J で複素数の反転を

SHIMURA Masato
jcd02773@nifty.ne.jp

2019 年 9 月 13 日

目次

はじめに

大学入試から線形数学が追い出され、複素数が戻っている。この 2 つはペアになっており、どちらかが出入りする。情報教育をうたいながらいきなり大学で線形数学を一から教えろと言われれば大学の先生も面食らうが、複素数とて一筋縄ではいかない。

グラフィックスの世界では複素数は実に複雑な挙動をするが、学習の場では全て切り落としてあたかも整然と数式で処理をしている。

1 複素数の極表示と J の関数

大学入試で複素数の反転問題は頻出問題とされる。ポアンカレディスクの基礎となっているので丁寧に追ってみよう。

最初の作業

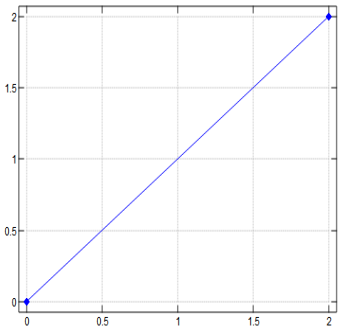
```
require 'plot numeric trig png'
```

$2 + 2i$

$$z = r(cos\theta + isin\theta)$$

1. 複素数生成

```
   2j2
2j2
```



2. 複素数の絶対値はピタゴレアンの斜辺 = r

```
   | 2j2
2.82843
```

3. r. Angle $e^{i\theta} = cos\theta + isin\theta$

```
   r. 2j2
_0.0563193j0.12306
```

4. 複素数の分離 (real/image)

```
  +. r. 2j2
_0.0563193 0.12306
```

5. 絶対値 r と偏角 θ

```
*. ( LengthAngle)/
```

$\sqrt{8} = 2\sqrt{2}$, 1r4p1=$\frac{1}{4}\pi$

```
   *. 2j2
2.82843 0.785398
```

6. 共役 (+ conjugate)

```
   2j2
2j2
```

```
  + 2j2
2j_2
```

移動、拡大縮小、回転は 3×3 の斉時変換で済ませている。これは複素数でも可能であり、複素数には更に逆数による反転機能がある。

7. 移動、拡大 · 縮小、回転
   - 平行移動 → 足し算
   - 回転/拡大 → かけ算
   - 反転/折り返し → 逆数

2　複素数の反転

$\omega = \dfrac{1}{z} = \dfrac{1}{r}(cos\theta + isin\theta)$
$z = r(cos\theta + isin\theta)$ に対して、
$w = \dfrac{1}{z}$ を対応させると、$\dfrac{1}{r}cos(-\theta) + i(-\ sin\theta)$ である
$\dfrac{1}{z}$ は極形式では，絶対値を逆数にして偏角を -1 倍するという変換

```
   *. 2j_2 2j2
2.82843 _0.785398
2.82843  0.785398

   *. %   2j2
0.353553 _0.785398
```

逆数で偏角は反転されている
$\dfrac{1}{z}$ は原点に関する反転と x 軸に関する折り返しの合成。

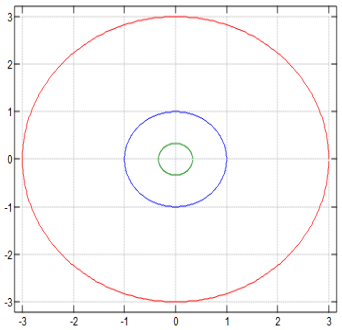
$$z^{-1} = \frac{z^*}{|z|^2}$$

$$|z|^2 = zz^*$$

幾つかの実例と図

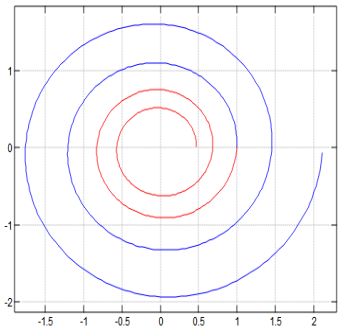
1. 複素領域での拡大と縮小
   基準円 $r = 1$ と拡大 $r = 3$, 縮小 $r = \dfrac{1}{3}$

```
plot_circle1=: 3 : 0
NB. Usage: u ''
pd 'new'
t=. 2p1 * (i.360)%360
pd r. t
pd  3* r.  t
pd  1r3*  r. t
pd 'show'
)
```



2. 対数螺旋
   反転は赤で中心に向かって縮小

```
plot_circle0=: 3 : 0
NB. Usage: u ''
pd 'new'
tx=.(^0.06*t)r. t=.4p1 * (i.360)%3
pd tx
pd  % tx
pd 'show'
)
```



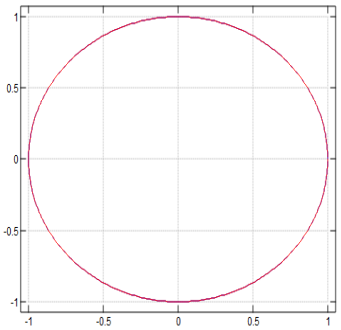
3. 基準円の反転

$$\omega = \frac{1}{z} = \frac{1}{r}e^{-i\theta} = \frac{1}{r}(cos\theta - isin\theta)$$

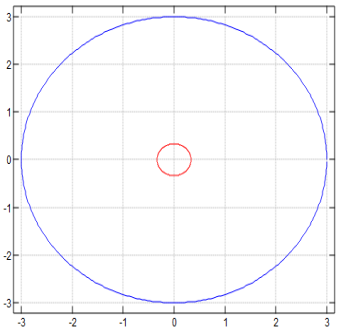
$$r = |ajb| = \sqrt{x^2 + y^2}, \theta = tan^{-1}\frac{y}{x}$$

単位円では同じ図を上書きする

```
plot_circle2=: 3 : 0
NB. Usage: u ''
pd 'new'
t=: 2p1 * (i.360)%360
pd   r.  t
pd   %   r.  t
pd 'show'
)
```

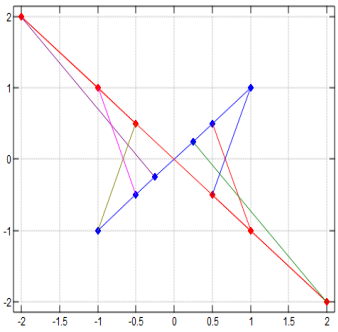


4. 基準円の 3 倍の円とその反転
   問題なく反転されている



5. 基準円を離れると複雑な挙動を取る

```
 plot_circle3 ''
  plot_circle3=: 3 : 0
NB. Usage: u ''
pd 'new'
pd 'type line marker'
t=:  1j1 0.5j0.5 0.25j0.25 _0.25j_0.25 _0.5j_0.5 _1j_1
pd     t
pd   %  t
pd 'type line'
pd t ,. % t
pd 'show'
)
```



3　複素数の挙動を概観する

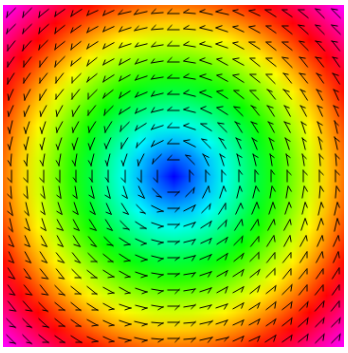
3.1　viewmat と複素数

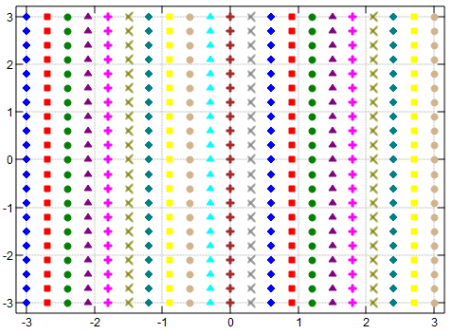
J の viewmat は複素数をサポートし方向をベクトルで示してくれる優れものだが、スケールは表示されない。

1. 準備

```
reauire 'viewmat'
  ] a0=. steps _3 3 20
_3 _2.7 _2.4 _2.1 _1.8 _1.5 _1.2 _0.9 _0.6 _0.3
0 0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4 2.7 3
```

viewmat a0 j./ a0



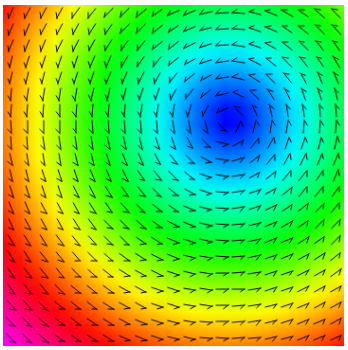


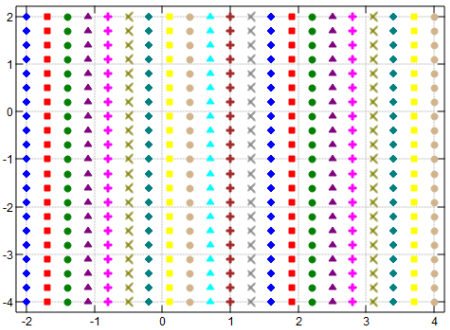
```
'marker' plot a0 j./ a0
```

2. + 1j_1

```
viewmat 1j_1+ a0 j./ a0
```

```
'marker' plot 1j_1 +  a0 j./ a0
```

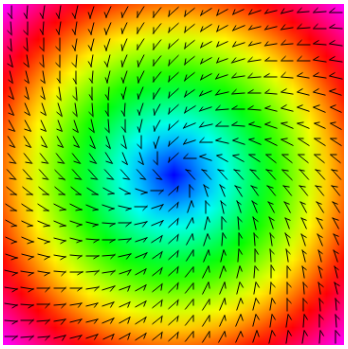


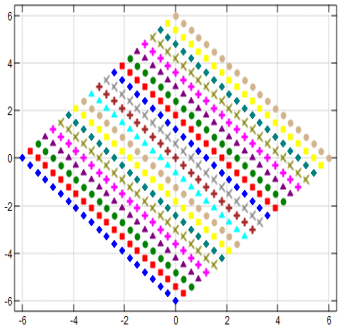


3. * 1j1

```
viewmat 1j1 * a0 j./ a0
```

```
'marker' plot 1j1 * a0 j./ a0
```





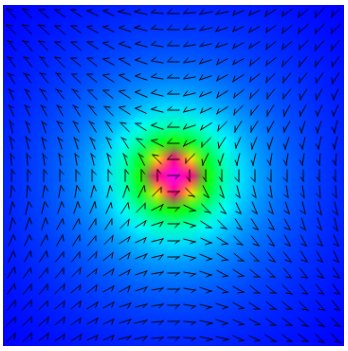
4. % 逆数

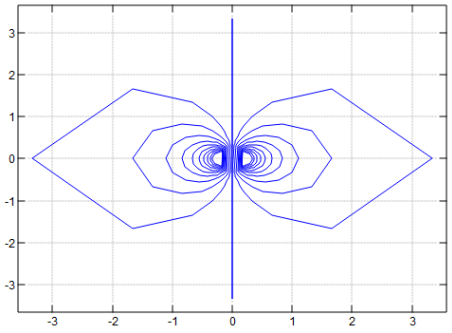
$$f(z) = \frac{1}{z}$$

```
viewmat  % a0 j./ a0
```

グラフィックスは無限大が入ると落ちるので無限大のデータを振るい落す。

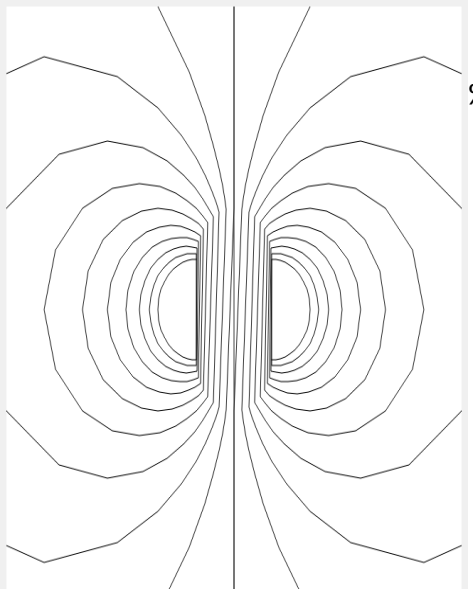
```
a0=: steps _3 3 20
plot  remove_infinity % a0 j./ a0
```






```
remove_infinity=: 3 : 0
NB. u y
index=.    -. tmp=:  ( ; y) e. _
1!:2&2 I. tmp       NB. write number
index # ; y
)
```

5. C.Reiter の Dwin を取り出して描いてみる。

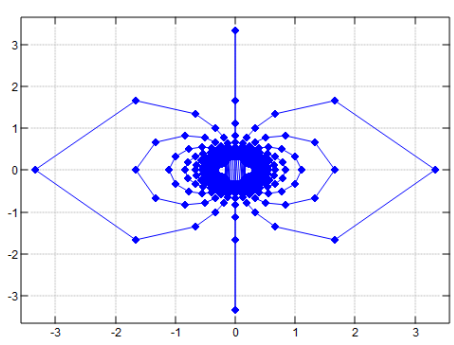


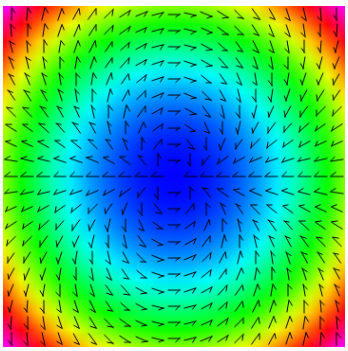
```
% a0 j./ a0
```

3.2 Conformal mapping

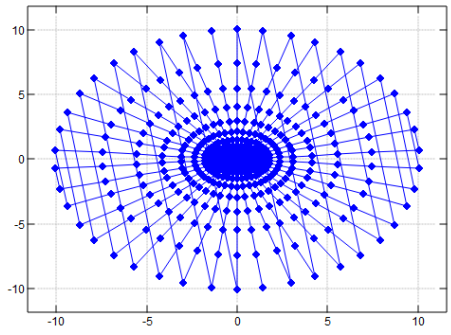
Wolfram の Mathworld では $\frac{1}{z}$ を駆使したグラフィックスを *Conformal Mapping* として一項を設けている。等角写像と訳され、流体力学や測量で活躍しているが、一般にはなじみが薄い。

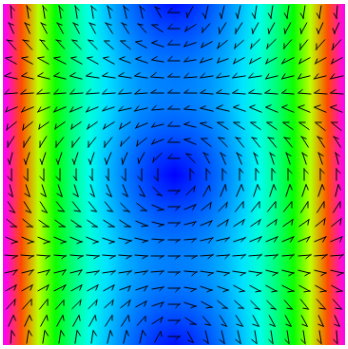
1. $\frac{z^2}{2}$



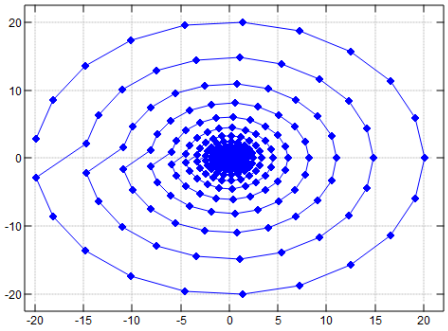


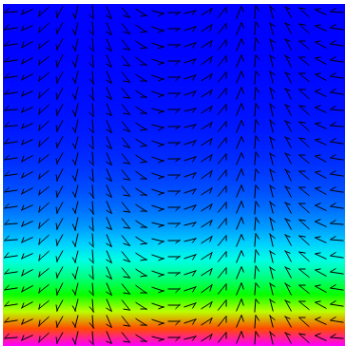
2. $sin(z)$





3. e^z





4　マンデルブローと集合とジュリア集合

ここからは複素力学系と言われる領域。

4.1　マンデルブロート集合

マンデルブロートが晩年日本数学教育学会の招きで夫人と共に来日した時に講演を聞く機会があった。

第二次大戦ではフランスで活躍していた数学者の叔父を頼ってポーランドからナチス支配下のフランスに逃れて生き延び、*US・IBM* に長く在籍した。

大戦終結後は少しゆったりしたエコール・ノルマルに進み、数学者の叔父の研究室のごみ箱にあった本から研究テーマを選ぶなど、わが道派。

1. 基本フォーミュラ

$$\begin{cases} z_{n+1} = z_n^2 + C \quad (n = 0, 1, 2 \cdots) \\ z_0 = 0 \end{cases}$$

2. z_n が無限大に発散しない箇所を図で表したものがマンデルブロート集合である。
特異点は *0* である

$c = 2$

$$\begin{aligned} z_1 &= 0^2 + 2 &&= 2 \\ z_2 &= 2^2 + 2 &&= 6 \\ z_3 &= 6^2 + 2 &&= 38 \\ z_4 &= 38^2 + 2 &&= 1446 \\ &\cdots \end{aligned}$$

$c = 1j1$

$$\begin{aligned} z_1 &= 0^2 + 1j1 &&= 1j1 \\ z_2 &= 1j1^2 + 1j1 &&= 1j3 \\ z_3 &= 1j3^2 + 1j1 &&= -7j7 \\ z_4 &= -7j7^2 + 1j1 &&= 1j - 97 \\ z_5 &= 1j - 97^2 + 1j1 &&= -9407j - 193 \\ &\cdots \end{aligned}$$

$c = 0.1$			$c = 0.2j1$		
.			.		
$z_1 =$	$0^2 + 0.1$	$= 0.1$	$z_1 =$	$0^2 + 0.2j1$	$= 0.2j1$
$z_2 =$	$0.1^2 + 0.1$	$= 0.11$	$z_2 =$	$0.2j1^2 + 0.2j1$	$= -0.76j1.4$
$z_3 =$	$0.11^2 + 0.1$	$= 0.1121$	$z_3 =$	$-0.76j1.4^2 + 0.2j1$	$= -1.1824j - 1.128$
$z_4 =$	$0.1121^2 + 0.1$	$= 0.112566$	$z_4 =$	$-1.1824j - 1.128 + 0.2j1$	$= 0.325686j3.66749$
$\cdots$			$\cdots$		

3. *comformal mapping* のようにカーペットを敷き詰めた各点を計算して発散するかどうか見極め、発散しない点を集めたのがマンデルブロート集合である。

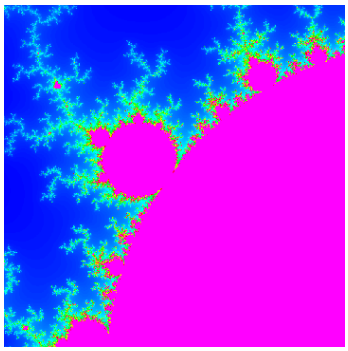
4.2 マンデルブロート集合のグラフィックス

マンデルブロート集合とジュリア集合は *C.Reiter[Fractal Visualization and J]3rd.Edition* のスクリプトを愛用している。

ここからの *Script* は `jullia_1.ijs` となる。観賞用に色分けしてあるので複雑である

```
 viewmat  mandelt  500 z1_cccr _0.2j0.8 _0.15j0.8
```

（時間がかかる）



1. $C + z^2$

```
   f=. +&1j1@*:

   (+&1j1@*:) 1j1 _1.5
1j3 3.25j1

   f ^:(i.5) 1j1
1j1 1j3 _7j7 1j_97 _9407j_193
```

2. カーペットの作成
   中心とメッシュの幅を与える

```
     5 z1_cccr 1j1  0.2j0.8
  2j0.4 1.6j0.3 1.2j0.2 0.8j0.1     0.4
1.9j0.8 1.5j0.7 1.1j0.6 0.7j0.5 0.3j0.4
1.8j1.2 1.4j1.1     1j1 0.6j0.9 0.2j0.8
1.7j1.6 1.3j1.5 0.9j1.4 0.5j1.3 0.1j1.2
  1.6j2 1.2j1.9 0.8j1.8 0.4j1.7   0j1.6

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

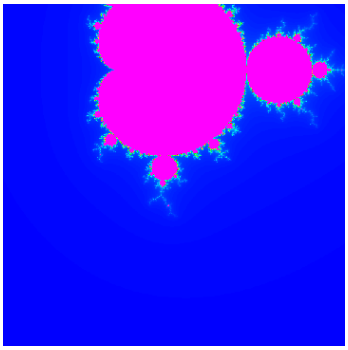
3. カラーコード生成

```
escapetc-: 2 : '#@((,u@{: )^:(<&({: n)@# *. (<&({. n)@|@:{: ))^:_) f. ' "0
NB.  viewmat 用のカラーコードを打ち出し
```

4. メインプログラム

```
mandelt=: (3 : 'y&+@*: escapetc (10 255) 0')"0
mandelt2=: (3 : 'y&+@(^&3) escapetc (10 255) 0')"0 NB. z^3
NB. Usage: viewmat  mandelt  500 z1_cccr _0.2j0.8 2j0.8
```

- `viewmat  mandelt  500 z1_cccr _0.2j0.8 2j0.8`
- 左引数の *500* はカーペットのサイズ 500×500。
- *15* とすると 15×15 のマトリクスの集合
- 右引数はエリア



5. 別のカーペットでも問題ない。*fmx* の引数はセンターと画像サイズ

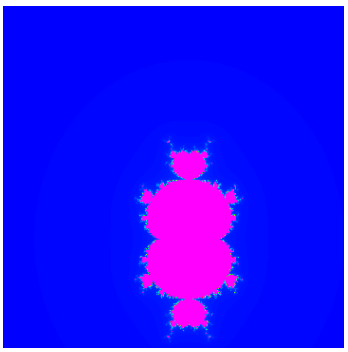
```
viewmat mandelt fmx _2j_1.5  256
viewmat mandelt2 fmx _2j_1.5  256
```

4.3 マンデルブローと集合での関数の変化と反転

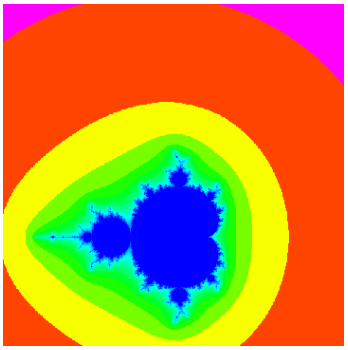
1. *3* 乗

$$z_{n+1} = z_n^3 + C$$

```
mandelt2=: (3 : 'y&+@(^&3) escapetc (10 255) 0')"0 NB. z^3
```

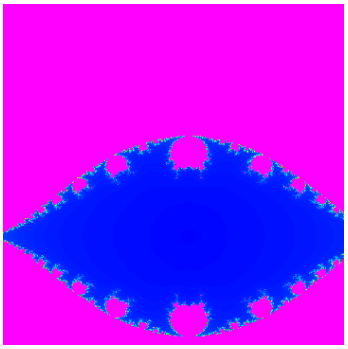


2. 反転
   出来上がったマンデルの図を反転すると次になる
   `viewmat % mandelt  500 z1_cccr _0.2j0.8 2j0.8`



カーペットをひっくり返してからマンデルをかけると次になる

```
viewmat  mandelt2 % 500 z1_cccr _0.2j0.8 2j0.8
```



4.4 ジュリア集合

8 元数を創成したケイリー卿が複素数を初期値とするニュートン法の解法に出した懸賞問題。第一次大戦を挟みファトゥとジュリアが解いた。

ジュリア集合にグラフィックス系が参入してくると数学部分を抜いて絵だけ描ければいいという事例が多くなっている。

1. ニュートン法/ニュートンラフソン法
   - 古代メソポタミアから *2000* 年を経て蘇った反復法。ニュートンはメモを残しただけで、洗練した形にまとめたのはラフソン
   - 数式

$$x_{n+1} = z_n + \frac{P(x_n)}{P'(x_n)}$$

   - メソッド

$$
\begin{aligned}
x_1 &= x_0 - \frac{P(x_0)}{P'(x_0)} \\
x_2 &= x_1 - \frac{P(x_1)}{P'(x_1)} \\
x_3 &= x_2 - \frac{P(x_2)}{P'(x_2)} \\
&\vdots
\end{aligned}
$$

- $f = x^2 - 10$ をニュートン・ラフソン法の公式に入れる

$$
x - \frac{x^2 - 10}{2x} = \frac{2x^2 - x^2 + 10}{2x} = \frac{x^2 + 10}{2x} = \frac{1}{2}\left(x + \frac{10}{x}\right)
$$

- *Script* ほとんど *J* のイディオムになっている

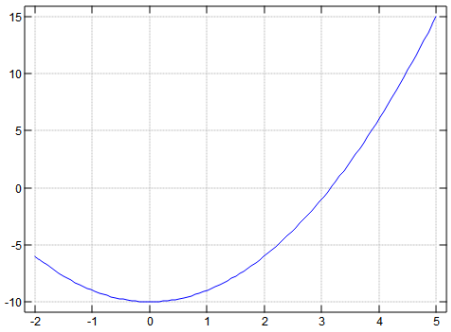
```
new_1=: 1 : ' ] - x % x D.1' (^:_)("0)
```

- 実行例（実数）

```
      _10 0 1&p. new_1 >:i.5
3.16228 3.16228 3.16228 3.16228 3.16228
```

- *Grammer*
  - `_10 0 1&p.` は $f = -10 + x^2$
  - *J* では数直線上の指定範囲の初期値を沢山与えて計算してしまうのであまり初期値を気にしなくともよい
  - *D.1* は一回微分するプリミティブ
  - *0* のポイントではエラーとなる（*i:5* は使えない）
  - `^:_` は収束まで計算する。エラーになりやすいので `(^:100)` などとしてもよい

  ニュートン・ラフソン法は非線形方程式の数値解法で初期値が一定の範囲にあると素早く収束する
- 図で確認



2. 尤もニュートン法を持ち出さなくとも多項式の解は求まるし、初期値は不要。

```
      p. _10 0 1
+-+----------------+
|1|3.16228 _3.16228|
+-+----------------+
```

3. 複素数を初期値とするニュートン法の解

```
       _10 0 1&p. new_1 3j4 3j3 3j2 3j1 3j0 2j2
3.16228j3.08149e_33 3.16228 3.16228j_4.93038e_32 3.16228 3.16228 3.16228
```

ケイリー卿も変なことを考えたものだ。
ジュリアは父の出稼ぎ先のアルジェリア生まれ。奨学金を得てエコール・ノルマルに学ぶ。第一次大戦に従軍し（歩兵少尉）顔面に銃弾を受けた。エコール・ノルマルやポリテクニークで教えた。

4. *J Grammer*

```
new_1=: 1 : ' ] - x % x D.1' (^:_)("0)
```

- *1 : 0* は副詞
- 引数に動詞/関数をとる。通常は関数は u v　名詞は \verb*m* n/を用いる
- *new_1* のように他に何もない場合は *x* でも通る。
- *D.1* は微分 （*d.1* もある）

```
new_1=: 1 : ' ] - x % x D.1' (^:_)("0)
new_2=: 1 : ' ] - u % u D.1' (^:_)("0)
      _10 0 1&p. new_1 >:i.5
3.16228 3.16228 3.16228 3.16228 3.16228

   _10 0 1&p. new_2 >:i.5
3.16228 3.16228 3.16228 3.16228 3.16228
```

- *2 : 0* は接続詞　u v m n を複雑に組み合わせるが *escapetc* 以外に見た記憶はない。
- ニュートン法は *J8* では不安定

4.5　充填ジュリア集合

充填ジュリア集合はマンデルブロート集合のどこか一部のハイライトである

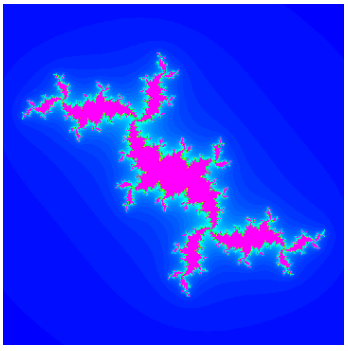
1. *Newton* 法で解を求めていない

2. *C.Reiter の Script*
   FVJ3 では背景色に変化を持たせた新しいスクリプトがあるが、安定していないし、今回のテーマは背景色よりも関数の変化なので、差し替えていない

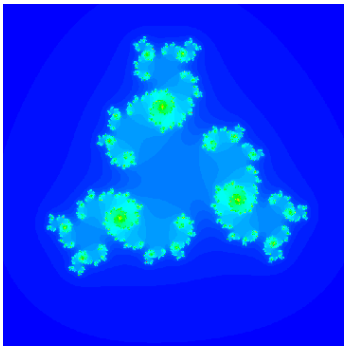
```
julia0=: 2 : '>(u escapet n) L:0 {@> y'

fjx=: 3 : '|.|: j./ ~ ({. y)+3*(i.%<:) {: y'

escapetc=: 2 : 0
#@((,u@{: )^:(<&({:n)@# *. (<&({. n)@|@{:))^:_) f. "0
)
```

3. 用いた関数

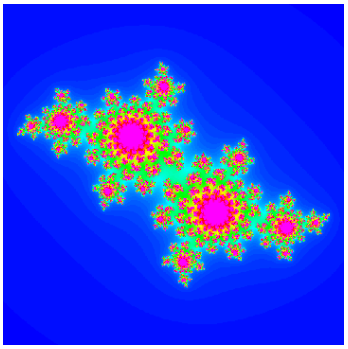
```
f5=:  +&_0.2j0.8@*:    NB. square -> add _0.2j0.8
f7=:  +&_0.2j0.8@(^&3) NB. (fine!) triple -> add _0.2j0.8
f8=:  +&_0.4j0.6@*:    NB. fine square -> add _0.2j0.8

viewmat  (f5 julia0 4 100) fjx _1.5 512
```



```
viewmat  (f7 julia0 4 100) fjx _1.5 512
```



```
viewmat  (f8 julia0 4 100) fjx _1.5 512
```



4. 複素数のカーペット *fjx*
   - 引数
     _1.5 初期値（センター）
     512 展開する範囲 512×512
   - 経過

```
   (i.%<:) 5      NB. (i.5) % 5-1
0 0.25 0.5 0.75 1

 _1.5 +3* (i.%<:) 5
_1.5 _0.75 0 0.75 1.5

   j. /~ _1.5 _0.75 0 0.75 1.5    NB. 複素数に展開
 _1.5j_1.5  _1.5j_0.75  _1.5  _1.5j0.75  _1.5j1.5
_0.75j_1.5 _0.75j_0.75 _0.75 _0.75j0.75 _0.75j1.5
    0j_1.5     0j_0.75     0     0j0.75     0j1.5
 0.75j_1.5  0.75j_0.75  0.75  0.75j0.75  0.75j1.5
```

```
1.5j_1.5   1.5j_0.75   1.5   1.5j0.75   1.5j1.5
```

References

牟田　淳「アートを生み出す 7 つの数学」オーム社　*2013*

C.Reiter [Fractal Visualization and J 3rd.Ed] 2007 Lulu