Pengolahan Sinyal Digital

Adhi Harmoko Saputro

Report Structure

- **Introduction:** A general description of the area of your project and why you're doing it.
- **Problem Specification:** A clear and succinct technical description of the problem you're addressing. Formulating a general problem (e.g., transcribing music) into a well-defined technical goal (e.g., reporting a list of estimated fundamental periods at each time frame) is often the most important part of a project.
- **Data:** What are the real-world and/or synthetic signals you are going to use to develop and evaluate your work?
- **Evaluation Criteria:** How are you going to measure how well your project performs? The best criteria are objective, quantitative, and discriminatory. You want to be able to demonstrate and measure improvements in your system.
- **Approach:** A description of how you went about trying to solve the problem. Sometimes you can make a nice project by contrasting two or more different approaches.

Report Structure

- **Results and Analysis:** What happened when you evaluated your system using the data and criteria introduced above? What were the principal shorfalls? (This may require you to choose or synthesize data that will reveal these shortcomings.) Your analysis of what happened is one of the most important opportunities to display your command of signal processing concepts.
- **Development:** If possible, you will come up with ideas about how to improve the shortcomings identified in the previous section, and then implement and evaluate them. Did they, in fact, help? Were there unexpected side-effects?
- **Conclusions:** What did you learn from doing the project? What did you demonstrate about how to solve your problem?
- **References:** Complete list of sources you used in completing your project, with explanations of what you got from each.

Some project suggestions

- **DTMF decoder -** convert a recording of 'touch tones' from a real telephone into the corresponding set of digits.
- **Channel equalization**. This is a classic signal processing problem, where the signal has been subject to some unknown filter, and the goal is to infer what that filter was, then invert its effects.
- **Signal denoising**. Signals get corrupted by noise all kinds of ways - by electrical interference, by mechanical damage of recording media, or simply because there were unwanted sounds present during the original recording.
- **Speech endpointing -** find the beginning and end of each speech phrase or utterance in a recording (which may include background noise)

Some project suggestions

- **Speech/music discrimination -** classify example fragments as speech, music, or some other class
- **Pitch extraction**. This is a widespread problem in speech and music processing: identifying the local periodicity of a sound with a perceived pitch. Autocorrelation is the classic method, but it makes a lot of common errors, so there are many approaches to improving it.
- **Modeling musical instruments**. It turns out that many musical instruments can be modeled with surprisingly good quality by relatively simple signal processing networks.

Some project suggestions

- **Timescale modification**. In the very first lecture I played an example of speech that had been slowed down *without* lowering the pitch.
- **Sound visualization**. We've seen the spectrogram as an example of rendering a sound as an image. However, there are very many parameters to vary, with pros and cons to each variation. This project would choose a particular goal, say a clear of a certain kind of sound in a variety of backgrounds, then investigate the best possible processing to facilitate that display.

Some project suggestions

- **Steganography/watermarking**. There are various motivations for 'hiding' data in a soundfile without making the alteration audible. One is watermarking, so that a sound can be recognized as 'valid' without knowing its content ahead of time. Another is embedding copyright markers that cannot easily be removed by counterfeiters. This project will investigate some mechanisms for encoding data in sound, and examine the limits of how much data can be included, what degradation to sound quality is entailed, and how hard it is to remove, or to simulate, the marking.

Some project suggestions

- **Artificial reverberation**. I mentioned that one use of allpass filters is in the simulation of room reverberation. This project will build a simulated room reverberator and investigate several enhancements to increase the realism. There are several good papers to start from.
- **Compression**. Audio signal compression is a current hot topic. This project would involve implementing one or more simple compression schemes, and investigating how they perform for different kinds of signals, as well as the kinds of distortion they introduce.

Some project suggestions

- **Time-delay angle-of-arrival estimation**. We use our two ears to be able to detect the direction from which a sound arrives. The strongest cue is probably the slight time differences that occur due to the finite speed of sound traveling to each side. Cross-correlation can reveal this time difference, and indicate the azimuth from which sounds occur.
- **Doubletalk detection**. In speech processing we frequently assume that the signal contains just a single voice; in many cases this is not true, for instance when people interrupt one another on a telephone call or in a meeting. Separating these voices is hard, but we would like at least to be able to detect when it is happening, so we know not to attempt normal processing.

Some project suggestions

- **Synthesizing 3D sound**. Since we have some understanding of how the ear uses binaural (stereo) cues to infer the direction of different sound sources, we should be able to construct artificial sounds including those cues that will appear to come from particular directions
- **Cross-synthesis**. An interesting effect in electronic music synthesis is to somehow 'combine' two sounds into a single sound that appears to have properties of both sources. This project will implement some variants of how this can be done.

Implementation of Discrete-time Filters

Introduction

- To process signals, we have to design and implement systems called *filters* (or spectrum analyzers in some contexts)
- The filter design issue is influenced by such factors as the type of the filter (i.e., IIR or FIR) or the form of its implementation (structures).
- IIR filters as designed and used in DSP, can be modeled by rational system functions or, equivalently, by difference equations.

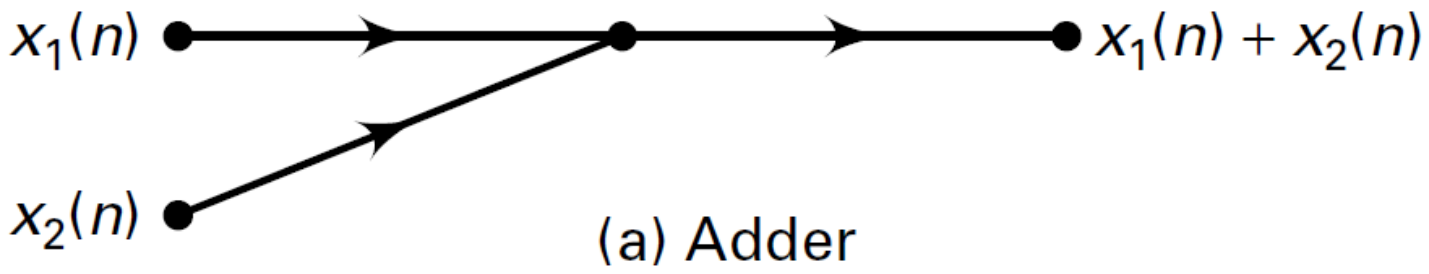
Introduction

- We begin to consider problems associated with quantization effects when finite-precision arithmetic is used in the implementation.
- Digital hardware contains processing elements that use finite-precision arithmetic. When filters are implemented either in hardware or in software, filter coefficients as well as filter operations are subjected to the effects of these finite-precision operations

Basic Elements

Basic Elements

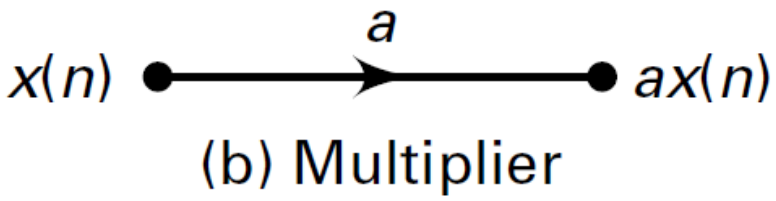
- **Adder:**
  - This element has two inputs and one output
  - Note that the addition of three or more signals is implemented by successive two-input adders.



(a) Adder

Basic Elements

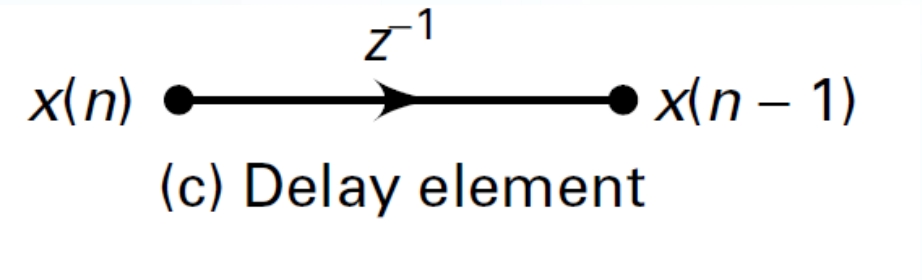
- **Multiplier (gain):**
  - This is a single-input, single-output element
  - Note that the multiplication by 1 is understood and hence not explicitly shown



(b) Multiplier

Basic Elements

- **Delay element (shifter or memory):**
  - This element delays the signal passing through it by one sample
  - It is implemented by using a shift register



(c) Delay element

IIR Filter Structures

IIR Filter Structures

- The system function of an IIR filter is given by

$$H(z)=\frac{B(z)}{A(z)}=\frac{\sum_{n=0}^{M} b_n z^{-n}}{\sum_{n=0}^{N} a_n z^{-n}}=\frac{b_o+b_1 z^{-1}+\ldots+b_M z^{-M}}{1+a_1 z^{-1}+\ldots+a_N z^{-N}};\quad a_o=1$$

  - b_n and a_n are the coefficients of the filter
  - The order of such an IIR filter is called N if $a_N \neq 0$.

IIR Filter Structures

- The difference equation representation of an IIR filter is expressed as

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{m=1}^{N} a_m y(n-m)$$

IIR Filter Structures

- Three different structures can be used to implement an IIR filter
  1. **Direct form:** In this form the difference equation is implemented directly as given. There are two parts to this filter, namely the moving average part and the recursive part (or equivalently, the numerator and denominator parts). Therefore this implementation leads to two versions: direct form I and direct form II structures.
  2. **Cascade form:** In this form the system function $H(z)$ is factored into smaller 2nd-order sections, called *biquads*. The system function is then represented as a *product* of these biquads. Each biquad is implemented in a direct form, and the entire system function is implemented as a *cascade* of biquad sections.

IIR Filter Structures

- Three different structures can be used to implement an IIR filter
  3. **Parallel form:** This is similar to the cascade form, but after factorization, a partial fraction expansion is used to represent $H(z)$ as a *sum* of smaller 2nd-order sections. Each section is again implemented in a direct form, and the entire system function is implemented as a *parallel* network of sections..

Direct form

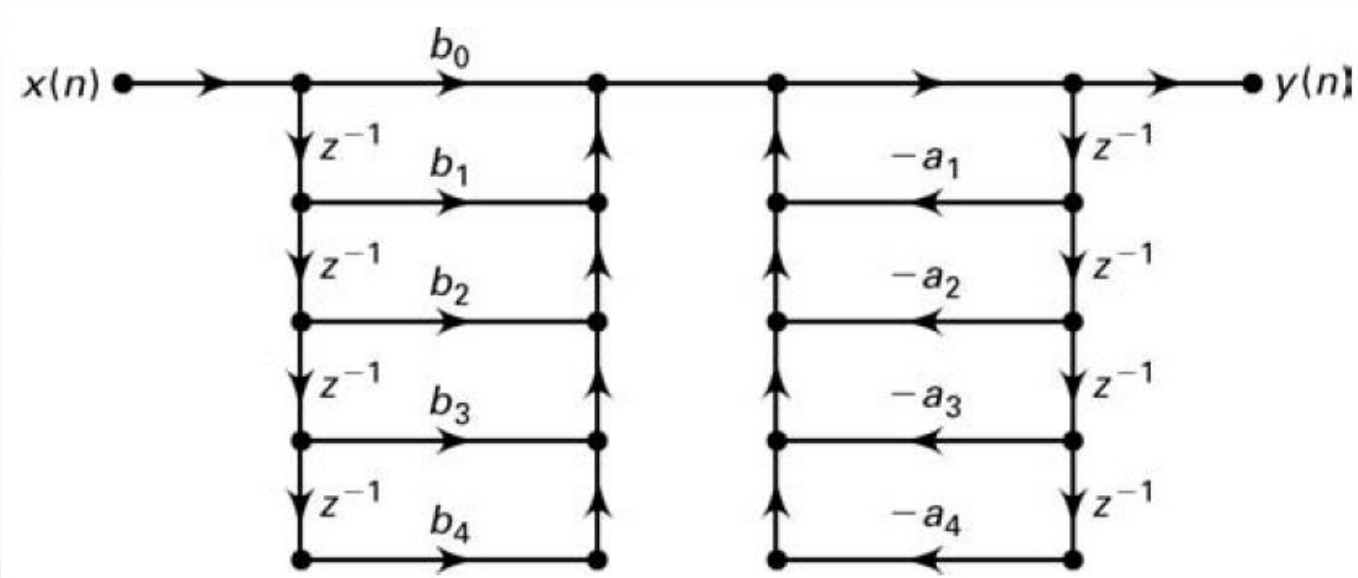
- The difference equation is implemented as given using delays, multipliers, and adders.
- For the purpose of illustration, let $M = N = 4$. Then the difference equation is

$$y(n) = b_o x(n) + b_1 x(n-1) + b_2 x(n-2) + b_4 x(n-4)$$
$$- a_1 y(n-1) - a_2 y(n-2) - a_3 y(n-3) - a_4 y(n-4)$$

Direct form

- The difference equation can be implemented as

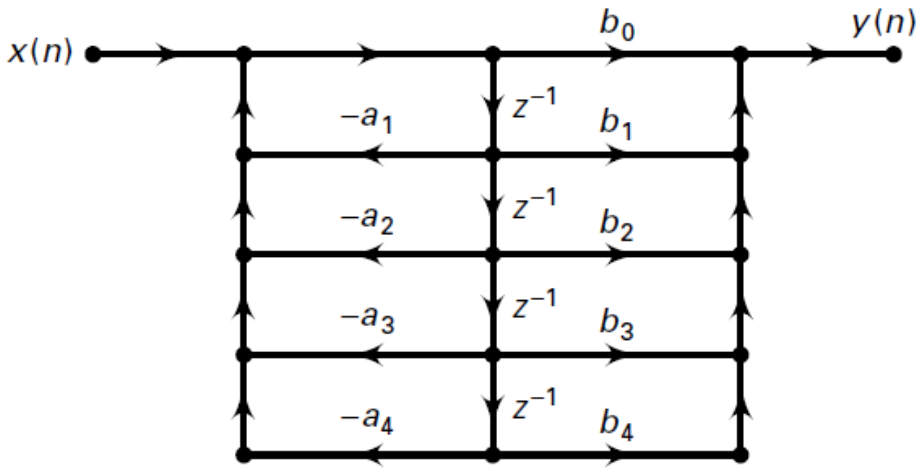
$$y(n) = b_o x(n) + b_1 x(n-1) + b_2 x(n-2) + b_3 x(n-3) + b_4 x(n-4)$$
$$- a_1 y(n-1) - a_2 y(n-2) - a_3 y(n-3) - a_4 y(n-4)$$



- This block diagram is called *direct form I* structure

Direct form II structure

- The two delay lines are close to each other, connected by a unity gain branch

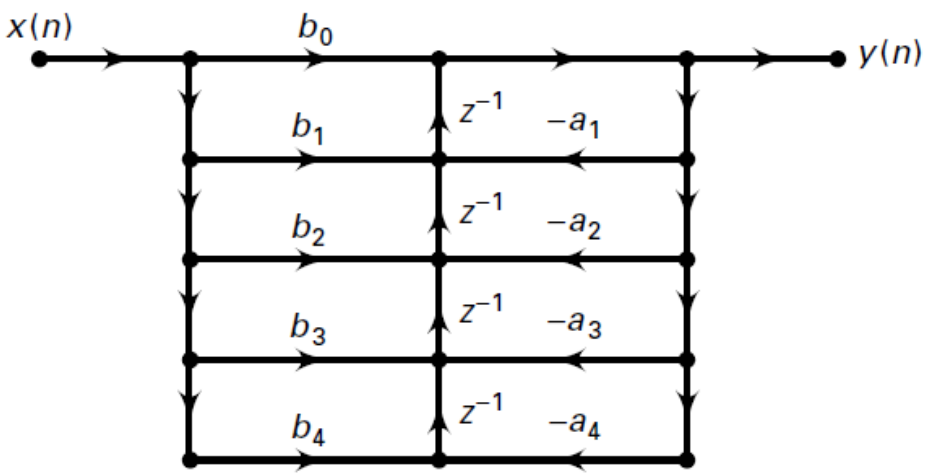


Transposed Structure

- An equivalent structure to the direct form can be obtained using a procedure called *transposition*.
- Three steps are performed:
  1. All path arrow directions are reversed.
  2. All branch nodes are replaced by adder nodes, and all adder nodes are replaced by branch nodes.
  3. The input and output nodes are interchanged.

Transposed Structure

- The transposed direct form II structure



MATLAB Implementation

- The direct form structure is described by two row vectors;
  - b containing the $\{b_n\}$ coefficients
  - a containing the $\{a_n\}$ coefficients
- The filter function implements the transposed direct form II structure.

Cascade Form

- In this form the system function $H(z)$ is written as a product of 2nd-order sections with real coefficients.
- This is done by factoring the numerator and denominator polynomials into their respective roots and then combining either a complex conjugate root pair or any two real roots into 2nd-order polynomials.

$$H(z) = \frac{b_o + b_1 z^{-1} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}} = b_o \frac{1 + \frac{b_1}{b_o} z^{-1} + \ldots + \frac{b_N}{b_o} z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

$$= b_o \prod_{k=1}^{K} \frac{1 + B_{k,1} z^{-1} + B_{k,2} z^{-2}}{1 + A_{k,1} z^{-1} + A_{k,2} z^{-2}}; \quad k = 1, \ldots, K$$

- K is equal to $N/2$, and $B_{k,1}$, $B_{k,2}$, $A_{k,1}$, and $A_{k,2}$ are real numbers representing the coefficients of 2nd-order sections

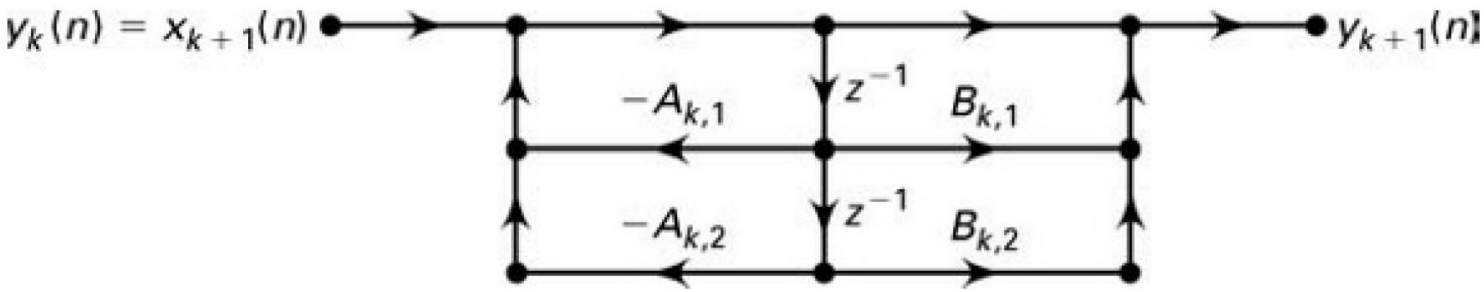
Biquad section

- The 2nd-order section

$$H(z)=\frac{Y_{k+1}(z)}{Y_k(z)}=\frac{1+B_{k,1}z^{-1}+B_{k,2}z^{-2}}{1+A_{k,1}z^{-1}+A_{k,2}z^{-2}};\quad k=1,\ldots,K$$

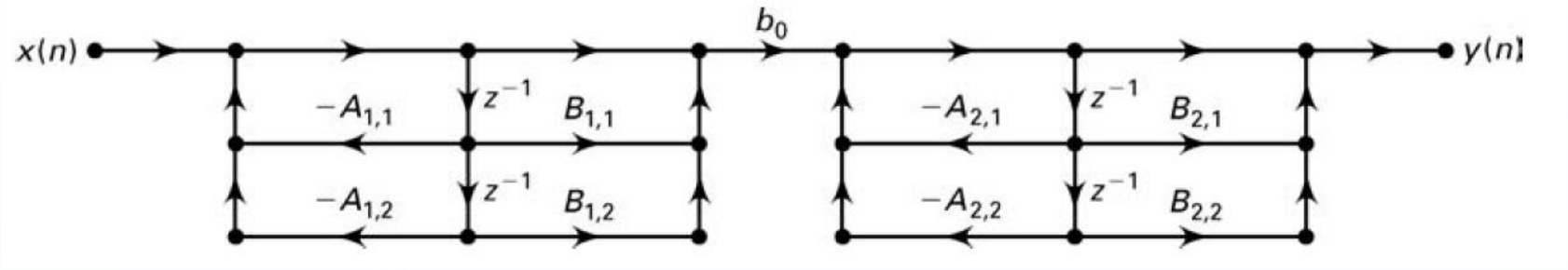
- with

$$Y_1(z)=b_o X(z);\quad Y_{K+1}(z)=Y(z)$$



Biquad section structure

- a cascade form structure for this 4th-order IIR filter (N = 4)



MATLAB Implementation

```
function [b0,B,A] = dir2cas(b,a);
% DIRECT-form to CASCADE-form conversion (cplxpair version)
% ------------------------------------------------------
% [b0,B,A] = dir2cas(b,a)
% b0 = gain coefficient
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
% b = numerator polynomial coefficients of DIRECT form
% a = denominator polynomial coefficients of DIRECT form
% compute gain coefficient b0
b0 = b(1); b = b/b0; a0 = a(1); a = a/a0; b0 = b0/a0;
%
M = length(b); N = length(a);
if N > M
b = [b zeros(1,N-M)];
```

MATLAB Implementation

```
elseif M > N
a = [a zeros(1,M-N)]; N = M;
else
NM = 0;
end
%
K = floor(N/2); B = zeros(K,3); A = zeros(K,3);
if K*2 == N;
b = [b 0]; a = [a 0];
end
%
broots = cplxpair(roots(b)); aroots = cplxpair(roots(a));
for i=1:2:2*K
Brow = broots(i:1:i+1,:); Brow = real(poly(Brow));
B(fix((i+1)/2),:) = Brow;
Arow = aroots(i:1:i+1,:); Arow = real(poly(Arow));
A(fix((i+1)/2),:) = Arow;
end
```

MATLAB Implementation

```
function y = casfiltr(b0,B,A,x);
% CASCADE form realization of IIR and FIR filters
% ----------------------------------------------
% y = casfiltr(b0,B,A,x);
% y = output sequence
% b0 = gain coefficient of CASCADE form
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
% x = input sequence
%
[K,L] = size(B);
N = length(x); w = zeros(K+1,N); w(1,:) = x;
```

MATLAB Implementation

```
function y = casfiltr(b0,B,A,x);
% CASCADE form realization of IIR and FIR filters
% ----------------------------------------------
% y = casfiltr(b0,B,A,x);
% y = output sequence
% b0 = gain coefficient of CASCADE form
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
% x = input sequence
%
[K,L] = size(B);
N = length(x); w = zeros(K+1,N); w(1,:) = x;
for i = 1:1:K
w(i+1,:) = filter(B(i,:),A(i,:),w(i,:));
end
y = b0*w(K+1,:);
```

MATLAB Implementation

```
function [b,a] = cas2dir(b0,B,A);
% CASCADE-to-DIRECT form conversion
% --------------------------------
% [b,a] = cas2dir(b0,B,A)
% b = numerator polynomial coefficients of DIRECT form
% a = denominator polynomial coefficients of DIRECT form
% b0 = gain coefficient
% B = K by 3 matrix of real coefficients containing bk's
% A = K by 3 matrix of real coefficients containing ak's
%
[K,L] = size(B);
b = [1]; a = [1];
for i=1:1:K
b=conv(b,B(i,:)); a=conv(a,A(i,:));
end
b = b*b0;
```

Example

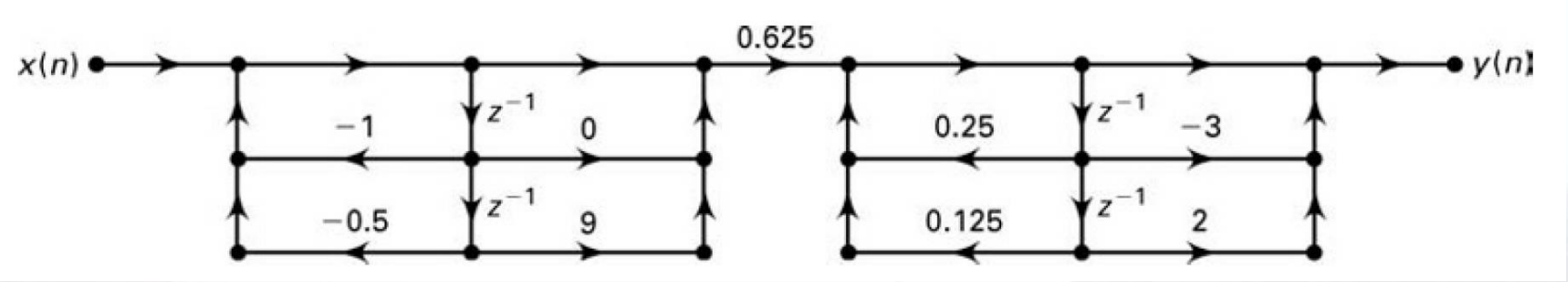
- A filter is described by the following difference equation

$$16y(n) + 12y(n-1) + 2y(n-2) - 4y(n-3) - y(n-4)$$
$$= x(n) - 3x(n-1) + 11x(n-2) - 27x(n-3) + 18x(n-4)$$

- Determine its cascaded form structure.

Example

```
>> b=[1 -3 11 -27 18]; a=[16 12 2 -4 -1];
>> [b0,B,A]=dir2cas(b,a)
b0 = 0.0625
B =     1.0000 -0.0000 9.0000
        1.0000 -3.0000 2.0000
A =     1.0000 1.0000 0.5000
        1.0000 -0.2500 -0.1250
```



Example

```
>> delta = impseq(0,0,7)
delta = 1 0 0 0 0 0 0 0
>> format long
>> hcas=casfiltr(b0,B,A,delta)
>> hdir=filter(b,a,delta)
```

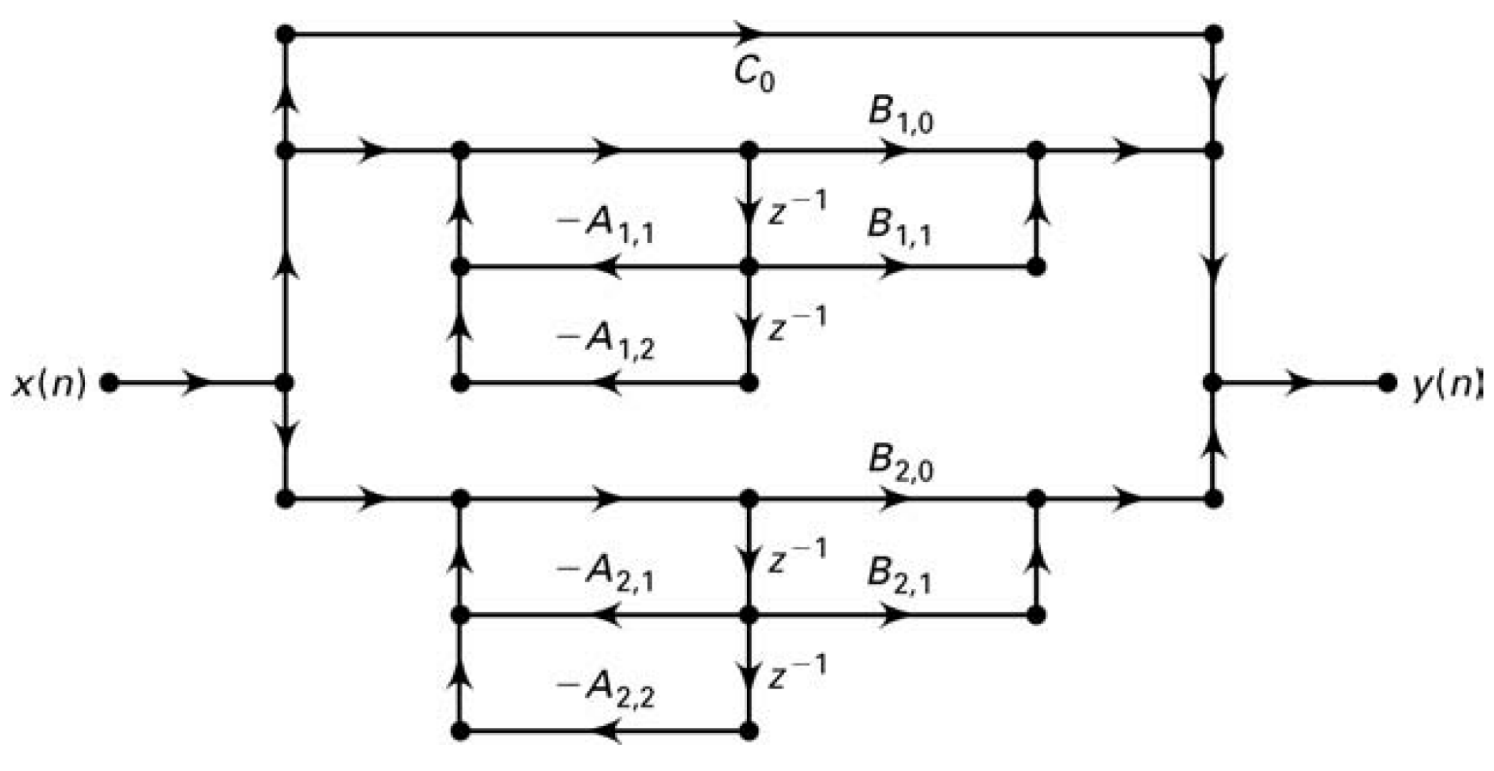
Parallel Form

- The system function $H(z)$ is written as a sum of 2nd-order sections using partial fraction expansion

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_o + b_1 z^{-1} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

Parallel Form

- A parallel-form structure for this 4th-order IIR filter.



Matlab Implementation

- A function dir2par converts the direct-form coefficients *{bn}* and *{an}* into parallel form coefficients *{Bk,i}* and *{Ak,i}*.

- The dir2cas function first computes the *z*-domain partial fraction expansion using the residuez function.

- The cplxpair function from MATLAB can be used to arrange pole-and-residue pairs into complex conjugate pole-and-residue pairs followed by real poleand- residue pairs
  - this sorts a complex array into complex conjugate pairs

Example

- A filter is described by the following difference equation

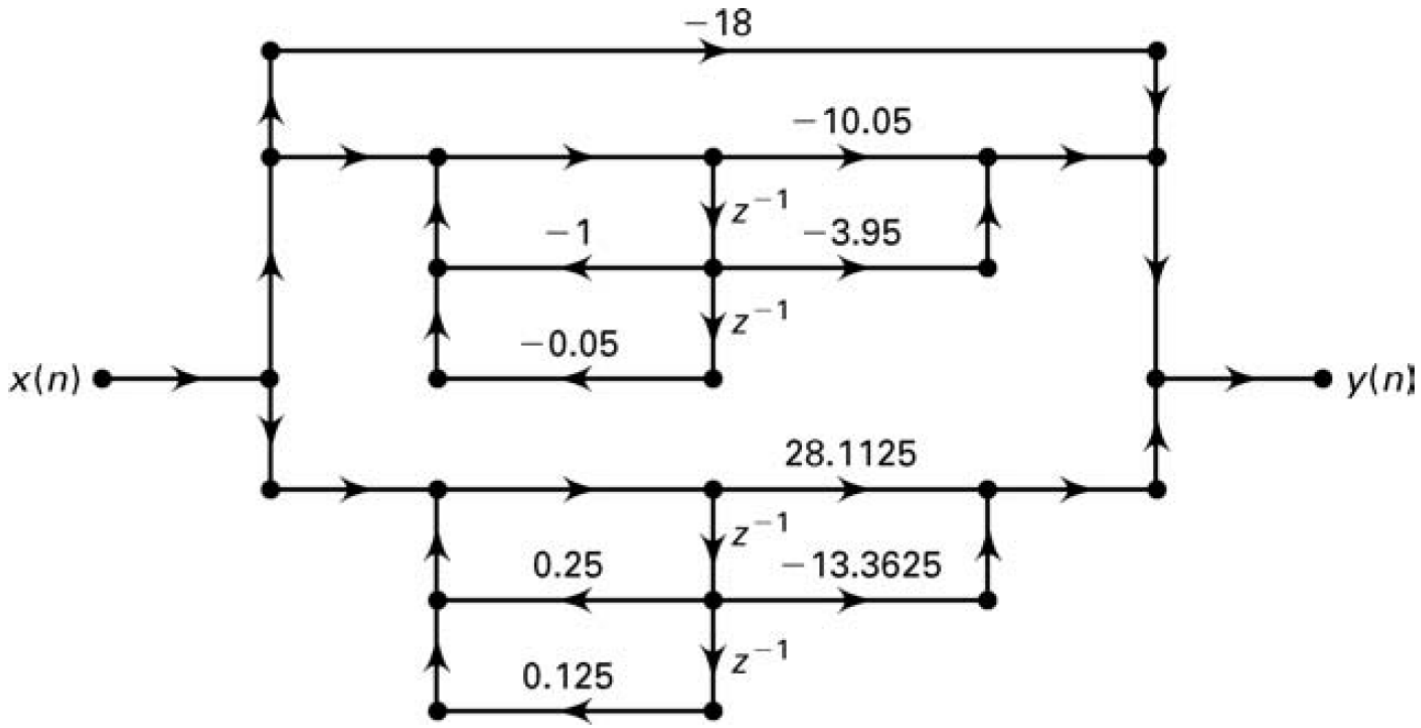
$$16y(n) + 12y(n-1) + 2y(n-2) - 4y(n-3) - y(n-4)$$
$$= x(n) - 3x(n-1) + 11x(n-2) - 27x(n-3) + 18x(n-4)$$

- Now determine its parallel form.

Example

>> b=[1 -3 11 -27 18]; a=[16 12 2 -4 -1];

>> [C,B,A]=dir2par(b,a)



Example

- To check our parallel structure, let us compute the first 8 samples of the impulse response using both forms

```
>> format long; delta = impseq(0,0,7); hpar=parfiltr(C,B,A,delta)

>> hdir = filter(b,a,delta)
```

Terima Kasih