

J 研究会報告資料:2014(平成26)11月

複素平面:「三角形の重心と一つの頂点の座標を与えて、原三角形を求めよ」

中野 嘉弘 (92歳翁、札幌市・J会員) 2014/11/16

0: 「Yahoo! Japan 知恵袋 Q&A」の質問 (2014/11/7)に、複素数平面上に
重心 G 、座標 $g = 5j4$ を持つ正三角形 ABC がある。

頂点 A の座標 $a = 3j2$ である時、他の頂点 B と C の複素座標 b と c を求めよ。」

回答者は2名、「正三角形」にこだわる一人と、他に「それは不必要」とし、未だ
ベストアンサーは決まらない。

これまでの考察を J 言語流に纏めたので、J 研究会メンバーに御披露したい。

入門的資料には、本稿末尾の文献 *1)がある。

Norman Thomson 著 J: The Natural Language for Analytic Computing
(2001)pp.179-186 , Chapter21: Complex Numbers である。

1: 複素ベクトルの回転:

・ 頂点 B の件:

重心 G を中心にして、頂点 A を、120度だけ、反時計方向 anti-clockwise に
回転すれば 得られる。

・ 回転 rotation 関数

数学書の式 $\exp(i\theta) = \cos\theta + i\sin\theta$ 、

Jの関数 $\text{csj} = \cdot^{\wedge}j$. (虚数単位 j 付きの表示で)

実数部と虚数部を分離する関数 $+$. があるので、

$\text{cs} = : +.\wedge j$. (虚数単位 j 付かずの表示で) も可能。

度数からラジアンへの変換関数 $\text{dtr} = : \%180\%o$.

例) 120度は $2 * \text{pi} \% 3$ radian 故、

$\text{dtror } 120 \rightarrow 2.0944$ 。そして、

$\text{csj } 2.0944 \rightarrow _0.500004j0.866023$

実践作業では、重心 G を中心にすれば、頂点 A の座標は $(a - g) = _2j_2$ で

あり、反時計回転の結果は、

$(a - g)ip _0.500004j0.866023 \rightarrow 2.732038j_0.732038$ 。

此処に ip は、ベクトルや行列の内積関数で $ip =: +/ . *$ で定義済み。

さらに、重心値を加えて、

$2.732038j_0.732038 + g \rightarrow 7.73205j3.26796$ が、原座標系での頂点 B

の複素座標 b である。

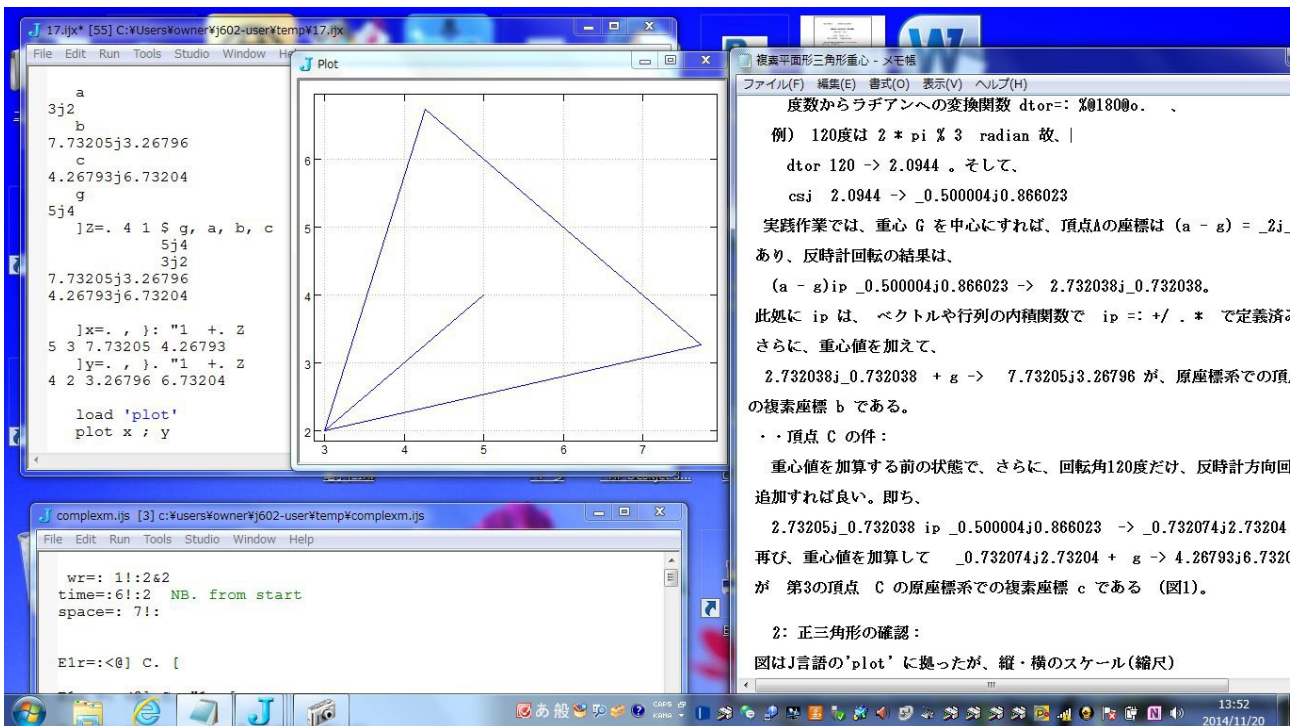
・・頂点 C の件:

重心値を加算する前の状態で、さらに、回転角 120 度だけ、反時計方向回転を

追加すれば良い。即ち、

$2.73205j_0.732038 ip _0.500004j0.866023 \rightarrow _0.732074j2.73204$ 。

再び、重心値を加算して $_0.732074j2.73204 + g \rightarrow 4.26793j6.73204$



が 第3の頂点 C の原座標系での複素座標 c である (図1)。

2: 正三角形の確認:

作図は J 言語の 'plot' に拠ったが、縦・横のスケール(縮尺)が等しいとは限らぬので、正三角形は目視だけで安心は出来ない。数值的に計算しよう。

Thomson(文献 *1)に拠れば、複素数の関数で、

絶対値 $\text{mag} =: \{.\@*\}$ と 偏角 $\text{amp} =: \{:\@*\}$ があるので、比較しよう。

三角形の辺長、 $AB = \text{mag} (b - a) =$

$\text{mag} (7.73205j3.26796 - 3j2) = 4.89898$ 。

他の 2 辺長 $BC = \text{mag} (b - c) = 4.89898$ 、

$CA = \text{mag} (c - a) = 4.89898$ で、3 者は確かに、等しいので、期待通り、

正三角形である。

3: 正三角形 と 限らぬ場合:

3.1 ベクトルのサイズを変更する関数 `trans` の例がある。

```
5 4 1 ip trans 1 1 -> 6 5 1
```

左辺の先頭の (5 4) は ベクトル $5j4$ (今は重心座標)を、右側の (1 1)

相当(実部で 1 単位、虚部で1単位分ずつだけ)の拡大の操作をする。

その回数が、`ip` の直前の 1 である。

その結果は、実部が $5 \rightarrow 6$ 、虚部で $4 \rightarrow 5$ となる。

記録として回数をも示したのが、`6 5 1` の最後の数 1 である。

3.2 今の重心の例では、

頂点 A と重心 G を結んで、その先に 1 単位ずつ延長した事を意味する。即ち、

頂点 A に向かい合う対辺の中点 M_a 、その座標は、`6j5` を示したのである。

同様にして、頂点 B の対辺の中点座標 M_b は、計算式

```
5 4 1 ip trans _1 1 -> 4 5 1
```

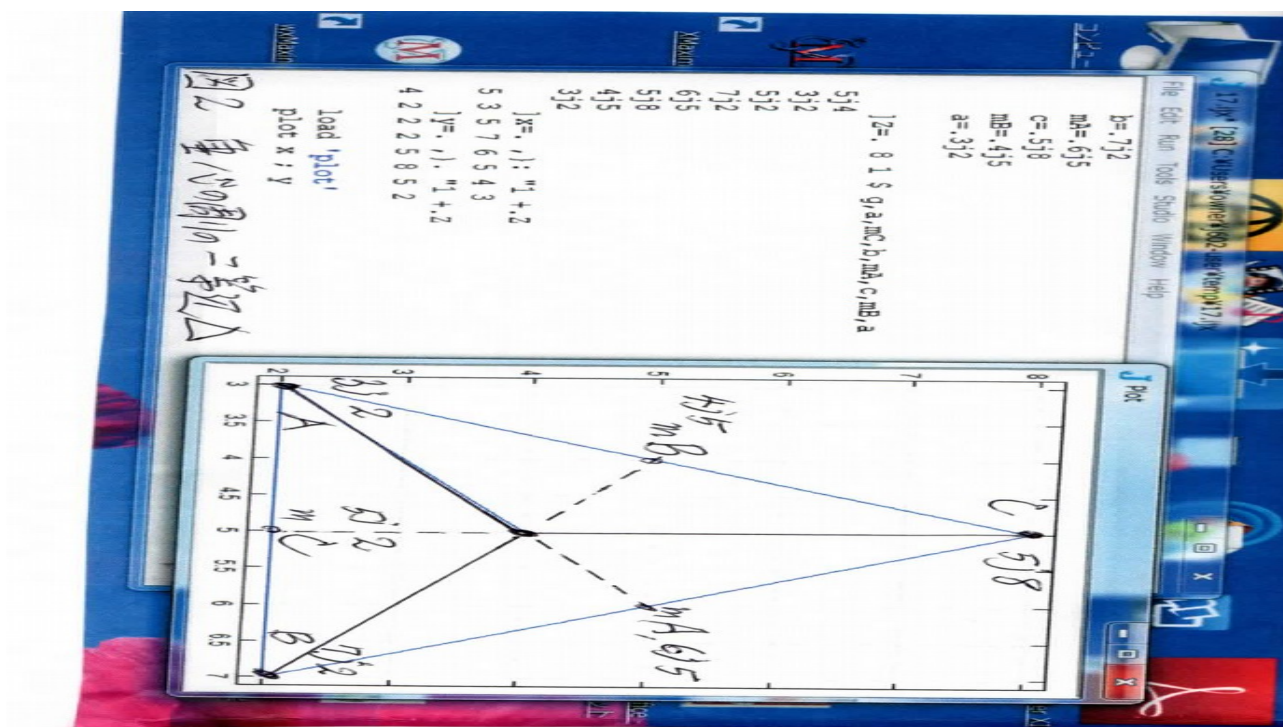
 から $4j5$ と判る。

これらを結べば、頂点 C が作図出来る。

また、辺 AB の中点座標 M_c は、明らかに $5j2$ である。対する頂点 C の座標 c

は、先の中点 M_c と 重心 G を結んで、y 方向に 2 単位分 延長すれば良い。

即ち、trans 関数を使えば、 $5\ 4\ 2\ \text{ip}\ \text{trans}\ 0\ 2 \rightarrow 5\ 8\ 2$ であり、
座標は $c = 5j8$ である (図2)。



結果は二等辺三角形であり、正三角形とは限らない。

辺長 $AB = \text{mag}(7j2 - 3j2) = 4$ 、

辺長 $AC = \text{mag}(5j8 - 3j2) = 6.32456$ 、

辺長 $BC = \text{mag}(7j2 - 5j8) = 6.32456$ 。

確かに、二等辺三角形である。

白状すれば、この解は、質問文中の条件文、「正三角形」の4文字を読み飛ばした
そそっかしい私のものであった。

4: ベクトル三角形の作図(plot)

簡単な例を先にやろう。

4.1 二等辺三角形の場合

ベクトル座標、重心 $G(g= 5j4)$ 、頂点 $A(a= 3j2)$ 、

B(b= 7j2)、C (c= 5j8) 。

全体 Z と 実数部 x、虚数部 y の分離抽出:

```
]Z=. +. (a, g, c, b)
```

```
3 2
```

```
5 4
```

```
5 8
```

```
7 2
```

```
] x =. }. "1 Z -> 3 5 5 7 、
```

```
] y =. }. "1 Z -> 2 4 8 2 。
```

```
load 'plot'
```

```
plot (3 5 5 3 7 5) ; ( 2 4 8 2 2 8) 相当が plot される (図2)。
```

記号 ; は セミコロンのことです。

一見、正三角形に見えるが、実は 縦横の尺度が不等なのである。

4.2 正三角形の場合

ベクトル座標は、

重心 G(5j4)、頂点 A (3j2)、B (7.732j3.26795), C (4.26794j6.73204)。

前項同様にして、

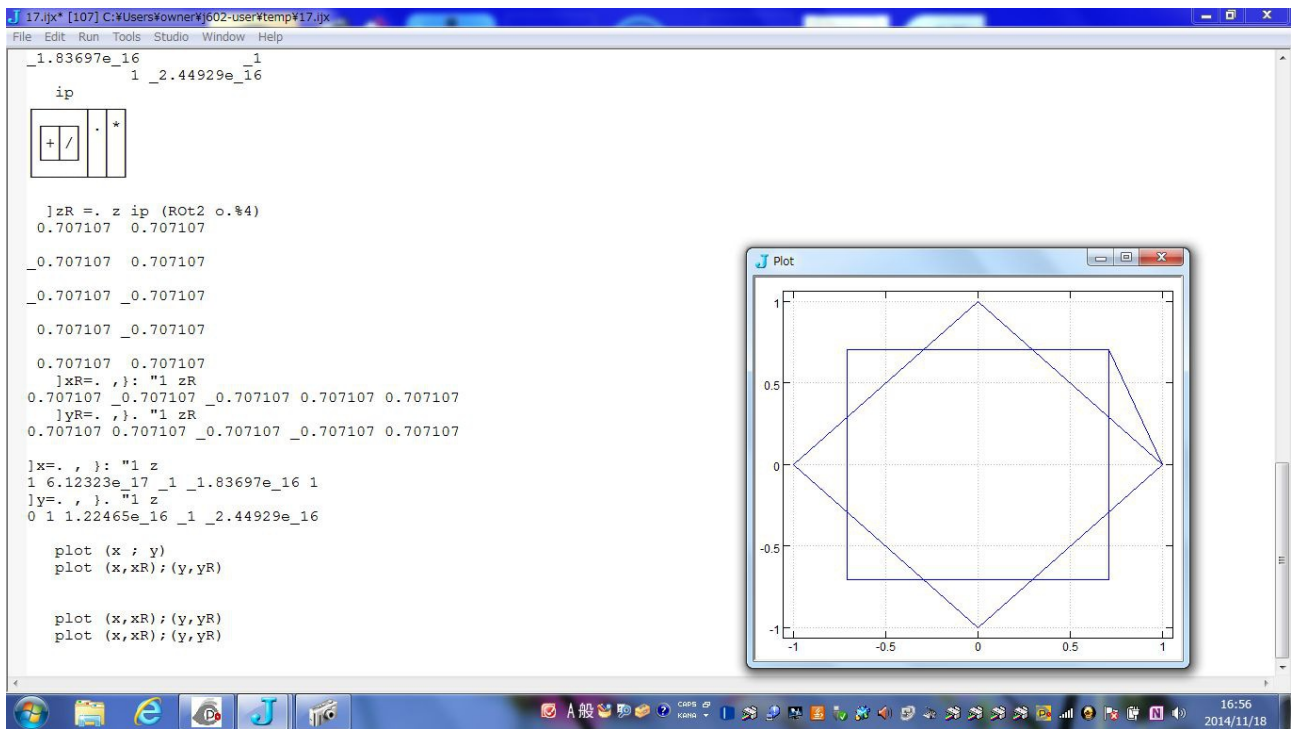
```
x3=.5 3 7.732 6 5 4.26794 5 3 3 4.26794 6
```

```
y3=.4 2 3.26795 5 4 6.73204 4 2 2 6.73204 5
```

```
plot x3 ; y3
```

から、plot される(図4)。縦横の尺度が不等の問題があるので、図は概念的な

場合が多い事に注意せよ。



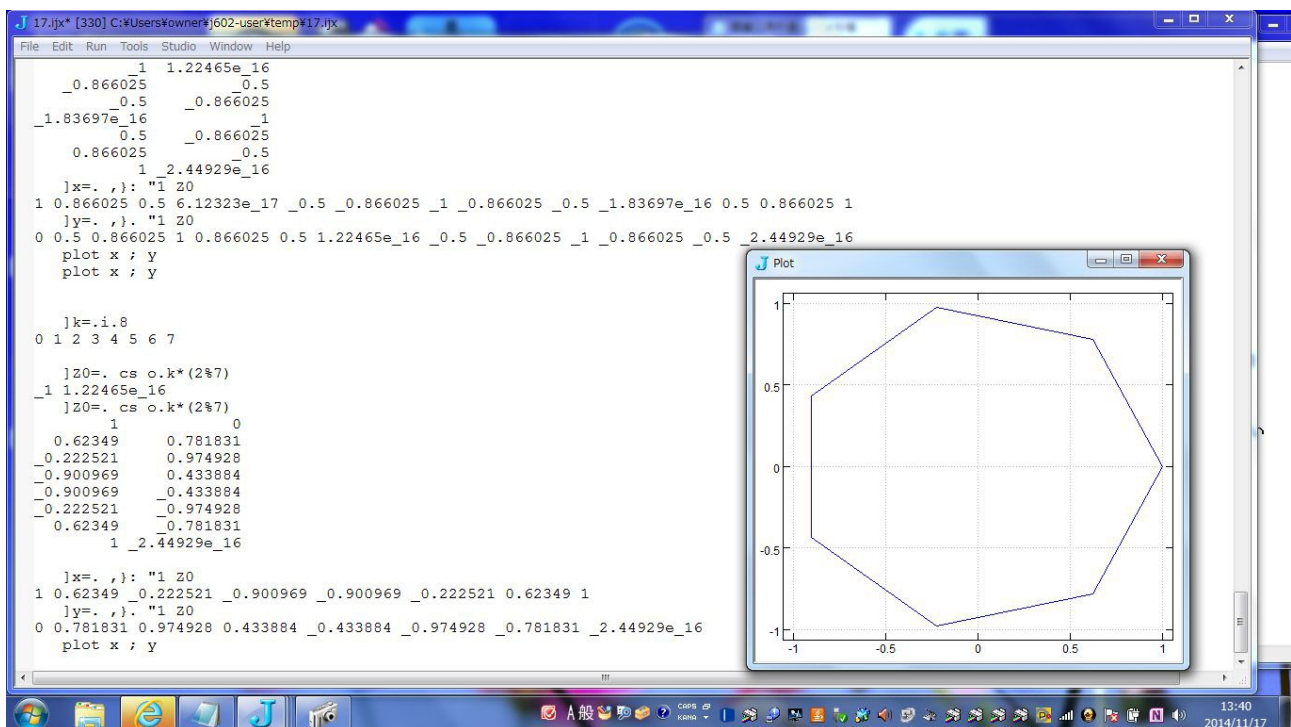
5: ベクトル多角形(正n角形)の作図

正七角形 の プロット作業の例を、そのまま、貼り付けて置く (図 3)。

プログラムの最初は、

$n = 7$ 、 $k = i.(n+1)$ として、

$z = . cs\ o.k*((2\%n))$



その実部 $x = . , } : "1 z$

その虚部 $y = . , \}$. "1 z

```
load 'plot', plot x ; y
```

とすることで良い。

なお、正八角形ならば、

```
k = . i. 9
```

```
z = . cs 0.k%8 としてスタートする。
```

また、正六角形ならば、

```
k = . i. 7
```

```
z = . cs 0.k%6 で出発する。色々、トライされたい。
```

6: 正方形回転図形

上記の方法で、正方形等の作図をする時は、回転図形の所謂、菱形と正方形とを合成されて出力したい場合もある。それらの処理法の例を示す。

```
n = . 4
```

```
]k = . i.(n+1)
```

```
]z = . cs 0. k*(2%n)
```

```
] x = . , } : "1 z
```

```
] y = . , } . "1 z
```

ここまでは、菱形を生産する。

正方形は、先の z を 45 度回転するを要する。

即ち、

```
]zR = . z ip (R0t2 0.%4)
```

文献 1 の Thomson の回転関数 rot2 を、1ヶ所 コンマに改訂した関数

```
R0t2 =: cs , (|. &cs &-)/. を、私は用いている。
```

その 45 度回転結果の zR は、

```
0.707107 0.707107
```

```
_0.707107 0.707107
_0.707107 _0.707107
0.707107 _0.707107
0.707107 0.707107 。
```

あとは以前通りで、

```
]xR=. ,}: "1 zR
0.707107 _0.707107 _0.707107 0.707107 0.707107 と
]yR=. ,}. "1 zR
0.707107 0.707107 _0.707107 _0.707107 0.707107 。
```

これを、以前の x や y の後に繋いで、

plot (x, xR) ; (y, yR) の形で プロットすれば良い。

菱形:座標(1,0)から一周し、次いで正方形:座標(0.707107) から一周が、
繋がれて描かれている(図4)。

なお、図2の二等辺三角形に、同様な回転(90度)を描き込むのも、面白い
事です。

(ヒント: 横向き二等辺三角形を、90度回転、zR=. z ip (R0t2 0.%_6)

したものも続けて実行せよ。)

The screenshot shows a Jupyter Notebook interface with three windows:

- Code Editor (Top Left):** Contains J language code for plotting a triangle. It defines variables `a`, `b`, `c`, `g`, `z`, `x`, and `y`, and uses `plot` to visualize the triangle.
- Plot Window (Top Center):** Displays a 2D plot of a triangle on a grid. The x-axis ranges from 3 to 7, and the y-axis from 2 to 6. The triangle's vertices are approximately at (3, 2), (4.5, 6), and (7, 3).
- Code Editor (Bottom Left):** Shows the execution output of the code, including timing information and the coordinates of the triangle's vertices.
- Text Window (Right):** Contains Japanese text explaining the rotation process. It discusses the conversion of degrees to radians, the use of the `ip` function for rotation, and the resulting complex coordinates for the vertices of the rotated triangle.

7: むすび

初等幾何学の図形操作に J 言語を利用するのは面白い。

文献・参考書

*1) Norman Thomson 著

J: The Natural Language for Analytic Computing

2001, Reserch Studies Press Ltd.

Baldock, Hertfordshire, England.

Chapter21: Complex Numbers pp. 179-186 。

図 1 重心の周りの正三角形

図 2 重心の周りの二等辺三角形

図 3 正七角形

図 4 回転形(菱形と正方形)