複素グラフィックスの世界（1）

SHIMURA Masato
jcd02773@nifty.ne.jp

2013 年 10 月 18 日

目次

オイラーの公式

一番美しい数学の公式と言われるオイラーの公式 $e^{i\theta} = cos\theta + isin\theta$ の幾何学の顔は複素数の極座標。タートルグラフィクスの描画に採用されている。
J の関数 r. は $e^{i\theta}$ を一瞬で計算し、plot のガウス・アルガン座標を用いて描くことができる。

1　円と螺旋

配列計算言語　J では数列を生成して一気に直線を描く。複素数の極座標はガウス座標上で円になるのでこれを変形して螺旋や各種の曲線に挑戦する。

1.1 円

- π を求める時の 4096 角形を描く。r. は $y = e^{i\theta}$ で、$-\pi$ から π までの区間の値を求めている。2π で一周するのでほとんど円になる。steps の最後の数 (n) の指定で任意の多角形を描くことがでる。

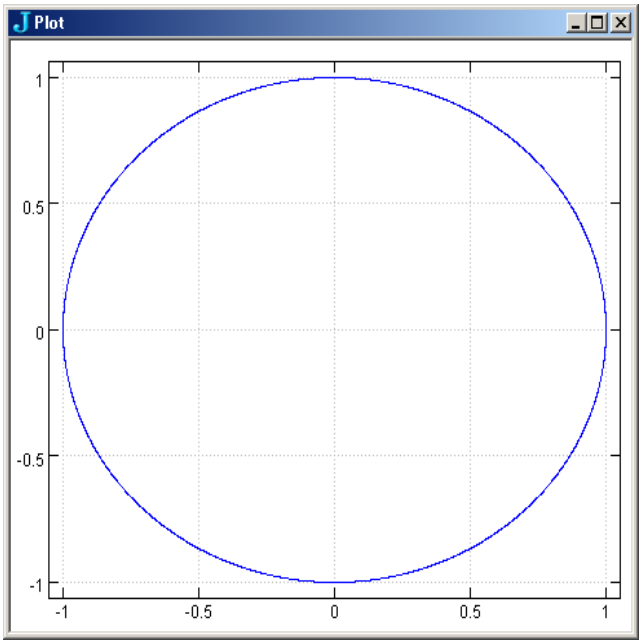
```
 require 'plot numeric trig'
plot  r. steps _1p1 1p1 4096

plot r.  t=. steps 0 2p1 100  NB. 2 pi OK
```

4096 角形はアルキメデスが π の値を求めたときに用いた
r. は Angle/Polar で複素極座標。 $r.y \leftrightarrow$ ^ j. y 即ち $e^{i\theta}$ である

```
 plot r. steps _1p1 1p1 100
   plot ^ j. steps _1p1 1p1 100
```

- xy 座標での数値を求める　$e^{i\theta} = cos\theta + isin\theta$。どちらでも可能。滑らかにするには「線は点の集合である」ので点＝刻みを多くする。

```
plot  (cos;sin) t=. steps _1p1 1p1 100
 plot r. t=. steps _1p1 1p1 100
```



1.2 螺旋

- アルキメデス螺旋

$$x = a(cos\theta + \theta sin\theta)$$

$$y = a(sin\theta - \theta cosn\theta)$$

1. プロトタイプ

```
plot ((cos + (] * sin)); (sin - (] * cos) )) t =. steps  0 10p1 1000
```

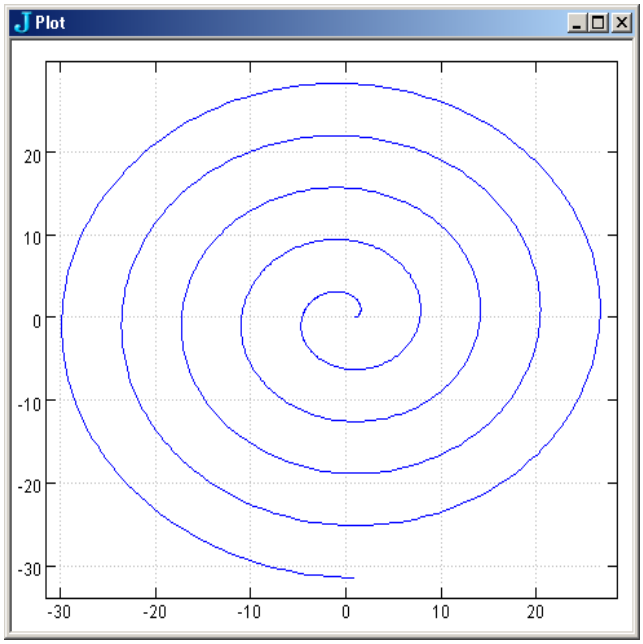
区間は 0 から 10π まで

2. もっと簡単に

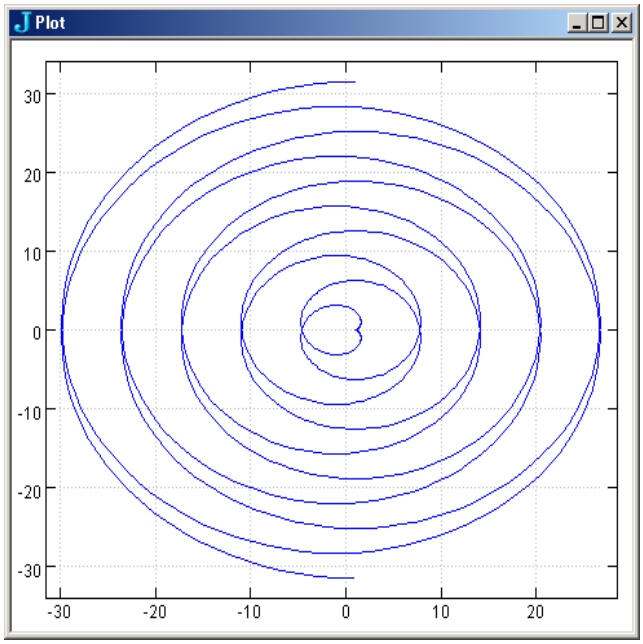
```
plot t * L:0 (cos;sin) t=. steps 0 10p1 1000
```

3. 極座標系は $r = a\theta$,　J らしく

```
plot t *  r. t=. steps 0 10p1 1000
```



-10π から 10π までとすると



- 対数螺旋　ヤコブ・ベルヌイが惚れ込んだ螺旋。最初に認識したのはデカルトのようだ。

$$r = a \cdot e^{k\theta}$$

または

$$r = a^{\theta} \qquad a > 0$$

1. プロトタイプ　螺旋の長さは steps の $n_2(60)$ を増やせば長くなる

```
plot (^ 0.06*t) * L:0 (cos;sin)t=. steps 0 60 1000
```

2. 螺旋の巻き方　0.06 → 0.1 で大分粗くなる

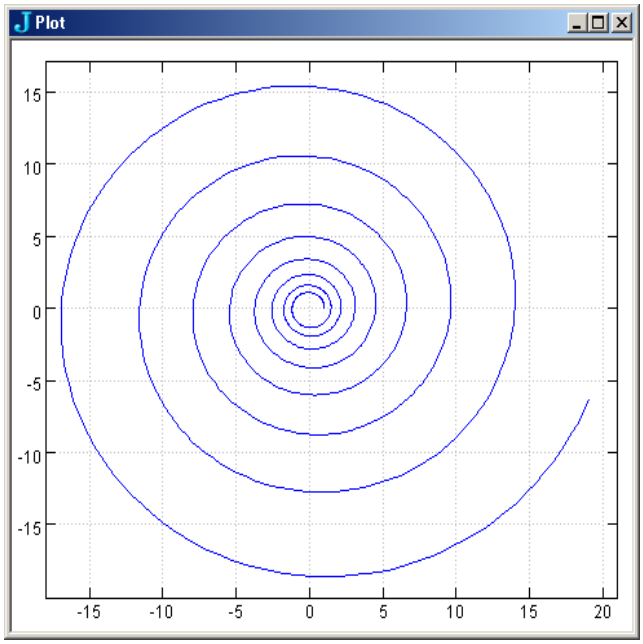
```
plot (^ 0.1*t) * L:0 (cos;sin)t=. steps 0 30 1000
```

3. J らしく

```
plot   (^ 0.1 * t)   r. t=. steps 0 30 1000
```

4. 左巻き

```
plot   (^ _0.1 * t)   r. t=. steps 0 30 1000
```



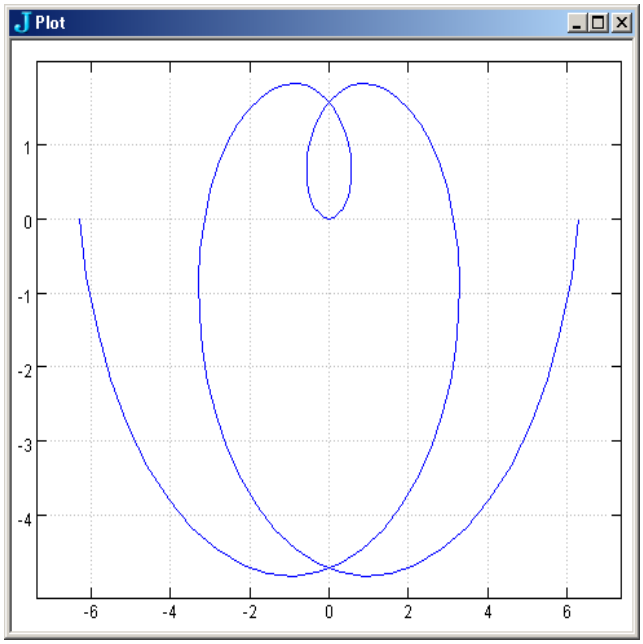
1.3　図形を探そう

スクリプトを作ってパラメーターを探し楽しい図形を描こう

- アルキメデス螺旋

```
NB. Usage: archi 0 60 1000
```

```
archi=: 3 : ' plot t * L:0 r. t=. steps y'
```



- ベルヌイ螺旋

```
NB. Usage: 0.06 ber 0 60 1000
ber=: 4 : ' plot   (^ x * t)   r. t=. steps y'

0.1 ber 0 30p1 1000
```

2　ニュートン法

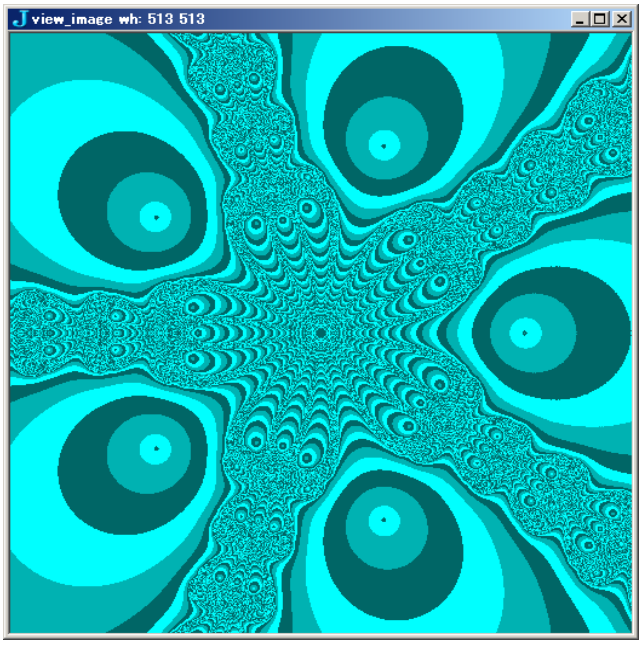
addons/media/image3/view_m.ijs をロードしておく[*1]
非線型方程式の解法として簡易で最強はニュートン法で, ラフソン版は 1690 年である。

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

根への収束列 x_n を得ることができる。次は $fx = x^5 - 2$ の場合

[*1] Run File で読み込む。　File Open で読み込んだ場合は Ctrl+W でロードする

```
view_image  newton_p1 ''
```



- ニュートン法のスクリプト

```
newt0=: 1 : '- u % u D.1  "0'
```

- J は多数とポイントでの解を一度に求められ、初期値を一々詮索することも無い。

$$f = x^5 - 2$$

```
fn1=: -2: + ^&5
```

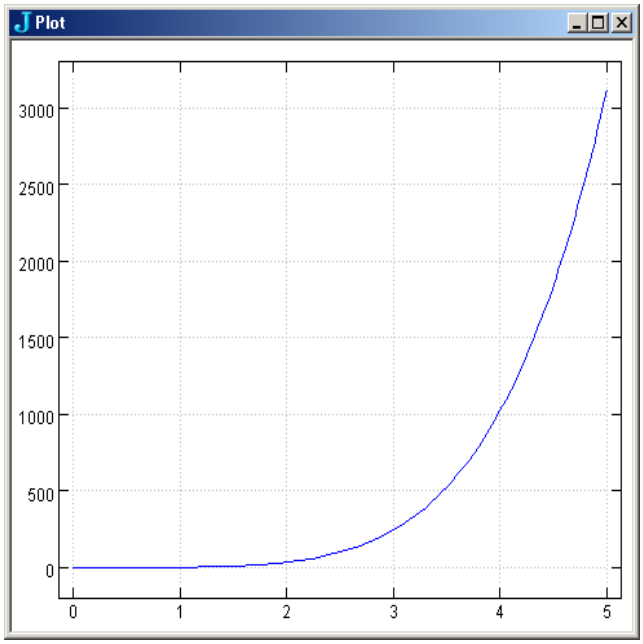
```
   fn1 newt0 ^:(i.10) 1 2 3 4 5
      1       2       3       4       5
    1.2   1.625 2.40494 3.20156 4.00064
 1.1529 1.35736 1.93591 2.56506 3.20207
1.14873 1.20373 1.57721 2.06129 2.56546
 1.1487  1.1535 1.32641 1.67119 2.06161
 1.1487 1.14874 1.19035 1.38823 1.67143
 1.1487  1.1487 1.15151 1.21828 1.38839
 1.1487  1.1487 1.14871 1.15621 1.21836
 1.1487  1.1487  1.1487  1.1488 1.15622
```

```
1.1487   1.1487   1.1487   1.1487   1.1488
```

初期値が多少離れていても強引に解は得られるが、収束には差がある。

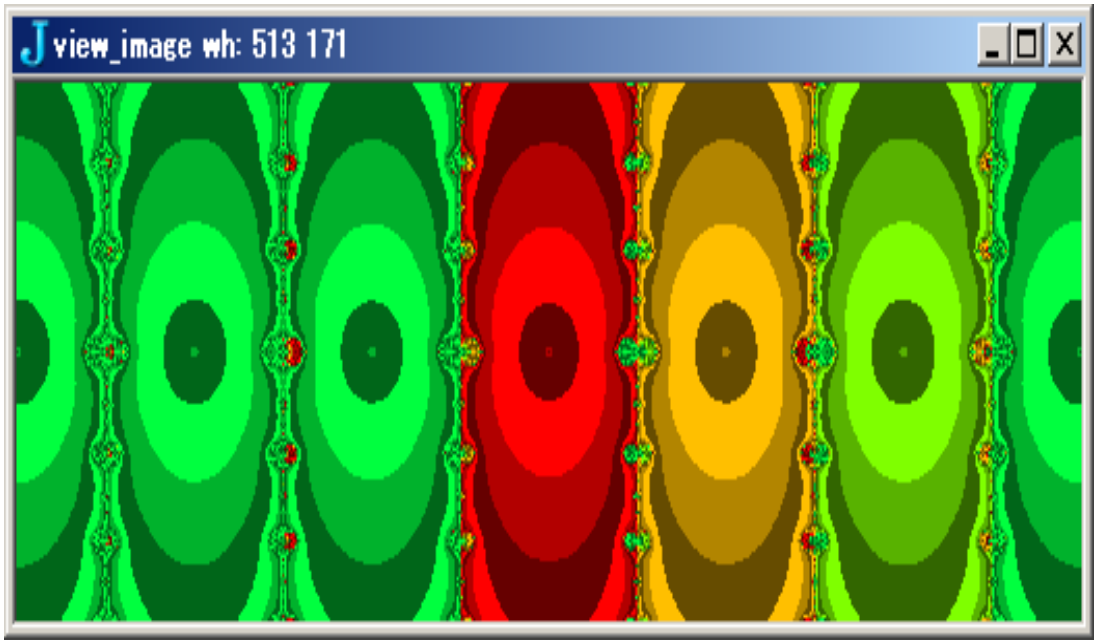
- サイエンスプロット　plot にはサイエンスプロットの機能があり、区間と関数を指定すると X,Y 軸を自動生成して描画する

  ```
  plot  0 5;'fn1'
  ```



- 関数にメッシュの布をかけ、各ポイントを初期値としてニュートン法の収束の具合を見る。フラクタル図形ができる。
- sin の場合

  ```
  view_image  newton_p0 ''
  ```



- C.Reiter のスクリプト

```
NB. -------------------------------
NB. newton method
NB. written by C.Reiter [Fractal Visualization and J 3rd.edition]
NB. slightly modified by M.Shimura
NB. Hue is in media3
Hue=:<.@(255.9999&*)@((-.,])@(1&|)+/ . *{&(#:7|3^10-i.8)@(0 1&+)@<.)@(6&*)"0

fn0=: 1&o.                  NB. sin
fn1=: _2: + ^&5             NB. x^(5) - 2
zclean=: (* * |)&. +.       NB. complex clean
newt0=: 1 : ' - u % u D.1 "0'
newton=: 1 : ' (,u newt0@{:)^:({: -.@e. }:)^:_'   NB. OK

NB. -------------------------------
NB. fn1 is sin
newtp0=: (- fn0 % fn0 D.1) :: _:
test=: 100&>@# *. 1e6&>@|@{: *. {: -.@e. }:
```

```
newtonp0=: (,zclean@newtp0@{:)^: test ^:_
newtonp_sb0=:  (# ,{:)@:newtonp0 "0  f.

newton_p0=: 3 : 0
P24=. make_p1 ''
sb=.  newtonp_sb0 512 z1_clur 1p1 * _3 3j1
's b'=: 0 1 |: sb
B=. (1p1 * i.3) i. b
bb=. 3 8 #. 0|:(3|s),: B
P24;bb
)

newtp1=: (- fn1 % fn1 D.1) :: _:
newtonp1=: (,zclean@newtp1@{:)^: test ^:_
newtonp_sb1=:  (# ,{:)@:newtonp1 "0  f.


NB. --------------------
NB. fn0 is _2:+^&5
NB. _2:+^&5
newton_p1=: 3 : 0
P18=. make_px 6
sb=.  newtonp_sb1 512 z1_cccr 0 1.75
's b'=. 0 1 |: sb
B=. (~.,b) i. b
bb=. 3 6 #. 0|:(3|s),: B
P18;bb
)

NB. Usage:  u ''
make_px=: 3 : ' <.,/ 1 0.7 0.4 */ Hue (i. % ]) y'

NB. -----------------------------------
```

```
z1_cccr=: 4 : 0  NB. OK
w=. - -/ y
({. y)+ w * ((i:% j.) +/ (i: %])) <. -: x
)

z1_clur=: 4 : 0  NB. OK
w=. -~/ 9 o. y
h=. -~/ 11 o. y
xs=. ({. y)+ w *(i.%<:) 1+x
ys=. h * (i:%j.)<. 0.5+x*h%w
ys +/ xs
)
```

view_image は addons/media/image3/view_m.ijs に入っている。

3 References

Cliff Reiter [Fractal Visualization and J 3rd Edition] Lulu 2007
H.M.S. コセクター　銀林　浩訳　「幾何学入門　上」　ちくま学芸文庫　2009
西沢清子　関口晃司　吉野邦生「フラクタルと数の世界」海文堂　1991