CORDIC Accuracy
Octave Programming

20151023

```
function b = binary(n)

  nn = 2^n;
  a = dec2bin(0:nn-1);

  b = zeros(nn, n);

  for i=1:nn
    for j=1:n
      if (a(i,j) == '1')
    b(i,j) = +1;
      else
    b(i,j) = -1;
      endif
    endfor
  endfor
```

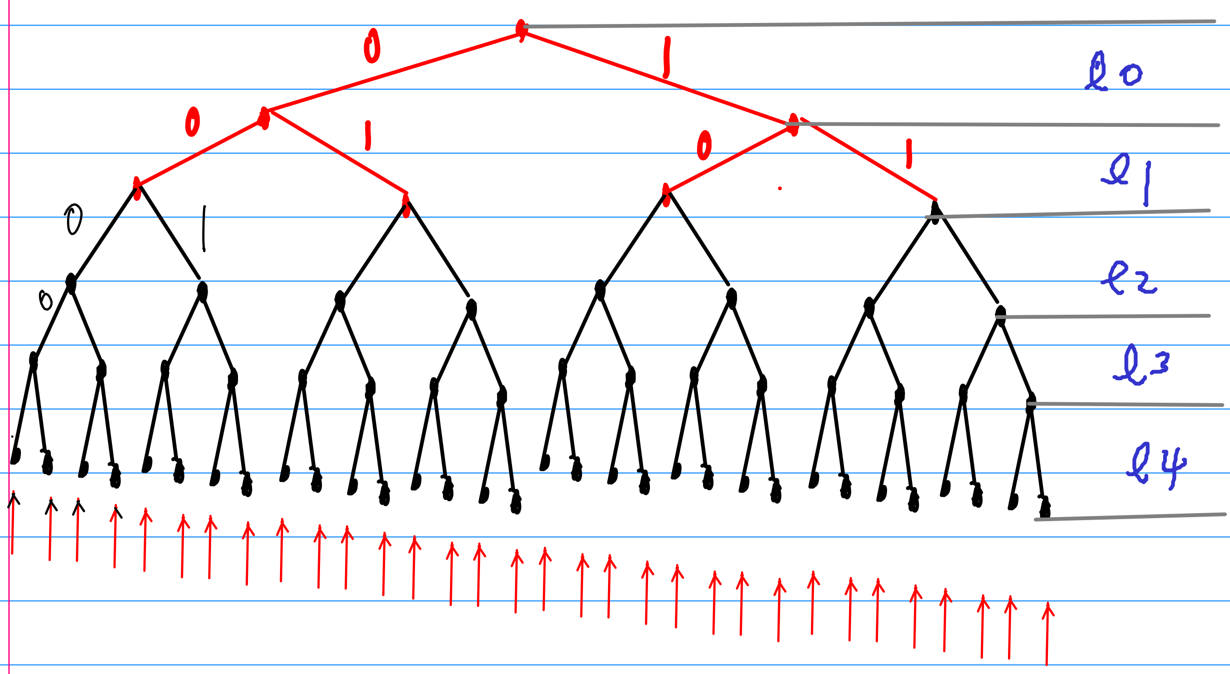
```
function A = angles(n)

  nn = 2^n;

  b = binary(n);


  L = 0:n-1;
  K = 2 .^ (-L);

  theta = atan(K);

  for i=1:nn
    A(i) = sum( theta .* b(i, :) ) ;
  endfor

  A = A';
```

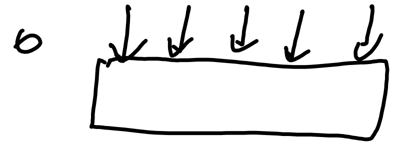
```
if (nAngles == (1 << nIters)) {
  Leaf = 1;
  cout << "A LeafAngles Object is created " ;
} else {
  Leaf = 0;
  cout << "An AllAngles Object is created " ;
}
```

$nAngles = 2^{nIters}$

$1024 = 2^{10}$

$32 = 2^{5}$





```c
angle = 0.0;
for (i=0; i<level; i++) {
  j = 1 << i;
  if (idx & (1 << ((level-i-1))) {
    angle += atan( 1. / j );
    s[i] = '1';
  } else {
    angle -= atan( 1. / j );
    s[i] = '0';
  }
 }
s[i] = '\0';
```

i=0	j=1	4	2^4	atan(1/1)	
i=1	j=2	3	2^3	atan(1/2)	00001
i=2	j=4	2	2^2	atan(1/4)	00010
i=3	j=8	1	2^1	atan(1/8)	00100
i=4	j=16	0	2^0	atan(1/16)	01000
					10000

```
function angles(n)

  nn = 2^n;

  b = binary(n);
  c = 2*b - 1;

  % disp(b);
  % disp(c);

  L = 0:n-1;
  K = 2 .^ (-L);

  theta = atan(K);

  % disp(theta');

  for i=1:nn
    A(i) = sum( theta .* c(i, :) ) ;
  endfor

  A = A';

  %%{
  for i=1:nn
    printf("A(%d) \t= %20.15f b= ", i, A(i));
    printf("%d", b(i,:));
    printf("\n");
  endfor
  %%}
```

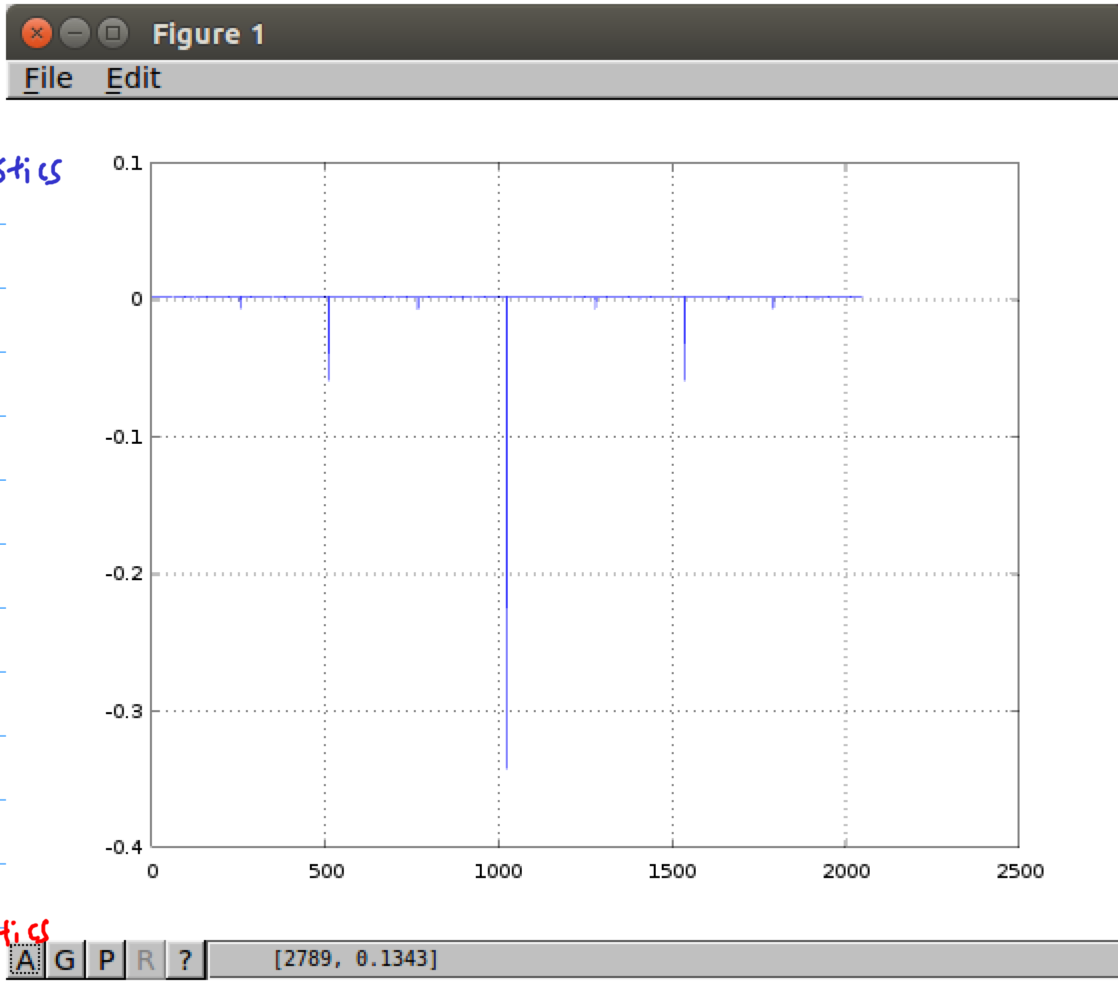
```
THETA = [
  7.8539816339744830962E-01,
  4.6364760900080611621E-01,
  2.4497866312686415417E-01,
  1.2435499454676143503E-01,
  6.2418809995957348474E-02,
  3.1239833430268276254E-02,
  1.5623728620476830803E-02,
  7.8123410601011112965E-03,
  3.9062301319669718276E-03,
  1.9531225164788186851E-03,
  9.7656218955931943040E-04,
  4.8828121119489827547E-04,
  2.4414062014936176402E-04,
  1.2207031189367020424E-04,
  6.1035156174208775022E-05,
  3.0517578115526096862E-05,
  1.5258789061315762107E-05,
  7.6293945311019702634E-06,
  3.8146972656064962829E-06,
  1.9073486328101870354E-06,
  9.5367431640596087942E-07,
  4.7683715820308885993E-07,
  2.3841857910155798249E-07,
  1.1920928955078068531E-07,
  5.9604644775390554414E-08,
  2.9802322387695303677E-08,
  1.4901161193847655147E-08,
  7.4505805969238279871E-09,
  3.7252902984619140453E-09,
  1.8626451492309570291E-09,
  9.3132257461547851536E-10,
  4.6566128730773925778E-10,
  2.3283064365386962890E-10,
  1.1641532182693481445E-10,
  5.8207660913467407226E-11,
  2.9103830456733703613E-11,
  1.4551915228366851807E-11,
  7.2759576141834259033E-12,
  3.6379788070917129517E-12,
  1.8189894035458564758E-12,
  9.0949470177292823792E-13,
  4.5474735088646411896E-13,
  2.2737367544323205948E-13,
  1.1368683772161602974E-13,
  5.6843418860808014870E-14,
  2.8421709430404007435E-14,
  1.4210854715202003717E-14,
  7.1054273576010018587E-15,
  3.5527136788005009294E-15,
  1.7763568394002504647E-15,
  8.8817841970012523234E-16,
  4.4408920985006261617E-16,
  2.2204460492503130808E-16,
  1.1102230246251565404E-16,
  5.5511151231257827021E-17,
  2.7755575615628913511E-17,
  1.3877787807814456755E-17,
  6.9388939039072283776E-18,
  3.4694469519536141888E-18,
  1.7347234759768070944E-18 ]' ;
```

```
%{
    for i=1:n
        delta = THETA(i) - theta(i);
        printf("T(%d)= %f ", i, THETA(i));
        printf("t(%d)= %f ", i, theta(i));
        printf("delta= %20.16f ", delta);
        printf("\n");
    endfor
%}

        A = sort(A);

        for i=1:nn-1
          diff(i) = A(i+1) - A(i);
        endfor


        plot(1:nn-1, diff);

        d = sort(diff')

        % plot(1:nn-1, d);
```

Difference Statistics

before sort (A)

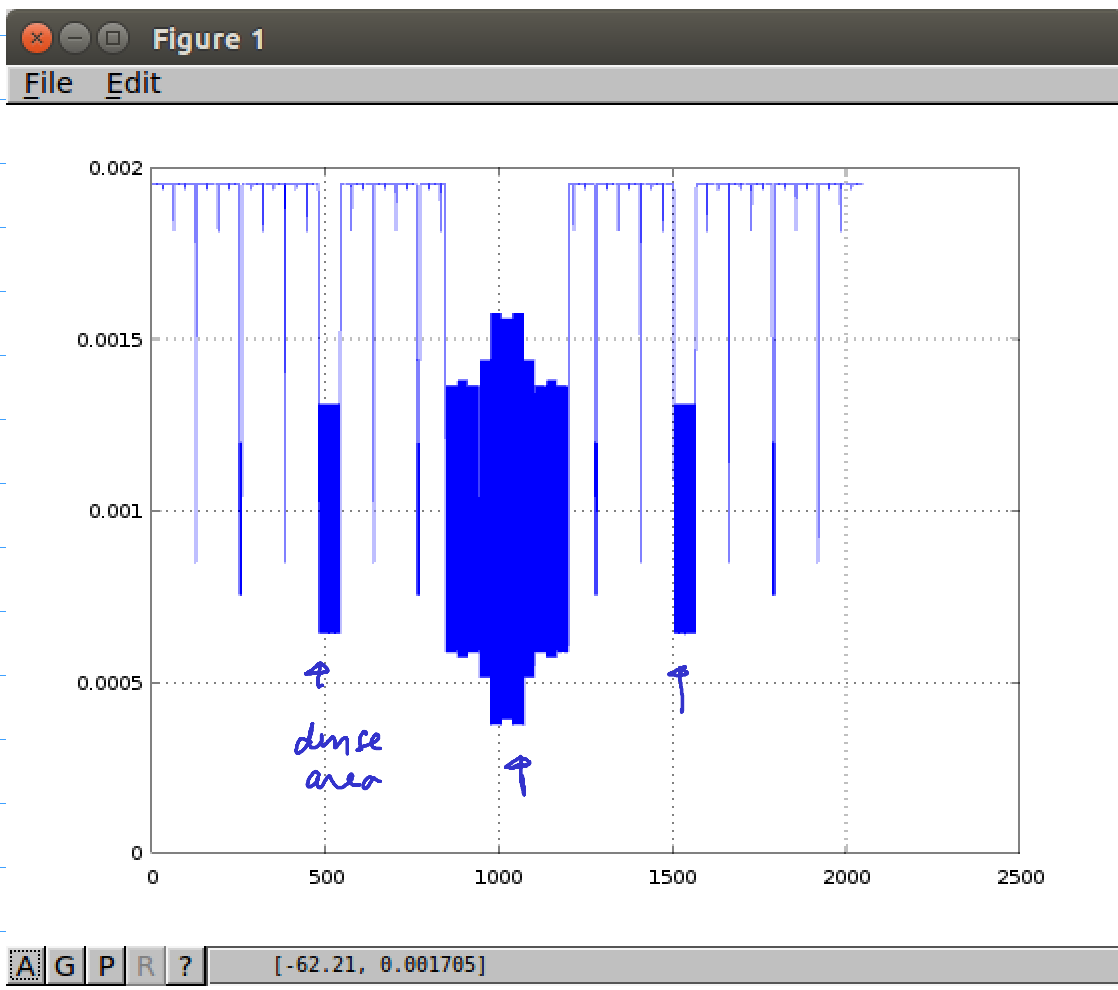
original index order



Difference statistics

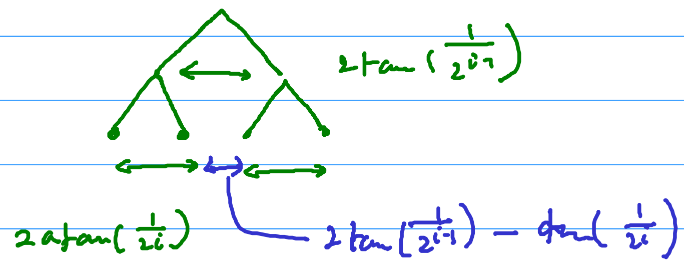
after sort(A)

increasing angle order



* reason?

$\pm \text{atan}\left(\frac{1}{2^i}\right)$



For a fixed point simulation

```
% plot(1:nn-1, diff);

d = sort(diff');

% plot(1:nn-1, d);


disp(theta');
mintheta = theta(n);
theta(1:n) = int16( theta(1:n) / mintheta);
disp(theta');

A(1:nn) = A(1:nn) / mintheta;
B = int32( A - A(1) )
C = dec2bin(B)


%{
for i=1:nn
  printf("A(%d) \t= %d b= ", i, dec2bin(int32(A(i)-A(1))));
  printf("%d", b(i,:));
  printf("\n");
endfor
%}
```

minimum angle spacing → resolution?

what is the representative angle spacing values

choose min theta and divide angle values

by this min value.

and convert this into an integer / binary

number

```
octave:7> angles(5)
   0.785398
   0.463648
   0.244979
   0.124355
   0.062419
   13
    7
    4
    2
    1
```

B =

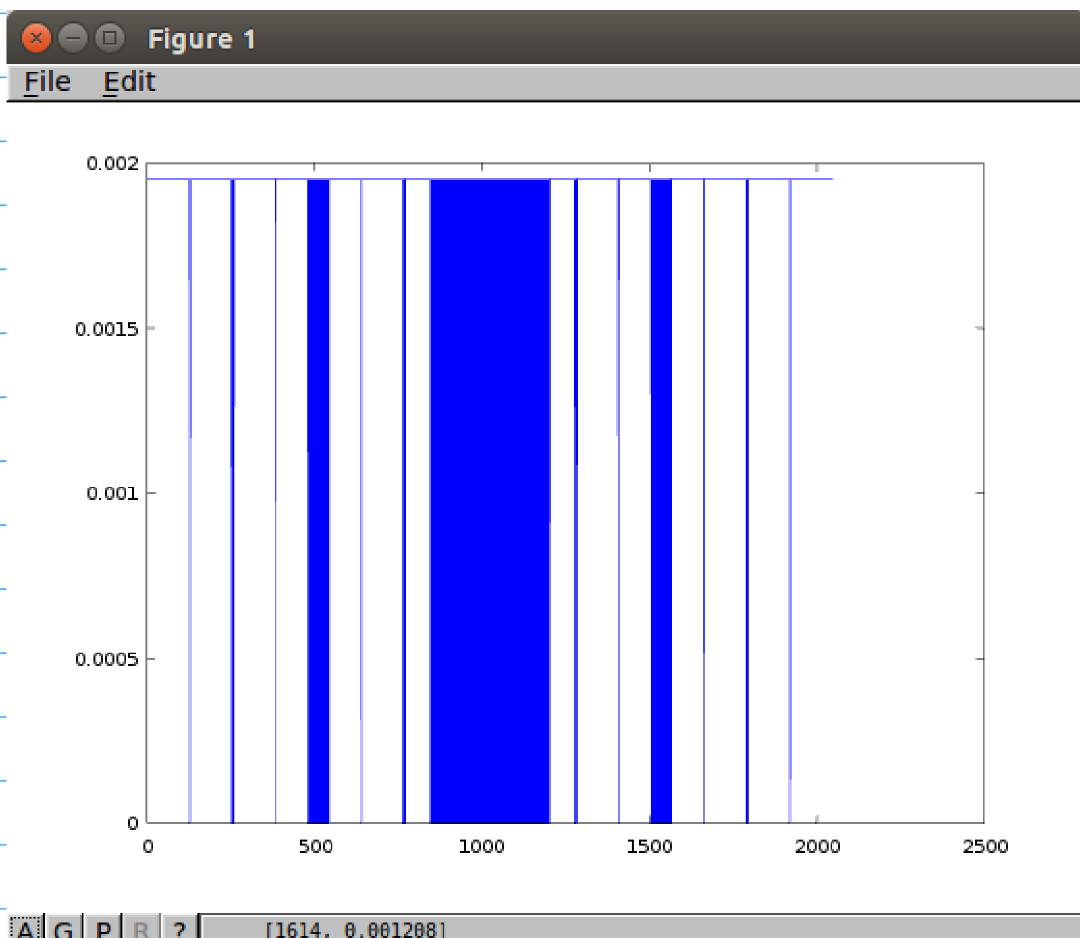
```
 0
 2
 4
 6
 8
10
12
14
15
17
19

39
40
42
44
46
48
50
52
54
```

C =

```
000000
000010
000100
000110
001000
001010
001100
001110
001111
010001

100111
101000
101010
101100
101110
110000
110010
110100
110110
```

```
function B = fixednum(A, minnum)

  B = double(int32(A/minnum)) * minnum;
```

```
L = 0:n-1;
K = 2 .^ (-L);

theta = atan(K);

theta = fixednum(theta, theta(n));
```



$-\frac{\pi}{2} \sim +\frac{\pi}{2}$

k = linspace (-1, 1, 20);

$$degree = \frac{\pi}{2} \times k \times \frac{180}{\pi}$$

minimum angle spacing → resolution ?

what is the representative angle spacing values

linear angle   use bin num

choose min theta and divide angle values

by this min value.

and convert this into an integer / binary

number

representative angle spacing value?