Course Notes 7 – Structures for Discrete-Time Systems

7.0 Introduction

The actual implementation of an LTI digital filter can be either in software or hardware form, depending on applications. In either case, the signal variables and the filter coefficients cannot be represented with infinite precision.

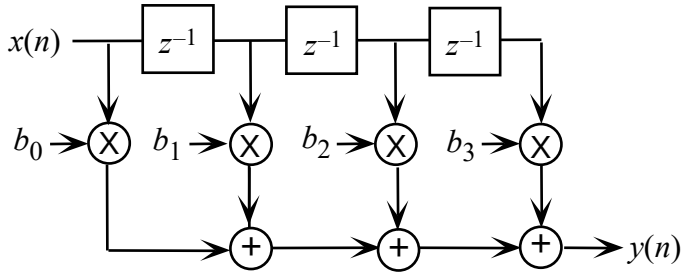
Hence, a direct implementation of a digital filter based on either the difference equation or the finite convolution sum may not provide satisfactory performance due to finite precision arithmetic.

It is thus of practical interest to develop alternate realizations and choose the structure that provides satisfactory performance under finite precision arithmetic.

In the time domain, the input-output relations of an LTI digital filter is given by the convolution sum or, by the linear constant coefficient difference equation

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k) \qquad \text{Convolution sum}$$

$$y(n) = -\sum_{k=1}^{N} d_k y(n-k) + \sum_{k=1}^{N} p_k x(n-k) \qquad \text{Algorithm based on difference equation}$$



Finite Impulse Response (FIR) Configuration

Difference equation (algorithm):

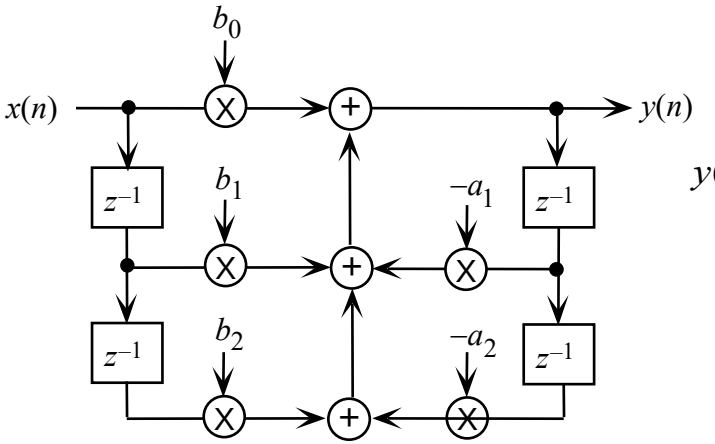
$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3)$$

System Function:

$$H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}$$

Frequency Response:

$$H(\omega) = b_0 + b_1 e^{-j\omega} + b_2 e^{-j2\omega} + b_3 e^{-j3\omega}$$



Infinite Impulse Response (IIR) Configuration

$$y(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) - a_1 y(n-1) - a_2 y(n-2)$$

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

$$H(\omega) = \frac{b_0 + b_1 e^{-j\omega} + b_2 e^{-j2\omega}}{1 + a_1 e^{-j\omega} + a_2 e^{-j2\omega}}$$

7.1 Block Diagram Representation of LCCDE

The focus in this section is on the realization of LTI discrete-time systems in either hardware or software.

We could implement the *N*-th order LCCDE directly using the conventional formula as shown below:

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

And by allowing $a_o = 1$ we have:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$$

Note, however, that there are a number of good reasons why we might want to select a different implementation than the one suggested by the general algorithm.

Some important implementation issues are:

- **Computational complexity**: The # of operations required and how those can be executed
- **Memory requirements**: The # of memory locations required to store system parameters
- **Finite word-length effects**: Quantization and register overflow
- **Availability of specialized processing techniques**: Pipelining or parallel processing

We previously represented systems with:

- Linear Constant Coefficient Difference Equations
- $H(z)$ - the system function
- $h(n)$ - the impulse response

The object now is to convert these into efficient and accurate computational algorithms which can be implemented in software or hardware. The systems, represented graphically, will serve as "schematics" for the discrete-time system.

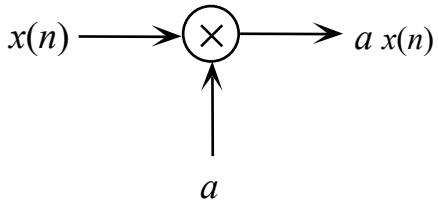
Recall that the LCCDE describes the discrete system in terms of

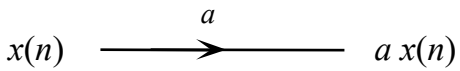
- Adders
- Multipliers
- Delay elements (shift registers)

Here's how we graphically express these elements

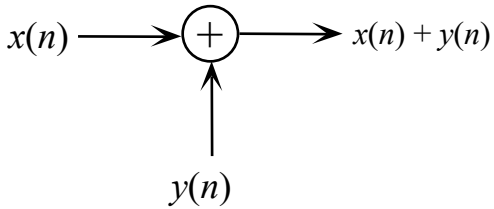
Signal flow-graph representations

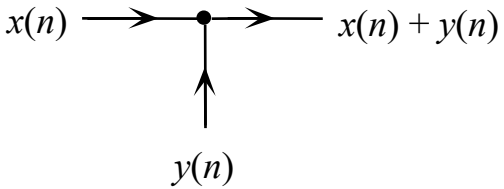
- Multiplier



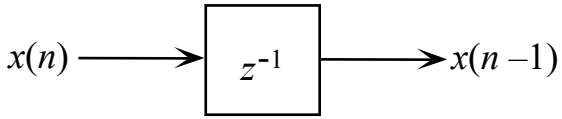


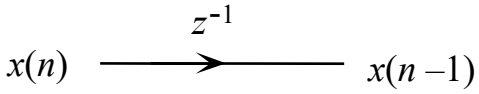
- Adder





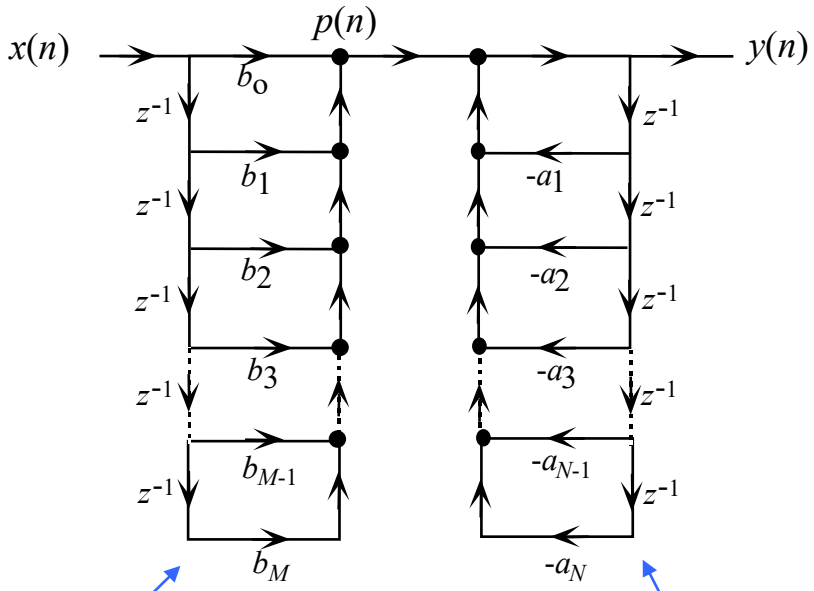
- Delay





Direct Form Structures

The general *N*-th order difference equation can be implemented as shown below:



Direct Form I

$$p(n) = \sum_{k=0}^{M} b_k x(n-k)$$

implements zeroes of the system
- *i.e.* the numerator of $H(z)$

$$y(n) = p(n) - \sum_{k=1}^{N} a_k y(n-k)$$

implements poles of the system
- *i.e.* the denominator of $H(z)$

This implementation is called the Direct Form I implementation. It requires:

N+*M* Delays

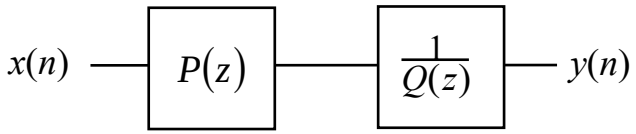
N+*M* Adders

N+*M*+1 Multipliers

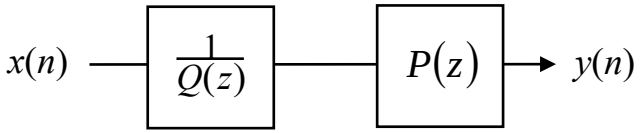
We can develop an equivalent form by using the principle of linearity:

$$H(z)=\frac{P(z)}{Q(z)}=P(z)\cdot\frac{1}{Q(z)}$$

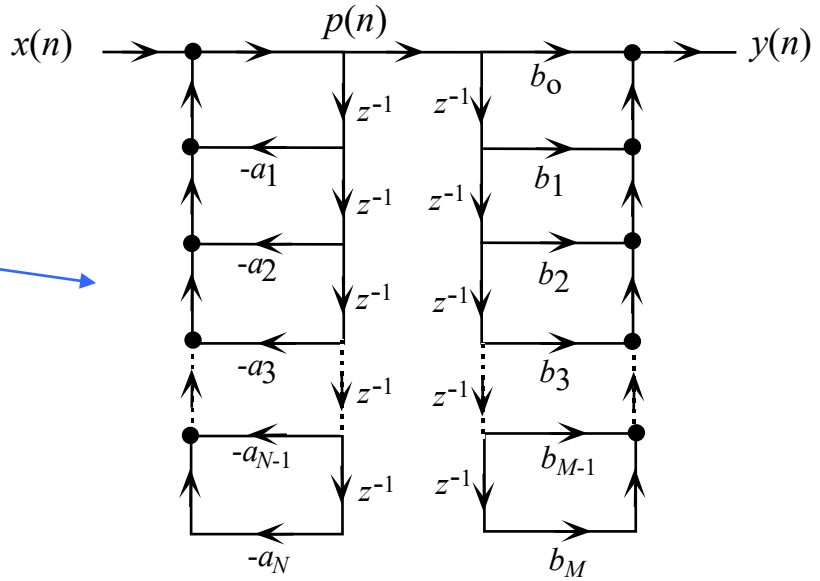
$$Y(z)=\left[P(z)\cdot\frac{1}{Q(z)}\right]X(z)$$ Direct Form I

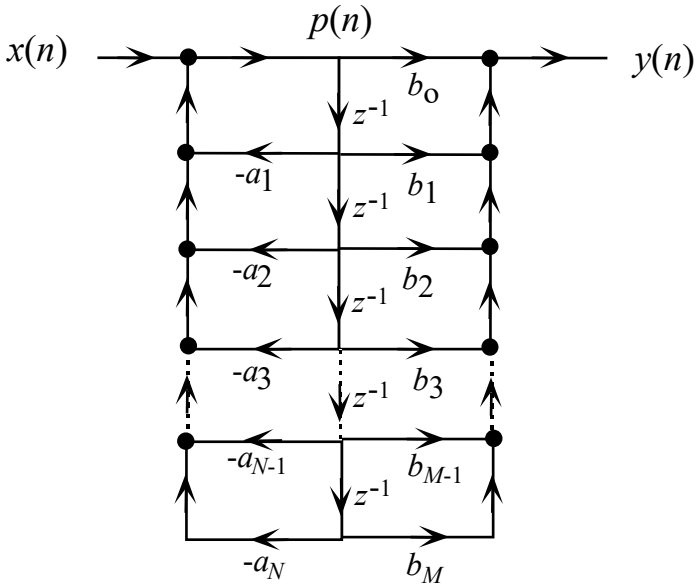


$$Y(z)=\left[\frac{1}{Q(z)}\cdot P(z)\right]X(z)$$ Direct Form II



Interchange pole/zero section in the DF I graph to obtain





one side extends if $N > M$, or vice-versa (as depicted here, $N = M$)

Recognizing duplication in storage, an equivalent structure can be obtained.

This is called the Direct Form II implementation:

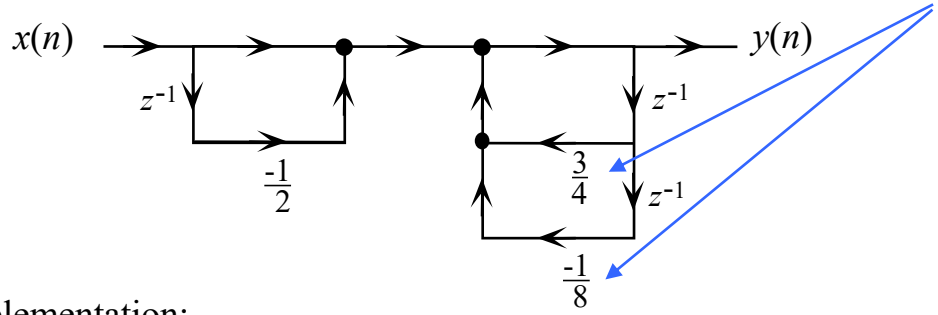
- $N+M$ Adders
- $N+M+1$ multipliers
- max(N, M) delays

Here's an example:

$$H(z)=\frac{1-\frac{1}{2}z^{-1}}{\left(1-\frac{1}{2}z^{-1}\right)\left(1-\frac{1}{4}z^{-1}\right)}=\frac{1-\frac{1}{2}z^{-1}}{1-\frac{3}{4}z^{-1}+\frac{1}{8}z^{-2}}$$

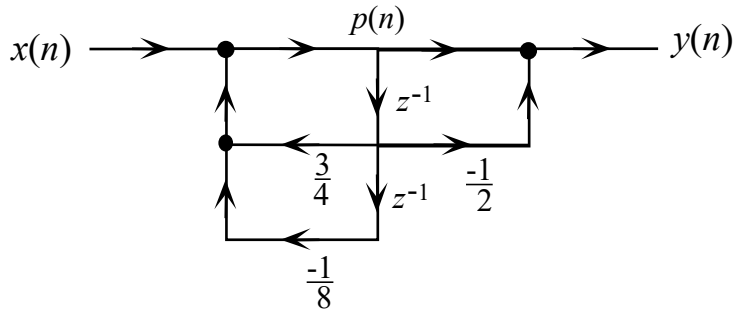
$$y(n)=x(n)-\frac{1}{2}x(n-1)+\frac{3}{4}y(n-1)-\frac{1}{8}y(n-2)$$

The Direct Form I Implementation:



Note that the denominator coefs. are multiplied by -1, (except for $y(n)$ term) since they move to the other side of the equation

The Direct Form II Implementation:



For Direct Form I:

$$y(n) = x(n) - \tfrac{1}{2}x(n-1) + \tfrac{3}{4}y(n-1) - \tfrac{1}{8}y(n-2)$$

For Direct Form II:

$$p(n) = x(n) + \frac{3}{4}p(n-1) - \frac{1}{8}p(n-2) \qquad \text{(A)}$$

$$y(n) = p(n) - \frac{1}{2}p(n-1) \qquad \text{(B)}$$

Verify that these are equivalent:

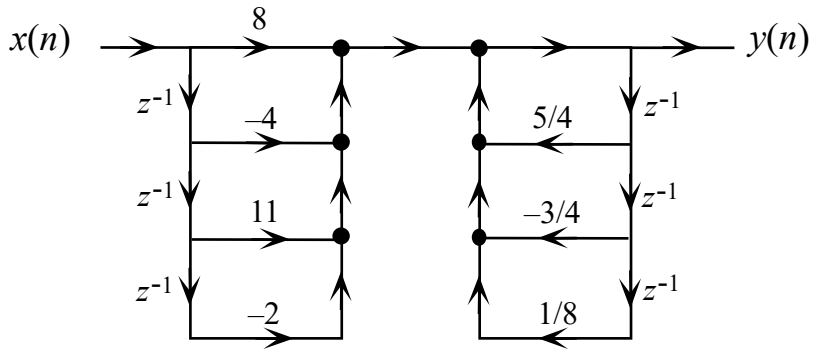
Substitute (A) into (B) and associate terms

$$\begin{aligned} y(n) = \quad & x(n) \quad + \quad \frac{3}{4}p(n-1) \quad - \quad \frac{1}{8}p(n-2) \\ & -\frac{1}{2}x(n-1) \quad \underbrace{- \quad \frac{3}{8}p(n-2)}_{\frac{3}{4}y(n-1)} \quad \underbrace{+ \quad \frac{1}{16}p(n-3)}_{\frac{1}{8}y(n-2)} \end{aligned}$$

based on (B)

Example:

$$H(z)=\frac{8-4z^{-1}+11z^{-2}-2z^{-3}}{1-\frac{5}{4}z^{-1}+\frac{3}{4}z^{-2}-\frac{1}{8}z^{-3}}$$

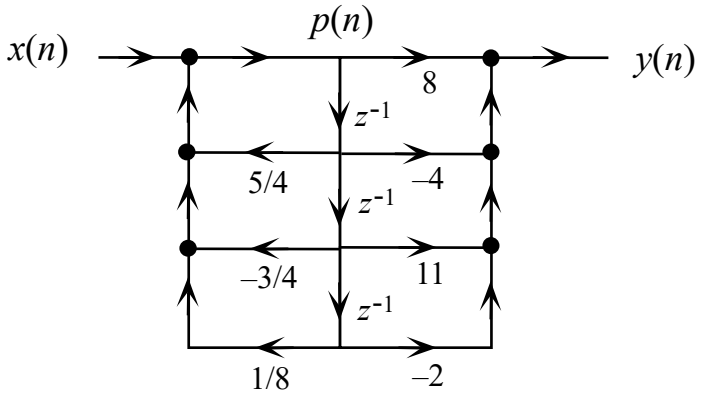


Note that the denominator coefs. are multiplied by -1, (except for $y(n)$ term) since they move to the other side of the equation

The algorithm:

$$y(n)=8x(n)-4x(n-1)+11x(n-2)-2x(n-3)+\frac{5}{4}y(n-1)-\frac{3}{4}y(n-2)+\frac{1}{8}y(n-3)$$

In Direct Form II



The Algorithm

$$p(n) = x(n) + \frac{5}{4}p(n-1) - \frac{3}{4}p(n-2) + \frac{1}{8}p(n-3)$$

$$y(n) = 8p(n) - 4p(n-1) + 11p(n-2) - 2p(n-3)$$

7.3 Basic Structures for IIR Systems

Cascade Structures

Another important representation is the Cascade Form:

begin with $Y(z) = H(z)\,X(z)$

2nd order in numerator and denominator

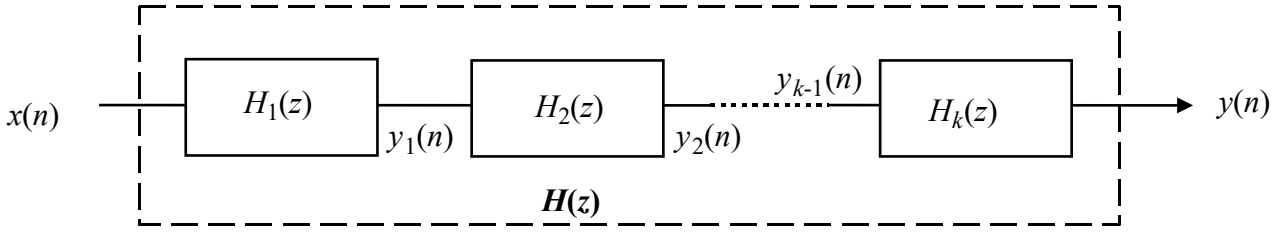
factor $H(z)$ into rational quadratic products:

$$Y(z) = H_k(z)\,H_{k-1}(z)\,\ldots\,H_1(z)\,X(z)$$

each is a quadratic factor of the form:

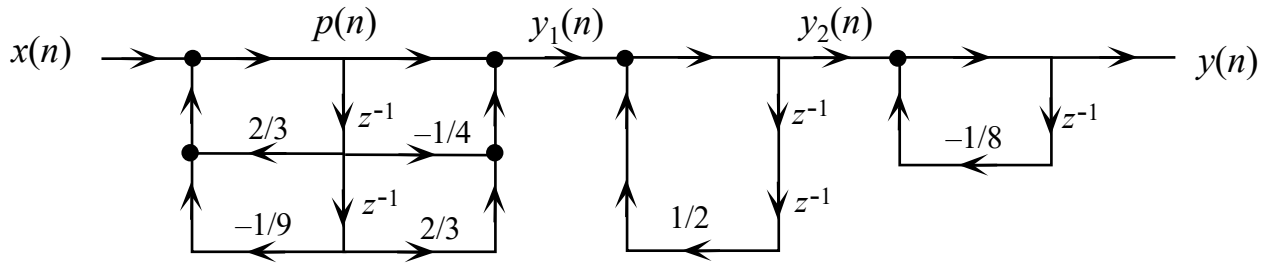
$$H_k(z) = \frac{b_{0,k} + b_{1,k}\,z^{-1} + b_{2,k}\,z^{-2}}{1 + a_{1,k}\,z^{-1} + a_{2,k}\,z^{-2}} \qquad k = 1, 2, \ldots, K$$

this is a "bi-quadratic section"



Example

$$H(z) = \frac{1 - \frac{1}{4} z^{-1} + \frac{2}{3} z^{-2}}{(1 - \frac{1}{2} z^{-2})(1 - \frac{1}{3} z^{-1})^2 (1 + \frac{1}{8} z^{-1})} = \frac{1 - \frac{1}{4} z^{-1} + \frac{2}{3} z^{-2}}{1 - \frac{2}{3} z^{-1} + \frac{1}{9} z^{-2}} \cdot \frac{1}{1 - \frac{1}{2} z^{-2}} \cdot \frac{1}{1 + \frac{1}{8} z^{-1}}$$



The algorithm

$$p(n) = x(n) + \frac{2}{3} p(n-1) - \frac{1}{9} p(n-2)$$

$$y_1(n) = p(n) - \frac{1}{4} p(n-1) + \frac{2}{3} p(n-2)$$

$$y_2(n) = y_1(n) + \frac{1}{2} y_2(n-2)$$

$$y(n) = y_2(n) - \frac{1}{8} y(n-1)$$

Parallel Structures

A final important realization is the Parallel form:

by Partial Fraction Expansion
then recombine in **pairs**

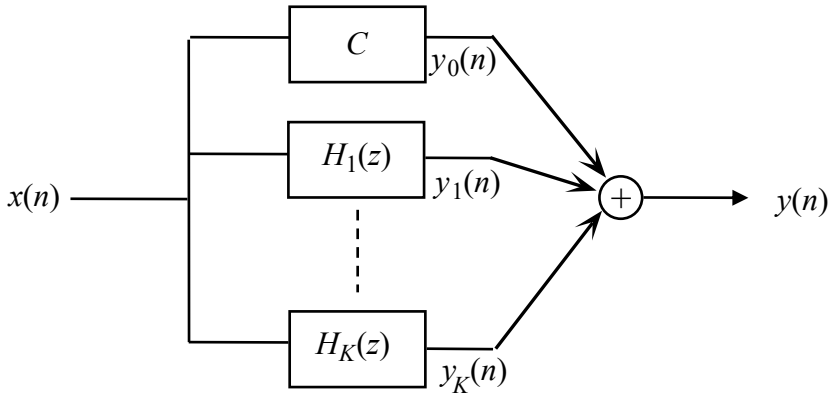
begin with $Y(z) = H(z)\,X(z)$

factor $H(z)$ into rational quadratic sums:

$$Y(z) = \left[C + H_1(z) + H_2(z) + \ldots + H_K(z)\right]X(z) = Y_0(z) + Y_1(z) + Y_2(z) + \ldots + Y_K(z)$$

each is a quadratic factor of the form:

$$H_k(z) = \frac{b_{0,k} + b_{1,k}\,z^{-1}}{1 + a_{1,k}\,z^{-1} + a_{2,k}\,z^{-2}} \qquad k = 1, 2, \ldots, K$$

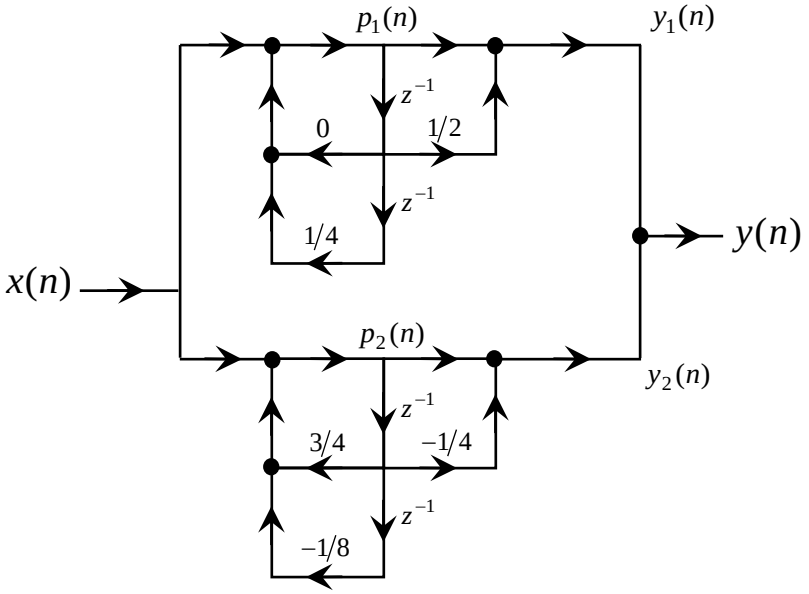


For example:

$$H(z) = \frac{2 - \frac{1}{2}z^{-1} - \frac{1}{2}z^{-2} + \frac{1}{8}z^{-3}}{1 - \frac{3}{4}z^{-1} - \frac{1}{8}z^{-2} + \frac{3}{16}z^{-3} - \frac{1}{32}z^{-4}}$$

Using PFE and then recombining pairs yields:

$$H(z) = H_1(z) + H_2(z) \quad \text{where} \quad H_1(z) = \frac{1 + \frac{1}{2}z^{-1}}{1 - \frac{1}{4}z^{-2}} \quad \text{and} \quad H_2(z) = \frac{1 - \frac{1}{4}z^{-1}}{1 - \frac{3}{4}z^{-1} + \frac{1}{8}z^{-2}}$$



algorithm:

$$p_1(n) = x(n) + \frac{1}{4}p_1(n-2)$$

$$y_1(n) = p_1(n) + \frac{1}{2}p_1(n-1)$$

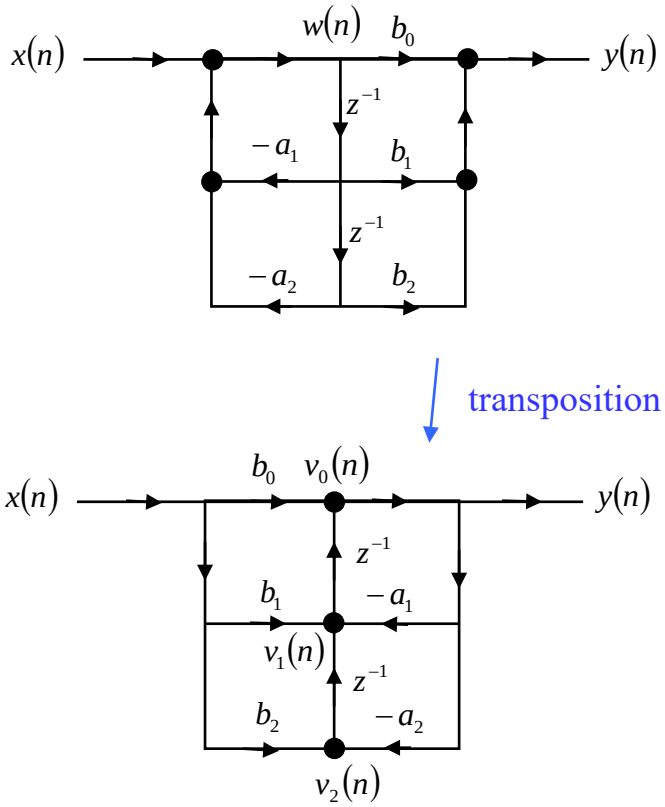
$$p_2(n) = x(n) + \frac{3}{4}p_2(n-1) - \frac{1}{8}p_2(n-2)$$

$$y_2(n) = p_2(n) - \frac{1}{4}p_2(n-1)$$

$$y(n) = y_1(n) + y_2(n)$$

7.4 Transposed Forms

Transposition of a flow graph is accomplished by 1) reversing the directions of all branches in the network while 2) keeping the branch transmittances the same and then 3) reversing the roles of the input and output.



$$w(n) = -a_1 w(n-1) - a_2 w(n-2) + x(n)$$
$$y(n) = b_0 w(n) + b_1 w(n-1) + b_2 w(n-2)$$

Note that the nodes where addition occurs change due to branch directions

$$v_0(n) = b_0 x(n) + v_1(n-1)$$
$$y(n) = v_0(n)$$
$$v_1(n) = -a_1 y(n) + b_1 x(n) + v_2(n-1)$$
$$v_2(n) = -a_2 y(n) + b_2 x(n)$$

Both systems have the same difference equation:

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) + b_0 x(n) + b_1 x(n-1) + b_2 x(n-2)$$

7.5 Basic Network Structures for FIR Systems

For FIR systems the computational algorithm is generally non-recursive, so $H(z)$ is of the form:

$$H(z) = \sum_{k=0}^{M-1} h(k) z^{-k} = \frac{Y(z)}{X(z)}$$

so

$$Y(z) = \sum_{k=0}^{M-1} h(k) z^{-k} X(z)$$

For an impulse response of duration M samples, $H(z)$ will be a polynomial in z^{-1} of order $M - 1$. Therefore $H(z)$ will have $M - 1$ poles at $z = 0$ and $M - 1$ zeros anywhere in the z-plane.

The direct form implementation for FIR systems follows directly from the convolution sum:

$$y(n) = \sum_{k=0}^{M-1} h(k) x(n-k)$$

$$= h(0)x(n) + h(1)x(n-1) + \cdots + h(M-1)x(n-M+1)$$

Which is implemented as:

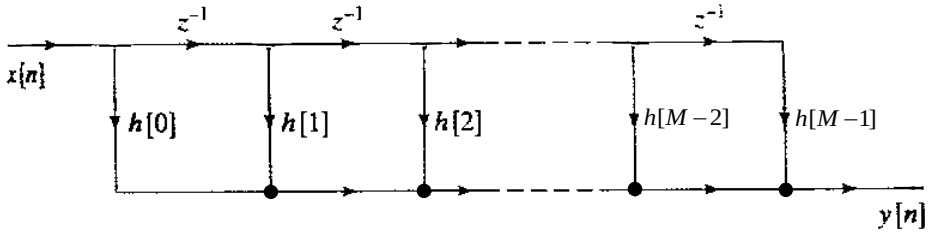


Figure 6.31 Direct-form realization of an FIR system.

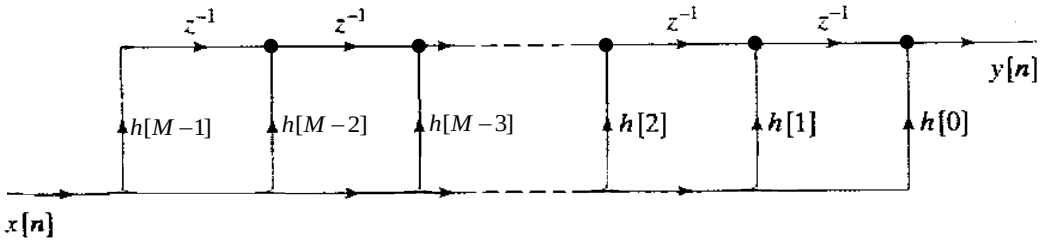
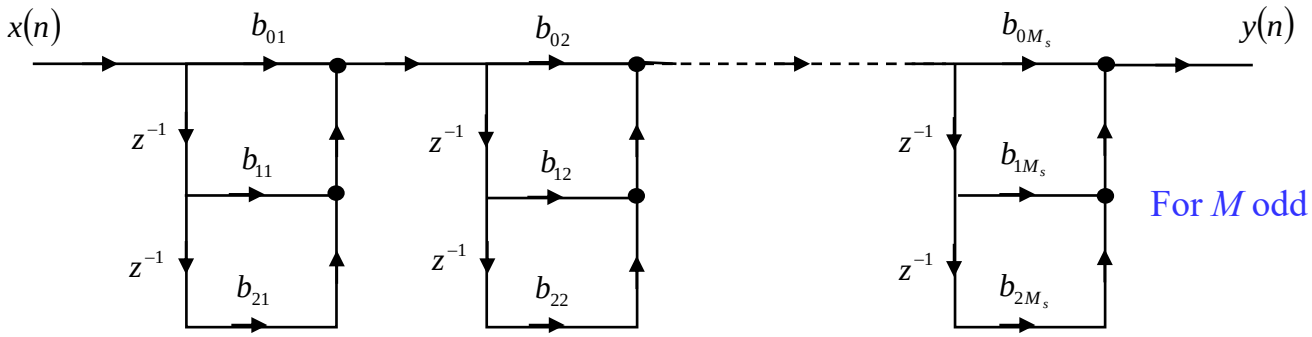


Figure 6.32 Transposition of the network of Figure 6.31.

This is a digital tapped delay line (or transversal) filter. It is the same as a Direct Form I or II for the IIR case with $a_k = 0$.

The cascade form for FIR systems is obtained by factoring the polynomial system function as follows:

$$H(z)=\sum_{n=0}^{M}h(n)z^{-n}=\prod_{k=1}^{M_s}\left(b_{0k}+b_{1k}z^{-1}+b_{2k}z^{-2}\right)$$



For M odd (so order $M-1$ is even)

where $M_s=\lfloor (M-1)/2 \rfloor$

If M is even, there are an odd number of zeros.
Hence, an extra order 1 term is required.

Linear Phase systems allow some economy in reducing the number of coefficient multiplies required

$$y(n) = h(0)x(n) + h(1)x(n-1) + h(2)x(n-2) + h(3)x(n-3)$$

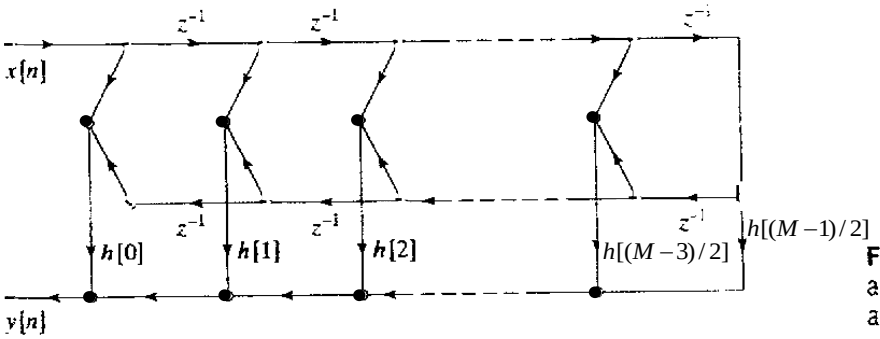


Figure 6.34 Direct-form structure for an FIR linear-phase system when M is an **odd** integer.

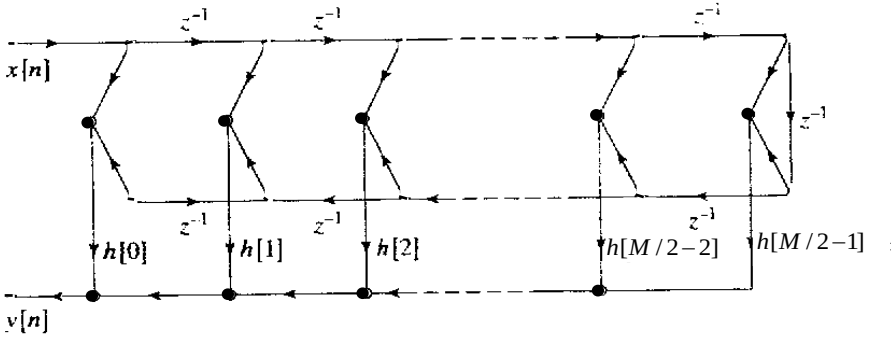
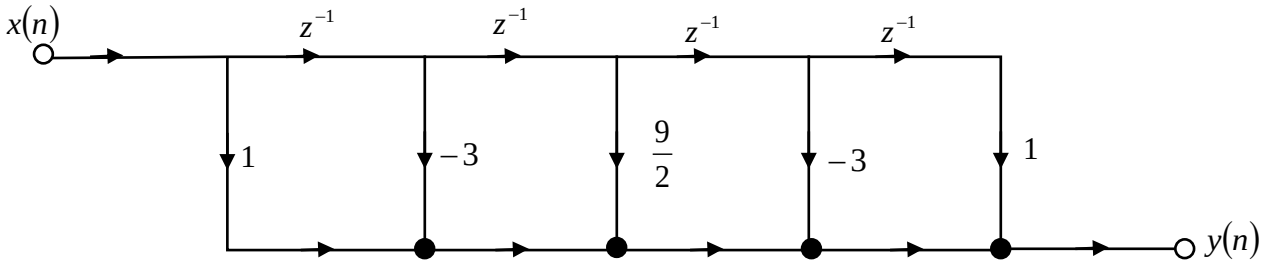


Figure 6.35 Direct-form structure for an FIR linear-phase system when M is an **even** integer.

As an example, consider the following symmetric FIR filter:

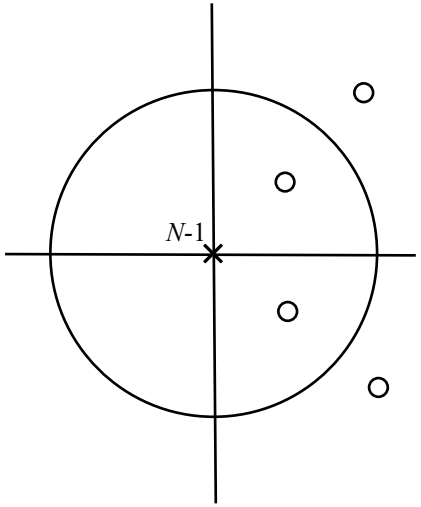


$$y(n) = x(n) - 3x(n-1) + \frac{9}{2}x(n-2) - 3x(n-3) + x(n-4)$$

$$H(z) = \frac{Y(z)}{X(z)} = 1 - 3z^{-1} + \frac{9}{2}z^{-2} - 3z^{-3} + z^{-4}$$

Which can be factored to determine the zero locations:

$$H(z) = [z - (1+j)][z - (1-j)]\left[z - \left(\frac{1-j}{2}\right)\right]\left[z - \left(\frac{1+j}{2}\right)\right]$$



Because it is linear phase, this filter could be rearranged such that only 3 multiplications are needed.

7.6 Overview of Finite-Precision Numerical Effects

There are a number of factors to consider when choosing a filter to implement a desired transfer function:

- Structure with fewest delays and multipliers
- Linear phase requirement
- Complexity

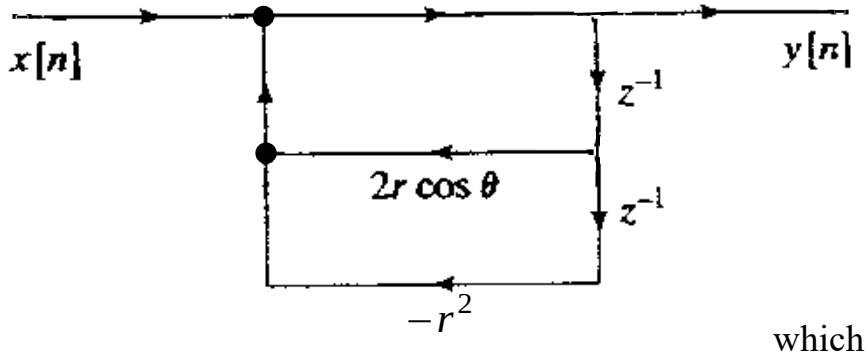
The effect of finite precision is to cause errors in pole-zero locations. Large errors may cause filter design specs to be missed; or in the case of IIR filters, errors may cause poles to move outside the unit circle yielding an unstable and useless system.

How can we predict parameter quantization effects?

The actual effect of coefficient quantization is highly dependent on filter structure employed. Currently no systematic procedure exists for obtaining the "best" realization given constraints of multipliers, delays, and word lengths.

The design approach is to use one of the structures we've discussed and use simulations to analyze quantization effects of a given structure.

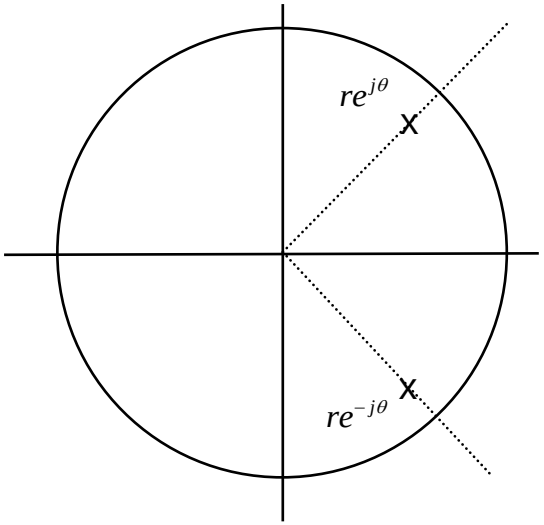
Example – Consider a second order system implemented in direct form



We can show that:

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

which has the following pole-zero diagram

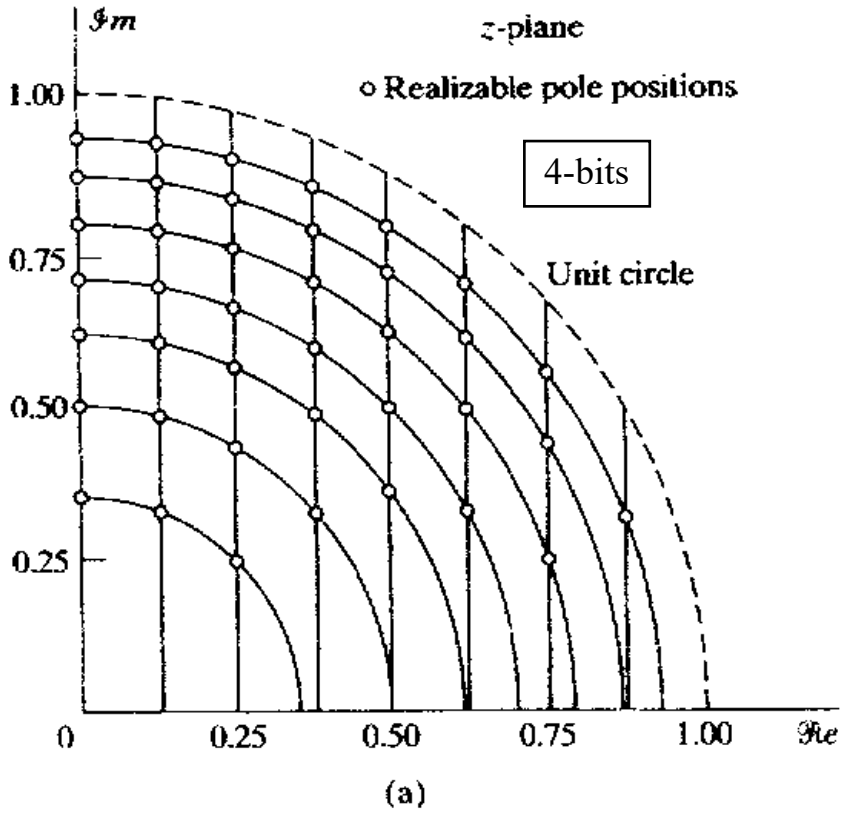


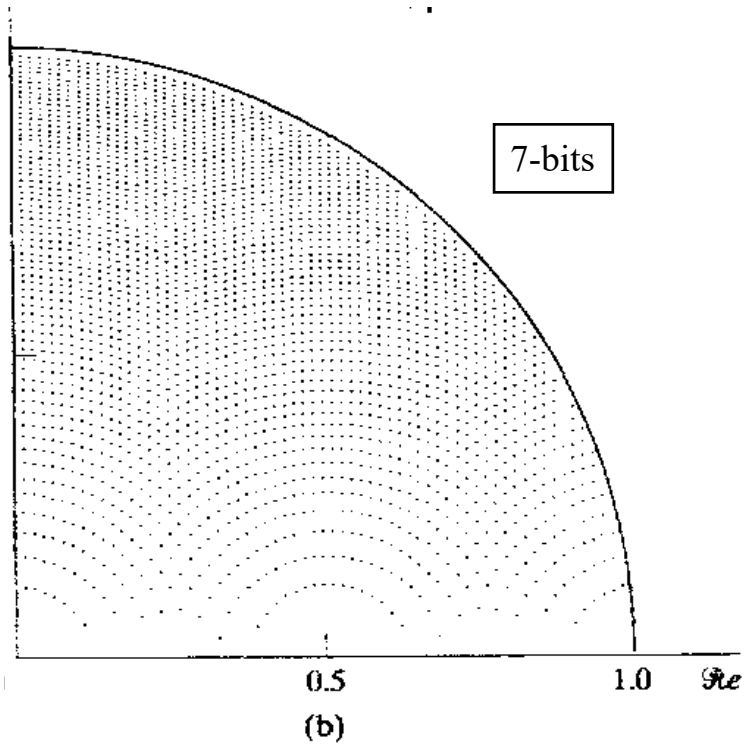
For a causal and stable system, the poles can lie anywhere within the unit circle (in conjugate pairs). That is:

$$\left.\begin{aligned} 0 < r < 1 \\ 0 \le \theta \le \pi \end{aligned}\right\}$$ with infinite precision

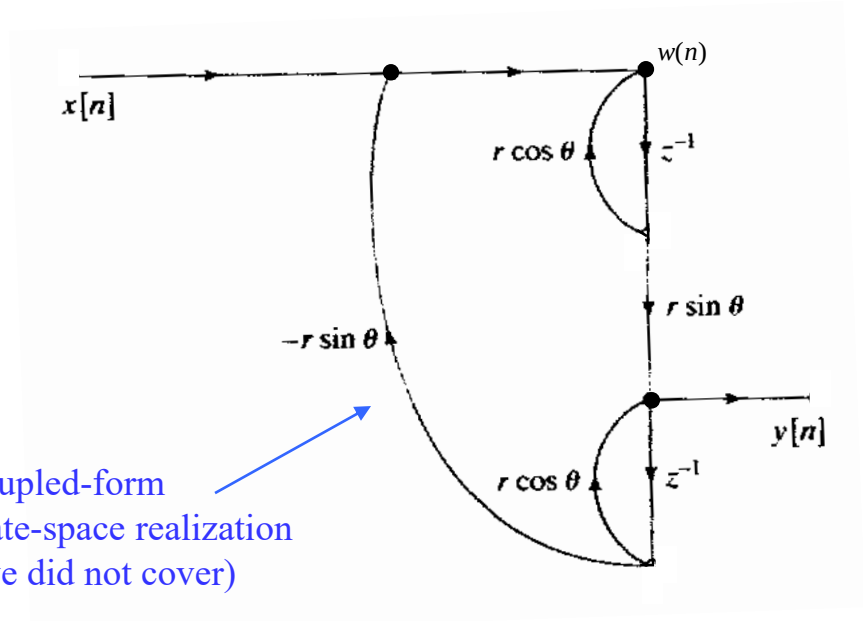
However, if we quantize the coefficients using 16-levels (i.e. 4-bits), and 128-levels (7-bits) the following occurs:

The grid of possible pole locations is a result of quantizing the coefficients: r^2 and $2r\cos\theta$

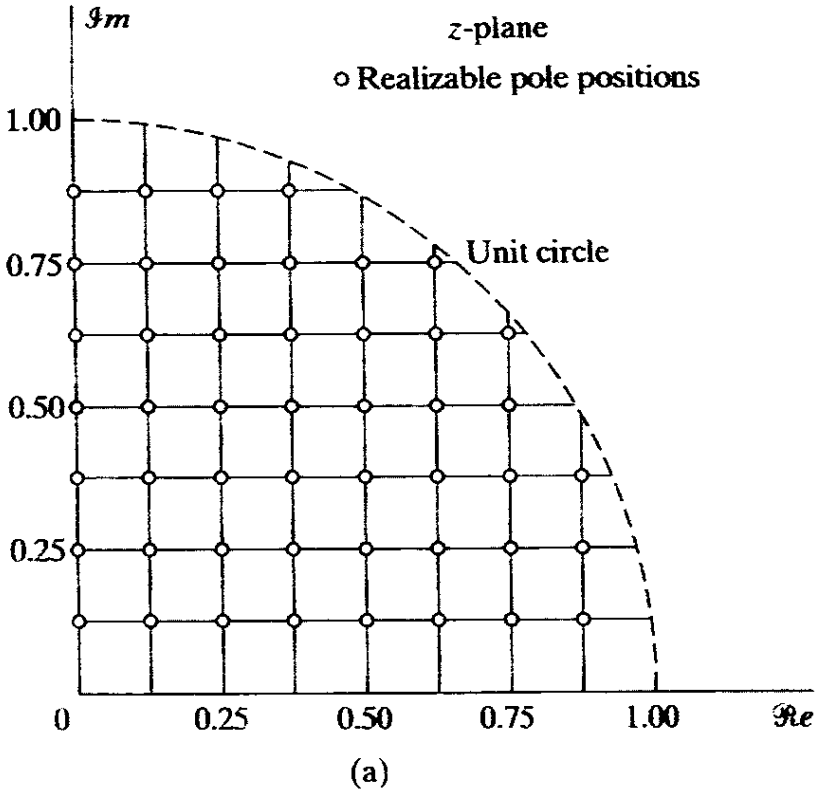




Consider an alternate representation as shown below. The infinite precision filter has the same pole locations as the direct implementation.



coupled-form
state-space realization
(we did not cover)



(a)

algorithm:

$$w(n) = x(n) - r\cos\theta\, w(n-1) - r\sin\theta\, y(n-1)$$

$$y(n) = r\cos\theta\, y(n-1) + r\sin\theta\, w(n-1)$$

Note that with this implementation, the quantization of coefficients using 4-bits results in a uniform grid

$r\cos\theta \rightarrow$ vertical lines

$r\sin\theta \rightarrow$ horizontal lines