JAPLA 研究会資料　2011/9/24

# 森沢の問題「円に内接する三角形の整数解を求める」
## －計算とグラフィックス－

西川　利男

### はじめに

今年、恒例の JAPLA 蓼科合宿で森沢一弘氏より、次のような問題が持ち出された。

**一般に直角三角形は円に内接するが（Thales の定理）、直角でなく、例えば 60°や 120°で、辺の長さが整数解になるのはどんな内接三角形だろうか？**

実は、先々月、私自身、以下のようなテーマで発表を行った。

**ピタゴラス数は何通りあるか？　そして、どんな数か？－その３－Beiler の方法によるピタゴラス数の計算—**　西川利男、JAPLA 研究会資料 2011/5/28

これを受けて、森沢氏が友人の数学の先生とともに、上のようにテーマを発展させたとのことである。

三角形の２辺（$a$, $b$）とその夾角（$\theta$）から、もう一つの辺を求めるには有名な余弦第二定理がある。

$$c^2 = a^2 + b^2 - 2ab\cos\theta$$

これから

$\theta = 60^\circ$　のときには　$c^2 = a^2 + b^2 - ab$

$\theta = 120^\circ$　のときには　$c^2 = a^2 + b^2 + ab$

となるので、上の式を満足する（$a$, $b$, $c$）の整数解を求めるという問題を解くことになる。

以下、整数解の計算、グラフィック表示の順で、Ｊプログラミングの実際を述べることにする。なお、今回は$\theta = 60^\circ$ と$\theta = 120^\circ$ のときに限った。

### １．整数解の計算

まず、$\theta = 60^\circ$ と$\theta = 120^\circ$ とに対応して、２つの辺からもう一つの辺（対辺）の２乗を求めるＪの関数 func と func1 とは次のようになる。

```
func =: 3 : '((*: x.) + (*: y.)) - (x. * y.)'   NB. 60 degree

func1 =: 3 : '((*: x.) + (*: y.)) + (x. * y.)'  NB. 120 degree
```

また、これを使って、この値が平方数かどうかをテストする関数 tsq は次のようになる。

```
tsq =: 3 : 0
if. 0 = y. do. 0 return. end.
y. = *: <. %: y.
)
```

例えば、$\theta = 60^\circ$ のときに、次のように実行される。

```
   3 func 8
49
   4 func 8
48
   5 func 8
49

   tsq 3 func 8
1
   tsq 4 func 8
0
   tsq 5 func 8
1
```

すなわち、(3, 8)の組では、３辺が(3, 8, 7)となる整数の三角形が可能である。同様に(5, 8)の組では、３辺が(5, 8, 7)となる整数の三角形が可能である。しかし、(4, 8)の組ではだめである。

多数の（*a, b*）の組に対して、対辺の２乗とそれに対してテスト関数 tsq を操作し、これにより整数解を得るのだが、これをループでなく配列で行うことを考える。

例えば、(1, 1) から(8, 8)の 8x8 の組のそれぞれに$\theta = 60^\circ$ の対辺の２乗を求めて、さらにテスト関数 tsq を操作すると次のようになる。

```
   X
1 2 3 4 5 6 7 8
   Y
1 2 3 4 5 6 7 8
   X func″(0)/ Y
 1  3  7 13 21 31 43 57
 3  4  7 12 19 28 39 52
 7  7  9 13 19 27 37 49
13 12 13 16 21 28 37 48
21 19 19 21 25 31 39 49
31 28 27 28 31 36 43 52
43 39 37 37 39 43 49 57
57 52 49 48 49 52 57 64
   tsq″(0) X func″(0)/ Y
1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 1
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 1
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 1 0 1 0 0 1
```

上の配列で対角項を除いて、1となる要素(3, 8)と(5, 8)が望むものである。

これらの処理をまとめてものが、プログラムmorisである。

```
calc =: 3 : 0   NB. calculate c from a, b, then return a, b, c
:
if. x. = 0
  do.    y. , %: func / y.    NB. 60 degree
  else. y. , %: func1 / y.   NB. 120 degree
end.
)

NB.  60 degree => moris 20
NB. 120 degres => 1 moris 20
moris =: 3 : 0
0 moris y.
:
N =. y.
X =: >: i. N
Y =: >: i. N
XY =: {X ; Y            NB. generate index pair by using  { catalog
V =: (> </ L:0 XY)      NB. extract V index
if. x. = 0
 do.    Z =: X func"(0) / Y    NB. 60 degree
 else. Z =: X func1"(0) / Y   NB. 120 degree
end.
T =: V * Z                     NB. squared values
W =: (, tsq"(0) T) # (, XY)    NB. flag of squared term index
if. x. = 0
 do.    0 calc L:0 W
 else. 1 calc L:0 W
end.
)
```

実行のようすは以下のようである。

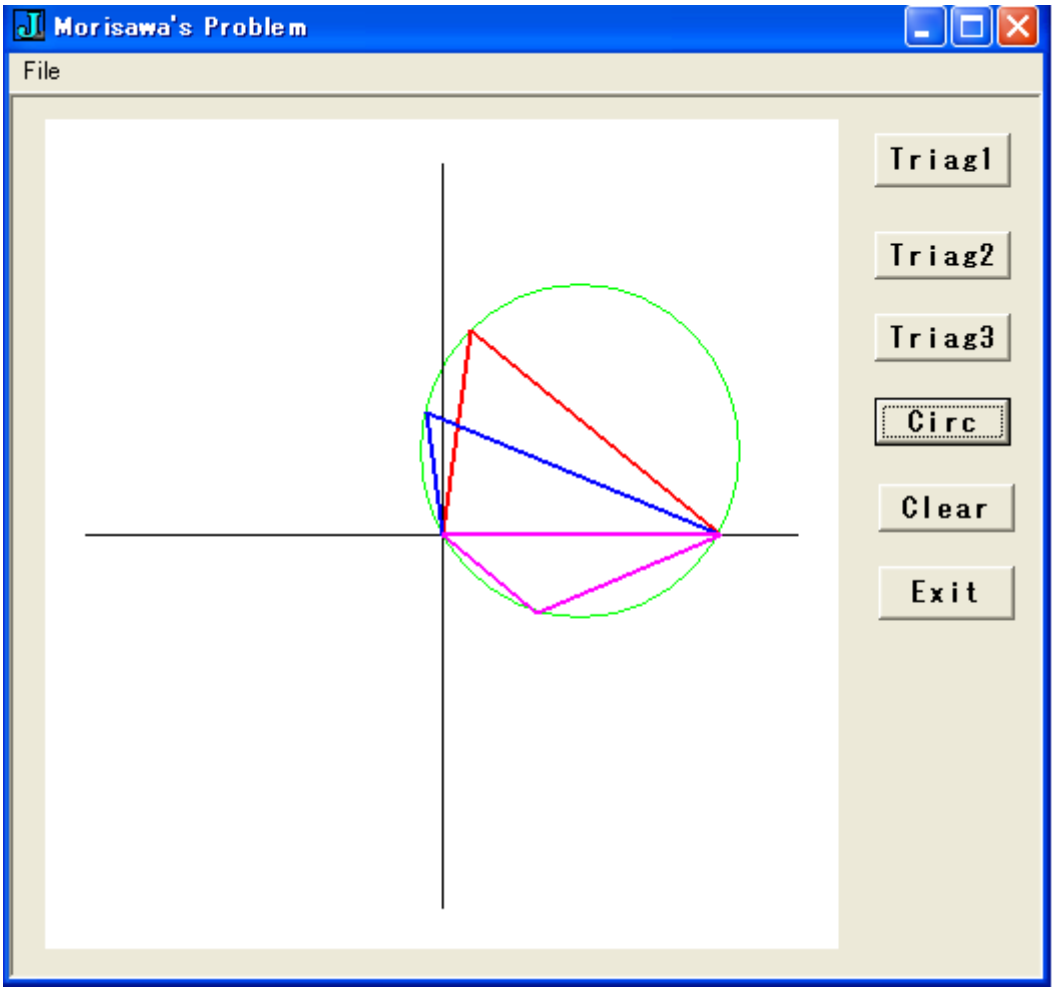
```
NB.  60 degree
NB.    moris 20
NB. +-----+-----+-------+-------+-------+--------+
NB. |3 8 7|5 8 7|6 16 14|7 15 13|8 15 13|10 16 14|
NB. +-----+-----+-------+-------+-------+--------+
NB. 120 degree
NB.   1 moris 20
NB. +-----+-------+-------+------+-------+--------+--------+
NB. |3 5 7|5 16 19|6 10 14|7 8 13|9 15 21|12 20 28|14 16 26|
NB. +-----+-------+-------+------+-------+--------+--------+
```

最終的には、$\theta = 60^{\circ}$ と$\theta = 120^{\circ}$ をまとめてユニークなものだけをソートして結果を得る関数morisnとした。

## ２．内接三角形のグラフィックス

$\theta = 60^\circ$ の場合、整数解をとる内接三角形（3, 8, 7)と（5, 8, 7）とをボタンTriag1(赤)とTriag2(青)とにより描く。

$\theta = 120^\circ$ の場合、整数解をとる内接三角形(3, 5, 7)はボタンTriag3(紫)で描く。ボタンCircを押すと、三角形に外接する円が緑で描かれる。




いろいろな大きさの内接三角形をグラフィックスで描くため、いくつかの計算が必要である。三角形は辺の長さで与えられるので、３つの長さから頂点座標を求めるのに関数triaxyを作った。また、Jグラフィックスで円を描くには周囲のワクと左下の位置が必要なので、３つの長さから外接円のこの値を求める関数circleを作った。

これらを含めて、グラフィックス・プログラムの詳細はJのソースリストを参照のこと。

```
NB. Morisawa's Problem
NB.   programmed by T.N. 2011/8/7-9

require 'trig'

func =: 3 : 0   NB. 60 degree
:
((*: x.) + (*: y.)) - (x. * y.)
)

func1 =: 3 : 0  NB. 120 degree
:
((*: x.) + (*: y.)) + (x. * y.)
)



tsq =: 3 : 0
if. 0 = y. do. 0 return. end.
y. = *: <. %: y.
)


NB.  60 degree
NB.    moris 20
NB. +-----+-----+-------+-------+-------+--------+
NB. |3 8 7|5 8 7|6 16 14|7 15 13|8 15 13|10 16 14|
NB. +-----+-----+-------+-------+-------+--------+
NB. 120 degree
NB.   1 moris 20
NB. +-----+-------+-------+------+-------+--------+--------+
NB. |3 5 7|5 16 19|6 10 14|7 8 13|9 15 21|12 20 28|14 16 26|
NB. +-----+-------+-------+------+-------+--------+--------+

calc =: 3 : 0   NB. calculate c from a, b, then display a, b, c
:
if. x. = 0
  do.   y. , %: func / y.
  else. y. , %: func1 / y.
end.
)

NB.  60 degree => moris 20
NB. 120 degres => 1 moris 20
moris =: 3 : 0
```

```
0 moris y.
:
N =. y.
X =: >: i. N
Y =: >: i. N
XY =: {X ; Y
if. x. = 0
 do.
    Z =: X func"(0) / Y
 else.
    Z =: X func1"(0) / Y
end.
V =: (> </ L:0 XY)
T =: V * Z
W =: (, tsq"(0) T) # (, XY)
if. x. = 0
 do.   R =. 0 calc L:0 W
 else. R =. 1 calc L:0 W
end.
R
)

NB. calc. overall for 60 degree and 120 degree, then sort them
morisn =: 3 : 0
RA =. > moris y.
RB =. > 1 moris y.
RC =. RA , RB
R0 =. |."(1) > RC
R1 =. R0 /: R0
(|.^:_1) "(1) R1
)

NB. lengths of triangle => XY-coordinate of points
NB.   revised 2011/9/13
NB. Usage: triaxy a, b, c => AXY, BXY, CXY
triaxy =: 3 : 0
sinC =. (%: 3) % 2
'a b c' =. y.
sinA =. sinC * a % c
sinB =. sinC * b % c
'A B' =. (180 % 1p1) * arcsin sinA, sinB
if. 180 = 120 + A + B
  NB. 120 degree
  do.
```

```
     'BXY AXY CXY' =. (0, 0); (({:y.), 0); ({.y.) * ((cos , (-@sin)) 1r180p1
* B)
  NB.  60 degree
  else.
   if. 180 = A + B + 60
     do.    B =. B
     else. B =. 180 - B
   end.
     'BXY AXY CXY' =. (0, 0); (({:y.), 0); ({.y.) * ((cos , sin) 1r180p1 * B)
end.
BXY; AXY; CXY
)


NB. calc. diameter and center of circumcircle for J graphics
circle =: 3 : 0
((60%180)*1p1) circle y. NB. 60 degree
:
sinC =. sin x.
C =. (180 % 1p1) * arcsin sinC
'a b c' =. y.
r =. c % (2 * sinC)
cent =. (r * sin x.), (r * cos x.)
NB. diameter, circ-rectangle positon for J graphics
(2*r), ((0{cent)-r), ((1{cent)-r)
)

NB. Graphical Representation ========================
require 'gl2'

MORISA=: 0 : 0
pc morisa;pn "Morisawa's Problem";
menupop "File";
menu new "&New" "" "" "";
menu open "&Open" "" "" "";
menusep ;
menu exit "&Exit" "" "" "";
menupopz;
xywh 215 8 34 12;cc Triag1 button;
xywh 216 104 34 12;cc cancel button;cn "Exit";
xywh 8 5 198 184;cc triangle isigraph;
xywh 215 67 34 11;cc Circ button;
xywh 215 30 34 11;cc Triag2 button;
xywh 216 86 34 11;cc Clear button;
xywh 215 48 34 11;cc Triag3 button;
```

```
pas 6 6;pcenter;
rem form end;
)

dataset =: 3 : 0
n =. _3 {. i. 3 * y.
n { morisn 40
)

run =: morisa_run
morisa_run=: 3 : 0
wd MORISA
NB. initialize form here
DA =: dataset y.
'BB AA CC' =: triaxy 1{ DA
'BB AA DD' =: triaxy 2{ DA
'BB AA EE' =: triaxy 0{ DA
CIR =: circle 1{ DA
wd 'pshow;'
)


run0 =: morisa_run0
morisa_run0=: 3 : 0
wd MORISA
NB. initialize form here
'BB AA CC' =: triaxy 5 8 7
'BB AA DD' =: triaxy 3 8 7
'BB AA EE' =: triaxy 3 5 7
CIR =: circle 5 8 7
wd 'pshow;'
)

morisa_close=: 3 : 0
wd'pclose'
)

morisa_cancel_button=: 3 : 0
morisa_close''
)


SC =: 50   NB. Scaling Factor
OR =: 220  NB. Origin of Graph
```

```
morisa_Triag1_button=: 3 : 0
NB. X-Y coordinate
glrgb 0 0 0
glpen 1, 0
gllines 50, OR, 950, OR
gllines OR, 50, OR, 950
NB. Triangle Graph
glrgb 255 0 0
glpen 4, 0

NB. gllines 100 500 900 500 500 800 100 500
NB. gllines BP, AP, CP, BP
gllines OR + SC * BB, AA, CC, BB
glshow ''
)

morisa_Triag2_button=: 3 : 0
glrgb 0 0 255
glpen 4, 0
gllines OR + SC * BB, AA, DD, BB
glshow ''
)

morisa_Triag3_button=: 3 : 0
glrgb 255 0 255
glpen 4, 0
gllines OR + SC * BB, AA, EE, BB
glshow ''
)


morisa_Circ_button=: 3 : 0
NB. Circle Graph
glrgb 0 255 0
glpen 2, 0
glellipse (OR + SC*1{CIR), (OR + SC*2{CIR), (SC*0{CIR), (SC*0{CIR)
NB. X-Y coordinate
glrgb 0 0 0
glpen 1, 0
gllines 50, OR, 950, OR
gllines OR, 50, OR, 950
NB. Triangle1 Graph
glrgb 255 0 0
glpen 4, 0
gllines OR + SC * BB, AA, CC, BB
```

```
NB. Triangle2 Graph
glrgb 0 0 255
glpen 4, 0
gllines OR + SC * BB, AA, DD, BB
glrgb 255 0 255
glpen 4, 0
gllines OR + SC * BB, AA, EE, BB
glshow ''
)

morisa_Clear_button=: 3 : 0
glclear ''
NB. X-Y coordinate
glrgb 0 0 0
glpen 1, 0
gllines 50 500 950 500
gllines 500 50 500 950
glshow ''
)
```