FINITE WORD LENGTH EFFECTS

Fixed point and floating point number representations – ADC –Quantization- Truncation and Rounding errors - Quantization noise – coefficient quantization error – Product quantization error - Overflow error – Round off noise power - limit cycle oscillations due to product round off and overflow errors – Principle of scaling

5.1.INTRODUCTION:

Digital signal processing algorithms are realized either with special purpose digital hardware or as programs for a general purpose digital computer. In both cases the number and coefficients are stores in finite length registers. Therefore coefficients and numbers are quantizes by truncation or rounding when they are stored.

The following errors arise due to quantization of numbers

1. Input quantization error
2. Product quantization error
3. Coefficient quantization error

5.2 NUMBER REPRESENTATION:

When working with any kind of digital electronics in which numbers are being represented, it is important to understand the different ways numbers are represented in these systems. Almost without fail, numbers are represented by two voltage levels which can represent a one or a zero. The number system based on ones and zeroes is called the binary system (because there are only two possible digits). Before discussing the binary system, a review of the decimal (ten possible digits) system is in order, because many of the concepts of the binary system will be easier to understand when introduced alongside their decimal counterpart.

The subscript 10 denotes the number as a base 10 (decimal) number.

$125_{10} = 1*100 + 2*10 + 5*1 = 1*10^2 + 2*10^1 + 5*10^0$

$25.43_{10} = 2*10 + 5*1 + 4*0.1 + 3*0.01 = 2*10^1 + 5*10^0 + 4*10^{-1} + 3*10^{-2}$

The only pertinent observations here are:

- If there are m digits to the right of the decimal point, the smallest number that can be represented is 10^{-m}. For instance if m=4, the smallest number that can be represented is $0.0001=10^{-4}$.

A number N is represented by finite series as

$$N = \sum_{i=n_1}^{n_2} c_i r^i$$

r = 10 for decimal representation

$$30.285 = \sum_{i=-3}^{1} c_i 10^i$$
$$= 3 \times 10^1 + 0 \times 10^0 + 2 \times 10^{-1} + 8 \times 10^{-2} + 5 \times 10^{-3}$$

r = 2 for binary representation

$$110.010 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}$$
$$= (6.25)_{10}$$

1. Convert the decimal number 30.275 to binary form.

Integer part					Remainder	Fractional part					Integer part	Binary number
30	/	2	=	15	0	0.275	x	2	=	0.550	0	
15	/	2	=	7	1	0.55	x	2	=	1.10	1	
7	/	2	=	3	1	0.1	x	2	=	0.2	0	
3	/	2	=	1	1	0.2	x	2	=	0.4	0	
1	/	2	=	0	1	0.4	x	2	=	0.8	0	
					Binary number	0.8	x	2	=	1.6	1	
						0.6	x	2	=	1.2	1	
						0.2	x	2	=	0.4	0	

$(30.275)_{10} = (11110.01000110)_{10}$

5.2.1 Types of number representation:

There are three common forms that are used to represent the numbers in a digital computer or any digital hardware.

1. Fixed point representation
2. Floating point representation
3. Block floating point representation

5.2.1.1 Fixed point representation:

Fixed-point numbers are useful for representing fractional values, usually in base 2 or base 10, when the executing processor has no floating point unit (FPU) or if fixed-point provides improved performance or accuracy for the application at hand. Most low-cost embedded microprocessors and microcontrollers do not have an FPU.

A value of a fixed-point data type is essentially an integer that is scaled by a specific factor determined by the type. For example, the value 1.23 can be represented as 1230 in a fixed-point data type with scaling factor of 1/1000, and the value 1230000 can be represented as 1230 with a scaling factor of 1000. Unlike floating-point data types, the scaling factor is the same for all values of the same type, and does not change during the entire computation.

The shifting process above is the key to understand fixed point number representation. To represent a real number in computers (or any hardware in general), we can define a fixed point number type simply by implicitly fixing the binary point to be at some position of a numeral. We will then simply adhere to this implicit convention when we represent numbers. For example

$$00010.110_2$$

$$= 1 * 2^1 + 1 * 2^{-1} + 1 * 2^{-2}$$
$$= 2 + 0.5 + 0.25$$
$$= 2.75$$

$$000.10110_2$$

$$= 1 * 2^{-1} + 1 * 2^{-3} + 1 * 2^{-4}$$
$$= 0.5 + 0.125 + 0.0625$$
$$= 0.6875$$

There are three common forms that are used to represent the Fixed point numbers

1.Sign magnitude form
2.One's complement
3.Two's complement

Sign magnitude form:

It was noted previously that we will not be using a minus sign (-) to represent negative numbers. We would like to represent our binary numbers with only two symbols, 0 and 1. There are a few ways to represent negative binary numbers. The simplest of these methods is called ones complement, where the sign of a binary number is changed by simply toggling each bit (0's become 1's and vice-versa). This has some difficulties, among them the fact that zero can be represented in two different ways (for an eight bit number these would be 0000 0000 and 1111 1111)., we will use a method called two's complement notation which avoids the pitfalls of one's complement, but which is a bit more complicated.

- MSB is set to 1 to represent the negative sign
- Zero has two representations
- With 'b' bits only 2b-1 numbers can be represented

Eg. $(1.75)_{10}$ = $(01.110000)_2$
$(-1.75)_{10}$ = $(11.110000)_2$

One's complement:

The ones' complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number (swapping 0's for 1's and vice-versa). The ones' complement of the number then behaves like the negative of the original number in most arithmetic operations. However, unlike two's complement, these numbers have not seen widespread use because of issues like negative zero, end-around borrow, etc.

A ones' complement system or ones' complement arithmetic is a system in which negative numbers are represented by the arithmetic negative of the value. In such a system, a number is negated (converted from positive to negative or vice versa) by computing its ones' complement. An N-bit ones' complement numeral system can only represent integers in the range $-(2^{N-1}-1)$ to $2^{N-1}-1$
For example:

$0.875_{10}=0.111000_2$

$-0.875_{10}=1.000111_2$

Positive numbers are represented as in sign magnitude form. Negative number is represented by complementing all the bits of the positive number

Eg. $(0.875)_{10}$ = $(0.111000)_2$
$(-0.875)_{10}$ = $(1.000111)_2$ – one's complement form

The same is obtained by subtracting the magnitude from $2-2^{-b}$, b is the number of bits (without sign bit).In the above example b=6 therefore $2-2^{-6}$ = 10.000000 – 0.000001 = 1.111111

Now for $(-0.875)_{10}$ 1.111111
-0.111000
=1.000111 (one's complement)

Also the magnitude for the negative number is given by

$$1-\sum_{i=1}^{b} c_i 2^{-i} - 2^{b}$$

For $(-0.875)_{10}$ $= (1.000111)_2$

$= 1 - (0 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6}) - 2^{-6}$

$= 1 - (2^{-4} + 2^{-5} + 2^{-6}) - 2^{-6}$

$= (0.875)_{10}$

Two's complement:

Two's complement is a mathematical operation on binary numbers, as well as a representation of signed binary numbers based on this operation. The two's complement of an N-bit number is defined as the complement with respect to 2^N, in other words the result of subtracting the number from 2^N. This is also equivalent to taking the ones' complement and then adding one, since the sum of a number and its ones' complement is all 1 bits. The two's complement of a number behaves like the negative of the original number in most arithmetic, and positive and negative numbers can coexist in a natural way.

In two's-complement representation, negative numbers are represented by the two's complement of their absolute value in general, negation (reversing the sign) is performed by taking the two's complement. This system is the most common method of representing signed integers on computers.[2] An N-bit two's-complement numeral system can represent every integer in the range $-(2^{N-1})$ to $+(2^{N-1} - 1)$ while ones' complement can only represent integers in the range $-(2^{N-1} - 1)$ to $+(2^{N-1} - 1)$.

To calculate the 2's complement of an integer, invert the binary equivalent of the number by changing all of the ones to zeroes and all of the zeroes to ones (also called 1's complement), and then add one.

For example:

$0.875_{10}=0.111000_2$

$\underline{\qquad\qquad 1.000111_2}$

$-0.875_{10}=1.001000_2$

Positive numbers are represented as in sign magnitude form. Negative number is represented in two's complement form of the positive number

Eg. $(0.875)_{10} = (0.111000)_2$

$(-0.875)_{10} = (1.000111)_2$ – one's complement form

$= +0.000001 \quad +1$

$= (1.001000)_2$ – two's complement form

The same is obtained by subtracting the magnitude from 2

Now for $(-0.875)_{10}$ 10.000000 (2)
-0.111000 (+0.875)
= 1.001000 (-0.875 in two's complement form)

The magnitude for the negative number is given by

$$1-\sum_{i=1}^{b} c_i 2^{-i}$$

For $(-0.875)_{10}$ $= (1.001000)_2$
$= 1-(0 \times 2^{-1}+0 \times 2^{-2}+1 \times 2^{-3}+0 \times 2^{-4}+0 \times 2^{-5}+0 \times 2^{-6})$
$= 1-2^{-3}$
$= (0.875)_{10}$

Addition of two fixed point numbers:

The two numbers are added bit by b it starting from right, with carry being added to the next bit.

Eg., $(0.5)_{10}+(0.125)_{10}$

$(0.5)_{10}$	=	0.100		
$+(0.125)_{10}$	=	+0.001		
	=	0.101	=	$(0.625)_{10}$

When two number of 'b' bits are added and the sum cannot be represented by 'b' bits an overflow is said to occur. In general, the addition of fixed point numbers causes an overflow

Eg., $(0.5)_{10}+(0.625)_{10}$

$(0.5)_{10}$	=	0.100
$+(0.625)_{10}$	=	+0.101
=1.125	=	1.001 But which is $(-0.125)_{10}$ in sign magnitude

Subtraction of two fixed point numbers:

$(0.5)_{10}-(0.25)_{10}$

$(0.5)_{10}$	=			0.100
$-(0.25)_{10}$	=	0.010	= 1.101 (1'S) + 0.001	= +1.110 (2'S)
				= 10.010
				= 0.010 (neglect carry) = $(0.25)_{10}$

$(0.25)_{10} - (0.5)_{10}$

$(0.25)_{10}$ = 0.010

$-(0.5)_{10}$ = 0.100 = 1.011 (1'S) + 0.001 = +1.100 (2'S)

= 1.110 (No carry, result is negative take 2's complement)

= 0.001 (1's complement)

= +0.001 (+1)

= $0.010\ (-0.25)_{10}$

Multiplication in fixed point:

Sign and magnitude components are separated. The magnitude of the numbers are multiplied then sign of the product is determined and applied to result With 'b' bit multiplicand and 'b' bit multiplier the product may contain 2b bits. If b = bi + bf, where bi represents integer part and bf represents the fraction part then the product may contain 2bi + 2bf bits. Multiplication of the two fraction results in a fraction and overflow can never occur.

5.2.1.2 Floating point representation:

In floating point representation a positive number is represented as $F=2^c m$.Where M called mantissa is a fraction such that $0.5 \leq M \leq 1$ and c the exponential can be either positive or negative. For negative numbers the sign of the floating point number is obtained from the first bit of mantissa. The decimal numbers 4.5,1.5,6.5 have following representation

$$4.5=2^3*0.5625=2^{011}*0.1001$$
$$1.5=2^1*0.75=2^{001}*0.1100$$
$$6.5=2^3*0.8125=2^{011}*0.1100$$

Multiplication:

If $F_1 = 2^{c1} \cdot M_1$ and $F_2 = 2^{c2} \cdot M_2$ then $F_3 = F_1 \times F_2 = (M_1 \times M_2) \cdot 2^{(c1+c2)}$

Eg., $(1.5)_{10} = 2^{001} \times 0.1100$

$(1.25)_{10} = 2^{001} \times 0.1010$

$(1.5)_{10} \times (1.25)_{10} = 2^{001} \times 0.1100 \times 2^{001} \times 0.1010$

$= 2^{(001+001)} \times (0.1100 \times 0.1010)$

$= 2^{010} \times 0.01111$

Addition and subtraction of two floating point numbers are more difficult than addition and subtraction of two fixed point numbers. To carry out addition, first adjust the exponent of the smaller number until it matches with the exponent of the larger number. The mantissa are then added or subtracted

Eg., $(3)_{10} + (0.125)_{10}$

$(3)_{10} = 2^{010} \times 0.110000$

$(0.125)_{10} = 2^{000} \times 0.001000 = 2^{010} \times 0.000010$ (adjust the exponent of smaller number)

$(3)_{10} + (0.125)_{10} = 2^{010} (0.110000 + 0.000010)$

$= 2^{010} \times 0.110010$

Comparison:

Fixed pont number	Floating point number
1.Fast operation 2.Small dynamic range 3.Relatively economical 4.Overflow occur in addition 5. Roundoff errors occur only for addition 6. Used in small computers	Slow operation Increased dynamic range More expensive Round off error can occur Roundoff errors can occur with both addition and multiplication Used in larger, general purpose computers

5.2.1.3 Block floating point representation:

A compromise between fixed and floating point systems is the block floating point arithmetic. The set of signals to be handled is divided into blocks .Each block have the same value of exponent. The arithmetic operations within the block uses fixed point arithmetic and only one exponent per block is stored, thus saving memory.Suitable for FFT flow graphs and digital audio applications.

5.3 QUANTIZATION NOISE:

For most of the engineering applications the input signal is continuous in time or analog waveform .This signal is to be converted into digital using ADC. The process of converting analog to digital is shown below.

.

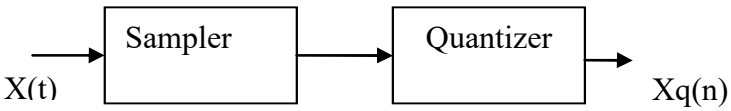


Fig 5.1:The process of converting analog to digital

At first x(t) is sampled at regular intervals at t=nT where N=0,1,2… to create a sequence x(n).This is done by a sample. The difference signal e(n)=xq(n)-x(n).Let as assume a sinusoidal signal varying between +1 and -1 having a dynamic range 2.If ADC is used to convert the sinusoidal signal employs (b+1) bits including sign bit, the number of levels available for quantizing x(n) is 2^{b+1}.Thus the interval between successive level is quantization step size..Where q is known as quantization step size.

$$q = \frac{2}{2^{b+1}} = 2^{-b}$$

The common methods of quantization are,

1. Truncation
2. 2.Rounding

Truncation:

Truncation is a process of discarding all bits less significant than least significant bit that is retained.

e.g. 0.00110011 = 0.0011 (8 bits to 4 bits)
1.01001001 = 1.0100

Rounding:

Rounding of a number of b bits is accomplished by choosing the rounded result as the b-bit number closest to the original number unrounded. For example 0.11010 rounded to three bits is either 0.110 or 0.111

e.g. 0.11010 = 0.110 or 0.111
0.110111111 = 0.11011111 or 0.11100000

Rounding up or down will have negligible effect on accuracy of computation

5.3.1 Error due to truncation and rounding

If the quantization method is truncation ,the number is approximated by the nearest level that does not exceed it. In this case, the error xT-x is negative or zero where xT is truncation value of x.

The error made by truncating a number to *b* bits following the binary point satisfies the inequality The given equation (1) indicates binary point satisfies the in equality.

$$0 \geq x_T - x \geq -2^{-b} \ldots\ldots\ldots (1)$$

e.g. $(0.12890625)_{10} = (0.00100001)_2$

Truncate to 4 bits $x_T = (0.0010)_2 = (0.125)_{10}$

Now the error $(x_T - x) = -0.00390625$ which is $> -2^{-b} = -2^{-4} = -0.0625$ satisfy the inequality

The equation (1) holds for both sign magnitude,1's complement and 2's complement. if x>0.Consider first the 2's complement representation. The magnitude of negative number is,

$$x = 1 - \sum_{i=1}^{b} c_i 2^{-i}$$

If we truncate to N bits then

$$x_T = 1 - \sum_{I=1}^{N} c_i 2^{-i}$$

The change in magnitude

$$x_T - x = \sum_{i=1}^{b} c_i 2^{-i}$$

$$\geq 0 \ldots\ldots\ldots (2)$$

From equation (2) we can find the truncation increases the magnitude which implies that error is negative and satisfy the equality.

$$0 \geq x_T - x \geq -2\text{-}b$$

For 1's complement representation. The magnitude of negative number is,

$$x = 1 - \sum_{i=1}^{b} c_i 2^{-i} - 2^{-b}$$

When the number is truncated to N bits, then

$$x_T = 1 - \sum_{i=1}^{N} c_i 2^{-i} - 2^{-b}$$

The change in magnitude due to truncation is

$$X_T - X = \sum_{i=1}^{b} c_i 2^{-i} - (2^{-N} - 2^{-b}) \ldots\ldots\ldots\ldots 3$$

$$< 0$$

From equation (3) we can find the truncation decreases the magnitude which implies that error is positive and satisfy the equality.

$$0 \geq x_T - x \geq -2^{-b} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

The above equation holds for sign magnitude representation also, In floating point systems the effect of truncation is visible only in the mantissa.In floating point number the effect of truncation is visible only in the mantissa.

$$\text{If } x = 2^c.M$$
$$x_T = 2^c.M_T$$
$$e = x_T - x = (M_T - M)2^c$$

From equation (4) with 2's complement representation of mantissa we have

$$0 \geq M_T - M \geq -2^{-b}$$
$$0 \geq e \geq -2^{-b}2^c \ldots\ldots\ldots\ldots\ldots (5)$$

We define relative error $\varepsilon = e/x$

$$0 \geq \varepsilon x \geq -2^{-b}2^c$$
$$0 \geq \varepsilon 2^c M \geq -2^c 2^{-b}$$
$$\text{Now } 0 \geq \varepsilon M \geq -2^{-b}$$

If M = 1/2 the relative error is maximum. Therefore, $0 \geq \varepsilon > -2^{-b}$

If M = -1/2 the relative error range is $0 \leq \varepsilon < 2^{-b}$

In one's complement representation, the error for truncation of the values of the mantissa is

$$0 \geq M_T - X > -2^{-b}$$
$$0 \geq e > -2^{-b}2^c$$

With $e = \varepsilon x = \varepsilon 2^c M$ and M=1/2 we get the maximum range of the relative error for positive mantissa is

$$0 \geq \varepsilon > -2 \cdot 2^{-b}$$

For negative mantissa, value of error is

$$0 \leq M_T - X < 2^{-b}$$
$$\text{Or} \quad 0 \leq e < 2^c 2^{-b}$$

With M=-1/2 the maximum range of the relative error negative mantissa is

$0 \geq \varepsilon > -2^{-b}$,which is same as positive mantissa.

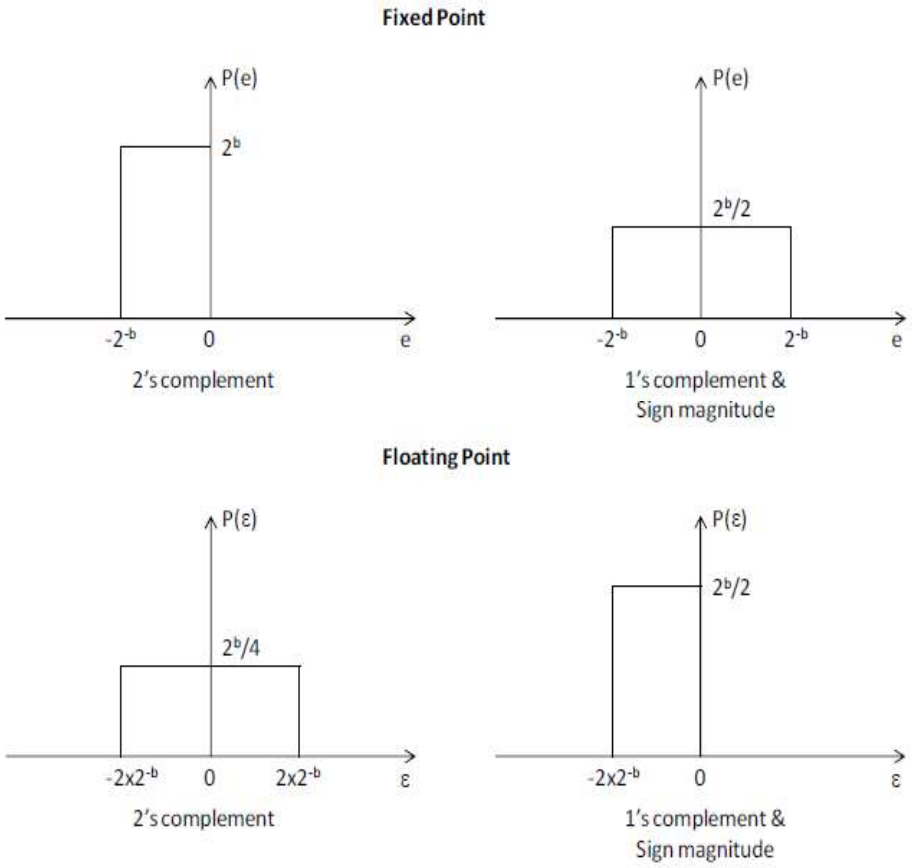


Fig 5.2:The probability density function for p(e) for fixed and floating point

The probability density function for p(e) for truncation of fixed point and floating point number are,

In fixed point arithmetic the error due to rounding a number to b bit produces an error e=xT-x which satisfies the in equality.

$$-2^{-b}/2 \leq x_T - x \leq 2^{-b}/2$$

This is because with rounding if the value lies half way between two level or by the nearest lower level. For fixed point number satisfies regardless of whether sign-magnitude,1's complement is used for negative number. In floating point arithmetic, only mantissa is affected by quantization, If

$$x = 2^c.M$$
$$x_T = 2^c.M_T$$
$$e = x_T - x = (M_T - M)2^c$$

For rounding $-2^{-b}/2 \leq M_T - M \leq 2^{-b}/2$

$$-2^{-b}/2 \le e \le 2^{-b}/2$$
$$-2^{c}\, 2^{-b}/2 \le \varepsilon x \le 2^{-b}/22^{c}$$
$$-2^{c}\, 2^{-b}/2 \le \varepsilon 2^{c}\, M \le 2^{-b}/22^{c}$$

Which gives,

$$-2^{-b}/2 \le \varepsilon\, M \le 2^{-b}/2$$
$$1/2 \le M < 1$$

If M=1/2 we get the maximum range of relative error

$$-2^{-b} \le \varepsilon < 2^{-b}$$

The probability density function for rounding is as follows

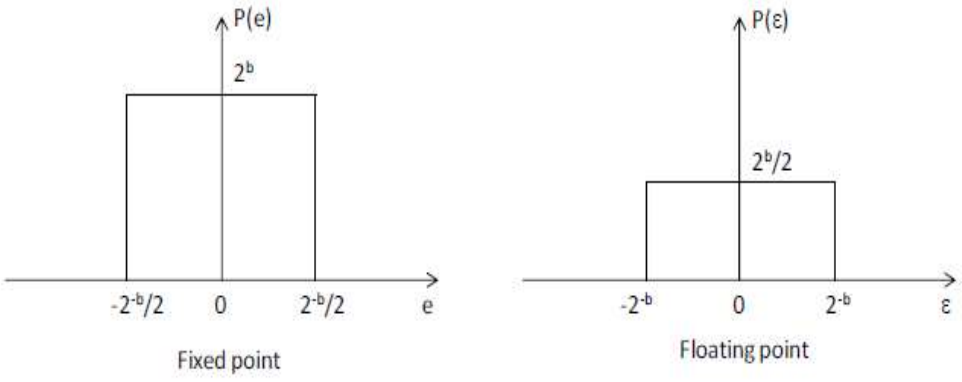


Fig 5.3:In rounding p(e) for fixed and floating point number

5.4 Input quantization error:

The quantization error arises when a continuous signal is converted into digital value. The quantization error is given by e(n)=xq(n)-x(n). where xq(n) = sampled quantized value x(n) = sampled unquantized value Depending on the way in which x(n) is quantized different quantization distributions of quantization noise may be obtained. If rounding of a number is used to get xq(n) then the error signal satisfies the relation.

$$-2^{-b}/2 \le X_T - x \le 2^{-b}/2$$
$$-q/2 \le e(n) \le q/2$$

Because the quantized signal may be greater or less than actual signal.

Eg., let x(n) = (0.7)10 = (0.10110011…)2

After rounding x(n) to 3 bits

$$xq(n) = (0.110)2 = (0.75)10$$

Now the error e(n) = 0.75 – 0.7 = 0.05 which satisfies the inequality

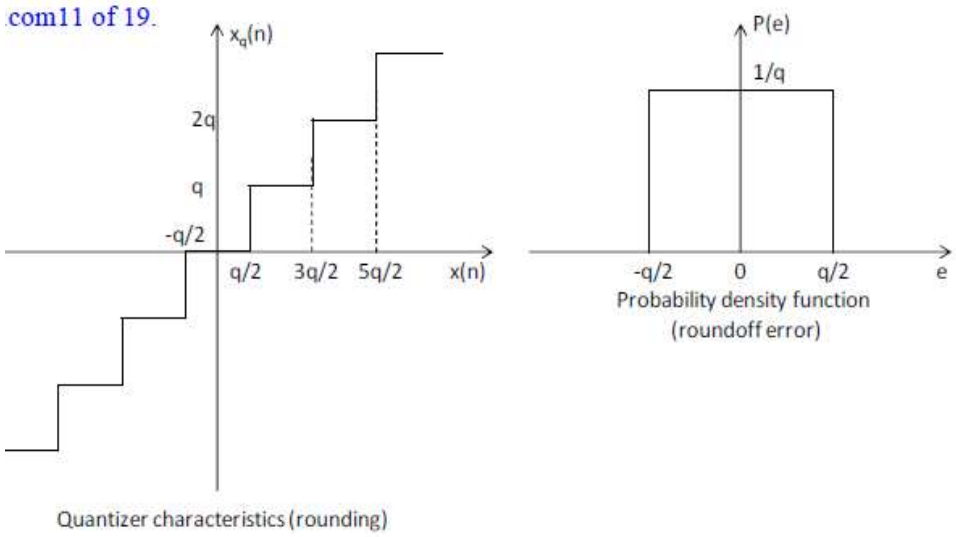


Fig 5.4: quantization characteristic for rounding

The probability density function p(e) for roundoff error and quantization characteristics with rounding is given below

$$0 \geq xT\text{-}x \geq -2\text{-}b$$

The quantizer characterized for truncation and probability density function p(e) for 2's complement is given below.

The other type of quantization can be obtained by truncation. In truncation, the signal is represented by the highest quantization level that is not greater than the signal.

In two's complement truncation, the error e(n) is always negative and satisfied the Inequality

$$-q \leq e(n) < 0$$

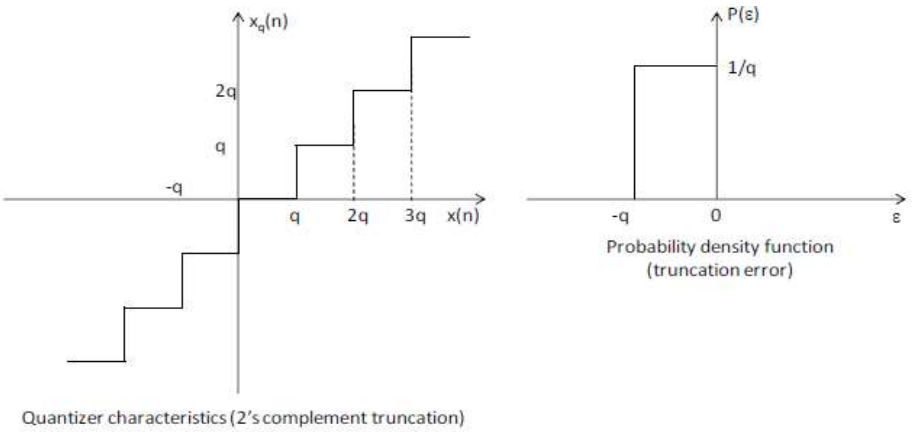


Fig 5.5: quantization characteristic for truncation

The quantization error mean value is zero for rounding and –q/2 for 2's complement truncation.

5.4.1 Steady state input noise power:

In digital processing of analog signals the quantization error is commonly viewed as an additive noise signal that is ,

$$xq(n)=x(n)+e(n)$$

In analog-to-digital conversion, the difference between the actual analog value and quantized digital value is called quantization error or quantization distortion. This error is either due to rounding or truncation. The error signal is sometimes considered as an additional random signal called quantization noise because of its stochastic behavior. The quantizer noise model is given below.

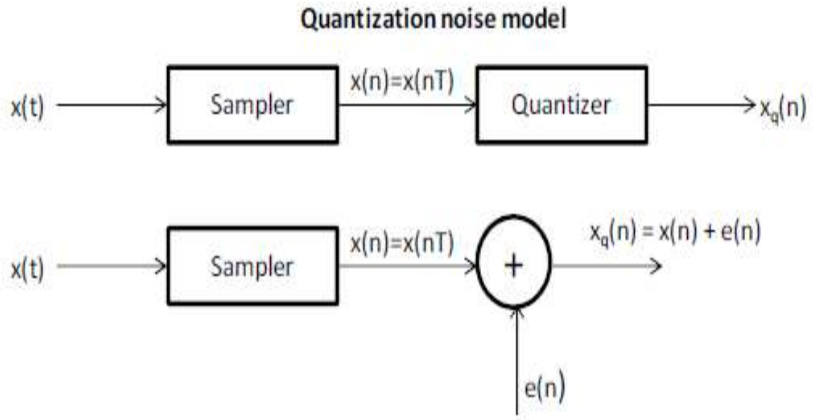


Fig 5.6: The quantizer noise model

Quantization noise is a model of quantization error introduced by quantization in the analog-to-digital conversion (ADC) in telecommunication systems and signal processing. It is a rounding error between the analog input voltage to the ADC and the output digitized value. The noise is non-linear and signal-dependent. It can be modeled in several different ways.

Therefore, the A/D converter output is the sum of the input signal x(n) and the error signal e(n).If the rounding is used for quantization then the quantization error e(n)=xq(n) – x(n) is bounded by –q/2 ≤e(n)≤q/2. In most cases, assume that A/D conversion error e(n) has the following properties

- The error sequence e(n) is a sample sequence of a stationary random process
- The error sequence is uncorrelated with x(n) and other signals in the system
- The error is a white noise process with uniform amplitude probability distribution over the range of quantization error.

In case of rounding the e(n) lies between –q/2 and q/2 with equal probability. The variance of e(n) is given by,

$$\sigma e2=E[e^2(n)]-E^2[e(n)$$

where E[e2(n)]is the average of e2(n) and E[e(n) is mean value of e(n).

$$\sigma_e^2 = \int_{-\infty}^{\infty} e^2(n)\, p(e)\, de - (0)^2$$

$$\sigma_e^2 = \int_{-\frac{q}{2}}^{\frac{q}{2}} e^2(n) \cdot \frac{1}{q} de$$

$$\sigma_e^2 = \frac{1}{q} \int_{-\frac{q}{2}}^{\frac{q}{2}} e^2(n) \cdot de$$

$$\sigma_e^2 = \frac{1}{q}\left[\frac{e^3(n)}{3}\right]_{-\frac{q}{2}}^{\frac{q}{2}}$$

$$\sigma_e^2 = \frac{1}{q}\left[\frac{q^3}{24} + \frac{q^3}{24}\right]$$

$$sub\, q = 2^{-b}$$

$$\sigma_e^2 = \frac{\left(2^{-b}\right)^2}{12}$$

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

$\sigma e^2 = 2^{-2b}/12$

In case of two's complement truncation the e(n) lies between 0 and –q having mean value of –q/2. The variance or power of the error signal e(n) is given by

$$\sigma_e^2 = \int_{-q}^{0} e^2(n)\, p(e)\, de - (-\frac{q}{2})^2$$

$$\sigma_e^2 = \int_{-q}^{0} e^2(n) \cdot \frac{1}{q}\, de - \left(-\frac{q}{2}\right)^2$$

$$\sigma_e^2 = \frac{1}{q}\left[\frac{e^3(n)}{3}\right]_{-q}^{0} - \frac{q^2}{4}$$

$$\sigma_e^2 = \left(\frac{1}{q}\right)\frac{q^3}{3} - \frac{q^2}{4}$$

$$\sigma_e^2 = \frac{4q^2 - 3q^2}{12} = \frac{q^2}{12}$$

$sub\, q = 2^{-b}$

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

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

In both cases the value of $\sigma e^2=2^{-2b}/12$, which is also known as the steady state noise power due to input quantization.

If the input signal is x(n) and its variance is σ_x^2 then the ratio of signal power to noise power which is known as signal to noise ratio for rounding is

$$\frac{\sigma_x^2}{\sigma_e^2} = \frac{\sigma_x^2}{2^{-2b}/12} = 12(2^{2b}\sigma_x^2)$$

$$= 10\log_{10}\frac{\sigma_x^2}{\sigma_e^2}$$

$$= 10\log_{10} 12(2^{2b}\sigma_x^2)$$

$$= 10\log_{10} 12 + 10\log_{10} 2^{2b} + 10\log_{10}\sigma_x^2)$$

$$= 10.73 + 10x\,2b\,x\log_{10} 2 + 10\log_{10}\sigma_x^2)$$

$$= 10.79 + 6.02b + 10\log_{10}\sigma_x^2$$

BY using 1's and 2's complement representation also we will get the same value of output noise power. In an ideal analog-to-digital converter, where the quantization error is uniformly distributed between −1/2 LSB and +1/2 LSB, and the signal has a uniform distribution covering all quantization levels, the Signal-to-quantization-noise ratio (SQNR) can be calculated from

$$SNR=2b+10.79+10\log10\ ^{\sigma x2}$$

Here, the quantization noise is once again assumed to be uniformly distributed. When the input signal has a high amplitude and a wide frequency spectrum this is the case. n this case a 16-bit ADC has a maximum signal-to-noise ratio of 98.09 dB. The 1.761 difference in signal-to-noise only occurs due to the signal being a full-scale sine wave instead of a triangle/sawtooth. Quantizer characteristics with rounding is given below.

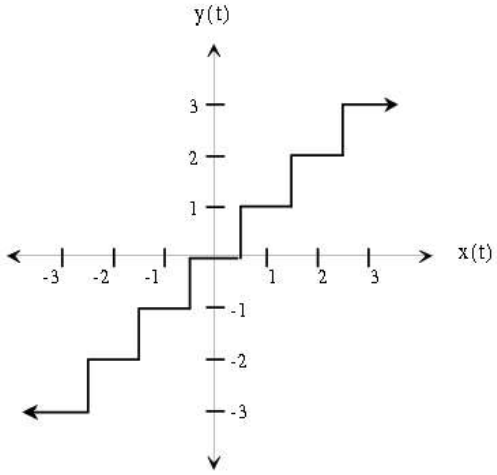


Fig 5.7: Quantizer characteristics with rounding

5.4.2 Steady state output noise power:

Due to ADC noise one can represent the quantized input to a digital system with impulse response h(n).Let ε(n) be the output noise due to quantization of the input. Then we get,

$$\varepsilon(n) = e(n)*h(n)$$

The variance of any term in the above sum is equal to $\sigma e^2\ h^2(n)$.The variance of independent random variable is the sum of their variance. If the quantization errors are assumed to be independent at different sampling instances then the variance of the output

$$\sigma^2\varepsilon(n) = \sigma e^2 \sum_{n=0}^{k} h^2(n)$$

To find the steady state variance extend the limit k up to infinity. Then we have

$$\sigma^2 \varepsilon(n) = \sigma e^2 \sum_{n=0}^{\alpha} h^2(n)$$

Using parseval's theorem the steady state output noise variance due to the quantization error is given by $\sigma^2 \varepsilon(n) = \frac{\sigma e^2}{2\pi j} \oint H(Z)H(Z^{-1})Z^{-1}dZ$

Where the closed contour of integration is around the unit circle |z| in which case only one poles that lie inside unit circle are evaluated using the residue theorem.

5.5 Product quantization error:

In the fixed point arithmetic the product of two 'b' bit number results in 2b bit long. In digital signal processing applications. It is necessary to round this product to a b bit number which produce an error known as product quantization error or product round off noise.

The model for fixed point round off noise following a multiplier is shown in figure shown below. Here the multiplier is modeled as an infinite precision multiplier followed by an adder. Where round off noise model is added to the product. So that overall result some quantization level.

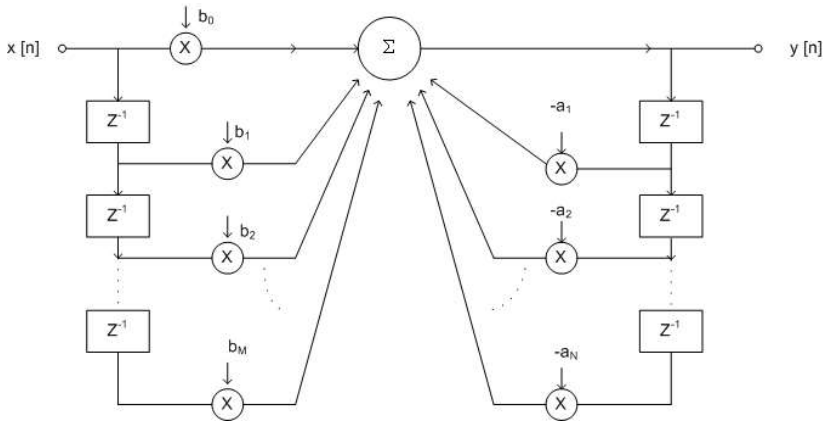


Fig 5.8: Product quantization error:

From the above figure all the noise sources are added at the same points in the filter hence the noise source e(n)=e0(n)+e1(n)+e2(n)……with zero mean and variance $\sigma^2 = \sigma^2 0 + \sigma^2 1 + \sigma^2 2$….If hk(n) is the filter's impulse response from the noise source to the filter output the response due to noise source ek(n) can be obtained by convolution as εk(n)=∑hk(m)ek(n-m)

variance, $\sigma \varepsilon^2 = \sigma e^2 \sum_{n=0}^{k} h^2(n)$

It can be written as $\sigma^2 \varepsilon(n) = \frac{\sigma e^2}{2\pi j} \oint H_k(Z) H_k(Z^{-1}) Z^{-1} dZ$

Hk(z) is defined as the noise transfer function.

5.6 Coefficient quantization error:

Ordinary in the FIR filter design the filter coefficients are evaluated with infinite precision. FIR filters are often implemented using finite word length multipliers and adders, and therefore the high precision coefficients need to be rounded off. This rounding process introduces errors to the filter and the quantized filter may not meet the filter requirements any more. Normally the rounding is performed using a conventional "round to nearest" approach resulting in a frequency domain error that is spread out over the entire frequency range.

If certain frequency bands are more critical than others, there is a way to control the effects of quantization in the frequency domain.

A technique known as "noise shaping" is applied to data on a regular basis in A/D and D/A converters. Running at very high sampling frequencies (over sampling) D/A converters need only one bit to represent the low frequency signal components with very high accuracy. Analog low-pass filters outside the D/A converter takes care of the unwanted noise at higher frequencies.

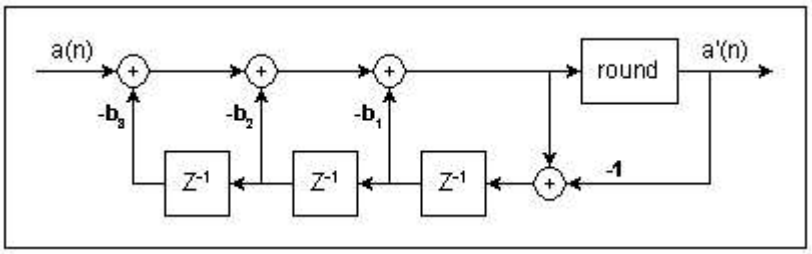


Fig 5.9: Noise shaping the filter coefficients

The above figure shows how high precision filter coefficients, a(n), are converted to fixed word length coefficients, a'(n), using a simple error feedback mechanism. The "round" function performs a simple "round to nearest" operation. The multipliers (b1,b2,b3) control the noise shaping frequency characteristic.If the poles of the desired filter are closed to the unit circle then those of the filter with quantized coefficients may lie just outside the unit circle.

5.7 LIMIT CYCLE OSCILLATIONS

5.7.1 Zero input limit cycle oscillations:

When a stable IIR filter is excited by a finite input sequence, that is constant, the output will ideally decay to zero. However, the non-linearity due to the finite-precision

arithmetic operations often cause periodic oscillations in the recursive systems are called zero input limit cycle oscillations. Consider a first order IIR filter with difference equation

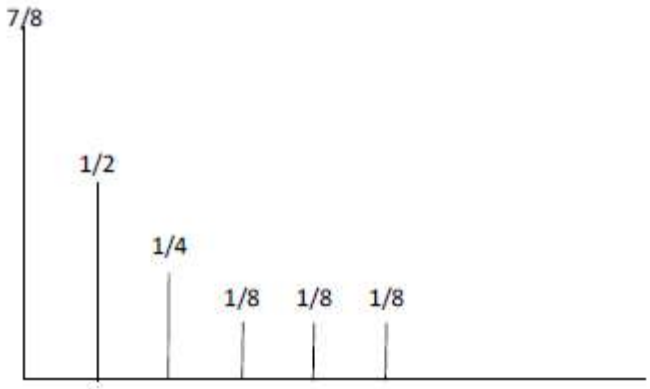
$$y(n) = x(n) + \alpha y(n-1)$$

Let's assume α = ½ and the data register length is 3 bits plus a sign bit. If the input is,

$$x(n) = \begin{cases} 0.875 & for\ n = 0, \\ 0 & otherwise. \end{cases}$$

and rounding is applied after the arithmetic operation. Here Q[·] represents the rounding operations.

n	$x(n)$	$y(n-1)$	$\alpha y(n-1)$	$Q[\alpha y(n-1)]$	$y(n) = x(n) + Q[\alpha y(n-1)]$
0	0.875	0	0	0.000	7/8
1	0	7/8	7/16	0.100	1/2
2	0	1/2	1/4	0.010	1/4
3	0	1/4	1/8	0.001	1/8
4	0	1/8	1/16	0.001	1/8
5	0	1/8	1/16	0.001	1/8



From the above table it is found that for n ≥3 the output remains constant and gives 1/8 as steady output causing limit cycle behavior. Round a value in the above table:

7/16 = 0.4375

0.4375 x 2 = 0.875
0.875 x 2 = 1.75

Let's assume α = -1/2

n	x(n)	y(n-1)	α y(n-1)	Q[α y(n-1)]	$y(n)=x(n)+Q[\alpha y(n-1)]$
0	0.875	0	0	0.000	7/8
1	0	7/8	-7/16	1.100	-1/2
2	0	-1/2	1/4	0.010	1/4
3	0	1/4	-1/8	1.001	-1/8
4	0	-1/8	1/16	0.001	1/8
5	0	1/8	-1/16	1.001	-1/8
6	0	-1/8	1/16	0.001	1/8

When α = -1/2 the output oscillates between 0.125 to -0.125

5.7.2 Dead band

The limit cycles occur as a result of the quantization effects in multiplications. The amplitudes of the output during a limit cycle are confined to a range of values that is called the dead band of the filter. Let us consider a single pole IIR system whose difference equation is given By,

$$y(n)= \alpha y(n-1)+x(n),n>0$$

After rounding the product term we have $yq(n)=Q[\alpha y(n-1)]+x(n)$

During the limit cycle oscillations $Q[\alpha y(n-1)]=y(n-1)$ for $\alpha>0$

$=-y(n-1)$ for $\alpha<0$

By the definition of rounding we have $|Q[\alpha y(n-1)]- \alpha\, y(n-1)| \leq 2^{-b}/2$

$$y(n-1)\,[1-|\alpha|] \leq 2^{-b}/2$$

$$y(n-1) \leq 0.52^{-b}/1-|\alpha|$$

The above equation defines the dead band for the first order filter. Dead band range $l \leq 0.5/1-|\alpha|$

5.7.3 Overflow limit cycle oscillations:

In addition to limit cycle oscillations caused by rounding the result of multiplications, there are several types of oscillations caused by addition, which makes the filter output oscillates between maximum and minimum amplitudes such limit cycles have referred to as overflow oscillations. An overflow in addition of two or more binary numbers occurs when the sum exceeds the word size available in the digital implementation of the system. Let's consider two positive number n1 and n2

$n_1 = (7/8)_{10} = (0.111)_2$
$n_2 = (6/8)_{10} = (0.110)_2$
$n_1 + n_2 = (1.101)_2$

(-5/8 in sign magnitude ,but actual total is 13/8)

In the above example, when two positive numbers are added the sum is wrongly interrupted as a negative number.

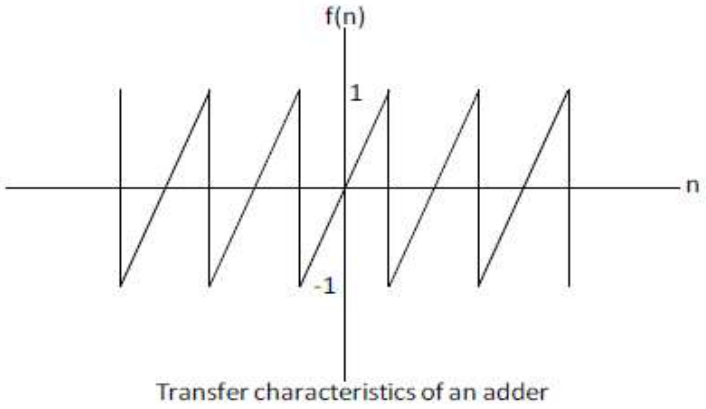


Fig 5.10:Transfer characteristics of an adder

The overflow occurs if the total input is out of range, this problem can be eliminated by modifying adder characteristics. When an overflow is detected, the sum of adder is set equal to the maximum value

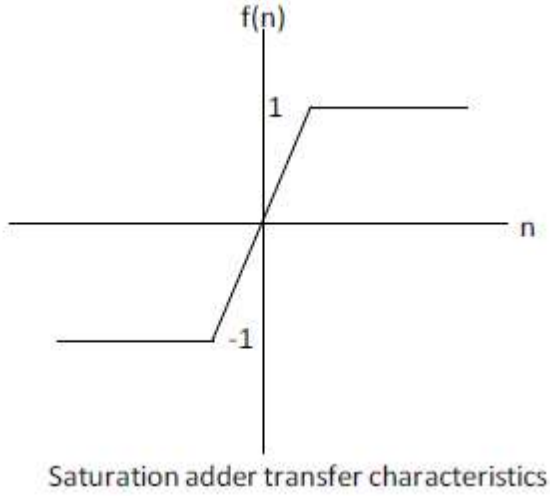


Fig 5.11:Saturation adder transfer characteristics

Saturation arithmetic eliminates limit cycle due to overflow ,but it causes undesirable signal distortion due to non-linearity. In order to limit the amount of non-linear distortion, It is important to scale the input signal and unit sample response between input and any internal summing node in the system such that overflow becomes a rare event. Let us consider realization of a second order IIR filter.

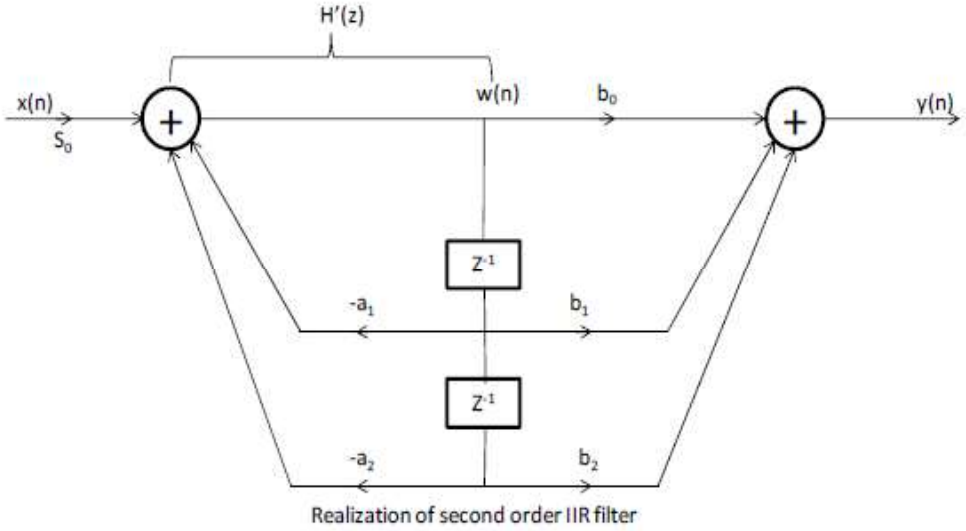


Fig 5.12:Realization of a second order IIR filter

In the above figure a scale factor s0 is introduced between the input and the adder 1 to prevent overflow at the output adder 1.Now the overall input-output transfer function is

$$H(z) = S_0 \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$= S_0 \frac{N(z)}{D(z)}$$

$$H'(z) = \frac{W(z)}{X(z)} = \frac{S_0}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{S_0}{D(z)}$$

If the instantaneous energy in the output sequence w(n) is less than the finite energy in the input sequence then, there will not be any overflow.

$$W(z) = \frac{S_0 X(z)}{D(z)} = S_0 S(z) D(z) \qquad where\, S(z) = \frac{1}{D(z)}$$

We have

$$w(n)=\frac{S_0}{2\pi}\int S(e^{j\theta})X(e^{j\theta})(e^{jn\theta})d\theta$$

Which gives

$$w^2(n)=\frac{{S_0}^2}{4\pi^2}\left|\int S(e^{j\theta})X(e^{j\theta})(e^{jn\theta})d\theta\right|^2$$

Using Schwartz inequality

$$w^2(n)\le {S_0}^2\left[\frac{1}{2\pi}\int_{2\pi}\left|S(e^{j\theta})\right|^2 d\theta\right]\left[\frac{1}{2\pi}\int_{2\pi}\left|X(e^{j\theta})\right|^2 d\theta\right]$$

Applying Parsavel's theorem

$$w^2(n)\le {S_0}^2\sum_{n=0}^{\infty}x^2(n)\left[\frac{1}{2\pi}\int_{2\pi}\left|S(e^{j\theta})\right|^2 d\theta\right] \text{ ---------- } 1$$

We know

$$z=e^{j\theta}$$

Differentiate with respect to

$$\frac{dz}{d\theta}=je^{j\theta}$$

$$dz=je^{j\theta}d\theta$$

$$d\theta=\frac{dz}{je^{j\theta}}=\frac{dz}{jz} \text{ ---------- } 2$$

Substitute equation 2 in equation 1

$$w^2(n) \le {S_0}^2\sum_{n=0}^{\infty}x^2(n)\left[\frac{1}{2\pi j}\oint_c\left|S(z)\right|^2 z^{-1}dz\right]$$

$$\le {S_0}^2\sum_{n=0}^{\infty}x^2(n)\left[\frac{1}{2\pi j}\oint_c S(z)S(z^{-1})z^{-1}dz\right]$$

$$w^2(n) \le \sum_{n=0}^{\infty}x^2(n) \text{ when } {S_0}^2\left[\frac{1}{2\pi j}\oint_c S(z)S(z^{-1})z^{-1}dz\right]=1$$

$$\text{Therefore, } S_0^2 = \frac{1}{\frac{1}{2\pi j}\oint_c S(z)S(z^{-1})z^{-1}dz}$$

$$= \frac{1}{\frac{1}{2\pi j}\oint_c \frac{z^{-1}dz}{D(z)D(z^{-1})}}$$

$$= \frac{1}{I}$$

$$\text{Where } I = \frac{1}{2\pi j}\oint_c \frac{z^{-1}dz}{D(z)D(z^{-1})}$$

Example *A digital system is characterised by the difference equation*

$$y(n) = 0.9y(n-1) + x(n)$$

with $x(n) = 0$ *and initial condition* $y(-1) = 12$. *Determine the deadband of the system.*

Solution A comparison between exact values of $y(n)$ and rounded values of $y(n)$ is given below.

n	$y(n)$-*exact*	$y(n)$-*rounded*
−1	12	12
0	10.8	11
1	9.72	10
2	8.748	9
3	7.8732	8
4	7.08588	7
5	6.377292	6
6	5.73956	5
7	5.16561	5
8	4.64905	5

From the above table, it is seen that for any value of $|y(-1)| \leq 5$, $y(n) = y(-1)$, $n \geq 0$ for zero input. Thus, the deadband is the interval $[-5, 5]$.

Example *A cascaded realisation of the two first-order digital filters is shown below. The system functions of the individual sections are*

$$H_1(z) = \frac{1}{1-0.9z^{-1}} \quad \text{and} \quad H_2(z)\frac{1}{1-0.8z^{-1}}$$

Draw the product quantisation noise model of the system and determine the overall output noise power.

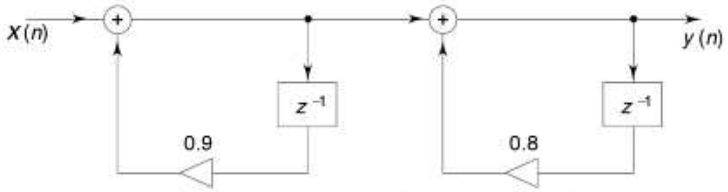
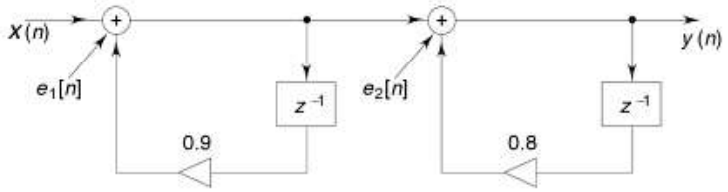


Fig. E10.5(a) *Figure for Example 10.5*



The transfer function of the system is

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

The noise transfer function seen by $e_1(n)$ is $H_1(z)$ and by $e_2(n)$ is $H_2(z)$. The overall output noise power is given by

$$\sigma_{err}^2 = \sigma_{o1}^2 + \sigma_{o2}^2$$

where, from Eq.10.52,

$$\sigma_{o1}^2 = \frac{\sigma_e^2}{2\pi j}\oint_C H(z)H(z^{-1})z^{-1}dz$$

$$= \frac{\sigma_e^2}{2\pi j}\oint_C \frac{1}{(1-0.9z^{-1})(1-0.8z^{-1})} \times \frac{1}{(1-0.9z)(1-0.8z)} \times z^{-1}dz$$

$$= \frac{\sigma_e^2}{2\pi j}\oint_C \frac{z}{(z-0.9)(z-0.8)(1-0.9z)(1-0.8z)}dz$$

$$\sigma_{o1}^2 = \sigma_e^2 \times I_1$$

where

$$I_1 = \frac{1}{2\pi j}\oint_C \frac{z}{(z-0.9)(z-0.8)(1-0.9z)(1-0.8z)}\mathrm{d}z$$

The integral I_1 can be solved by the method of residues.

I_1 = sum of the residues at the poles within the unit circle.

= (residue at $z = 0.9$) + (residue at $z = 0.8$)

$$= (z-0.9)\frac{z}{(z-0.9)(z-0.8)(1-0.9z)(1-0.8z)_z}\Bigg|_{z=0.9}$$

$$+(z-0.8)\frac{z}{(z-0.9)(z-0.8)(1-0.9z)(1-0.8z)}\Bigg|_{z=0.8}$$

$$= 89.8079$$

Therefore,

$$\sigma_{o1}^2 = \sigma_e^2 \times I_1 = 89.8079\sigma_e^2$$

Similarly,

$$\sigma_{o2}^2 = \frac{\sigma_e^2}{2\pi j}\oint_C H(z)H(z^{-1})z^{-1}\mathrm{d}z$$

$$= \frac{\sigma_e^2}{2\pi j}\oint_C \frac{1}{(1-0.8z^{-1})}\times\frac{1}{(1-0.8z)}\times z^{-1}$$

$$= \frac{\sigma_e^2}{2\pi j}\oint_C \frac{\mathrm{d}z}{(z-0.8)(1-0.8z)}$$

$$\sigma_{o2}^2 = \sigma_e^2 \times I_2$$

where

$$I_2 = \frac{1}{2\pi j}\oint_C \frac{\mathrm{d}z}{(z-0.8)(1-0.8z)}$$

The integral I_2 can be solved by the method of residues.

I_2 = sum of the residues at poles within the unit circle.

= residue at $z = 0.8$

$$= (z-0.8)\frac{1}{(z-0.8)(1-0.8z)}\Bigg|_{z=0.8}$$

$$= 2.778$$

Therefore,

$$\sigma_{o2}^2 = \sigma_e^2 \times I_2 = 2.778\ \sigma_e^2$$

The overall output noise power is,

$$\sigma_{err}^2 = \sigma_{o1}^2 + \sigma_{o2}^2$$
$$= \sigma_e^2(89.8079 + 2.778) = 92.586\sigma_e^2$$

Substituting $\sigma_e^2 = \frac{2^{-2B}}{12}$, we get

$$\sigma_{err}^2 = 92.586 \times \frac{2^{-2B}}{12} = 7.715 \times 2^{-2B}$$

Method-II

Solution Given

$$H_1(z) = \frac{1}{1-0.9z^{-1}} \text{ and } H_2(z) = \frac{1}{1-0.8z^{-1}}$$

From Eq. 10.51,

$$\sigma_{err}^2 = \sigma_{o1}^2 + \sigma_{o2}^2$$

where

$$\sigma_{o1}^2 = \sigma_e^2 \sum_{n=0}^{\infty} h_1^2(n) \text{ and } \sigma_{o2}^2 = \sigma_e^2 \sum_{n=0}^{\infty} h_2^2(n)$$

Let us first determine $h(n)$ and $h_2(n)$.

$$H(z) = \frac{1}{(1-0.9z^{-1})(1-0.8z^{-1})}$$
$$= \frac{z^2}{(z-0.9)(z-0.8)}$$

Therefore

$$\frac{H(z)}{z} = \frac{z}{(z-0.9)(z-0.8)} = \frac{9}{(z-0.9)} - \frac{8}{(z-0.8)}$$

and

$$H(z) = \frac{9}{(1-09z^{-1})} - \frac{8}{(1-0.8z^{-1})}$$

Taking inverse z-transform,

$$h_1(n) = [9(0.9)^n - 8(0.8)^n]u(n)$$

and

$$h_1^2(n) = [9(0.9)^n - 8(0.8)^n]^2 u(n)$$

Similarly,

$$h_2(n) = Z^{-1}[H_2(z)] = Z^{-1}\left[\frac{1}{1-0.8z^{-1}}\right] = (0.8)^n u(n)$$

and

$$h_2^2(n) = (0.8)^{2n} u(n)$$

$$\sigma_{o1}^2 = \sigma_e^2 \sum_{n=0}^{\infty} h_1^2(n) = \sigma_e^2 \sum_{n=0}^{\infty} \left[9(0.9)^n - 8(0.8)^n\right]^2$$

$$= \sigma_e^2 \sum_{n=0}^{\infty} (81(0.81)^n + 64(0.64)^n - 144(0.72)^n)$$

$$= \sigma_e^2 \left[\frac{81}{1-0.81} + \frac{64}{1-0.64} - \frac{144}{1-0.72}\right]$$

$$= 89.80\sigma_e^2$$

$$\sigma_{o2}^2 = \sigma_e^2 \sum_{n=0}^{\infty} h_2^2(n) = \sigma_e^2 \sum_{n=0}^{\infty} (0.8)^{2n}$$

$$= \sigma_e^2 \sum_{n=0}^{\infty} (0.64)^n$$

$$= \sigma_e^2 \left(\frac{1}{1-0.64}\right)$$

$$= 2.778\sigma_e^2$$

The overall output noise power is,

$$\sigma_{err}^2 = \sigma_{o1}^2 + \sigma_{o2}^2$$

$$= \sigma_e^2(89.80 + 2.778) = 92.578\sigma_e^2$$

Substituting $\sigma_e^2 = \frac{2^{-2B}}{12}$, we get

$$\sigma_{err}^2 = 92.578 \times \frac{2^{-2B}}{12} = 7.715 \times 2^{-2B}$$

Two Mark questions and Answers:

1. What do finite word length effects mean?

The effects due to finite precision representation of numbers in a digital system are called finite word length effects.

2.what are the three-quantization errors to finite word length registers in digital filters?

1. Input quantization error
2. Coefficient quantization error
3. Product quantization error

3. What is input quantization error?.

The filter coefficients are computed to infinite precision in theory. But in digital computation the filter coefficients are represented in binary and are stored in registers. If a b bit register is used the filter coefficients must be rounded or truncated to b bits, which produces an error.

4 .What is product quantization error?.

The product quantization errors arise at the out put of the multiplier. Multiplication of a b bit data with a b bit coefficient results a product having 2b bits. Since a b bit register is used the multiplier output will be rounded or truncated to b bits which produces the error.

5.what are the different types of arithmetic in digital systems.?

There are three types of arithmetic used in digital systems. They are fixed point arithmetic, floating point ,block floating point arithmetic.

6.What do you understand by input quantization error?

In digital signal processing, the continuous time input signals are converted into digital using a b-bit ACD. The representation of continuous signal amplitude by a fixed digit produce an error, which is known as input quantization error.

7. List some of the finite word length effects in digital filters?

1. Errors due to quantization of input data.
2. Errors due to quantization of filter co-efficient
3. Errors due to rounding the product in multiplications