Part VI

Algorithms & C

"To die for a religion is easier than to live it absolutely"

—Jorge Luis Borges

59 CORDIC

"Whoever makes himself humble will be made great."

CORDIC (**CO**ordinate **R**otation **DI**gital **C**omputer) Algorithm is heavily used for implementing mathematical functions, especially in scientific calculators. But unfortunately this neat algorithm is not much known to people. Also people who know this algorithm keep it closed and badly documented. So I thought this good algorithm should be known to the programming community. I have managed to collect materials from many sources and I have understood the real stuff of CORDIC.

59.1 Birth of CORDIC

CORDIC was introduced by Volder in 1959 to calculate trigonometric values like sine, cosine, etc. In 1971, Walther extended this algorithm to calculate hyperbolic, logarithmic and other functions.

59.2 Advantages

This algorithm uses only minimal hardware (adder and shift) for computation of all trigonometric and other function values. It consumes fewer resources than any other techniques and so the performance is high. Thus, almost all scientific calculators use the CORDIC algorithm in their calculations.

59.3 Principle

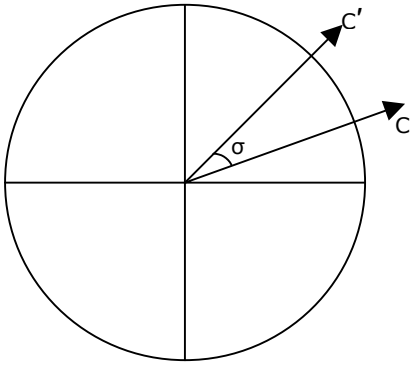
CORDIC works by rotating the coordinate system through constant angles until the angle is reduced to zero. So with this principle we are changing the given angle each time to reduce to zero. Here we are using addition, subtraction and shift to calculate the function values.

Now let us see, how we can calculate sine and cosine values using CORDIC. Consider a vector C with coordinate (X, Y) that is to be rotated through an angle σ. The new coordinate (X′,Y′) after rotation is

$$C' = \begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} X\cos(\sigma) - Y\sin(\sigma) \\ Y\cos(\sigma) - X\sin(\sigma) \end{bmatrix}$$

This equation can be represented in tangent form as

$$X/\cos(\sigma) = X - Y \times \tan(\sigma)$$
$$Y/\cos(\sigma) = Y - X \times \tan(\sigma)$$



The angle is broken into smaller and smaller pieces, such that the tangent of the angle is always power of 2. The pre-calculated angles are also added to the total angle and thus the above equation can be written as

$$X(i+1) = t(i) \times (X(i) - Y/2^i)$$
$$Y(i+1) = t(i) \times (Y(i) - X/2^i)$$

where $t(i) = \cos(\arctan(1/2^i))$

i varies from 0 to n

According to the above iterative equation t(i) will converge to a 'constant' after first few iterations (i.e., when i get varies). So it is better to pre-calculate this 'constant' for a greater value of n as:

$$T = \cos(\arctan(1/2^0)) \times \cos(\arctan(1/2^1)) \times \ldots\ldots\ldots \times \cos(\arctan(1/2^n))$$

Calculated value of T will *always* be `0.60725293500888`. We can use any precision for T. But the accuracy of the calculation of sine and cosine depends on the precision we use and so it is recommended to use at least 6 precision in your calculation.

While we program, the value of the angle $\arctan(1/2^i)$ can be pre-calculated and stored in an array. This value can be used in the iterations and it avoids the calculation at the iterative time.

59.4 Algorithm

The steps for CORDIC algorithm are:

1. Get the angle and store it in `Angle`. Store the pre-calculated arctan values in an array
2. Assign X = 0.607252935 (i.e., X=T), Y=0
3. Find X′ and Y′
4. If sign of `Angle` is positive then
   X = X - Y′
   Y = Y + X′
   else (If sign of `Angle` is negative)
   X = X + Y′
   Y = Y - X′
5. Repeat steps (3) and (4) till the `Angle` approaches 0
6. Print "Value of cos = " X
7. Print "Value of sin = " Y
8. Exit

59.5 Program

Following is the program for calculating sine and cosine value for a given angle. In this program the variable `Angle` holds the supplied angle (for which we have to find the cosine and sine values). `Arctans[i]` holds the precalculated angle of arctan's. Then in each iteration the value of `Arctans[i]` is subtracted from or added to `Angle` according to the sign of the `Angle` value. We can finish the iteration when `Angle` becomes 0 or to a nearer value (say, 0.00001). The value of X and Y will also incremented or decremented according to `Angle` value.

After the completion of this program, cosine value will be stored in X and sine value will be stored in Y for a given `Angle`.

```
#define    T         (0.60725293500888)
#define    SIZE      (50)
#define    ZERO      (0.00000001)  /* approximation for zero */

#include <math.h>
int main( void )
{
   int i = 0;
   double X = T, Y = 0.0, Angle;
   double dx, dy;
   double Arctans[SIZE] =
      {
      45.0000000000000, 26.5650511770780, 14.0362434679265,
      7.1250163489018,  3.5763343749974,  1.7899106082461,
      0.8951737102111,  0.4476141708606,  0.2238105003685,
      0.1119056770662,  0.0559528918938,  0.0279764526170,
      0.0139882271423,  0.0069941136754,  0.0034970568507,
      0.0017485284270,  0.0008742642137,  0.0004371321069,
      0.0002185660534,  0.0001092830267,  0.0000546415134,
      0.0000273207567,  0.0000136603783,  0.0000068301892,
      0.0000034150946,  0.0000017075473,  0.0000008537736,
      0.0000004268868,  0.0000002134434,  0.0000001067217,
      0.0000000533609,  0.0000000266804,  0.0000000133402,
      0.0000000066701,  0.0000000033351,  0.0000000016675,
      0.0000000008338,  0.0000000004169,  0.0000000002084,
      0.0000000001042,  0.0000000000521,  0.0000000000261,
      0.0000000000130,  0.0000000000065,  0.0000000000033,
      0.0000000000016,  0.0000000000008,  0.0000000000004,
      0.0000000000002,  0.0000000000001
      };
   printf( "Enter the Angle : " );
   scanf( "%lf", &Angle );
   printf("I\tX\t\tY\t\tAngle\t\tPreCal arctan()\n");
```

```
        while( fabs(Angle) >= ZERO && i < SIZE )
          {
           printf("\n%2d   %3.11lf   %+3.11lf   %+3.11lf   %3.11lf",
                       i, X, Y, Angle, Arctans[i]);
           dx = X / pow(2, i);
           dy = Y / pow(2, i);
           if( Angle >= 0.0 )
              {
                Angle -= Arctans[i];
                X -= dy;
                Y += dx;
              }
            else
              {
                 Angle += Arctans[i];
                 X += dy;
                 Y -= dx;
              }
            ++i;
          }
        return(0);
      } /*--main( )------*/
```

Here is the output of our program for `Angle = 3`.

```
I        X              Y              Angle          PreCal arctan()

 0   0.60725293501   +0.00000000000   +3.00000000000   45.00000000000
 1   0.60725293501   +0.60725293501   -42.00000000000   26.56505117708
 2   0.91087940251   +0.30362646750   -15.43494882292   14.03624346793
 3   0.98678601939   +0.07590661688   -1.39870535500   7.12501634890
 4   0.99627434650   -0.04744163555   +5.72631099391   3.57633437500
 5   0.99923944872   +0.01482551111   +2.14997661891   1.78991060825
 6   0.99877615150   +0.04605174388   +0.36006601066   0.89517371021
 7   0.99805659300   +0.06165762125   -0.53510769955   0.44761417086
 8   0.99853829317   +0.05386030412   -0.08749352869   0.22381050037
 9   0.99874868498   +0.04995976391   +0.13631697168   0.11190567707
10   0.99865110732   +0.05191044493   +0.02441129461   0.05595289189
11   0.99860041352   +0.05288569016   -0.03154159728   0.02797645262
12   0.99862623661   +0.05239809230   -0.00356514466   0.01398822714
13   0.99863902912   +0.05215428706   +0.01042308248   0.00699411368
14   0.99863266263   +0.05227619124   +0.00342896880   0.00349705685
15   0.99862947194   +0.05233714294   -0.00006808805   0.00174852843
16   0.99863106914   +0.05230666719   +0.00168044038   0.00087426421
17   0.99863027101   +0.05232190509   +0.00080617617   0.00043713211
18   0.99862987182   +0.05232952403   +0.00036904406   0.00021856605
19   0.99862967220   +0.05233333351   +0.00015047801   0.00010928303
```

```
20   0.99862957238   +0.05233523824   +0.00004119498   0.00005464151
21   0.99862952247   +0.05233619061   -0.00001344653   0.00002732076
22   0.99862954743   +0.05233571442   +0.00001387422   0.00001366038
23   0.99862953495   +0.05233595252   +0.00000021385   0.00000683019
24   0.99862952871   +0.05233607156   -0.00000661634   0.00000341509
25   0.99862953183   +0.05233601204   -0.00000320125   0.00000170755
26   0.99862953339   +0.05233598228   -0.00000149370   0.00000085377
27   0.99862953417   +0.05233596740   -0.00000063993   0.00000042689
28   0.99862953456   +0.05233595996   -0.00000021304   0.00000021344
```

The value of `cos(3)` is stored in X and `sin(3)` is stored in Y. Thus, according to the precision we use for T, the accuracy of the cosine and sine values can be increased or decreased.