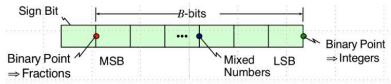
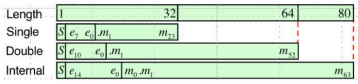
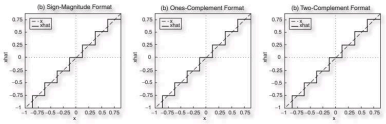
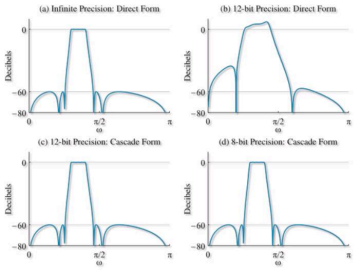
# Finite Word-Length Effects on

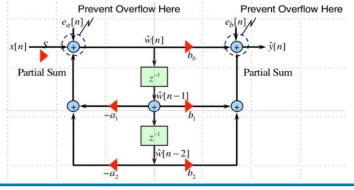
## Filter Coefficients and Arithmetic Operations






0

# Practical Aspects of DSP

### DSP in Theory

All signals, filter coefficients, twiddle factors, other quantities, and the results of any computations, can assume *any* value, i.e., they can be represented with *infinite* accuracy.

### DSP in Practice

- Any number can be represented in a digital computer or other digital hardware using a finite number of binary digits (bits), i.e., with *finite* accuracy.
- Each number is represented by a combination of a finite number of bits (0's or 1's) known as *binary word* or *word*.
- A word length of $B$ bits we can represent at most $2^B$ different values.



1

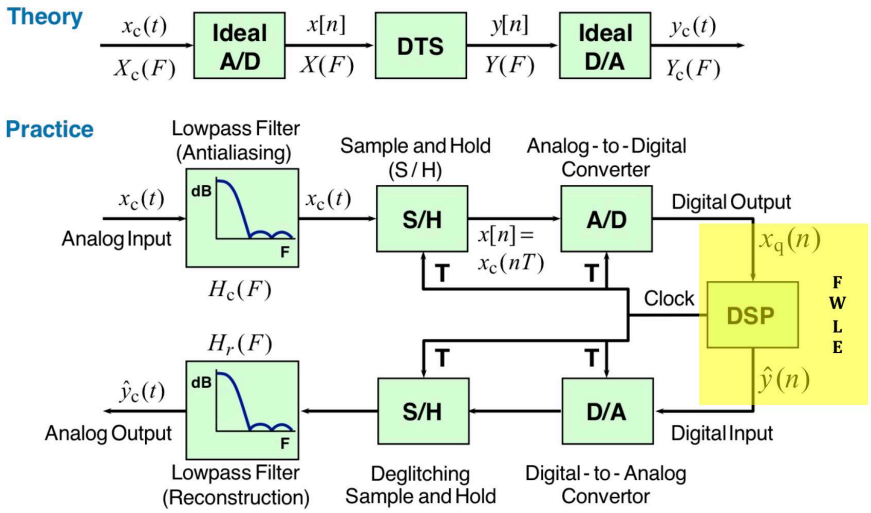
## Finite Wordlength Effects (FWLE)

1. Conversion of signals with a continuous-amplitude to signals with a discrete-amplitude, i.e., signals where each sample is represented by a $(B+1)$ bit word.
   - This leads to A/D quantization error or noise.
2. Conversion of filter coefficients, twiddle factors, and other fixed quantities into numbers described by a $(B+1)$ bit word.
   - In digital filters this results in a different frequency response!
3. Use of finite wordlength numbers to perform the arithmetic operations in a digital filter or DFT computations.
   - This introduces arithmetic noise or other undesirable effects (e.g., overflow oscillations or limit cycles)



2

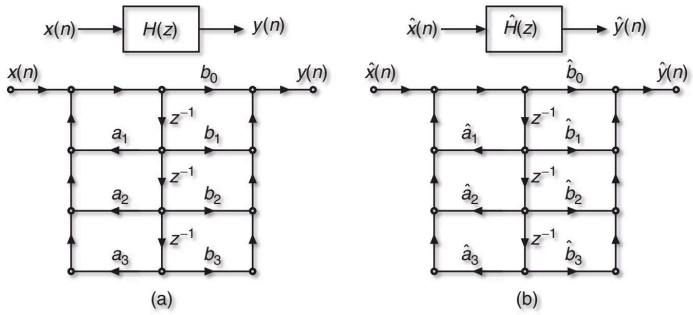
## Digital Processing of Analog Signals






3

## Finite Wordlength Effects






4

## Computers and Numbers

A decimal (base-10) number is a string of digits (0 through 9) with a decimal point. The value of each digit depends on its position relative to the decimal point, e.g.,

$$124.325_{10} = 1\times10^{2} + 2\times10^{1} + 4\times10^{0} + 3\times10^{-1} + 2\times10^{-2} + 5\times10^{-3}$$

In a computer, or any other digital hardware, numbers are represented by a string of binary digits or bits that take the values of 0 and 1.

The decimal value of a binary number is determined as

$$10.101_{2} = 1\times2^{1} + 0\times2^{0} + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} = 2.625_{10}$$

These bits are organized into groups of 8 bits called bytes or groups containing 16 or 32 bits called words

The location of the decimal or binary point determines the value of each bit. Therefore this representation of numbers is called fixed-point format.

In the floating-point format

$x = M\times10^{E}$ (decimal) or $x = M\times2^{E}$ (binary)

$M$ = Mantissa

$E$ = Exponent

where $E$ is an integer and $0.5 \le |M| < 1$

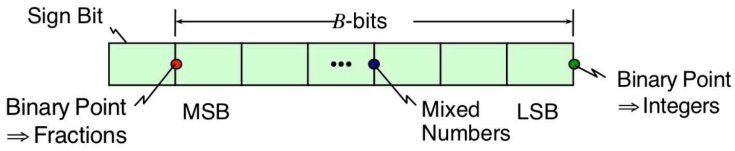
Decimal: $124.325_{10} \Rightarrow 1.24325\times10^{2}$

Binary: $10.101_{2} \Rightarrow 0.10101\times2^{4}$



5

## Binary Number Representations




- Mixed numbers and integers are difficult to multiply and the number of bits cannot be reduced by rounding or truncation ⇒ Focus on Fractions
- Sign and Magnitude: Sign Bit + Magnitude Bits

For Negative Numbers:
- One's Complement: Complement all bits ($1 \rightarrow 0,\ 0 \rightarrow 1$) of the corresponding positive number
- Two's Complement: Add 1 to the LSB of the one's complement

**Example**

| Decimal Value | Sign and Magnitude | One's Complement | Two's Complement |
|---|---|---|---|
| +3.625 | 00011.101 | 00011.101 | 00011.101 |
| –3.625 | 10011.101 | 11100.010 | 11100.011 |



6

## Binary Integer Number Representation

| Decimal Value | Sign and Magnitude | One's Complement | Two's Complement |
|---|---|---|---|
| +3 | 011 | 011 | 011 |
| +2 | 010 | 010 | 010 |
| +1 | 001 | 001 | 001 |
| +0 | 000 | 000 | 000 |
| –0 | 100 | 111 | — |
| –1 | 101 | 110 | 111 |
| –2 | 110 | 101 | 110 |
| –3 | 111 | 100 | 101 |
| –4 | — | — | 100 |



7

## Binary Integer Number Representations

- DSPUM Toolbox Function

```
function y = OnesComplement(x,b)
%
% y = OnesComplement(x,b)
% Decimal equivalent of
% Sign-Magnitude format integer to b-bit One's-Complement conversion
%
%    x: integer between -2^(b-1) < x <  2^(b-1) (sign-magnitude)
%    y: integer between        0 <= y <= 2^b-1   (2's-complement)

if any((x <= -2^(b-1) | (x >= 2^(b-1))))
    error('Numbers must satisfy -2^(b-1) < x <  2^(b-1)')
end

s = sign(x);  % sign of x (-1 if x<0, 0 if x=0, 1 if x>0)
sb = (s < 0); % sign-bit  (0 if x>=0, 1 if x<0));

y = (1-sb).*x + sb.*bitcmp(abs(x),b);
```



8

## Binary Integer Number Representations

- DSPUM Toolbox Function

```
function y = TwosComplement(x,b)
%
% y = TwosComplement(x,b)
% Decimal equivalent of
% Sign-Magnitude format integer to b-bit Two's-Complement
conversion
%
%    x: integer between -2^(b-1) <= x <  2^(b-1) (sign-magnitude)
%    y: integer between        0 <= y <= 2^b-1   (2's-complement)

if any((x < -2^(b-1) | (x >= 2^(b-1))))
    error('Numbers must satisfy -2^(b-1) <= x <  2^(b-1)')
end

s = sign(x);  % sign of x (-1 if x<0, 0 if x=0, 1 if x>0)
sb = (s < 0); % sign-bit  (0 if x>=0, 1 if x<0));

y = (1-sb).*x + sb.*(2^b+x);
```



9

## Two's Complement Representation

A binary fractional number with one sign bit and $B$ fractional bits is given by

$$\hat{x}_B = b_0 \blacktriangle b_1 b_2 \ldots b_B = -b_0 + \sum_{i=1}^{B} b_i 2^{-i} \quad \Rightarrow \quad -1 \le \hat{x}_B < 1 - 2^{-B}$$

- Useful Features
  - It has a single representation for the number 0.
  - The same hardware can be used for addition and subtraction.
  - If the final result of a long series of additions is within range, we get the correct answer even if some intermediate results are outside the range.

| Decimal | | Two's Complement |
|---|---|---|
| 1 | | 001 |
| +2 | | +010 |
| 3 | = | 011 |
| +3 | | +011 |
| 6 | ≠ | 110 |
| −2 | | 110 |
| 4 | ≠ | (1)100 |
| −2 | | 110 |
| 2 | = | (1)010 |

EECE-5666: DSP Finite Word=Length Effects 10/73

10

## Binary Fractional Representation

| *Binary* | *Sign-Magnitude* | *one's* | *two's* |
|---|---|---|---|
| 0▲111 | 7/8 | 7/8 | 7/8 |
| 0▲110 | 6/8 | 6/8 | 6/8 |
| 0▲101 | 5/8 | 5/8 | 5/8 |
| 0▲100 | 4/8 | 4/8 | 4/8 |
| 0▲011 | 3/8 | 3/8 | 3/8 |
| 0▲010 | 2/8 | 2/8 | 2/8 |
| 0▲001 | 1/8 | 1/8 | 1/8 |
| 0▲000 | 0 | 0 | 0 |
| 1▲000 | −0 | −7/8 | −1 |
| 1▲001 | −1/8 | −6/8 | −7/8 |
| 1▲010 | −2/8 | −5/8 | −6/8 |
| 1▲011 | −3/8 | −4/8 | −5/8 |
| 1▲100 | −4/8 | −3/8 | −4/8 |
| 1▲101 | −5/8 | −2/8 | −3/8 |
| 1▲110 | −6/8 | −1/8 | −2/8 |
| 1▲111 | −7/8 | −0 | −1/8 |

EECE-5666: DSP Finite Word=Length Effects 11/73

11

## Binary Fractional Representations

**DSPUM Toolbox Functions**

- The Matlab functions given on slides 10 and 11 can easily be used for fractional numbers with proper modifications.
- The function `sm2oc` converts a decimal sign-magnitude fraction into its one's-complement format.
- The function `oc2sm` performs the inverse operation.
- The function `sm2tc` converts a decimal sign-magnitude fraction into its two's-complement format.
- The function `tc2sm` performs the inverse operation.



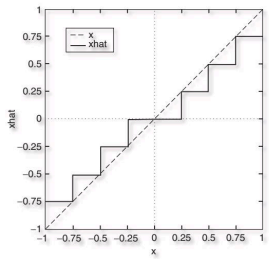
12

## One's Complement Representation for Fractions

**Example:** Let $-1 < x < 1$ which is a fractional $\pm$ number. Let $B = 2$ with resolution $\Delta = 2^{-B} = 0.25$. Using the `sm2oc` function compute and plot $x$ and $\hat{x}$ values.

**Solution:** MATLAB Script:

```
% Sign-Magnitude numbers between -1 and 1
   x = [-1+2^(-10):2^(-10):1-2^(-10)];
% Select bits for Truncation
   B = 2;
% Sign-Mag to One's Complement
   y = sm2oc(x,B);
% Truncation
   yhat = fix(y*2^B)/2^B;
% Ones'-Complement to Sign-Mag
   xhat = oc2sm(yhat,B);
% Plot
   plot(x,x,'g',x,xhat,'r','linewidth',1);
```





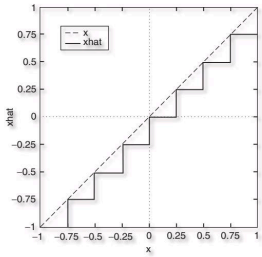

13

## Two's Complement Representation for Fractions

**Example:** Let $-1 < x < 1$ which is a fractional $\pm$ number. Let $B = 2$ with resolution $\Delta = 2^{-B} = 0.25$. Using the `sm2tc` function compute and plot $x$ and $\hat{x}$ values.

**Solution:** MATLAB Script:

```
% Sign-Magnitude numbers between -1 and 1
   x = [-1+2^(-10):2^(-10):1-2^(-10)];
% Select bits for Truncation
   B = 2;
% Sign-Mag to Two's Complement
   y = sm2tc(x,B);
% Truncation
   yhat = fix(y*2^B)/2^B;
% Two'-Complement to Sign-Mag
   xhat = tc2sm(yhat,B);
% Plot
   plot(x,x,'g',x,xhat,'r','linewidth',1);
```




EECE-5666: DSP Finite Word=Length Effects 14/73

14

## Comments

- Any integer or fraction can be converted to a fraction between zero and one by dividing by the highest power of 2.
- The binary point does not exist physically in the computer. Simply, the logic circuits in the computer are designed so that the computations result in numbers that correspond to the assumed binary point location.
- An unambiguous representation of a number is possible if a fixed-word length and a fixed binary-point location have been agreed upon.
- A disadvantage of signed-magnitude representation is that it is difficult to add positive and negative numbers. When a minus sign occurs either the adder should be switched over to subtraction mode or use both an adder and a subtracter.

EECE-5666: DSP Finite Word=Length Effects 15/73

15

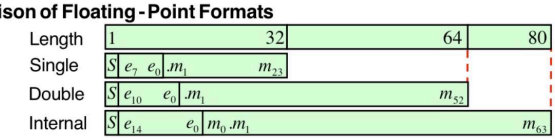
## Floating Point Binary Numbers

**Decimal:** $(X)_{10} = M \times 10^E$ $M$ = Mantissa **Binary:** $(X)_2 = M \times 2^E$ $E$ = Exponent

**IEEE Standard P754**

| IEEE Format | Word Length | Sign $S$ | Exponent Length $E$ | Range | Mantissa Length $M$ | Precision |
|---|---|---|---|---|---|---|
| Single | 32 bit | 1 bit | 8 bit | $2^{\pm 127} \approx 10^{\pm 38}$ | 23 bit $\triangleq$ 7 digits$_{dec}$ | |
| Double | 64 bit | 1 bit | 11 bit | $2^{\pm 1023} \approx 10^{\pm 308}$ | 52 bit $\triangleq$ 16 digits$_{dec}$ | |
| Internal | 80 bit | 1 bit | 15 bit | $2^{\pm 16383} \approx 10^{\pm 4932}$ | 64 bit $\triangleq$ 19 digits$_{dec}$ | |

**Comparison of Floating-Point Formats**




**Usually, the mantissa is normalized to $m_0 = 1$, i.e.,**

$$M = 1 + m_1 2^{-1} + \cdots + m_k 2^{-k} = 1 + \sum_{i=1}^{k} m_i 2^{-i} \quad \Rightarrow 1 \le M < 2$$

$$X = (-1)^s M \; 2^{E\text{-offset}}$$

$$\text{offset} = \begin{cases} 2^7 - 1 = 127 & \text{for single precision} \\ 2^{10} - 1 = 1023 & \text{for double precision} \\ 2^{14} - 1 = 16383 & \text{for internal precision} \end{cases}$$

Mantissa is called the significand in P754 standard.

EECE-5666: DSP Finite Word=Length Effects 16/73

16

## IEEE P754 Floating-Point Standard

Features of the IEEE P754 standard (singal precision) are:

- If the sign bit is 0, the number is positive; if the sign bit is 1, the number is negative.
- The exponent is coded in 8-bit excess-127 (and not 128) format. Hence the uncoded exponents are between $-127$ and 128.
- The mantissa is in 23-bit binary. A normalized mantissa always starts with a bit 1, followed by the binary point, followed by the rest of the 23-bit mantissa. However, the leading bit 1, which is always present in a normalized mantissa, is hidden (not stored) and needs to be restored for computation. All IEEE 754 normalized numbers have a significand that is in the interval $1 \le M < 2$.

EECE-5666: DSP Finite Word=Length Effects 17/73

17

## IEEE P754 Floating-Point Standard

Features of the IEEE P754 standard (singal precision) are: (continued)

- The smallest normalized number is $2^{-126}$, and the greatest normalized number is almost $2^{128}$. The resulting positive decimal range is roughly $10^{-38}$ to $10^{38}$ with a similar negative range.
- If $E = 0$ and $M = 0$, then the representation is interpreted as a *de-normalized* number (i.e., the hidden bit is 0) and is assigned a value of $\pm 0$, depending on the sign bit (called the *soft zero*). Thus $0$ has two representations.
- If $E = 255$ and $M \neq 0$, then the representation is interpreted as a *not-a-number* (abbreviated as NaN). MATLAB assigns a variable `NaN` when this happens—e.g., $0/0$.
- If $E = 255$ and $M = 0$, then the representation is interpreted as $\pm\infty$. MATLAB assigns a variable `inf` when this happens—e.g., $1/0$.



18

## Quantization and Overflow

**Addition**

| Binary | Decimal |
|---|---|
| 0.1101 | 0.8125 |
| +0.1001 | +0.5625 |
| 01.0110 | +1.3750 |

**Multiplication**

| Binary | Decimal |
|---|---|
| 0.1101 | 0.8125 |
| ×0.1001 | ×0.5625 |
| 0.01110101 | +0.45703125 |

**Addition:**
Two $(B+1)$-bit words $\Rightarrow (B+2)$-bits
Increases the # bits before the point

Problem: Overflow

**Multiplication:**
Two $(B+1)$-bit words $\Rightarrow (2B+1)$-bits
Increases the # bits after the point

Problem: Quantization

**In General**

Quantization: A quantity $x$ is converted into a quantity $x_q$ that is close to $x$, but can assume fewer different values than $x$

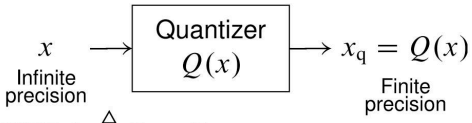
- Quantization of a infinite-precision variable
- Reduction of the number of bits of a fractional number by rounding or truncation, e.g., round $3.95$ to $4$ or $101.01101$ to $101.01$



19

## Process of Quantization

- There are two operations by which quantization, that is, an assignment of the given infinite-precision number to one of the finite representable number, is performed.
  - Truncation operation
  - Rounding operation
- The quantization operation is represented by a Quantizer:




- Quantization error: $e_q \triangleq x_q - x$
- The quantizer is characterized by (for fractional numbers)

  $B$: The number of fractional bits (plus a sign bit)

  $\Delta$: Resolution or the difference between consecutive levels.

$$\Delta = 0.\underbrace{00\ldots1}_{B\text{-bits}} = 2^{-B}$$



20

## Process of Quantization

- **Truncation operation:** The given number $x$ is truncated beyond $B$ significant bits, that is, the rest of the bits are eliminated to obtain $x_q$.
- The MATLAB fragment is `xq = fix(x*2^B)/2^B;`
- Quantization error: $e_T \triangleq x_q - x$
  - Sign-magnitude form:

$$\begin{aligned} -2^{-B} \le e_T \le 0 \quad & \text{for } x \ge 0 \\ 0 \le e_T \le 2^{-B} \quad & \text{for } x < 0 \end{aligned}$$

  - One's-complement format:

$$\begin{aligned} -2^{-B} \le e_T \le 0 \quad & \text{for } x \ge 0 \\ 0 \le e_T \le 2^{-B} \quad & \text{for } x < 0 \end{aligned}$$

  - Two's-complement format:

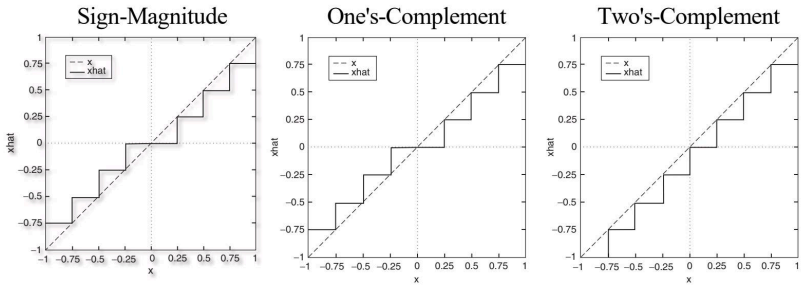
$$\begin{aligned} -2^{-B} \le e_T \le 0 \quad & \text{for } x \ge 0 \\ -2^{-B} \le e_T \le 0 \quad & \text{for } x < 0 \end{aligned}$$



21

## Process of Quantization

- **Truncation operation:**



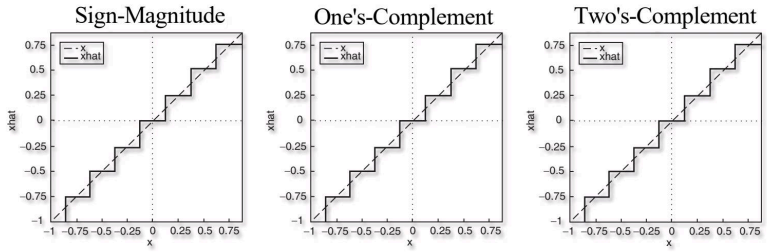

EECE-5666: DSP Finite Word=Length Effects 22/73

22

## Process of Quantization

- **Rounding operation:** The given number $x$ is assigned to the nearest representable level $x_q$. The Matlab fragment is `xq = round(x*2^B)/2^B;`
- Then the quantization error $e_R \triangleq e_q - x$ us same across all three formats:

$$-\tfrac{1}{2}2^{-B} \le e_R \le \tfrac{1}{2}2^{-B}; \quad |e_R|_{max} = \frac{\Delta}{2} = \frac{2^{-B}}{2} = 2^{-(B+1)}$$




EECE-5666: DSP Finite Word=Length Effects 23/73

23

## Quantization in Two's Complement

If $b_0 \blacktriangle b_1 b_2 \ldots$ represents a two's complement number, its value is

$$x = -b_0 + \sum_{i=1}^{\infty} b_i 2^{-i}$$

Using $(B+1)$-bits $\Rightarrow$

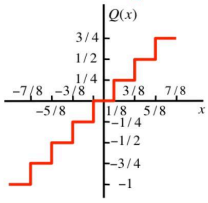
$$x_q = Q_B[x] = -b_0 + \sum_{i=1}^{B} b_2 2^{-i} \quad \text{i.e., } b_0 \blacktriangle b_1 b_2 \cdots b_B$$

$Q_B[\,]$ is nonlinear

Rounding
$0101\blacktriangle010 \rightarrow 0101\blacktriangle01$
$0101\blacktriangle011 \rightarrow 0101\blacktriangle10$

Truncation
$0101\blacktriangle010 \rightarrow 0101\blacktriangle01$
$0101\blacktriangle011 \rightarrow 0101\blacktriangle01$

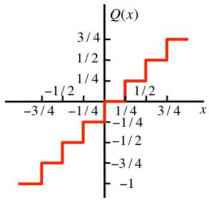
Rounding




Quantization Error $e = Q_B[x] - x$

| | Range | Resolution |
|---|---|---|
| Rounding | $-\frac{\Delta}{2} < e \le \frac{\Delta}{2}$ | $\Delta = 2^{-B}$ |
| Truncation | $-\Delta < e \le 0$ | $\Delta = 2^{-B}$ |

Truncation



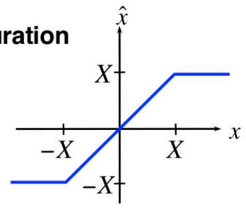

EECE-5666: DSP Finite Word=Length Effects 24/73

24

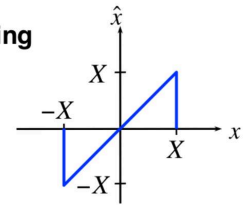
## Overflow Characteristics

**Overflow**: What happens when a variable $x$ takes a value outside the allowable range $(-X, X)$

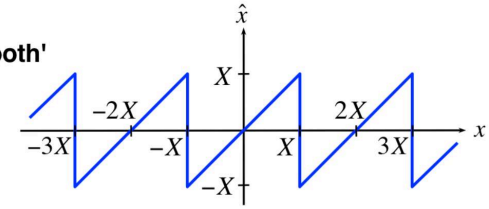
**Saturation**




**Zeroing**



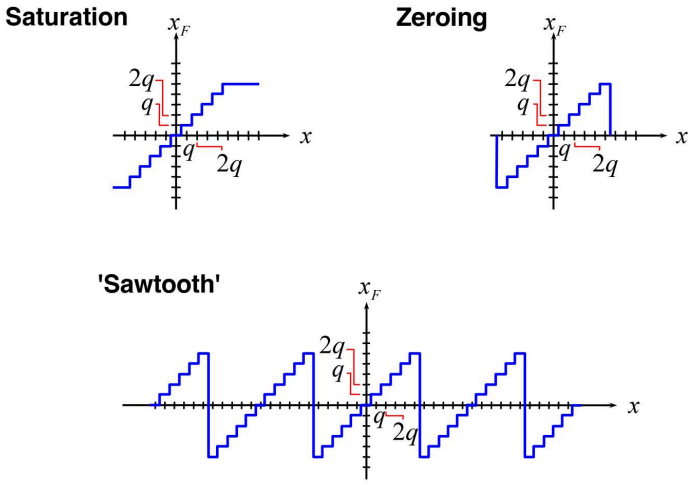

**'Sawtooth'**




EECE-5666: DSP Finite Word=Length Effects 25/73

25

## Quantized Overflow Characteristics






26

## MATLAB Functions for Representations

- MATLAB provides functions for conversion between binary and decimal numbers as well as for quantization.
- Book toolbox also provides few additional functions.
- MATLAB Functions:
  - `dec2bin`: Converts positive decimal integers into binary bit representations
  - `bin2dec`: Converts binary bit arrangement into positive decimal integer.
  - `round`: Quantization to nearest integer using rounding
  - `fix`: Quantization to the integer part using truncation
- Book Toolbox Functions:
  - `dec2beqR`: Decimal to binary equivalent using rounding
  - `dec2beqT`: Decimal to binary equivalent using truncation



27

## MATLAB Functions for Representations

```
function [beq,E,B] = d2beqR(d,L)
% Binary equivalent decimal (beq) representation of a decinal (d)
% using L=1+E+B bit Representation using Rounding operation:
%  [beq,E,B] = d2beqR(d,L)
%   d can be scalar or vector or matrix.
%   L must be greater than the number of integer bits required
dm = abs(d);
 E = max(max(0,fix(log2(dm(:)+eps)+1))); % Integer bits
if (E >= L)
     errmsg = [' *** L must be greater than ',num2str(E),' ***'];
     error(errmsg);
end
B = L-1-E; % Fractional bits
beq = round(dm./(2^E).*(2^(L-1))); % Rounding to L bits
beq = sign(d).*beq*(2^(-B)); % 1+E+B bit representation
```

Example:

```
>> x=[-1.8184,2.426];
>> [xq,E,B] = dec2beqR(x,6)
xq =
   -1.8750    2.3750
```

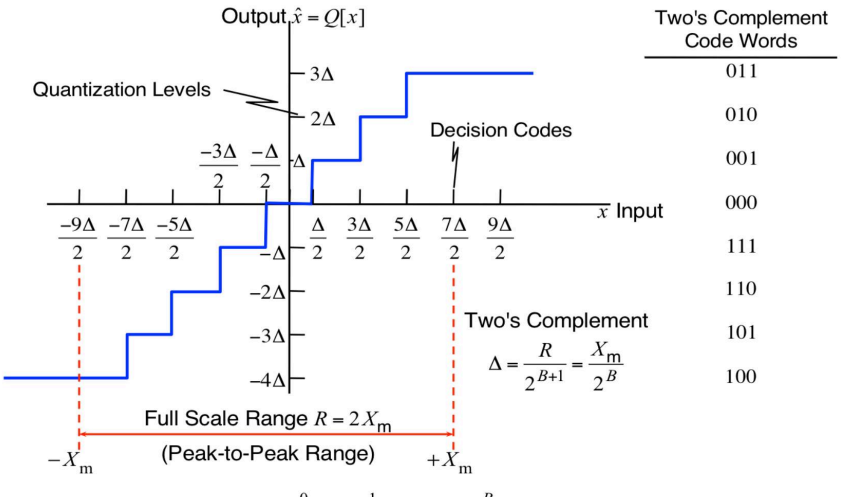
```
E =
     2
B =
     3
```

EECE-5666: DSP Finite Word=Length Effects 28/73

28

## Statistical Analysis of Quantization Error

### Two's Complement




$$b_0 b_1 \cdots b_B = -b_0 2^0 + b_1 2^{-1} + \cdots + b_B 2^{-B}, \quad b_k = 1 \text{ or } 0$$

EECE-5666: DSP Finite Word=Length Effects 29/73

29

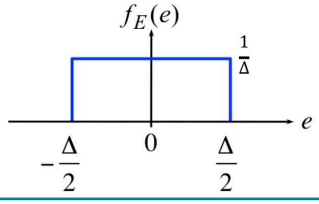
## Statistical Analysis of Quantization Error

**Actual System:** $x[n]$ → Quantizer $Q(x[n])$ → $x_q[n]$ — Nonlinear System

**Additive Model:** $x[n]$ → (+) → $x_q[n] = x[n] + e[n]$, with $e[n]$ into (+) — Linear System

**Quantization error (noise):** $e[n] = x_q[n] - x[n]$

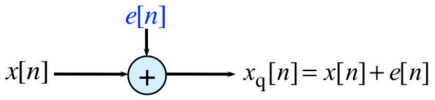
Probability density function for the quantization roundoff error in A/D conversion






30

## Statistical Model for Quantization Noise




**Assumptions:**

1. The noise sequence $e[n]$ is a stationary random sequence.
2. The noise sequence $e[n]$ is uncorrelated with the input sequence $x[n]$.
3. The noise sequence $e[n]$ is an independent process, that is, samples are independent of each other (white noise process).
4. The probability density function (pdf) of each sample $e[n]$ is uniformly distributed over the quantizer resolution of width $\Delta = 2^{-B}$, that is

$$f_E(e) = \begin{cases} \frac{1}{\Delta} = 2^B, & \text{over interval } \Delta, \\ 0, & \text{elsewhere.} \end{cases}$$

**For two's complement format:**

Truncation: $-\Delta \le e[n] \le 0$

Rounding: $-\frac{\Delta}{2} \le e[n] \le \frac{\Delta}{2}$

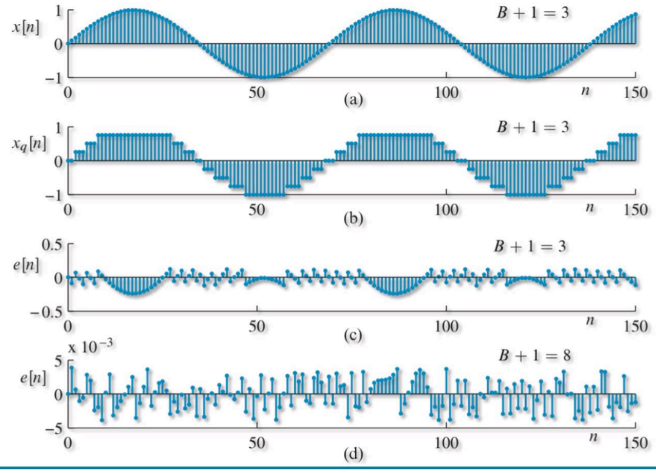


31

## Statistical Model for Quantization Noise

**Example 15.1:**

$$x[n] = 0.99[\sin(n/11) + \sin(n/31) + \cos(n/67)]/3$$



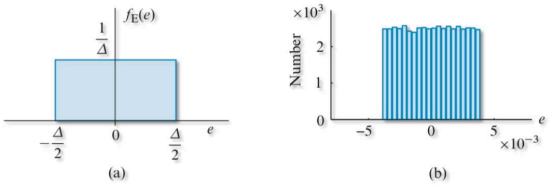



32

## Statistical Model for Quantization Noise

**Example 15.1:**

$$x[n] = 0.99[\sin(n/11) + \sin(n/31) + \cos(n/67)]/3$$




8-bits

Hence mean of $e[n] = 0$

**ADSP Book Toolbox Function:**

```
[ebin,fE] = epdf(e,Nbins); % fE contains pdf values at ebin locations
```

Graph pdf using the `bar` function: `bar(ebin,fE,'linestyle','none');`

**MATLAB Functions:**

```
histogram(e,nbins); % plots histogram (unnormalized)
[fE,ebin] = histcounts(e,nbins,'Normalization','pdf'); % normalized histogram
```



33

## Statistical Analysis of Quantization Error

**Signal-to-Quantization-Noise Ratio:**

$$\text{SQNR} = 10\log_{10}\frac{\text{Signal Power}}{\text{Quantization Noise Power}} = 10\log_{10}\frac{P_x}{P_n}$$

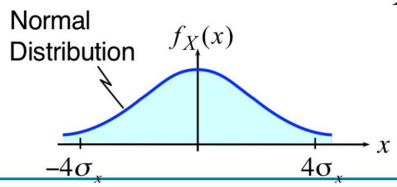
$$P_n = \sigma_e^2 = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e^2 f_E(e)\,de = \frac{1}{\Delta}\int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} e^2\,de = \frac{\Delta^2}{12}, \quad \Delta = \frac{R}{2^{B+1}} = \frac{2X_m}{2^{B+1}} = \frac{X_m}{2^B}$$

$$\Rightarrow \text{SQNR} = 20\log_{10}\frac{\sigma_x}{\sigma_e} = 6.02B + 10.8 - 20\log_{10}\frac{X_m}{\sigma_x}\ \text{dB}$$

$X_m = 4\sigma_x \Rightarrow 0.064\%$ outside range

$\Rightarrow$ SQNR $\simeq 6B - 1.25$ dB

6 dB for each added bit $(b_0 b_1 \cdots b_B)$

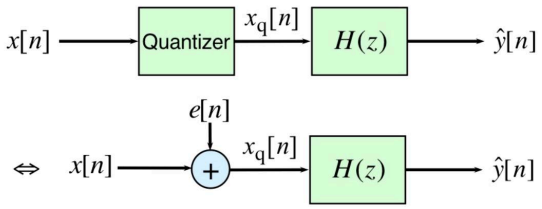





34

## Input A/D Quantization Noise through LTI Systems

Now consider a quantized input applied to a filter




Then

$$\hat{y}[n] = h[n] * x_q[n] = \underbrace{h[n]*x[n]}_{y[n]} + \underbrace{h[n]*e[n]}_{\triangleq g[n]}$$

$y[n]$: Correct Output; $g[n]$: Output Noise

From the theory of stochastic processes (see Chapter 13), we have

Mean of $g[n] = 0$ and variance $\sigma_g^2 = \sigma_e^2 \sum_{n=-\infty}^{\infty} |h[n]|^2 = \frac{\sigma_e^2}{2\pi}\int_{-\pi}^{\pi} |H(\omega)|^2\,d\omega$



35

## Output Noise Due to Input Signal Quantization

$$\hat{y}[n] = h[n] * x_q[n] = \underbrace{h[n] * x[n]}_{y[n]} + \underbrace{h[n] * e[n]}_{\triangleq g[n]}$$

- Output noise sequence:

$$g[n] = h[n] * e[n]$$

- Output noise mean is zero and variance is:

$$\sigma_g^2 = \sigma_e^2 \sum_{-\infty}^{\infty} |h(n)|^2 = \frac{\sigma_e^2}{2\pi} \int_{-\pi}^{\pi} |H(\omega)|^2 \, \mathrm{d}\omega$$

- Variance Gain or normalized output variance:

$$\mathrm{VG} \triangleq \frac{\sigma_g^2}{\sigma_e^2} = \sum_{-\infty}^{\infty} |h(n)|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(\omega)|^2 \, \mathrm{d}\omega$$

where $\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(\omega)|^2 \, \mathrm{d}\omega = \frac{1}{2\pi} \oint_{\mathrm{UC}} H(z) H(z^{-1}) z^{-1} \, \mathrm{d}z$

$= \sum \left[ \text{Residues of } H(z) H(z^{-1}) z^{-1} \text{ inside UC} \right]$

$= \mathcal{Z}^{-1} \left[ H(z) H(z^{-1}) \right]_{n=0}$



36

## Output Noise Due to Input Signal Quantization

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(\omega)|^2 \, \mathrm{d}\omega = \sum_{-\infty}^{\infty} |h(n)|^2 = \mathcal{Z}^{-1} \left[ H(z) H(z^{-1}) \right]_{n=0}$$

- FIR Filter: $h[n] = \{b_0, b_1, \ldots, b_{M-1}\}$ $\Rightarrow \frac{\sigma_g^2}{\sigma_e^2} = \sum_{n=0}^{M-1} |b_n|^2$

- IIR Filter:

$$H(z) = \frac{\sum_{\ell=0}^{N-1} b_\ell z^{-\ell}}{1 + \sum_{k=1}^{N-1} a_k z^{-k}} = R_0 + \sum_{k=1}^{N-1} \frac{R_k}{z - p_k} \quad \Rightarrow \frac{\sigma_g^2}{\sigma_e^2} = R_0^2 + \sum_{k=1}^{N-1} \sum_{\ell=1}^{N-1} \frac{R_k R_\ell^*}{1 - p_k p_\ell^*}$$

Exact

or $\frac{\sigma_g^2}{\sigma_e^2} \simeq \sum_{k=0}^{K-1} |h(n)|^2, \quad K \gg 1$ Approximate



37

## Output Noise Due to Input Signal Quantization

**Example 15.4**: A/D quantization noise through IIR filter

Consider a first-order IIR filter

$$H(z)=\frac{1}{1-0.9z^{-1}} \quad \text{or} \quad h[n]=(0.9)^n u[n]$$

Then the variance-gain is given by

$$\mathrm{VG}=\frac{1}{1-0.9^2}=5.26$$

If the pole is moved to 0.99 then $\mathrm{VG}=50.25.$

$\Rightarrow$ Proper scaling of filter coefficients is necessary to make

$$VG \simeq 1$$

to avoid excessive output noise.

**DSPUM Toolbox Function:**

```
VG = VarGain(b,a)
```



38

## Quantization of Filter Coefficients

- Quantization of filter coefficients is a one time process that changes the filter coefficients leading to a filter with a different frequency response.
- The new filter is still LTI (maybe unstable); simply it is different!
- If it does not satisfy the design requirements, we obtain another that does!

$$H(z)=\frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} \qquad \begin{matrix}\hat{b}_k = Q_B\left[b_k\right] \\ \hat{a}_k = Q_B\left[a_k\right]\end{matrix} \qquad \hat{H}(z)=\frac{\sum_{k=0}^{M} \hat{b}_k z^{-k}}{1+\sum_{k=1}^{N} \hat{a}_k z^{-k}}$$

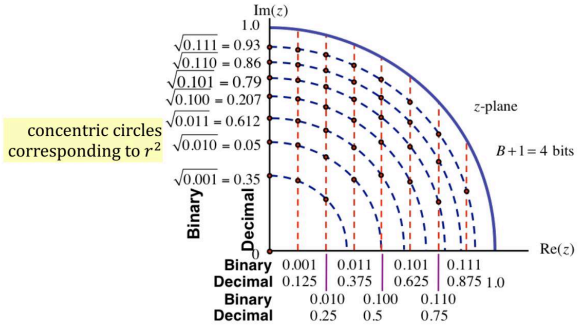


39

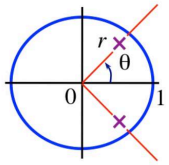
## Different Structures: Different Sensitivities

Consider the filter $H(z) = \dfrac{b}{1 + a_1 z^{-1} + a_2 z^{-2}}$ with $a_1 = -2r\cos\theta$, $a_2 = r^2$

Quantizing $a_1$, $a_2$ $\Rightarrow$ Only certain pole locations are allowed!




concentric circles corresponding to $r^2$



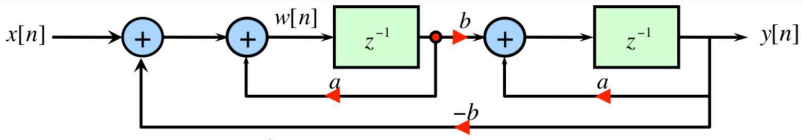

- Non-uniform pole distribution
- Problems for LP and HP filters with poles close to the real axis, i.e., frequencies $\omega = 0$ or $\omega = \pi$

vertical lines corresponding to $2r\cos\theta$

EECE-5666: DSP Finite Word=Length Effects 40/73

40

## Coupled Form Second-Order Section

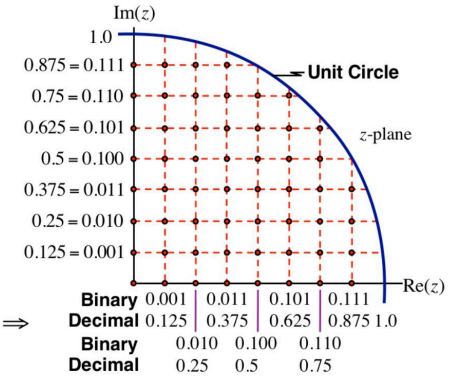



$$\Rightarrow H(z) = \frac{bz^{-2}}{1 - 2az^{-1} + \left(a^2 + b^2\right)z^{-2}} = \frac{b}{(z - a + \mathrm{j}b)(z - a - \mathrm{j}b)}$$

$p_1 = a + \mathrm{j}b = re^{\mathrm{j}\theta}$ $\quad a = r\cos\theta$

$p_2 = a - \mathrm{j}b = re^{-\mathrm{j}\theta}$ $\quad b = r\sin\theta$

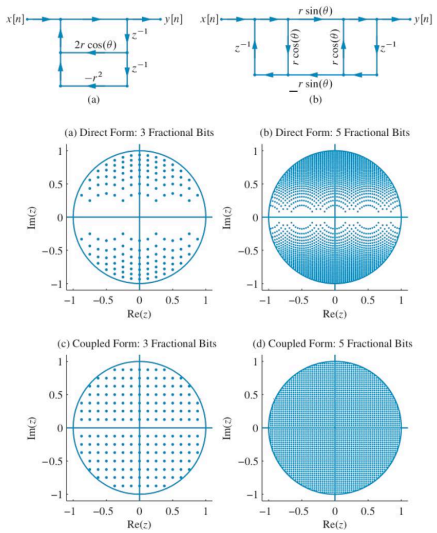
Quantization of $a$, $b$ to $(B+1) = 4$ bits $\Rightarrow$




EECE-5666: DSP Finite Word=Length Effects 41/73

41

## Different Structures: Different Sensitivities






42

## Quantization of Filter Coefficients

Effect on Pole Locations:

- Since the filter coefficients are changed due to quantization, there will be changes to the pole locations.
- These can be analyzed with a reasonable complexity.
- Consider the denominator polynomial

$$D(z) \triangleq 1 + \sum_{k=1}^{N} a_k z^{-k} = \prod_{\ell=1}^{N} \left(1 - p_\ell z^{-1}\right) = D(z; p_1, \ldots, p_N)$$

where

$$p_\ell = f(a_1, \ldots, a_N); \quad \ell = 1, \ldots N$$

Then the change in the denominator $D(z)$ due to change in $a_k$ is

$$\left(\frac{\partial D(z)}{\partial a_k}\right) = \left(\frac{\partial D(z)}{\partial p_1}\right)\left(\frac{\partial p_1}{\partial a_k}\right) + \left(\frac{\partial D(z)}{\partial p_2}\right)\left(\frac{\partial p_2}{\partial a_k}\right) + \cdots + \left(\frac{\partial D(z)}{\partial p_N}\right)\left(\frac{\partial p_N}{\partial a_k}\right)$$



43

## Quantization of Filter Coefficients

**Effect on Pole Locations**:

$$\left(\frac{\partial D(z)}{\partial a_k}\right)=\left(\frac{\partial D(z)}{\partial p_1}\right)\left(\frac{\partial p_1}{\partial a_k}\right)+\left(\frac{\partial D(z)}{\partial p_2}\right)\left(\frac{\partial p_2}{\partial a_k}\right)+\cdots+\left(\frac{\partial D(z)}{\partial p_N}\right)\left(\frac{\partial p_N}{\partial a_k}\right)$$

where

$$\left(\frac{\partial D(z)}{\partial p_i}\right)=\frac{\partial}{\partial p_i}\left[\prod_{\ell=1}^{N}\left(1-p_\ell z^{-1}\right)\right]=-z^{-1}\prod_{\ell\neq i}\left(1-p_\ell z^{-1}\right)$$

Now

$$\left(\frac{\partial D(z)}{\partial p_i}\right)\Big|_{z=p_\ell}=0,\quad \text{for } \ell\neq i$$

Hence

$$\left(\frac{\partial D(z)}{\partial a_k}\right)\Big|_{z=p_\ell}=\left(\frac{\partial D(z)}{\partial p_\ell}\right)\Big|_{z=p_\ell}\left(\frac{\partial p_\ell}{\partial a_k}\right)\quad\text{or}\quad\left(\frac{\partial p_\ell}{\partial a_k}\right)=\frac{\left(\frac{\partial D(z)}{\partial a_k}\right)\Big|_{z=p_\ell}}{\left(\frac{\partial D(z)}{\partial p_\ell}\right)\Big|_{z=p_\ell}}$$



44

## Quantization of Filter Coefficients

**Effect on Pole Locations**:

$$\left(\frac{\partial D(z)}{\partial a_k}\right)\Big|_{z=p_\ell}=\left(\frac{\partial D(z)}{\partial p_\ell}\right)\Big|_{z=p_\ell}\left(\frac{\partial p_\ell}{\partial a_k}\right)\quad\text{or}\quad\left(\frac{\partial p_\ell}{\partial a_k}\right)=\frac{\left(\frac{\partial D(z)}{\partial a_k}\right)\Big|_{z=p_\ell}}{\left(\frac{\partial D(z)}{\partial p_\ell}\right)\Big|_{z=p_\ell}}$$

Now

$$\left(\frac{\partial D(z)}{\partial a_k}\right)\Big|_{z=p_\ell}=\frac{\partial}{\partial a_k}\left(1+\sum_{i=1}^{N}a_i z^{-i}\right)\Bigg|_{z=p_\ell}=z^{-k}\Big|_{z=p_\ell}=p_\ell^{-k}$$

Hence

$$\left(\frac{\partial p_\ell}{\partial a_k}\right)=\frac{p_\ell^{-k}}{-z^{-1}\prod_{i\neq\ell}\left(1-p_i z^{-1}\right)\Big|_{z=p_\ell}}=-\frac{p_\ell^{N-k}}{\prod_{i\neq\ell}\left(p_\ell-p_i\right)}=\left(\frac{\partial p_\ell}{\partial a_k}\right)$$

or

$$\Delta p_\ell=\sum_{k=1}^{N}-\frac{p_\ell^{N-k}}{\prod_{i\neq\ell}\left(p_\ell-p_i\right)}\Delta a_k$$

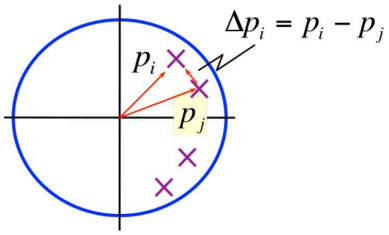


45

## Quantization of Filter Coefficients

**Effect on Pole Locations**:

$$\Delta p_\ell = \sum_{k=1}^{N} -\frac{p_\ell^{N-k}}{\prod_{i\neq\ell}(p_\ell - p_i)}\Delta a_k$$




- Filters with tightly clustered poles (e.g., narrowband) are very sensitive
- To minimize sensitivity, maximize $|p_i - p_j|$, by using second-order sections with complex-conjugate poles that are usually far apart
- Similar sensitivity formula also applies to zero movements.





## Example: Elliptic Filter Coefficient Quantization

Consider a bandpass filter with the specifications:

$$\omega_{s_1} = 0.2\pi, \omega_{p_1} = 0.3\pi, \omega_{p_2} = 0.4, \omega_{s_2} = 0.5\pi, A_p = 0.1 \text{ dB}, A_s = 60 \text{ dB}$$

MATLAB script: (infinite precision)

```
>> omegap = [0.3,0.4]; omegas = [0.2,0.5]; Ap = 0.1; As = 60;
>> [N,omegac] = ellipord(omegap,omegas,Ap,As);
>> [b,a] = ellip(N,Ap,As,omegac);
```

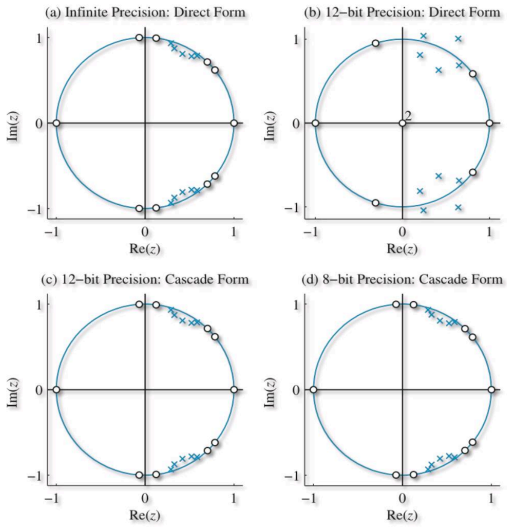
Filter coefficient quantization using 12 bits

```
>> L1 = 12; [bahat,E1,B1] = dec2beqR([b;a],L1);
bhat1 = bahat(1,:); ahat1 = bahat(2,:); B1
B1 =
     6        ⇒  6 fractional bits
```

Direct to cascade → Filter coefficient quantization using 12 bits

```
>> [sos,G] = tf2sos(b,a);
>> L2 = 12; [soshat,E2,B2] = d2beqR(sos,L2);
>> [bhat2,ahat2] = sos2tf(soshat,G); B2
B2 =
    10        ⇒  10 fractional bits
```

## Example: Effects of Coefficient Quantization on Poles






48

## Effect on Frequency Response

- The frequency response of an IIR filter with unquantized coefficients is

$$H(\omega) = \frac{\sum_{k=0}^{M} b_k e^{-j\omega k}}{1 + \sum_{k=1}^{N} a_k e^{-j\omega k}}$$

- When coefficients $\{a_k\}$ and $\{b_k\}$ are quantized to $\{\hat{a}_k\}$ and $\{\hat{b}_k\}$, respectively, the new frequency response is

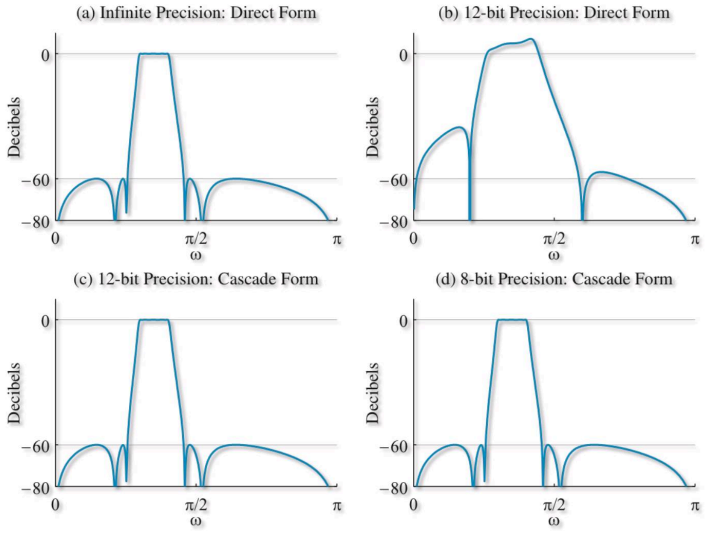
$$\hat{H}(\omega) = \frac{\sum_{k=0}^{M} \hat{b}_k e^{-j\omega k}}{1 + \sum_{k=1}^{N} \hat{a}_k e^{-j\omega k}}$$

- It is difficult to obtain a closed form expression for the new frequency response in terms of the old set of coefficients or otherwise analyze it analytically.
- MATLAB is the best vehicle to study the effect on frequency response due to coefficient quantization.



49

## Example: Effects of Coefficient Quantization on $H(\omega)$






50

## Quantization of FIR Filter Coefficients

$$y[n]=\sum_{k=0}^{M} h[k]x[n-k], \qquad H(z)=\sum_{k=0}^{M} h[k]z^{-k}$$

$$h[k] \xrightarrow{Q_{B+1}\{h[k]\}} \hat{h}[k]=h[k]+\Delta h[k] \Rightarrow z_i \to \hat{z}_i = z_i + \Delta z_i$$

$$\hat{H}(z)=\sum_{k=0}^{M} \hat{h}[k]z^{-k} = H(z)+\sum_{k=0}^{M} \Delta h[k]z^{-k}$$

$$\Delta z_i = \sum_{k=1}^{M} \frac{\partial z_i}{\partial h[k]}\Delta h[k], \quad \frac{\partial z_i}{\partial h[k]} = \frac{z_i^{N-k}}{\prod_{j=1, j\neq i}^{N}\left(z_i - z_j\right)}$$

$\Rightarrow$ Problem, if zeros are tightly clustered



51

## Quantization of Linear-Phase FIR Filter Coefficients

- For FIR filters with linear phase, the zeros are more or less well spread out in the $z$-plane, which poses no problem with coefficient quantization
- Quantization preserves linear phase
- Consider the new frequency response $\widehat{H}(\omega)$, which can be expressed as

$$\widehat{H}(\omega) = H(\omega) + \Delta H(\omega)$$

where $\Delta H(\omega)$ is due to $-2^{-(B+1)} \leq \Delta h(k) \leq 2^{-(B+1)}$

- Then the bound on the change is given by

$$|\Delta H(\omega)| = \left|\sum_{k=0}^{M} \Delta h(k) e^{-j\omega k}\right| \leq \sum_{k=0}^{M} |\Delta h(k)| \leq (M+1)2^{-(B+1)}$$

which is a pessimistic bound



52

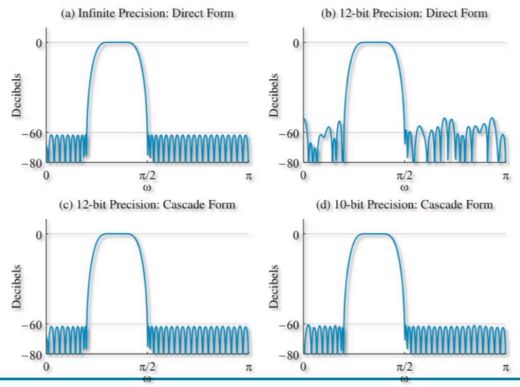
## Example: FIR Equiripple Filter Coefficient Quantization

Consider a bandpass filter with the specifications:

$$\omega_{s_1} = 0.2\pi, \omega_{p_1} = 0.3\pi, \omega_{p_2} = 0.4, \omega_{s_2} = 0.5\pi, A_p = 0.1 \text{ dB}, A_s = 60 \text{ dB}$$

We design an equiripple FIR filter using

```
>> b = firpm(69,[0,0.2,0.3,0.4,0.5,1],[0,0,1,1,0,0]);
```






53

## Round-off Effects in Intermediate Arithmetic Operations

- Issues in intermediate calculations:
  - Quantization after multiplications
  - Overflow after additions
- These effects are more serious and difficult to study due to feedback paths in IIR filters and possible overflow in both FIR and IIR filters
- **Uncorrelated errors:** Result of using linear statistical models
  - Multiplication model
  - FIR filter analysis
  - IIR Filter analysis
- **Correlated errors:** Caused due to nonlinearities in quantization
  - Limit cycle oscillations: A zero-input limit cycle is a nonzero periodic output sequence produced by a nonlinear quantizer in the feedback loop of a digital filter.

  **Granular:** Due to nonlinearities in multiplication
  **Overflow:** Due to overflow in additions

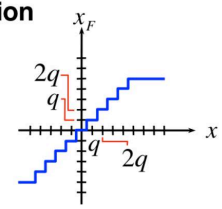


54

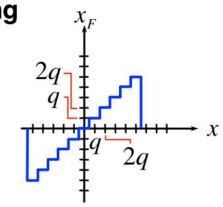
## Quantizer Overflow Characteristics

**Overflow**: What happens when a variable $x$ takes a value outside the allowable range $(-X, X)$

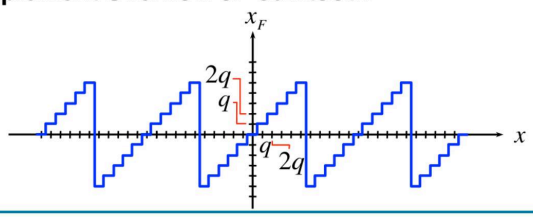
**Saturation**




**Zeroing**




**2's - complement Overflow or 'Sawtooth'**






55

## Quantizer Overflow Characteristics

**Book Toolbox Function:**

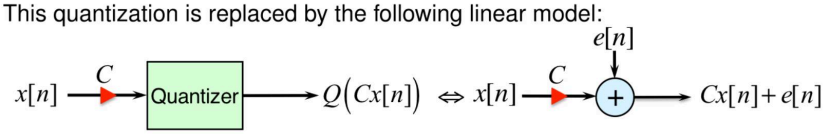
```
function [y] = QFix(x,B,Qmode,Omode)
% Fixed-point Arithmetic using (B+1)-bit Representation
% ------------------------------------------------------
%   [y] = QFix(x,B,Qmode,Omode)
%      y: Decimal equivalent of quantized x with values in [-1,1)
%      x: a real number array
%      B: Number of fractional bits
%  Qmode: Quantizer mode
%         'round': two's-complement rounding characteristics
%         'trunc': Two's complement truncation characteristics
%  Omode: Overflow mode
%         'satur': Saturation limiter
%         'twosc': Two's-complement overflow
```



56

## Multiplication Quantization Errors

- When two $(B+1)$ bit numbers are multiplied, the result is a $(2B+1)$ which must be quantized back to $(B+1)$ bits.
- This quantization is replaced by the following linear model:



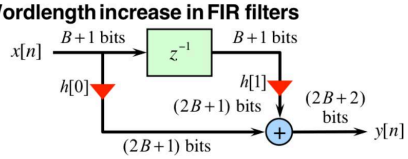

- Assumptions:
  1. The random signal $e(n)$ is uncorrelated with the sequence $x(n)$ for rounding operation (or two's-complement truncation operation) in the quantizer.
  2. The signal $e(n)$ is an independent process (i.e., the samples are independent of each other).
  3. The pdf $f_E(e)$ of $e(n)$ for each $n$ is uniformly distributed over the interval of width $\Delta = 2^{-B}$, which is the quantizer resolution.



57

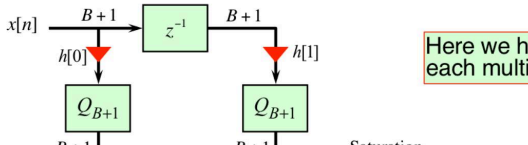
## Multiplication Quantization in FIR Filters

**Wordlength increase in FIR filters**

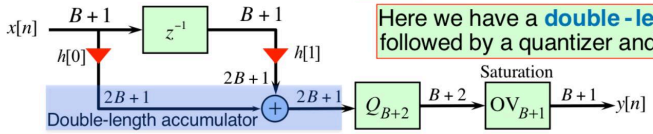



We assume that the filter coefficients $h[0],\ h[0],\cdots h[M-1]$ have been **quantized** to $(B+1)$-bits and the frequency response of the obtained filter satisfies the design specifications

**Ways to limit the wordlength in FIR filters**




Here we have quantization after each multiplier followed by saturation

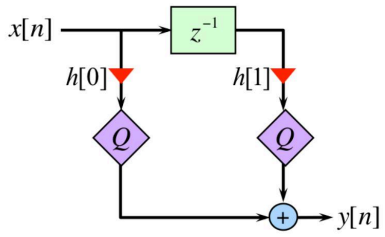



Here we have a **double-length accumulator** followed by a quantizer and saturation

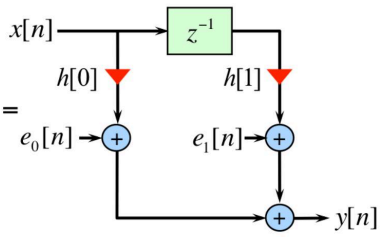


58

## Quantizers Modeled as Noise Sources

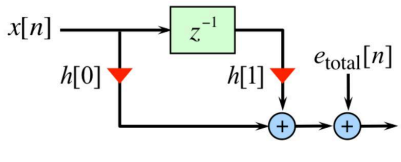



=




$$\sigma_{e_i}^2 = \frac{\Delta^2}{12},\quad \sigma_{\text{total}}^2 = 2\frac{\Delta^2}{12} = \frac{2}{3}2^{-2(B+1)};$$

$$\Delta = 2^{-B}$$




$$\sigma_{\text{total}}^2 = \frac{\Delta^2}{12} = \frac{1}{3}2^{-2(B+1)}$$

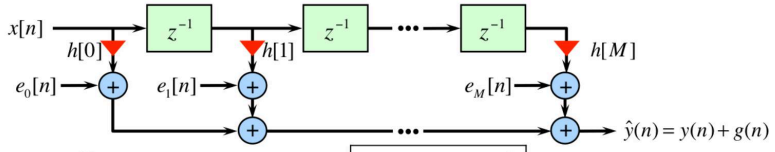


59

## Round-off Noise in FIR Filters

First we assume **no overflow** $\Rightarrow$

a) $\hat{y}(n) = \sum_{k=0}^{M} Q_{B+1}\{h[k]x[n-k]\} = \sum_{k=0}^{M} h[k]x(n-k) + \sum_{k=0}^{M} e_k[n]$




$g(n) = \sum_{k=0}^{M} e_k(n) \Rightarrow \mathrm{E}\{g(n)\} = 0, \quad \boxed{\sigma_g^2 = (M+1)\frac{\Delta^2}{12}}$

Rounding $\Rightarrow \sigma_e^2 = \frac{\Delta^2}{12}$

b) $\hat{y}(n) = Q_{B+1}\left\{\sum_{k=0}^{M} h[k]x[n-k]\right\} \Rightarrow$ **Use double length accumulator**

$\Rightarrow \boxed{\sigma_g^2 = \frac{\Delta^2}{12}}$ (only one quantizer)



60

## Scaling to Avoid Overflow

If we use two's complement we may avoid overflow in the final sum $y[n]$.
For $b_0 \blacktriangle b_1 b_2 \cdots b_B \Rightarrow |y[n]| < 1$. Hence,

$$|y[n]| = \left|\sum_{k=0}^{M} h[k]x[n-k]\right| \le \sum_{k=0}^{M} |h[k]||x[n-k]| < 1$$

Using a scaling factor $S$ and assuming that $|x[n]| \le X_m$ we can avoid overflow

If $S X_{\mathrm{m}} \sum_{k=0}^{M} |h[k]| < 1$

Worst-Case Scaling: $\boxed{S X_{\mathrm{m}} < 1 / \sum_{k=0}^{M} |h[k]|}$

Harmonic Scaling: $x[n] = X_{\mathrm{m}} \cos(\omega_0 n) \rightarrow y[n] = X_{\mathrm{m}} \left|H(\mathrm{e}^{\mathrm{j}\omega})\right| \cos\left[\omega_0 n + \angle H(\mathrm{e}^{\mathrm{j}\omega})\right]$

No overflow $\Rightarrow$ $\boxed{S X_{\mathrm{m}} < \frac{1}{\max_{\omega} \left|H(\mathrm{e}^{\mathrm{j}\omega})\right|}}$

Power Scaling: $S^2 < 1 / \sum_{n=0}^{M} h^2[n] = \boxed{1 / \frac{1}{2\pi} \int_{-\pi}^{\pi} \left|H(\mathrm{e}^{\mathrm{j}\omega})\right|^2 \mathrm{d}\omega}$

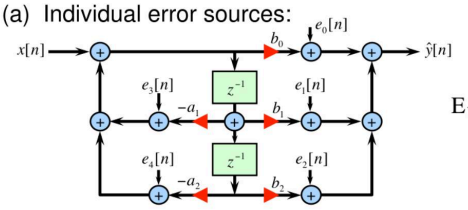


61

## Round-off Noise in Direct Form II Structures

- To avoid excessive coefficient quantization error effects, IIR filters are implemented as cascade structures using second-order sections.
- Consider direct form II normal structure for each SoS:

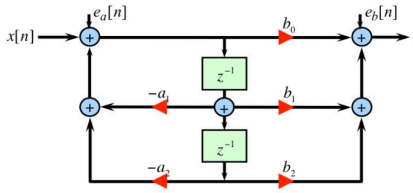
(a) Individual error sources:




$$\mathrm{E}\{e_i[n]\} = 0 \text{ and } \sigma_{e_i}^2 = \frac{\Delta^2}{12}, \quad i = 0,\ldots,4$$

$$\Delta = 2^{-B}$$

(b) Combined error sources:




$$e_a[n] = e_3[n] + e_4[n] \Rightarrow \sigma_{e_a}^2 = 2\frac{\Delta^2}{12}$$

$$e_b[n] = e_0[n] + e_1[n] + e_2[n] \Rightarrow \sigma_{e_b}^2 = 3\frac{\Delta^2}{12}$$

$$\hat{y}[n] = y[n] + g_a[n] + g_b[n]$$

$$g_a[n] = h[n] * e_a[n] \Rightarrow \sigma_{g_a}^2 = 2\frac{\Delta^2}{12}\sum_0^\infty |h[n]|^2$$

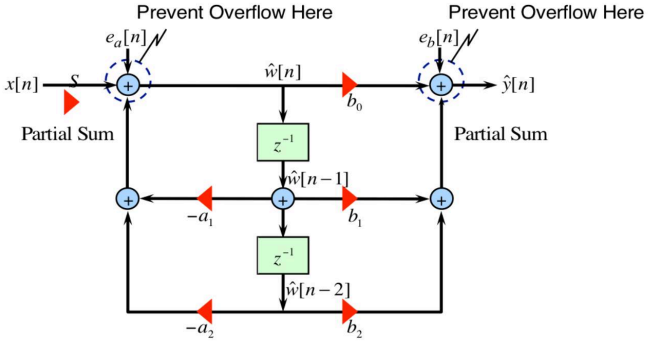
$$g_b[n] = e_b[n] \Rightarrow \sigma_{g_b}^2 = 3\frac{\Delta^2}{12}$$



62

## Overflow of Intermediate Results - Scaling

Consider the combined error model:

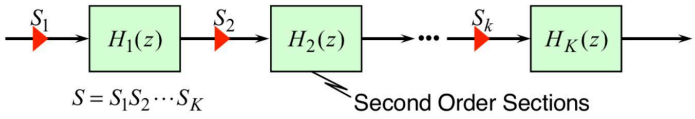



- We need to prevent overflow at the circled adder nodes
- Thus we need to scale the input using the scaling factor $S$
- This scaling can be done using any one of the scaling criteria on Slide-49: (a) Worst-case, (b) Harmonic, to (c) power-scaling



63

## Cascade Structure




- Choose $S$ so that $\left|H\left(e^{j\omega}\right)\right| \approx 1$ in the passband
- Choose $S_k$ so that $S_k X_{m,k} < \dfrac{1}{\max\limits_{\omega}\left|H_k\left(e^{j\omega}\right)\right|}$
- **Pairing:** See next slide
- **Section Ordering:** Use increasing or decreasing closeness of poles to the unit circle



64

## Issues in Cascade Connection of Second-Order Sections

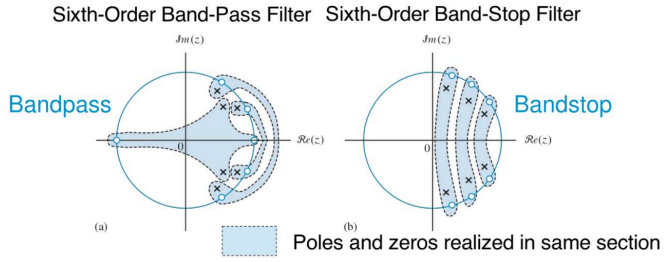
1. **Pairing:** Which poles and zeros are paired together in a particular section?
2. **Ordering:** In which sequence are the sections cascaded?
3. **Structuring:** What does each second-order section look like (e.g. direct form 1, direct form 2, number and position of quantizers, quantization characteristic, overflow characteristic)?
4. **Scaling:** What constant factor should be used to multiply the signals between the different sections?

MATLAB function: `[sos,G] = tf2sos(b,a,'order','scale');`
where `order` is `up` (default) or `down`
`scale` is `none` (default), `2`, or `inf`



65

## Pairing of Poles and Zeros




Rules of Thumb (Jackson-1970):

**Rule 1.** Pair the pole which is closest to the unit circle with the zero which is nearest to it. Repeat this process until all poles and zeros have been paired.

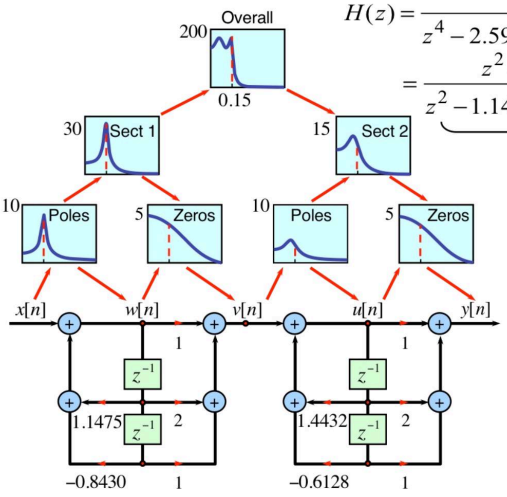
**Rule 2.** Order the second-order sections obtained by using Rule 1 according to the closeness of their poles to the unit circle. Use either decreasing or increasing closeness.

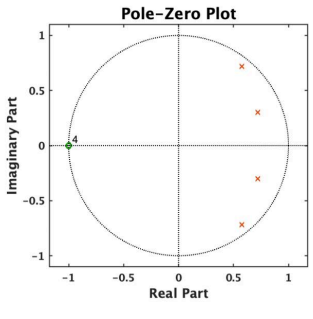


66

## Scaling Example: Lowpass Filter

$$H(z) = \frac{z^4 + 4z^3 + 6z^2 + 4z + 1}{z^4 - 2.5907z^3 + 3.1119z^2 - 1.9198z + 0.5166} = \underbrace{\frac{z^2 + 2z + 1}{z^2 - 1.1475z + 0.8430}}_{H_1(z)} \cdot \underbrace{\frac{z^2 + 2z + 1}{z^2 - 1.4432z + 0.6128}}_{H_2(z)}$$






67

## Limit Cycle Oscillations

- In the round-off noise analysis, we assumed that the quantization errors were uncorrelated with signal values.
- When signals have low (almost-zero) amplitude or have periodic values tuned to the sampling rate, then the errors are **correlated**.
- These correlated errors give rise to oscillatory behavior even in the absence of input signal.
- This behavior is called **limit cycle oscillations**.
- There are two types of limit cycles:
  - Granular limit cycles: These have small amplitudes and can be minimized using more bits in the quantizer.
  - Overflow limit cycles: These have large amplitudes and are due to overflow characteristics. Probleantic and should be avoided.



68

## Granular Limit Cycles

As an example, consider a first-order IIR filter given by

$$y(n) = -\tfrac{1}{2}y(n-1) + \tfrac{7}{8}\delta(n); \quad y(-1) = 0, \quad n \geq 0$$

Highpass filter: pole: $z = -1$

Then

$$\hat{y}(n) = Q\left[-\tfrac{1}{2}\hat{y}(n-1)\right] + \tfrac{7}{8}\delta(n); \quad \hat{y}(-1) = 0, \quad n \geq 0$$

Let $B = 3 \Rightarrow -\tfrac{1}{2} \equiv 1\text{▲}110$ and $\tfrac{7}{8} \equiv 0\text{▲}111$. Then

$$\hat{y}(0) = x(0) = +\tfrac{7}{8} \equiv 0\text{▲}111$$

$$\hat{y}(1) = Q\left[-\tfrac{1}{2}\hat{y}(0)\right] = Q\left[-\tfrac{1}{2}\left(+\tfrac{7}{8}\right)\right] = Q\left[-\tfrac{7}{16}\right] = -\tfrac{1}{2} \equiv 1\text{▲}100$$

$$\hat{y}(2) = Q\left[-\tfrac{1}{2}\hat{y}(1)\right] = Q\left[-\tfrac{1}{2}\left(-\tfrac{1}{2}\right)\right] = Q\left[+\tfrac{1}{4}\right] = -\tfrac{1}{4} \equiv 0\text{▲}010$$

$$\hat{y}(3) = Q\left[-\tfrac{1}{2}\hat{y}(2)\right] = Q\left[-\tfrac{1}{2}\left(+\tfrac{1}{4}\right)\right] = Q\left[-\tfrac{1}{8}\right] = -\tfrac{1}{8} \equiv 1\text{▲}111$$

$$\hat{y}(4) = Q\left[-\tfrac{1}{2}\hat{y}(3)\right] = Q\left[-\tfrac{1}{2}\left(-\tfrac{1}{8}\right)\right] = Q\left[+\tfrac{1}{16}\right] = +\tfrac{1}{8} \equiv 0\text{▲}001$$

$$\hat{y}(5) = Q\left[-\tfrac{1}{2}\hat{y}(4)\right] = Q\left[-\tfrac{1}{2}\left(+\tfrac{1}{8}\right)\right] = Q\left[-\tfrac{1}{16}\right] = -\tfrac{1}{8} \equiv 1\text{▲}111$$



69

## Granular Limit Cycles

Example (continued)

$$y(n) = -\tfrac{1}{2}y(n-1) + \tfrac{7}{8}\delta(n); \quad y(-1) = 0, \quad n \geq 0$$

Highpass filter: pole: $z = -1$

$$\hat{y}(3) = Q\left[-\tfrac{1}{2}\hat{y}(2)\right] = Q\left[-\tfrac{1}{2}\left(+\tfrac{1}{4}\right)\right] = Q\left[-\tfrac{1}{8}\right] = -\tfrac{1}{8} \equiv 1\blacktriangle 111$$

$$\hat{y}(4) = Q\left[-\tfrac{1}{2}\hat{y}(3)\right] = Q\left[-\tfrac{1}{2}\left(-\tfrac{1}{8}\right)\right] = Q\left[+\tfrac{1}{16}\right] = +\tfrac{1}{8} \equiv 0\blacktriangle 001$$

$$\hat{y}(5) = Q\left[-\tfrac{1}{2}\hat{y}(4)\right] = Q\left[-\tfrac{1}{2}\left(+\tfrac{1}{8}\right)\right] = Q\left[-\tfrac{1}{16}\right] = -\tfrac{1}{8} \equiv 1\blacktriangle 111$$

Thus $y(n) = \pm\tfrac{1}{8}$ for $n \geq 5$ while $y(n) = \tfrac{7}{8}\left(-\tfrac{1}{2}\right)^n u(n) \rightarrow 0$. Hence

$$e(n) = \hat{y}(n) - y(n) \rightarrow \pm\tfrac{1}{8}.$$

Hence for the first-order systems, it can be showed that

$$Q[\alpha\hat{y}(n-1)] = \begin{cases} \hat{y}(n-1), & \alpha > 0, \\ -\hat{y}(n-1), & \alpha < 0. \end{cases}$$

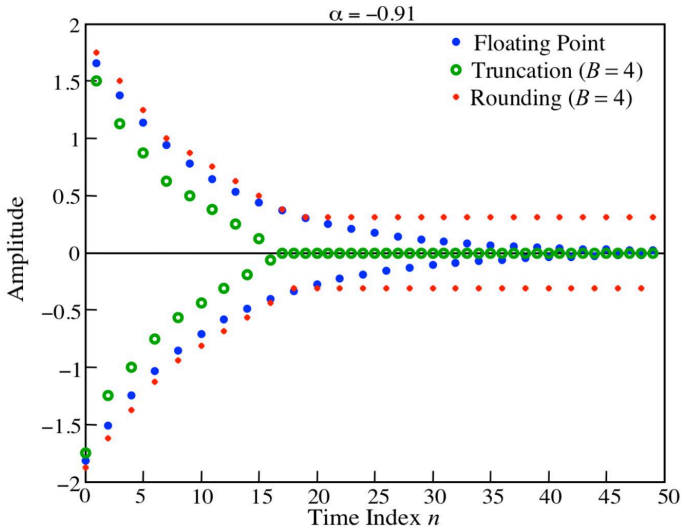
$$\left|Q\left[\alpha\hat{y}(n-1)\right] - \alpha\hat{y}(n-1)\right| \leq \frac{\Delta}{2} \quad \Rightarrow \quad \left|\hat{y}(n-1)\right| \leq \frac{\Delta}{2(1-|\alpha|)}$$

Dead Band



70

## Example: Granular Limit Cycle First-order system






71

## Overflow Limit Cycles

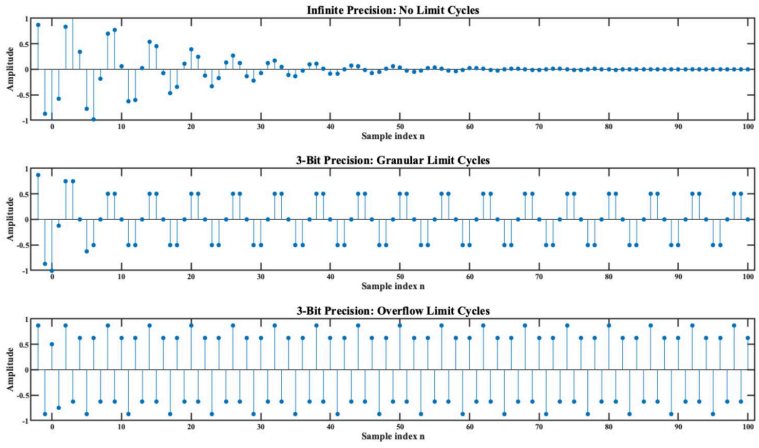
- This type of limit cycle is also a zero-input behavior that gives an oscillatory output.
- It is due to overflow in the addition even if we ignore multiplication or product quantization in the filter implementation.
- This is a more serious limit cycle because the oscillations can cover the entire dynamic range of the quantizer.
- It can be avoided in practice by using the saturation characteristics instead of overflow in the quantizer.
- In the modern digital signal processors, a double-length internal accumulator is used to collect additions with overflow characteristics to preserve intermediate calculations and then the results are saturated if an overflow is detected.
- This architecture effectively eliminates overflow limit cycles.



72

## Example on Limit Cycles

$$y[n] = 0.875y[n-1] - 0.875y[n-2], \quad y[-1] = -0.875, \quad y[-2] = 0.875$$
$$B = 3$$






73