オイラーの美しい公式をグラフィックスにどう用いるか

SHIMURA Masato

2018 年 12 月 10 日

概要

J は複素数も原始関数でサポートしており、そのままで複素演算ができる。J の複素関数特にオイラーの美しい数式といわれる $e^{i\theta} = cos\theta + isin\theta$ の演算を行う r. の応用例とその表現を幾つかのグラフィックスで表す。

1　複素グラフィックスへの道

複素数のグラフィックス座標はヴェッセル、アルガン、ガウスが開発した。

J は複素数へも配列環が通っていて、普通の数として取り扱うことができる。更に plot は複素数をサポートしており、アルガン・ガウス座標を意識せずに描くことができる。

1.1　plot

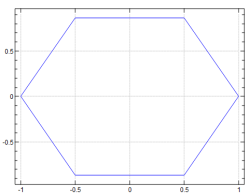
```
plot mk_polygon  r. 2p1* (i.6)%6
```

ポリゴンになるよう終点と機転を結ぶスクリプトを作成する

```
mk_polygon=: 3 : ' ({.y),~y' 
```

NB. connect for polygon

正六角形を複素座標で描いている。2π の区間を六等分し、$e^{i\theta}$ の演算をし、複素極座標を求めている



∗1

[*1] plot は図形を画面に収めるためデフォルトでは縦横比を自動調整するので図形がゆがむことがある

1.2 gl2/dwin

dwin は C.Reiter がフラクタルを描いたときに作成したツールで,addons/fvi4 に入っている。J8 にも対応しており、多少ピーキーな gl2 よりも数段使い勝手がいいので常用している。

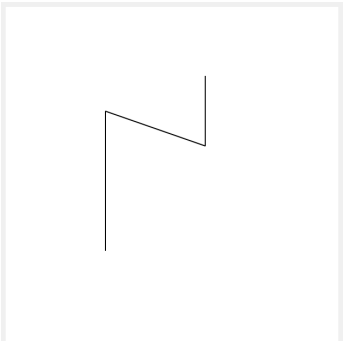
J オリジナルの gl2 は複素数をサポートしておらず、そのアドオンである dwin も同様である。

- +. で複素数を分離して xy 座標に戻す
- 自分でサポートするツールを作成する。

1.3 dwin で試し書き

- dwin でキャンバスを開く
  ```
  _5 _5 5 5 dwin 'test'
  ```
- ラインは一筆書きになる

```
dline _2 _2, _2 2,1 1,: 1 3
```
最後の連結は (,:) として縦型のデータとする



- 一本づつ引くときはボックスで
- dwin で開くキャンバスの大きさを自動計算する関数を作る。

```
minmax0=: <./,>./
find_minmax=: 3 : 'minmax0&minmax0 > minmax0&minmax0 L:0 y'

open_dwin=: 3 : 0
NB. Usage: open_dwin S10// L:0 DATH
tmp0=.>. 2 # _2 1 + find_minmax y
tmp0 dwin ''
)
```

1.4 メッシュを描く

一筆書にならないようボックスで分離する。(ループを用いない)

1. データの定義

```
NB. example
NB. horizontal
DATH=: (0 1,: 1 1);(0 0.8,: 1 0.8);(0 0.6,: 1 0.6)
DATH=: DATH,(0 0.4,: 1 0.4);(0 0.2 ,: 1 0.2);(0 0 ,: 1 0)
NB. vertical
DATV=: (0 1,: 0 0);(0.2 1,: 0.2 0);( 0.4 1,:  0.4 0)
DATV=: DATV,(0.6 1,: 0.6 0);(0.8 1 ,: 0.8 0);(1 1 ,: 1 0)
```

```
   DATH,DATV
+---+-----+-----+-----+-----+---+---+-----+-----+-----+-----+---+
|0 1|0 0.8|0 0.6|0 0.4|0 0.2|0 0|0 1|0.2 1|0.4 1|0.6 1|0.8 1|1 1|
|1 1|1 0.8|1 0.6|1 0.4|1 0.2|1 0|0 0|0.2 0|0.4 0|0.6 0|0.8 0|1 0|
+---+-----+-----+-----+-----+---+---+-----+-----+-----+-----+---+
```

2. 新しいキャンバスを準備し、メッシュを描く

```
  find_minmax DATH,DATV
0 1
   open_dwin DATH,DATV

   dline L:0 DATH,DATV
```

2 極座標グラフィックスのレッスン

2.1 タートルグラフィックス

極座標を利用したものに *Logo* で有名なタートルグラフィックスがある。*J* への適用はオーストリア人が個人で行い *J* 6 までは何の問題もなく使えた。
J8 の *QT* 版への適用は遅れていたがめどが立ったとのアナウンスがあった。

2.2 極座標への変換

XY 座標の *3 4* を複素数に組上げた後極座標に変換する。

**.* は *Length/angle* で極座標変換

```
   *.  3j4
5 0.927295
```

50zw 3j4 ↔ $\sqrt{3^2+4^2}=5$

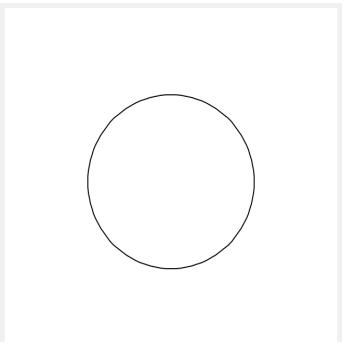
```
   arctan 4r3
0.927295
```

2.3 円と２つの渦巻

1. *dwin* で円を描く

   *(j. *.* を使わずいきなり *r.)*

```
tmp=. +. r. steps _1p1 1p1 360
open_dwin tmp
dline tmp
```



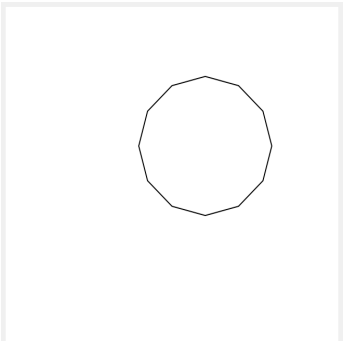
- *r=1* の単位円
- $-\pi$ から π まで 2π の区間
- *r.* を用いて極座標で $e^{i\theta}$ を計算
- *+.* で *xy* 座標に戻す
- *1* 度毎の *360* 角形を円と見做している

$$e^{i\theta} = cos\theta + isin\theta$$

とは *e* と *cos* と *sin* をマクローリン展開したとき *e* と *cos*, *sin* の合計とが複素数を介在させることで等しくなることを複素数を使いのオイラーが証明した。オイラーの美しい数式により数の理解が格段に進んだ。

2. $\theta = 30$ 度で正 *12* 角形を描く

```
tmp=: +. r. steps  _1p1  1p1  12
open_dwin ''
dline mk_polygon tmp
```



3. アルキメデス螺旋

$$r = a\theta$$

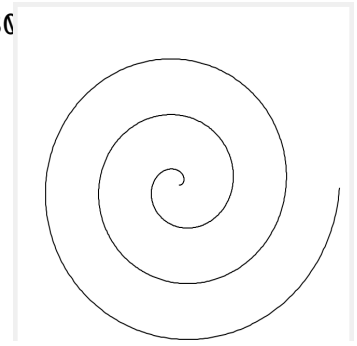
これがアルキメデス螺旋の数式であるが、この式を眺めてグラフィックスのアルゴリズムを導き出すのはなかなか難しい。かって使ったスクリプトを引っ張り出してきた。

（*a*）`archi=: 3 : 't  r. t=. steps y'`

（*b*）*r.* の両項は次の作用をする

$x\ r.\ y \leftrightarrow x{*}r.\ y$

```
tmp=. +. a r.  a=.6p1*  (i.1080
open_dwin tmp
dline  tmp
```

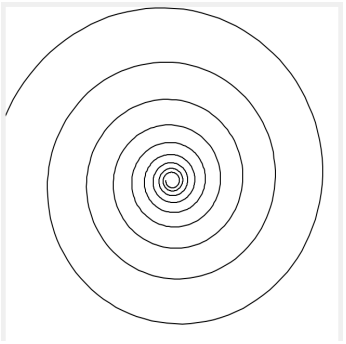


4. ベルヌイ螺旋も描いておこう

$$r = ae^{b\theta}$$

```
plot (^ 0.06* t) r.  t=: steps _9p1 9p1 1000
```

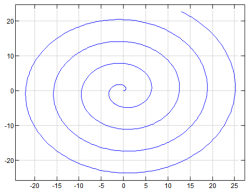
```
a=. steps  _9p1 9p1 1000
  _5 _5 5 5 dwin ''
  dline1 +. (^0.06* a ) r. a
```



2.4 plot でアルキメデス螺旋を描く

plot には左に*'polar 1'* を付けると極座標で描いてくれる機能がある。

```
'polar 1'  plot  a ; a=. 6p1* (i.360)%360
           plot  a r. a=.  6p1*  (i.360)%360
```



*2

2.5 θ とは、どう作るか

1. ラディアン
   J はラディアンが基本である。
2. *rfd* で確認 *(radian from degree , numeric.ijs* に入っている*)*
   *3

```
      rfd 0 30 45  60 90 180

      30        45      60       90     180
0 0.523599 0.785398 1.0472 1.5708 3.14159
```

*2 'polar 1' plot xx で極座標も描けるが多少バグが残っており失敗するとハングする
*3 dfr degree from radian, 同じく numeric.ijs に入っている

3. 30^o毎に 2π の区間で $\theta(rad)$ を生成する

```
    ,.  2p1 * (>:i.12)%12  NB. 30=360/12
0.523599
  1.0472
  1.5708  NB. 90
  2.0944
 2.61799
 3.14159  NB. 180
 3.66519
 4.18879
 4.71239  NB. 270
 5.23599
 5.75959
 6.28319  NB. 360
```

ラディアンは数学上の意味は分かってもグラフィックスでは使いづらい。度数はタートルグラフィックスでは根幹だが近い勝手は今一

4. ラディアンから座標の値を求める
   - *r.* で $e^{i\theta}$ を計算し、+. で複素数を分離する

```
   clean +. r.  2p1 * (>:i.12)%12
      x         y
-------------------------
 0.866025       0.5
      0.5  0.866025
        0         1  NB. 90
     _0.5  0.866025
_0.866025       0.5
       _1         0  NB. 180
_0.866025      _0.5
     _0.5 _0.866025
        0        _1
      0.5 _0.866025
 0.866025      _0.5
        1         0
```

   - $e^{i\theta}$ は座標の値を求める計算！

$$e^{i\theta} = cos\theta + sini\theta$$

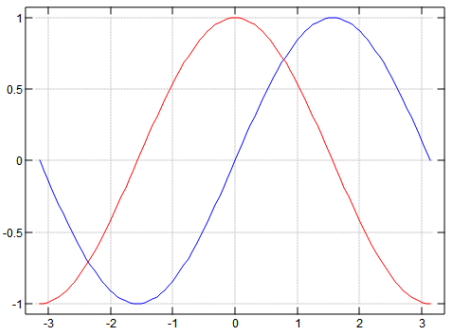
この + は片方が複素数なので足算ではなく *j.* の機能、すなわち複素数 $a + bi$ を組上げる。これでガウス座標でもデカルト座標でも座標計算ができている。
計算は $e^{i\theta}$ の方が簡単なので *r.* の方で行う。

- *cos*, *sin* で計算

```
   |: clean >(cos;sin)(L:0)  3{. 2p1 * (>:i.12)%12
0.866025      0.5
     0.5 0.866025
       0        1
```

5. サイエンスプロットで $-1\pi \rightarrow \pi$ の区間の *cos,sin* の値をを描いてみる

```
plot _1p1 1p1 ;'sin,cos'
```



3　曲線に図形をのせる

3.1　メッシュを円にのせる

単位円をメッシュに合わせ *5* 倍した円周上に 30^o 毎にメッシュを乗せる

- メッシュデータを複素数に変換し、*12* 個コピーする（ループは用いない）

```
ac=:mk_complex L:0 a=: DATH,DATV
a12=:  >12# <ac
```

- 30^o のポイントデータを取得し、*5* 倍の延ばす

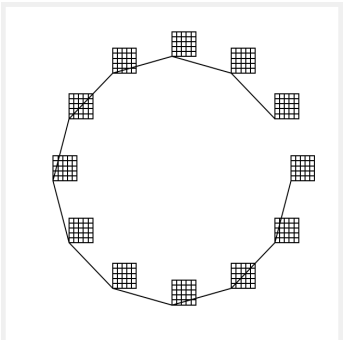
```
b=: r. 2p1 * (>:i.12)%12
```

- 双方を足し合わせる

```
c=:  a12 +"1 L:0 {@>5* b
```

- 描く

```
open_dwin c
dline L:0 +. L:0 c   NB. Mesh
dline +. 5* b         NB. Circle
```



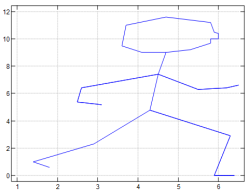
4　スティックラー博士

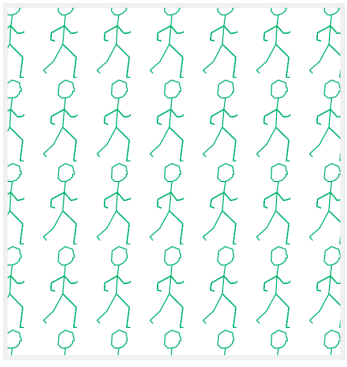
1. さて、「インドラの真珠」で活躍するスティックラー博士の登場です

```
open_dwin S10
dline S10
```

2. 江戸小紋ツール *hokusai* で小紋風に描くと

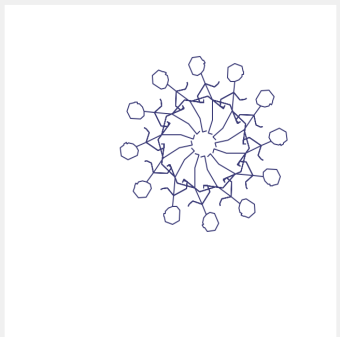
```
0 181 110 hokusai_stick 10 10
```





3. 先のメッシュと同じ手法で描く。*SCI* はスティックラー博士を *1/10* に縮小して複素数に変換したもの

```
b0=:  2p1 * (>:i.12)%12   NB. 30 度毎 12 ポイント
( r. b0) spiral_dline SC1
```

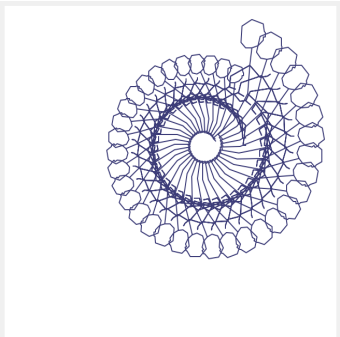


4. 曲線に乗せるスクリプト

```
spiral_dline=: 4 : 0  NB. OK
NB. y is complx
NB. (bernulle 0 2p1 36) spiral_dline SC1
tmp0=.  +. L:0 y * L:0 {@> x
open_dwin tmp0
60 60 120 dline L:0 tmp0  NB. RGB
)
```

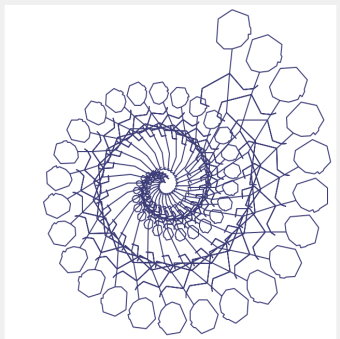
5. ベルヌイ螺旋に乗せる

```
(bernulle 0 2p1 36) spiral_dline SC1
```



6. アルキメデス螺旋に乗せる

```
 (archi 1 4p1 48) spiral_dline SC1 +-
```



5　メビウス変換

$$f(z) = \frac{az + b}{cz + d} \qquad ad - bc \neq 0$$

付録 A　J のグラフィックス環境の準備

A.1　キャンバスの用意

plot ではなく、*gl2* 系を用いる。

J の正規のグラフィックス *gl2* よりも *addons* に入っている *C.Reiter* の *dwin* の方が数段使いやすいのでこちらにする。

```
require 'plot trig numeric gl2 png '
coinsert 'jgl2'  NB. object gl2
load '~addons/graphics/fvj4/dwin.ijs'
```

1. キャンバスの構成（１）最初に *PC* 上に出す画面の大きさを指定する。（ソースコードに書けばよい）

   ```
   WIN_WH=: 640 640
   ```

2. キャンバスの構成（２）画面上に描く座標の範囲を左引数で指定する（左下:右上）次でキャンバスがポップアップする

   ```
   _5 _5 5 5  dwin ''
   dline 0 0 ,: 100 100
   ```