DIGITAL FILTER STRUCTURES AND QUANTIZATION EFFECTS

6.1: Direct-form network structures

- So far, we have assumed infinite-precision arithmetic when implementing our digital controllers.
- In reality, this is not possible.
  - If we are lucky, we have the luxury of using floating-point arithmetic (single- or double-precision).
  - Often times, however, we will be restricted to fixed-point arithmetic due to cost constraints.
  - Both suffer quantization and overflow effects (although to a much lower degree in floating-point implementations).
- We look at these real-world considerations in this unit of notes.

Canonical network structures

- The first consideration is how to implement a transfer function $D(z)$ as a "network" of primitive arithmetic operations: add, multiply, delay.
  - For software realizations, the network corresponds to a flowchart of the filter algorithm.
  - For hardware realizations, the network describes the actual circuit elements and their interconnection.

- We will see that the performance of a digital implementation is affected substantially by the choice of network structure.
- There are a number of named "canonical" network structures for implementing a transfer function, all of which have
  - N delay elements,
  - $2N$ (2-input) adders,
  - $2N+1$ multipliers.

Direct-form II canonical form

- To obtain the so-called "direct forms" of filter structure, we start with the familiar LCCDE equation:

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{m=0}^{M} b_m x[n-m]$$

- For convenience,
  - Assume $a_0 = 1$ (We can scale other a_k and b_k so that this is true).
  - Assume $N = M$ (We can always make some coefficients a_k or b_k equal to zero to make this true).
- Then, we get that the transfer function is equal to:

$$H(z) = \frac{\sum_{n=0}^{N} b_n z^{-n}}{1 + \sum_{n=1}^{N} a_n z^{-n}} = \frac{1}{1 + \sum_{n=1}^{N} a_n z^{-n}} \times \sum_{n=0}^{N} b_n z^{-n}.$$

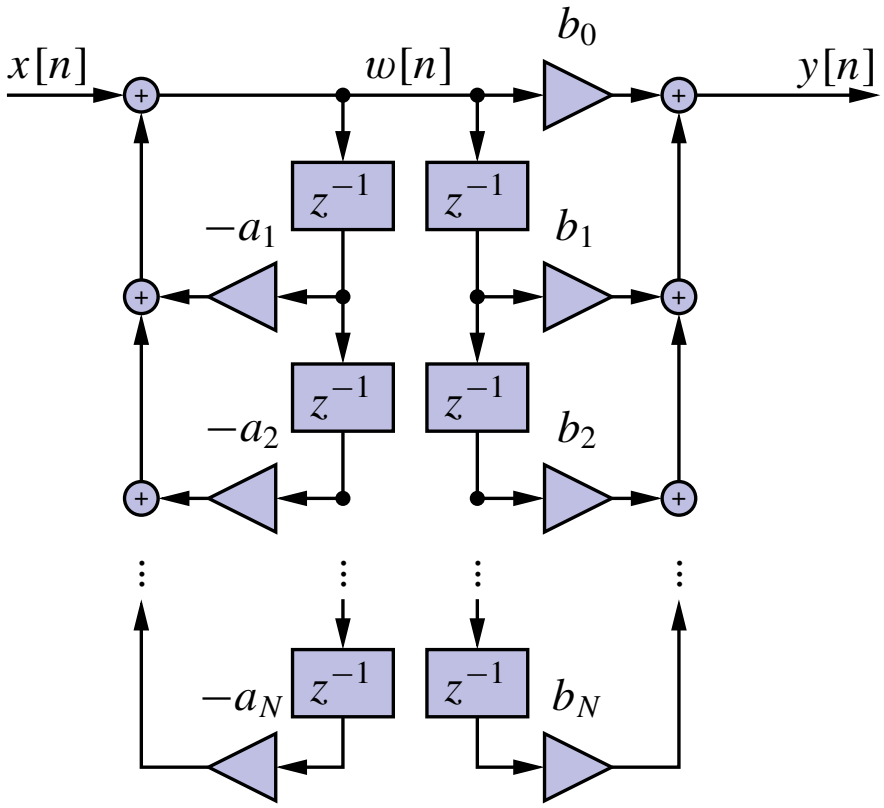
- We can realize this transfer function as a cascade of the denominator dynamics followed by the numerator dynamics:
  - The denominator dynamics implement a feedback path with

$$w[n] = x[n] - a_1 w[n-1] - a_2 w[n-2] - \cdots - a_N w[n-N].$$

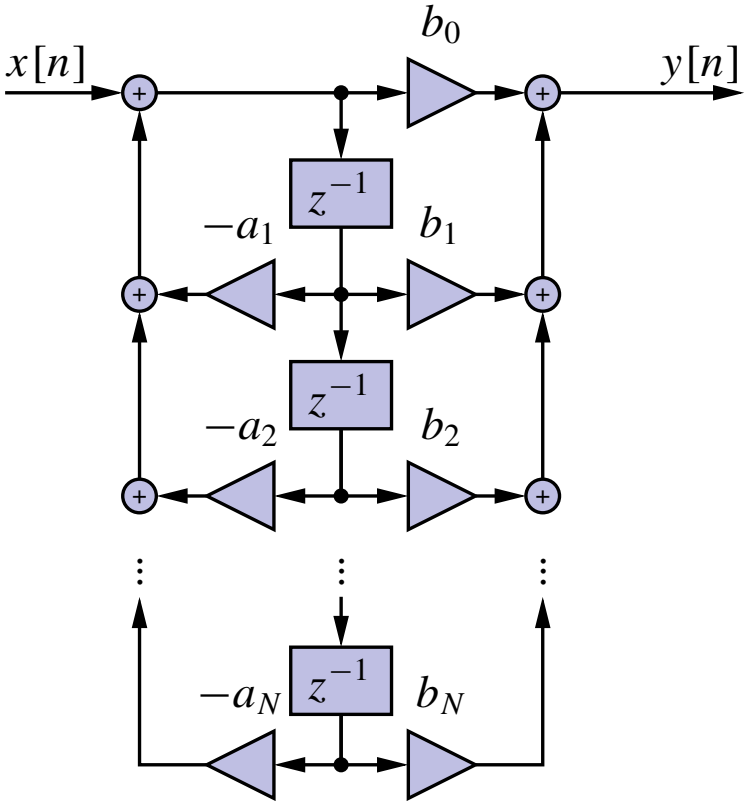
- The numerator dynamics implement a feedforward path with

$$y[n] = b_0 w[n] + b_1 w[n-1] + b_2 w[n-2] + \cdots + b_N w[n-N].$$

- We can realize the overall transfer function as shown:

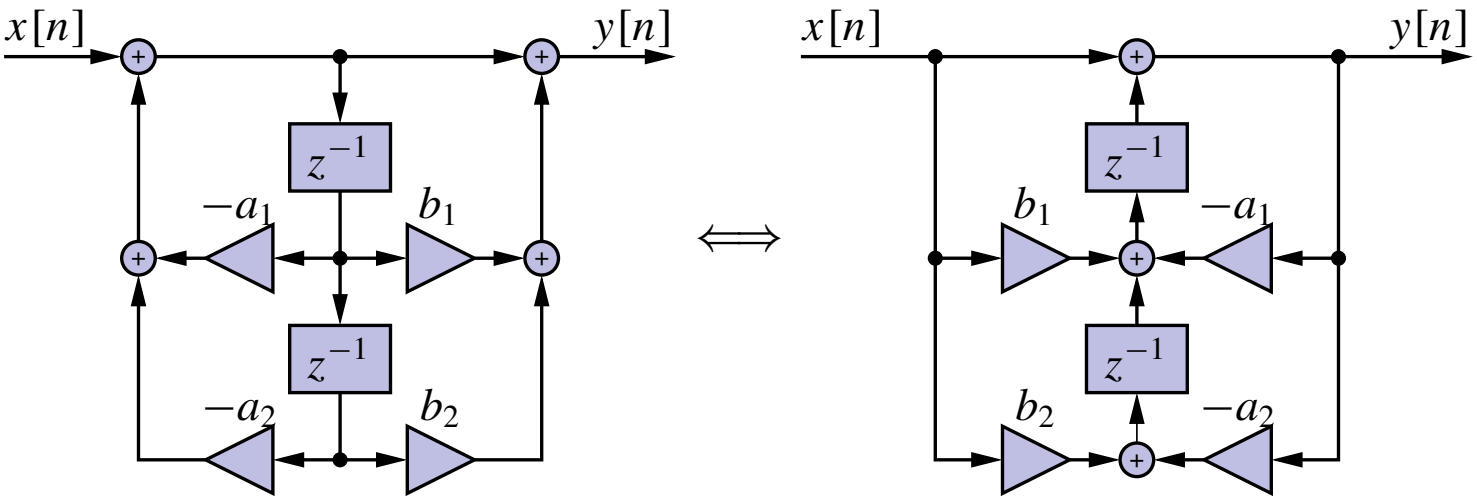


- Notice that it is possible to combine the delay elements and end up with a simpler structure:



Transpose networks and the direct-form I canonical form

- For any given network with a certain transfer function $H(z)$, we can generate a "transpose" of the network that has a different structure but the same transfer function.
- To obtain the transpose, follow these steps:
  1. Start with the original network structure.
  2. Replace all data flows with flows in the reverse direction.
  3. Replace all summation nodes with branch nodes and all branch nodes with summations.
  4. Exchange $x[n]$ and $y[n]$.



- If we take the transpose of a "Direct Form II" network (example on left), we obtain a "Direct Form I" network structure (example on right).
- The direct forms are simple to grasp and to implement, but have some serious problems for high-order transfer functions.
- Therefore, we look at several other canonical forms.

6.2: Parallel and cascade-form network structures

Parallel canonical forms I and II

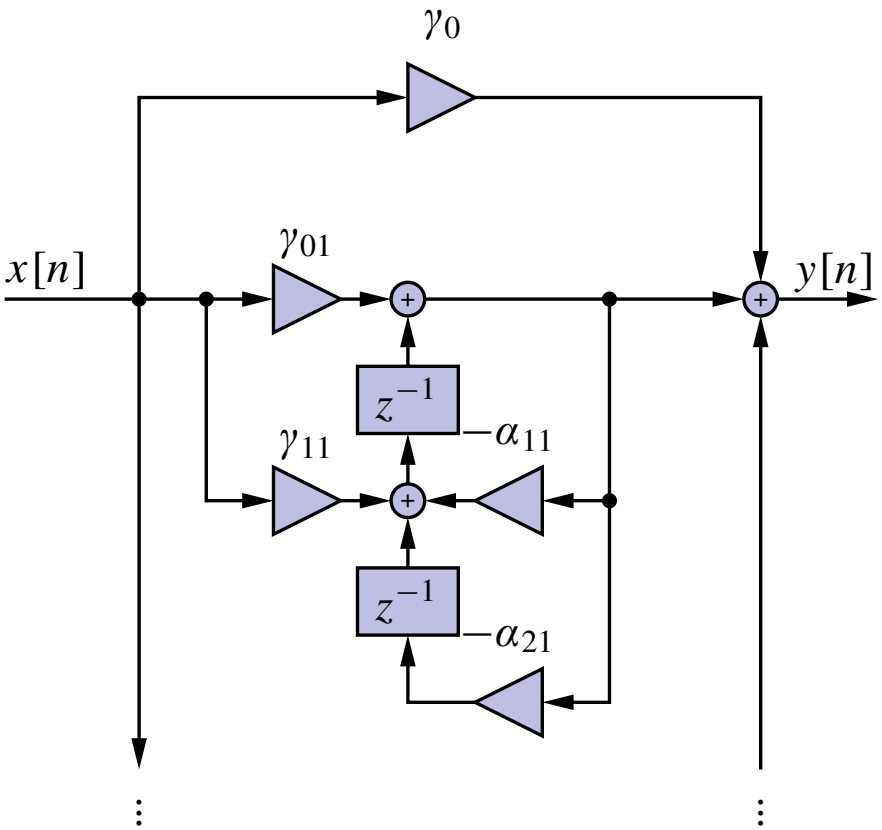
- To obtain the "Parallel Forms" of filter structure, re-write $H(z)$ as a partial fraction expansion:

$$H(z) = \gamma_0 + \sum_{i=1}^{L} \frac{\gamma_{0i} + \gamma_{1i} z^{-1}}{1 + \alpha_{1i} z^{-1} + \alpha_{2i} z^{-2}}$$

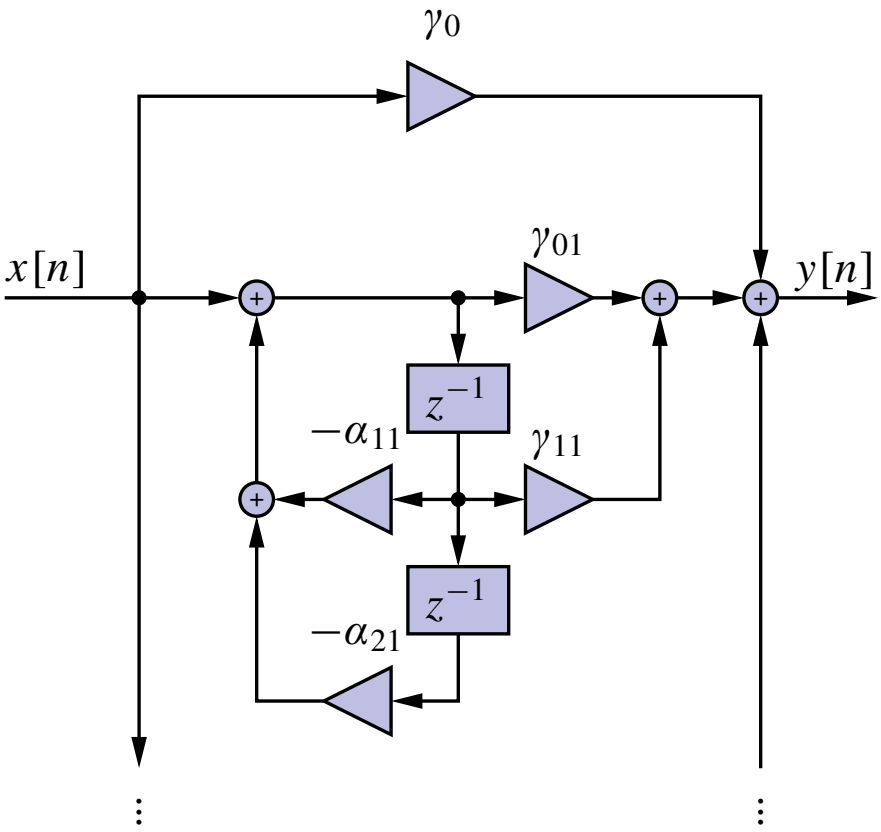
  where $L = \left\lfloor \frac{N+1}{2} \right\rfloor$.

- If N is odd, there will be one first-order term in the summation.
- The structures of the two parallel forms look like:

PARALLEL FORM I:



PARALLEL FORM II:



- Again, these two forms are transposes of each other.
- Also, they have the same number of delays, (two input) adders and multipliers as the direct form implementations.

Cascade canonical forms I and II

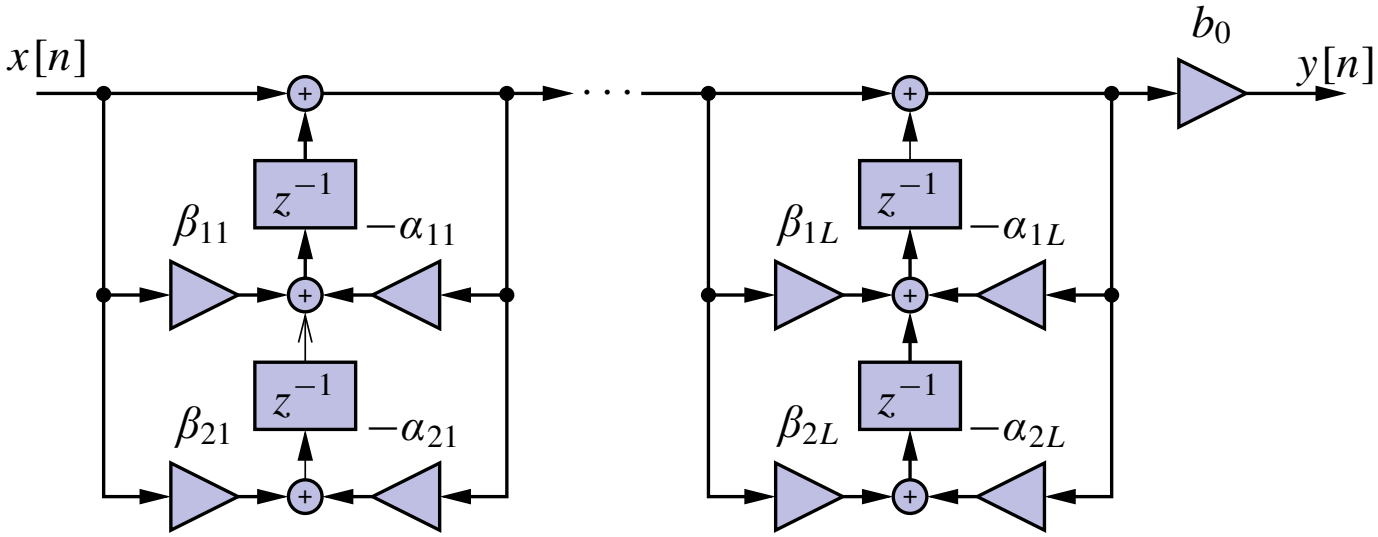
- To obtain the "Cascade Forms" of filter structure, re-write $H(z)$ as a product of second-order polynomials:

$$H(z) = b_0 \prod_{i=1}^{L} \frac{1 + \beta_{1i} z^{-1} + \beta_{2i} z^{-2}}{1 + \alpha_{1i} z^{-1} + \alpha_{2i} z^{-2}}$$

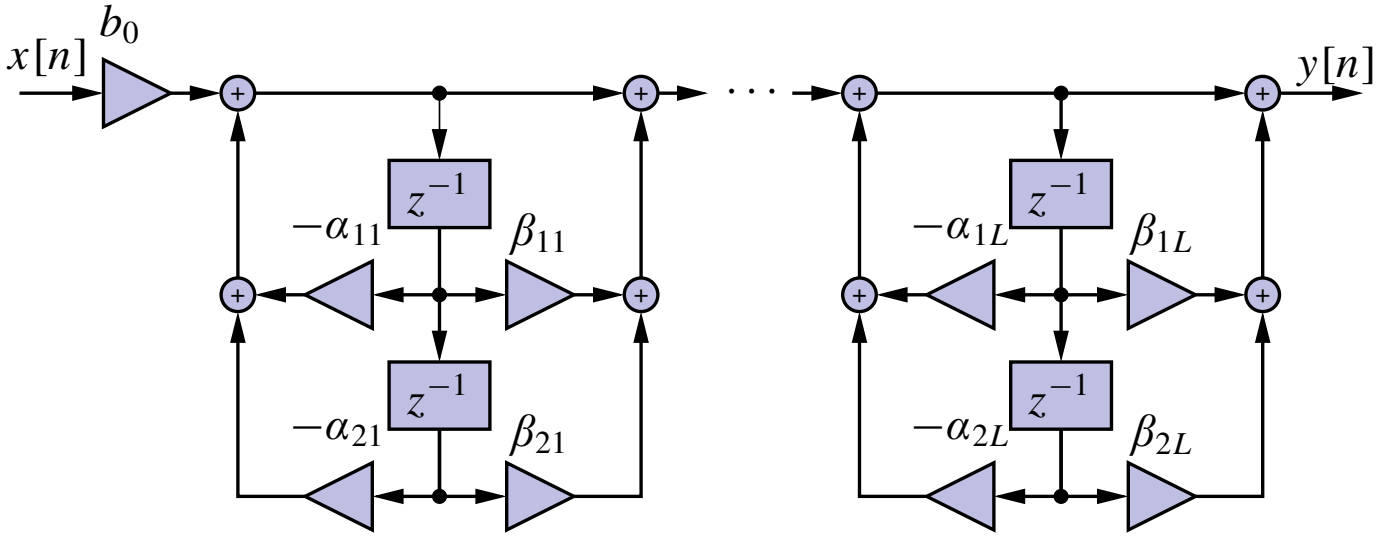
  where $L = \left\lfloor \frac{N+1}{2} \right\rfloor$.
- If N is odd, there will be one first-order term in the product ($\alpha_{2L} = \beta_{2L} = 0$.)

- The structures of the two cascade forms look like:

CASCADE FORM I:



CASCADE FORM II:



6.3: Implications of fixed-point arithmetic

- Now that we have studied the most common filter structures, we examine some of the real-life implications of using each structure.
- There are tradeoffs: To determine which is best for a particular case, there are four different factors that we must look at:

1. Coefficient quantization: The filter parameters are quantized such that they are implemented in finite precision.
   - Thus, the filter we implement is not exactly $H(z)$, the desired filter, but $\widehat{H}(z)$, which we hope is "close" to $H(z)$.
   - We can check whether $\widehat{H}(z)$ meets the control design specifications, and modify it if necessary.
   - The structure of the filter network has a drastic effect on its sensitivity to coefficient quantization.
2. Signal quantization: Rounding or truncation of signals in the filter.
   - It occurs during the operation of the filter and can best be viewed as a random noise source added at the point of quantization.
3. Dynamic range scaling: We sometimes need to perform scaling to avoid overflows (where the signal value exceeds some maximum permitted value) at points in the network.
   - In cascade systems, the ordering of stages can significantly influence the attainable dynamic range.
4. Limit cycles: These occur when there is a zero or constant input, but there is an oscillating output.
   - Note that this arises from nonlinearities in the system (quantization), since a linear system cannot output a frequency different from those present at its input.

Binary representations of real numbers

- There are two basic choices (with some variations) when implementing non-integer numbers on a digital computer: floating-point and fixed-point representation.
- The fixed-point number system we will consider uses $B+1$ bits:

$$\hat{x} = X_m \left(-b_0 + \sum_{i=1}^{B} b_i 2^{-i} \right) = X_m \hat{x}_B.$$

- We represent our number in binary as: $\hat{x}_B = b_0.b_1 b_2 b_3 \cdots b_B$.
  - If $b_0 = 0$, then $0 \leq \hat{x} \leq X_m(1 - 2^{-B})$.
  - If $b_0 = 1$, then $-X_m \leq \hat{x} < 0$.
- The "step size" of quantization is the smallest interval between two quantized numbers $= X_m 2^{-B} = \Delta$.
- The fixed-point conversion error is: $e = X_m \hat{x}_B - x$.
  - For twos-complement truncation, $-\Delta < e \leq 0$;
  - For twos-complement rounding $-\Delta/2 < e \leq \Delta/2$.
- Floating point arithmetic splits the $B+1$ bits into "mantissa" bits M and "exponent" bits E such that

$$\hat{x} = M 2^E \qquad \text{where } -1 \leq |M| < 1.$$

- The exponent basically allows moving the decimal point (actually, binary point) around in the number.
  - With a fixed number of bits, floating-point representation allows much greater dynamic range of the values it can represent.

- However, mathematical operations using floating point arithmetic are much harder to implement.
  - Hardware implementations require significantly more logic than fixed-point implementations;
  - Software implementations require significantly more operations, so run much more slowly on the same processor.

- Floating point arithmetic, although much harder to implement, poses less of a problem to the digital filter designer than fixed point arithmetic. Why?

EXAMPLE: Multiply $13/16$ by $9/128$.

- Note, $\frac{13}{16} = \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{16}\right) \times 2^0$ and $\frac{9}{128} = \left(\frac{1}{2} + \frac{1}{16}\right) \times 2^{-3}$.

Fixed Point	Floating Point
0.11010000	0.11010 000
$\times$0.00010010	$\times$0.10010 101
110100000	$= 0.01110101\ 101$
110100000000	$= 0.11101\ 100$
$= 0.000011101010000$	

- Comments:
  - To store the result without losing precision, fixed point requires 12 bits but floating point requires 11 bits.
  - Generally, floating-point results will be much better than fixed-point results: double-precision math almost never gives problems to the control engineer.
  - Hardware DSP chips exist with floating point arithmetic built in.

6.4: Coefficient quantization effects

Direct-form implementations

- Recall that for a Direct Form implementation of an IIR system, we write the transfer function of the system as:

$$H(z) = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}.$$

- In our filter designs so far, we have assumed that the coefficients a_k and b_k of the filters can be arbitrary real numbers.
- To realize this system we need to quantize the coefficients a_k, and b_k to fit our maximum precision.
- Thus, the transfer function that we actually realize is:

$$\widehat{H}(z) = \frac{\sum_{k=0}^{N} \hat{b}_k z^{-k}}{1 + \sum_{k=1}^{N} \hat{a}_k z^{-k}}.$$

- The question is: "What effect does this have on our system's implementation?"
- Alternately, "What happens to the poles and zeros of the system when we quantize the coefficients?"
  - Clearly, the poles and zeros realized by the system will be the poles and zeros of $\widehat{H}(z)$, not of $H(z)$.
  - The zeros are obtained by factoring the numerator of $\widehat{H}(z)$, and the poles are obtained by factoring the denominator of $\widehat{H}(z)$.
    - We see that the position of one particular zero is affected by *all* of the coefficients $\hat{b}_k$, and that the position of a particular pole is affected by *all* of the coefficients $\hat{a}_k$.

- Thus, the effect of quantizing a specific coefficient is not isolated in any way, but affects all poles or zeros in the function.

- A non-obvious result, which may be obtained with a little math, is that closely clustered poles and zeros are most affected.
  - Thus, narrow-band filters of any sort will be most strongly changed.

QUESTION: So, what do we do about this?

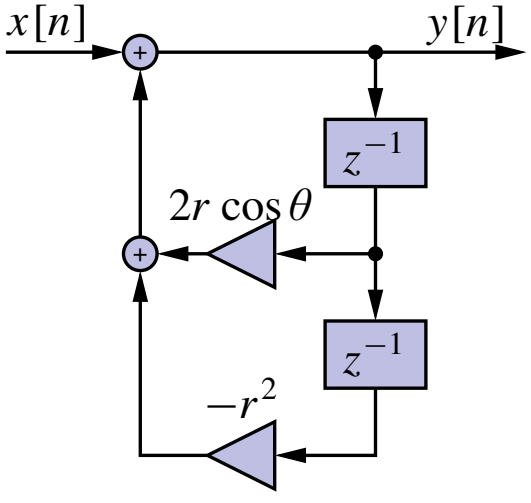
ANSWER: We try to control the effect by localizing it using parallel or cascade forms.

Coefficient quantization in a 2nd-order direct-form section

- Cascade and parallel form implementations are made up of sections of cascaded or parallel second-order direct-form filters.
- Thus, to be able to see how coefficient quantization affects these filter forms, we must first examine how it affects the 2nd-order section.
- A second-order section with complex poles can be written as:

$$H(z) = \frac{1}{(1 - re^{j\theta}z^{-1})(1 - re^{-j\theta}z^{-1})} = \frac{1}{(1 - 2r\cos\theta z^{-1} + r^2 z^{-2})}.$$

- These can be realized by the filter form on the right.
- The coefficients that require quantization are: $r\cos\theta$ and $-r^2$.
- Note: If both $r^2 < 1$ and $|r\cos\theta| < 1$ then the system is stable.

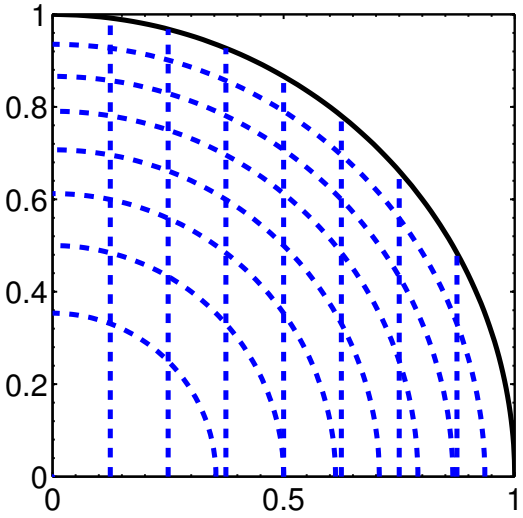


QUESTION: Why do we quantize $r\cos\theta$ and not $2r\cos\theta$?

ANSWER: The multiply by 2 can be performed exactly by a shift register.

EXAMPLE: To see which poles can be implemented by this structure using fixed-point arithmetic, suppose that four bits are available.

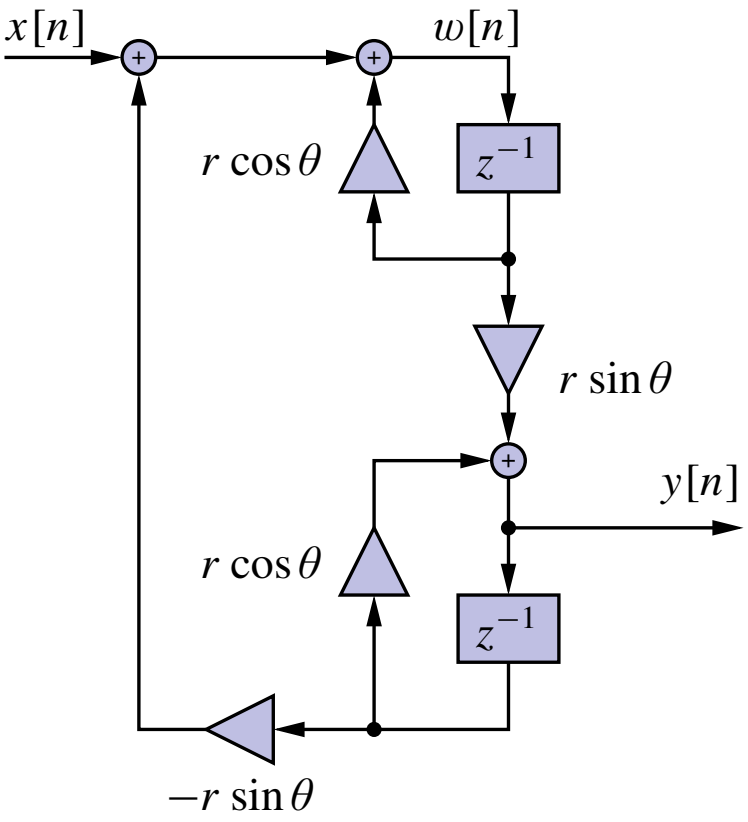
- The coefficient structure is then $a.bcd$.
- The positive values we can represent are: 0.001, 0.010, $0.011\ldots$ and 0.111. (We can similarly compute negative values.)
- So, $r\cos\theta$ can take on these eight values, and so can r^2.
- The intersection of allowable coefficients give us the grid shown.

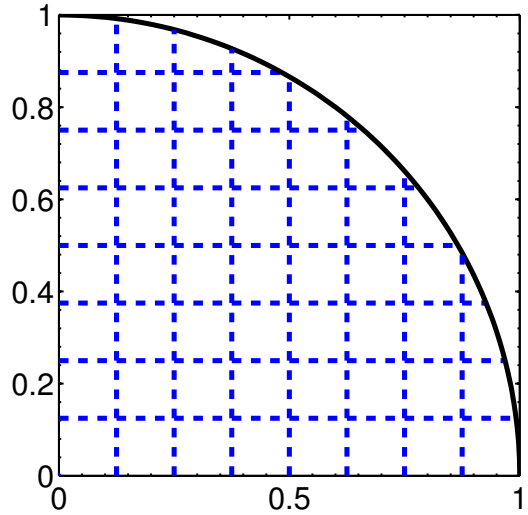


- Observations:
  - Narrow-band lowpass and narrow-band highpass filters are most sensitive to coefficient quantization, and thus require more bits to implement.
  - Sampling at too high a rate is bad, since it pushes poles closer to $+1$ (where the implementable pole density is smallest.)
  - Sensitivity increases for higher-order direct form realizations. *Poles may end up outside the unit circle after rounding—unstable!*

SOLUTION: Rather than implementing the second-order section in direct form, we can choose an alternate network structure. For example, the coupled form implementation is shown,

- All of the coefficients of this network are of the form $r\cos\theta$ and $r\sin\theta$.
- The quantized poles are at intersections of evenly spaced horizontal and vertical lines.





- The price we pay for this improved pole distribution is an increased number of multipliers: four instead of two.

Coefficient quantization in cascade- and parallel-form implementations

- Cascade- and parallel-form realizations consist of combinations of second-order direct-form systems.
- However, in both cases, each pair of complex-conjugate poles is realized independently of all the other poles.
- Thus, the error in a particular pole pair is independent of its distance from the other poles of the transfer function.
  - For the cascade form, the same argument holds for the zeros since they are realized as independent second-order factors.

- Thus the cascade form is generally much less sensitive to coefficient quantization than the equivalent direct-form realization.

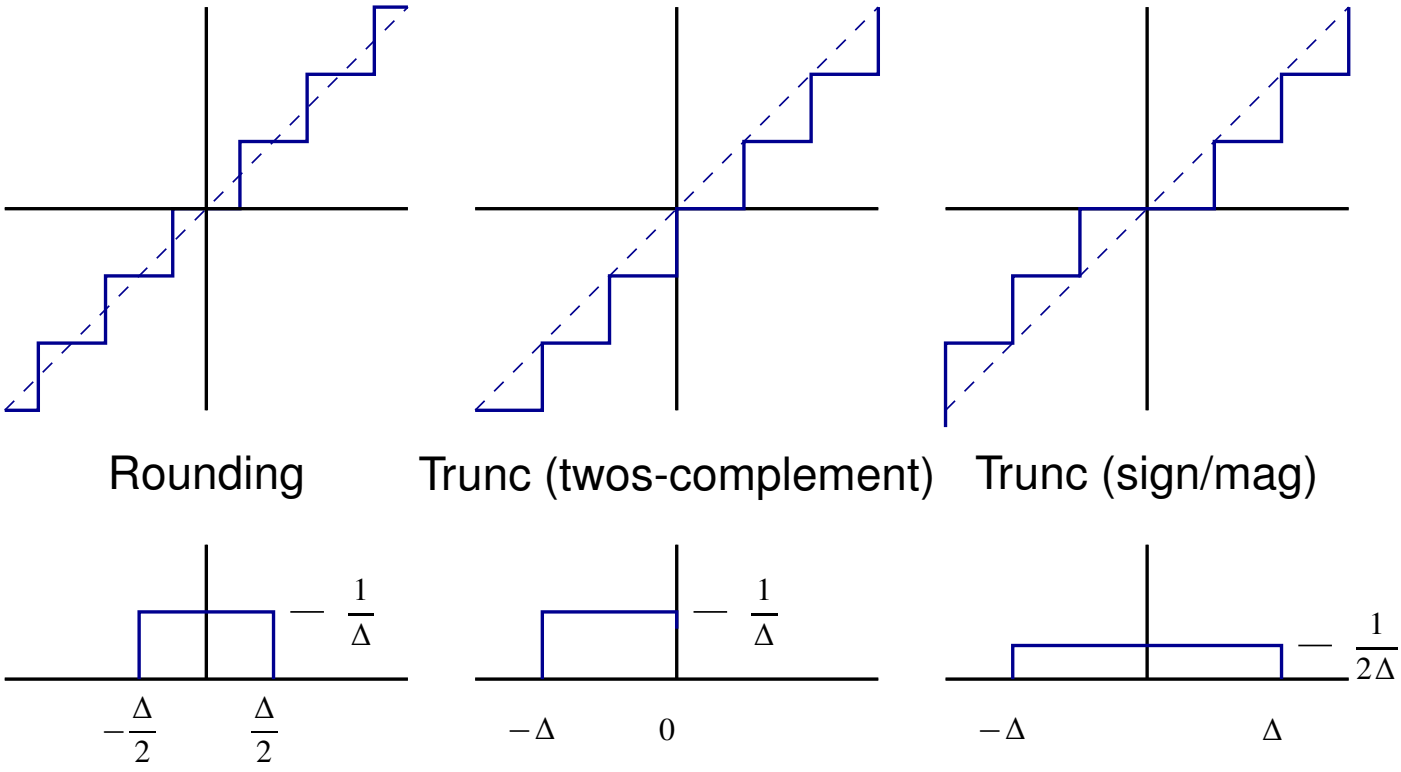
- The zeros of the parallel form function are realized implicitly.
  - They result from combining the quantized second-order sections to obtain a common denominator.
  - Thus, a particular zero location is affected by quantization errors in the numerator and denominator coefficients of all the second-order sections.

6.5: A2D/D2A signal quantization effects

- In a typical digital filtering system there will be multiple sources of signal quantization:

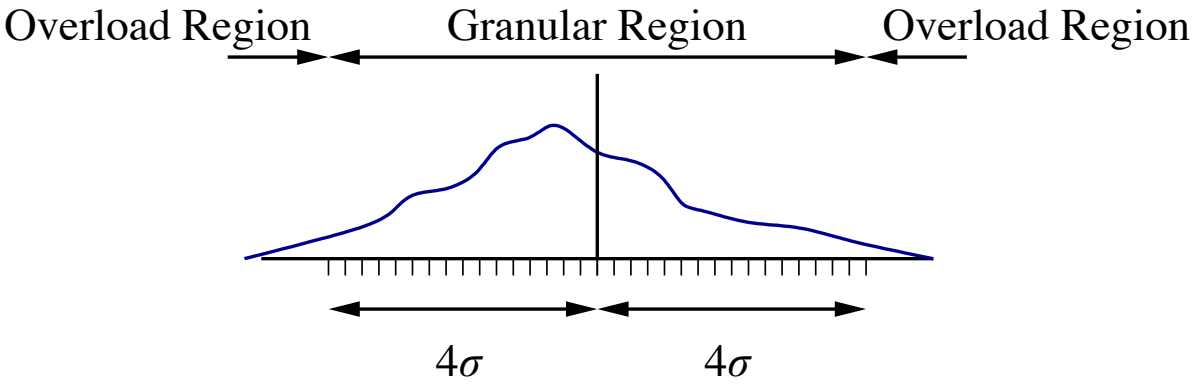
Quantization in the A2D conversion process

- The first source of quantization error is the A2D conversion process.
- We can quantize a signal in at least three different ways:

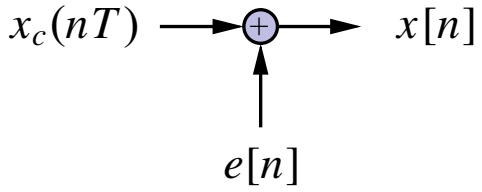


- Rounding is the most common as it generally gives the best results.
- The limits $\pm X_m$ of the quantized range might be selected using known maximum values of a particular input signal.

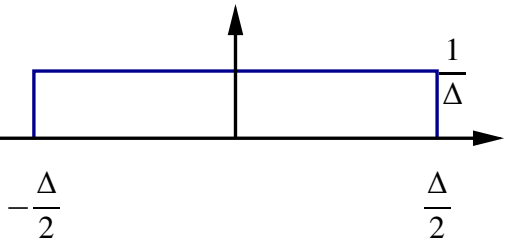
- Alternately, we might choose X_m based step size based on the variance of the input signal.
  - Assume a random input with variance σ^2 and zero mean.
  - Then, we might let $X_m = 4\sigma$.
  - Any sample greater in magnitude than 4σ gets quantized to the closest bin (this is called "four sigma scaling".)
  - We could have chosen $X_m = 3\sigma$ (three-sigma scaling), or even $X_m = 5\sigma$ (five-sigma scaling).
- For a given number of quantizer bins (and therefore bits in the representation), we are trading off quantization distortion in the overload or granular regions.



- The quantizer error $e[n]$ is equal to $x[n] - x_c(nT)$. If we assume minimal overload error, we can model $e[n]$ as:



where $e[n]$ is a random process having probability density function:



- The mean of the quantization error is:

$$\mathbb{E}\big[e[n]\big] = \int_{-\Delta/2}^{\Delta/2} e\frac{1}{\Delta}\,\mathrm{d}e = \frac{1}{\Delta}\cdot\frac{e^2}{2}\bigg|_{-\Delta/2}^{\Delta/2} = \frac{1}{\Delta}\left[\frac{\Delta^2}{8} - \frac{\Delta^2}{8}\right] = 0.$$

- The variance (power) of the quantization error is:

$$\mathbb{E}\big[(e[n] - \overline{e}[n])^2\big] = \int_{-\Delta/2}^{\Delta/2} e^2\frac{1}{\Delta}\,\mathrm{d}e = \frac{\Delta^2}{12}.$$

- The quantization error $e[n]$ and the quantized signal $x[n]$ are independent.
- $e[n]$ *does* depend on $x_c(nT)$, but for reasonably complicated signals and for small enough Δ, they are considered uncorrelated.
- What is the signal to noise ratio?
- The signal power is $\sigma^2 = \left(\frac{X_m}{4}\right)^2$ (assuming four-sigma scaling). The quantizer error power is $\mathbb{E}[e^2] = \frac{\Delta^2}{12}$.
- Now, if we have $B+1$ bits to store the quantized result, then there will be 2^{B+1} bins of width Δ.

$$\Delta = \frac{2X_m}{2^{B+1}} = X_m 2^{-B}$$

- Therefore,

$$\mathbb{E}[e^2] = \frac{X_m^2}{12}\cdot 2^{-2B}.$$

- The signal to noise ratio is then

$$\begin{aligned}\text{SNR} &= 10\log_{10}\left[\frac{\left(\frac{X_m}{4}\right)^2}{\frac{X_m^2}{12}\cdot 2^{-2B}}\right] \\ &= 10\log_{10}\left[2^{2B}\right] + 10\log_{10}\left[12/16\right] \\ &= 6.02B - 1.25 \quad \text{[dB]}\end{aligned}$$

- So, for each additional bit, we get a 6.02 dB increase in SNR.
- For 16 bits ($B = 15$), SNR=89 dB.
- It would seem that changing the k-sigma scaling of the quantizer will only change the 1.25 dB constant, but recall that we assumed minimal overload error.
  - Changing the k-sigma scaling will change the pdf of the error process, and make our approximation less (or more) realistic. The formula is good for values of k around four.

Quantization in the D2A conversion process

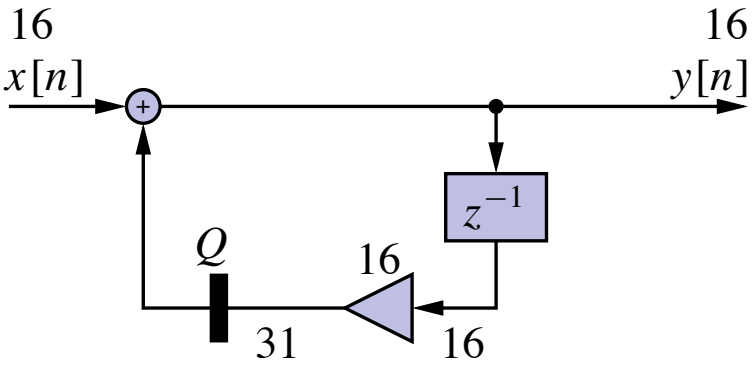
- Now, let's consider the quantizer in the D2A converter.
- Assuming that we quantize the internal 32-bit data to 16 bits to be used by the D/A converter, the quantization noise can be approximated by a continuous random process uniformly distributed between half of the step size, similar to the case of the A2D converter.
- This lowers the total SNR by another 3dB. (To calculate total SNR, all the noise powers are added up first, and then the ratio is calculated. 3dB simply means that the signal-to-noise power ratio is halved, as the noise power is doubled.)

$$\text{SNR} = \frac{S}{N_1 + N_2 + \ldots}$$

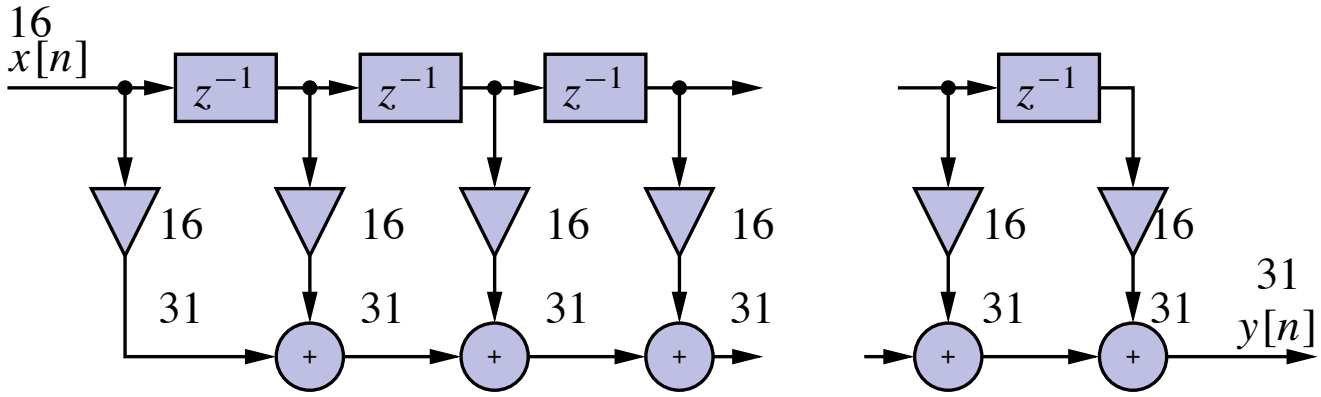
- So, if we started out with 89 dB because we performed 16 bit quantization with our sampler, then the resulting SNR after D2A conversion is 86 dB.

6.6: DSP-calculation signal quantization effects

- The second quantization noise source, introduced by rounding and truncation inside a DSP, is more complicated.
- First, let's take any two numbers, a and b, of 16 bits each.
- If we multiply them together, ab will need at most 31 bits to represent the result without losing precision.
- Let's look at quantization noise in IIR filters:



- IIR filters cannot be implemented with perfect precision.
  - A quantizer must be placed at the output of a multiplier to limit the number of bits in a recursive calculation.
  - We need to quantify this quantization noise in order to determine the number of bits needed to satisfy a given SNR.
- FIR filters, on the other hand, can be implemented without quantization noise, if the number of bits increases with the filter order.
  - We seldom do this, however as the A2D and D2A converters have already introduced some quantization noise.
  - There is no point to enforce perfect precision inside a DSP when a non-zero noise power already exists.

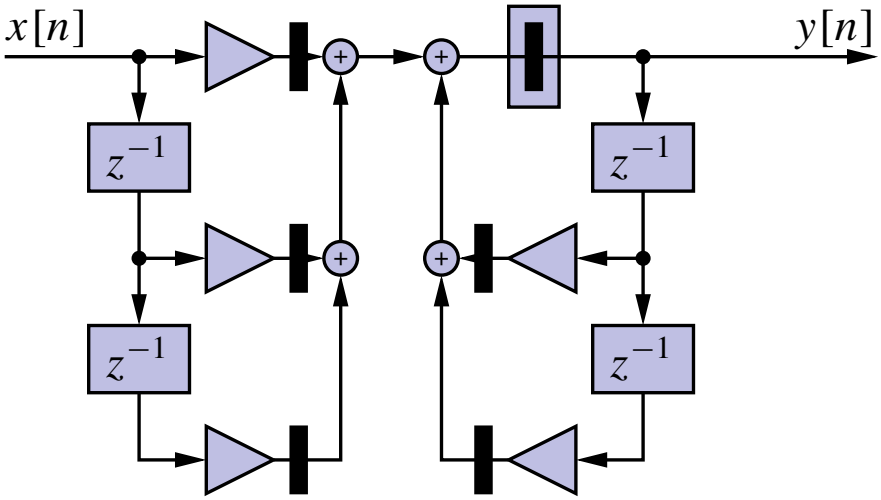


- There are two effects to be considered when we design a quantizer for multipliers and accumulators:

  1. Quantizer step size should be kept as small as possible because of the square term in quantization noise power, $\Delta^2/12$.
  2. X_m should be large enough to prevent overflow.

- Of course, these two effects contradict each other and represent a design trade-off.

EXAMPLE: Consider the following second-order section:



QUESTION: How many quantization noise sources are there in the filter?

ANSWER: It depends on where the quantizers are placed. We can have either 1 or 5.

- Assuming we have M noise sources, then the total noise power will be $M \times \Delta^2/12$.
- To calculate the noise power after filtering, we need to borrow some math from random processes.
  - The Fourier transform of a random process doesn't exist, but the Fourier transform of the autocorrelation function of a stationary random process does exist, called the **power spectrum density**.
  - The power spectrum density at the output of a filter is the multiplication of the power spectrum density of the input and the square of the filter frequency response, as shown in the following figure:

$$\underset{x[n]}{P_X(e^{j\omega})} \longrightarrow \boxed{H(e^{j\omega})} \longrightarrow \underset{y[n]}{P_Y(e^{j\omega})} = P_X(e^{j\omega})|H(e^{j\omega})|^2$$

- The noise power at the output of a filter is the integration of its power spectrum density:

$$\sigma_f^2 = \sigma_e^2 \times \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|H(e^{j\omega})\right|^2 \,\mathrm{d}\omega$$

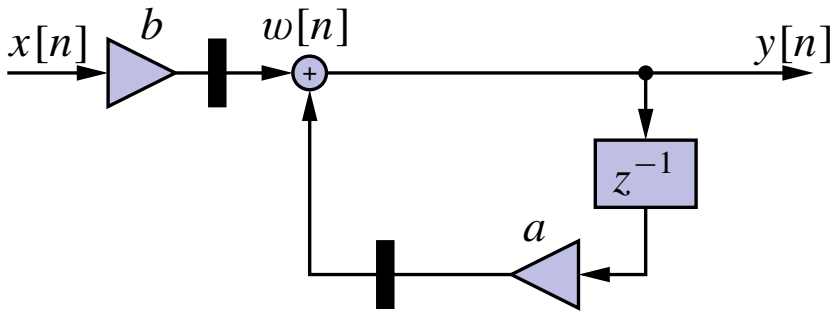
  where we have assumed that the input noise power spectrum density is a constant variance σ_e^2 (a white random process with zero mean).
- By Parseval's Theorem,

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

$$\text{so: } \sigma_f^2 = \sigma_e^2 \sum_{n=-\infty}^{\infty} |h[n]|^2.$$

- The output noise power can be calculated by either equation, as an integral of the power spectrum density over all frequencies, or as a scaled infinite summation of the impulse response squared.
- Usually it is easier to calculate the infinite summation if the impulse response takes on a closed form.

EXAMPLE: Consider the following filter:



- We first find the transfer function from $w[n]$ to $y[n]$

$$y[n] = w[n] + ay[n-1]$$

$$Y(z)(1 - az^{-1}) = W(z)$$

$$H_{yw}(z) = \frac{Y(z)}{W(z)} = \frac{1}{1 - az^{-1}} = \frac{z}{z - a}.$$

- We next find the impulse response from $w[n]$ to $y[n]$

$$h_{yw}[n] = a^n 1[n].$$

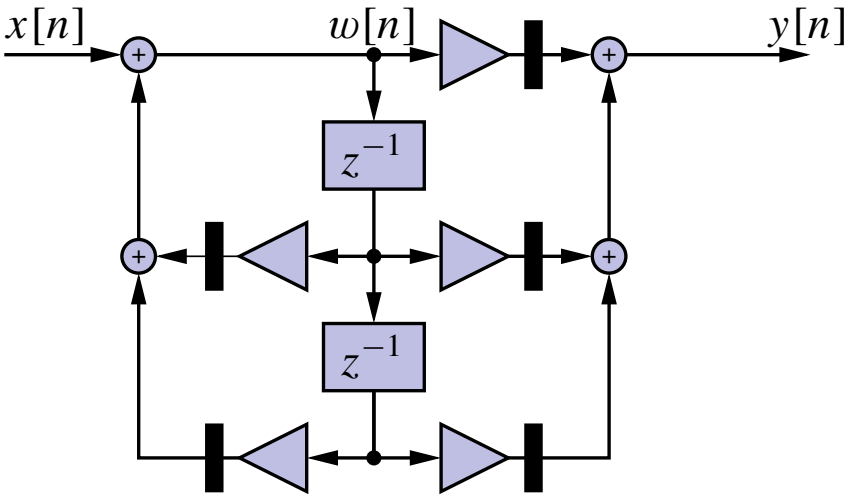
- Next, the infinite summation of the squared absolute value of the impulse response

$$\sum_{n=-\infty}^{\infty} |a^n 1[n]|^2 = \sum_{n=0}^{\infty} |a^2|^n = \frac{1}{1 - a^2}.$$

- The noise power at the filter output is:

$$\underbrace{2}_{\substack{2\\ \text{sources}}} \times \underbrace{\frac{X_m^2 2^{-2B}}{12}}_{\sigma_e^2} \times \underbrace{\frac{1}{1-a^2}}_{\sum |h[n]|^2}$$

EXAMPLE: Consider the following filter:



- The noise power at the filter output is:

$$2 \times \frac{X_m^2 2^{-2B}}{12} \sum_{n=-\infty}^{\infty} |h_{yx}[n]|^2 + 3 \times \frac{X_m^2 2^{-2B}}{12}$$

- To calculate the signal power, we use the same concept of power spectrum density since the input signal is merely another random process.
- The signal power at the output of a filter is usually expressed as:

$$\sigma_s^2 = \sigma_{\text{input}}^2 \sum_{n=-\infty}^{\infty} |h[n]|^2$$

  where σ_{input}^2 is the input signal power. [Input assumed "white" too!]
- The total SNR is the ratio of the signal power to the noise power.

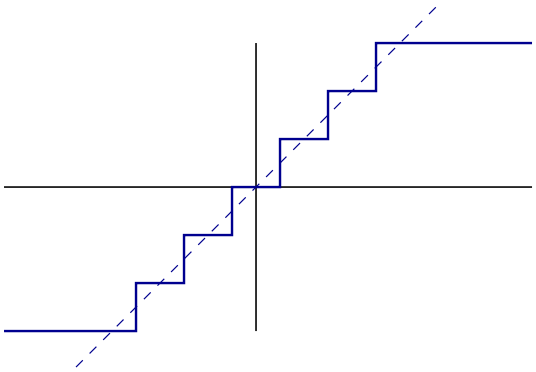
6.7: Dynamic range scaling: Criteria

- Overflow occurs when two numbers are added and the result requires more than $B+1$ bits to store.

EXAMPLE: For four-bit quantization:

$$\begin{array}{rll} & 0.111 & \text{positive number } (7/8) \\ + & 0.011 & \text{positive number } (3/8) \\ \hline & 1.010 & \text{negative number } (-6/8) \end{array}$$

- In this overflow situation we can either "clip" the result or leave it as is.
- To clip the result, replace a positive result with 0.1111...; a negative result with 1.00000....
- The input/output relationship of a clipping (saturating) quantizer is:
- We would like to ensure that saturation does not occur.



Scaling criterion #1: Bounded-input-bounded output.

- In this criterion we want to make sure that every node in the filter network is bounded by some number.
- If we follow the convention that each number represents a fraction of unity (with a possible implied scaling factor) each node in the network must be constrained to have a magnitude less than 1 to avoid overflow.
- Pick any node in the network $w[n]$. Its response can be expressed as:

$$w[n] = \sum_{m=-\infty}^{\infty} x[n-m]h_{wx}[m]$$

$$|w[n]| = \left| \sum_{m=-\infty}^{\infty} x[n-m]h_{wx}[m] \right|$$

$$\leq \sum_{m=-\infty}^{\infty} |x[n-m]h_{wx}[m]|$$

$$\leq x_{\max} \sum_{m=-\infty}^{\infty} |h_{wx}[m]|.$$

- To make sure that $|w[n]| < 1$, we need to introduce a scaling factor s such that:

$$x_{\max} s \leq \frac{1}{\sum_{m=-\infty}^{\infty} |h_{wx}[m]|}$$

- s should always be less than 1.
- If s is greater than 1, then $x_{\max}$ is less than 1, implying that the previous stage didn't use the full dynamic range available. Dynamic range is wasted, reducing signal power unnecessarily.
- The design of the scaling factor is therefore to always give the signal the maximum dynamic range possible, as the quantization noise power is a constant once the step size Δ is determined.
- Bounded-input-bounded-output criterion usually results in a very small s, which reduces the signal power and therefore the overall SNR.

Scaling criterion #2: Frequency-response criterion.

- This criterion assumes we are inputting a narrow-band signal $x[n] = x_{\max} \cos \omega_0 n$ to the filter.
- To avoid overflow at node $w[n]$ given this input signal, we need to ensure that

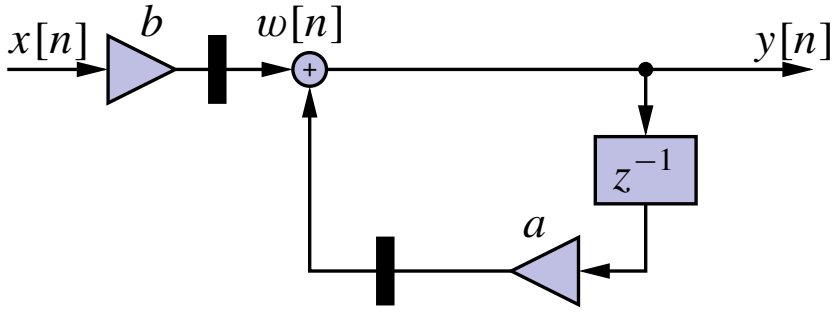
$$|w[n]| = |H_{wx}(e^{j\omega_0})| x_{\max} \leq 1.$$

- The scaling factor s should be chosen so that:

$$x_{\max} s \leq \frac{1}{\max_{0 \leq \omega \leq \pi} |H_{wx}(e^{j\omega})|}.$$

- Because $\max\limits_{0 \leq \omega \leq \pi} |H_{wx}(e^{j\omega})| \leq \sum\limits_{m=-\infty}^{\infty} |h_{wx}[m]|$, the scaling factor derived using Criterion #1 is always smaller than the scaling factor derived using Criterion #2.

EXAMPLE: Consider the following simple filter. Assuming that $x_{\max} = 1$, scale the input so that $|y[n]|$ is always less than 1.



- To find the scaling factor using Criterion #1, we need to calculate the summation of the absolute values of the impulse response:

$$s = \frac{1}{\sum_{m=-\infty}^{\infty} |h[m]|} = \frac{1}{\sum_{m=0}^{\infty} |b||a|^m} = \frac{1 - |a|}{|b|}.$$

EXAMPLE: Find the output SNR

- The output noise power is simply $2 \times \Delta^2/12 \times 1/(1-a^2)$.
- To calculate the signal power, assume that $x[n]$ is a white random signal uniformly distributed between 1 and -1.
  - Its mean is 0 and its variance is $1/3$.
  - The signal power at the output of the filter is $\frac{1}{3}s^2\frac{b^2}{1-a^2}$.
- The total SNR is:

$$\frac{1}{3}s^2\frac{b^2}{1-a^2} \times \frac{12(1-a^2)}{2\Delta^2} = \frac{2s^2b^2}{\Delta^2} = \frac{2(1-|a|)^2}{\Delta^2}.$$

6.8: Dynamic range scaling: Application

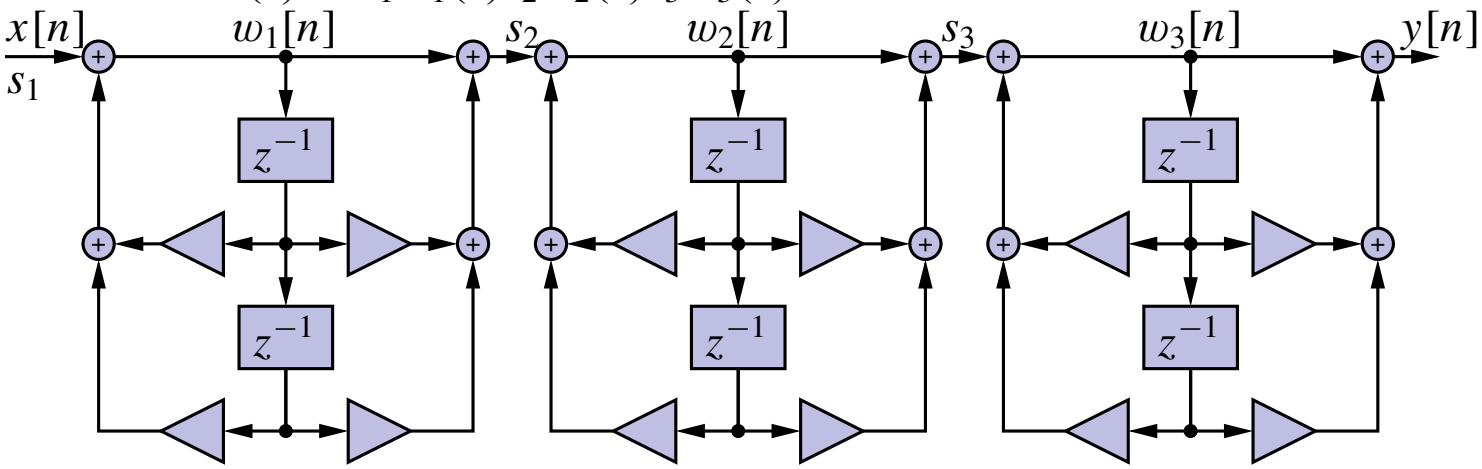
Scaling for parallel and cascade structures

- Scaling strategy for cascade or parallel second-order IIR filters is a trade-off between avoiding overflow and reducing signal-to-quantization noise of the overall system.
- For cascade IIR filters, let's take a look at the following example, a three-stage lowpass filter:

$$H(z) = b_0 \prod_{k=1}^{3} \frac{(1 + b_1 k z^{-1} + b_{2k} z^{-2})}{(1 - a_{1k} z^{-1} - a_{2k} z^{-2})}.$$

- Because there is a small gain constant b_0, the three cascade stages actually produce high gain at some internal nodes to produce an overall gain of unity at dc.
  - If we put the gain constant at the *input* to the filter, the signal power is reduced immediately, while the noise power is to be amplified by the rest of the system.
  - On the other hand, if we put the gain constant at the *end* of the filter, we will have overflow along the cascade because of the high gain introduced internally by the three stages.
- The scaling strategy is to distribute the gain constant along the three stages so that overflow is just avoided at each stage of the cascade.

EXAMPLE: $H(z) = s_1 H_1(z) s_2 H_2(z) s_3 H_3(z)$.



- Noise power,

$$P_f(\omega) = \frac{2^{-2B}}{12}\left[\frac{3s_2^2|H_2(e^{j\omega})|^2 s_3^2|H_3(e^{j\omega})|^2}{|A_1(e^{j\omega})|^2} + \frac{5s_3^2|H_3(e^{j\omega})|^2}{|A_2(e^{j\omega})|^2} + \frac{5}{|A_3(e^{j\omega})|^2} + 3\right]$$

- Note: $1/A_k(e^{j\omega})$ is the transfer function of the feedback from $w_k[n]$ to the output of that filter stage.
- To maximize the total SNR is not a trivial task. Let's first make sure that no overflow can happen at all nodes. Using our Criterion #2:
  - Scaling s_1 according to $s_1 \max_\omega |H_1(e^{j\omega})| < 1$,
  - Scaling s_2 according to $s_1 s_2 \max_\omega |H_1(e^{j\omega}) H_2(e^{j\omega})| < 1$,
  - Scaling s_3 according to $s_1 s_2 s_3 = b_0$.

QUESTION: How do we pair poles and zeros in H_1, H_2, and H_3 to make our stages?

Pole-zero pairing in cascade IIR filters

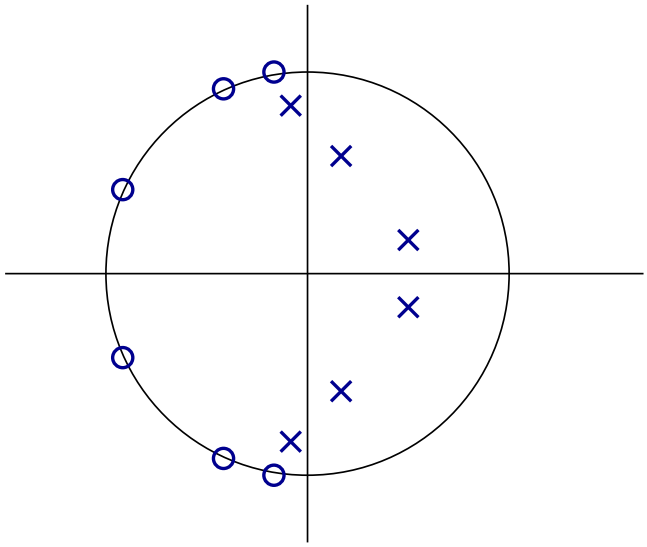
- There are $N!$ ways to pair zeros and poles if there are N cascade stages, which is obviously a very computationally intensive problem.

- Optimal solutions can be found through dynamic programming, but a very good heuristic algorithm proposed by Jackson is usually used.[1]

  1. The pole that is closest to the unit circle should be paired with the zero that is closest to it in the z-plane.
  2. Step 1 should be repeatedly applied until all the poles and zeros have been paired.

- The intuition behind Step 1 is based on the observation that a second-order section with high gain (its poles close to the unit circle) is undesirable because it can cause overflow and amplify quantization noise.

- Pairing a pole that is close to the unit circle with an adjacent zero tends to reduce the peak gain of the section, equalizing the gain along the cascade stages.



[1] Leland Jackson, Digital Filters and Signal Processing, 3d, Kluwer Academic Publishers, 1996.

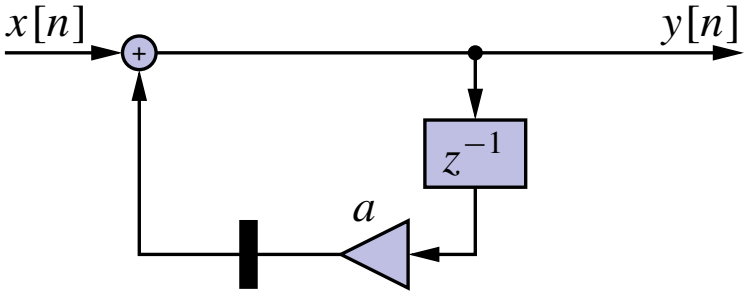
Cascade stage ordering

- Jackson's advice isn't as useful here: "Second-order sections should be ordered according to the closeness of the poles to the unit circle, either in order of increasing closeness to the unit circle or in order of decreasing closeness to the unit circle."
- The reason for the uncertainty is this:
  - A highly peaked filter section (poles close to the unit circle) should be put toward the *beginning* of the cascade so that its transfer function doesn't occur often in the noise power equation.
  - On the other hand, a highly peaked section should be placed toward the *end* of the cascade to avoid excessive reduction of the signal level in the early stages of the cascade (small s values at the beginning of a cascade attenuate signal power but not quantization noise power).
  - Therefore without extensive simulation, the best we can have is a set of "rules of thumb", one of which is stated as above.
- Parallel IIR filters are a bit easier to deal with, because the issue of pairing and ordering does not arise.
  - Jackson concluded that the *total signal-to-quantization noise of the parallel form* is comparable to that of the ***best*** *pairing and ordering of the cascade form*.
  - But cascade form is more commonly used because of its control over zeros as we discussed in coefficient quantization.

6.9: Zero-input limit cycles

- The term "zero-input limit cycle" refers to the phenomenon where a system's output continues to oscillate indefinitely while the input remains equal to zero, caused by the nonlinear characteristic of a quantizer in the feedback loop of the system.
- We will consider the limit-cycle effects for only first-order and second-order filters.

EXAMPLE: Consider the filter $y[n] = ay[n-1] + x[n]$.



- Limit cycles appear when the system behaves **as if** there is a pole on the unit circle. How could this happen?
- Let's take a closer look at quantizing $ay[n-1]$.

$$|Q(ay[n-1]) - ay[n-1]| \leq \frac{1}{2}(2^{-B}).$$

- Therefore,

$$|Q(ay[n-1])| - |ay[n-1]| \leq \frac{1}{2}(2^{-B}).$$

- If $|Q(ay[n-1])| = |y[n-1]|$, this implies

$$|y[n-1]| - |ay[n-1]| \leq \frac{1}{2}(2^{-B}).$$

- In other words, we can define a deadband of:

$$|y[n-1]| \leq \frac{\frac{1}{2}(2^{-B})}{1-|a|}.$$

- Whenever $y[n]$ falls within this deadband when the input is zero, the filter remains in the limit cycle until an input is applied.

EXAMPLE: Suppose that we use a 4-bit quantizer to implement a first-order filter of $a = 1/2$.

- We calculate the deadband size to be

$$\frac{\frac{1}{2}(2^{-3})}{1-\frac{1}{2}} = 2^{-3} = \frac{1}{8}.$$

- If the initial condition $y[-1]$ is equal to 7/8, or 0.111, then:

$$\begin{aligned} ay[-1] &= y[0] = 0.0111 \rightarrow 0.100 \\ ay[0] &= y[1] = 0.0100 \rightarrow 0.010 \\ ay[1] &= y[2] = 0.0010 \rightarrow 0.001 \\ ay[2] &= y[3] = 0.0001 \rightarrow 0.001 \\ ay[3] &= y[4] = 0.0001 \rightarrow 0.001 \end{aligned}$$

QUESTION: What if $a = -1/2$?

ANSWER: The output will oscillate between $1/8$ and $-1/8$.

- For second-order sections, the analysis is a bit more complicated:

$$y[n] = x[n] + Q(a_1 y[n-1]) + Q(a_2 y[n-2]).$$

- Recall that a_1 corresponds to $2r\cos\theta$ and a_2 corresponds to r^2.
- We concentrate on the a_2 term, as it defines the distance of the two complex-conjugate poles to the origin.
- If we have:

$$Q(a_2 y[n-2]) = -y[n-2] \quad \text{and} \quad |Q(a_2 y[n-2]) - a_2 y[n-2]| \leq \frac{1}{2}(2^{-B})$$

then $|y[n-2]| \leq \dfrac{\frac{1}{2}(2^{-B})}{|1+a_2|}$, the deadband for a second-order filter.

- When the input is zero and the output falls within this range, the effective value of a_2 is such that the poles are on the unit circle.

QUESTION: What does a_1 do?

ANSWER: a_1 controls the oscillation frequency.

EXAMPLE: $y[n] = Q(1/2y[n-1]) + Q(-7/8y[n-2])$ with a four-bit quantizer.

- The deadband is equal to $2^{-3} \times \frac{1}{2} \times 8 = \frac{1}{2}$.
- If the initial condition is such that $y[-1] = y[-2] = 1/8$, the output will oscillate between 0, $-1/8$, $-1/8$, 0, $1/8$, $1/8$, etc.
- Adding dither to the input signal can sometimes be effective in eliminating limit cycles. See Franklin.

Where to from here?

- We have now completed the fundamental material of this course.
- We can analyze and design hybrid control systems using traditional means.
- We have seen that some less common methods, such as direct coefficient optimization and Ragazzini's method, can improve on simple lead/lag (and so forth) designs.
- How far can we take this idea? Can we adaptively optimize control-system design?
- That's the topic we look at last in this course.