Angle Recording CORDIC

1. Hu

20180914 Fri

to reduce the number of CORDIC iterations ← greedy

by encoding the angle of rotation
as a linear combination of
selected elementary angle of micro-rotations

Signal / Image processing DFT & DCT
– the rotation angle known a priori

greedy algorithms to perform angle recoding

linear combination of
elementary rotation angles (EAS)

FFT, Chirp-z

a circular rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$\cos\theta$, $\sin\theta$

CORDIC : a sequence of successive rotation

n elementary rotation angles

$a(i)$, $i = 0, \cdots, n-1$

$\tan[a(i)] = 2^{-i}$

only shifts and adds operations

Conventional CORDIC

$$\theta = \sum_{i=0}^{n-1} u(i)\, a(i) + \varepsilon$$

ε : an angle approximation error

$$|\varepsilon| \leq a(n-1)$$

the direction of rotation angle

$$u(i) = +1 \text{ or } -1$$

$$z(0) = \theta$$

$$z(i+1) = z(i) - u(i)\, a(i) \qquad i = 0, \cdots, n-1$$

$$u(i) = \text{sign}(z(i))$$

Initialization $x(0) = x$, $y(0) = y$

For $i = 0$ to $n-1$ do

CORDIC Rotation

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & \tan u(i)a(i) \\ -\tan u(i)a(i) & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$$

scaling Operation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \prod_{i=0}^{n-1} \cos u(i)a(i) \cdot \begin{bmatrix} x(n) \\ y(n) \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} \leftarrow \begin{bmatrix} x(n) \\ y(n) \end{bmatrix} \leftarrow \cdots \leftarrow \begin{bmatrix} x(1) \\ y(1) \end{bmatrix} \leftarrow \begin{bmatrix} x(0) \\ y(0) \end{bmatrix}$$

shift and add operations

$$\prod_{i=0}^{N-1} \cos u(i)a(i) = \frac{1}{K(n)}$$ norm correction

a known constant

once the set $\{u(i)a(i) : i = 0, \cdots, n-1\}$ is determined

a multiplier recoding method can be applied

Booth's algorithm.

For convenience, assume

$$|\theta| < 2a(0) = \frac{\pi}{2}$$

(a) if $\theta > 2\pi$, $\quad \theta \leftarrow \theta \bmod 2\pi$

(b) if $2\pi > \theta > \pi$, $\quad \begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} -x \\ -y \end{bmatrix}$, $\quad \theta \leftarrow \theta - \pi$

(c) if $\pi > \theta > \frac{\pi}{2}$ $\quad \begin{bmatrix} x \\ y \end{bmatrix} \leftarrow \begin{bmatrix} y \\ -x \end{bmatrix}$, $\quad \theta \leftarrow \theta - \frac{\pi}{2}$

CORDIC Angle Recoding Problem

$u(i) = 0$ is allowed. (repetition)

$$u(i) = \begin{cases} +1 \\ 0 \\ -1 \end{cases}$$

desirable to minimize $\sum_{i=0}^{n-1} |u(i)|$

→ reduce CORDIC iterations

← greedy algorithm

} Angle Recoding

given $a(i)$, $i = 0, \cdots, n-1$ $\quad a(i) \in EAS$

θ an angle

find $u(i)$, $i = 0, \cdots, n-1$ $\quad u(i) \in \{-1, 0, +1\}$

such that

(i) $\theta = \sum_{i=0}^{n-1} u(i)\, a(i) + \varepsilon \qquad \varepsilon < a(n-1)$

(ii) $\sum_{i=0}^{n-1} |u(i)|$ is minimized

CORDIC Angle Recoding Algorithm

Greedy Algorithm

Initialization : $\theta(0) = \theta$, $\{u(i) = 0,\ 0 \le i \le n-1\}$, $k = 0$

Repeat until $|\theta(k)| < a(n-1)$ Do

① choose i_k, $0 \le i_k \le n-1$ such that

$$\left| |\theta(k)| - a(i_k) \right| = \min_{0 \le i \le n-1} \left| |\theta(k)| - a(i) \right|$$

② $\theta(k+1) = \theta(k) - u(i_k)\, a(i_k)$

$u(i_k) = \text{sign}(\theta(k))$

at every step, represent the remaining angle

using a closest elementary CORDIC angle

→ greedy

→ minimizes $\sum_{i=0}^{n-1} |u(i)|$

draw the closest angle $a(i_k)$ without replacement

initialize $\theta_0 = \theta$

$\sigma_i = 0 \qquad i = 0, 1, \cdots, n-1$

$k = 0$

repeat until $|\theta_k| < \tan^{-1}(2^{-n+1})$ do

1. choose i_k, $i_k = 0, 1, 2, \cdots, n-1$
   such that
   $$\left| |\theta_k| - \tan^{-1}(2^{-i_k}) \right| = \min_{i \in [0:n-1]} \left| |\theta_k| - \tan^{-1}(2^{-i}) \right|$$

2. $\theta_{k+1} = \theta_k - \sigma_{i_k} \tan^{-1}(2^{-i_k})$

   $\sigma_{i_k} = \text{sign}(\theta_k)$

Angle Replacement

choosing i_k, $\boxed{0 \le i_k \le n-1}$

(1) with angle replacement

always choose i_k
among n numbers $(0 \dots n-1)$

angle repetition is allowed

$\boxed{\theta(k+1) \le \theta(k)}$ monotonically decreasing

(2) without angle replacement

at each step choose i_k
among one less numbers
than the previous step's numbers

$\boxed{\theta(k+1) < \theta(k)}$ Strictly decreasing

* the greedy algorithm

Strictly Decreasing

$$\left| |\theta(k)| - a(i_k) \right| = \min_{0 \le i \le n-1} \left| |\theta(k)| - a(i) \right|$$

$$\theta(k+1) = \theta(k) - u(i_k)\, a(i_k)$$

$$\begin{aligned} |\theta(k+1)| &= |\theta(k) - u(i_k)\, a(i_k)| \\ &= \left| |\theta(k)| - a(i_k) \right| \\ &\le \left| |\theta(k)| - a(n-1) \right| \\ &< |\theta(k)| \end{aligned}$$

loop while
$|\theta(k)| > a(n-1)$
$a(i_k) > a(n-1)$

Strictly decreasing

Angle Update

$i = 0, 1, 2, \cdots, n-1$ ⟵ n-bit word

$|\theta(k)| < a(n-1)$ termination condition

$k = 0, 1, \cdots, k'-1$ hopefully less than $n-1$

$k=0 \quad 0 \le i_0 \le n-1$

$k=1 \quad 0 \le i_1 \le n-1$

$\vdots$

$k=k'-1 \quad 0 \le i_{k'-1} \le n-1$

$u(i) = 0$ initialization $i = 0, 1, \cdots, n-1$

$\theta(k) > 0 \quad u(i_k) = +1 \quad \theta(k+1) = \theta(k) - a(i_k)$

find i_k s.t. $\min_{0 \le i \le n-1} |\theta(k) - a(i)|$

$\theta(k) < 0 \quad u(i_k) = -1 \quad \theta(k+1) = -\theta(k) + a(i_k)$

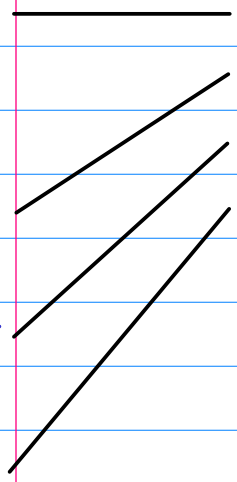
find i_k s.t. $\min_{0 \le i \le n-1} |\theta(k) - a(i)|$

$u(j) = 0 \quad j \in \{0, 1, \cdots, n-1\}$ initialization

$j \notin \{i_0, i_1, \cdots, i_{k'-1}\}$

repetition allowed in MVR

no repeatition



0	+ a(0)	θ(0)	u(0) = +1	a(0) = atan(2^{-0})
1	− a(3)	θ(1)	u(1) = 0	a(1) = atan(2^{-1})
2	− a(5)	θ(2)	u(2) = 0	a(2) = atan(2^{-2})
3	+ a(7)	θ(3)	u(3) = -1	a(3) = atan(2^{-3})
4			u(4) = 0	a(4) = atan(2^{-4})
5			u(5) = -1	a(5) = atan(2^{-5})
6			u(6) = 0	a(6) = atan(2^{-6})
7			u(7) = +1	a(7) = atan(2^{-7})
8			u(8) = 0	a(8) = atan(2^{-8})
9			u(9) = 0	a(9) = atan(2^{-9})
10			u(10) = 0	a(10) = atan(2^{-10})
11			u(11) = 0	a(11) = atan(2^{-11})
12			u(12) = 0	a(12) = atan(2^{-12})
13			u(13) = 0	a(13) = atan(2^{-13})
14			u(14) = 0	a(14) = atan(2^{-14})
15			u(15) = 0	a(15) = atan(2^{-15})

k= 0 theta= 0.6376100 ik= 0 uik=+1 a[0]= 0.7853982, new theta=-0.1477882
k= 1 theta=-0.1477882 ik= 3 uik= -1 a[3]= 0.1243550, new theta=-0.0234332
k= 2 theta=-0.0234332 ik= 5 uik= -1 a[5]= 0.0312398, new theta= 0.0078067
k= 3 theta= 0.0078067 ik= 7 uik=+1 a[7]= 0.0078123, new theta=-0.0000057

i= 0 u[0]=+1 a[0]=0.785398 angle=0.785398
i= 1 u[1]=+0 a[1]=0.463648 angle=0.785398
i= 2 u[2]=+0 a[2]=0.244979 angle=0.785398
i= 3 u[3]= -1 a[3]=0.124355 angle=0.661043
i= 4 u[4]=+0 a[4]=0.062419 angle=0.661043
i= 5 u[5]= -1 a[5]=0.031240 angle=0.629803
i= 6 u[6]=+0 a[6]=0.015624 angle=0.629803
i= 7 u[7]=+1 a[7]=0.007812 angle=0.637616
i= 8 u[8]=+0 a[8]=0.003906 angle=0.637616
i= 9 u[9]=+0 a[9]=0.001953 angle=0.637616
i=10 u[10]=+0 a[10]=0.000977 angle=0.637616
i=11 u[11]=+0 a[11]=0.000488 angle=0.637616
i=12 u[12]=+0 a[12]=0.000244 angle=0.637616
i=13 u[13]=+0 a[13]=0.000122 angle=0.637616
i=14 u[14]=+0 a[14]=0.000061 angle=0.637616
i=15 u[15]=+0 a[15]=0.000031 angle=0.637616

Elementary Angle Set

$$S = \{ (\sigma \cdot \tan^{-1}(2^{-r})) : \sigma \in \{+1, -1\}, \; r \in \{1, 2, \cdots, n-1\} \}$$

n-bit angle as a linear combination

$$\theta = \sum_{i=0}^{n-1} \sigma_i \cdot \tan^{-1}(2^{-i})$$

AR : $\sigma \in \{-1, 0, +1\}$

EAS (Elementary Angle Set) for AR methods

$$S_{EAS} = \{ (\sigma \cdot \tan^{-1}(2^{-r})) : \sigma \in \{+1, 0, -1\}, \; r \in \{1, 2, \cdots, n-1\} \}$$

Simple angle recording — Hu's greedy algorithm

tries to represent the remaining angle using the closest elementary angle $\pm \tan^{-i}$

- rotation mode — Angle Recording
- vectoring mode — Backward Angle Recording (BAR)

Scaling - Norm Correcting Rotation

$$\prod_{i=0}^{n-1} \cos u(i)\, a(i) = 1/K(n)$$

a modified Booth's multiplier recoding representation for $1/K(n)$

the total number of 1's and $\bar{1}$'s

→ additional iterations needed for CORDIC scaling (norm correction operation)

the total number of iterations

for CORDIC rotation operations
for norm correction operations

Properties

if the algorithm terminates at $k = k^*$, $k^* < \frac{n}{2}$

$$g(i) = a(i) - a(i+1) \qquad i = 0, 1, \cdots, n-2$$
$$a(i) = \tan^{-1} 2^{-i}$$

① $g(i) > 0$

② $a(i+2) < g(i) < a(i+1)$

③ $g(i) > g(i+1)$

① $a(i) - a(i+1) > 0$

② $a(i+2) < a(i) - a(i+1) < a(i+1)$

③ $a(i) - a(i+1) > a(i+1) - a(i+2)$

if $|\theta| \leq a(0) = \frac{\pi}{4}$

$$\sum_{i=0}^{n-1} |u(i)| < \frac{n}{2}$$

$g(i) > 0$

$a(i+2) < g(i) < a(i+1)$

$g(i) > g(i+1)$

$g(i) = a(i) - a(i+1) \qquad i = 0, 1, \cdots, n-2$

$a(i) = \tan^{-1}(2^{-i})$

$g(i) > 0 \qquad a(i) > a(i+1) \qquad$ strictly decreasing

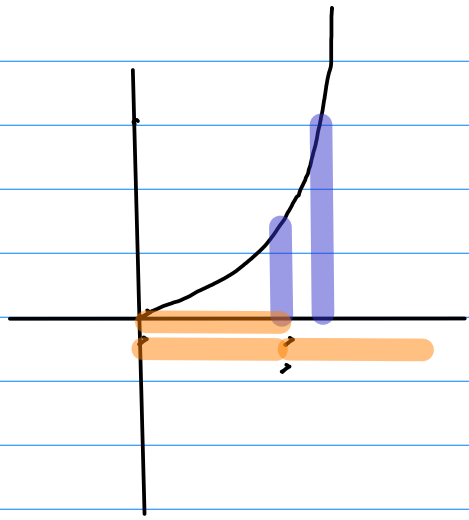
$g(i) = a(i) - a(i+1) = \tan^{-1}(2^{-i}) - \tan^{-1}(2^{-i-1}) > 0$

$g(i+1) = a(i+1) - a(i+2) = \tan^{-1}(2^{-i-1}) - \tan^{-1}(2^{-i-2}) > 0$

$g(i) - g(i+1) = a(i) - 2a(i+1) + a(i+2) > 0$

$$\tan(a(i)) = \tan(\tan^{-1}(2^{-i})) = 2^{-i}$$
$$\tan(a(i+1)) = \tan(\tan^{-1}(2^{-i-1})) = 2^{-i-1}$$

$$\tan(a(i)) = 2 \cdot \tan(a(i+1))$$



$$a(i) < 2 \cdot a(i+1) < 2\,a(i)$$

$$a(i+2) < g(i) = a(i) - a(i+1) < a(i+1)$$

$$a(i) < 2 \cdot a(i+1)$$

Total Iteration Number k^*

$$|\theta| \le a(0) = \frac{\pi}{4}$$

$$\sum_{i=0}^{n-1} |u(i)| \le \frac{\pi}{2}$$

$$i_k \ge 2k$$

$$a(n-1) < |\theta| < a(0)$$

$$|\theta| \in [a(i_0+1),\ a(i_0)] \quad \text{or}$$
$$|\theta| \in [a(i_0),\ a(i_0-1)]$$

$$\theta(1) = \big||\theta(0)| - a(i_0)\big|$$

$$< \max\{g(i_0)/2,\ g(i_0-1)/2\}$$
$$< g(0)/2$$
$$< a(1)/2$$
$$< a(2)$$

$k = 1$

$$|\theta(k)| < a(2k)$$

$$|\theta(k+1)| < g(2k)/2$$
$$< a(2k+1)/2$$
$$< a(2k+2)$$
$$= a(2(k+1))$$

$2k \ge n-1$ $\qquad k^* \le \frac{(n-1)}{2}$

```
#include <stdio.h>
#include <math.h>

#define N 16


int find_index(double a[], double atheta) {
  int i, ik = N;
  double minval = 1.0e+10;

  for (i=0; i<N; ++i) {
    if (minval > fabs(atheta - a[i])) {
      ik = i;
      minval = fabs(atheta - a[i]);
    }
  }

  return ik;
}

int main(void) {
  double a[N];
  double theta = 0.63761;
  int ik, uik;
  int k, i;
  int u[N];
  double angle;


  for (i=0; i<N; ++i)
    a[i] = atan(1./pow(2, i));

  for (i=0; i<N; ++i)
    u[i] = 0;
```

```
k = 0;
while ((fabs(theta) >= a[N-1]) && (k < N)) {
  ik = find_index(a, fabs(theta));

  uik = (theta >= 0) ? +1 : -1;

  printf("k=%2d theta=%10.7f ", k, theta);
  printf("ik=%2d uik=%+d ", ik, uik);

  theta = theta - uik * a[ik];

  printf("a[%2d]=%10.7f, new theta=%10.7f \n",
  ik, a[ik], theta);

  u[ik] = uik;

  k++;
}


angle = 0.0;
for (i=0; i<N; ++i) {
  angle += u[i] * a[i];

  printf("i=%2d u[%2d]=%+d a[%2d]=%f angle=%f \n",
  i, i, u[i], i, a[i], angle);
}




}
```

k= 0 theta= 0.6376100 ik= 0 uik=+1 a[0]= 0.7853982, new theta=-0.1477882
k= 1 theta=-0.1477882 ik= 3 uik=-1 a[3]= 0.1243550, new theta=-0.0234332
k= 2 theta=-0.0234332 ik= 5 uik=-1 a[5]= 0.0312398, new theta= 0.0078067
k= 3 theta= 0.0078067 ik= 7 uik=+1 a[7]= 0.0078123, new theta=-0.0000057
i= 0 u[0]=+1 a[0]=0.785398 angle=0.785398
i= 1 u[1]=+0 a[1]=0.463648 angle=0.785398
i= 2 u[2]=+0 a[2]=0.244979 angle=0.785398
i= 3 u[3]=-1 a[3]=0.124355 angle=0.661043
i= 4 u[4]=+0 a[4]=0.062419 angle=0.661043
i= 5 u[5]=-1 a[5]=0.031240 angle=0.629803
i= 6 u[6]=+0 a[6]=0.015624 angle=0.629803
i= 7 u[7]=+1 a[7]=0.007812 angle=0.637616
i= 8 u[8]=+0 a[8]=0.003906 angle=0.637616
i= 9 u[9]=+0 a[9]=0.001953 angle=0.637616
i=10 u[10]=+0 a[10]=0.000977 angle=0.637616
i=11 u[11]=+0 a[11]=0.000488 angle=0.637616
i=12 u[12]=+0 a[12]=0.000244 angle=0.637616
i=13 u[13]=+0 a[13]=0.000122 angle=0.637616
i=14 u[14]=+0 a[14]=0.000061 angle=0.637616
i=15 u[15]=+0 a[15]=0.000031 angle=0.637616