Lecture 6

Quantization Effects

Fixed Point Implementation

- In previous lecture, we talked about the implementation of LTI systems without taking into consideration the fact that these systems are implemented using processing units that have limited number of bits to represent data as well as filter coefficients.
- In this lecture, we consider the effect of the quantization of the filter coefficient and truncation of the signal needed to avoid overflow.
- We first, consider the representation of numbers as it appears in digital computers.
- The limited number of bits used to represent the numbers, results in loss of precision that results in round-off errors and nonlinear effects.
- The objective would be to quantify these effects.

Representation of the Numbers

- In the arithmetic we use ever day, when we write 123.45, we mean:

$$1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2}$$

- the reason being that our arithmetic uses base 10. In fact, for someone not familiar with this convention, we should have written the number as:

$$(123.45)_{10}$$

- to emphasize that the base was 10.
- Computers, use base 2 instead of base 10. So, a typical number intended for machine may be written as:

$$(101.01)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$$

Representation of the Numbers

- In general a binary number can be represented as

$$X = (b_{-A}, \ldots, b_{-1} \cdot b_0, b_1, \ldots, b_B)_2 = \sum_{i=-A}^{B} b_i \, 2^{-i}$$

- Let's consider the case of A=n-1 and B=0, then we have an n-bit integer between 0 and $2^n - 1$.
- Next consider the case of A=0 and B=n-1. Then we have a fraction between 0 and $1 - 2^{-n}$.
- Note that if we factor 2^A and integer or a mix of integer and fraction can be change into a fraction, we only need to focus on fractions, i.e., numbers between 0 (0.000…) and 1 (0.111…).

Negative Numbers

- There are three ways to represent negative numbers:
  - Sign-magnitude format,
  - 1's complement format and,
  - 2's complement format.
- In all three formats a positive number X is represented as:

$$X = 0.b_1 b_2 \cdots b_B = \sum_{i=1}^{B} b_i \cdot 2^{-i}, \qquad X \geq 0$$

- Note that the Most Significant Bit (MSB) is a zero.

Negative Numbers

- A negative number: $$X = -0.b_1 b_2 \cdots b_B = -\sum_{i=1}^{B} b_i \cdot 2^{-i}$$
- In Sign-magnitude will be represented as:

$$X_{\mathrm{SM}} = 1.b_1 b_2 \cdots b_B, \qquad \text{for } X \leq 0$$

- i.e., with MSB changed to 1.
- In 1's complement all bits will be complemented:

$$X_{1\mathrm{C}} = 1.\bar{b}_1 \bar{b}_2 \cdots \bar{b}_B, \qquad X \leq 0$$

- where $\bar{b}_i = 1 - b_i$.
- Note that:

$$X_{1\mathrm{C}} = 1 \times 2^0 + \sum_{i=1}^{B} (1 - b_i) \cdot 2^{-i} = 2 - 2^{-B} - |X|$$

Negative Numbers

- In two's complement format, the representation for a negative number is obtained by subtracting the corresponding positive number (the absolute value of) the negative number from 2. Equivalently, we can add a list significant bit (LSB) to the 1's complement:

$$X_{2C} = 1.\bar{b}_1\bar{b}_2\cdots\bar{b}_B + 00\cdots01, \qquad X < 0$$

- It is easily seen that, this is equivalent to subtraction the absolute value of the number from 2:

$$X_{2C} = X_{1C} + 2^{-B} = 2 - |X|$$

- Now, let's use geometric progression sum formula to show that $X_{2C}$is in fact the negative number .

$$X = -0.b_1b_2\cdots b_B = -\sum_{i=1}^{B} b_i \cdot 2^{-i}$$

Negative Numbers

- We have: $S_n = 1 + q + q^2 + q^3 + \cdots + q^{n-1} = \frac{1-q^n}{1-q}$.
- Or, equivalently: $S_n = q + q^2 + q^3 + \cdots + q^n = q\frac{1-q^n}{1-q}$.
- Letting $q = \frac{1}{2}$ and $n = B$, we get: $\sum_{i=1}^{B} 2^{-i} = 1 - 2^{-B}$ or $\sum_{i=1}^{B} 2^{-i} + 2^{-B} = 1$.
- Using this we write:

$$X = -\sum_{i=1}^{B} b_i \cdot 2^{-i} + 1 - 1 = -1 + \sum_{i=1}^{B} (1 - b_i)2^{-i} + 2^{-B}$$

$$= -1 + \sum_{i=1}^{B} \bar{b}_i \cdot 2^{-1} + 2^{-B} = 1 + \sum_{i=1}^{B} (1 - b_i)2^{-i} + 2^{-B} - 2$$

$$= X_{2C} - 2 = -|X|$$

Negative Numbers: Example

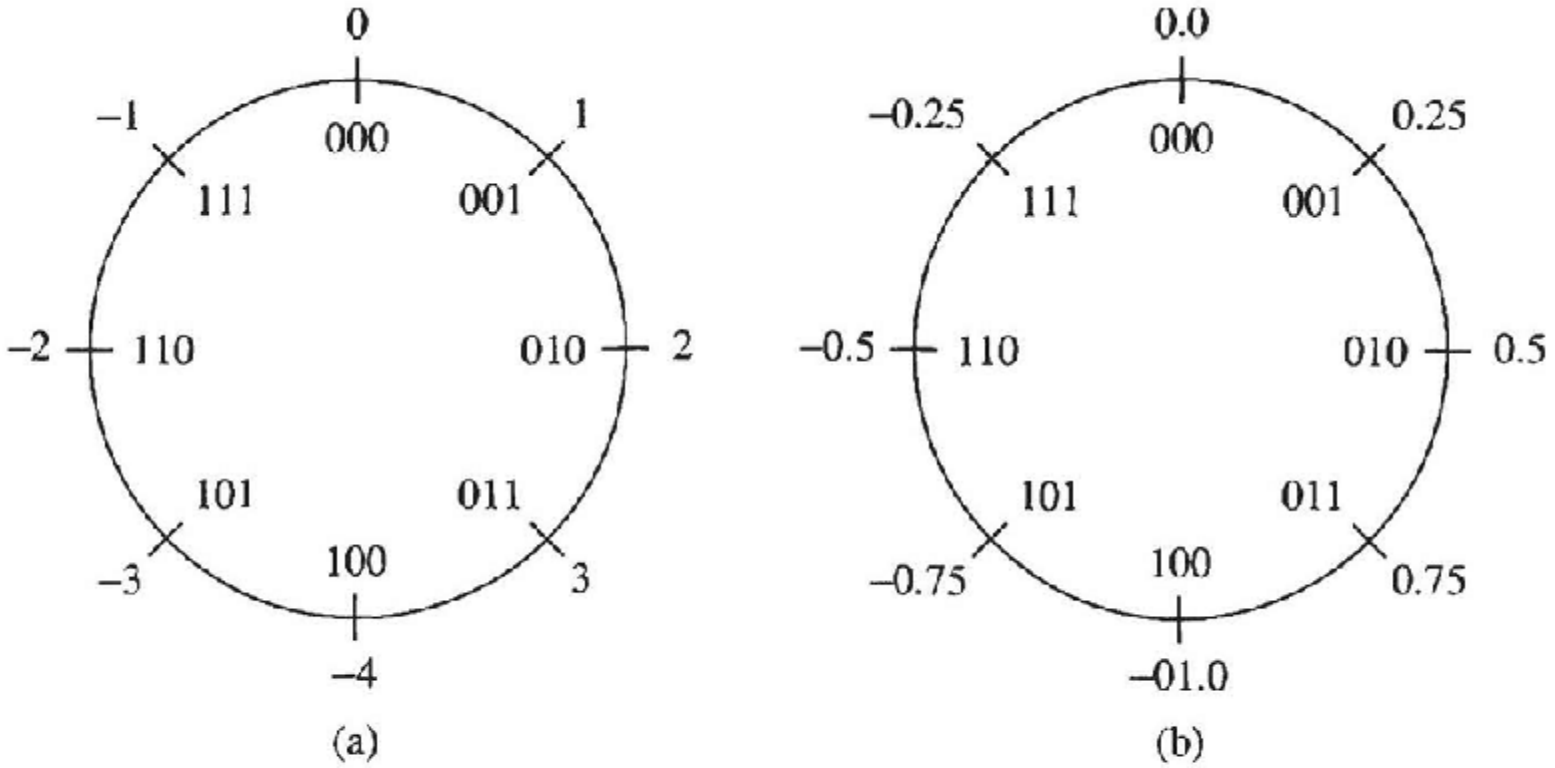
- Express: $\frac{7}{8}$ and $-\frac{7}{8}$ in one's and two's complement formats.
- Solution:
- $X = \frac{7}{8}$ is equal to $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} = 2^{-1} + 2^{-2} + 2^{-3}$, So X=0.111.
- $-\frac{7}{8}$ in 1's complement format is: $X_{1C} = 1.000$.
- In 2's complement: $X_{2C} = 1.000 + 0.001 = 1.001$.

Addition

- In 1's and 2's complement addition is performed bit by bit modulo-2.
- In one's complement if there is a carry in MSB, it is moved to the LSB.
- IN two's complement if there is a carry in the MSB it is deleted.
- Example: $\frac{1}{2} - \frac{3}{8}$.
- In 1's complement: $\frac{1}{2} = \frac{4}{8} = 0100$, $\frac{3}{8} = 0011$ and $-\frac{3}{8} = 1100$.
- So, $0100 \oplus 1100 = 1000$.
- After moving the carry in MSB to LSB, we get 0001, i.e., $\frac{1}{8}$.
- In 2's complement , $0100 \oplus 1101 = 1001$.
- Deleting the carry, we get 0001.

Modulo Arithmetic

- Most Digital Signal Processors use 2's complement format. So for (B+1)-bit numbers the range is from -1 to $1 - 2^{-B}$. We can represent numbers in this range on a wheel:



Counting wheel for 3-bit two's-complement numbers: (a) integers and (b) fractions.

Modulo Arithmetic

- 2's complement operation is actually modulo 2^{B+1}arithmetic. That is, any number that falls outside this range is brought back in-range by subtractiong as many times 2^{B+1}as necessary.

Multiplication:

Multiplication of a two b-bit numbers results in a number with 2b bits. In fixed point arithmetic, we need to truncate the number, i.e., dropping the b Least Significant Bits (LSB's). This results in Truncation or Round-off Error.

Floating-Point Representation

- So far, we have talked about Fixed-Point representation of the numbers. In fixed-point representation the numbers in the range of, say between x_{min} to x_{max} are represented each with b bits. Therefore there are $m = 2^b$ values and the resolution will be $\Delta = \frac{x_{max} - x_{min}}{m-1}$. So, in fixed point representation the resolution is fixed. It is reasonable to demand for higher resolution for smaller numbers and lower resolution for large numbers.
- **Floating-Point** representation allows different step sizes (resolution) and also wider dynamic range for the same number of bits.
- In floating-point representation a number X is represented as: $X = M.2^E$
- where the mantissa M is the fractional part of the number and falls in the range of $0.5 \leq M < 1$. The exponent E is a positive or negative integer.

Floating-Point Representation

- As an example, the number $X_1 = 5$ is represented by:
  - $M_1 = 0.101000$ and $E_1 = 011$
- and $X_2 = \frac{3}{8}$ is represented by:
  - $M_2 = 0.110000$ and $E_2 = 101$,
- where MSB of the exponent is the sign bit.
- To multiply floating point numbers, we multiply their mantissa's and add their exponents. So,
  - $X_1 X_2 = M_1 M_2 . 2^{E_1 + E_2} = (0.011110) . 2^{010} = (0.111100) . 2^{001}$.
- For addition, the exponents have to be the same. We can achieve this by shifting the mantissa of the smaller number to the right and compensating the corresponding exponent: $M_2 = 0.000011$ and $E_2 = 011$. Now:
  - $X_1 + X_2 = (0.101011) . 2^{011}$
- Note that this may result in loss of precision if we lose bits as a result of the shift.

Floating-Point versus Fixed-Point

- Assume that we have a 32-bit processor and we use fixed point. The largest number that we can show is:

$$2^{32} - 1 = 4{,}294{,}967{,}295$$

- To show negative and positive number, we can use the MSB as a sign bit and other 31 bits to represent magnitude. Then the range of numbers will be:

$$-(2^{31} - 1) = -2{,}147{,}483{,}647 \quad \text{to} \quad (2^{31} - 1) = 2{,}147{,}483{,}647$$

- Note that the resolution is one no matter what the number is.
- We can increase the resolution by assigning 10 bits to a fraction parts, 21 bits to the integer part and 1 sign bit. Then the dynamic range is:

$$-(2^{31} - 1) \cdot 2^{-10} = -(2^{21} - 2^{-10}) \quad \text{to} \quad (2^{31} - 1) \cdot 2^{-10} = 2^{21} - 2^{-10}$$

- or $-2{,}097{,}151.999 \quad \text{to} \quad 2{,}097{,}151.999$
- We have increased the resolution 1000 times ($2^{10} = 1024$) while reducing the dynamic range 1000 times.

Floating-Point versus Fixed-Point

- Now let's use floating-point representation. Assume that we use 23 bits plus a sign bit for the mantissa and 7 bits plus a sign bit for the exponent.
- The smallest number we can have is:

sign	23 bits	sign	7 bits	
0.	$100\cdots0$	1	1111111	$= \frac{1}{2} \times 2^{-127} \approx 0.3 \times 10^{-38}$

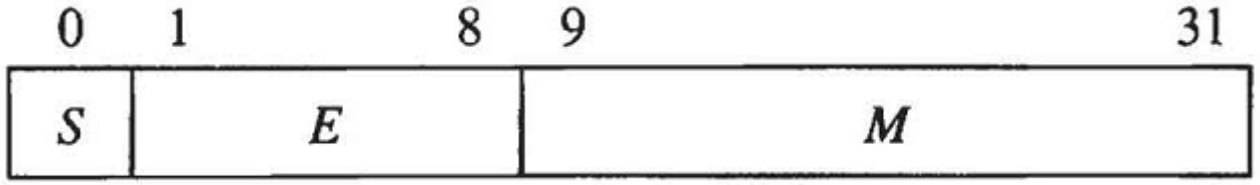
- And the largest number is:

sign	23 bits	sign	7 bits	
0	$111\cdots1$	0	1111111	$= (1 - 2^{-23}) \times 2^{127} \approx 1.7 \times 10^{38}$

- So, we have achieved a dynamic range of $10^{76}$with varying resolution. That is, we have finer resolution for small numbers and coarser resolution for larger numbers.

IEEE 754 Standard

- When using floating point representation, there can be ambiguities concerning the representation zero, overflow and other issues.
- In order to define a common floating-point format, IEEE has introduced the IEEE 754 standard. IEEE 754 is widely used.
- For a 32-bit machine, the IEEE 754 standard single precision, floating point number is represented as $X = (-1)^s . 2^{E-127}(M)$, where:



- The dynamic range of IEEE 754 floating-point numbers is:
- i.e.,

$$2^{-126} \times 2^{-23} \text{ to } (2 - 2^{-23}) \times 2^{127}$$

$$\text{from } 1.18 \times 10^{-38} \text{ to } 3.40 \times 10^{38}$$

IEEE 754 Standard

- This number has the following interpretations:

$$\text{If } E = 255 \text{ and } M \neq 0, \text{ then } X \text{ is not a number}$$

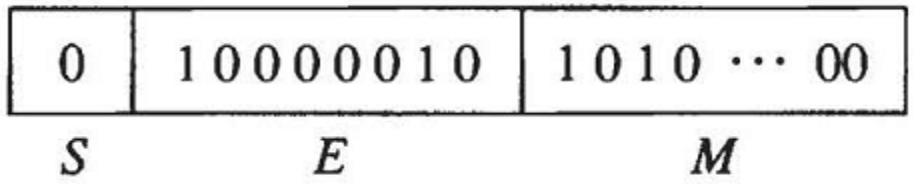
$$\text{If } E = 255 \text{ and } M = 0, \text{ then } X = (-1)^S \cdot \infty$$

$$\text{If } 0 < E < 255, \text{ then } X = (-1)^S \cdot 2^{E-127}(1.M)$$

$$\text{If } E = 0 \text{ and } M \neq 0, \text{ then } X = (-1)^S \cdot 2^{-126}(0.M)$$

$$\text{If } E = 0 \text{ and } M = 0, \text{ then } X = (-1)^S \cdot 0$$

- For example:



- represents:

$$X = -1^0 \times 2^{130-127} \times 1.1010\ldots0 = 2^3 \times \frac{13}{8} = 13$$

Truncation Error

- Assume that in the course of computation, we have got a number with b_u bits, while our machine can only handle up to $b < b_u$ bits.
- Quantizing the number

$$x = \overbrace{0.1011\cdots01}^{b_u}$$

- to

$$x = \overbrace{0.101\cdots1}^{b}$$

- Results in the truncation error: $E_t = Q_t(x) - x$
- For positive numbers, truncation results in a smaller number. So,

$$-(2^{-b} - 2^{-b_u}) \leq E_t \leq 0$$

- For a negative number on the other hand: $0 \leq E_t \leq (2^{-b} - 2^{-b_u})$
- So, in signed-magnitude format:

Truncation Error

- In 2's complement format, the negative of a number is the result of subtracting the corresponding positive number from 2. So, the effect of truncation of a negative number, in 2's complement is to increase its magnitude, therefore,

$$x > Q_t(x)$$

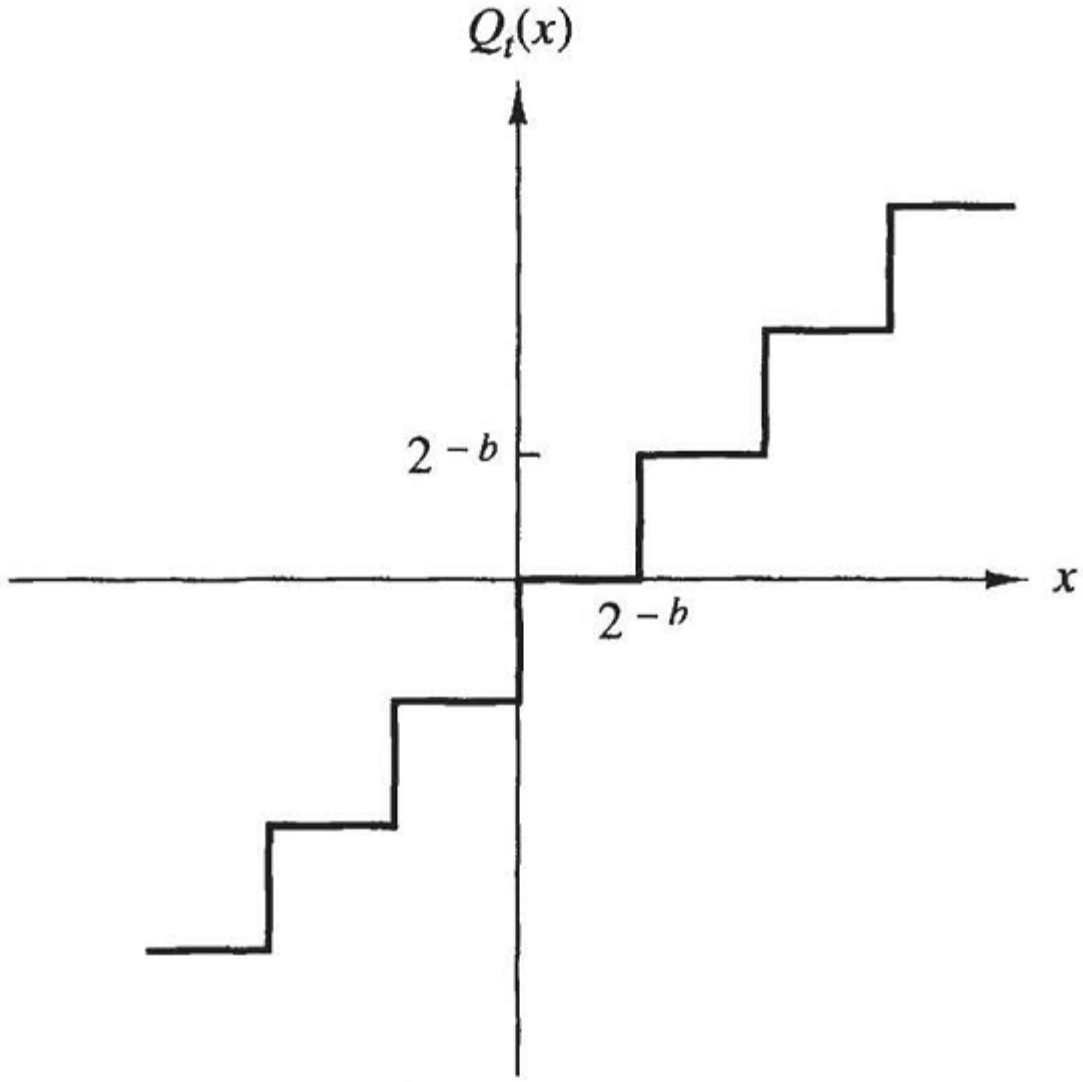
- and hence,

$$-(2^{-b} - 2^{-b_u}) \leq E_t \leq 0$$

- In summary:
  - In signed magnitude format $-(2^{-b} - 2^{-b_u}) \leq E_t \leq (2^{-b} - 2^{-b_u})$
  - and in 2's complement format $-(2^{-b} - 2^{-b_u}) \leq E_t \leq 0$

Truncation Error

- This figure shows the truncation
- error in 2's complement format.

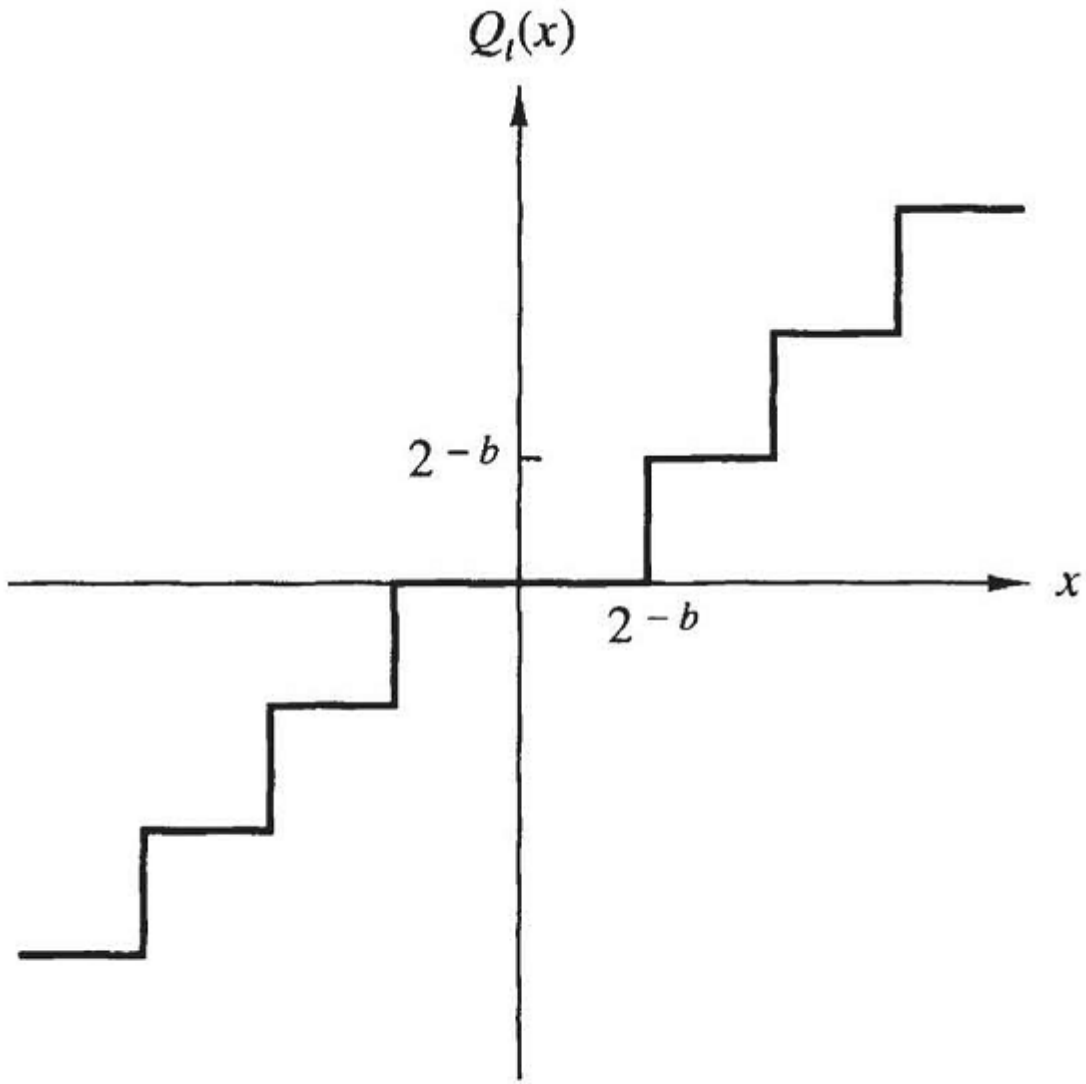


$$E_t = Q_t(x) - x$$

$$2^{-b} \le E_t \le 0$$

Truncation Error

- This figure shows the truncation
- error in sign-magnitude format.



$$E_t = Q_t(x) - x$$

$$-2^{-b} \le E_t \le 2^{-b}$$

Rounding Error

- Rounding a b_u bit number down to b bits, results in quantization error:

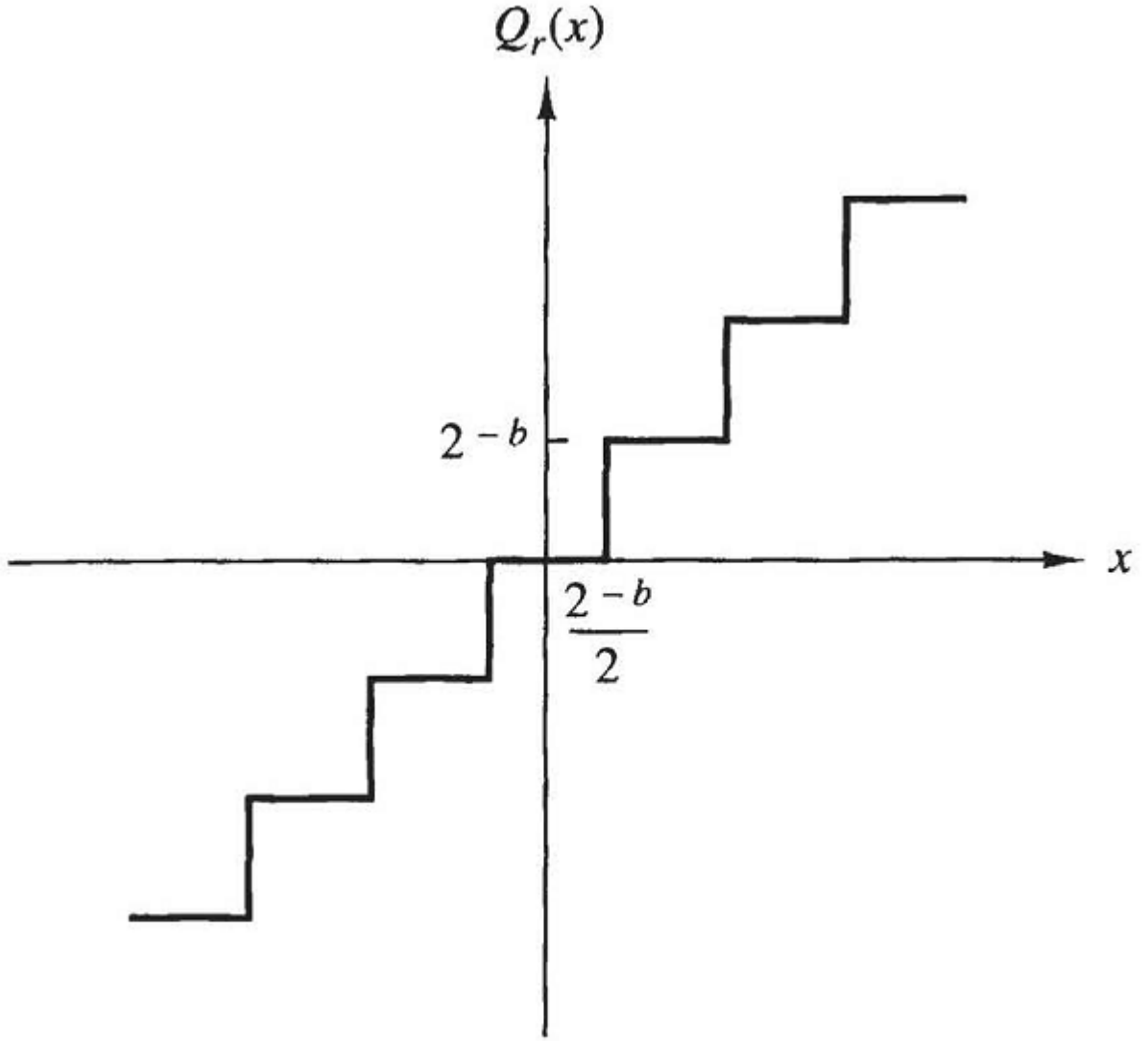
$$E_r = Q_r(x) - x$$

- Rounding only deals with the magnitude and therefore, is independent of the format used. It is symmetric around zero:

$$-\frac{1}{2}(2^{-b} - 2^{-b_u}) \leq E_r \leq \frac{1}{2}(2^{-b} - 2^{-b_u})$$

Rounding Error

- Following figure shows
- the rounding error effect:



$$E_r = Q_r(x) - x$$

$$-\frac{1}{2} \cdot 2^{-b} \leq E_r \leq \frac{1}{2} 2^{-b}$$

Floating-point Truncation and Rounding Errors

- In a floating-point format, the mantissa is either rounded or truncated. Unlike the fixed-point representation, the error depends on the number being quantized. It is possible to model the quantization as: $Q(x) = x + ex$ where e is called the relative error. So, the error is $Q(x) - x = ex$.
- For truncation in 2's complement format, for positive numbers:

$$-2^E 2^{-b} < e_t x < 0$$

- Since $2^{E-1} \leq x < 2^E$, we have:

$$-2^{-b+1} < e_t \leq 0, \qquad x > 0$$

- For negative numbers:

$$0 \leq e_t x < 2^E 2^{-b}$$

- So,

$$0 \leq e_t < 2^{-b+1}, \qquad x < 0$$

- For rounding

$$-2^E \cdot 2^{-b}/2 < e_r x \leq 2^E \cdot 2^{-b}/2$$

- Since x falls in the range of $2^{E-1} \leq x < 2^E$, we have: $-2^{-b} < e_r \leq 2^{-b}$

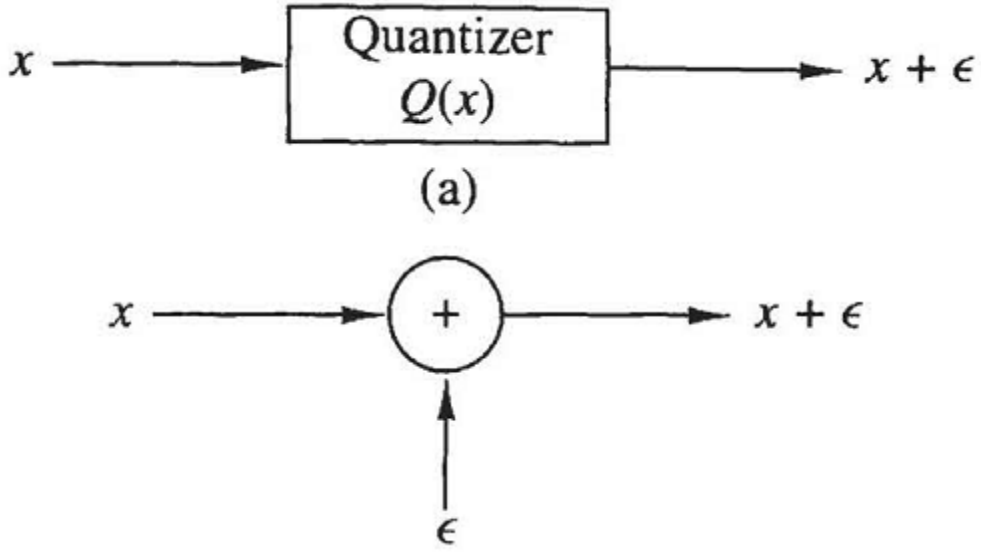
Statistical Modeling of Errors

- We can write the quantized value as:

$$Q(x) = x + \epsilon$$

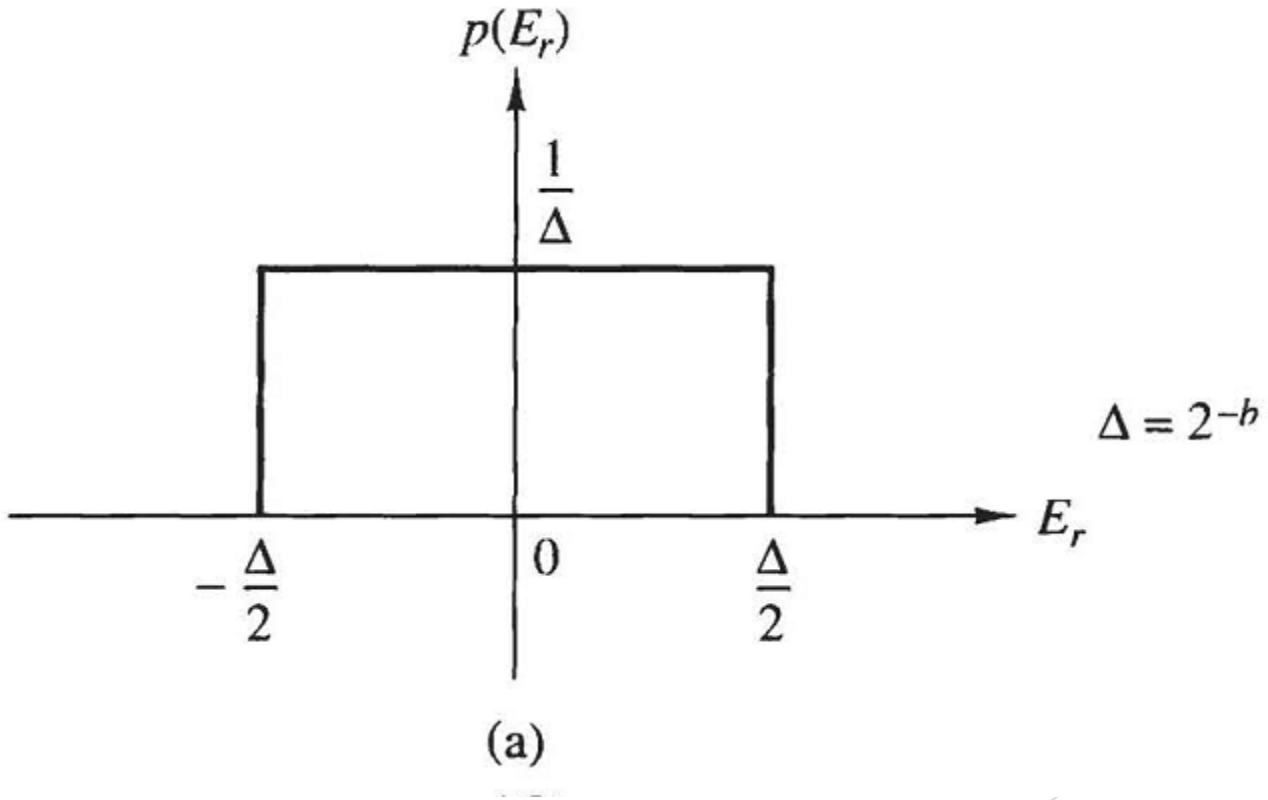
- where ε is either truncation or rounding error.

- The error ε is usually modelled as a random variable.



Statistical Modeling of Errors

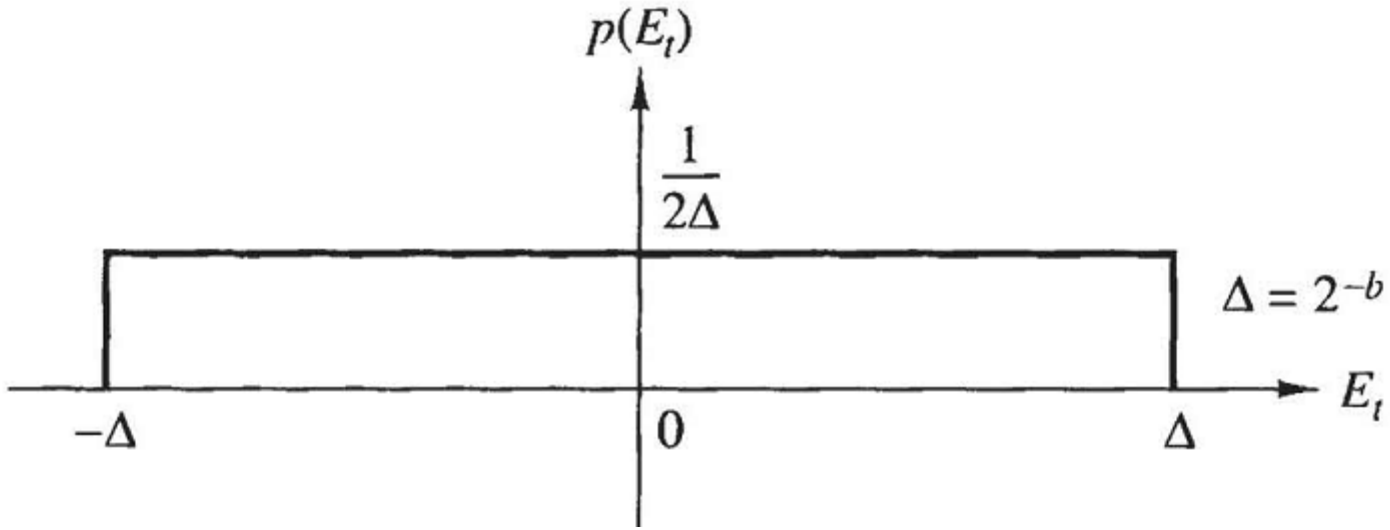
- For round-off error, we have



(a)

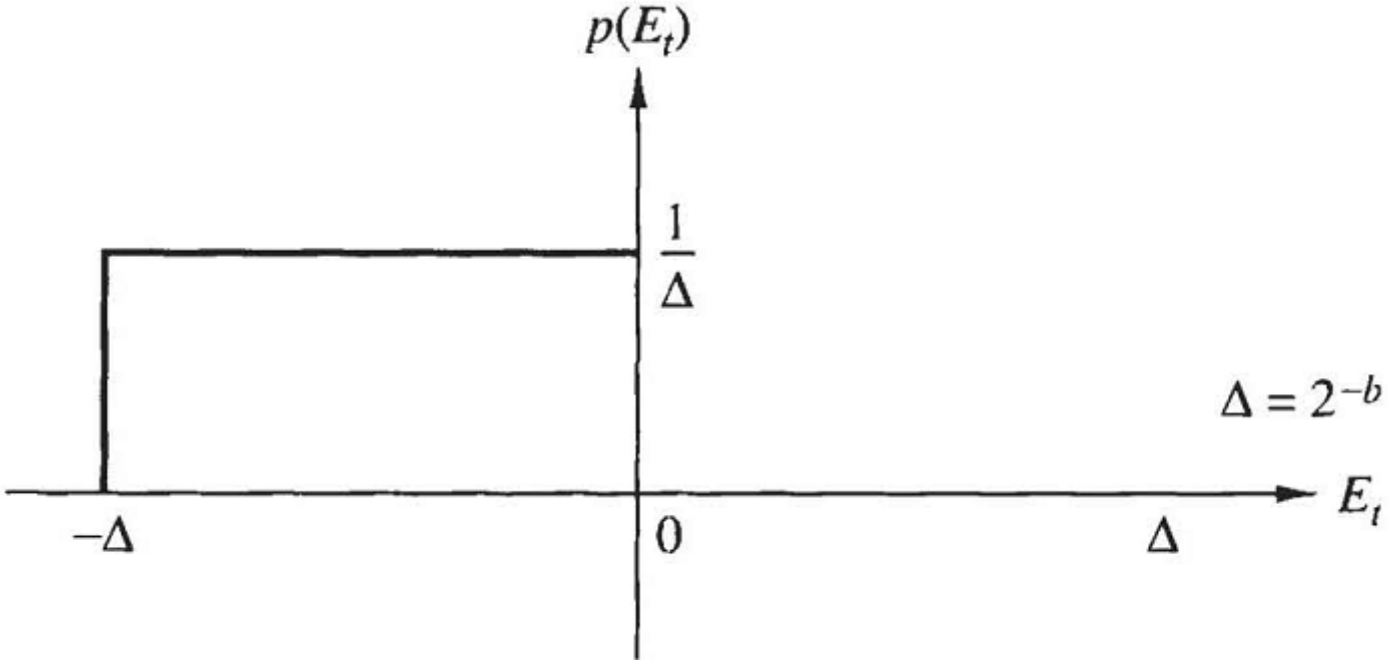
Statistical Modeling of Errors

- For truncation error in sign-magnitude format, we have



Statistical Modeling of Errors

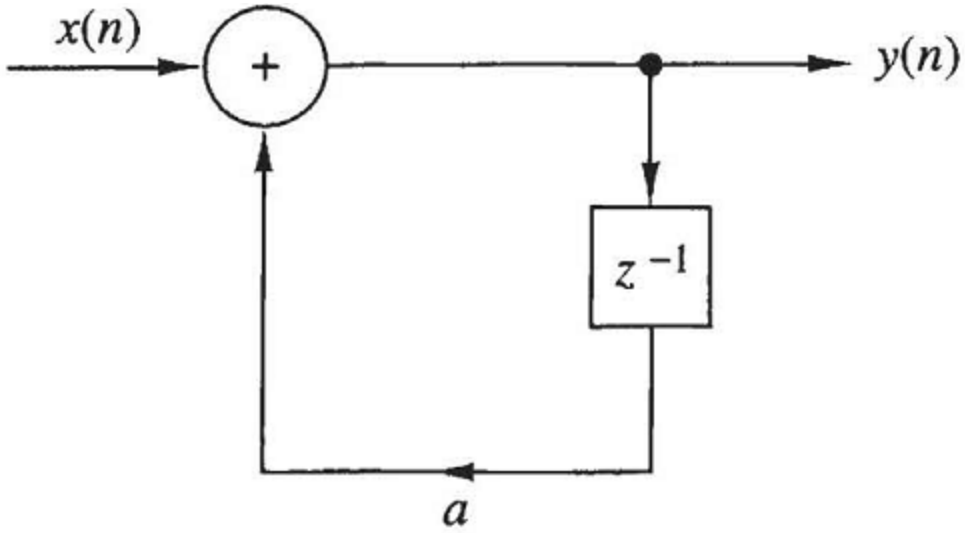
- For truncation error in two's complement format, we have



Statistical Characterization of Quantization Effects

- Consider a single-pole system:

$$y(n) = ay(n-1) + x(n)$$

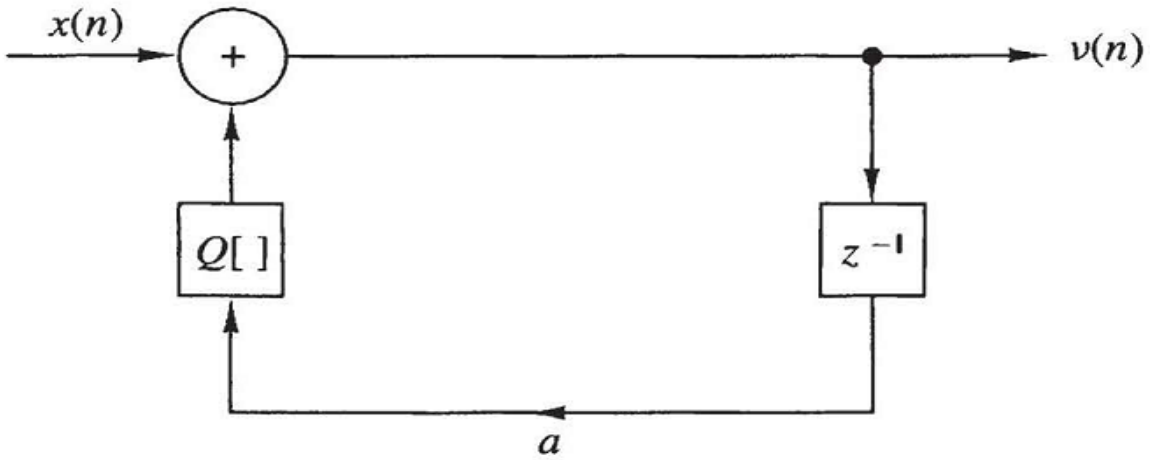


- Assume the quantized version of this system, i.e.,

$$v(n) = Q[av(n-1)] + x(n)$$

Statistical Characterization of Quantization Effects

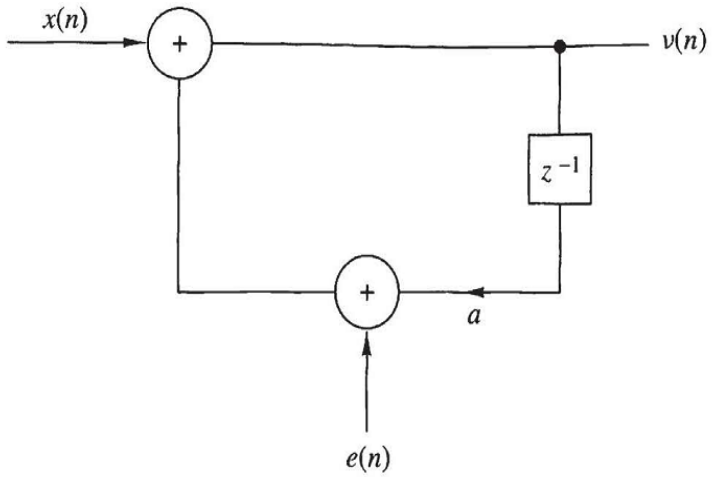
- This is a nonlinear difference equation represented by:



- The effect of rounding can be modelled as a noise sequence added to the actual product $av(n-1)$, i.e.,
  - $Q_r[av(n-1)] = av(n-1) + e(n)$
- The system now can be described by a linear difference equation
  - $v(n) = av(n-1) + x(n) + e(n).$

Statistical Characterization of Quantization Effects

- This linear difference equation can be represented by:



- It is clear that the output can be written as $v(n) = y(n) + q(n)$, where $y(n)$ is the output due to input $x(n)$ and $q(n)$ is the output due to the input $e(n)$.
- So we have:

$$y(n) + q(n) = ay(n-1) + aq(n-1) + x(n) + e(n)$$

Statistical Characterization of Quantization Effects

- To simplify the analysis, let's make the following assumptions:

1. For any n, the error sequence $\{e(n)\}$ is uniformly distributed over the range $(-\frac{1}{2} \cdot 2^{-b}, \frac{1}{2} \cdot 2^{-b})$. This implies that the mean value of $e(n)$ is zero and its variance is

$$\sigma_e^2 = \frac{2^{-2b}}{12}$$

2. The error $\{e(n)\}$ is a stationary white noise sequence. In other words, the error $e(n)$ and the error $e(m)$ are uncorrelated for $n \neq m$.
3. The error sequence $\{e(n)\}$ is uncorrelated with the signal sequence $\{x(n)\}$.

Statistical Characterization of Quantization Effects

- The last assumption allows us to separate the difference equation into two separate difference equations: one with input $x(n)$ and output $y(n)$ and another one with input $e(n)$ and output $q(n)$:

$$y(n) = ay(n-1) + x(n)$$

$$q(n) = aq(n-1) + e(n)$$

- To proceed, we use two properties of random variables going through linear systems that you may have seen in your probability course and we will also later see when we review random processes.
- Assume that $q(n)$ is the response of an LTI system with unit sample response $h(n)$ when the input is a random sequence $e(n)$ with mean m_e. Then,
  - $m_q = m_e \sum_{-\infty}^{\infty} h(n) = m_e H(0).$

Statistical Characterization of Quantization Effects

- The second property is concerned with the auto correlation function of $q(n)$:

$$\gamma_{qq}(n) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h(k)h(l)\gamma_{ee}(k-l+n)$$

- In the important special case where the random sequence is **WHITE** (memoryless or equivalently, spectrally flat), i.e.,

$$\gamma_{ee}(n) = \sigma_e^2 \delta(n)$$

- we have:
- $$\gamma_{qq}(n) = \sigma_e^2 \sum_{k=-\infty}^{\infty} h(k)h(k+n)$$

- The variance σ_q^2 is computed by evaluating $\gamma_{qq}(n)$ at $n = 0$. So,

$$\sigma_q^2 = \sigma_e^2 \sum_{k=-\infty}^{\infty} h^2(k) = \frac{\sigma_e^2}{2\pi} \int_{-\pi}^{\pi} |H(\omega)|^2 d\omega$$

Statistical Characterization of Quantization Effects

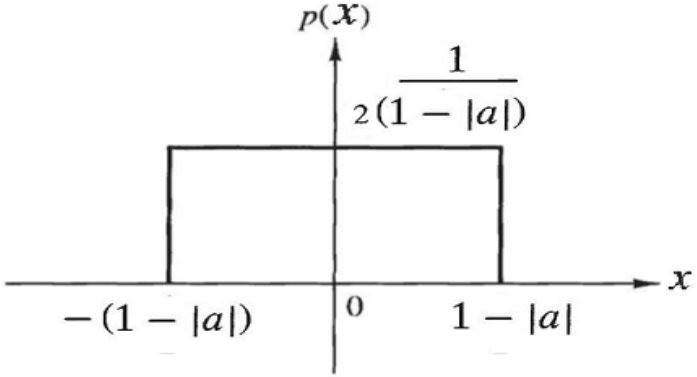
- For the single-pole filter under consideration, the unit sample response is:
  - $h(n) = a^n u(n)$
- So,

$$\sigma_q^2 = \sigma_e^2 \sum_{k=0}^{\infty} a^{2k} = \frac{\sigma_e^2}{1 - a^2}$$

- Now, let's find the variance of the signal $y(n)$.
  - $y(n) = \sum_{m=-\infty}^{\infty} h(m)x(n-m)$.
- Assume that the dynamic range of the computer is limited to (-1, 1), i.e., $|y(n)| < 1$ and assume that the input $x(n)$ is bounded by A_x. In order not to have overflow, we require that:
  - $|y(n)| = |\sum_{m=-\infty}^{\infty} h(m)x(n-m)| \leq A_x \sum_{m=-\infty}^{\infty} |h(m)| < 1$
- or $A_x < \frac{1}{\sum_{m=-\infty}^{\infty} |h(m)|}$.

Statistical Characterization of Quantization Effects

- For the single-pole filter under consideration, the unit sample response is:
  - $h(n) = a^n u(n)$
- So, $\sum_{m=-\infty}^{\infty} |h(m)| = \frac{1}{1-|a|}$. So, $A_x = 1 - |a|$.
- Assume that the input is uniformly distributed between $-A_x$ and $+A_x$.



- The variance of the input will be $\sigma_x^2 = (1 - |a|)^2/3$ and the output will have
- Variance

$$\sigma_y^2 = \sigma_x^2 \sum_{k=0}^{\infty} a^{2k} = \frac{\sigma_x^2}{1 - a^2}$$

Statistical Characterization of Quantization Effects

- The signal-to-Noise-Ratio (SNR) is defined as the ratio of the signal power σ_y^2 to quantization noise power σ_q^2:

$$\frac{\sigma_y^2}{\sigma_q^2} = \frac{\sigma_x^2}{\sigma_e^2} = (1 - |a|)^2 \cdot 2^{2(b+1)}$$

- We note that there is a considerable penalty paid as a consequence of scaling the input. This is particularly true when the pole is near the unit circle.

Quantization Effects: Example

- Example: Determine the variance of the round-off noise at the output of two cascade realizations of the filter with system function:

$$H(z) = H_1(z)H_2(z)$$

- where, $$H_1(z) = \frac{1}{1 - \frac{1}{2}z^{-1}}$$
- and $$H_2(z) = \frac{1}{1 - \frac{1}{4}z^{-1}}$$

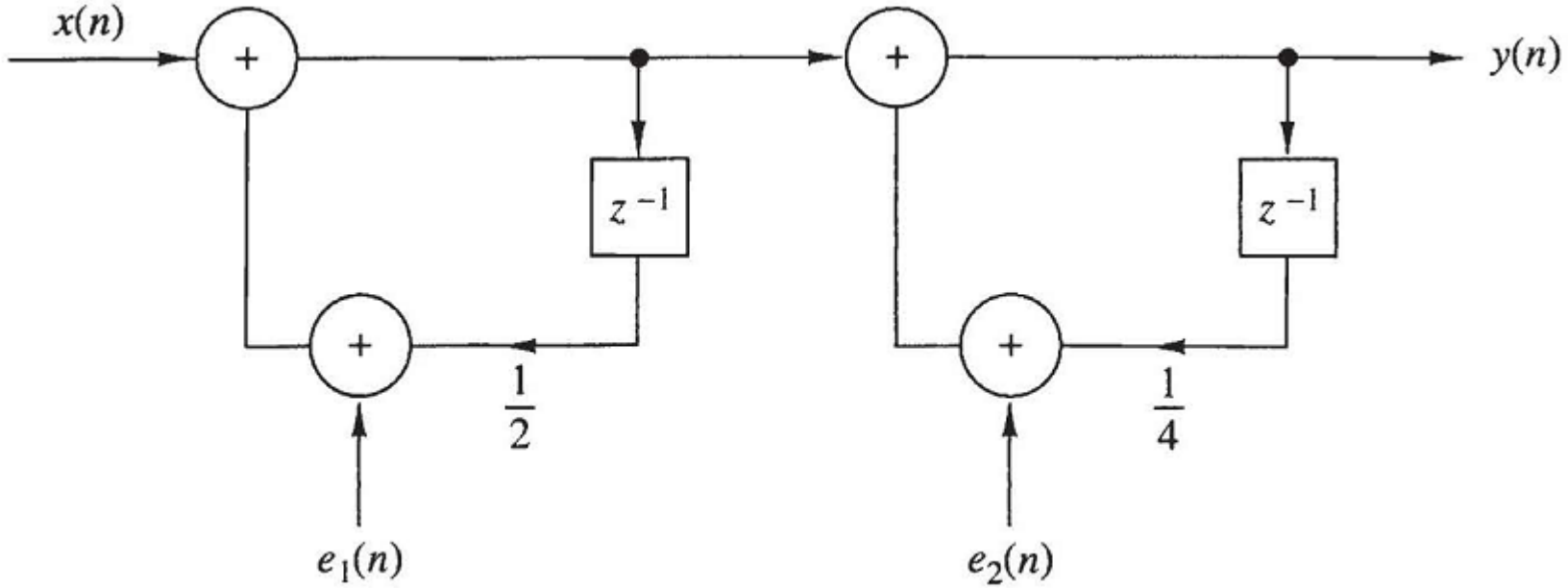
Solution: Let $h(n)$, $h_1(n)$ and $h_2(n)$ represent the sample responses corresponding to $H(z)$, $H_1(z)$ and $H_2(z)$, respectively. We have:

$$h_1(n) = (\tfrac{1}{2})^n u(n), \qquad h_2(n) = (\tfrac{1}{4})^n u(n)$$

$$h(n) = [2(\tfrac{1}{2})^n - (\tfrac{1}{4})^n]u(n)$$

Quantization Effects: Example

- In the first realization:

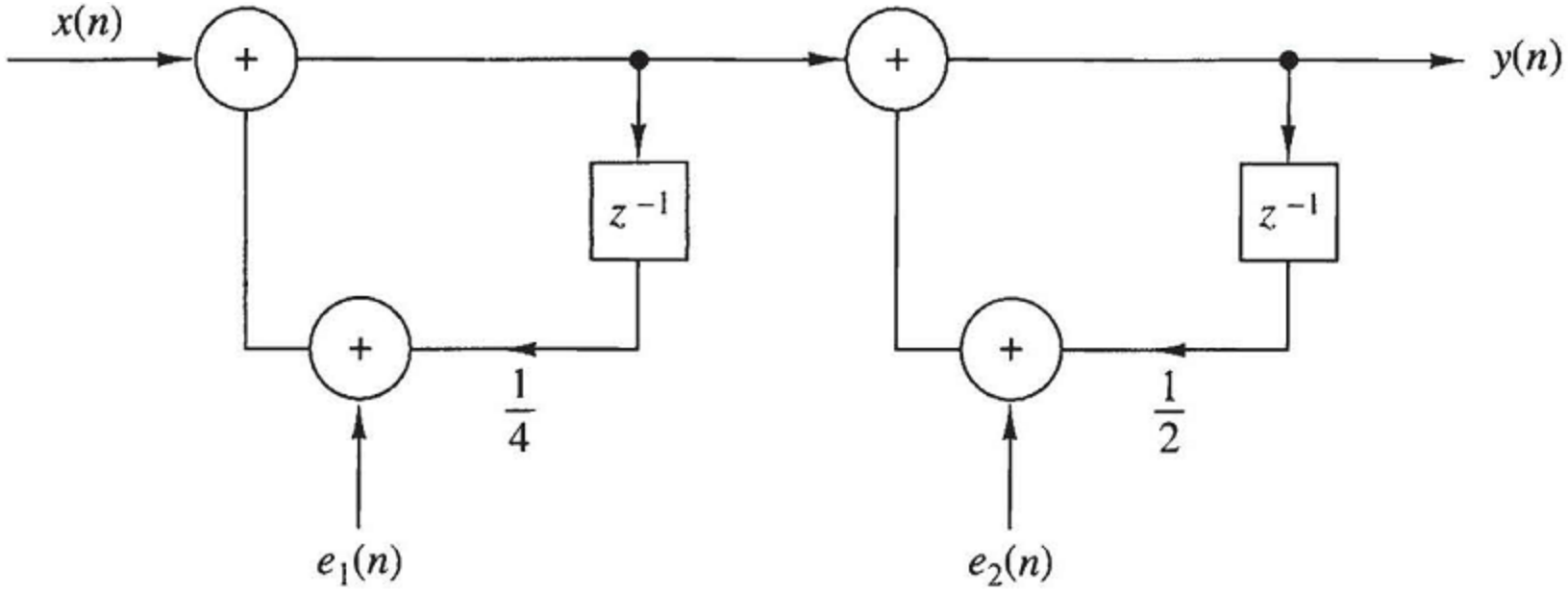


- we have:

$$\sigma_{q1}^2 = \sigma_e^2 \left[\sum_{n=0}^{\infty} h^2(n) + \sum_{n=0}^{\infty} h_2^2(n) \right]$$

Quantization Effects: Example

- In the second realization:



- we have:

$$\sigma_{q2}^2 = \sigma_e^2 \left[\sum_{n=0}^{\infty} h^2(n) + \sum_{n=0}^{\infty} h_1^2(n) \right]$$

Quantization Effects: Example

- We have:

$$\sum_{n=0}^{\infty} h_1^2(n) = \frac{1}{1-\frac{1}{4}} = \frac{4}{3}$$

$$\sum_{n=0}^{\infty} h_2^2(n) = \frac{1}{1-\frac{1}{16}} = \frac{16}{15}$$

- So:

$$\sum_{m=0}^{\infty} h^2(n) = \frac{4}{1-\frac{1}{4}} - \frac{4}{1-\frac{1}{8}} + \frac{1}{1-\frac{1}{16}} = 1.83$$

$$\sigma_{q1}^2 = 2.90\sigma_e^2$$

$$\sigma_{q2}^2 = 3.16\sigma_e^2$$

- and

$$\frac{\sigma_{q2}^2}{\sigma_{q1}^2} = 1.09$$

- So, the noise power in the second realization is 9% more than the first realization.