J研究会資料　2006/10/28

Jによる微分方程式のグラフィック・アプローチーその１

１階常微分方程式－数値解と方向場表示の活用

西川　利男

１．コンピュータ時代の微分方程式へのアプローチ

これまでの微分方程式の教科書ではもっぱら方程式を手で解く技術（いわく変数分離形同次形、…）の習得にのみ終始してきた。前報[1]にも述べたが、微分方程式は現代において最も重要な数学のひとつであるが、その理解のしかたにはもっと現代に合ったアプローチがとられるべきと思う。すなわち、コンピュータによる数値解とそのグラフによる理解である。これにより微分方程式はもっと身近かなものになるであろう。今回、まずは１階常微分方程式について取り上げる。

たとえば、$y' = -x / y$　なる微分方程式が変数分離で解けることを知っても、それだけでは大して役にも立たない。これは円に対するものであることがわかって初めて微分方程式のイメージがつかめる。ちなみに微分方程式 $y' = y/x$ は、微分がその座標の傾きに等しいことから、当然原点(0, 0)を通る直線である。このことから、これに直交する先の $y' = -x / y$　は円だとすぐわかる。

あるいは式としては簡単な $y' = y^2 - x^2$　を解析的に解こうとすると普通には解けない。後に示すが、数値計算で解きグラフ化すると仲々おもしろい曲線になる。試しに数式処理システム Maple V で解析解を求めるとベッセル関数の恐ろしく大変な式が出てきた。

しかし、実用としての微分方程式の価値は、こんなマジックもどきのデモにあるのではない。自然現象、社会事象の表式として、発散、減衰、振動の問題、その１つとしてのロジスティックの式 $y' = a\,y(1-y)$ など…、これらはずっと地味な式である。しかし、これらの微分方程式により現代文明は支えられているといってもよいのである。

ここで、微分方程式の方向場(Direction　Field)について一言述べる。方向場表示とは微分方程式を解くことなく、x , y の各値に対する微係数（＝方向）を示す図である[3]。これによって大づかみの解のようすがイメージできる。筆者はこれと数値解とをつぎのように位置づけしている。

方向場 … 一般解 (General Solution)

数値解 … 個別解 (Particular Solution 特殊解なる用語は内容を表していない)

２．J-GridSheetによる微分方程式の予備的アプローチ

前回[1]の報告ではExcelによる微分方程式の本が多数出ていることに合わせて、J-GridSheetによっても容易に出来ることを示した。しかしながら、これは微分方程式を身近かにするメリットはあるが、多量の数値を見せるだけで、必ずしも微分方程式を理解させることにはなっていない。単なる初歩の予備的なアプローチに過ぎない。

３．微分方程式の数値解とそのＪプログラム

微分方程式の初期値問題の近似解の手法には以下のようないろいろな方法がある。[2]

Euler法 … 原理的だが、精度はよくない

Heun法 … わずかの改良で、精度が改善されている

Runge-Kutta法 … 実用にたえる近似手法で、もっとも広く用いられている

Adams-Bashforth法 … 多段階の近似計算で精度を高める

いずれも、初期値と初期値における微係数から次々と方程式に合った値を求めて行く。直前の微係数だけではなく、何段階か前のものを使ったりして、近似の精度を高める。

いずれの手法もプログラミングに当たって、その基本は次のような漸化計算のアルゴリズムを繰り返しおこなう。

漸化計算

入力値 x ， y ⟶ 出力値 x ， y

以下の微分方程式を、もっとも簡単なEuler法で解く例により、Ｊプログラムを作成、改良していく過程を示してみよう。

微分方程式 $\frac{dy}{dx} = y - x,$ 初期値 $x = 0, y = 0$ を解け

初期値(x, y)を右引数とした微分方程式の関数をｆとして定義する。、次にきざみを左引数、繰り返しの初期値(x,y)を右引数として、Euler法の処理を行う関数をeulとして定義する。またここでは関数ｆはグローバルとして用いている。

```
f =: 3 : 0
'X Y' =. y.
Y - X
)
eul =: 3 : 0
:
h =. x.
'X Y' =. y.
k =. f X, Y
(X+h), (Y + h*k)
```

)

実行は以下のようにして行い、結果が得られる。

```
   0.1 eul^:(i.11) 0 0
  0          0
0.1          0
0.2      _0.01
0.3     _0.031
0.4    _0.0641
0.5   _0.11051
0.6 _0.171561
0.7 _0.248717
0.8 _0.343589
0.9 _0.457948
  1 _0.593742
```

しかし、使い易くするためには、いくつかのプログラムの改良が必要である。

まず、微分方程式を ’あらわ’ な引数の形で取り込みたいものである。そのためには、J の副詞定義を用いれば次のように ’エレガントに’ 行える。

```
NB. Euler Adverb Definition
euler0 =: 1 : 0
:
h =. x.
'X Y' =. y.
k =. u. X, Y
(X+h), (Y + h*k)
)
```

実行は次のようになる。

```
   0.1 ((1&{)-(0&{)) euler0^:(i.11) 0 0
```

このとき、副詞の引数とするためには、動詞は tacit の形でなければならず、必ずしも分かり易くはない。もっと日常の数学の式に近づけられないだろうか。

ここで J の正規表現を利用すれば、簡単に変換できることがわかった[4]。その変換プログラムは以下のとおりである。

```
df =: 3 : 0
require 'regex'
y =. y.
if. '-' = {. y do. y =. '0 ', y end.
y =. '[[:digit:]]+¥.*[[:digit:]]*' (,&'″_') rxapply y
```

```
y =. ('¥/';'%') rxrplc y
y =. ('sqrt';'%:@') rxrplc y
y =. ('sin';'1: o. ') rxrplc y
y =. ('cos';'2: o. ') rxrplc y
y =. ('tan';'3: o. ') rxrplc y
y =. ('cot';'4: o. ') rxrplc y
y =. ('exp';'^@') rxrplc y
y =. ('log';'^.@') rxrplc y
y =. ('x';'(0&{)') rxrplc y
y =. ('y';'(1&{)') rxrplc y
'(', y, ')'
)
```

そして、Euler法プログラムは以下のようになる。

```
euler =: 1 : 0
:
h =. x.
'X Y' =. y.
k =. ". u. , (":X), ',', (":Y)
(X+h), (Y + h*k)
)
```

実行は次のように通常の数式で行える。

```
0.1 (df'y-x') euler^:(i.11) 0 0
```

変換プログラム(df)のコーディングから分かるように、

割り算は／、√はsqrtとし、sin, cos, tan, cot, exp, logはそのままで良いなどずっと分かりよくなった。数値も日常のとおりで入れればよい。

付録にEuler法、Heun法、改良Euler法、Runge-Kutta法の各Jプログラムをあげたが、解析解(Math)とそれぞれの計算法の比較結果を示す。

X	Math	Euler	Heun	Ext_Euler	Runge-Kutta
0.0	0.000000	0.000000	0.000000	0.000000	0.000000
0.1	_0.005171	0.000000	_0.005000	_0.005000	_0.005171
0.2	_0.021403	_0.010000	_0.021025	_0.021025	_0.021403
0.3	_0.049859	_0.031000	_0.049233	_0.049233	_0.049858
0.4	_0.091825	_0.064100	_0.090902	_0.090902	_0.091824
0.5	_0.148721	_0.110510	_0.147447	_0.147447	_0.148721
0.6	_0.222119	_0.171561	_0.220429	_0.220429	_0.222118
0.7	_0.313753	_0.248717	_0.311574	_0.311574	_0.313752

```
0.8  _0.425541  _0.343589  _0.422789  _0.422789  _0.425540
0.9  _0.559603  _0.457948  _0.556182  _0.556182  _0.559601
1.0  _0.718282  _0.593742  _0.714081  _0.714081  _0.718280
```

４．方向場とグラフ表示およびそのＪプログラム

方向場のグラフ表示の原理は至極簡単である。まず、表示したい範囲をグリッドに区切って、その座標値を配列として得る。次に各座標における微係数を求め、これに対応した傾きの小さな矢印をその座標位置で描けばよい。

位置と傾きとを与えて矢印を描くＪのプログラムは次のようになる。

```
NB. Draw arrow symbol
NB.  size draw_arrow center slope  => eg. 1 draw_arrow _2 1 0.5
draw_arrow =: 3 : 0
:
'ax ay' =. x. arrow y.
AR =: , adj ax,.ay
wd 'glines ', '': AR
wd 'gshow'
)
adj =: %&4.0@(500&*@(4.0&+))    NB. adjust scales
arrow =: 3 : 0
:
d =. x.                         NB. size of arrow
'x0 y0 a' =. y.                 NB. position of arrow
'p0x p0y' =: (-d), 0            NB. arrow figure
'p1x p1y' =. d, 0               NB. arrow figure
'p2x p2y' =. (0.9*d), (0.1*d)   NB. arrow figure
'p3x p3y' =. (0.9*d), (_0.1*d)  NB. arrow figure
'p4x p4y' =. d, 0               NB. arrow figure
px =. p0x, p1x, p2x, p3x, p4x   NB. arrow figure
py =. p0y, p1y, p2y, p3y, p4y   NB. arrow figure
Cos =. 1 % %: 1 + *: a          NB. rotate arrow
Sin =. a % %: 1 + *: a          NB. rotate arrow
'xx yy' =. (2 2$Cos, (-Sin), Sin, Cos) (+/ . *) (px ,: py)
xx =. (x0) + xx                 NB. move arrow to position
yy =. (y0) + yy                 NB. move arrow to position
xx;yy
```

)

これを用いて、方向場をグラフ表示するＪプログラムは次のようになる。

```
NB. Direction Field Display
dfield_go_button=: 3 : 0
I =. 0.4 * (-@(i.@-) , }.@i.) 10
II =. >,{ I;I
0.2 draw_arrow"(1) (df Eq) dfunc"(1) II
)
dfunc =: 1 : 0
'x y' =. y.
z =. ". u. ,' ','"(1) ', ("":x),' ',("":y)
x, y, z
)
```

５．微分方程式・グラフィック解析システムとその実行例

微分方程式（今回は１階常微分方程式に限るが）を与えて、方向場と初期値による数値解をグラフィック表示するシステムをWindows環境で作成した。WindowsグラフィックスのＪプログラミングはこれまで何遍も報告したが、今回とくに目新しい手法は使っていない。入力する方程式の書式に、たとえば1.3*y/(sqrt(x^2)+1)のように通常の数値、変数x, y、数学関数名などで行えるようにしたが、この部分に最も苦労した。

実行は正規形（y'=f(x,y)）の微分方程式で f(x,y)部分の式を入力して、 Dir.Field ボタンを押すと方向場が描かれる。続いてInitial Position:のエディットボックスに初期値x, y の値を入れ、Part.Sol.　ボタンを押すと数値解がプロットされる。なお近似計算は初期値から正、負の両方向に行われるので、左右に伸びた曲線になる。

６．最後に

前報[1]にも述べたが、微分方程式はもっと、身近かなものでなくてはならない。コンピュータによるグラフィックの利用をもっと活用すべきである。

これまで、数学の教科書ではあまり取り上げられていないが、方向場表示は全体を見通す意味で非常に重要である。また「方向場＝一般解、初期値問題＝個別解」なる私見に対してもご批判とご意見を頂きたい。

文献

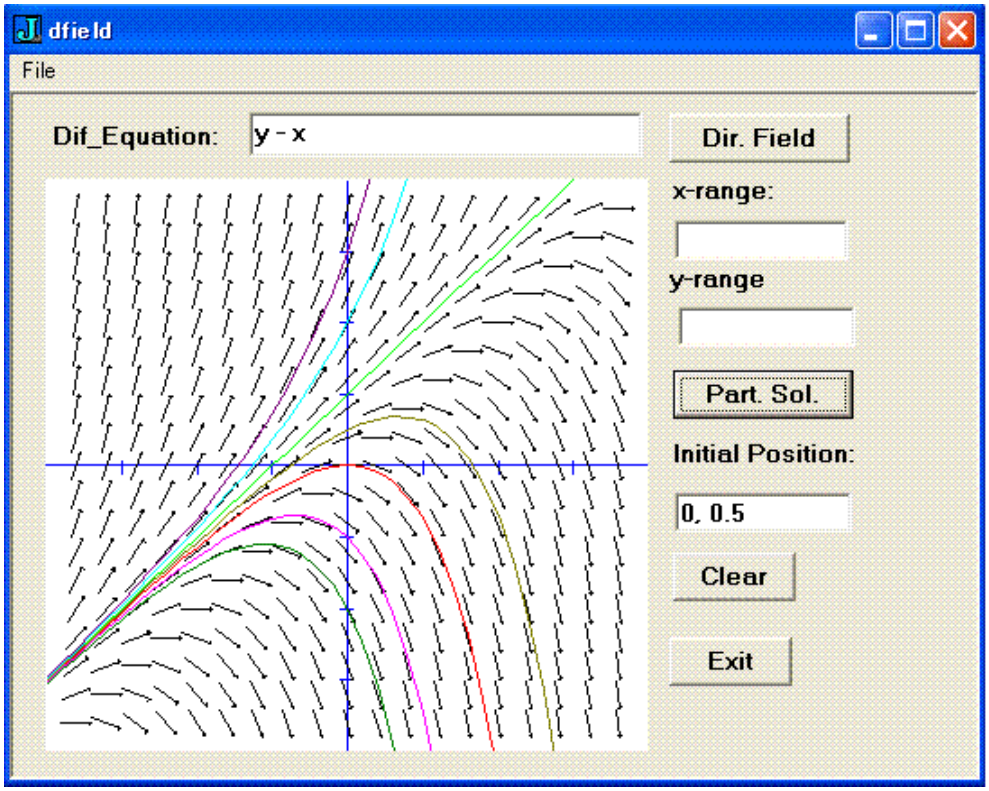
[1] 西川利男「微分方程式をAPL/J思考から理解する」Ｊ研究会資料(2006/9/20)

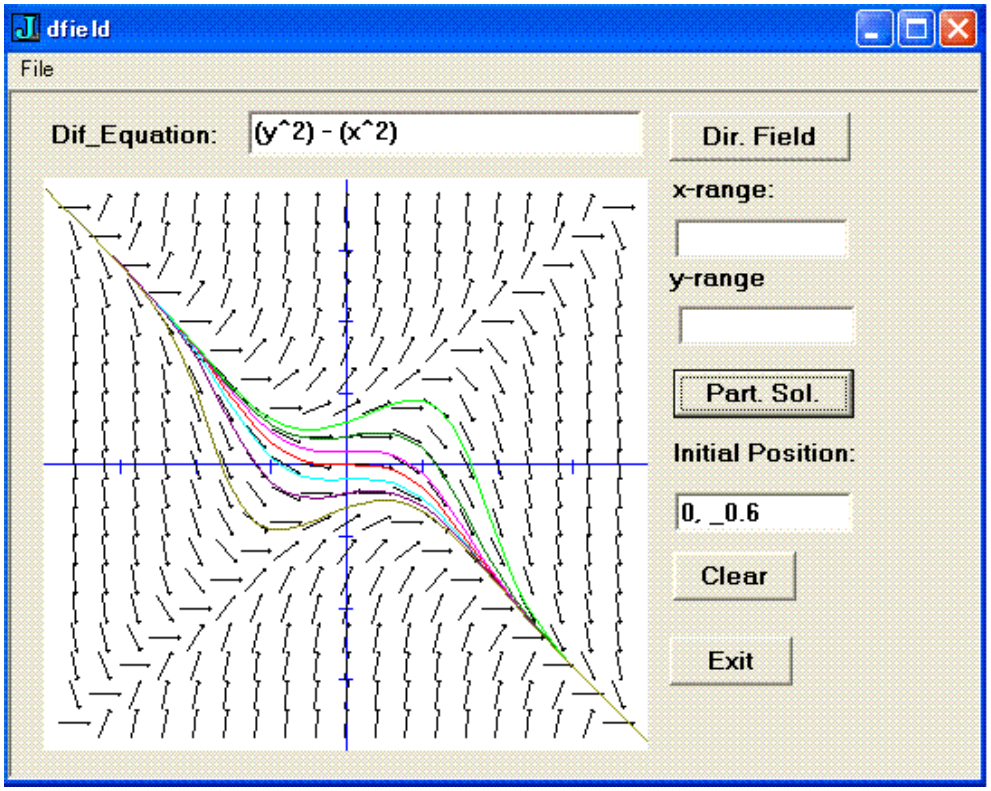
[2] 洲之内治男、寺田文行、四条忠雄「FORTRANによる演習数値計算」サイエンス社(1980)

[3] MAPLE-V ユーザ・マニュアルなど

［4］ 西川利男「Ｊの正規表現(regex)でＪの有効名をテストする」Ｊ研究会資料(2006/9/20)

実行例


dfield
File
Dif_Equation:
y - x
Dir. Field
x-range:
y-range
Part. Sol.
Initial Position:
0, 0.5
Clear
Exit


dfield
File
Dif_Equation:
(y^2) - (x^2)
Dir. Field
x-range:
y-range
Part. Sol.
Initial Position:
0, _0.6
Clear
Exit

プログラムリスト

```
NB. Differential Equation ================================
NB. Runge-Kutta argumented Dif_Equation /2006/9/5
NB.   inc (dif_eq)  rk^:(cycles) initial x, y
NB.   0.1 (df 'y-x') rk^:(i.11) 0 0
rk =: 1 : 0
:
h =. x.
x =: 0{ y.
Y =: 1{ y.
F =: u.
k1 =: ". F, ("":x),',',("":Y)
k2 =. ". F, ("":x + h%2),',',("":Y + h*k1%2)
k3 =. ". F, ("":x + h%2),',',("":Y + h*k2%2)
k4 =. ". F, ("":x + h),',',("":Y + h*k3)
(x+h), (Y + h*(k1 + (2*k2) + (2*k3) + k4)%6)
)

NB. Euler Adverb Defiition
eulerf =: 1 : 0
:
h =. x.
x =. {. y.
Y =. {: y.
(x+h), (Y + h*(u. x, Y))
)

NB. Heun Adverb Definition
heunf =: 1 : 0
:
h =. x.
x =. {. y.
Y =. {: y.
k1 =. u. x, Y
k2 =. (u. (x+h), Y) + h*k1
(x+h), (Y + h*(k1 + k2)%2)
```

```
)

NB. Extended Euler Adverb Definition
eulerxf =: 1 : 0
:
h =. x.
x =. {. y.
Y =. {: y.
k1 =. u. x, Y
k2 =. (Y + h*k1%2) - (x + h%2)
(x+h), (Y + h*k2)
)
```

```
NB. Ordinary Differential Equation(ODE)
NB. Direction Field Graphic Display 2006/9/27
NB. and DE Numerical Solution       2006/9/28

NB. Conversion 'y / x' => '(1&{) % (0&{)'
df =: 3 : 0
require 'regex'
y =. y.
if. '-' = {. y do. y =. '0 ', y end.
y =. '[[:digit:]]+¥.*[[:digit:]]*' (,&'"_') rxapply y
y =. ('¥/';'%') rxrplc y
y =. ('sqrt';'%:@') rxrplc y
y =. ('sin';'1: o. ') rxrplc y
y =. ('cos';'2: o. ') rxrplc y
y =. ('tan';'3: o. ') rxrplc y
y =. ('cot';'4: o. ') rxrplc y
y =. ('exp';'^@') rxrplc y
y =. ('log';'^.@') rxrplc y
y =. ('x';'(0&{)') rxrplc y
y =. ('y';'(1&{)') rxrplc y
'(', y, ')'
)

NB. (d '*:@y - *:@x') dfunc 2 3 => 5
dfunc =: 1 : 0
'x y' =. y.
z =. ". u. ,' ',','"(1) ', ("':x),' ',("':y)
x, y, z
)
```

```
NB. base form
DFIELD=: 0 : 0
pc dfield closeok;
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 183 113 34 12;cc clear button;cn "Clear";
xywh 182 134 34 12;cc cancel button;cn "Exit";
xywh 9 21 167 141;cc graph isigraph;
xywh 182 5 50 12;cc go button;cn "Dir. Field";
xywh 11 7 50 10;cc label static;cn "Dif_Equation:";
xywh 65 4 110 12;cc DE edit ws_border es_autohscroll;
xywh 183 68 50 12;cc sol button;cn "Part. Sol.";
xywh 183 85 78 12;cc label static;cn "Initial Position:";
xywh 183 98 50 11;cc POS edit ws_border es_autohscroll;
xywh 183 20 42 10;cc label static;cn "x-range:";
xywh 183 31 49 11;cc xrange edit ws_border es_autohscroll;
xywh 182 42 50 10;cc label static;cn "y-range";
xywh 184 52 50 11;cc yrange edit ws_border es_autohscroll;
pas 6 6;pcenter;
rem form end;
)


run =: dfield_run
dfield_run=: 3 : 0
wd DFIELD
NB. initialize form here
wd 'grgb 0 0 255'
wd 'gpen 1'
draw"(1) > (_4, 0, 4, 0);(0, _4, 0, 4)          NB. draw x-axis and y-axis
draw"(1) ((_3+i.7),._0.1),"(1) ((_3+i.7),.0.1)  NB. draw x-graduations
draw"(1) (_0.1,.(_3+i.7)),"(1) (0.1,.(_3+i.7))  NB. draw y-graduations
```

```
wd 'gshow'
icol =: 0
wd 'pshow;'
)

NB. draw collection of lines, given as x, y in array
draw =: 3 : 0
'x0 y0 x1 y1' =. y.
wd 'glines ', '': adj x0, y0, x1, y1
)



dfield_cancel_button=: 3 : 0
wd 'pclose;'
)

NB. Input Equation in Edit Box
dfield_DE_button=: 3 : 0
Eq =: DE
icol =: 0
)

NB. Color Data
COL =: 255 0 0;255 0 255;0 128 0;0 255 0;0 255 255;128 0 128;128 128 0;0 128 128

NB. DE numerical solution
dfield_sol_button=: 3 : 0
wd 'grgb ','': > (8|icol){COL
icol =: icol + 1
wd 'gpen 1'
DA =. 0.2 (df Eq) rk^:(i. >: >. 5*XRB) PX, PY
'DXA DYA' =. |: DA
DXYA =. , (XR Adj DXA),.(YR Adj DYA)
DB =. _0.2 (df Eq) rk^:(i. >: >. 5*(-XRA)) PX, PY
'DXB DYB' =. |: DB
DXYB =. , (XR Adj DXB),.(YR Adj DYB)
```

```
wd 'glines ', '': DXYB
wd 'glines ', '': DXYA
wd 'gshow'
)

NB. Input Initial Position(X, Y) in Edit Box
dfield_POS_button=: 3 : 0
'PX PY' =: '', POS
)

NB. Direction Field Display
dfield_go_button=: 3 : 0
I =. 0.4 * (-@(i.@-) , }.@i.) 10
II =: >,{ I;I
wd 'grgb 0 0 0'
wd 'gpen 1'
0.2 draw_arrow''(1) (df Eq) dfunc''(1) >,{ I;I
)

dfield_clear_button=: 3 : 0
wd 'gclear'
icol =: 0
wd 'grgb 0 0 255'
wd 'gpen 1'
draw''(1) > (_4, 0, 4, 0);(0, _4, 0, 4)                    NB. draw x-axis and y-
axis
draw''(1) ((_3+i.7),._0.1),''(1) ((_3+i.7),.0.1)  NB. draw x-graduations
draw''(1) (_0.1,.(_3+i.7)),''(1) (0.1,.(_3+i.7))  NB. draw y-graduations
wd 'gshow'
)

adj =: %&4.0@(500&*@(4.0&+))
draw_arrow =: 3 : 0
:
'ax ay' =. x. arrow y.
AR =: , adj ax,.ay
```

```
wd 'glines ', ": AR
wd 'gshow'
)

NB. Arrow Symbol
NB.  length arrow center slope
NB.   eg. 1 arrow _2 1 0.5
arrow =: 3 : 0
:
require 'plot'
d =. x.
'x0 y0 a' =. y.
NB. arrow figure
'p0x p0y' =: (-d), 0
'p1x p1y' =. d, 0
'p2x p2y' =. (0.9*d), (0.1*d)
'p3x p3y' =. (0.9*d), (_0.1*d)
'p4x p4y' =. d, 0
NB. plot (_4, 4, 4, _4, _4, p0x, p1x, p2x);(_4, _4, 4, 4, _4, p0y, p1y, p2y)
px =. p0x, p1x, p2x, p3x, p4x
py =. p0y, p1y, p2y, p3y, p4y
NB. rotate arrow
Cos =. 1 % %: 1 + *: a
Sin =. a % %: 1 + *: a
'xx yy' =. (2 2$Cos, (-Sin), Sin, Cos) (+/ . *) (px ,: py)
xx =. (x0) + xx
yy =. (y0) + yy
NB. plot (_4, 4, 4, _4, _4, xx);(_4, _4, 4, 4, _4, yy)
xx;yy
)
```