0.66134e_16 is 0

   (/:~ as ) sp a7
 1 _1  0
 0  0  0     P1
 1 _1  0

_1  1  1
_1  1  1    P2
_1  1  1

 1  0 _1
 1  0 _1    P3
 0  0  0
```

３本の固有ベクトルが取れたので、P1　P2　P3　から任意の一本づつ選び、対角化行列が作れる。

$A = 0 \cdot P_1 + 1 \cdot P_2 + 2 \cdot P_3$　　0,1,2 は固有値

explicit で同じスペクトル分解を書いてみた。

```
 0 1 2 spe a7
```

```
+------+------+------+
|1 _1 0|_1 1 1|1 0 _1|
|0  0 0|_1 1 1|1 0 _1|
|1 _1 0|_1 1 1|0 0  0|
+------+------+------+
```

重根の場合でもとおる

```
   a8
0 1 1
1 0 1
1 1 0
```

```
   2 _1 spe_d a8
+-----------------------------+
|0.333333 0.333333 0.333333   |
|0.333333 0.333333 0.333333   |
|0.333333 0.333333 0.333333   | NB. (1r3)(M+I)
+-----------------------------+
| 0.666667 _0.333333 _0.333333|
|_0.333333  0.666667 _0.333333|
|_0.333333 _0.333333  0.666667| NB. (-1r3)(M-2I)
+-----------------------------+
```

$$\mathrm{e}^{Mt} = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} e^{2t} - \frac{1}{3}\begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix} e^{-t}$$

example

$$\frac{dx}{dt} = x + 2y + z$$
$$\frac{dy}{dt} = -x + 4y + z$$
$$\frac{dz}{dt} = 2x - 42y$$

a9 1 2 1 _1 4 1 2 _4 0	1{ dgeev_jlapack_ a9 +-----+ |2 1 2| +-----+ 固有値	2 1 spe a9 +--------+ | 0 2 1| |_1 3 1| | 2 _4 _1|NB.P1 +--------+ | 1 _2 _1| | 1 _2 _1| |_2 4 2|NB.P2 +--------+

$$e^{Mt} = e^{t}P_1 + e^{2t}P_2$$

5.1.1 script

```
spe=: 4 : 0
NB. spectol explicit
NB. x. is /: eigenvalue
NB. y. is matrix
NB. Usage. x. spe y.
ind=: -. (i. # x.) =/ ~i. # x.
TMP0=: x. ,. ind #/ x.
TMP2=: */"1 TMP1=:({."1 TMP0) - }."1 TMP0 NB. bunshi
I=: =/~i. # y. NB. tani matrix
TMP3=: y. - L:0 <"2(}."1 TMP0) */ I
({@>% TMP2) * L:0  ({."1 TMP3) +/ . * L:0 {:"1 TMP3
if. 1=({: $ TMP3) do. ({@> % TMP2) * L:0 TMP3 end.
NB. if. double eigenbalue
)
```

5.1.2 **参考文献**

笠原皓司「行列の構造」（現代応用数学の基礎）日本評論社 1994

6 ヘッシアンと最適化

6.1 Hessian

newton 法による最適化は一階偏微分と二階偏微分（ヘッセ行列）を用いる。[*2]

ヘッセ行列は, 実数値関数 f の 2 次微分を次のように並べた n の行列である。

[*2] 2 階偏微分係数を要素とする行列は、目的関数に対してはヘッセ行列、勾配ベクトルに対してはヤコビ行列と呼ばれる

$$Hf_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}, (i, j = 1, \cdots, n)$$

偏微分を求めるスクリプトは、C.Reiter のスクリプトを用いる。

これは、２変数の一階偏微分をマトリックスに表し、接続詞　Ｐで一階偏微分を、P jac で二階偏微分を求める。

（例）

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

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

```
hs1=:4 4 $ _2 2 ( 1 4 )}  16#0


hs2=:4 4 $ _1 1 _2 ( 2 3 4)}  16#0
N=: 4
hs3=. a1,:a2

   hs3
```

	$x_2{}^0$	$x_2{}^1$	$x_2{}^2$	$x_2{}^3$
$x_1{}^0$	0	_2	0	0
$x_1{}^1$	2	0	0	0
$x_1{}^2$	0	0	0	0
$x_1{}^3$	0	0	0	0
$x_1{}^0$	0	0	_1	1
$x_1{}^1$	_2	0	0	0
$x_1{}^2$	0	0	0	0
$x_1{}^3$	0	0	0	0

*3

初期点 $x_0 = (3, 3)$ より $\nabla f(x^0)$ を求める。

```
hs3 P 3 3
0 12
```

初期点 $x_0 = (3, 3)$ より $Hf(x_0)$ を求める。

```
hs3  P D.1("0) 3 3
 2 _2
_2 21
```

6.2 newton 法による最適化

制約無し最適化の解法としての newton 法は反復法によるものである。
newton 法は正則なヘッセ行列の逆行列が必要である。
ヘッセ行列の解は、`x. P jac y.` で求めることが出来る。
x^0 を初期値とすると、ニュートン方向（ベクトル）は
$d = -Hf(x^0)^{-1}\nabla f(x^0)$ を計算し、　$x^{k+1} \leftarrow x^k + d^k$ の反復を収束するまで行う。

6.2.1 過程と結果

（最初に N =左引数のマトリクスサイズ（`1 { $ x.`） は手動で指定する。）

*3 $x_n{}^0 = 1$

$Hf(x_0)^{-1}$

```
(%. hs3 P  D.1("0) 3 3)
 0.552632 0.0526316
0.0526316 0.0526316
```

$\nabla f(x_0)$

```
   hs3 P 3 3
0 12
```

$d_0 = Hf(x_0)^{-1}\nabla f(x_0)$

```
  ( %. hs3 P  D.1("1) 3 3) +/ . * hs3 P 3 3
0.631579 0.631579
```

$x_1 = x_0 + d_0 = x_0 - Hf(x_0)^{-1}\nabla f(x_0)$

```
 3 3 - ( %. hs3 P  D.1("1) 3 3) +/ . * hs3 P 3 3
2.36842 2.36842
```

値を初期値に代入し、反復する

```
c=. 2.36842 2.36842
   c - (%. hs3 P D.1("0) c) +/ . * hs3 P c
2.07716 2.07716
```

「結果」

```
a3 new_ml 3 3     NB. １０回反復
2 2
```

停留点 2　2 **に収束した。**

ニュートン法の計算には、正定置で正則なヘッセ行列が必要である。

6.2.2 SCRIPT

```
NB. ======================
NB. Partial differential
NB. by C.Reiter
NB. ========================
NB.name of  del f(x1 x2) is fixed
a=: 3 5 2 *"1 (i.2 3 3 )e. 5 7 12
N=: 1{ $ a   NB. matrix size using in P
NB. P=: 1 : '((m. "_ +/ . * {:) +/ . * {.)@(^/&(i.3))'
P=: 1 : '((m. "_ +/ . * {:) +/ . * {.)@(^/&(i. N ))'

f=: a P
jac=: (D.1)"1

NB. ========================================
NB. Script is written by M. Shimura 08 July 2002
NB. ========================================
  new_ml=:  4 : 0
NB. Newton method for Maximum Likelyhood function
NB. x. Partial derivative matrix
NB. y. Start value
NB. N=: Matrix size
Y=: y.
NR=: 0
while. NR < 10 do. NB. repeat 10 times (change if you wish)
D=: (%. x. P  jac Y) +/ . * x. P Y
Y=:NT=: Y - D
NR=: >: NR
end.
Y
)
```

6.2.3 **参考文献**

C. Reiter [Fractal Visualization with J]

田村明久　村松正和「最適化法」工系数学講座 Vol １７　共立出版　2002

常微分方程式の解法

7 Euler 法

$given x - values \qquad x_0, x_1, x_2, \cdots, x_N$

$y' = f(x, y), y(x_0) = y_0$

$y_{n+1} = y_n + f(x_n, y_n)(x_{n+1} - x_n), n = 0, 1, 2, \cdots, N - 1,$

オイラー法は一次のルンゲ$cdotクッター法でもある。

$y_1 = 0 + f(1, 0)(1.1 - 1) = 0.1$

$y_2 = 0.1 + f(1.1, 0.1)(1.2 - 1.1) = 0.22$

$y_3 = 0.22 + f(1.2, 0.22)(1.4 - 1.2) = 0.504$

$y_4 = 0.504 + f(1.4, 0.504)(1.5 - 1.4) = 0.6944$

例

式	初期条件
$y' = 2x + 3y$	$y(0) = 1$

7.0.4 実行例

上の関数を実行。区分は１０２４

```
f15=:3 : '(2 * {. y.)+ 3 * {: y.'

f15 euler 1;0 1

0.993164 23.1313
   0.994141  23.201
   0.995117 23.2709
   0.996094 23.3411
    0.99707 23.4114
   0.998047 23.4819
   0.999023 23.5527
```

7.0.5 Script

```
NB. ====================
euler=: 1 : 0
NB. Usage: f euler y.
NB. u. is function //e.g. fn=: +/  is y'=y+x
NB. Usage: u. euler y. //e.g. f euler 1;0 1//(y0;y1)
NB. y0 is syokiti // y1 is  range of x  //e.g. 0 1
BAND=:+/ | tmp=: > 1 { y.
dh=: BAND % 1024  NB.  dh //pitch is 1024
XSTEP=: ({. >{: y.) + (i.1024)  * dh
ANS=: ( (# XSTEP) , 2) $ 0
 NB. xn=: {.>{: y. NB. X0
 NB.  yn=: >{. y.
X=: {.>{: y. NB. X valuer
Y=: >{. y. NB. y1
COUNTER=: 0
while. COUNTER < # XSTEP do.
TMP=:dh *(TMP=: X,Y)
Y=:Y+ u. TMP NB. u. TMP is f(x,y)
ANS=:(X, Y) (COUNTER)}ANS
X=: X+dh
COUNTER=: >: COUNTER
end.
ANS
)
```

7.0.6 Reference

Peterson/Sochacki [Linear Algebra and Defferential Equation] Addison-Wesley 2002

戸川　隼人「数値計算」　岩波書店　１９９１

8 Rungr-Kutter 法

ドイツの数学者　Carle Runge(1856-1927) が 1895 年にテーラー法を用いた解法を考案し, 1901 年に, 同じくドイツの数学者 M.W.Kutter(1867-1944) が runge 法を改良したものが Runge-Kutter 法である。

4th order initial value

$y' = f(x,y), y(x_0 = y_0$

weighted average of values

$x_0, x_0 + h, x_0 + 2h \cdots x_0 + Nh$

set

$a_n = f(x_n, y_n)$

$b_n = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}a_n)$

$c_n = f(x_n + \frac{h}{2}, y_n + \frac{h}{2}b_n)$

$f(x_n + h, y_n + hc_n)$

The approximation for $\phi(x_n + 1)$

$y_{n+1} = y_n + \frac{h}{6}(a_n + 2b_n + 2c_n + d_n)$

8.0.7 実行例

先のオイラーと同じ例題での結果。区分は 1024

```
f15 runge 1;0 1


0.993164 23.2383
   0.994141 23.3084
   0.995117 23.3787
   0.996094 23.4493
    0.99707   23.52
   0.998047  23.591
   0.999023 23.6622
```

8.0.8 Script

```
 runge=:1 : 0
 NB.  Runge Kutter Method
 NB.  y. A;B C //e.g. y.=: 2; 0 1
 NB. A is syokiti B C is start and goal of point to calc
 NB. ussage: u. runge y. // u. is verb(function) ex. f=.+/ + */
 NB. e.g. fn=: (^&2@{.)%}.
 BAND=:+/ | tmp=: > 1 { y.
 dh=: BAND % 1024  NB.  dh NB. pitch is 1024
 XSTEP=: ({. >{: y.) + (i.1024)  * dh
 ANS=: ( (# XSTEP) , 2) $ 0
  xn=: {.>{: y. NB. X0
  yn=: >{. y.
 COUNTER=:0
  while. COUNTER   < # XSTEP do.
NB.   if. COUNTER = PIECE do. goto_end. end.
  x1=:xn + dh % 2
  x3=:xn + dh
  k0=: (u.  x3 , yn) * dh
  k1=: (u.  x1,y1=: yn +  k0 * 0.5) * dh
  k2=: (u.  x1,y2=: yn +  k1 * 0.5)* dh
  k3=: (u.  x3,y3=: yn + k2)* dh
  yy=:   (+/ k0 , (2 * k1 ) ,(2 * k2) , k3)  % 6
  yn=:yn + yy
  ANS=: ((COUNTER{XSTEP),yn) COUNTER } ANS
NB. ANS=:(  x3 , yn) ((COUNTER , 0) ;( COUNTER , 1)) } ANS
  xn=:x3
  COUNTER=:>:COUNTER
  end.
NB. label_end.
```

```
ANS
)
```

8.0.9　Reference

Peterson/Sochachi [Linear Algebra and Differential Equation] Addison·Wesley 2002
戸川　隼人「数値計算」　岩波書店　１９９１