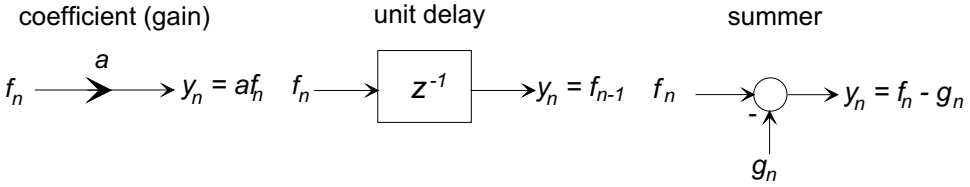
2.161 Signal Processing: Continuous and Discrete
Fall 2008

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING
2.161 Signal Processing - Continuous and Discrete

Direct Form Digital Filter Structures [1]

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



In this handout we examine common structures for FIR and IIR filters, and give tutorial MATLAB functions to implement these structures.

1 Transversal FIR Structure

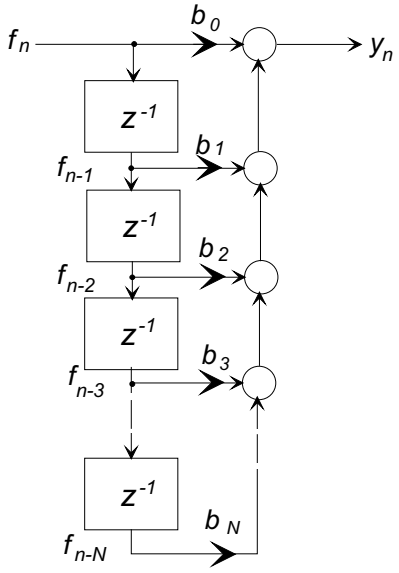
Let the FIR structure to be implemented be

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

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:

[1] D. Rowell November 18, 2008

```
% ------------------------------------------------------------------------
% 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
```

2 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

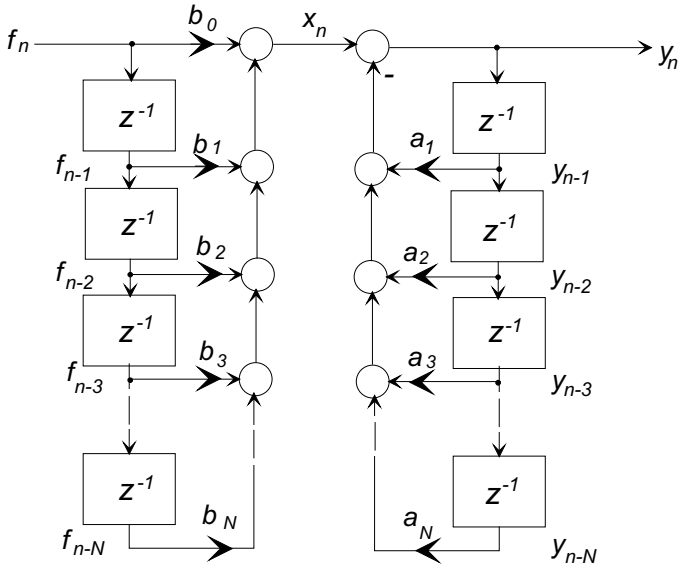
$$H_1(z) = \sum_{k=0}^{N} b_n z^{-k}, \quad \text{and} \quad H_2(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}}.$$

2.1 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.

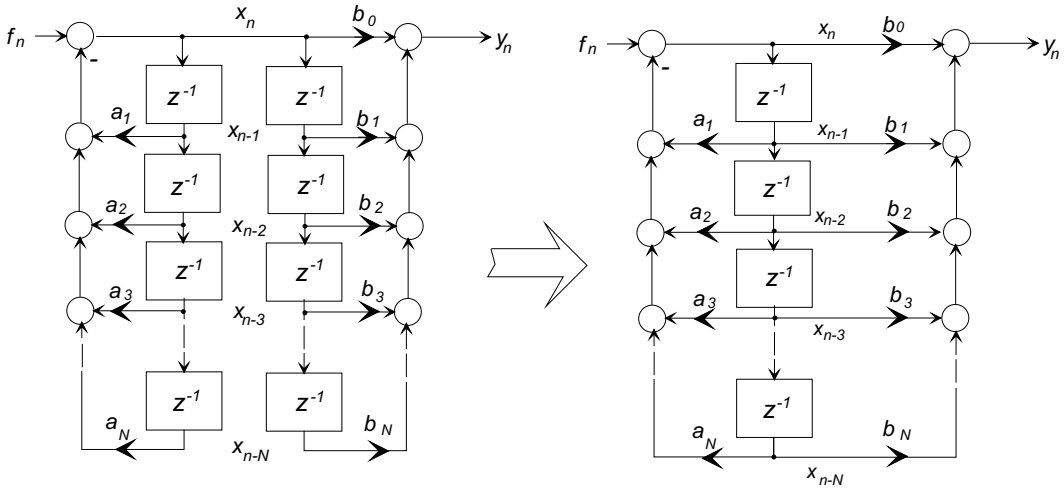
```
% ------------------------------------------------------------------------
% 2.161 Classroom Example - iirdf1 - Demonstration IIR Direct Form I
%                           implementation.
% Usage :   1) Initialization:
%            y = iirdf1('initial', b, a)
%             where b, a are the numerator 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
```

2.2 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:



Notice that only a single delay register is required for the Direct Form II structure, as is shown on the right.

The following MATLAB code on the following page 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
```

3 Transposed Direct Forms

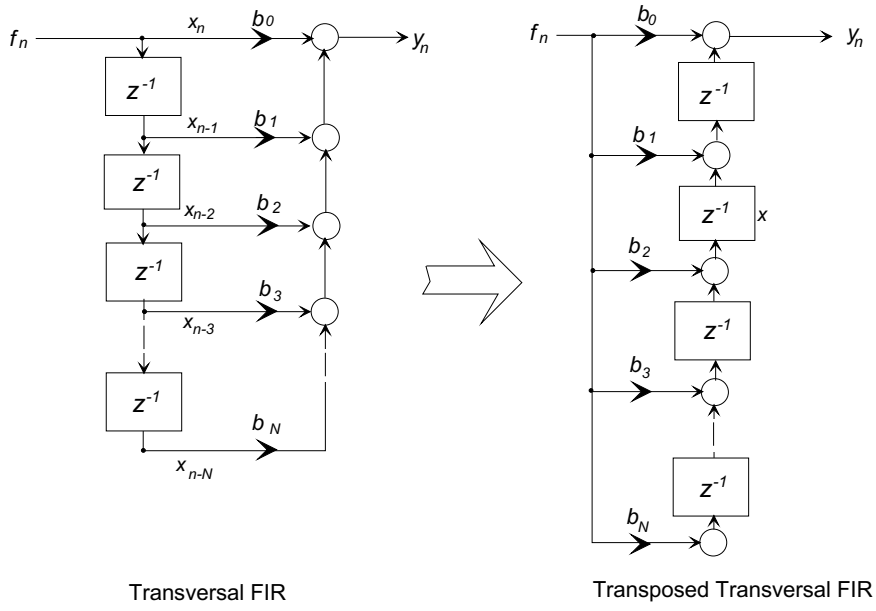
The transposed forms result from the transposition theorem from signal-flow graph theory, which states that in a signal-flow graph if

- 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.

3.1 Transposed Transversal FIR Filter

The transposed FIR structure is shown below:



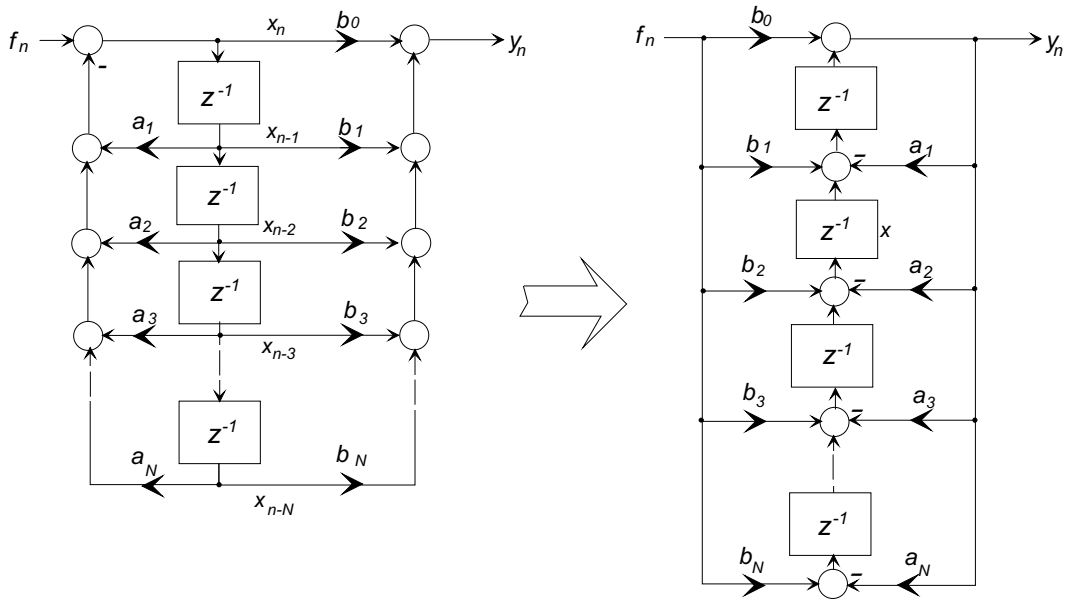
Transversal FIR

Transposed Transversal FIR

```
% ------------------------------------------------------------------------
% 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:   (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;
    end
    register(N-1) =  Bx(N)*f_n;
end
```

3.2 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
```

Example:

```
%-------------------------------------------------
% Test program for IIR Structure functions
% Compute and plot the step and impulse responses of a low-pass filter
%-------------------------------------------------
[b,a] = butter(7,0.4);
%
% Compute the step response using iirdf2()
y_step = zeros(1,30);
% Initialize by passing in the numerator and denominator polynomials
y = iirdf2('initial',b, a);
% Compute one point for each call to iirdf2()
for j = 1:30
     y_step(j) = iirdf2(1);
end
% Similarly, compute the impulse response using iirdf2()
y = iirtdf2('initial',b, a);
y_imp = zeros(1,30);
for j=1:30
    if j == 1
    y_imp(j) = iirtdf2(1);
    else
        y_imp(j) = iirtdf2(0);
    end
end
subplot(2,1,1), stem((0:29),y_step);
title('Step Response')
subplot(2,1,2), stem((0:29),y_imp);
title('Impulse Response')
```