JAPLA 研究会資料 2015/10/17

極形式表示によるらせん(Spiral)のグラフィックス(続き)
フェルマーのらせん
－Ｊの plot、pd および gl2 グラフィックスプログラム－

西川　利男

JAPLA の先月の例会では、極形式のらせん(Spiral)のグラフィックスがＪの plot ルーチンにより、簡便に行えることを示した。しかしながらもう少しいろいろな Spiral グラフィックスをやろうとすると、plot だけでは無理であり、pd さらには gl2 を使用するグラフィックスプログラムが必要になる。

今回は、続きとして Fermat の Spiral と呼ばれるきれいなグラフィックパターンの作成を題材として、plot, pd, gl2 グラフィックスのプログラムについて述べる。

０．Ｊのプログラム－計算処理とユーザインターフェース

ひと口にプログラミングというが、アルゴリズムに従った計算処理部分とその結果をコンピュータ上で表現するインターフェース部分とはかなり異なっている。

①　計算処理　計算手順をＪのプリミティブを使ってプログラムする

②　表示インターフェース　Windows などＪのシステム環境へのプログラム

また、先の３つのＪのグラフィックスを比較してみると

plot　数学の関数表示を線でつなぐ。

pd　　いろいろな図形を、点や線、色などを選んで表示する。

gl2　　さらに、２つ以上のあらゆる図形を Windows グラフィックスとして表示。

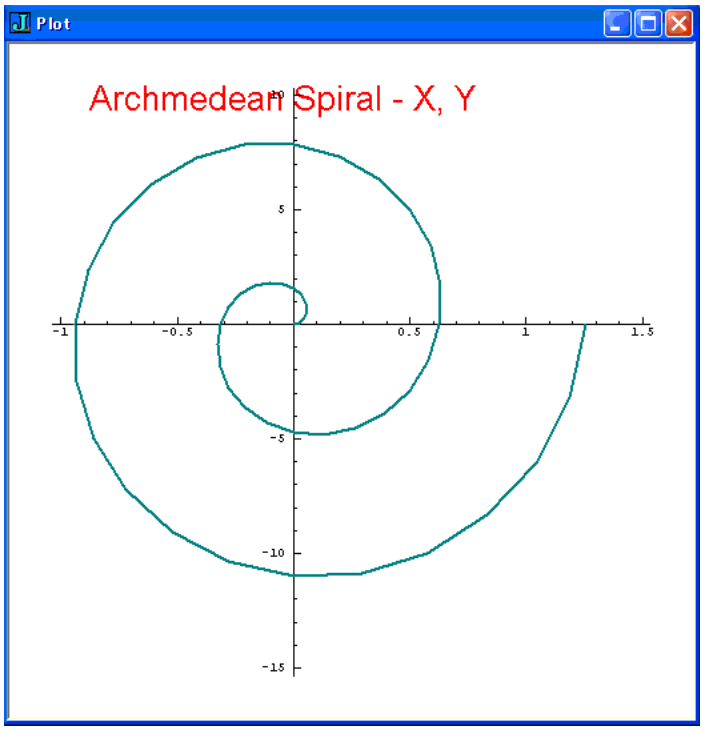
１．pd プログラミング

前回の plot によるアルキメデスのらせんのグラフィックスを、pd でプログラムしてみる。

```
load 'numeric trig plot'
archpd =: 3 : 0
pd 'reset'
pd 'aspect 1'
pd 'textcolor red'
pd 'textfont Arial 60'
pd 'textc 400 950 Archimedean Spiral'
pd 'type line'
pd 'pensize 2'
th=. (10 * i.49) * 1r120p1
a =. 1
r =. a * th
pd (0.1 * r * cos th); (r * sin th)
pd 'labelfont Courier New 60'
pd 'show'
)
```

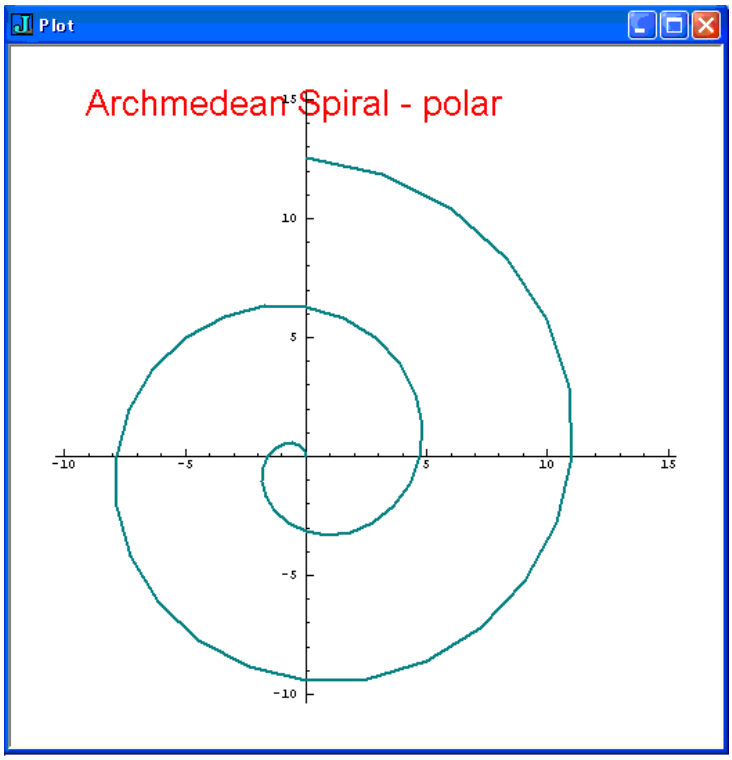
このように、pd(=plot driver の意)は plot ルーチンの中身を成すものであり、plot ルーチンは数学の関数値を手っ取り早く、１行だけで行えるよう特化したコマンドツールである。

pd のプログラム archpd を実行したものは、次のようになる。



また、プログラム archpdd で、次のように一部手直しすると、極形式のままで、表示することも出来る。

```
pd 'polar'
pd (a * th);(- th)
```



このように pd のコーディングでは長くなるが、ずっときめ細かな処理を行うことができる。

２．gl2 プログラミング

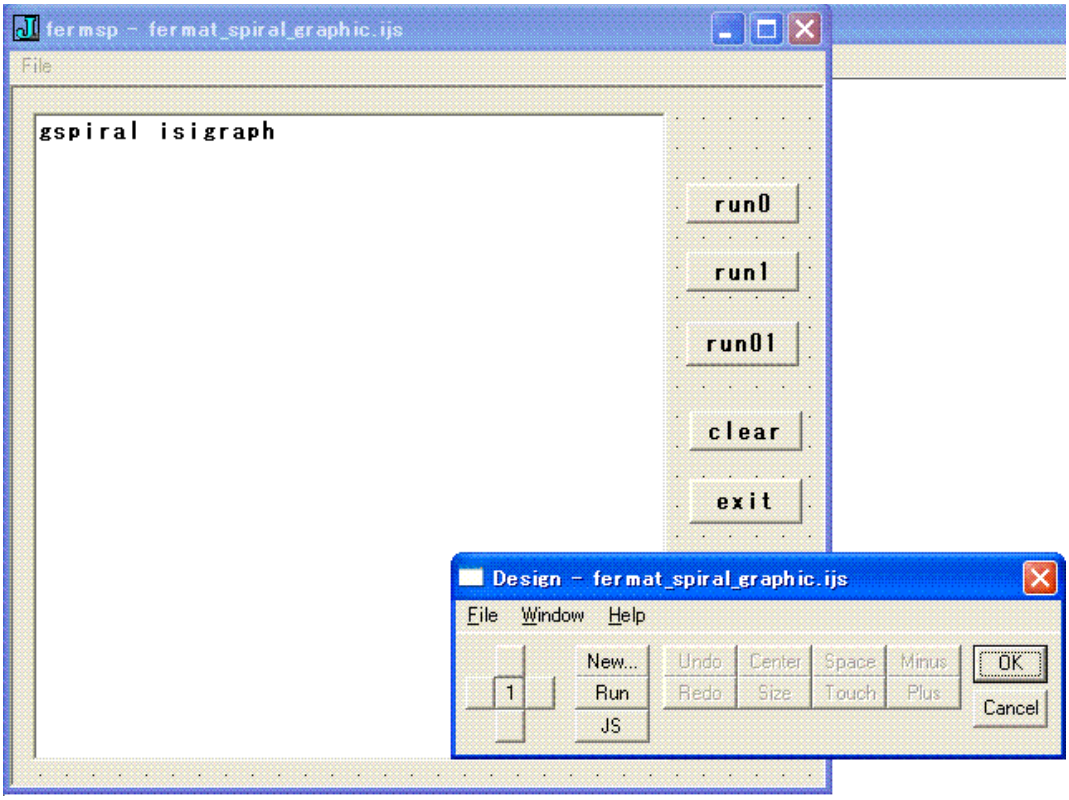
さて、いよいよ gl2 グラフィックスのプログラミングを行っていこう。
gl2 はインターフェースとして、J の Windows 環境の上で動くプログラムである。したがって、そのための J の Windows 環境、つまりフォームの設定が必要である。

これには、まずプログラム記述のスクリプト画面の中で、フォームエディタにより行う。編集ツールバーから

[Edit]-[Form Edit]

を起動する。

Form Id として、例えば fermsp のように指定すると、次のようなフォームエディタ画面から、グラフィックスとして isigraph、またいろいろな起動ボタンを配置し、さらにその実行プログラムのコーディングを行うことができる。



ここで対話的に配置した Windows のフォームは、フォームエディタを閉じると、J のスクリプトとして、そのコードは以下のように自動的に作成される。

```
load 'graph'
load 'numeric trig'

FERMSP=: 0 : 0
pc fermsp;
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
```

```
menupopz;
xywh 203 63 34 12;cc run01 button;
xywh 204 105 34 12;cc exit button;
xywh 6 7 191 174;cc gspiral isigraph;
xywh 203 26 34 11;cc run0 button;
xywh 203 44 34 11;cc run1 button;
xywh 204 87 34 11;cc clear button;
pas 6 6;pcenter;
rem form end;
)

run =: fermsp_run

fermsp_run=: 3 : 0
wd FERMSP
NB. initialize form here
wd 'pshow;'
)

fermsp_close=: 3 : 0
wd'pclose'
)

fermsp_exit_button=: 3 : 0
fermsp_close''
)
```

グラフィックスの表示は、例えばボタン run0 を押したときのイベントとして、その操作を gl2 プログラムにより記述することで、実行される。ここでは結ばないで点で表した。

```
gspiral_run0_button=: 3 : 0
ferm 148
i =. 0
while. i < #X
  do.
   glrgb 0 0 255 NB. Blue
   glbrush''
   glellipse (xyadj (i{X), (i{Y)), 10 10
   i =. i + 1
  end.
glshow''
wd'pshow'
)
```

ここで、glrgb, glbrush, glellipse, glshow などは gl2 の命令コマンドである。

ところでFermat の Spiral は、次の式で与えられる。

$r = c\sqrt{\theta}, \quad \theta = n \times 137.508°$

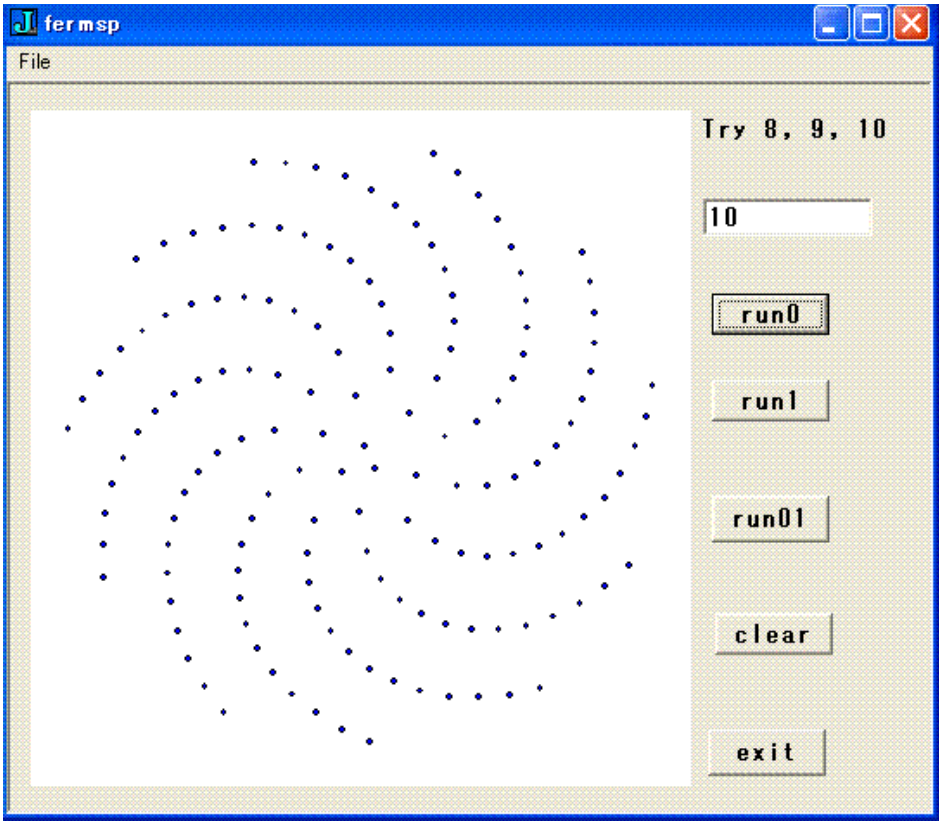
角度 θ はFibonacci 数の収束値から、H. Vogel が得たものである。[1][2]

[1] "Fermat's Spiral" from Wikipedia
[2] "Fibonnacci Number" from Wikipedia

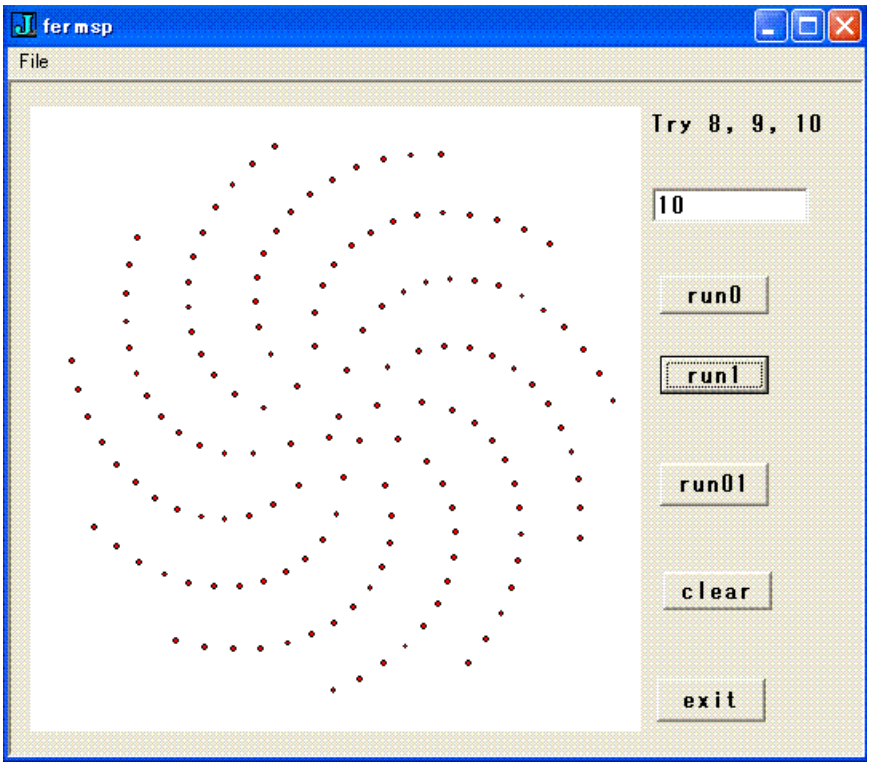
Fermat Spiral の計算およびグラフィックの大小の調整は次のようにして行う。

```
ferm =: 3 : 0
n =. y.
thd =. (10 * i. >: n) * 137.508
a =. 1
r =. a * %: thd
X =: (r * cosd thd)
Y =: (r * sind thd)
)
xyadj =: 3 : '500 + 1 * y.'
```

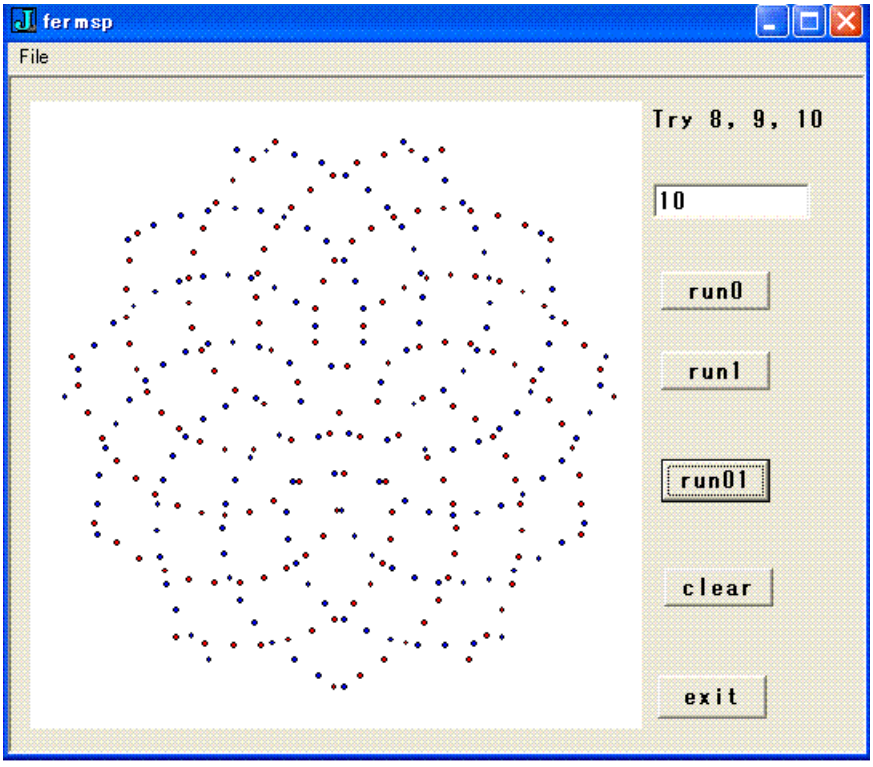
実行は、ボタン run0 を押したときには、次のようになる。



run1 では Spiral は逆向きとなる。



run01 では、２つの Spiral が重ねられ、きれいなパターンが現れる。



```
NB.  J Program Listing =============
NB.  T. Nishikawa, 2015/10/17

NB. Fermat Spiral Graphics

load 'gl2'
load 'numeric trig'

FERMSP=: 0 : 0
pc fermsp;
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 203 54 34 11;cc run0 button;
xywh 203 76 34 11;cc run1 button;
xywh 203 106 34 12;cc run01 button;
xywh 202 166 34 12;cc exit button;
xywh 6 7 191 174;cc gspiral isigraph;
xywh 204 136 34 11;cc clear button;
xywh 200 29 50 11;cc adj edit ws_border es_autohscroll;
xywh 200 8 60 10;cc adjm static;cn "Try 8, 9, 10 ...";
pas 6 6;pcenter;
rem form end;
)

run =: fermsp_run

fermsp_run=: 3 : 0
wd FERMSP
NB. initialize form here
Value =. '10'
wd 'set adj *', Value
ADJ =: ". Value
wd 'pshow;'
)

fermsp_close=: 3 : 0
wd'pclose'
)

fermsp_exit_button=: 3 : 0
fermsp_close''
```

```
)

spiral =: 3 : 0
n =. y.
th =. (10 * i. >: n) * 1r120p1
a =. 1
r =. a * th
X =: (r * cos th)
Y =: (r * sin th)
)

ferm =: 3 : 0
n =. y.
NB. ADJ is global value from adj edit input
thd =. (ADJ * i. >: n) * 137.508
a =. 1
r =. a * %: thd
X =: (r * cosd thd)
Y =: (r * sind thd)
)

NB. xyadj =: 3 : '500 + 25 * y.'
xyadj =: 3 : '500 + 1 * y.'


fermsp_run0_button=: 3 : 0
ferm 148
i =. 0
while. i < #X
  do.
   glrgb 0 0 255 NB. Blue
   glbrush''
   glellipse (xyadj (i{X), (i{Y)), 10 10
   i =. i + 1
  end.
glshow''
wd'pshow'
)

fermsp_run1_button=: 3 : 0
ferm 148
i =. 0
while. i < #X
  do.
   glrgb 255 0 0 NB. Red
```

```
   glbrush''
   glellipse (xyadj (- i{X), (i{Y)), 10 10
   i =. i + 1
  end.
glshow''
wd'pshow'
)

fermsp_run01_button=: 3 : 0
fermsp_run0_button ''
fermsp_run1_button ''
)


fermsp_clear_button=: 3 : 0
glclear ''
glshow ''
Value =. ''
wd 'set adj *', Value
ADJ =: ". Value
wd 'pshow'
)

fermsp_adj_button=: 3 : 0
ADJ =: ". adj
)
```