Lecture 2 : ~~[illegible]~~

S. A. Tretter, "Comm System Design using DSP Algorithms," Kluwer 2003, New York

1st Lab due Thurs, ~~Jan~~ ~~Feb 18~~ ~~Thurs~~, ~~Jan 29~~

Finite # Systems

Finite Number Systems

integer = 32-bit word

$$\longleftrightarrow (d_{31} \ldots d_1 d_0)$$

MSB LSB

$$0 \le V = \sum_{n=0}^{31} d_n 2^n \le 2^{32} - 1$$

(2's complement) **signed integer** : MSB is sgn bit :

: $d_{31} = 0$ +

: $d_{31} = 1$ −

$$-2^{31} \le V = -d_{31} 2^{31} + \sum_{n=0}^{30} d_n 2^n \le 2^{31} - 1$$

Q_L number : move radix pt from rt of d_0 to L^{th} location

: programmer keeps track

$$Q_L \# \longleftrightarrow (d_{31} \ldots d_L \cdot d_{L-1} \ldots d_0)$$

$$V = 2^{-L} \left(-d_{31} 2^{31} + \sum_{n=0}^{30} d_n 2^n\right)$$

↑ shift left by L

NB : "integer" of this type occupy one 32-bit register

IEEE Floating Pt. (single & dbl precision)

normal #s

subnormal (< smallest pos normal)

NaNs

∞ #s

fp (single) = 32-bit word

$\leftrightarrow (s\ e_7 \ldots e_0\ f_1 \ldots f_{23})$

sign — exp — mantissa

Let's define some intermediate vars:

$$e = \sum_{k=0}^{7} e_k 2^k \qquad f = \sum_{k=1}^{23} f_k 2^{-k} \qquad s \in \{0,1\}$$

$$0 \le e \le 255 \qquad 0 \le f \le 1 - 2^{-23}$$

(Convention or code)
1. if $e = 255$ $(11\ldots1)$ & $f \neq 0$, then X = NaN, indep of s

(Conv or code for $\pm\infty$)
2. if $e = 255$ $(11\ldots1)$ & $f = 0$ $(0\ldots0)$, then $X = (-1)^s \infty$ $[\pm\infty]$

3. if $0 < e < 255$, then X is a normal # with value

NB: $2^{-23} \simeq 0.00000011192$

$\leftrightarrow 1.f_1 \ldots f_{23} \leftrightarrow \sum_{0}^{23} f_k 2^{-k}$; $f_0 = 1$

$$X = (-1)^{s} \, 2^{e-127} \, (1+f)$$

sgn — move radix rt by 127 places or lft by 126 places

Insert Here

(notice f_1 not $(1+f)$)

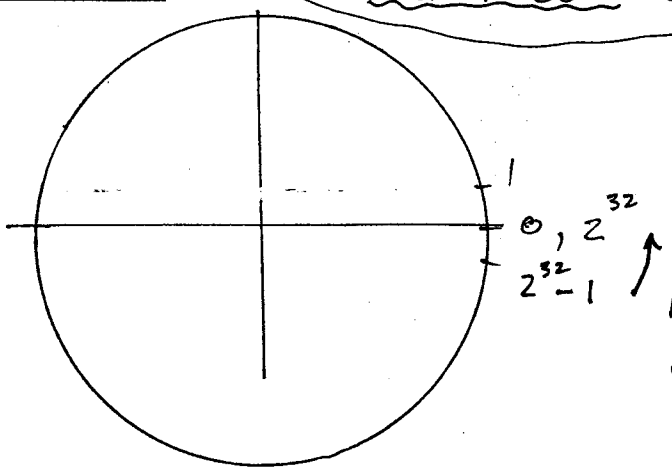
4. if $e = 0$ & $f \neq 0$, then $X = (-1)^{s} 2^{-126} f$ (subnormal)

NB: The subnormal #'s transition btwn 0 and the smallest minimum zone value in the same increments as the resolution of the smallest normal # zone. Systems that do not support subnormal numbers round these values to zero. This is known as a flush to zero operation

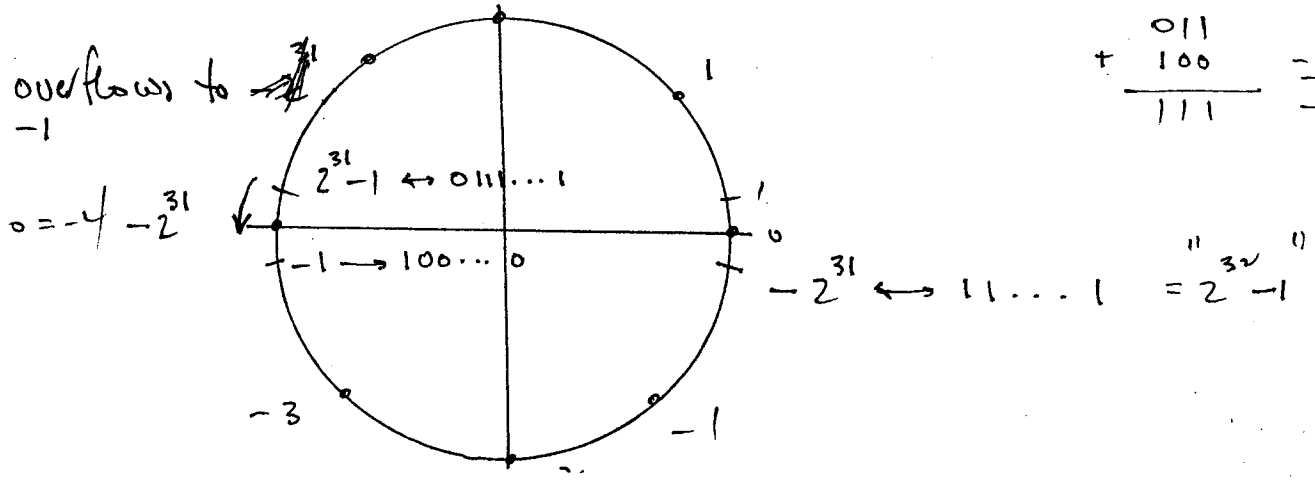
⇒ smallest positive number $(2^{-126})(2^{-23}) = 2^{-149} \approx 1.04 \times 10^{-45}$

5. if $e = 0$ & $f = 0$, then $X = (-1)^{s} \cdot 0 = \pm 0$

Overflow



11...1 + 0...01 overflows to zero

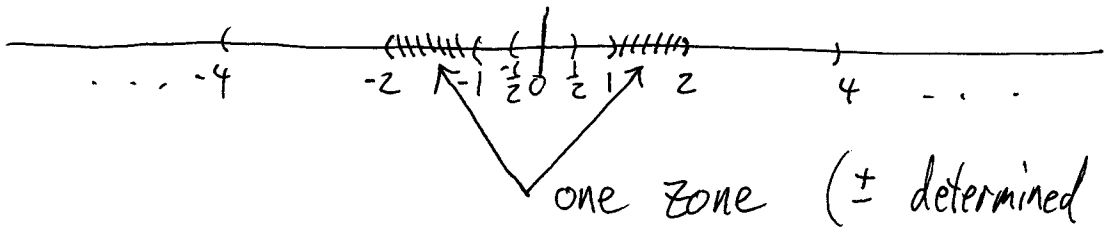


$$\begin{array}{r} 011 \\ +\ 100 \\ \hline 111 \end{array} \qquad \begin{array}{r} 3 \\ -4 \\ \hline -1 \end{array} = -2^{4} + 3$$

$-2^{31} \leftrightarrow 11\ldots1 = 2^{32} - 1$

IEEE Floating-Point (Insert)

We can think of the set of real numbers being split into zones



one zone ($\pm$ determined by sign bit)

$0 < e < 255$ (NB: $e = 0, 255$ are reserved)

$0 \le f \le 1 - 2^{-23}$ $\Rightarrow$ zone resolution $2^{-23} \approx 0.00000011 92$

If 2^a is the minimum zone value (associated with $f=0$), then the zone resolution is

$(2^{-23})\,2^a$ (NB: $a = e - 127$)

(Comment: $e=127$ and the minimum zone value is one in the example above)

The smallest normal number (and minimum resolution value) is $2^{-126} \approx 1.175 \times 10^{-38}$ and the zone resolution is $(2^{-23})(2^{-126}) \approx 1.04 \times 10^{-45}$

The largest minimum resolution value is $2^{127} \approx 1.7 \times 10^{38}$ and the zone resolution is $(2^{-23})(2^{127}) = 2^{104} \approx 2.03 \times 10^{31}$

There are always 2^{23} pts of resolution in a given zone and the larger the minimum zone value is, the larger the jumps btwn number is. Hence, worse resolution.

This means that for

really small #'s $0.000000000 \times\times\times\times\times\times$

small #'s $\times.\times\times\times\times\times$

large #'s $\times\times\times\times\times.\times\times$

really large #'s $\times\times\times\times\times\times\underbrace{0000000}_{\text{not necessarily zero, but will be in quantized levels}}.00$

(Hence floating-point)

NB: $2^{-23} \approx 0.00000011 92$

Have (accurate) resolution up to "6 decimal places" (or seven digits) ... see small #'s above and 9

(short and int have 5 decimal place resolution, respectively
max short = 32768
max int = 2147483648)

An aside

A number where the number of digits after the decimal point is fixed is known a fixed-point #.

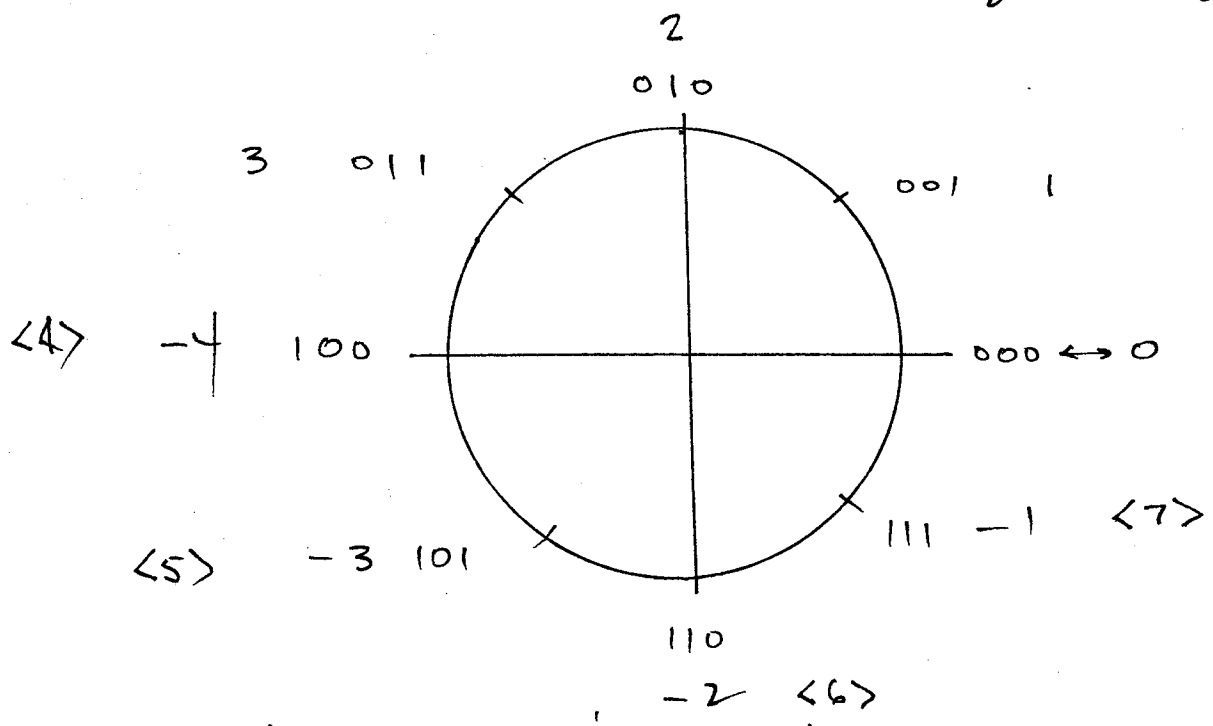
It is important to understand that fixed-point arithmetic is not limited to integer values (remember Qn formats). However, a fixed-point number may be scaled to an integer. In fact, this is required for processing. Upon process completion, the fixed-point number may be unscaled. This point will be clarified in the lectures for Lab 4: FIR Filter Design and Implementation.

Hopefully this explains the term "fixed-point approximation" to π, namely PI = 3.14159265359

2's Complement & Overflow

Illustrate for 3-bit word $(d_2\, d_1\, d_0)$

$$V = -d_2 2^2 + d_0 + d_1 2$$



Let's do binary addition in usual way

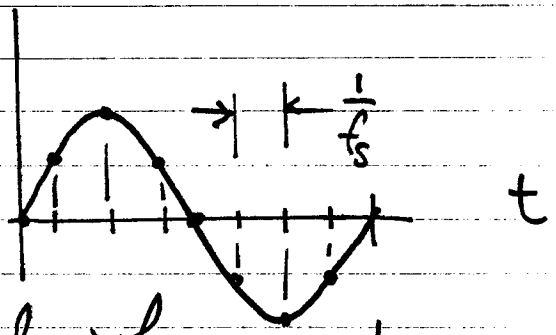
100	-4	011	3	
011	3	010	2	intermediate overflow to
100	-4	100	-4	101 = -3
011	3	001	1	(OK)
(OVERFLOW)				

Adds move you CCW thru virtual < > ≠
Subs move you CW thru virtual < >. Intermediate results can be bad but final results OK.

Remarks on Sampling

Consider a sinewave

$$x(t) = \sin 2\pi f_0 t$$



If we sample @ rate f_s, then $\frac{1}{f_0}$

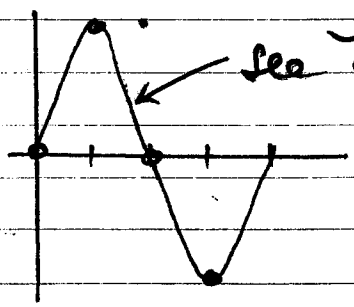
$$x[n] = \sin 2\pi \frac{f_0}{f_s} n = x\left(t = \frac{n}{f_s}\right) = x(t = nt_s)$$

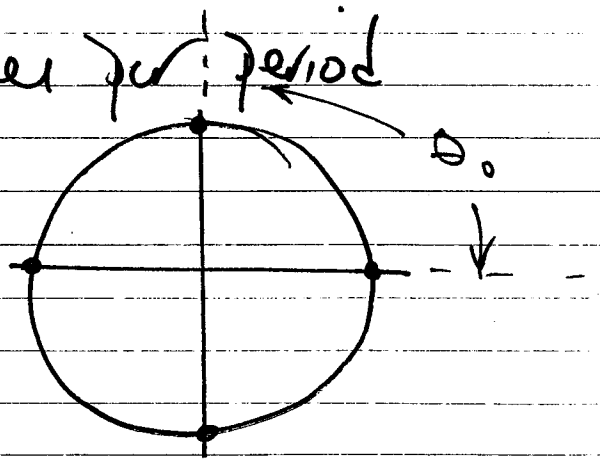
If $f_0 = \frac{1}{4} f_s$, then $\left(\frac{1}{f_0} = \frac{4}{f_s}\right)$

$$x[n] = \sin 2\pi \frac{1}{4} n \quad ; \quad \theta_0 = \frac{2\pi}{4}$$

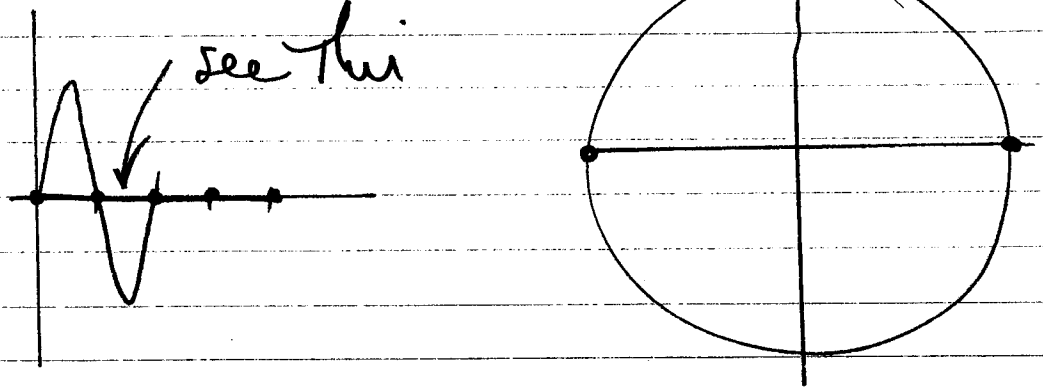
And we get 4 samples per period

0, 1, 0, 1



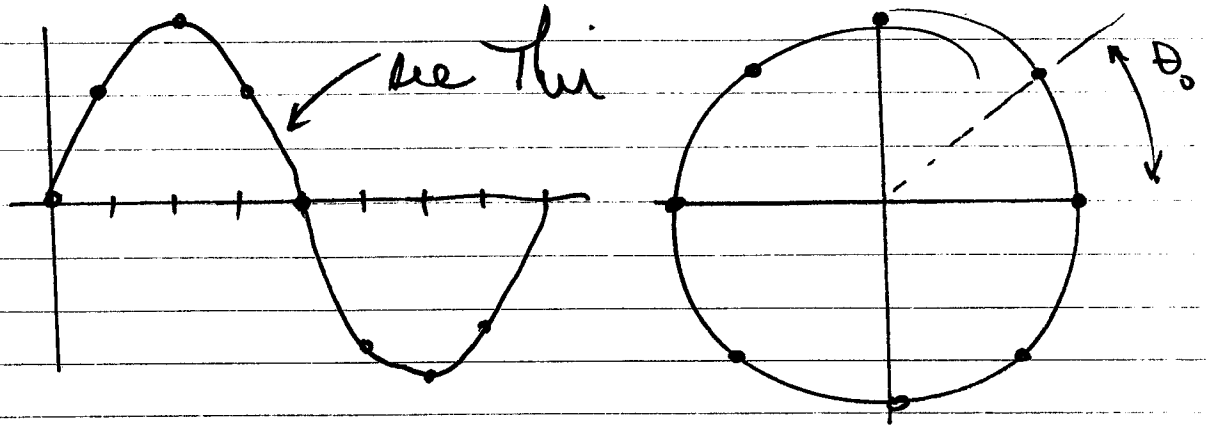


If $f_0 = f_s/2$, then $\frac{1}{f_0} = \frac{2}{f_s}$ and we get $x[n] = \sin 2\pi \frac{1}{2} n$ for 2 samples per period ; $\theta_0 = \frac{2\pi}{2}$



If $f_0 = f_s/8$, then $\frac{1}{f_0} = \frac{8}{f_s}$ and

$x[n] = \sin 2\pi \frac{1}{8} n$ and 8 samples/per



$\theta_0 = \frac{2\pi}{8}$

You will see dots in your code & contin curve on your oscilloscope

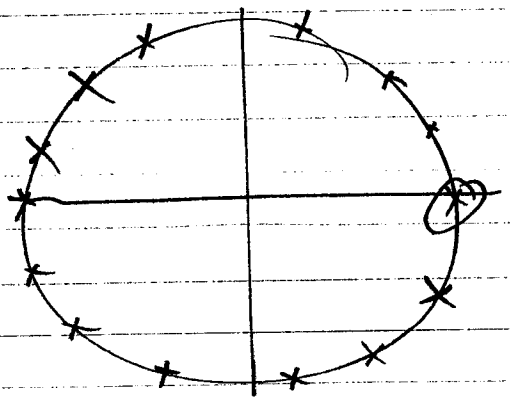
If $f_0 = \frac{p}{q} f_{AS}$, then

$$\theta_0 = \frac{p}{q}$$

and we go q samples to get $p\,2\pi$

$$\sin 2\pi \frac{p}{q} n$$

$p/q = \frac{7}{16}$



If $f_0 = \pi f_s$, then sampler never overwrite.

With "PI = 3.14159..." we don't actually overwrite the way the theory says.