

JでCORDIC（電卓での三角関数計算）のしくみをさぐる

西川 利男

0. CORDIC とは[1]

CORDIC(Coordinate Rotation Digital Computer)とは、四則計算だけで sin, cos, tan など三角関数の値を得る演算法である。アルゴリズムが比較的簡単なので、安価な電卓にも、ハードワイヤード・ロジックで作ることで、三角関数を実装することが出来る。この計算がどうやって行われるか、Jでさぐってみた。

[1] 戸田英雄、小野令美「入門 数値計算」p.140-144, オーム社(1983).

1. CORDIC の考え方[2]

具体的な例として、例えば角度 30° に対する CORDIC の計算を考えてみよう。

まず、 45° から tan の値が次々と半分になる角

θ	45°	26.57°	14.03°	7.13°	3.58°	1.79°
----------	------------	---------------	---------------	--------------	--------------	--------------

tan θ	1	0.5	0.25	0.125	0.0625	0.03125
--------------	---	-----	------	-------	--------	---------

を何らかの方法であらかじめ求め、表の形で保存しておく。

さて、ここで角度 30° は上の角度を使うと次のように

$$30^\circ = 45^\circ - 26.57^\circ + 14.03^\circ - 7.13^\circ + 3.58^\circ + 1.79^\circ$$

次々と精緻に近似される。

これら漸近的に半分になる角の和や差で出来る直角三角形は、その直前の直角三角形から、次の式を利用することで容易に求められる。

$$\tan \frac{A}{2} = \frac{\sqrt{1 - \cos A}}{\sqrt{1 + \cos A}} = \frac{-1 + \sqrt{1 + \tan^2 A}}{\tan A}$$

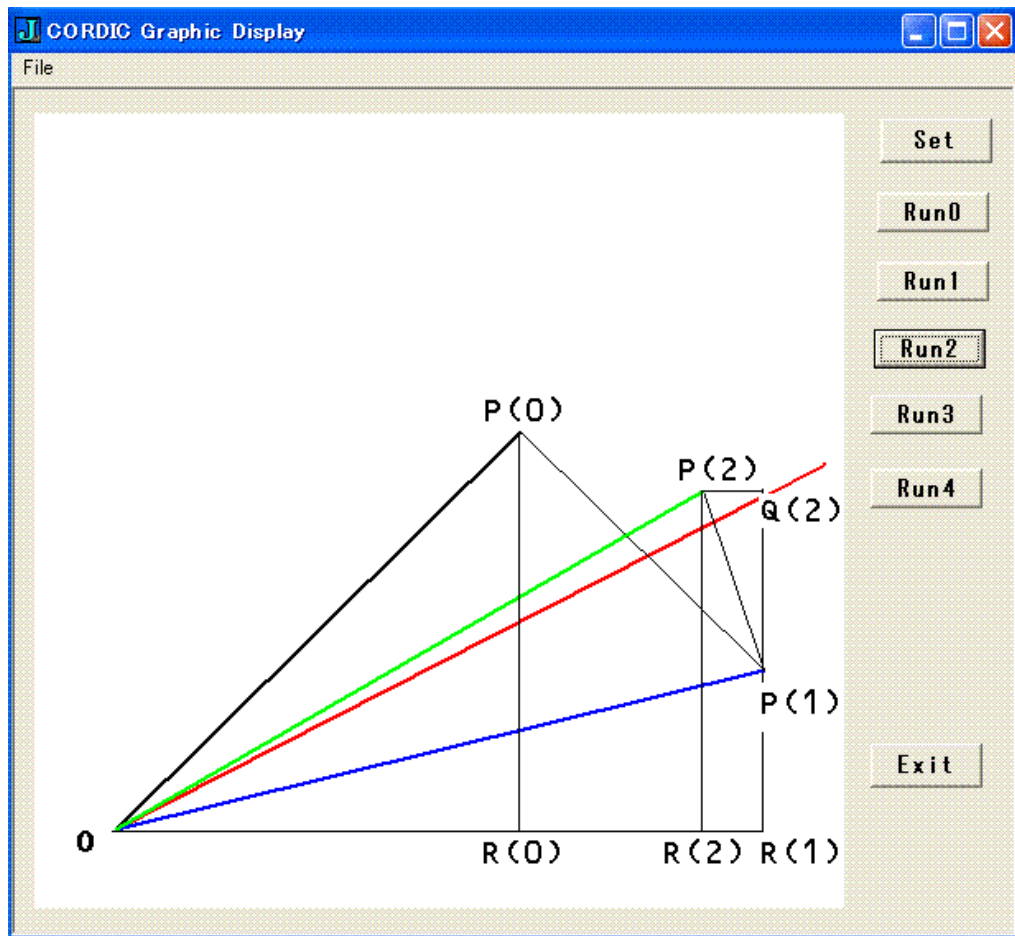
ここで、先の θ -tan θ の表の値を使えば、任意の角の直角三角形が得られて、その辺の比から三角関数値が得られる、というのが CORDIC の考え方のポイントである。

CORDIC は計算方法のアルゴリズムであるが、その理解はなかなかややこしい。そこで、ここでは、Jのグラフィックス表示を利用して、図により説明していこう。

[2] 関数電卓のしくみ (CORDIC アルゴリズム)

<http://teamcoil.sp.u-tokai.ac.jp/calculator/100224/>

続いて、次の近似へと進めていこう。



次は、直角三角形 $0Q_1P_1$ の斜辺の上に、次の中心角 $\angle P_2OP_1 = 14.03^\circ$ の直角三角形 OP_1P_2 をのせる。これはうまくいった。斜辺は目的とする 30° の線にぐっと近づいた。

直角三角形 OR_2P_2 、中心角 $\angle R_2OP_2 = 32.46^\circ (= 18.43^\circ + 14.03^\circ)$ となる。

このとき、点 P_2 の座標は次のようにして求められる。直角三角形 $P_1Q_2P_2$ は直角三角形 OR_1P_1 と相似であり、相似比は $1/4$ である。

$$\text{点 } P_2 \text{ の } x \text{ 座標は } 3/2 - 1/2 * 1/2^2 = 1.5 - 0.125 = 1.375$$

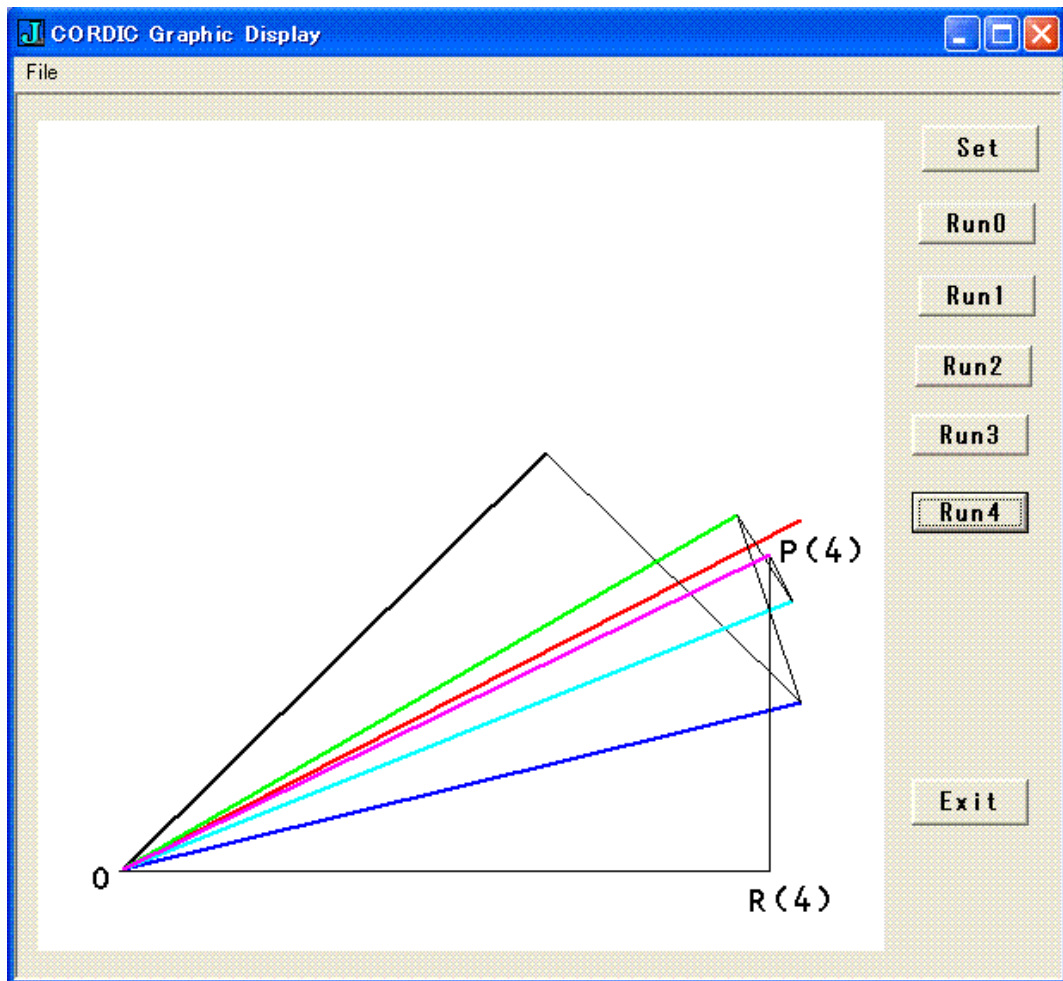
$$\text{点 } P_2 \text{ の } y \text{ 座標は } 1/2 - 3/2 * 1/2^2 = 0.5 + 0.375 = 0.875$$

このようにして

$$\text{点 } P_2(1.375, 0.875)$$

である。

さらに、同様の近似を続けていく。



このとき、大切なことは直角三角形の中心角を、次々と小さくなる θ を足したり引いたりして 30° に近づくように、試行錯誤で行うことにある。

上は、4回行って、直角三角形 OR_4P_4 を得たところである。

このように、次の近似となる直角三角形の各辺が、先の $\theta - \tan \theta$ の表の値を使って、四則計算で求められる。これがCORDICアルゴリズムの考え方である。

3. CORDIC の計算

3. 0 準備 三角関数の度数表示

ここでの CORDIC の計算では、角度の表示に度数を用いることにする。

```
require 'trig'  
arcsind =: ((180*1p_1)&*)@arcsin  
arccosd =: ((180*1p_1)&*)@arccos  
arctand =: ((180*1p_1)&*)@arctan
```

3. 1 準備 $\theta - \tan \theta$ の表の値(次々と半分になる \tan の値の表)の作成

```
I18 =: i. 18  
TH =: arctand 0.5^I18  
secd =: %@cosd  
9!:11 (10)  
SE =: */¥ secd TH  
SE18 =: I18 ,. SE
```

```
(2j0": ,. I18), "(1) 10j6 ": TH ,. SE  
0 45.000000 1.414214  
1 26.565051 1.581139  
2 14.036243 1.629801  
3 7.125016 1.642484  
4 3.576334 1.645689  
5 1.789911 1.646492  
6 0.895174 1.646693  
7 0.447614 1.646744  
8 0.223811 1.646756  
9 0.111906 1.646759  
10 0.055953 1.646760  
11 0.027976 1.646760  
12 0.013988 1.646760  
13 0.006994 1.646760  
14 0.003497 1.646760  
15 0.001749 1.646760  
16 0.000874 1.646760  
17 0.000437 1.646760
```

3. 2 CORDIC 計算の J プログラム

次のような J の CORDIC 計算のプログラムを作った。試行錯誤による足し算、引き算の選択は、if. do. else. end. の選択構文でおこなった。

```
cordic =: 3 : 0
Angle =. y.
S =. 45
X =. 1
Y =. 1
XA =. ''
YA =. ''
i =. 0
while. i < 17
  do.
    T =. 0 < Angle - S
    TA =. ((+:)@(_0.5&+)) T
    if. T
      do. NB. wr 'plus'
        S1 =. S + (>:i){TH
        X1 =. X - (0.5^>:i) * Y
        Y1 =. Y + (0.5^>:i) * X
      else. NB. wr 'neg'
        S1 =. S - (>:i){TH
        X1 =. X + (0.5^>:i) * Y
        Y1 =. Y - (0.5^>:i) * X
      end.
    XA =. XA, X
    YA =. YA, Y
    S =. S1
    X =. X1
    Y =. Y1
    i =. i + 1
  end.
```

3. 3 CORDIC 計算の実行例

角度 30° の CORDIC の計算例の途中経過とそれによる三角関数の値をしめした。

```
cordic 30
CORDIC test diminished add, sub      X      Y
-----
```

0	45.00000 - 26.56505 = 18.43495 /	1.00000	1.00000
1	18.43495 + 14.03624 = 32.47119 /	1.50000	0.50000
2	32.47119 - 7.12502 = 25.34618 /	1.37500	0.87500
3	25.34618 + 3.57633 = 28.92251 /	1.48438	0.70313
4	28.92251 + 1.78991 = 30.71242 /	1.44043	0.79590
5	30.71242 - 0.89517 = 29.81725 /	1.41556	0.84091
6	29.81725 + 0.44761 = 30.26486 /	1.42870	0.81879
7	30.26486 - 0.22381 = 30.04105 /	1.42230	0.82996
8	30.04105 - 0.11191 = 29.92915 /	1.42554	0.82440
9	29.92915 + 0.05595 = 29.98510 /	1.42715	0.82162
10	29.98510 + 0.02798 = 30.01307 /	1.42635	0.82301
11	30.01307 - 0.01399 = 29.99909 /	1.42595	0.82371
12	29.99909 + 0.00699 = 30.00608 /	1.42615	0.82336
13	30.00608 - 0.00350 = 30.00258 /	1.42605	0.82353
14	30.00258 - 0.00175 = 30.00083 /	1.42610	0.82344
15	30.00083 - 0.00087 = 29.99996 /	1.42612	0.82340
16	29.99996 + 0.00044 = 30.00040 /	1.42614	0.82338

tand 30 = 0.57735952

sind 30 = 0.50000601

cosd 30 = 0.86602193

*** CORDIC end ***

Jのプログラム

NB. CORDIC(Coordinate Rotation Digital Computer) on J

NB. T. Nishikawa 2015/4/13

NB. from 関数電卓のしくみ (CORDIC アルゴリズム)

NB. <http://teamcoil.sp.u-tokai.ac.jp/calculator/100224/>

```
tan_half =: 3 : 0
TAN =. y.
(_1 + %: 1 + TAN^2) % TAN
)
```

NB. e.g. cordic 30, cordic 60 etc.

```
require 'trig'
```

```
arcsind =: ((180*1p_1)&*)@arcsin
arccosd =: ((180*1p_1)&*)@arccos
arctand =: ((180*1p_1)&*)@arctan
```

```
I18 =: i. 18
TH =: arctand 0.5^I18
TH18 =: I18 ,. TH
```

```
secd =: %@cosd
```

```
9!:11 (10)
SE =: */¥ secd TH
SE18 =: I18 ,. SE
```

```
rd =: 1!:1
wr =: 1!:2&2
```



```

cordic =: 3 : 0
Angle =. y.
S =. 45
X =. 1
Y =. 1
XA =. ''
YA =. ''
wr '   CORDIC test diminished add, sub      X      Y'
wr '-----',
i =. 0
while. i < 17
  do.
    T =. 0 < Angle - S
    TA =. ((+:)@(_0.5&+)) T
NB.   wr (2":i), (10j5":i){TH}, (2": T), ' ', (3": TA), ' ', (10j5": X, Y)

NB.   if. 1 = ". rd 1
      if. T
        do. NB. wr 'plus'
          S1 =. S + (>:i){TH
          X1 =. X - (0.5^>:i) * Y
          Y1 =. Y + (0.5^>:i) * X
        else. NB. wr 'neg'
          S1 =. S - (>:i){TH
          X1 =. X + (0.5^>:i) * Y
          Y1 =. Y - (0.5^>:i) * X
        end.
      wr (2":i), (10j5":S), ' ', (T{'-+'}, (9j5":(i+1){TH}, ' = ', (8j5": S
+ TA*(i+1){TH}, ' /', (10j5": X, Y)
      XA =. XA, X
      YA =. YA, Y
      S =. S1
      X =. X1
      Y =. Y1
      i =. i + 1
    end.
end.

```

```

wr '          ',
wr 'tand ', (': Angle), ' = ', 10j8": TAN =. Y % X
wr 'sind ', (': Angle), ' = ', 10j8": SIN =. TAN % %: 1 + *: TAN
wr 'cosd ', (': Angle), ' = ', 10j8": COS =. %: 1 - *: SIN
XB =: XA
YB =: YA
'*** CORDIC end ***'
)

```