2.161 Signal Processing: Continuous and Discrete
Fall 2008

Massachusetts Institute of Technology
Department of Mechanical Engineering

2.161 *Signal Processing - Continuous and Discrete*
Fall Term 2008

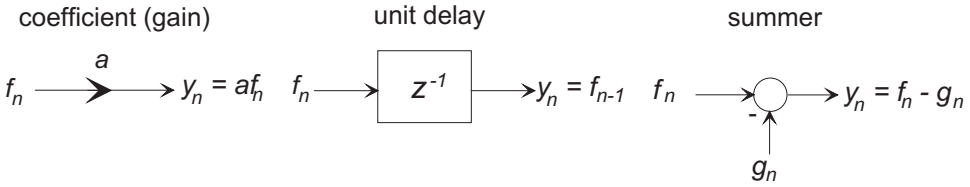
Lecture 20[1]

Reading:

- Class Handout: *Direct-Form Digital Filter Structures*
- Proakis and Manolakis: Sec. 9.1 – 9.3
- Oppenheim, Schafer, and Buck: 6.0 – 6.5

1 Direct-Form Filter Structures

Linear shift-invariant digital filters can be represented in block diagram form in terms of the three primitive elements



2 Transversal FIR Structure

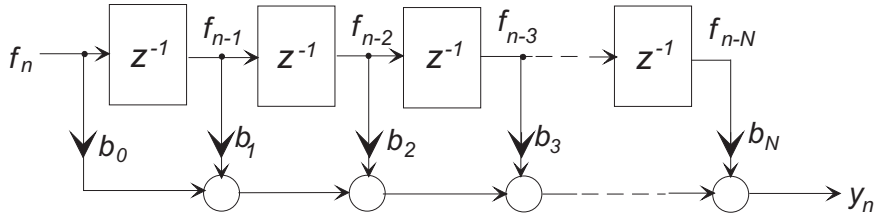
Let the FIR structure to be implemented be

$$H(z) = \sum_{k=0}^{N} b_n z^{-k}$$

so that the difference equation is

$$y_n = \sum_{k=0}^{N} b_k f_{n-k}.$$

The following block diagram is the *transversal* form of this system:



The following MATLAB code implements this structure in a point-by-point filtering function:

```
% ------------------------------------------------------------------------
% 2.161 Classroom Example - firdf - Demonstration FIR Direct Form
%                           implementation.
% Usage :   1) Initialization:
%                       b = [1 2 3 4 5 4 3 2 1];
%                       y = iirdf1('initial', b);
%             where b are the numerator polynomial coefficients.  Example:
%                           y = iirdf1('initial',[1 2 5 2 1]);
%             Note: firdf returns y = 0 for initialization
%           2) Filtering:
%            y_out = firdf(f);
%             where f is a single input value, and
%                   y_out is the computed output value.
%             Example: To compute the step response:
%                   for j=1:100
%                      y(j) = firdf(1);
%                   end
% ------------------------------------------------------------------------
%
function y_n = firdf(f_n,B)
persistent f_register Bx N
%
% The following is initialization, and is executed once
%
if (ischar(f_n) && strcmp(f_n,'initial'))
    N = length(B);
    Bx = B;
    f_register = zeros(1,N);
    y_n = 0;
    else
% Filtering:
    y_n = 0;
    for J = N:-1:2
        f_register(J) = f_register(J-1);
        y_n = y_n + Bx(J)*f_register(J);
    end
    y_n = y_n + Bx(1)*f_n;
    f_register(1) = f_n;
end
```

3 IIR Direct Form Structures

Let the IIR structure to be implemented be

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

where it is assumed that the orders of the numerator and denominator of $H(z)$ are equal. The difference equation is

$$y_n = -\sum_{k=1}^{N} a_k y_{n-k} + \sum_{k=0}^{N} b_k f_{n-k}.$$

Write $H(z)$ as a pair of cascaded sub-systems,

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

where

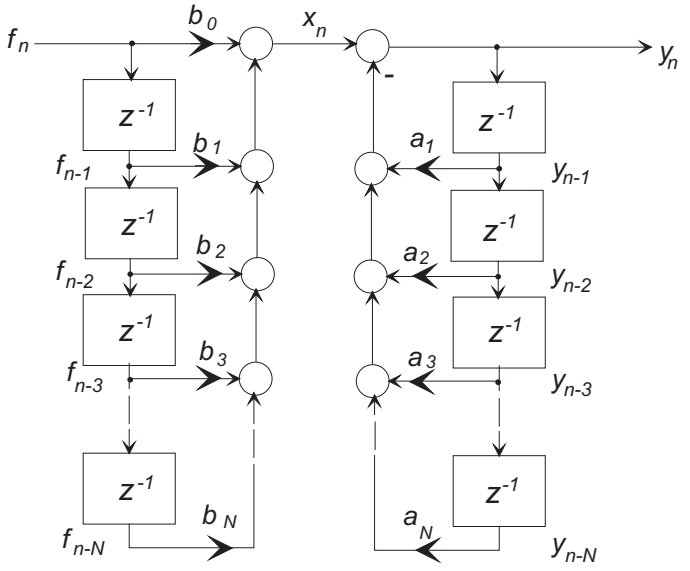
$$H_1(z) = \sum_{k=0}^{N} b_n z^{-k}, \quad \text{and} \quad H_2(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}}.$$

3.1 Direct Form I

Define an intermediate variable x_n, and implement as $X(z) = H_1(z)F(z)$ and $Y(z) = H_2(z)X(z)$, or in difference equation form as

$$\begin{aligned} x_n &= \sum_{k=0}^{N} b_k f_{n-k} \\ y_n &= -\sum_{k=1}^{N} a_k y_{n-k} + x_n \end{aligned}$$

as shown below:



The following MATLAB code implements the Direct Form I structure in a point-by-point filtering function.

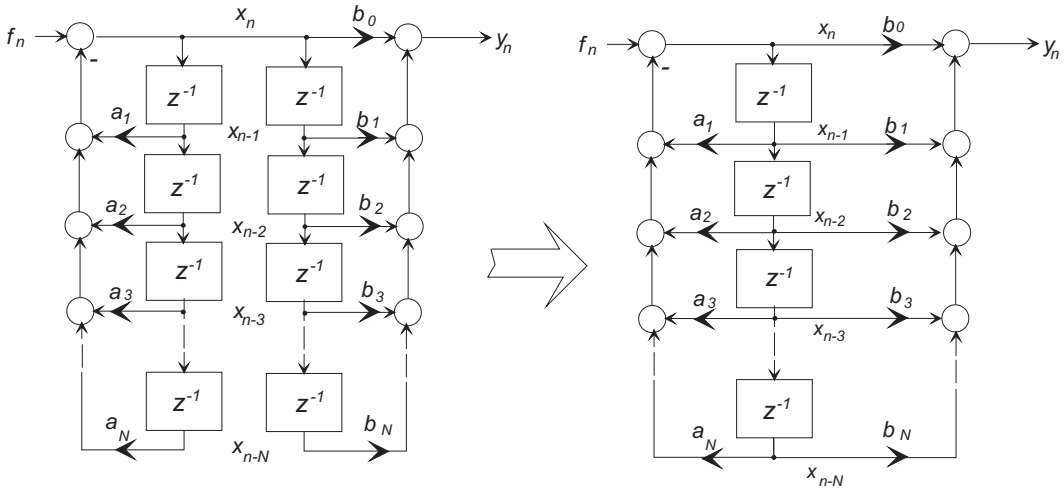
```
% ------------------------------------------------------------------------
% 2.161 Classroom Example - iirdf1 - Demonstration IIR Direct Form I
%                           implementation.
% Usage :   1) Initialization:
%            y = iirdf1('initial', b, a)
%             where b, a are the numerator and denominator polynomial
%             coefficients.  Example:
%                         [b,a] = butter(7,0.4);
%                          y = iirdf1('initial',b,a);
%             Note: iirdf1 returns y = 0 for initialization
%           2) Filtering:
%            y_out = iirdf1(f_{in};
%             where f_in is a single input value, and
%                   y_out is the computed output value.
%             Example: To compute the step response:
%                   for j=1:100
%                      y(j) = iirdf1(1);
%                   end
% ------------------------------------------------------------------------
function y_n = iirdf1(f_n,B,A)
persistent f_register y_register Bx Ax N
%
% The following is initialization, and is executed once
%
if (ischar(f_n) && strcmp(f_n,'initial'))
    N = length(A);
    Ax = A;
    Bx = B;
    f_register = zeros(1,N);
    y_register = zeros(1,N);
    y_n = 0;
    else
% Filtering:  (Note that a Direct Form I filter needs two shift registers.)
    x = 0; y = 0;
    for J = N:-1:2
        y_register(J) = y_register(J-1); % Move along the shift register
        f_register(J) = f_register(J-1);
        y = y - Ax(J)*y_register(J);
        x = x + Bx(J)*f_register(J);
    end
    x = x + Bx(1)*f_n;
    y_n = y + x;
    f_register(1) = f_n;
    y_register(1) = y_n;
end
```

3.2 Direct Form II

The Direct Form II structure results from reversing the order of $H_1(z)$ and $H_2(z)$ so that $X(z) = H_2(z)F(z)$ and $Y(z) = H_1(z)X(z)$, or in difference equation form as

$$\begin{aligned} x_n &= -\sum_{k=1}^{N} a_k f_{n-k} \\ y_n &= \sum_{k=0}^{N} b_k x_{n-k}. \end{aligned}$$

as shown below:



From the left hand figure it can be seen that the values x_{n-k}, $k = 0, \ldots N$, in the two shift registers is equal, and that they can be combined to create the Direct Form II structure, as is shown on the right.

The following MATLAB code implements the Direct Form II structure in a point-by-point filtering function:

```
% ------------------------------------------------------------------------
% 2.161 Classroom Example - iirdf2 - Demonstration IIR Direct Form II
%                           implementation.
% Usage :   1) Initialization:
%            y = iirdf2('initial', b, a)
%             where b, a are the numerator and denominator polynomial
%             coefficients.  Example:
%                         [b,a] = butter(7,0.4);
%                          y = iirdf2('initial',b,a);
%             Note: iirdf2 returns y = 0 for initialization
%           2) Filtering:
%            y_out = iirdf2(f_{in};
%             where f_in is a single input value, and
%                   y_out is the computed output value.
%             Example: To compute the step response:
```

```
%                   for j=1:100
%                      y(j) = iirdf2(1);
%                   end
% ------------------------------------------------------------------------
%
function y_n = iirdf2(f_n,B,A)
persistent register Bx Ax N
%
% The following is initialization, and is executed once
%
if (ischar(f_n) && strcmp(f_n,'initial'))
    N = length(A);
    Ax = A;
    Bx = B;
    register = zeros(1,N);
    y_n = 0;
    else
% Filtering:   (Note that a Direct Form II filter needs only a single
% shift register.)
    x = 0; y = 0;
    for J = N:-1:2
        register(J) = register(J-1);        % Move along the shift register
        x = x - Ax(J)*register(J);
        y = y + Bx(J)*register(J);
    end
    x   = x + f_n;
    y_n = y + Bx(1)*x;
    register(1) = x;
end
```

4 Transposed Direct Forms

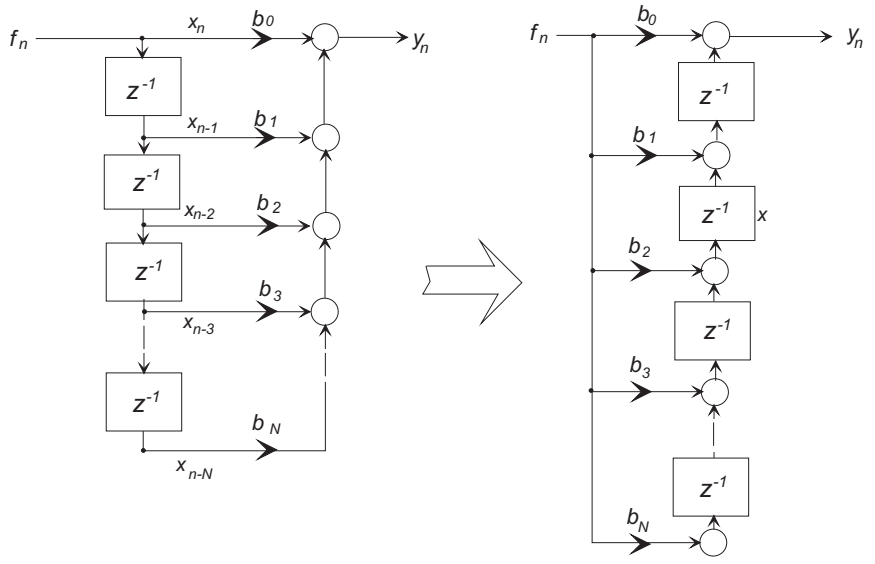
The transposed forms result from the transposition theorem from signal-flow graph theory, which states that in a signal-flow graph if

- The arrows on all graph branches are reversed.
- Branch points become summers, and summers become branch points.
- The input and output are swapped,

then the input/output relationships remain unchanged. The same applies to block diagrams.

4.1 Transposed Transversal FIR Filter

The transposed FIR structure is shown below:

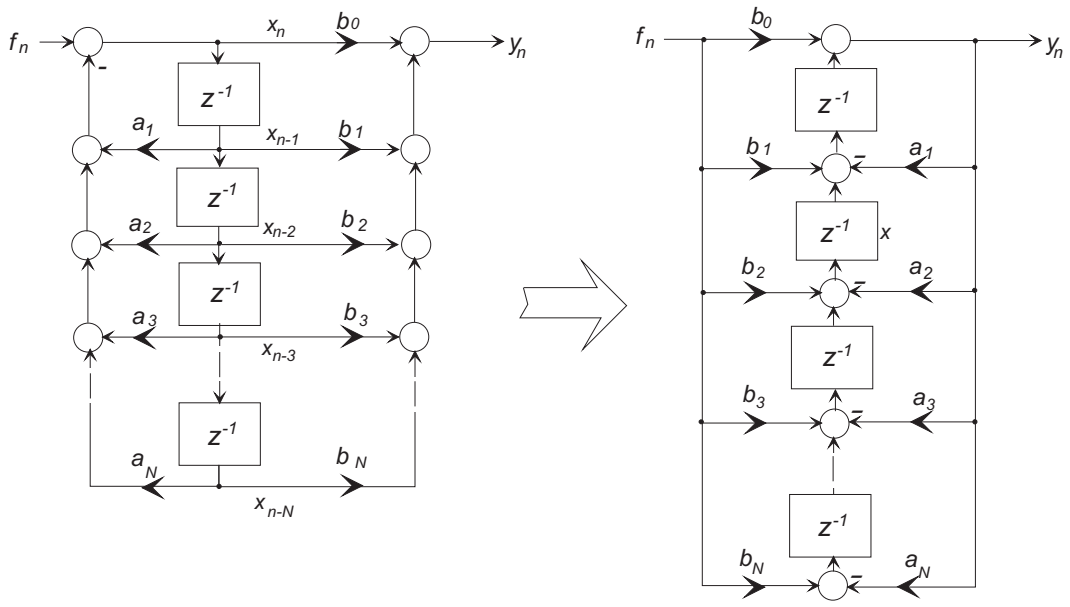



```
% ------------------------------------------------------------------------
% 2.161 Classroom Example - firtdf - Demonstration Transposed FIR Direct
%                                    Form implementation.
% Usage :   1) Initialization:
%                           y = firtdf('initial', b)
%             where b, a are the numerator and denominator polynomial
%             coefficients.  Example:
%                           b = [1 2 3 4 5 4 3 2 1];
%                           y = firtdf('initial',b);
%
%             Note: firtdf returns y = 0 for initialization
%           2) Filtering:
%            y_out = firtdf(f_{in});
%             where f_in is a single input value, and
%                   y_out is the computed output value.
%             Example: To compute the step response:
%                   for j=1:100
%                      y(j) = firtdf(1);
%                   end
% ------------------------------------------------------------------------
%
function y_n = firtdf(f_n,B)
persistent register Bx  N
%
% The following is initialization, and is executed once
%
if (ischar(f_n) && strcmp(f_n,'initial'))
    N = length(B);
    Bx = B;
    register = zeros(1,N-1);
```

```
    y_n = 0;
    else
% Filtering:
    y_n = register(1) + Bx(1)*f_n;
    % Update for the next iteration
    for J = 1:N-2
        register(J) = register(J+1) + Bx(J+1)*f_n;
    end
    register(N-1) =  Bx(N)*f_n;
end
```

4.2 Transposed Direct Form II

The following diagram shows the result when the transposition theorem is applied to a Direct Form II structure.



Direct Form II

Transposed Direct Form II

This block diagram simply reorganizes the difference equation as

$$y_n = b_0 f_n + \sum_{k=1}^{N} (b_k f_{n-k} - a_k y_{n-k})$$

which is implemented in the MATLAB function `iirtdf2()` on the next page.

```
% ------------------------------------------------------------------------
% 2.161 Classroom Example - iirtdf2 - Demonstration Transposed IIR Direct
%                                     Form II implementation.
% Usage :   1) Initialization:
%            y = iirtdf2('initial', b, a)
%             where b, a are the numerator and denominator polynomial
%             coefficients.  Example:
%                         [b,a] = butter(7,0.4);
%                          y = iirtdf2('initial',b,a);
%             Note: iirdf2 returns y = 0 for initialization
%           2) Filtering:
%            y_out = iirtdf2(f_{in};
%             where f_in is a single input value, and
%                   y_out is the computed output value.
%             Example: To compute the step response:
%                   for j=1:100
%                      y(j) = iirtdf2(1);
%                   end
% ------------------------------------------------------------------------
%
function y_n = iirtdf2(f_n,B,A)
persistent register Bx Ax N
%
% The following is initialization, and is executed once
%
if (ischar(f_n) && strcmp(f_n,'initial'))
    N = length(A);
    Ax = A;
    Bx = B;
    register = zeros(1,N-1);
    y_n = 0;
    else
% Filtering:   (Note that a Transposed Direct Form II filter needs only a single
% register.)  Also note that this is not strictly a shift register.
    y_n = register(1) + Bx(1)*f_n;
    % Update for the next iteration
    for J = 1:N-2
        register(J) = register(J+1) + Bx(J+1)*f_n - Ax(J+1)*y_n;
    end
    register(N-1) =  Bx(N)*f_n - Ax(N)*y_n;
end
```

5 Coefficient Sensitivity in Direct Form Filters

In practice high-order IIR Direct Form filters are rarely used because of the sensitivity of pole and zero positions to small perturbations in the values of the coefficients a_k and b_k in the difference

equation. If the transfer function is

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

and the poles are clustered near the unit circle, then small perturbations in any of the a_k from the desired value (perhaps because of finite precision limitations) may cause the filter to become unstable.

To demonstrate this, consider a low-pass filter with

$$A(z) = 1 + \sum_{k=1}^{N} a_k z^{-1} = \prod_{k=1}^{N} \left(1 - p_k z^{-1}\right)$$

where the poles p_k are within the unit circle, but close to $z = 1$, and write $p_k = 1 + \epsilon_k$, where $|\epsilon_k| \ll 1$.

Now let a single (arbitrary) coefficient a_r be be perturbed by δ to

$$a_r' = a_r + \delta$$

so that the denominator polynomial becomes

$$A'(z) = 1 + \sum_{k=1}^{N} a_k z^{-1} + \delta z^{-r}.$$

As $|\delta|$ increases, one or more of the poles may move outside the unit circle, leading to instability. It is difficult to define the general condition, but we can easily find the condition that leads to a pole migrating to $z = 1$, since then

$$A'(1) = A(1) + \delta = 0,$$

that is, there will be a pole at $z = 1$ if

$$\delta = -A(1),$$

or alternatively, if

$$\delta = \prod_{k=1}^{N} (-\epsilon(k).$$

■ Example 1

Consider a low-pass filter

$$H(z) = \frac{1}{(1 - 0.99z^{-1})} = \frac{1}{1 - 2.97z^{-1} + 2.9403z^{-2} - 0.970299z^{-3}}$$

with three poles at $z = 0.99$. Find the perturbation allowed in any coefficient that will create a marginally stable system with a pole at $z = 1$. Discuss some methods of decreasing the sensitivity.

Solution: For the third-order system $A(1) = -10^{-6}$, so any change of $\delta = -A(1) = 10^{-6}$ in *any* coefficient will move one of the poles from $z = 0.99$ to $z = 1$. Any perturbation larger than this will generate an unstable filter.

Now consider the effect of implementing this filter as a cascade connection of two filters, a second-order filter $H_1(z)$, and a first-order filter $H_2(z)$, that is

$$H(z) = H_1(z)H_2(z) = \frac{1}{(1-0.99z^{-1})^2} \cdot \frac{1}{1-z^{-1}}$$

with a pair of difference equations

$$\begin{aligned} x_n &= 1.98x_{n-1} - 0.9801x_{n-2} + f_n \\ y_n &= 0.99y_{n-1} + x_n. \end{aligned}$$

For $H_1(z)$, $A_1(1) = -10^{-4}$, while for $H_2(z)$ $A_2(1) = -10^{-2}$ and the sensitivity is significantly reduced.

If the filter is implemented as a cascade connection of three first-order filters,

$$H(z) = H_1(z)H_2(z)H(_3z)(z) = \frac{1}{1-z^{-1}} \cdot \frac{1}{1-z^{-1}} \cdot \frac{1}{1-z^{-1}}$$

with a set of difference equations

$$\begin{aligned} w_n &= 0.99w_{n-1} + f_n \\ v_n &= 0.99v_{n-1} + w_n \\ y_n &= 0.99y_{n-1} + v_n, \end{aligned}$$

for any of the first-order sections $H_k(z)$, $A_1(1) = -10^{-2}$, and the coefficient sensitivity is significantly reduced even further.

This example demonstrates that the sensitivity to coefficient precision can be often drastically reduced by implementing a filter with low order sections.

5.1 Cascade Structures

If the transfer function is written in terms of its poles and zeros

$$H(z) = \frac{\prod_{K=1}^{M_1}(1-e_k z^{-1}) \prod_{K=1}^{M_2}(1-g_k z^{-1})(1-\overline{g}_k z^{-1})}{\prod_{K=1}^{N_1}(1-c_k z^{-1}) \prod_{K=1}^{N_2}(1-d_k z^{-1})(1-\overline{d}_k z^{-1})}$$

where the c_k and e_k are real poles and zeros, and d_k, $\overline{d}_k$ and g_k, $\overline{g}_k$ are complex conjugate pole and zero pairs, it is common to realize the system as a cascade chain of first- and second-order sections (usually Direct Form II):

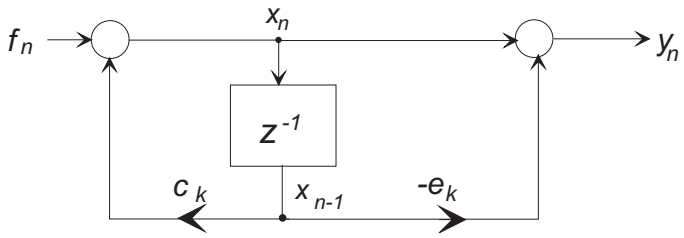
A first-order Direct Form II section, implementing a real pole c_k and zero e_k

$$H_k(z) = \frac{1-e_k z^{-1}}{1-c_k z^{-1}},$$

as

$$y_n = c_k y_{n-1} + f_n - e_k f_{n-1}$$

is shown below



A second-order Direct Form 2 section, implementing a conjugate pole pair

$$d_k, \overline{d}_k = r\,\mathrm{e}^{\pm \mathrm{j}\,\theta}$$

has a denominator polynomial

$$(1 - d_k z^{-1})(1 - \overline{d}_k z^{-1}) = 1 - 2r\cos(\theta)z^{-1} + r^2 z^{-2}$$

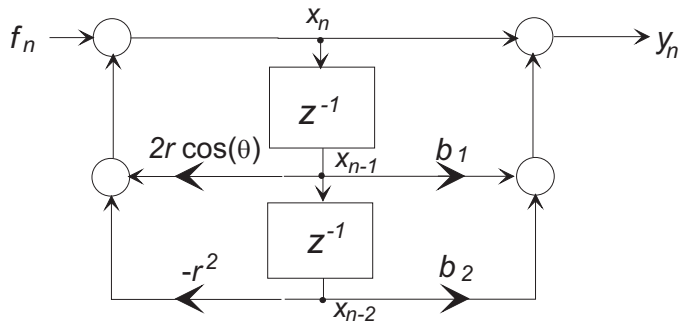
and when paired with a pair of zeros (either real, or a complex conjugate pair) to give a transfer function

$$H(z) = \frac{1 - b_1 z^{-1} + b_2 z^{-2}}{1 - 2r\cos(\theta)z^{-1} + r^2 z^{-2}}$$

and difference equation

$$y_n = 2r\cos(\theta)y_{n-1} + r^2 y_{n-2} + f_n + b_1 f_{n-1} + b_2 f_{n-2}$$

is shown below



■ Example 2

Implement the system

$$H(z) = \frac{0.04756z^3 + 0.14273z^2 + 0.14273z + 0.04756}{z^3 - 1.3146z^2 + 1.17043z - 0.47524}$$

as a set of cascaded first- and second-order systems.

Solution: Factor the transfer function and rewite as

$$H(z) = \frac{0.04756(1 + z^{-1})^3}{(1 - 0.6711z^{-1} + 0.7386z^{-2})(1 - 0.6435z^{-1})}.$$

Implement the filter as a cascaded pair

$$\begin{aligned} H_1(z) &= \frac{0.04756(1+2z^{-1}+z^{-2})}{1-0.6711z^{-1}+0.7386z^{-2}} \\ H_2(z) &= \frac{1+z^{-1}}{1-0.6435z^{-1}} \end{aligned}$$

with a pair of difference equations

$$\begin{aligned} x_n &= 0.6711x_{n-1} - 0.7386x_{n-2} + 0.04756(f_n + 2f_{n-2} + f_{n-2}) \\ y_n &= 0.6435y_{n-1} + x_n + x_{n-1}. \end{aligned}$$

There is a lot of flexibility in choosing which zeros to associate with the poles of each low order section, and how to distribute the overall gain between the sections. A general (rule-of-thumb) procedure is

(1) Select the poles closest to the unit circle.

(2) Find the closest zeros to those poles.

(3) Combine into a second-order section.

(4) Repeat until all zeros are accounted for.

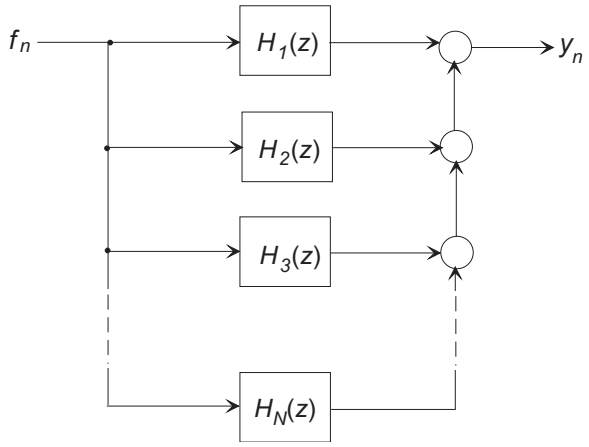
5.2 Parallel Structures

A high order filter may also be realized as a set of parallel second- and first-order sections using partial fractions, and implemented as

$$H(z) = H_1(z) + H_2(z) + H_3(z) \ldots + H_N(z)$$

so that

$$Y(z) = (H_1(z) + H_2(z) + H_3(z) \ldots + H_N(z))\, F(z)$$



■ Example 3

Implement the system of Example 2 as a parallel realization

$$H(z) = \frac{0.04756z^3 + 0.14273z^2 + 0.14273z + 0.04756}{z^3 - 1.3146z^2 + 1.17043z - 0.47524}$$

Solution: Using a partial fraction expansion

$$H(z) = 0.0476 + \frac{0.2929}{z - 0.6435} - \frac{.0877z - 0.2271}{z^2 - 0.6711z + 0.7386}.$$

Implement as three sections

$$\begin{aligned} H_1(z) &= 0.0476 \\ H_2(z) &= \frac{0.2929z^{-1}}{1 - 0.6435z^{-1}} \\ H_3(z) &= -\frac{.0877z^{-1} - 0.2271z^{-2}}{1 - 0.6711z^{-1} + 0.7386z^{-2}} \end{aligned}$$

with difference equations

$$\begin{aligned} u_n &= 0.0476f_n \\ v_n &= 0.6435v_{n-1} + 0.2929f_{n-1} \\ w_n &= 0.6711w_{n-1} - 0.7386w_{n-2} + 0.0877f_{n-1} - 0.2271f_{n-2} \\ y_n &= u_n + v_n - w_n \end{aligned}$$