Chapter 6

Structures for Discrete-Time Systems

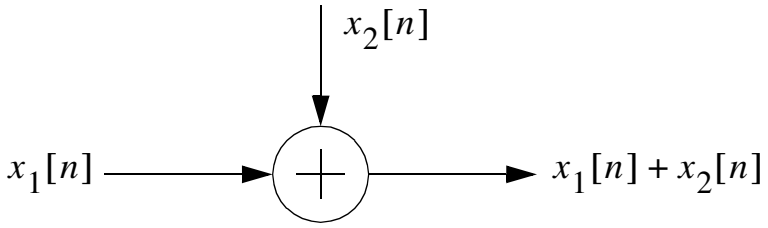
Contents

The actual implementation of an LTI discrete-time system requires that the system function be converted to an algorithm or a computational oriented block diagram. It turns out that LCCDEs can be represented by structures containing addition, multiplication, and unit sample delay blocks. For a given system input/output relationship there are many possible computational structures to consider. Some may be "better" than others depending on the application and overall system design constraints.

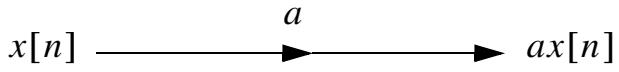
6.1 Block Diagram Representations of LCCDEs

To implement an LTI system which has an LCCDE representation requires evaluation of the recurrence formula discussed in Chapter 2.

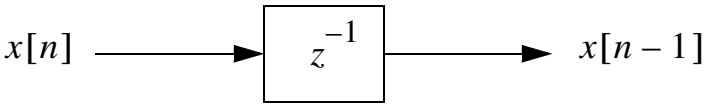
- The basic processing elements needed are:
  - Addition of two sequences



  - Multiplication of a sequence by a constant. A storage element is required for the coefficient



  - Unit sample sequence delay. A storage element is required.



Note: to get a delay of say M samples replace z^{-1} with z^{-M} which now requires M storage elements (e.g. an M stage clocked shift register)

- Consider the N^{th} order rational system function $H(z)$

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

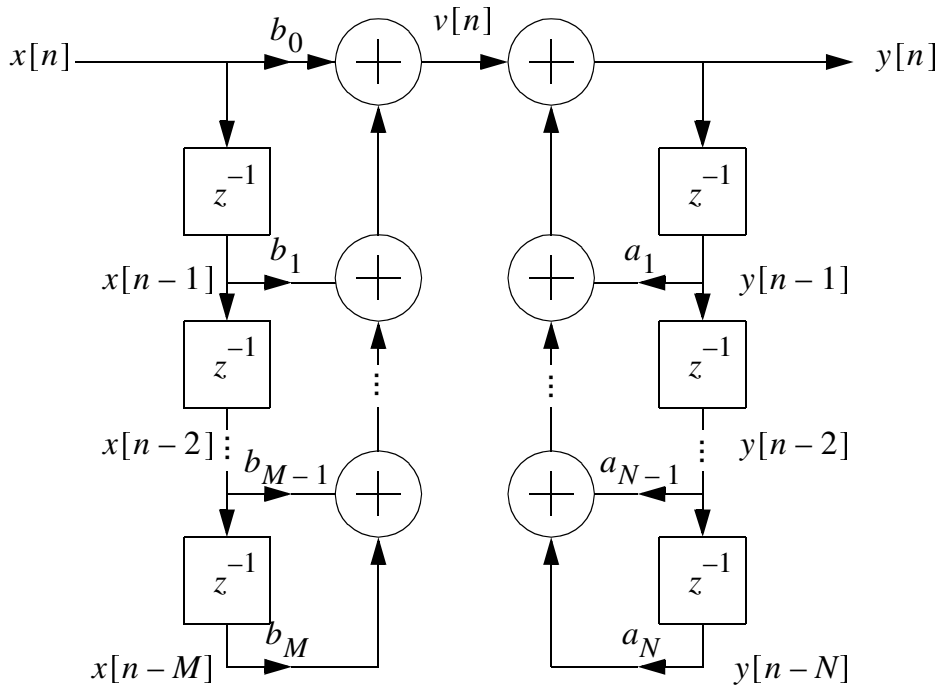
Note: In this chapter we have changed the signs on the a_k coefficients!

- The corresponding difference equation which serves as a recurrence relation for computing $y[n]$ from past values of the output and past and present values of the input is

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$$

where unless stated otherwise we assume $a_0 = 1$

- A block diagram representation of this difference equation using the addition, multiplication, and delay elements is shown below



Nth order difference equation; Direct-Form I

- The block diagram serves as a visual aid in the design of either a software computational algorithm or perhaps a hardware implementation using VLSI technology
- For this particular block diagram implementation we also see that storage for $M + N$ delayed variables must be provided along with storage for $M + 1 + N$ coefficients
- The block diagram can also be viewed as a cascade of two systems, $x[n]$ to $v[n]$ and $v[n]$ to $y[n]$ with respective difference equations

$$v[n] = \sum_{k=0}^{M} b_k x[n-k]$$

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + v[n]$$

- Each of the systems is LTI (assuming initial rest conditions), thus the cascade order may be reversed without altering the overall system function

- Define the following system functions

$$H_1(z) = \frac{V(z)}{X(z)} = \sum_{k=0}^{M} b_k z^{-k}$$
$$H_2(z) = \frac{Y(z)}{V(z)} = \frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

- Since

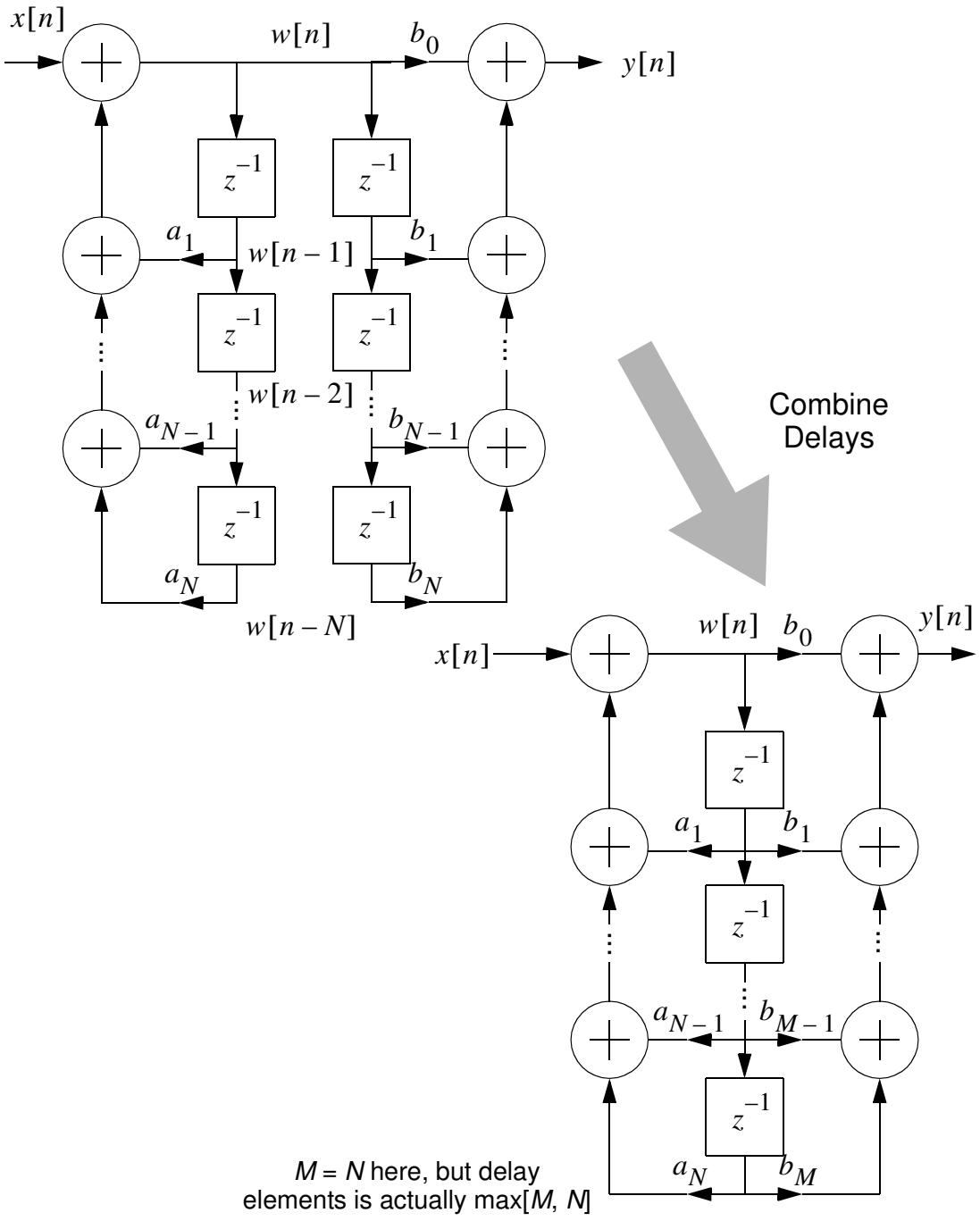
$$H(z) = H_1(z)H_2(z) = H_2(z)H_1(z)$$

  an equivalent block diagram representation for a general Nth order LCCDE can be obtained from

$$W(z) = H_2(z)X(z) = \left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right) X(z)$$
$$Y(z) = H_1(z)W(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right) W(z)$$

- The difference equations are now

$$w[n] = \sum_{k=1}^{N} a_k w[n-k] + x[n]$$
$$y[n] = \sum_{k=0}^{M} b_k w[n-k]$$



Rearranged block diagram with $M = N$ for convenience; with combined delays we have Direct Form II

- Note that the rearrangement of the original block diagram as shown above (or any appropriate rearrangement for that matter) will in general result in a different computational algorithm

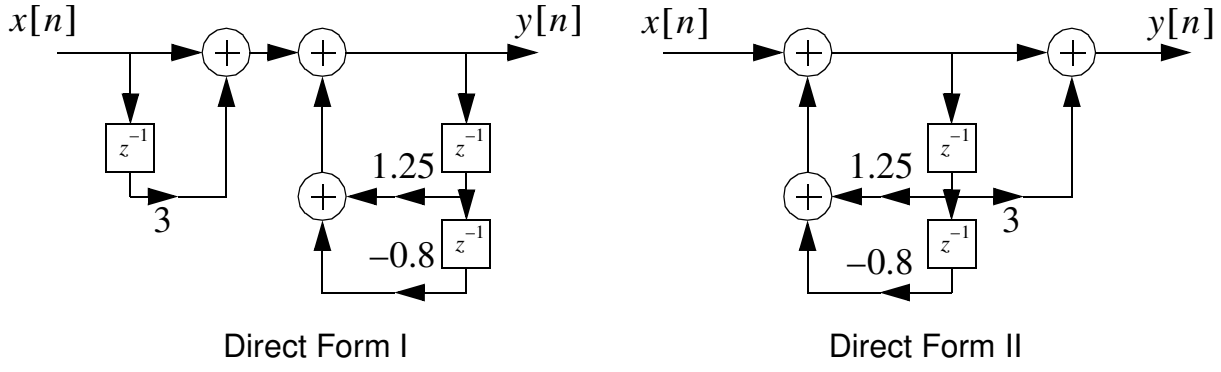
- The $H_1(z)H_2(z)$ representation implements the zeros first followed by the poles
- The $H_2(z)H_1(z)$ representation on the other hand implements the poles first followed by the zeros
- Assuming infinite precision arithmetic both systems are the same, but with finite precision arithmetic significant differences can result
- The $H_1(z)H_2(z)$ implementation which requires $M+N$ delay elements is known as a *direct form I* implementation
- The $H_2(z)H_1(z)$ implementation which requires in general only $\max(N, M)$ delay elements is known as a *canonic form* or *direct form II* implementation

---

Example 6.1: **Direct Form I and II Realizations** Consider

$$H(z) = \frac{1 + 3z^{-1}}{1 - 1.25z^{-1} + .8z^{-2}}$$

- The direct form I and II realizations are shown below

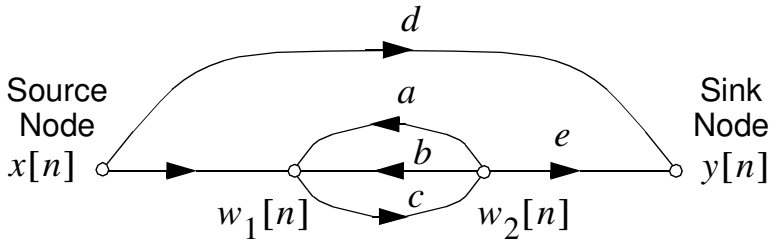


A direct form I and II realization for $N = M = 2$

6.2 SFG Representations of LCCDEs

A signal flow graph (SFG) representation of an LCCDE is really no different than a block diagram representation except (1) an SFG is more compact than a block diagram (2) there exists a large body of SFG theory with direct application to discrete-time systems.

- An SFG is a network of directed branches connected at nodes
  - Each node has an associated node variable or sequence say $w_k[n]$



SFG example

  - In linear SFGs each branch performs a linear transformation on the input node variable to obtain the output node variable
  - A simple branch transformation is multiplication by a constant (e.g., a, b, c, d, and e as shown above). If no constant multiplier is indicated next to the branch arrowhead then unity is assumed.
  - A unit delay branch (element) is represented by z^{-1}
  - In general each node value is just the sum of the outputs of the branches entering that node
  - A node which has no entering branches is called the *source node* (i.e. a node where a signal may be injected)

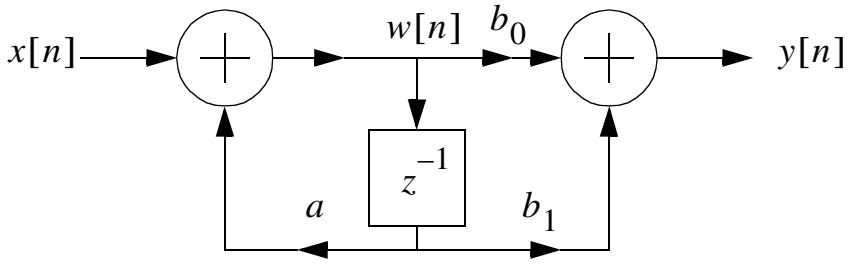
– A node with only entering branches is called a *sink node* (i.e. a node where a signal may be extracted)

Example 6.2: A First-Order System SFG

Consider a first order difference equation with system function

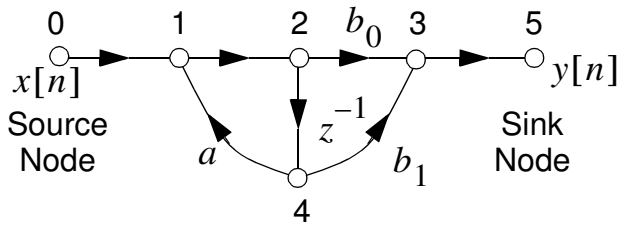
$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a z^{-1}}$$

and block diagram representation (direct form II) as shown below

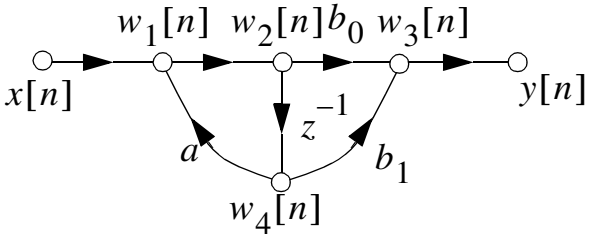


First order system direct form II block diagram

- The corresponding SFG is shown below



SFG with Numbered Nodes



SFG with Intermediate Node Signals

First order system SFG (a) With node variables labeled (b) with $w[n]$ denoted

- Note that the source and sink nodes are connected to the main body of the SFG by unity gain branches to clearly identify the input and output

- To write the system difference equation from the SFG we can write node variable equations in the time-domain or using z-transform representations of all the node variables

$$\begin{aligned} w_1[n] &= aw_4[n] + x[n] \\ w_2[n] &= w_1[n] \\ w_3[n] &= b_0 w_2[n] + b_1 w_4[n] \\ w_4[n] &= w_2[n-1] \\ y[n] &= w_3[n] \end{aligned}$$

- From the above list of equations we see that to calculate $y[n]$ from $x[n]$ requires an ordered sequence of calculations

- For the first order system $y[n]$ can be related to $x[n]$ by a single equation by making appropriate substitutions e.g.,

$$\begin{aligned} w_2[n] = w_1[n] &= aw_4[n] + x[n] \\ &= aw_2[n-1] + x[n] \\ y[n] = w_3[n] &= b_0 w_2[n] + b_1 w_2[n-1] \end{aligned}$$

Now what? Switch to the z-domain and write

$$\begin{aligned} W_2(z) &= az^{-1}W_2(z) + X(z) \\ W_2(z) &= \frac{1}{1-az^{-1}}X(z) \\ Y(z) &= b_0 W_2(z) + b_1 z^{-1} W_2(z) \end{aligned}$$

so

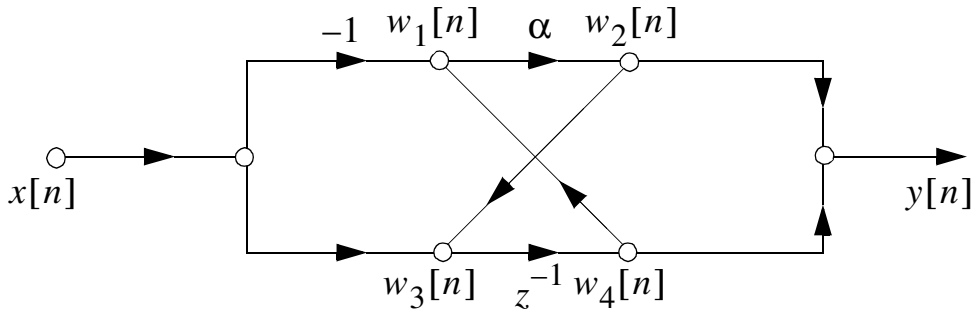
$$Y(z) = \frac{b_0 + b_1 z^{-1}}{1-az^{-1}}X(z)$$

and inverse transforming yields

$$y[n] = ay[n-1] + b_0 x[n] + b_1 x[n-1]$$

Example 6.3: Finding $H(z)$ for a Non-Standard SFG

Consider the following SFG



- We begin by writing the following node variable equations

$$\begin{aligned} w_1[n] &= w_4[n] - x[n] \\ w_2[n] &= \alpha w_1[n] \\ w_3[n] &= w_2[n] + x[n] \\ w_4[n] &= w_3[n-1] \\ y[n] &= w_2[n] + w_4[n] \end{aligned}$$

- As in the previous example we proceed by first converting to z-domain and ultimately solve for $Y(z)/X(z) = H(z)$

$$\begin{aligned} W_1(z) &= W_4(z) - X(z) \\ W_2(z) &= \alpha W_1[z] \\ W_3(z) &= W_2(z) + X(z) \\ W_4(z) &= z^{-1} W_3(z) \\ Y(z) &= W_2(z) + W_4(z) \end{aligned}$$

- We now progressively eliminate interior node variables

$$W_2(z) = \alpha\big(W_4(z) - X(z)\big)$$
$$W_4(z) = z^{-1}\Big(\alpha\big(W_4(z) - X(z)\big) + X(z)\Big)$$
$$Y(z) = \alpha\big(W_4(z) - X(z)\big) + W_4(z)$$

- Solve for $W_4(z)$ in terms of just $X(z)$

$$W_4(z) = z^{-1}\Big(\alpha\big(W_4(z) - X(z)\big) + X(z)\Big)$$
$$W_4(z)\big[1 - \alpha z^{-1}\big] = X(z)(1-\alpha)z^{-1}$$
$$\text{so} \quad W_4(z) = X(z)\frac{(1-\alpha)z^{-1}}{1-\alpha z^{-1}}$$

- Solve for $W_2(z)$ in terms of just $X(z)$

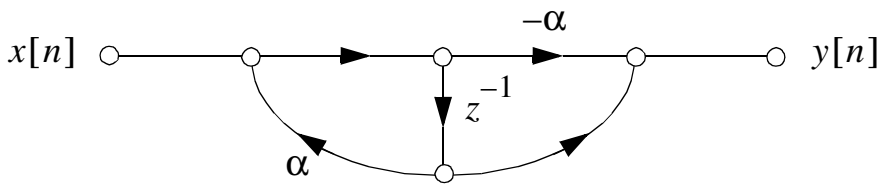
$$W_2(z) = \alpha\big[W_4(z) - X(z)\big]$$
$$= \alpha X(z)\left[\frac{(1-\alpha)z^{-1}}{1-\alpha z^{-1}} - 1\right]$$
$$= \alpha X(z)\frac{(z^{-1}-1)}{1-\alpha z^{-1}}$$

- Finally we can solve for $Y(z)$ in terms of just $X(z)$, and also $H(z)$

$$Y(z) = X(z)\left[\frac{\alpha(z^{-1}-1)}{1-\alpha z^{-1}} + \frac{(1-\alpha)z^{-1}}{1-\alpha z^{-1}}\right]$$
$$= X(z)\frac{z^{-1}-\alpha}{1-\alpha z^{-1}}$$

  which implies that

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



Direct-Form II equivalent SFG

6.3 Basic Structures for IIR Systems

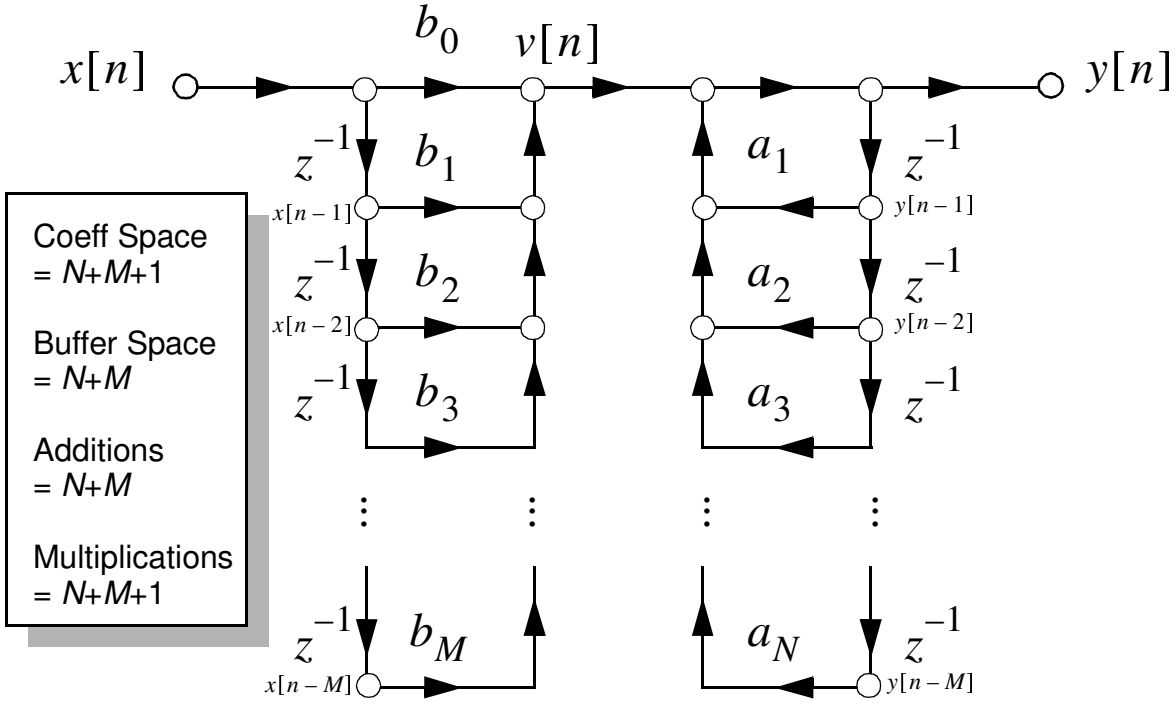
In this section we will consider several SFG representations for Nth order LCCDEs. Assuming that the corresponding system functions have non-zero poles, then the systems in general will be IIR. Important issues are:

- Computational complexity
  - Number of constant multipliers
  - Number of storage registers
- VLSI implementation tradeoffs
  - Chip area
  - Modularity
  - Data transfer simplicity
- Multiprocessor Implementations
  - Algorithm partitioning
  - Communication requirements between processors

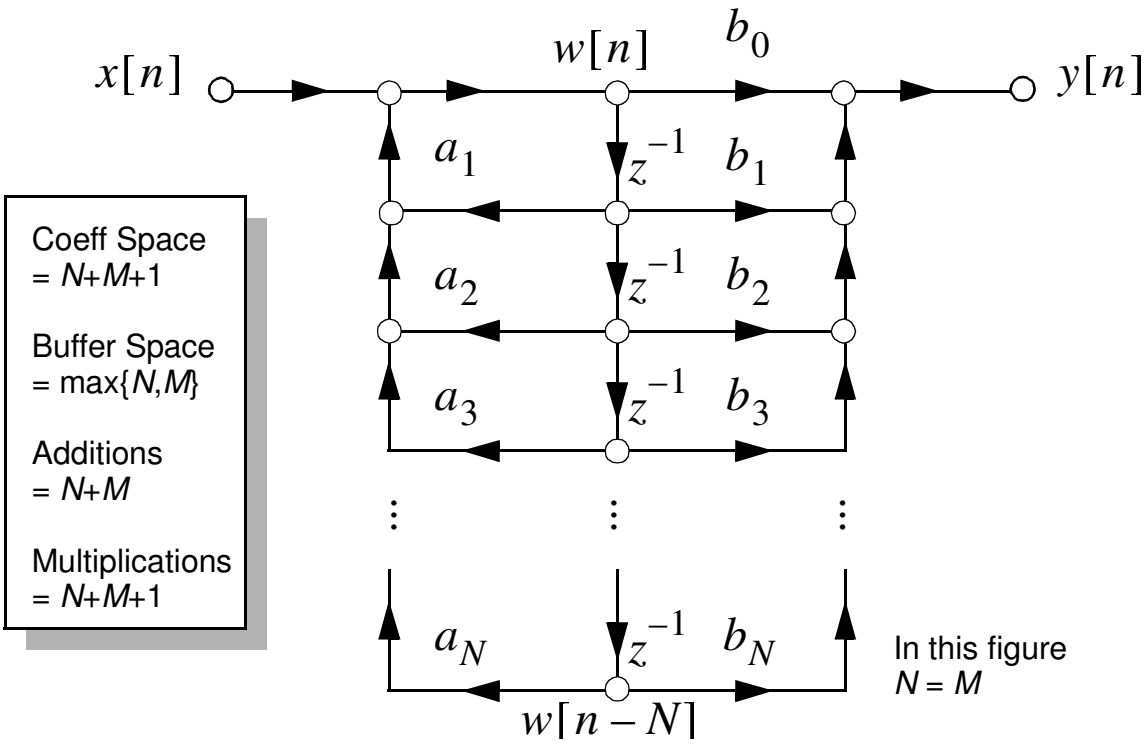
- Finite register length effects
  - Coefficient quantization
  - Product quantization
  - A network with the minimum number of multipliers and delay elements may not be the least sensitive to finite-precision arithmetic

6.3.1 Direct Form Structures

The SFG representation of the direct form I and direct form II structures are shown below (for convenience only $M = N$). In both figures each node has no more than two inputs. This is done to more closely represent practical implementation considerations in the SFG



Direct form I SFG for an Nth order system



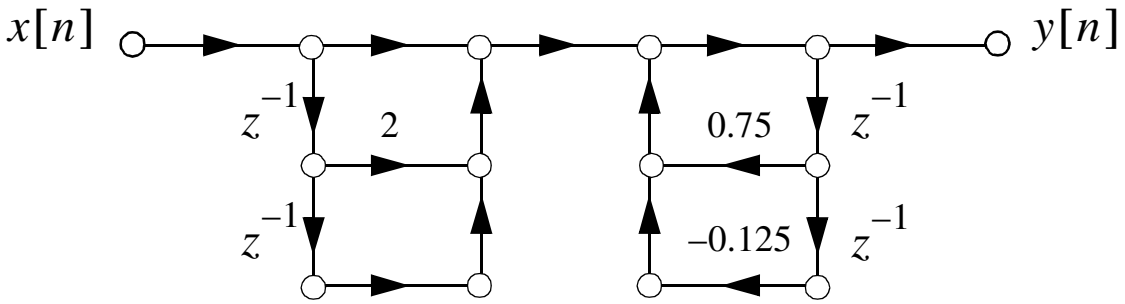
Direct form II SFG for an Nth order system

Example 6.4: A Second-Order System

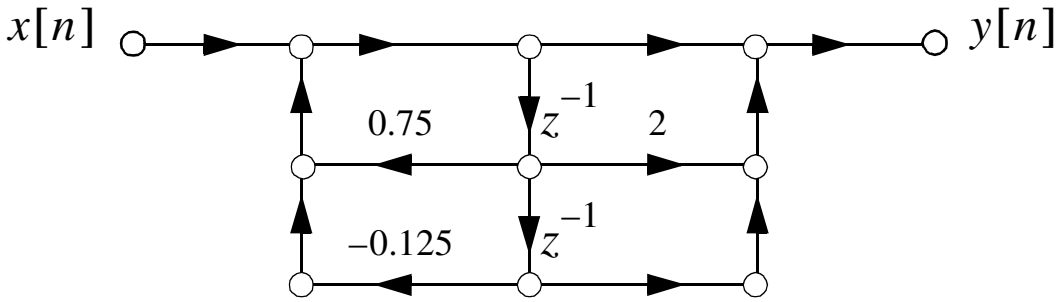
Consider the following second-order system

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}}$$

- The direct form I and direct form II can be obtained directly from the coefficients of the numerator and denominator polynomials of $H(z)$



Direct Form I SFG



Direct Form II SFG

Direct form I and II SFGs

6.3.2 Cascade Form

Assuming real coefficients factor the numerator and denominator polynomials of $H(z)$ into first and second-order polynomials

$$H(z) = A\frac{\prod_{k=1}^{M_1}(1 - f_k z^{-1}) \prod_{k=1}^{M_2}(1 - g_k z^{-1})(1 - g_k^* z^{-1})}{\prod_{k=1}^{N_1}(1 - c_k z^{-1}) \prod_{k=1}^{N_2}(1 - d_k z^{-1})(1 - d_k^* z^{-1})}$$

where $M = M_1 + 2M_2$ and $N = N_1 + 2N_2$.

- The first-order factors are used to represent the real zeros, f_k, and real poles, c_k, of $H(z)$

- The second-order factors are used to represent the complex conjugate zero pairs, g_k and g_k^*, and complex conjugate pole pairs, d_k and d_k^*

- The above formulation suggests that $H(z)$ can be represented as a cascade of first and second order sections

- It is common practice to group first-order factors (poles and zeros) into pairs, thus forming second-order factors. Many different groupings of poles and zeros are possible

- The standard cascade section is a second-order factor or *biquad* section

- Using biquad sections we can then write

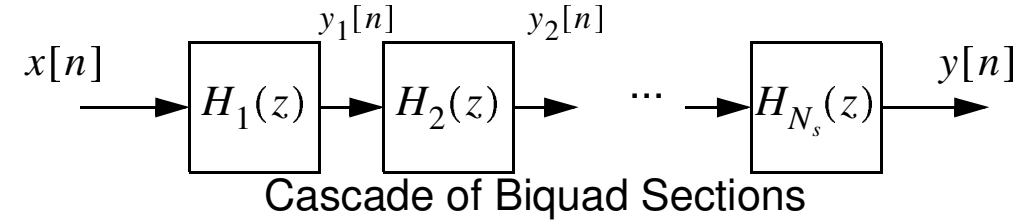
$$H(z) = \prod_{k=1}^{N_s} \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 - a_{1k}z^{-1} - a_{2k}z^{-2}}$$

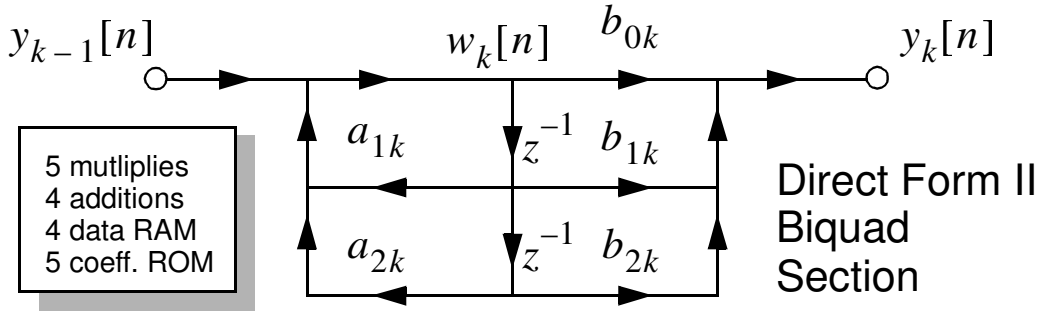
  where $N_s = \lfloor (N+1)/2 \rfloor$, which is the largest integer in $(N+1)/2$

- If the number of real zeros (poles) is odd, then in a particular section b_{2k} (a_{2k}) is zero

- Each biquad section can be implemented using a direct form II structure, thus requiring at most five constant multipliers and two delay registers

- There are $(N_s!)^2$ different pairings and orderings of N_s second-order sections





General cascade structure using direct form II sections

- The difference equations required for a cascade of direct form II sections are:

$$y_0[n] = x[n]$$
$$w_k[n] = a_{1k} w_k[n-1] + a_{2k} w_k[n-2] + y_{k-1}[n],\ k = 1, \ldots, N_s$$
$$y_k[n] = b_{0k} w_k[n] + b_{1k} w_k[n-1] + b_{2k} w_k[n-2]$$

- A simplified biquad section can be formed by factoring b_{0k} out of each section as follows

$$H(z) = b_0 \prod_{k=1}^{N_s} \frac{1 + \tilde{b}_{1k} z^{-1} + \tilde{b}_{2k} z^{-2}}{1 - a_{1k} z^{-1} - a_{2k} z^{-2}}$$

  where b_0 is the overall gain constant and $\tilde{b}_{ik} = b_{ik}/b_{0k}$

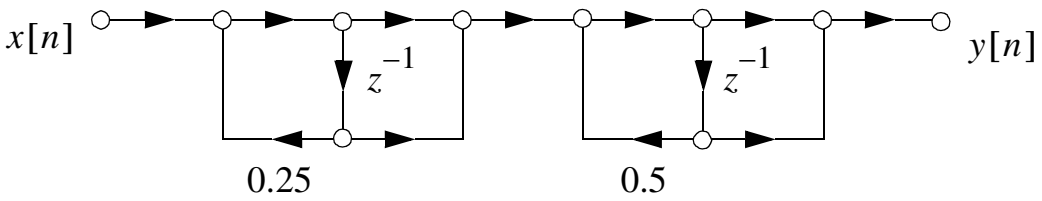
- The number of multipliers per section is reduced from five to four, but for fixed-point arithmetic the loss of stage-by-stage scaling is unacceptable

- This approach is best for floating point implementations

Example 6.5: **Second-order System cont.**

A previous example found the direct form I and direct form II SFGs for a particular second-order system. A cascade realization of this system is a single biquad section. Alternatively the biquad section can be broken down into a cascade of two first-order sections with real coefficients only if the poles and zeros are real.

$$\begin{aligned} H(z) &= \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}} \\ &= \frac{(1 + z^{-1})(1 + z^{-1})}{(1 - 0.5z^{-1})(1 - 0.25z^{-1})} \end{aligned}$$



Cascade structure using first-order direct form II sections

6.3.3 Parallel Form

A parallel structure implementation of $H(z)$ can be obtained by using a partial fraction expansion, i.e.,

$$H(z) = \sum_{k=0}^{N_p} C_k z^{-k} + \sum_{k=1}^{N_1} \frac{A_k}{1 - c_k z^{-1}} + \sum_{k=1}^{N_2} \frac{B_k(1 - e_k z^{-1})}{(1 - d_k z^{-1})(1 - d_k^* z^{-1})}$$

where $N = N_1 + 2N_2$. If $M \geq N$, then $N_p = M - N$, otherwise the first sum is omitted.

- If the coefficients a_k and b_k in the direct form representation of $H(z)$ are real, then all the coefficients exclusive of d_k are real in the partial fraction expansion

- The parallel structure consists of three types of parallel subsections:
  - Scale and delay branches
  - First-order pole sections
  - First-order zero second-order conjugate pole-pair sections

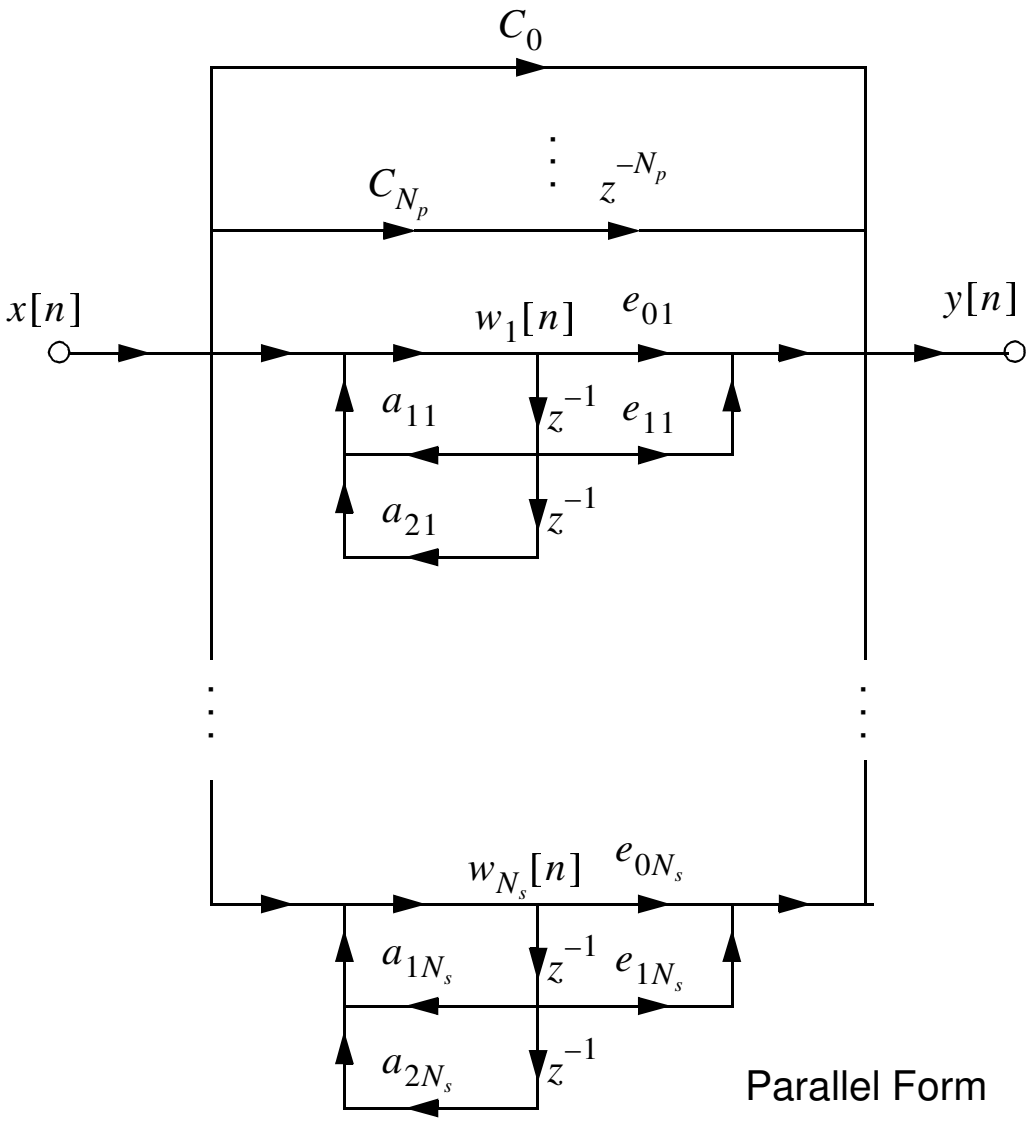
- If we wish we may group pairs of real poles into second-order sections as well

$$H(z) = \sum_{k=0}^{N_p} C_k z^{-k} + \sum_{k=1}^{N_s} \frac{e_{0k} + e_{1k} z^{-1}}{1 - a_{1k} z^{-1} - a_2 z^{-2}}$$

where here $N_s = \lfloor (N+1)/2 \rfloor$

- The difference equations required for each section are

$$w_k = a_{1k} w_k[n-1] + a_{2k} w_k[n-2] + x[n], \quad k = 1, \ldots, N_s$$
$$y_k[n] = e_{0k} w_k[n] + e_{1k} w_k[n-1], \quad k = 1, \ldots, N_s$$
$$y[n] = \sum_{k=0}^{N_p} C_k x[n-k] + \sum_{k=1}^{N_s} y_k[n]$$

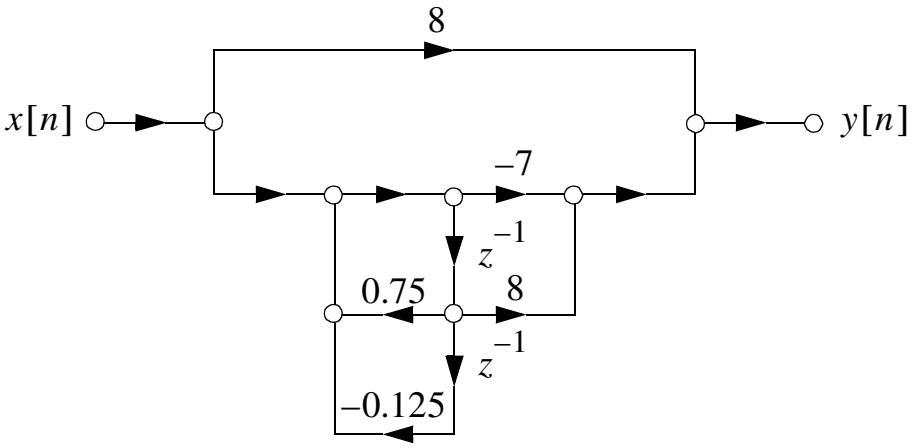


General parallel structure using direct form II sections

Example 6.6: Second-order System cont.

Continuing the previous example involving a second-order system, the partial fraction expansion of $H(z)$ yields

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.75z^{-1} + 0.125z^{-2}} = 8 + \frac{-7 + 8z^{-1}}{1 - 0.75z^{-1} + 0.125z^{-2}}$$



Parallel structure using a second-order section

- Since the poles are real we may decompose the second-order term into a sum of two first-order terms i.e.

$$H(z) = 8 + \frac{18}{1 - 0.5z^{-1}} - \frac{25}{1 - 0.25z^{-1}}$$

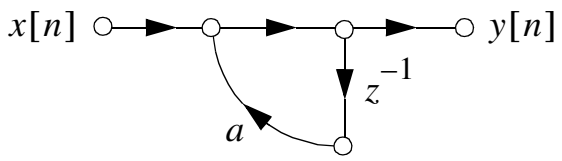
6.3.4 Feedback Loops in IIR & FIR Systems

An SFG with a feedback loop has a node variable that depends on itself directly or indirectly.

IIR Systems:

To create an infinitely long impulse response a feedback loop is required. As an example consider the first order system

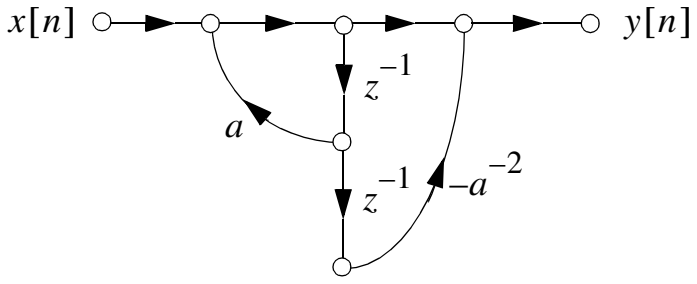
$$y[n] = ay[n-1] + x[n]$$



- An input sample continually recirculates in the feedback loop, decreasing if $|a| < 1$ and increasing if $|a| > 1$
- The impulse response $h[n] = a^n u[n]$ is infinitely long

FIR Systems:

An SFG with feedback loops **may not** be IIR. For example consider the SFG shown below

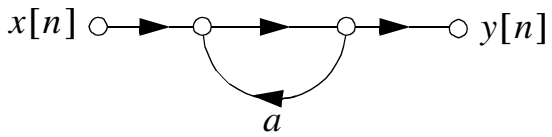


- If we write out $H(z)$ from the SFG we obtain

$$\begin{aligned} H(z) &= \frac{1 - a^2 z^{-2}}{1 - az^{-1}} = \frac{(1 - az^{-1})(1 + az^{-1})}{1 - az^{-1}} \\ &= 1 + az^{-1} \text{ due to pole zero cancellation} \end{aligned}$$

Noncomputable Systems:

When loops are present in an SFG we need to insure that the appropriate node variables become available when needed. For a certain class of SFGs called *noncomputable* there is no way to order the node computations to avoid having a noncompuable node variable. As an example consider the simple network shown below



- The network difference equation is

$$y[n] = ay[n] + x[n]$$

- The difference equation can be solved by simply writing

$$y[n] = \frac{x[n]}{1-a},$$

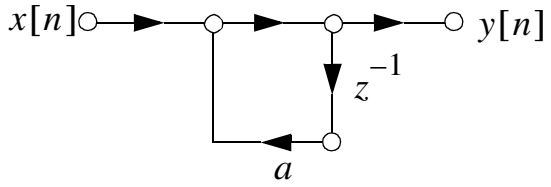
  but a different SFG is required

- In general to insure computability all loops must contain at least one delay element

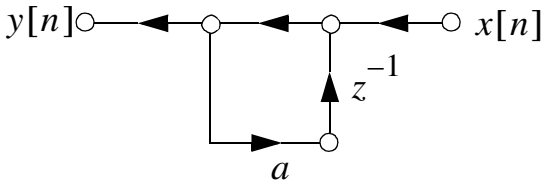
6.4 Transposed SFGs

A fundamental property of linear SFGs is that *transposition* or *flow graph reversal* does not alter the system function. To transpose an SFG all branch directions are reversed and sink nodes become source nodes and source nodes become sink nodes (i.e. $x[n] \rightarrow y[n]$ and $y[n] \rightarrow x[n]$). A general proof of this property using state variable techniques can be found in Jackson.

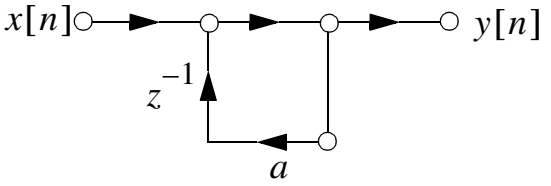
- A simple example which verifies the property is the SFG of a first order system shown below



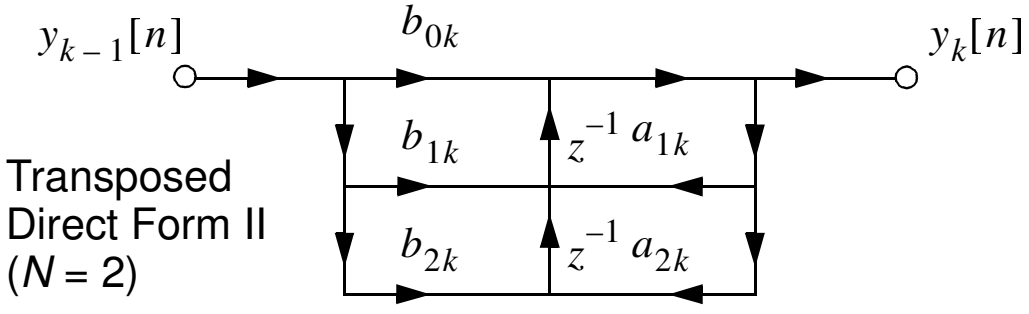
- The transposed network is the following



or flipped over



- A useful structure is the transposed direct form II structure



Transposed direct form II structure for $M = N = 2$

- Note that the transposed direct form II structure implements the zeros first followed by the poles, similar to the direct form I structure except only $\max(M, N)$ delays are required
- The MATLAB function `filter()` is implemented using a transposed direct form II structure

6.5 Structures for FIR Systems

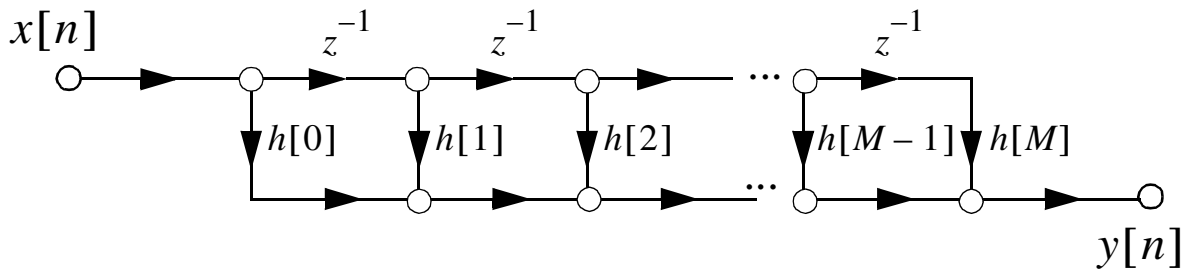
The direct form, cascade, and parallel form structures were developed assuming that $H(z)$ contained both poles and zeros (poles not at $z = 0$). FIR systems for which $H(z)$ contains only zeros (except poles at $z = 0$) can be implemented using similar structures, but without the feedback loops.

6.5.1 Direct Form

The direct form realization of the FIR difference equation

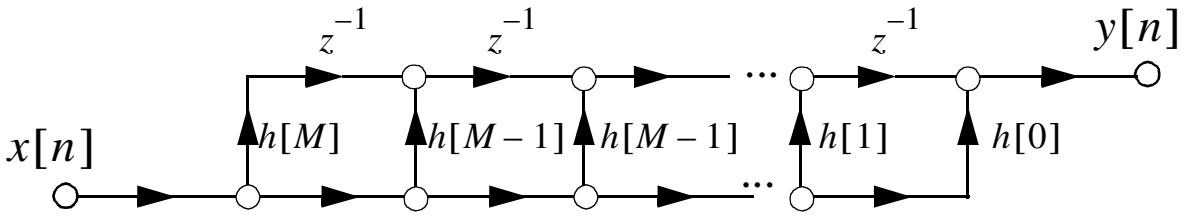
$$y[n] = \sum_{k=0}^{M} b_k x[n-k] = \sum_{k=0}^{M} h[k]x[n-k]$$

is the following



Direct form FIR structure

- The direct form FIR structure is also known as a *transversal filter* or *tapped delay line*
- The transposed network results in the following structure



Transposed direct form FIR structure

6.5.2 Linear Phase FIR Structures

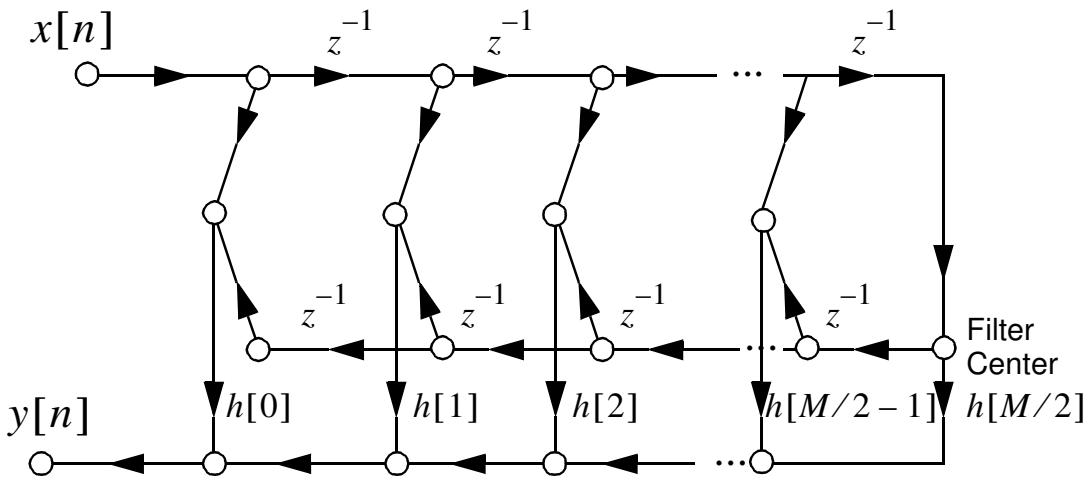
For causal linear phase FIR filters, designed with either a symmetric or antisymmetric impulse response, we can reduce the number of coefficient multipliers by about one half over the direct form structure.

- For type I or type III systems we can write

$$\begin{aligned} y[n] &= \sum_{k=0}^{M} h[k]x[n-k] \\ &= \sum_{k=0}^{M/2-1} h[k](x[n-k] \pm x[n-M+k]) \\ &\quad + h[M/2]x[n-M/2] \end{aligned}$$

  where $h[M/2] = 0$ for the type III system

- The above mathematical form suggests the following direct form structure

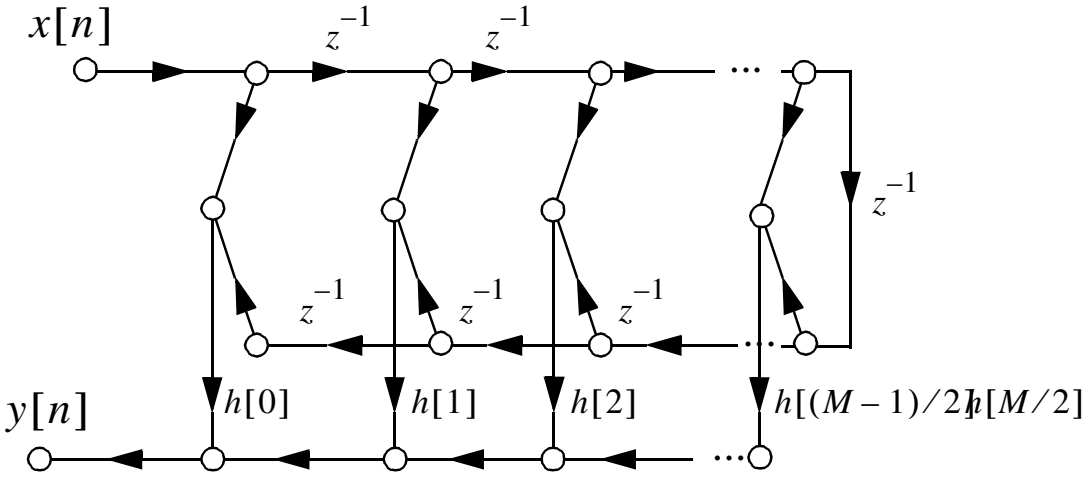


Direct form FIR structure for type I or III (-1) — M even

- For type II or type IV systems we can write

$$y[n] = \sum_{k=0}^{(M-1)/2} h[k](x[n-k] \pm x[n-M+k])$$

- The above mathematical form suggests the following direct form structure



Direct form structure for type II or IV (-1) — M odd

- Since the zeros may occur in quadruplets, pairs, or singly, a high-order linear phase FIR system may be implemented as a cascade of first-, second-,and fourth-order direct form FIR linear phase systems

Example 6.7: Using `iir_design_helper.py` to Design Cascade IIR Filters

```
import sk_dsp_comm.iir_design_helper as iir_d
```

```
b_b,a_b,sos_b = iir_d.IIR_lpf(1500,2500,1,70,10000,'butter')
b_c1,a_c1,sos_c1 = iir_d.IIR_lpf(1500,2500,1,70,10000,'cheby1')
b_c2,a_c2,sos_c2 = iir_d.IIR_lpf(1500,2500,1,70,10000,'cheby2')
b_e,a_e,sos_e = iir_d.IIR_lpf(1500,2500,1,70,10000,'ellip')
```

Second-Order Sections (SOS)

The SOS matrix (2D `ndarray`) has six columns. The first three are the numerator coefficients `b` and the last three are the denominator coefficients, `a`.

```
roots(sos_c1[0,:3])
```

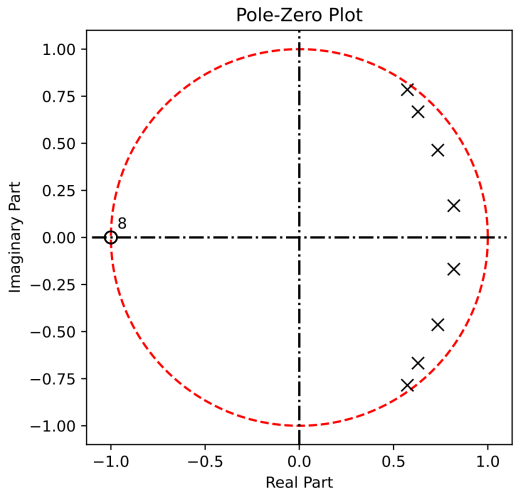
```
array([-1., -1.])
```

```
sos_c1
```

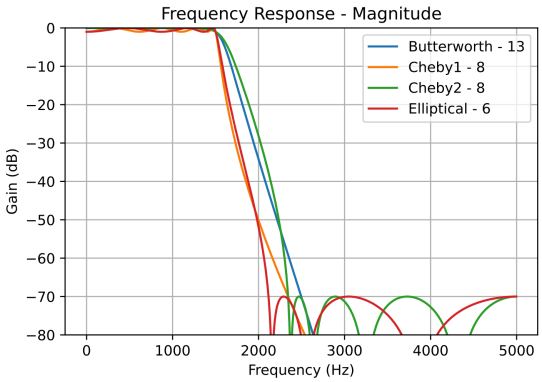
```
array([[ 2.81418610e-05,  5.62837220e-05,  2.81418610e-05,
         1.00000000e+00, -1.63955493e+00,  7.00480335e-01],
       [ 1.00000000e+00,  2.00000000e+00,  1.00000000e+00,
         1.00000000e+00, -1.46953427e+00,  7.54870129e-01],
       [ 1.00000000e+00,  2.00000000e+00,  1.00000000e+00,
         1.00000000e+00, -1.25970664e+00,  8.42420778e-01],
       [ 1.00000000e+00,  2.00000000e+00,  1.00000000e+00,
         1.00000000e+00, -1.14688834e+00,  9.44850837e-01]])
```

```
iir_d.sos_zplane(sos_c1)
```

```
(8, 8)
```



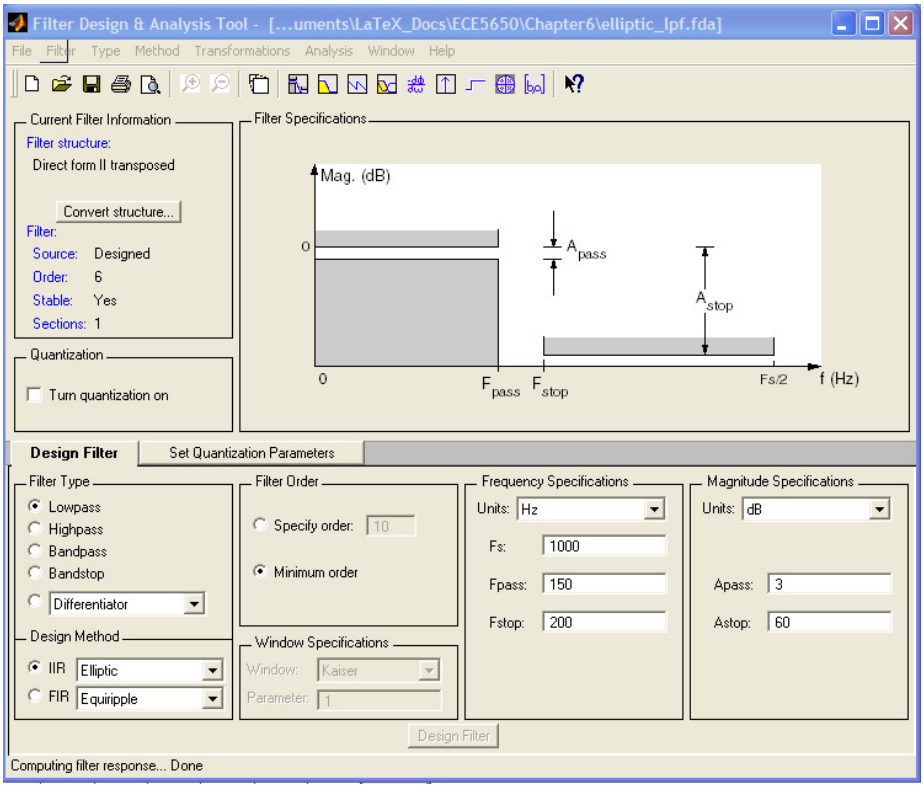

```
iir_d.freqz_resp_cas_list([sos_b,sos_c1,sos_c2,sos_e],'dB',10000)
ylim([-80,0])
legend((r'Butterworth - 13',r'Cheby1 - 8',r'Cheby2 - 8',r'Elliptical -␣
 ↪6'),loc='best')
grid();
```



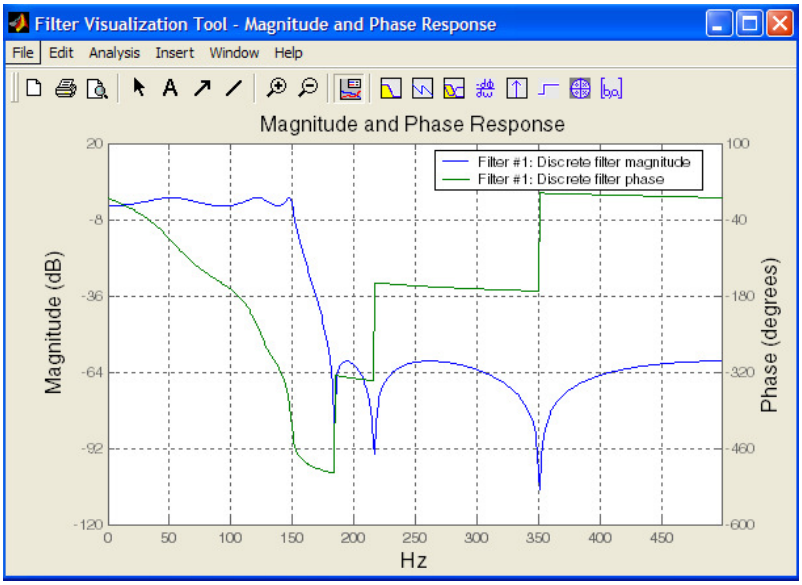
Example 6.8: Using MATLAB to Design Cascade IIR Filters

With the MATLAB *signal processing toolbox* a complete filter design environment is available within the application `sptool` and `fdatool`. Filter design can also be accomplished using just the command line tools as well. To enhance the basic capabilities of the signal processing toolbox a filter design toolbox, which includes the ability to model quantization effects, is now available.

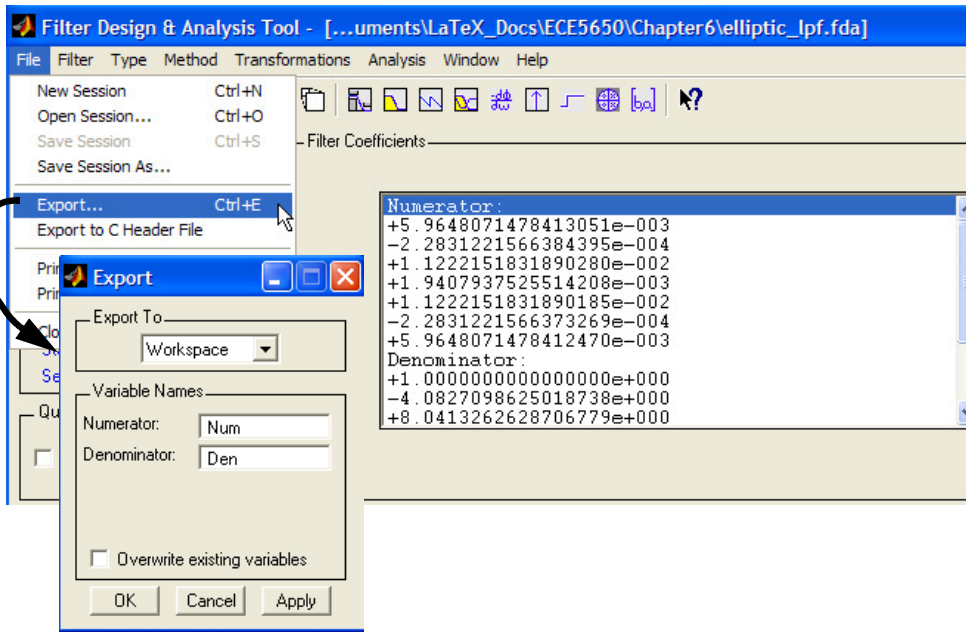
- Consider an elliptic lowpass design in `fdatool`:



- Once the filter design attributes are selected the filter is designed and then various responses can be plotted



- The above filter was designed as a direct form II structure, so the corresponding coefficient set can be exported for use elsewhere



- To save the filter design go to the `fdatool` file menu and choose Export; workspace, text file, and MAT-file are available options

- It is also possible to save the filter coefficients to a C header file, which is useful in real-time DSP applications
  - For real-time DSP quantization would likely be included, which requires the *filter design toolbox* in addition to just the signal processing toolbox
- Many filter structures are available in `fdatool`
- The transfer function form, which is really suitable for either direct form I or II structures, can also be converted to cascade form, we use the function `tf2sos()`

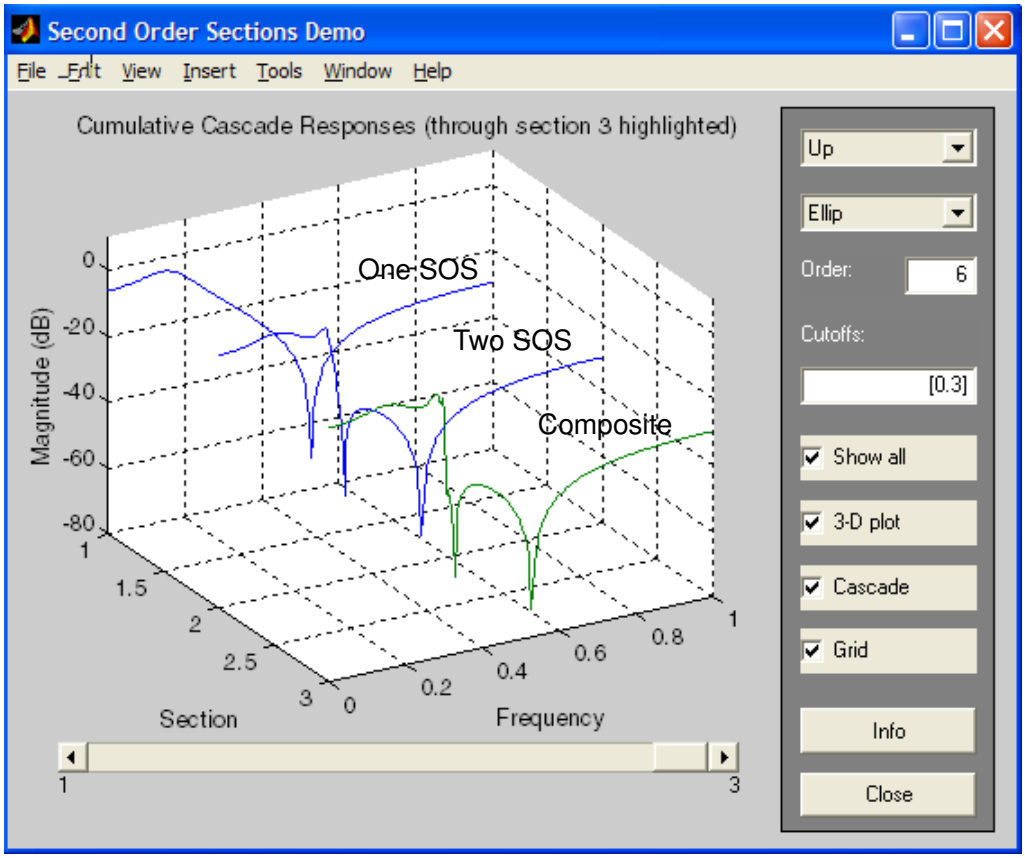
```
? help tf2sos

 TF2SOS Transfer Function to Second Order Section conversion.
    [SOS,G] = TF2SOS(B,A) finds a matrix SOS in second-order sections
    form and a gain G which represent the same system H(z) as the one
    with numerator B and denominator A.  The poles and zeros of H(z)
    must be in complex conjugate pairs. Because of the scaling done,
    all poles must be inside the unit circle, i.e, the system must be
    stable.

    SOS is an L by 6 matrix with the following structure:
        SOS = [ b01 b11 b21  1 a11 a21
                b02 b12 b22  1 a12 a22
                ...
                b0L b1L b2L  1 a1L a2L ]

    Each row of the SOS matrix describes a 2nd order transfer function:
                  b0k +  b1k z^-1 +  b2k  z^-2
        Hk(z) =  ---------------------------
                   1 +  a1k z^-1 +  a2k  z^-2
    where k is the row index.

    G is a scalar which accounts for the overall gain of the system. If
    G is not specified, the gain is embedded in the first section.
    The second order structure thus describes the system H(z) as:
        H(z) = G*H1(z)*H2(z)*...*HL(z)

    TF2SOS(B,A,DIR_FLAG) specifies the ordering of the 2nd order
```

```
    sections. If DIR_FLAG is equal to 'UP', the first row will contain
    the poles closest to the origin, and the last row will contain the
    poles closest to the unit circle. If DIR_FLAG is equal to 'DOWN', the
    sections are ordered in the opposite direction. The zeros are always
    paired with the poles closest to them. DIR_FLAG defaults to 'UP'.

    TF2SOS(B,A,DIR_FLAG,SCALE) specifies the desired scaling of the gain
    and the numerator coefficients of all 2nd order sections. SCALE can be
    either 'NONE', 'INF' or 'TWO' which correspond to no scaling, infinity
    norm scaling and 2-norm scaling respectively. SCALE defaults to 'NONE'.
    Using infinity-norm scaling in conjunction with 'UP' ordering will
    minimize the probability of overflow in the realization. On the other
    hand, using 2-norm scaling in conjunction with 'DOWN' ordering will
    minimize the peak round-off noise.

    See also ZP2SOS, SOS2ZP, SOS2TF, SOS2SS, SS2SOS, CPLXPAIR.
```

- Note that coefficient definitions have a sign difference from the text and notes Chapter 6
- A MATLAB listing of the coefficients are given below

```
>> % Display Direct Form Coefficients
>> [filt1.tf.num' filt1.tf.den']
>> ans =
     5.9648e-003  1.0000e+000
    -2.2831e-004 -4.0827e+000
     1.1222e-002  8.0413e+000
     1.9408e-003 -9.4022e+000
      .1222e-002  6.8379e+000
    -2.2831e-004 -2.9302e+000
      .9648e-003   5.8651e-001
>> % Display Cascade Form Coefficients
>> [ellip_sos, G] = tf2sos(filt1.tf.num,filt1.tf.den);
>> G
>>    G = 5.9648e-003
>> ellip_sos ellip_sos =
      1.0000    1.1880    1.0000    1.0000   -1.5916    0.7107
      1.0000   -0.4191    1.0000    1.0000   -1.3205    0.8568
      1.0000   -0.8072    1.0000    1.0000   -1.1707    0.9631
```

- For exploring the cascade form via the `tf2sos()` function MATLAB has a demo application called `sosdemo`



Graphically viewing the cascade in `sosdemo`

6.6 Overview of Finite Precision Effects

The main motivation for this section is to determine how different structures behave when implemented with finite numerical precision. Relevant issues are:

- Number representations

- Coefficient quantization effects
- Quantization noise effects
- Scaling in fixed-point systems
- Zero-input limit cycles in fixed-point systems

6.6.1 Number Representations

Signals and coefficients in digital signal processing systems are typically represented with finite precision binary numbers. The basic choices are fixed-point and floating-point representations.

Fixed-Point Representations

A fixed-point representation of a real number x using $B+1$ bits of precision takes the form

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

where X_m is a scale factor and

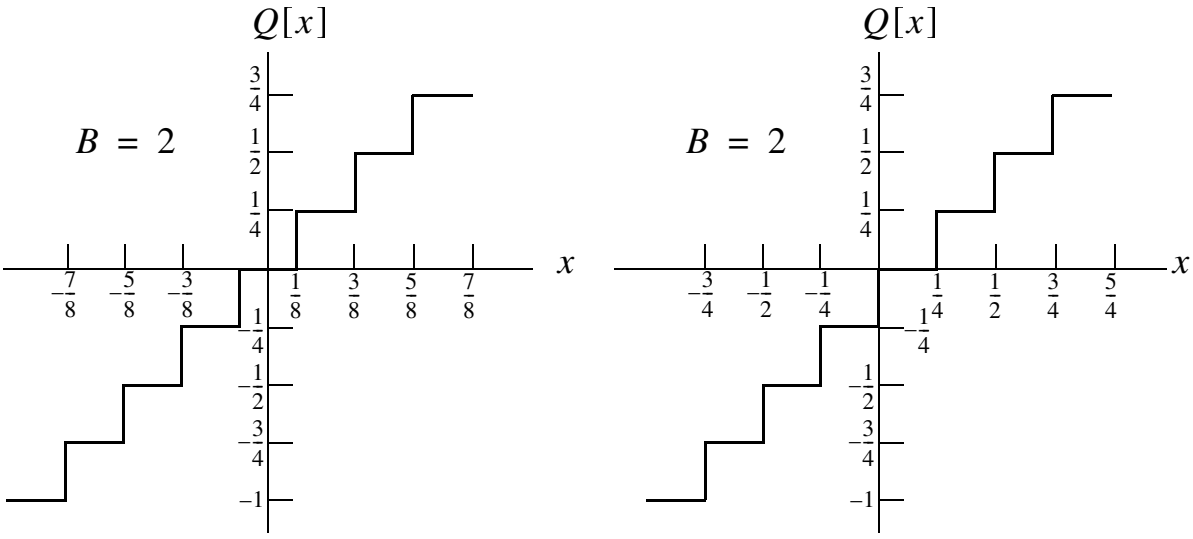
$$\hat{x}_B = b_0 \diamond b_1 b_2 b_3 \ldots b_B$$

is a binary fraction with $\diamond$ being the binary point and b_0 the sign bit.

- The smallest quantization level is again $\Delta = X_m 2^{-B}$
- The quantization error is

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

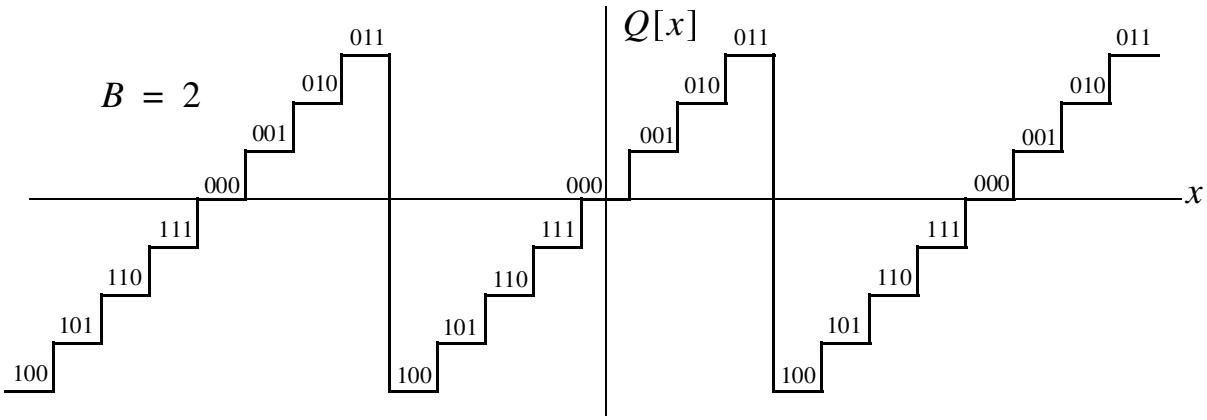
  The following figure shows two's complement rounding ($-\Delta/2 < e \leq \Delta/2$) and two's complement truncation for $B = 2$



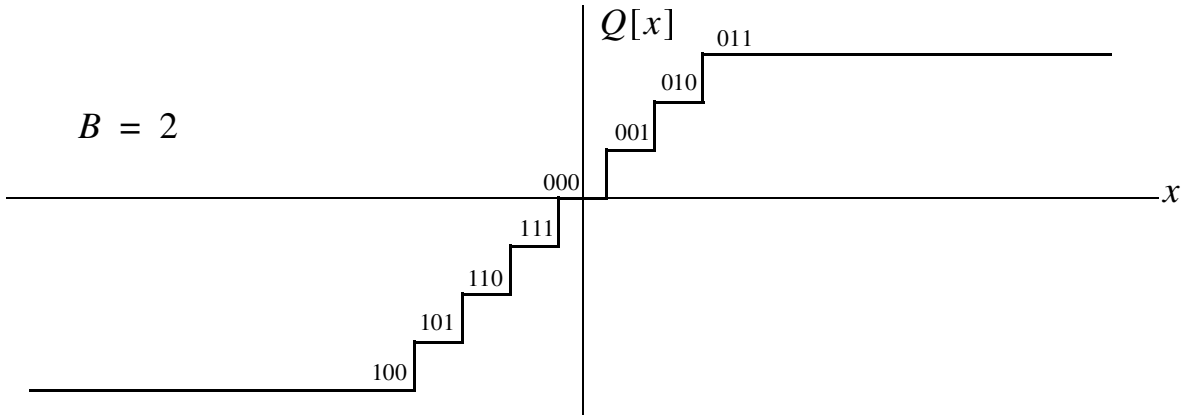
(a) Two's complement rounding (b) Two's complement truncation

Two's complement rounding and truncation for $B = 2$

- If x exceeds X_m then overflow occurs, which may result in a large error
- To minimize overflow we increase X_m, but then the quantization error increases
- In general to increase dynamic range both B and X_m must be increased
- Two ways to deal with overflow is *natural overflow* or *saturation* as shown below



(a) Rounding with Overflow



(b) Rounding with Saturation

Two's complement overflow for $B = 2$

- With natural two's complement overflow the true value wraps creating a large error
- Multiplication of two $(B+1)$-bit numbers results in a $(2B+1)$-bit number which can be rounded or truncated back to $(B+1)$-bits by removing least significant bits
- When adding two fixed-point numbers as long as overflow does not occur, so no truncation or rounding is needed
- If we view X_m as a binary scale factor 2^c, then if say $c = 2$ the binary point in $\hat{x}_B$ is actually shifted right two places e.g.

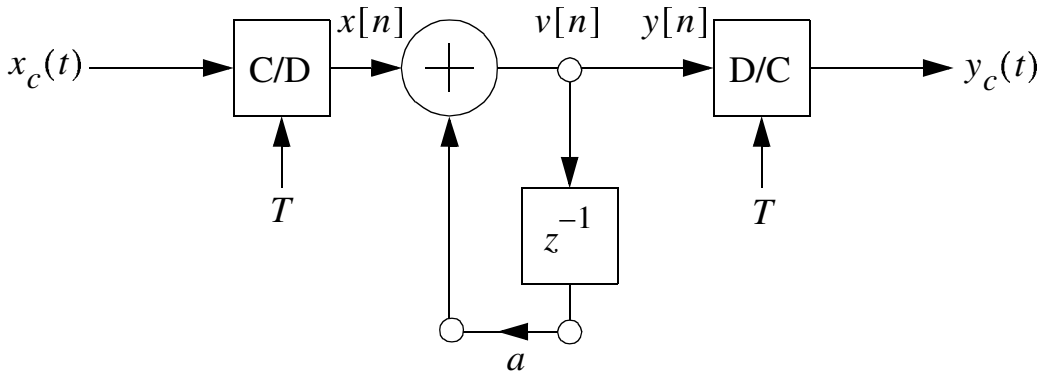
suppose

$$\hat{x} = 2^2 \cdot 0_\diamond 0010 = 4 \cdot \frac{2}{16} = \frac{1}{2}$$
$$= 000_\diamond 10 = \frac{1}{2}$$

Example 6.9: First-Order Filter A/D-$H(z)$-D/A System

Consider an A/D-$H(z)$-D/A system with $H(z)$ being the first-order system

$$H(z) = \frac{1}{1 - az^{-1}}$$

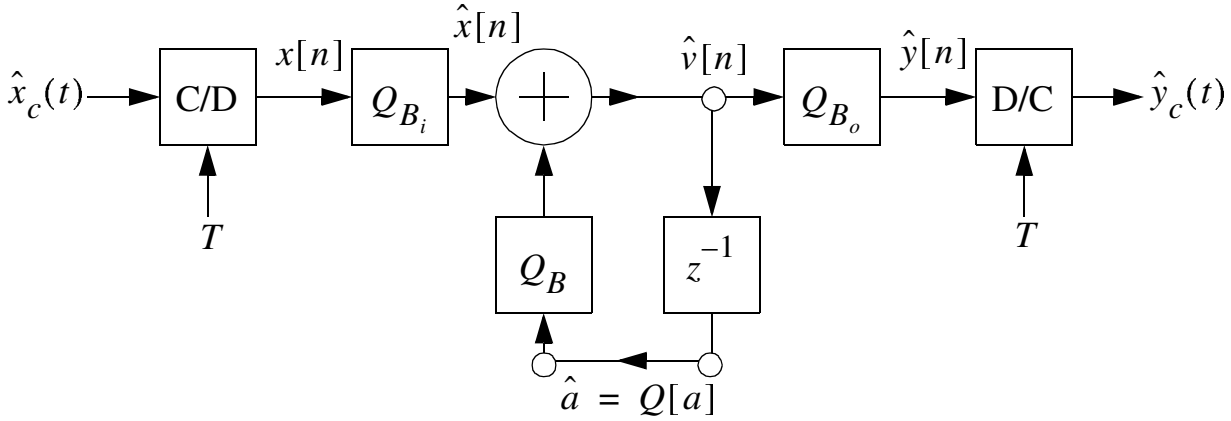


Ideal A/D-$H(z$-D/A linear system block diagram

- The quantized representation for $H(z)$ is

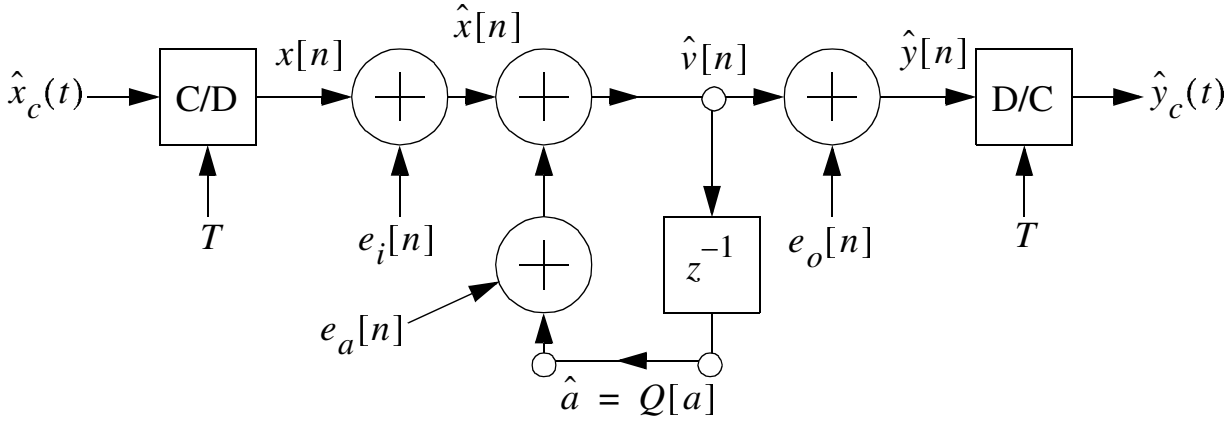
$$\hat{H}(z) = \frac{1}{1 - \hat{a}z^{-1}}$$

where we assume $\hat{a}$ has a $B + 1$ bit representation



Nonlinear A/D-$H(z)$-D/A system model

- The linearized model that follows can be used to analyze the effect of quantization noise. Here each quantization nonlinearity is replaced by an additive noise source.



Linear model with additive noise sources

6.7 Coefficient Quantization

Finite precision filter coefficients is first studied for IIR systems where insuring that poles do not shift to the unit circle or beyond is of great importance. FIR systems are studied next.

6.7.1 Coefficient Quantization in IIR Systems

Consider the rational system function

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

The coefficient sets $\{a_k\}$ and $\{b_k\}$ ideally are represented with infinite precision. With quantized coefficients the coefficient sets are modified.

- The new system function is

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

  where $\hat{a}_k = a_k + \Delta_k$ and $\hat{b}_k = b_k + \Delta_k$

- In this direct-form representation of $H(z)$ we see that quantization error in a given coefficient a_k (b_k) affects the locations of all poles (zeros)

- If we let the pole locations of $\hat{H}(z)$ be given by $z_i + \Delta z_i$ for $i = 1, 2, \ldots, N$, then a simple sensitivity analysis yields

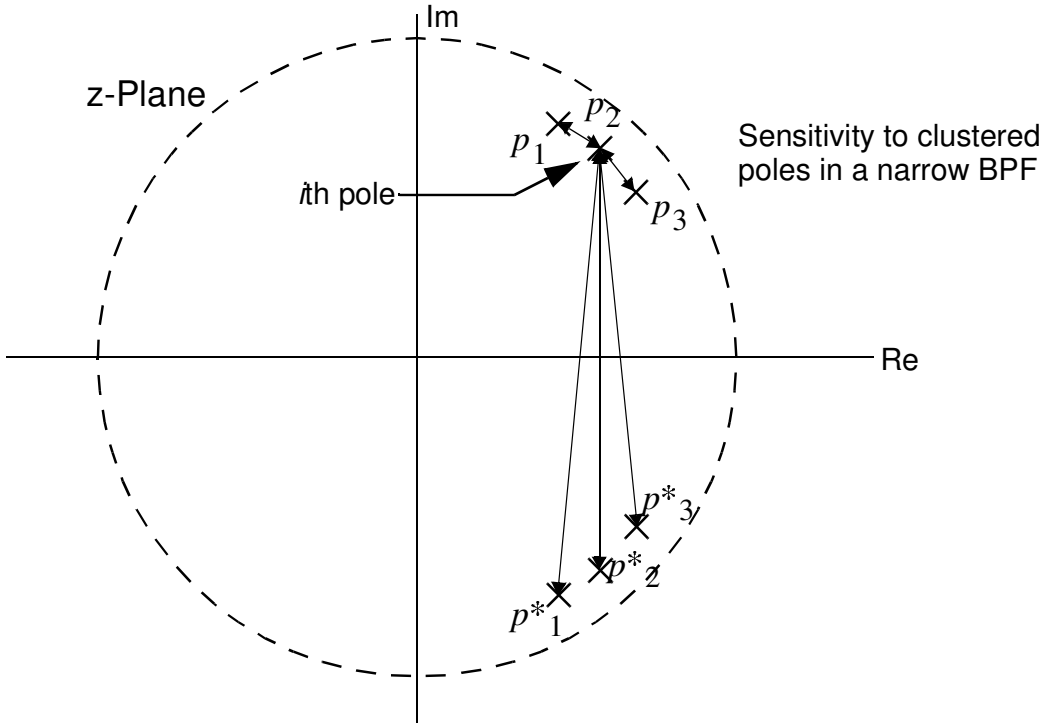
$$\Delta z_i = \sum_{k=1}^{N} \frac{\partial z_i}{\partial a_k} \Delta a_k, \quad i = 1, 2, \ldots, N$$

  where

$$\frac{\partial z_i}{\partial a_k} = \frac{z_i^{N-k}}{\prod_{j=1,\, j \neq i}^{N} (z_i - z_j)}$$

  for $i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, N$

- A similar analysis can be performed for the zero location sensitivities
- The sensitivity formulas indicate that if poles (zeros) are tightly clustered, then a high coefficient sensitivity can be expected
- Additionally the larger the number of clustered poles (zeros) the higher the sensitivity
- Tightly clustered poles occur in narrow bandwidth lowpass and bandpass filters



Pole sensitivity when tightly clustered

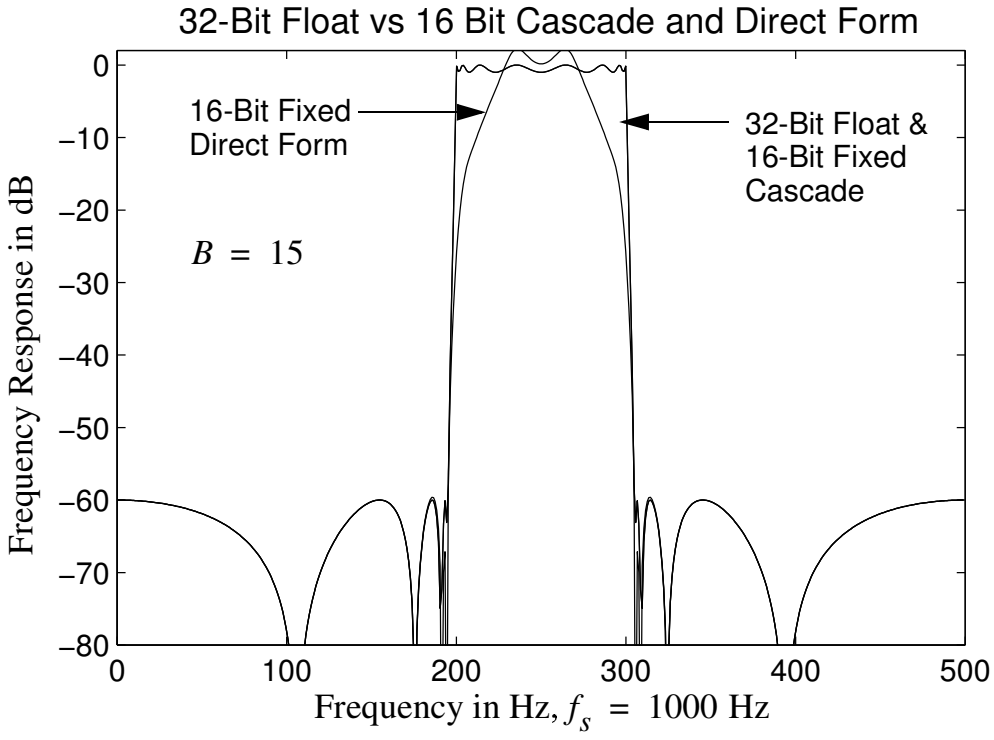
- By using cascaded second order sections the sensitivity can be greatly reduced due to the independence of pole (zero) pairs from each other

Example 6.10: Quantization in a High-Order Elliptic Filter

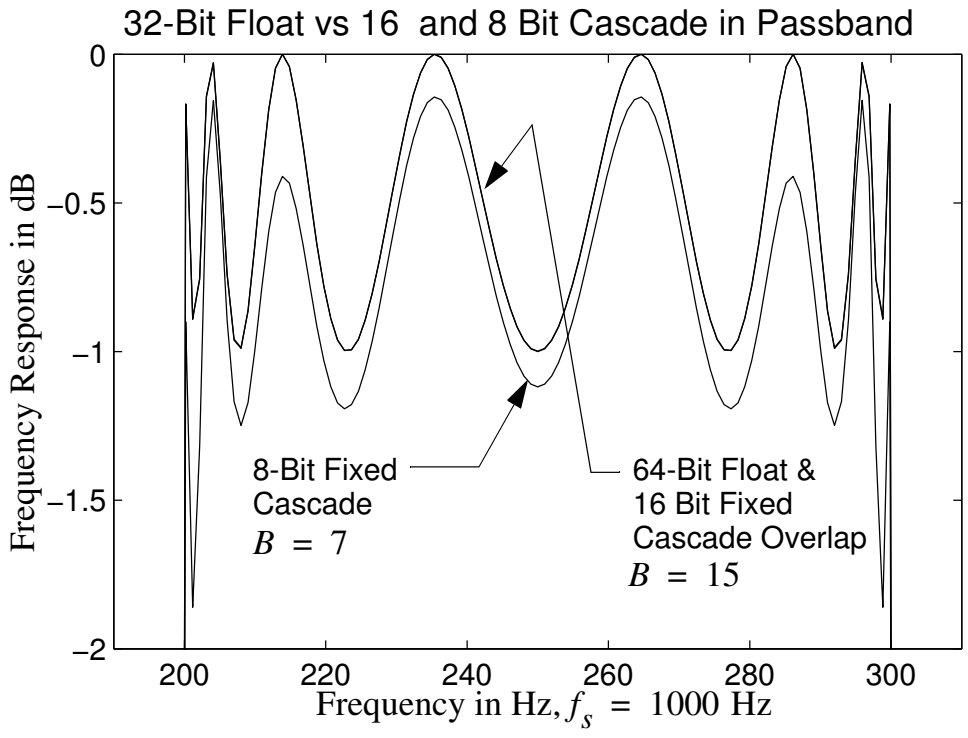
Consider the effect of coefficient quantization on the magnitude response of a 16th-order IIR bandpass filter (8 pole lowpass prototype). The filter is designed to meet the following amplitude specifications for a sampling rate of $f_s = 1$ kHz

$$\begin{aligned} -1.0 \le |H(e^{j2\pi f/f_s})|_{\text{dB}} \le 0, &\quad 200 \le f \le 300 \text{ Hz} \\ |H(e^{j2\pi f/f_s})|_{\text{dB}} \le -60, &\quad f \le 195 \text{ Hz} \\ |H(e^{j2\pi f/f_s})|_{\text{dB}} \le -60, &\quad 305 \le f \le 500 \text{ Hz} \end{aligned}$$

- The log magnitude response for ideal 32-bit floating-point and fixed-point coefficients using a cascade, and direct form realizations are compared below
- The number of magnitude bits, B, is a parameter



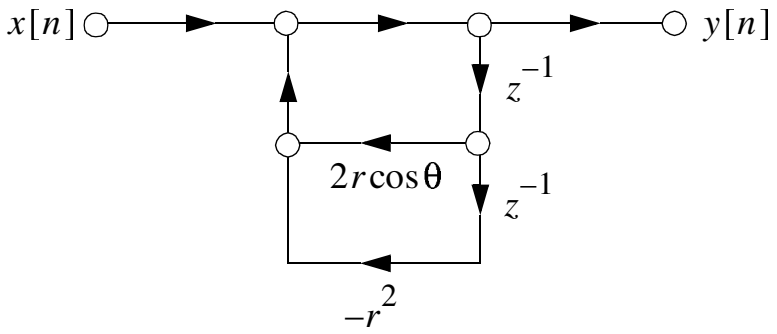
32-Bit floating point vs 16-bit fixed point

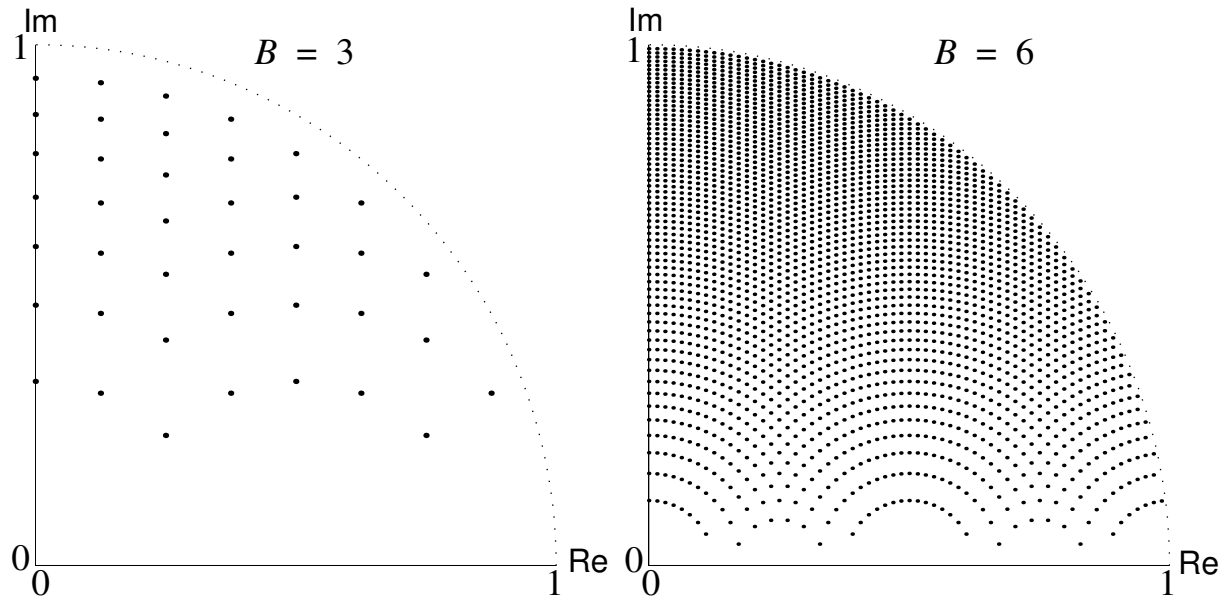


32-Bit float vs 16 & 8-bit fixed cascade in the passband

- A 16-bit coefficient direct-form realization results in a very distorted passband while the cascade realization is very robust
- Dropping the cascade down to just 8 bits (7 magnitude bits) we see there is now some passband error

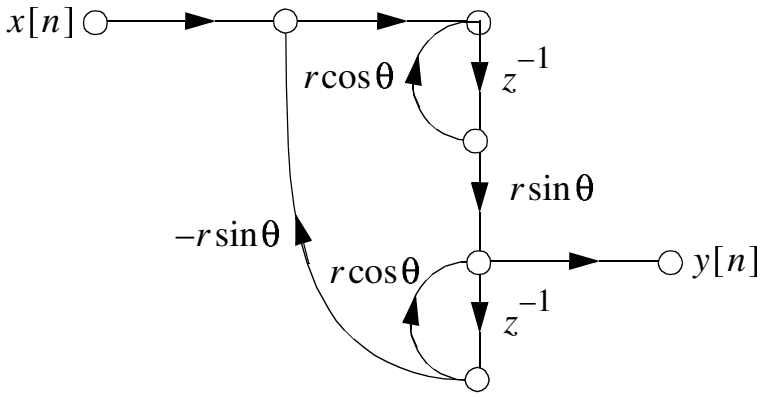
Example 6.11: **Quantized Pole Locations for a direct-form Second-Order Section**



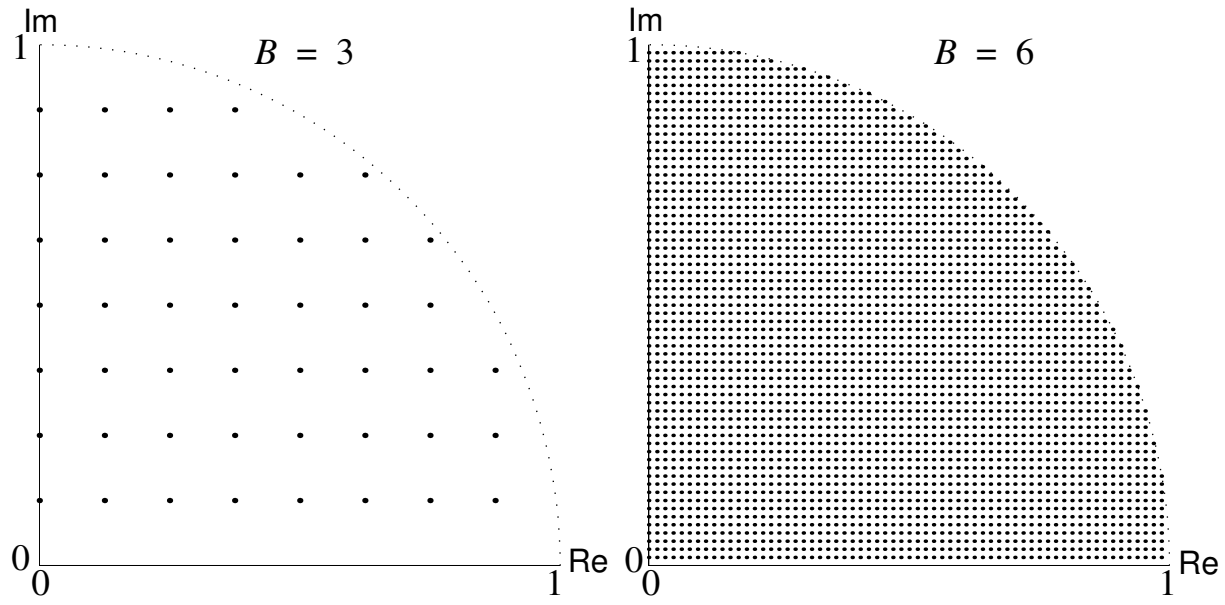


Quantized direct-form conjugate pole locations

Example 6.12: **Quantized coupled form pole locations**



Coupled form structure for conjugate poles



Quantized coupled form pole locations

6.7.2 Coefficient Quantization in FIR Systems

In FIR systems the quantization issue is simpler since we are only concerned with zeros, and stability is not an issue. The only poles of an FIR system are typically at $z = 0$.

- A direct-form FIR system is of the form

$$H(z) = \sum_{n=0}^{M} h[n] z^{-n}$$

- Quantizing the coefficients will produce

$$\hat{h}[n] = h[n] + \Delta h[n]$$

so the system function can be viewed as the parallel system

$$\hat{H}(z) = H(z) + \Delta H(z)$$

with $\Delta H(z) = \sum_{n=0}^{M} \Delta h[n] z^{-1}$

- By studying the error filter frequency response $\Delta H(e^{j\omega})$ we can see the impact of quantization

- We may also choose to simply study the movement or sensitivity of the zeros as a result of coefficient quantization; tightly clustered zeros will be more sensitive to quantization

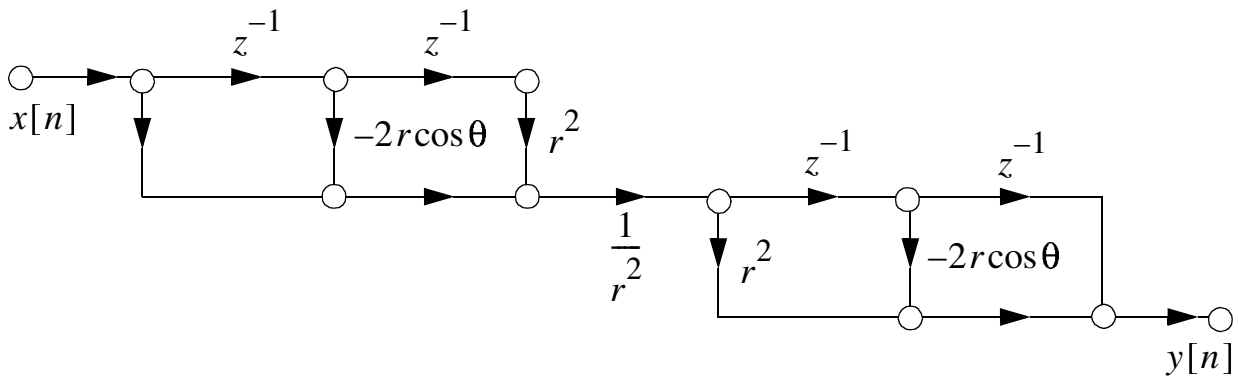
- Slight variations in amplitude response may be acceptable, but what about maintaining linear phase?

  - If the conjugate reciprocal zeros (those outside the unit circle) track the changes of the minimum phase zeros, linear phase will be maintained
  - The exact details of the quantization for signed numbers come into play here
  - Since the impulse response is either symmetric or antisymmetric, we must at most insure that say a and $-a$ have the same magnitude when quantized

- For situations where zeros are tightly clustered a cascade structure can be used

- Quadruplet zeros can be represented with a fourth-order section that insures perfect conjugate reciprocal symmetry, e.g.,

$$\begin{aligned} H_i(z) &= 1 + cz^{-1} + dz^{-2} + cz^{-3} + dz^{-4} \\ &= \left(1 - 2r\cos\theta z^{-1} + r^2 z^{-2}\right)\frac{1}{r^2}\left(r^2 - 2r\cos\theta z^{-1} + z^{-2}\right) \end{aligned}$$



Fourth-order linear phase cascade structure

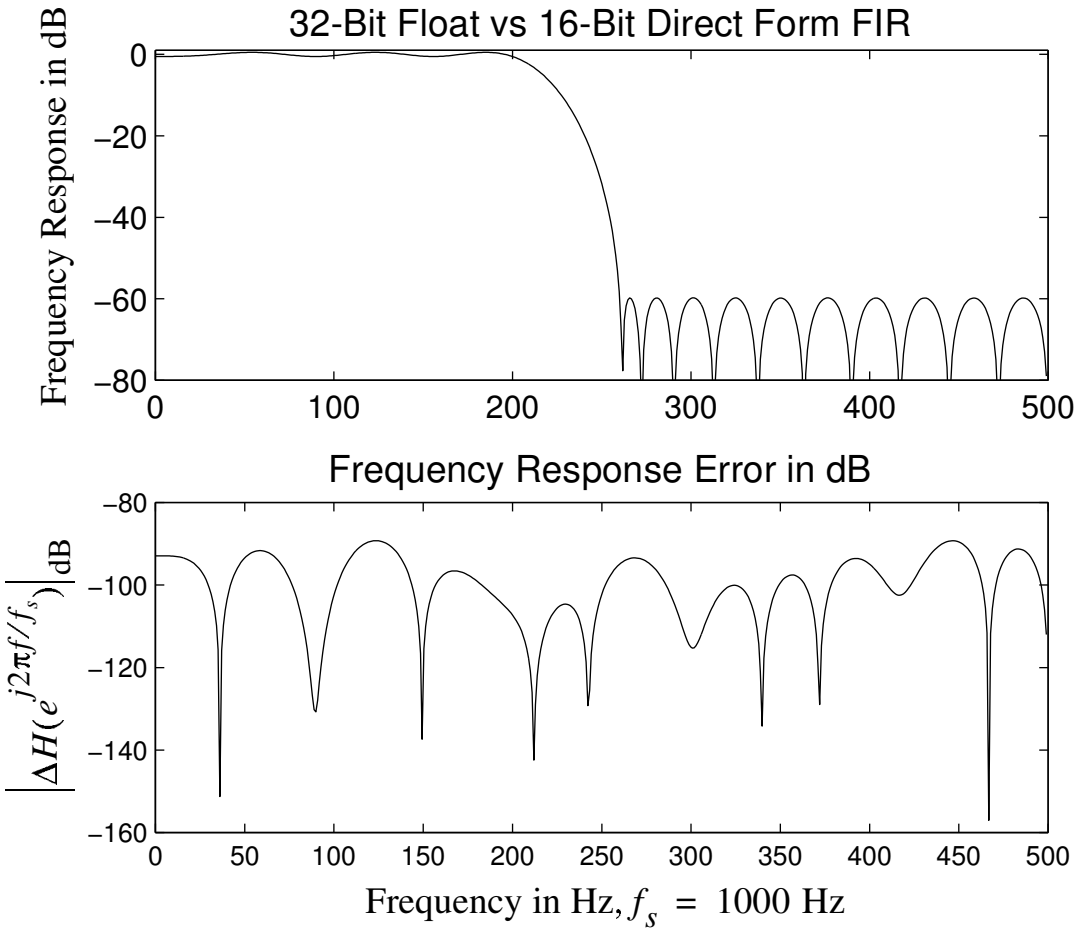
Example 6.13: Quantization in a Linear Phase FIR Filter

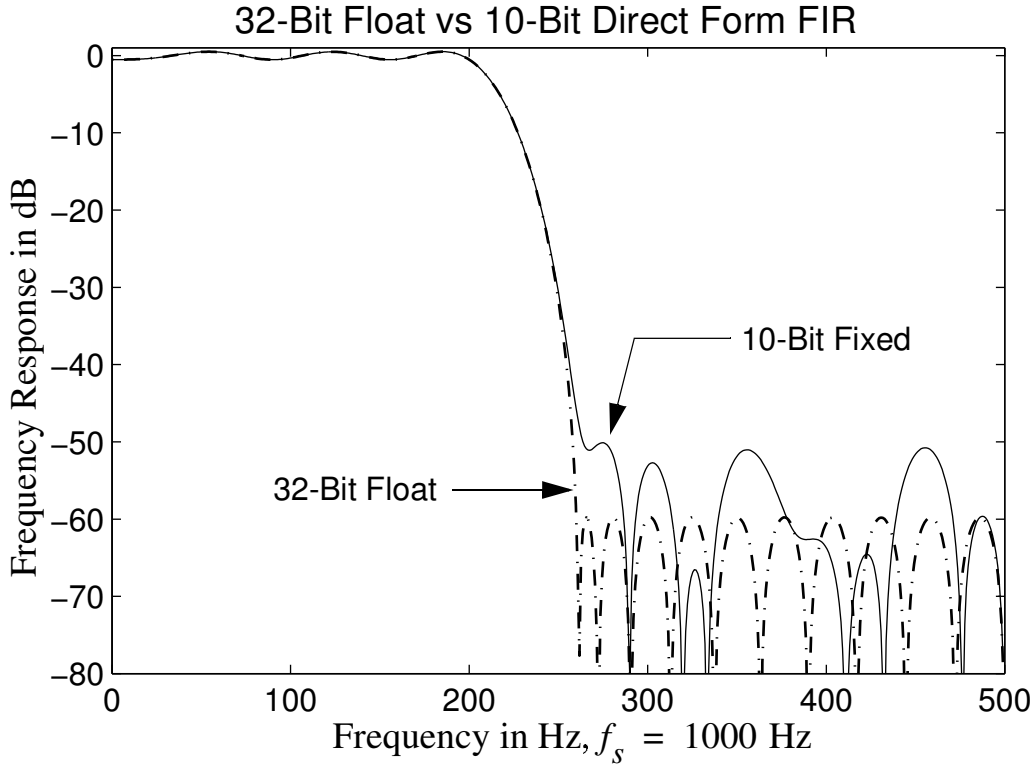
Consider the effect of coefficient quantization on the magnitude response of a $M = 33$ FIR lowpass filter. The filter is designed to meet the following amplitude specifications for a sampling rate of $f_s = 1$ kHz

$$\begin{aligned} -1.0 \le |H(e^{j2\pi f/f_s})|_{\text{dB}} &\le 1.0, \quad 0 \le f \le 200 \text{ Hz} \\ |H(e^{j2\pi f/f_s})|_{\text{dB}} &\le -60, \qquad 260 \le f \le 500 \text{ Hz} \end{aligned}$$

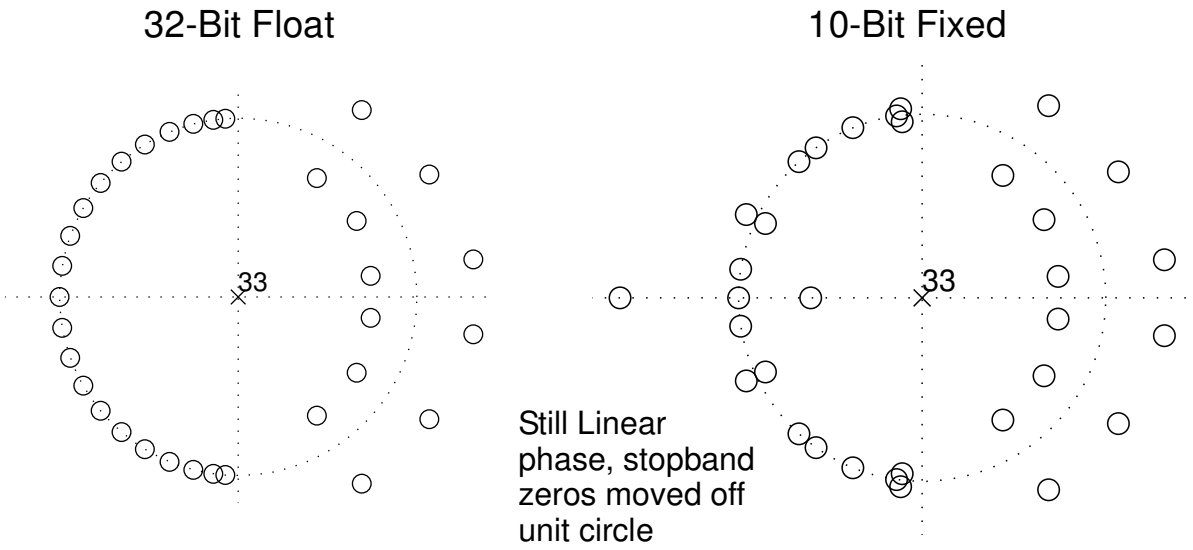
- The log magnitude response for ideal 32-bit floating-point and fixed-point coefficients using a direct-form realization are compared below

- The number of magnitude bits, B, is a parameter


32-Bit Float vs 16-Bit Direct Form FIR
Frequency Response in dB
0
−20
−40
−60
−80
0
100
200
300
400
500
Frequency Response Error in dB
$\left|\Delta H(e^{j2\pi f/f_s})\right|$ dB
−80
−100
−120
−140
−160
0
50
100
150
200
250
300
350
400
450
500
Frequency in Hz, f_s = 1000 Hz


32-Bit Float vs 10-Bit Direct Form FIR
Frequency Response in dB
0
−10
−20
−30
−40
−50
−60
−70
−80
10-Bit Fixed
32-Bit Float
0
100
200
300
400
500
Frequency in Hz, f_s = 1000 Hz

- With 10-bit quantization there is considerable amplitude error in the stopband
- The movement of the zeros is shown in the following pole-zero plots



Zeros movement as a result of 10-bit quantization

6.8 Quantization Noise in Digital Filters

Finite precision arithmetic converts an LCCDE into a nonlinear system. The performance of such systems can be carried out via simulation as discussed in the *Word Length Effects* chapter of the MATLAB workbook. Linear noise models are also useful and provide an analytical approach for obtaining performance results.

In this section we consider an approximate analysis of quantization noise affects due to rounding or truncation following a product operation. The analysis is carried out by replacing each coefficient multiply operation and the associated quantizer, $Q[\cdot]$, by an ideal coefficient multiply plus an additive noise source, i.e.,

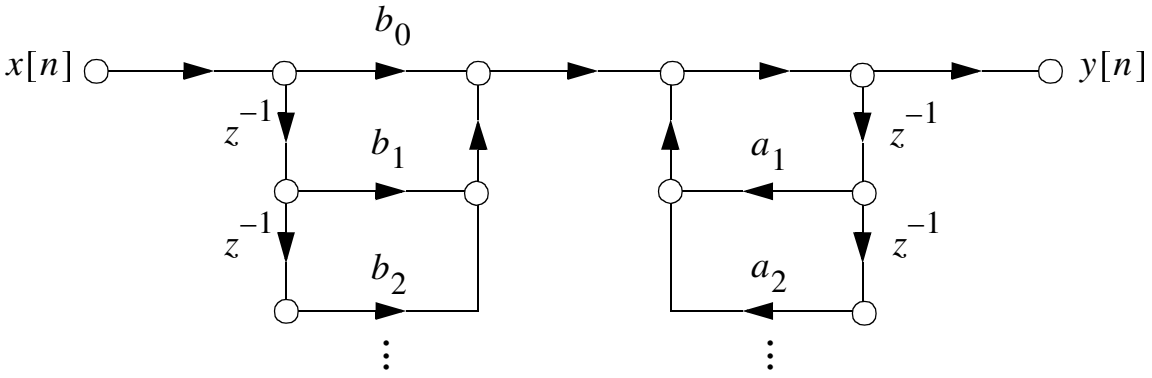
$$Q[bx[n]] = bx[n] + e[n]$$

This basic approach can be applied to a wide variety of DSP algorithms. We begin with the study of IIR systems and move to FIR systems.

6.8.1 Direct-Form IIR Analysis

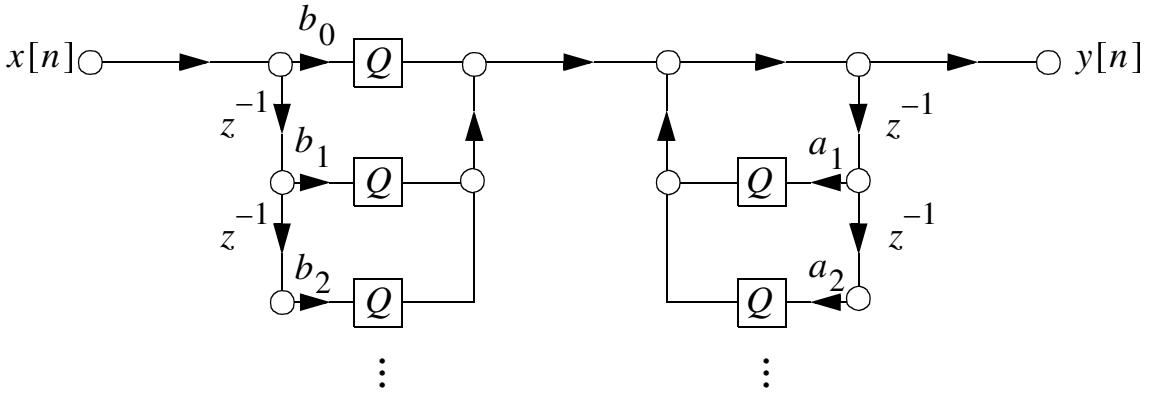
- Consider the familiar Nth-order difference equation

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$$



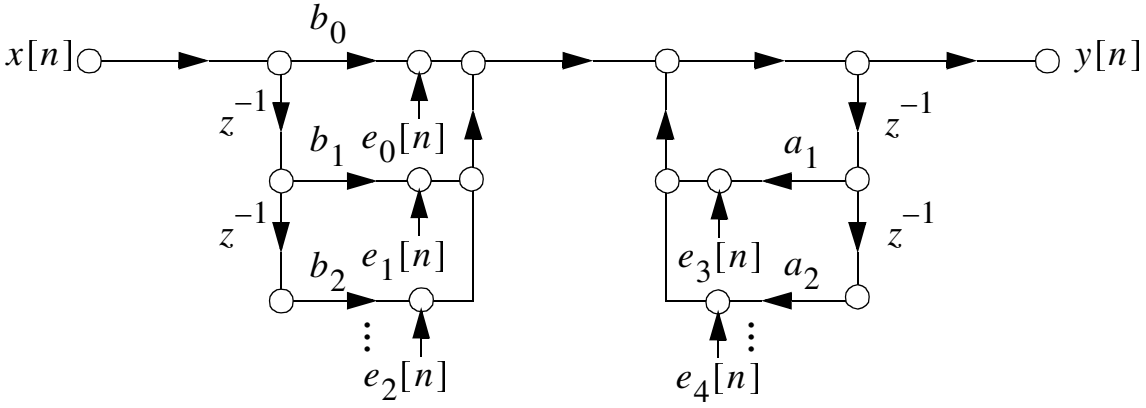
- Assume all signal values and coefficients are in a $(B+1)$-bit fixed-point binary format
- Coefficient multiplications produce a $(2B+1)$ product for which the least significant B bits (LSBs) must be removed using either truncation or rounding
- Additions are then performed using $(B+1)$-bit adders
- The resulting nonlinear difference equation is

$$\hat{y}[n] = \sum_{k=1}^{N} Q\big[a_k \hat{y}[n-k]\big] + \sum_{k=0}^{M} Q\big[b_k x[n-k]\big]$$



Nonlinear direct-form I filter

- We reduce the above nonlinear equation to a linear one by replacing the quantizers with an additive noise model as described earlier



Linear-noise model direct-form I filter

- For analysis purposes the quantization noise sources are modeled as was done in the A/D converter section:
  - Each noise source output, $e[n]$, is a WSS white process
  - $e[n]$ is uniformly distributed over one quantization interval
  - $e[n]$ is uncorrelated with the quantizer input, other noise sources, and the system input (i.e. $x[n]$)
- Assuming rounding $-2^{-B}/2 < e[n] \leq 2^{-B}/2 \Rightarrow$

$$m_e = 0, \qquad \sigma_e^2 = \frac{2^{-2B}}{12},$$

and

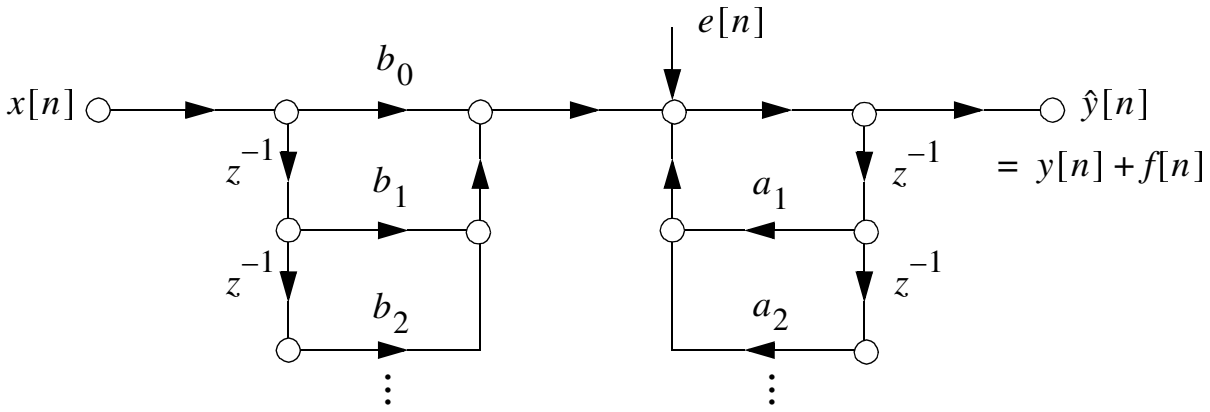
$$\phi_{ee}[n] = \sigma_e^2 \delta[n]$$

- Assuming truncation $-2^{-B} < e[n] \leq 0 \Rightarrow$

$$m_e = -\frac{2^{-B}}{2}, \qquad \sigma_e^2 = \frac{2^{-2B}}{12}$$

6.8.2 Direct-Form IIR Structures

The noise sources on each coefficient can be added together to form an equivalent noise source as shown in the following figure:



- Since the noise sources are mutually uncorrelated we can write the variance of $e[n]$ as the sum of the variances, thus

$$\sigma_e^2[n] = (M + 1 + N)\frac{2^{-2B}}{12}$$

- The output of the system consists of signal and noise

$$\hat{y}[n] = y[n] + f[n]$$

- From the SFG we can write the output noise component as

$$\begin{aligned} f[n] &= e[n] * h_{ef}[n] \\ &= \sum_{k=1}^{N} a_k f[n-k] + e[n] \end{aligned}$$

where $h_{ef}[n]$ is the impulse response due to just the poles

- The corresponding frequency response of the poles is

$$H_{ef}(e^{j\omega}) = \left[1 - \sum_{k=1}^{N} a_k e^{-j\omega n}\right]^{-1}$$

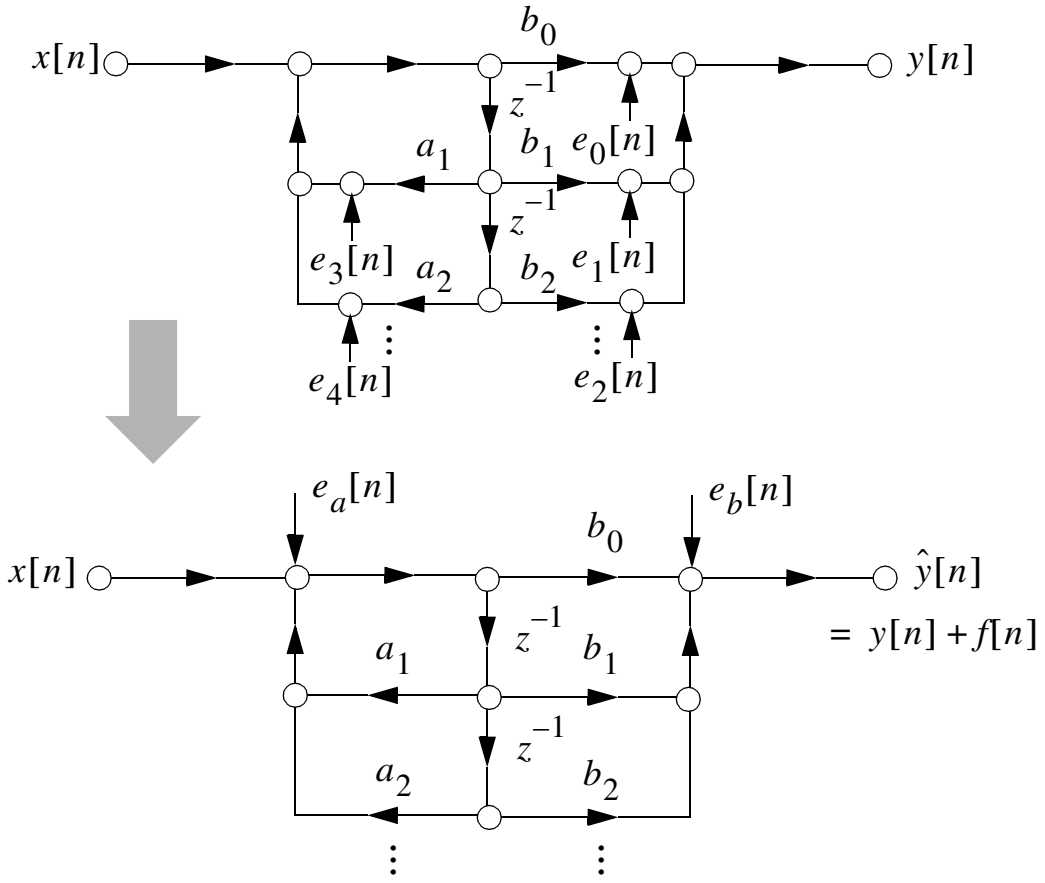
- Since the $e_i[n]$'s are white processes as well as being mutually uncorrelated, that further implies that $e[n]$ is also white, thus the output noise power spectrum and variance (assuming rounding) are

$$\begin{aligned} P_{ff}(\omega) &= \sigma_e^2 |H_{ef}(e^{j\omega})|^2 \\ \sigma_f^2 &= \frac{\sigma_e^2}{2\pi} \int_{-\pi}^{\pi} |H_{ef}(e^{j\omega})|^2 \, d\omega \\ &\overset{\text{or}}{=} \sigma_e^2 \sum_{n=-\infty}^{\infty} |h_{ef}[n]|^2 \\ &\overset{\text{also}}{=} \frac{\sigma_e^2}{2\pi j} \oint_C H_{ef}(z) H_{ef}(z^{-1}) z^{-1} \, dz \end{aligned}$$

where the last two lines result from Parseval's relation for the Fourier transform and z-transform respectively.

Direct-Form II Analysis

The SFG of a direct-form II structure with additive noise sources used to model quantization noise is shown in the following figure.



Direct-form II model with equivalent noise sources

- By combining the input stage noise sources into $e_a[n]$, and the output noise sources into $e_b[n]$, we can write the output noise spectrum as

$$P_{ff}(\omega) = N\frac{2^{-2B}}{12}|H(e^{j\omega})|^2 + (M+1)\frac{2^{-2B}}{12}$$

so the output noise variance is

$$\begin{aligned}\sigma_f^2 &= \left[N\sum_{n=-\infty}^{\infty}|h[n]|^2 + (M+1)\right]\frac{2^{-2B}}{12} \\ &= \left[\frac{N}{2\pi j}\oint_C H(z)H(z^{-1})z^{-1}\,dz + (M+1)\right]\frac{2^{-2B}}{12}\end{aligned}$$

- Note that the noise source associated with the zeros adds directly to the output and the noise from the pole coefficients is filtered by the entire system function.

Other Options

- Given that extended precision registers are available (most general purpose DSP microprocessors have this capability), we may choose to accumulate a sum of many products in a $(2B+1)$ or $(2B+2)$ bit register
  - A typical case in point would be a 16-bit word length with 32-bit extended precision registers (in C `short` and `long`)
- In a direct-form I for example the sum of products could be over the entire difference equation

$$\hat{y}[n] = Q\left[\sum_{k=1}^{N} a_k \hat{y}[n-k] + \sum_{k=0}^{M} b_k x[n-k]\right]$$

- Now we have only a single white noise source that is filtered (shaped) by the poles, but $(M+N+1) \rightarrow 1$
- In a direct-form II the sum of products might be done in two steps

$$\hat{w}[n] = Q\left[\sum_{k=1}^{N} a_k \hat{w}[n-k] + x[n]\right]$$

where $\hat{w}[n-k]$ are the values stored in the delay registers, and the final step is

$$\hat{y}[n] = Q\left[\sum_{k=0}^{M} b_k \hat{w}[n-k]\right]$$

- Now we have just two white noise sources, one at the input which is shaped by the poles and zeros, and one at the output which receives no shaping

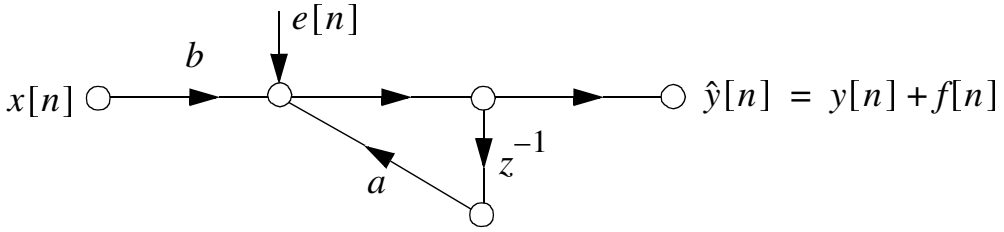
Example 6.14: A First-Order System Direct-Form I & II

Suppose the first-order system

$$H(z) = \frac{b}{1 - az^{-1}}$$

is implemented using binary fixed-point arithmetic (assume two's-complement rounding)

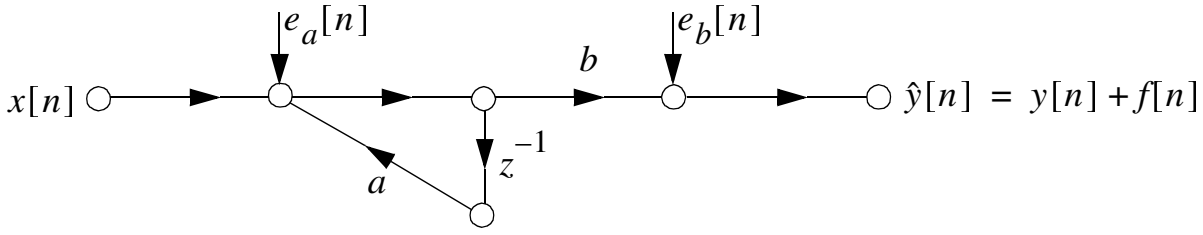
- For a direct-form I realization we have the following SFG:



- The noise variance at the output is

$$\sigma_f^2 = 2\frac{2^{-2B}}{12}\sum_{n=0}^{\infty} a^{2n} = 2\frac{2^{-2B}}{12}\frac{1}{1-a^2}$$

- For a direct-form II realization we have the following SFG:



- The noise variance at the output is

$$\sigma_f^2 = \frac{2^{-2B}}{12}\left[1 + \frac{b^2}{1-a^2}\right] = \frac{2^{-2B}}{12}\left[\frac{1+b^2-a^2}{1-a^2}\right]$$

- Note that for both realizations the output noise variance increases as the pole moves closer to the unit circle
- As the pole moves closer to the unit circle increase the word-length to combat increased noise variance

---

6.8.3 Scaling in Fixed-Point Systems

When using fixed-point arithmetic it is important to avoid overflow at all unique node variables. Assuming the computer represents numbers as binary fractions, then node variable $w_k[n]$ must be constrained such that

$$|w_k[n]| < 1 \quad \forall\ k \text{ in the network}$$

- Let $h_k[n]$ be the system impulse response from the input to the kth node variable $w_k[n]$,

$$|w_k[n]| = \left|\sum_{k=-\infty}^{\infty} x[n-m]h_k[m]\right| \leq \sum_{k=-\infty}^{\infty} |x[n-m]||h_k[m]|$$

$$|w_k[n]| \leq x_{max} \sum_{k=-\infty}^{\infty} |h_k[m]|$$

where $x_{\max}$ upper bounds $|x[n]|$

- In general to prevent overflow in a k node network choose $x_{\max}$ such that

$$x_{\max} < \frac{1}{\max_k \left[\sum_{m=-\infty}^{\infty} |h_k[m]|\right]}$$

- The above condition on $x_{\max}$ is actually quite conservative and may result in lost precision due to excessive scaling of the input $x[n]$ or $x_{\max}$ may be too large, then we may choose to replace $x_{\max}$ with a scaled version using scale multiplier s, i.e. $x_{\max}[n] \rightarrow s x_{\max}[n]$

- A second bounding technique, which is less conservative assumes the input is narrowband, in particular a sinusoid $x[n] = x_{max} \cos \omega_o n$

- In steady-state the kth node has value

$$w_k[n] = |H_k(e^{j\omega_o})| x_{max} \cos[\omega_o n + \angle H_k(e^{j\omega_o})]$$

- As an upper bound we can write

$$|w_k[n]| \leq \max_{|\omega| \leq \pi} |H_k(e^{j\omega})| x_{\max} < 1$$

  Thus when considering all potential overflow nodes choose $s x_{\max}$ such that

$$s x_{\max} < \frac{1}{\max_{k,\ |\omega| \leq \pi} |H_k(e^{j\omega})|}$$

- Yet another approach is to use the total signal energy at each node

- From Parsevals' theorem followed by the Schwartz inequality, we can write that the energy in the kth node variable is bounded by

$$\sum_{n=-\infty}^{\infty} |w_k[n]|^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_k(e^{j\omega})X(e^{j\omega})|^2\, d\omega$$
$$\leq \sum_{n=-\infty}^{\infty} |x[n]|^2 \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2\, d\omega$$

- If we again scale $x[n]$ using the factor s we insure that the kth node energy is bounded by the input energy if

$$s^2 \leq \left[\frac{1}{2\pi}\int_{-\pi}^{\pi} |H_k(e^{j\omega})|^2\, df\right]^{-1} = \left[\sum_{n=-\infty}^{\infty} |h_k[n]|^2\right]^{-1}$$

- An interesting result, which is not proven here, is that the three scaling approaches are related as follows

$$\left\{\sum_{n=-\infty}^{\infty} |h_k[n]|^2\right\}^{1/2} \leq \max_{k,\omega} |H_l(e^{j\omega})| \leq \sum_{n=-\infty}^{\infty} |h_k[n]|$$

- In any case a useful performance measure is the SNR at the system output

$$\text{SNR} = \frac{\sigma_y^2}{\sigma_f^2}$$

  - If the input must be scaled down by $s < 1$ the SNR is reduced accordingly due to less signal power entering the system

– In the text primed variables are introduced to denote signal and noise quantities present at the output as a result of scaling the input by s

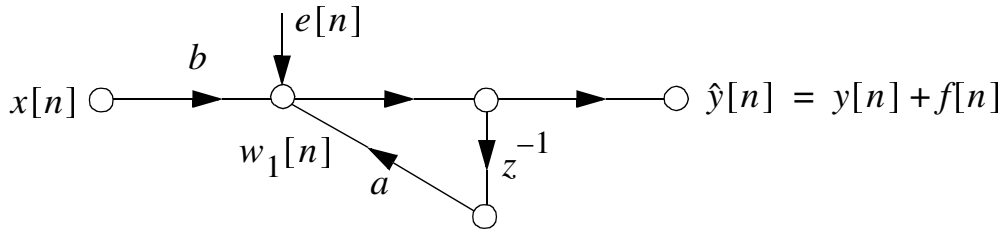
Example 6.15: A First-Order System

Consider the first-order system discussed earlier

$$H(z) = \frac{b}{1 - az^{-1}}$$

with input $x[n]$ assumed to be a white sequence uniformly distributed over the interval $[-x_{max}, x_{max}]$.

- For the direct-form I structure we have only one node, $w_1[n]$, to check for overflow



- To find $h_1[n]$ solve $w_1[n] = aw_1[n-1] + bx[n]$; clearly, $h_1[n] = ba^n u[n]$

- Using the conservative bound on x_{max} we have that

$$x_{max} < \left[\sum_{n=0}^{\infty} |b||a|^n \right]^{-1} = \frac{1 - |a|}{|b|}$$

- The output variances are

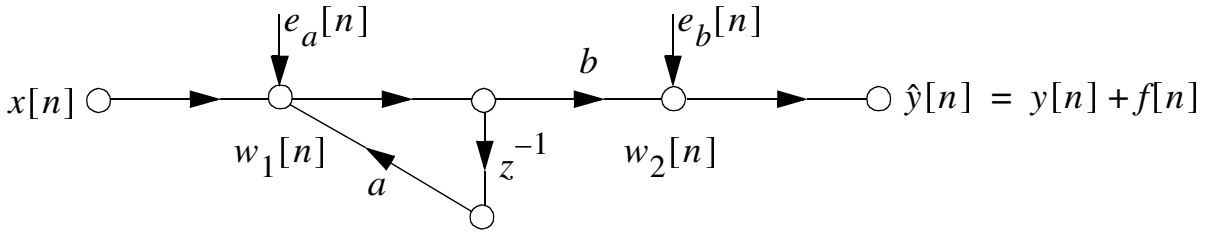
$$\sigma_f^2 = 2\frac{2^{-2B}}{12} \cdot \frac{1}{1-a^2}$$
$$\sigma_y^2 = \sigma_x^2 \cdot \frac{b^2}{1-a^2}$$

- Since $x[n]$ is uniform $\sigma_x^2 = x_{max}^2/3$, thus the output SNR is

$$\left(\frac{\sigma_y^2}{\sigma_f^2}\right) = (1-|a|)^2 2^{(2B+1)}$$

- Note that not only does σ_f^2 increase as a approaches the unit circle, but σ_y^2 also decreases due to scaling

- In the direct form II implementation of the system two nodes must be checked for overflow



- The node variables $w_1[n]$ and $w_2[n]$ have associated impulse responses

$$h_1[n] = a^n u[n] \text{ and } h_2[n] = ba^n u[n]$$

thus we must choose

$$x_{max} < \min\left[\frac{1-|a|}{|b|}, 1-|a|\right]$$

- The SNR is then given by

$$\left(\frac{\sigma_y^2}{\sigma_f^2}\right) = \min\left[\left(\frac{1-|a|}{|b|}\right)^2, (1-|a|)^2\right]\frac{b^2 2^{(2B+2)}}{1+b^2-a^2}$$

- As a special case consider $b = 2,\ a = 3/4$, and $B = 15$

$$\left(\frac{\sigma_y^2}{\sigma_f^2}\right)_{\text{form I}} = 2^{27} = 81.28 \text{ dB}$$

$$\left(\frac{\sigma_y^2}{\sigma_f^2}\right)_{\text{form II}} = \frac{2^{32}}{71} = 77.82 \text{ dB}$$

- The direct form I structure is about 3.5 dB better for this example

6.8.4 Comments on the Cascade Structure:

- The cascade realization of an IIR system was noted earlier as being relatively insensitive to coefficient quantization, what about quantization noise performance?

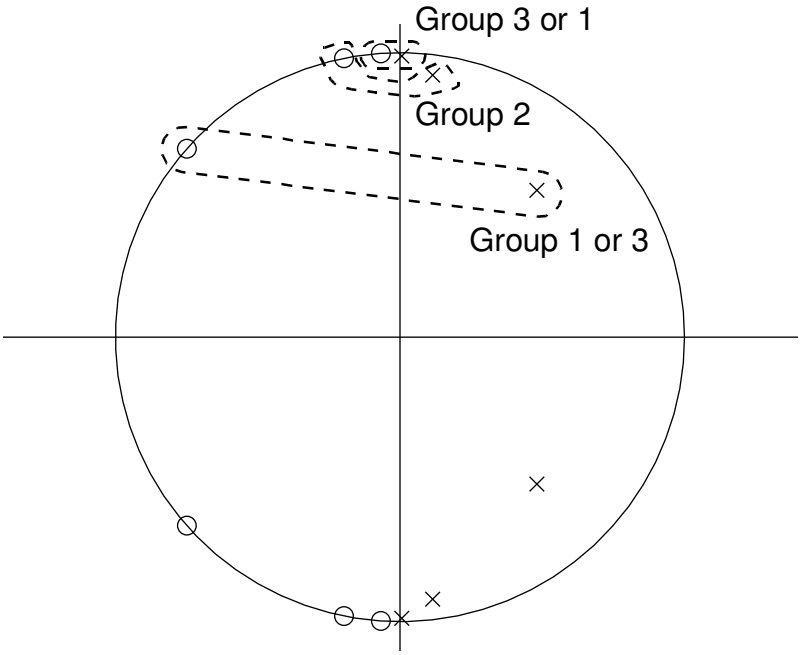
- Jackson has developed some simple rules for pairing and ordering the poles and zeros in a cascade realization

    1. Pair up the pole closest to the unit circle with the nearest zero
    2. Apply (1) repeatedly until all poles and zeros are paired
    3. In the cascade order the sections such that the pole closeness to the unit circle is either increasing or decreasing

- The motivation for pole-zero pairing as in rule 1, is to minimize high peak gain which causes overflow and increased quantization noise

Example 6.16: A Sixth-Order Elliptic

Pair ordering for a sixth order elliptic lowpass filter is shown in the following figure.



Pole-zero pairing in a sixth-order elliptic lowpass filter

6.8.5 Direct-Form FIR Structures

A linear direct-form I or II FIR system has difference equation of the form

$$y[n] = \sum_{k=0}^{M} h[k]x[n-k]$$

and the nonlinear version is of the form

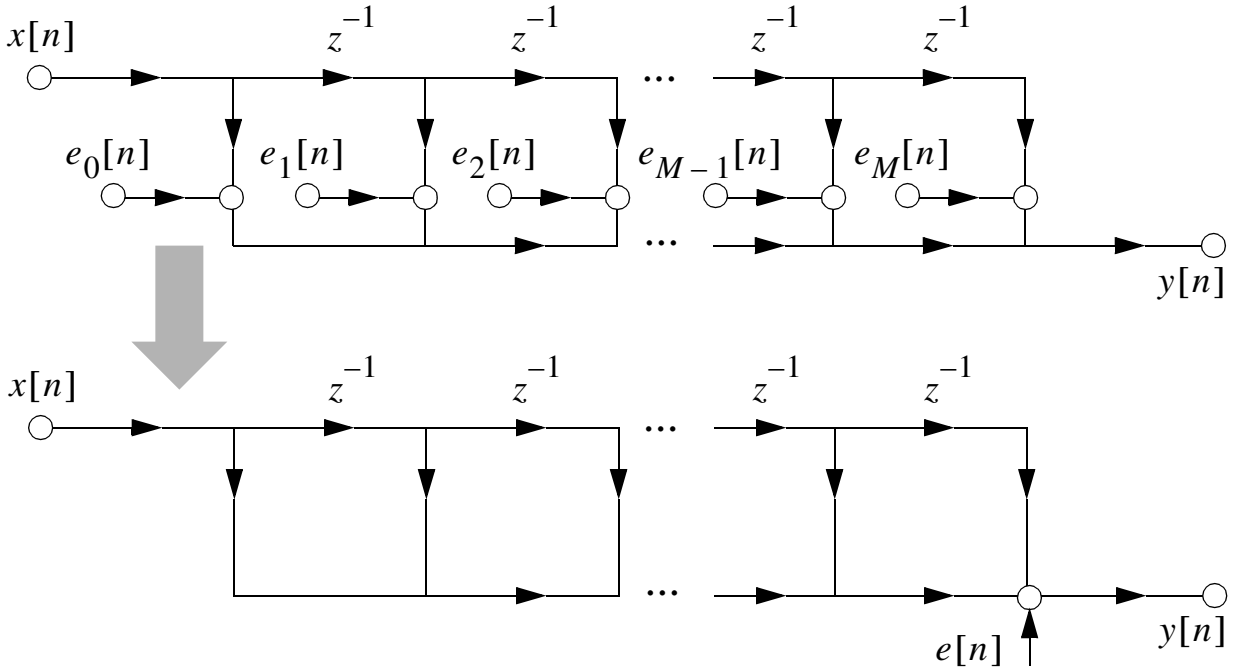
$$\hat{y}[n] = \sum_{k=0}^{M} Q\big[h[k]x[n-k]\big]$$

A linear noise model analysis would produce output

$$\hat{y}[n] = \sum_{k=0}^{M} h[k]x[n-k] + f[n]$$

where $f[n]$ is a white noise source applied directly to thge output with variance

$$\sigma_f^2 = (M+1)\frac{2^{-2B}}{12}$$



Direct-form FIR linear-noise models

- If a double-length accumulator is available for the sum of products we can reduce the noise significantly; $(M+1) \rightarrow 1$ for M possibly large

- Overflow of the partial sums is not a problem since two's complement arithmetic can handle it as long the true sum does not overflow; scaling the impulse response coefficients by s can be used, but the overall SNR will also be decreased as in the IIR case

6.9 Zero-Input Limit Cycles

A stable LTI IIR system ideally will have an output that decays toward zero once the input has been zero for some period of time. For systems implemented using finite-length register arithmetic it is possible for the ouput to oscillate even though the input is fixed at zero. When this happens the system is said to be in a *zero-input limit cycle*. Limit cycles may be caused by quantization of feedback parameters or node overflows in additions.

Example 6.17: Limit Cycles in a First-Order System

As a simple limit cycle example consider a first-order IIR system with difference equation

$$y[n] = ay[n-1] + x[n], \quad |a| < 1$$

- Assuming fixed-point arithmetic with upward rounding of the product $ay[n-1]$, the system difference equation becomes

$$\hat{y}[n] = Q\big[a\hat{y}[n-1]\big] + x[n]$$

  where $Q[\cdot]$ is the rounding operation

- Let $B = 3$, $a = 1/2 = 0_\diamond 100$, and the input be $x[n] = (7/8)\delta[n] = (0_\diamond 111)\delta[n]$

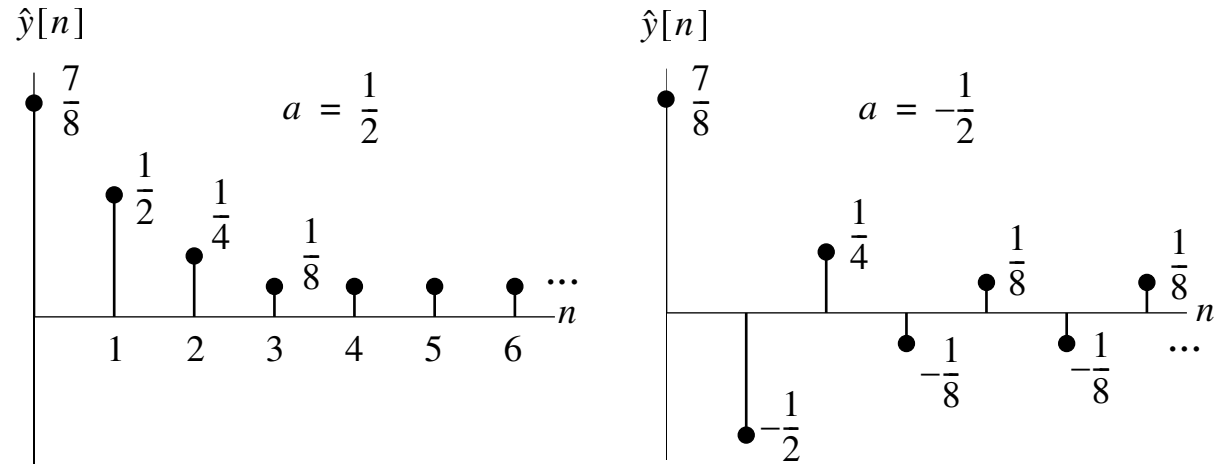
- The output assuming zero initial conditions is

$$\begin{aligned}
y[0] &= 7/8 \\
y[1] &= Q_3[(1/2)(7/8)] = Q_3[7/16] = 1/2 \\
y[2] &= Q_3[(1/2)(1/2)] = Q_3[1/4] = 1/4 \\
y[3] &= Q_3[(1/2)(1/4)] = Q_3[1/8] = 1/8 \\
y[4] &= Q_3[(1/2)(1/8)] = Q_3[1/16] = 1/8 \\
&\vdots \quad \vdots \\
y[n] &= 1/8
\end{aligned}$$

- Let $B = 3$, $a = -1/2 = 1_\diamond 100$, and the input be $x[n] = (7/8)\delta[n] = (0_\diamond 111)\delta[n]$

- The output assuming zero initial conditions is

$$\begin{aligned}
y[0] &= 7/8 \\
y[1] &= Q_3[(-1/2)(7/8)] = Q_3[-7/16] = -1/2 \\
y[2] &= Q_3[(-1/2)(-1/2)] = Q_3[1/4] = 1/4 \\
y[3] &= Q_3[(-1/2)(1/4)] = Q_3[-1/8] = -1/8 \\
y[4] &= Q_3[(-1/2)(-1/8)] = Q_3[1/16] = 1/8 \\
&\vdots \quad \vdots \\
y[n] &= 1/8 \\
y[n+1] &= -1/8
\end{aligned}$$



Plots of the output $\hat{y}[n]$ for $x[n] = (7/8)\delta[n]$

- For the two cases considered here the outputs appear as if the first-order pole was actually at ± 1 rather than $\pm 1/2$
- Limit cycles of this type can be characterized by a *dead band*, that is whenever $\hat{y}[n-1]$ falls within

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

  and the input is zero, a limit cycle begins
- The limit cycle will stop only if the input moves $\hat{y}[n-1]$ out of the dead band

---

- Another type of limit cycle are those due to *overflow oscillations*. Here the output often oscillates over a large portion of the full-scale amplitude

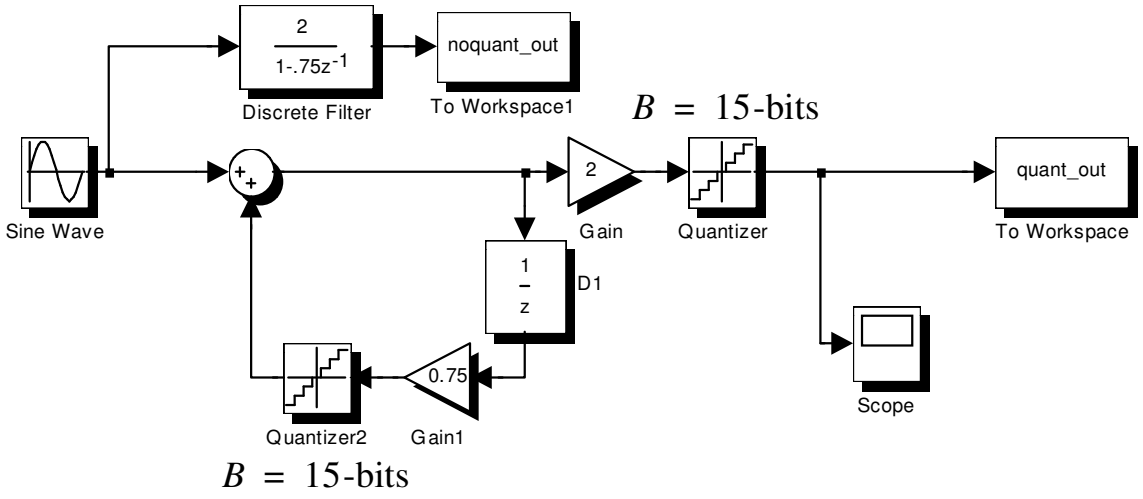
Example 6.18: Filter Simulation with Simulink

In this example we use MATLAB Simulink to construct a simulation of a first-order IIR filter in a direct-form II topology. The filter

$$H(z) = \frac{b}{1 - az^{-1}} = \frac{2}{1 - 0.75z^{-1}}$$

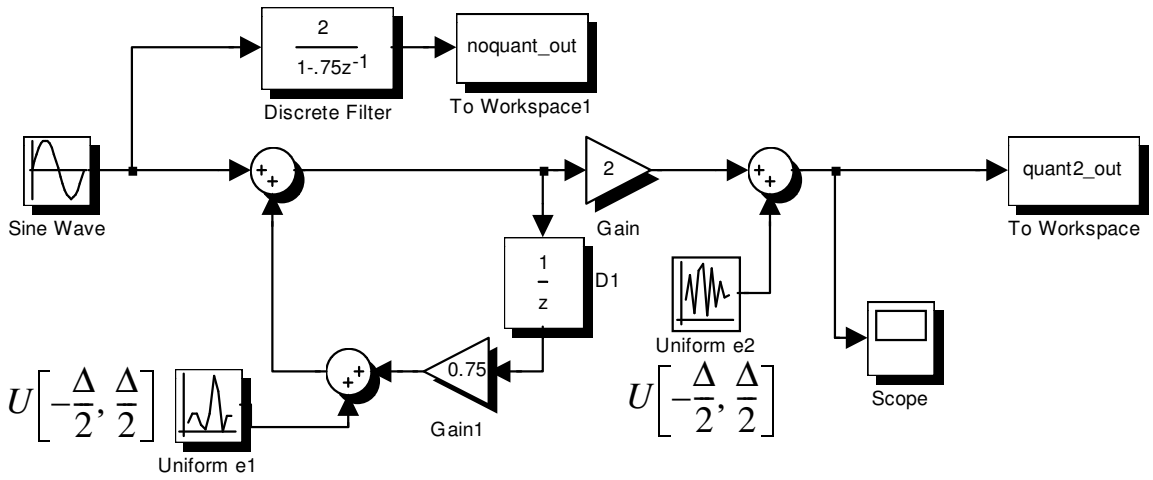
was considered in an earlier noise analysis example (note this example can now be reworked using the filter toolbox).

- A model is constructed with actual $B = 15$ quantizers for the a and b filter coefficients and the input is scaled to avoid overflow



Nonlinear first-order system

- A second linear noise model is also constructed for comparison purposes



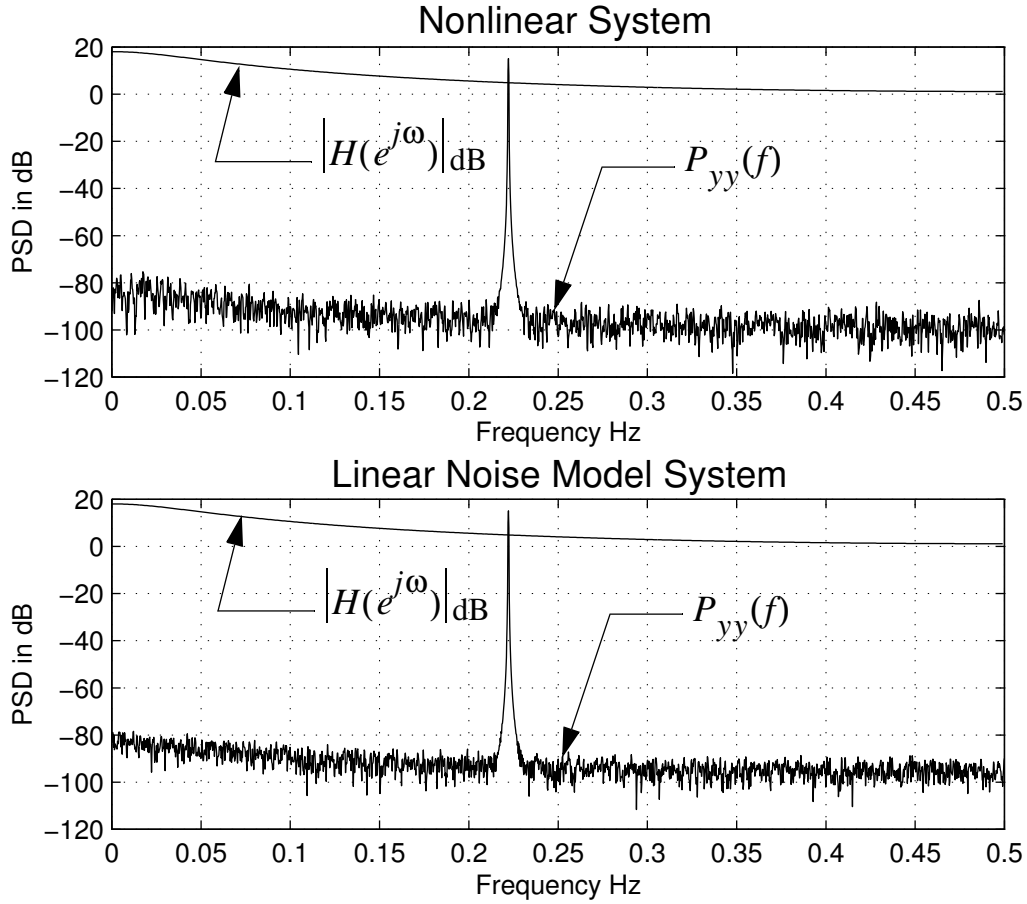
Linear noise model first-order system

- In both of the above Simulink models the sampling rate is 1 Hz and the simulation time is 10,000 seconds
- A test sinusoid is input at frequency $f_0 = 0.2222$ Hz
- A reference ideal filter is also created and the sinusoid is passed through it too
- Using the reference signal and the composite output from both round-off noise systems, the SNR is computed in dB

```
>> 10*log10(cov(noquant_out)/cov(quant_out - noquant_out))

ans = 74.394 dB

>> 10*log10(cov(noquant_out)/cov(quant2_out - noquant_out))

ans = 73.478 dB
```

- SNR results are quite similar to those predicted by the earlier analysis, except the input is a sinsoid as compared with a uniformly distributed white signal sequence

- Power specral density plots for the two systems are compared below, including an overlay of the frequency response to see how the noise spectrum is shaped



Output spectra (a) nonlinear model (b) linear noise model

- The MATLAB `psd()` function with 4096 points is used in both plots
- The quantization noise spectral shapes are as expected since one of the noise sources passes through the filter
- The similarity between the true nonlinear model and the linear noise model is impressive

.