Chapter 4
DISCRETE-TIME SYSTEMS
4.3 Characterization of Discrete-Time Systems
4.4 Discrete-Time System Networks

Victoria, BC, Canada
Email: aantoniou@ieee.org

July 14, 2018

Introduction

- Discrete-time systems can be characterized by means of mathematical equations or in terms of networks.
- The mathematical characterization can be deduced by *analyzing* a network representation of the discrete-time system.
- On the other hand, a network representation can be deduced from the mathematical characterization by a process called *realization*.
- This presentation deals with the characterization, the network representation, and the analysis of discrete-time systems.

Types of Discrete-Time Systems

Two types of discrete-time systems can be identified:

- nonrecursive
- recursive

Characterization of Nonrecursive Systems

- Discrete-time systems are characterized in terms of difference equations.

Characterization of Nonrecursive Systems

- Discrete-time systems are characterized in terms of difference equations.
- In a *nonrecursive* discrete-time system, the response at instant nT can be a function of a number of values of the excitation before nT, the value at nT, and a number of values of the excitation after nT.

Characterization of Nonrecursive Systems

- Discrete-time systems are characterized in terms of difference equations.
- In a *nonrecursive* discrete-time system, the response at instant nT can be a function of a number of values of the excitation before nT, the value at nT, and a number of values of the excitation after nT.
- If instant nT is taken to be the present and the present response depends on the past N values, the present value, and the future K values of the excitation, then

$$y(nT) = f[x(nT - NT), \ \ldots, \ x(nT - T), \ x(nT), \\ x(nT + T), \ \ldots, x(nT + KT)]$$

Nonrecursive Systems

- If we assume that the system is *linear*, then

$$
\begin{aligned}
y(nT) &= a_N x(nT - NT) + \cdots + a_1 x(nT - T) + a_0 x(nT) \\
&\quad + a_{-1} x(nT + T) + \cdots + a_{-K} x(nT + KT) \\
&= a_{-K} x(nT + KT) + \cdots + a_{-1} x(nT + T) \\
&\quad + a_0 x(nT) + a_1 x(nT - T) + \cdots + a_N x(nT - NT) \\
&= \sum_{i=-K}^{N} a_i x(nT - iT)
\end{aligned}
$$

Nonrecursive Systems

- If we assume that the system is *linear*, then

$$
\begin{aligned}
y(nT) &= a_N x(nT - NT) + \cdots + a_1 x(nT - T) + a_0 x(nT) \\
&\quad + a_{-1} x(nT + T) + \cdots + a_{-K} x(nT + KT) \\
&= a_{-K} x(nT + KT) + \cdots + a_{-1} x(nT + T) \\
&\quad + a_0 x(nT) + a_1 x(nT - T) + \cdots + a_N x(nT - NT) \\
&= \sum_{i=-K}^{N} a_i x(nT - iT)
\end{aligned}
$$

- If the system is *time-invariant*, then the parameters a_i are constants and independent of time.

Nonrecursive Systems

- If we assume that the system is *linear*, then

$$
\begin{aligned}
y(nT) &= a_N x(nT - NT) + \cdots + a_1 x(nT - T) + a_0 x(nT) \\
&\quad + a_{-1} x(nT + T) + \cdots + a_{-K} x(nT + KT) \\
&= a_{-K} x(nT + KT) + \cdots + a_{-1} x(nT + T) \\
&\quad + a_0 x(nT) + a_1 x(nT - T) + \cdots + a_N x(nT - NT) \\
&= \sum_{i=-K}^{N} a_i x(nT - iT)
\end{aligned}
$$

- If the system is *time-invariant*, then the parameters a_i are constants and independent of time.
- The above equation is a *difference* equation of order $N + K$ and it represents a *noncausal* nonrecursive system of the same order.

Nonrecursive Systems *Cont'd*

...

$$
\begin{aligned}
y(nT) &= a_{-K}x(nT+KT)+\cdots+a_{-1}x(nT+T)+a_0x(nT)\\
&\quad +a_1x(nT-T)+\cdots+a_Nx(nT-NT)\\
&= \sum_{i=-K}^{N} a_i x(nT-iT)
\end{aligned}
$$

If the system is causal, the response at instant nT is independent of $x(nT+T)$, $x(nT+2T)$, ..., $x(nT+KT)$ since these are future values of the input with respect to instant nT.

For a *causal* nonrecursive discrete-time system, we have

$$a_i = 0 \quad \text{for} \quad i \leq -1$$

Nonrecursive Systems *Cont'd*

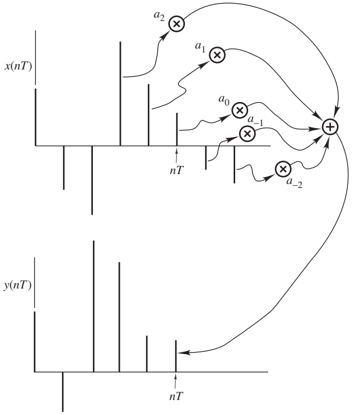
Therefore, the difference equation becomes

$$\begin{aligned} y(nT) &= a_0 x(nT) + a_1 x(nT - T) + \cdots + a_N x(nT - NT) \\ &= \sum_{i=0}^{N} a_i x(nT - iT) \end{aligned}$$

We now have an Nth-order difference equation that represents a *linear*, *time-invariant*, and *causal* digital system of the same order.

Example

$$y(nT) = a_2x(nT-2T) + a_1x(nT-T) + a_0x(nT) + a_{-1}x(nT+T) + a_{-2}x(nT+2T)$$



Fourth-order noncausal nonrecursive system

Characterization of Recursive Systems

- In *recursive* discrete-time systems, the response is a function of elements in the excitation as well as elements in the response sequence.

Characterization of Recursive Systems

- In *recursive* discrete-time systems, the response is a function of elements in the excitation as well as elements in the response sequence.

- An $(M+N)$th-order, *linear*, *time-invariant*, *noncausal* recursive discrete-time system can be represented by the difference equation

$$y(nT) = \sum_{i=-M}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

where the coefficients a_i and b_i are constants independent of time.

Characterization of Recursive Systems

- In *recursive* discrete-time systems, the response is a function of elements in the excitation as well as elements in the response sequence.

- An $(M+N)$th-order, *linear*, *time-invariant*, *noncausal* recursive discrete-time system can be represented by the difference equation

$$y(nT) = \sum_{i=-M}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

  where the coefficients a_i and b_i are constants independent of time.

- If nT is taken to be the present, the present response is a function of the present value, the past N values, and the future M values of the excitation and the past N values of the response.

...

$$y(nT) = \sum_{i=-M}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

- Some of the coefficients a_i and b_i can be zero.

...

$$y(nT) = \sum_{i=-M}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

- Some of the coefficients a_i and b_i can be zero.
- If *all* coefficients b_i are zero, then the above equation reduces to the equation of a nonrecursive system.

  In effect, *a nonrecursive can be regarded as a special case of a recursive system*.

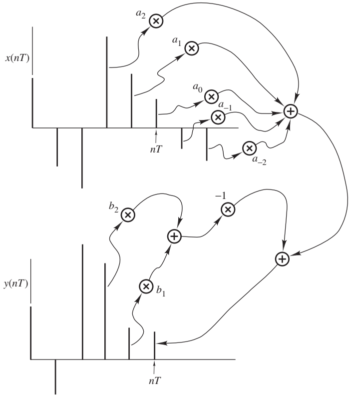
Example

A fourth-order linear, time-invariant, noncausal, recursive discrete-time system can be represented by the difference equation

$$\begin{aligned} y(nT) = {} & a_{-2}x(nT+2T) + a_{-1}x(nT+T) + a_0x(nT) + a_1x(nT-T) \\ & a_2x(nT-2T) - b_1y(nT-T) - b_2y(nT-2T) \end{aligned}$$

...

$$y(nT) = a_{-2}x(nT+2T) + a_{-1}x(nT+T) + a_0x(nT) + a_1x(nT-T)$$
$$a_2x(nT-2T) - b_1y(nT-T) - b_2y(nT-2T)$$



Fourth-order noncausal recursive system

Basic Elements

The basic elements of discrete-time systems are

- The unit delay
- The adder
- The multiplier

- The *unit delay* is a memory device that can store a number.

  When a clock pulse is received, it outputs the number stored and records the number appearing at the input.

- The *unit delay* is a memory device that can store a number.

  When a clock pulse is received, it outputs the number stored and records the number appearing at the input.

- The *adder* may have several inputs and its operation is to produce an output that is equal to the sum of the inputs when a clock pulse is received.

  In theory, the adder is assumed to operate instantaneously although in practice a small delay will occur.

- The *unit delay* is a memory device that can store a number.

  When a clock pulse is received, it outputs the number stored and records the number appearing at the input.

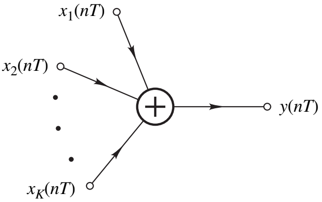
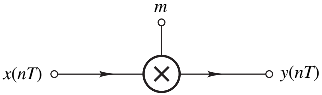
- The *adder* may have several inputs and its operation is to produce an output that is equal to the sum of the inputs when a clock pulse is received.

  In theory, the adder is assumed to operate instantaneously although in practice a small delay will occur.

- The *multiplier* has one input and one output and its operation is to multiply the input by a constant when a clock pulse is received.

  Like the adder, it is assumed to operate instantaneously although a small delay will occur in practice.

Table 4.1 Elements of discrete-time systems

Element	Symbol	Equation
Unit delay		$y(nT) = x(nT-T)$
Adder		$y(nT) = \sum_{i=1}^{K} x_i(nT)$
Multiplier		$y(nT) = mx(nT)$

Implementation of Basic Elements

The implementation of the basic elements of discrete-time systems can assume many forms depending on the

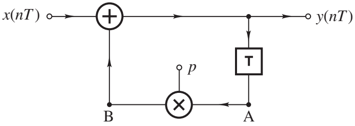
- type of arithmetic (e.g., fixed-point, floating-point)
- type of number representation (e.g., signed-magnitude, two's complement)

Implementation of Basic Elements

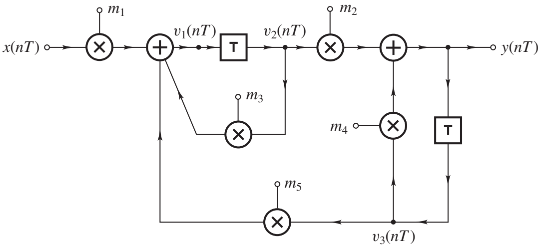
The implementation of the basic elements of discrete-time systems can assume many forms depending on the

- type of arithmetic (e.g., fixed-point, floating-point)
- type of number representation (e.g., signed-magnitude, two's complement)
- type of number quantization (e.g., truncation, rounding)
- mode of operation (serial or parallel), etc.

Like analog filters, discrete-time systems can be represented by networks which are collections of interconnected unit delays, adders, and multipliers:

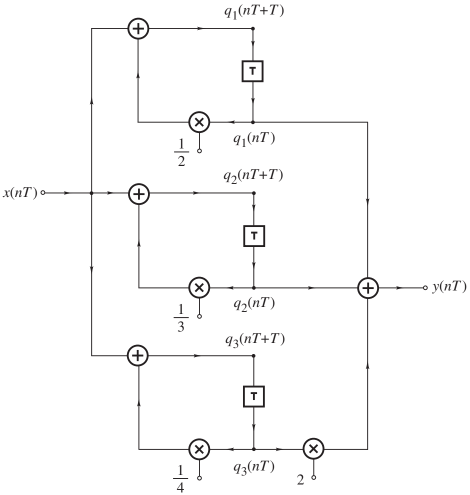


(*a*)



(*c*)

Representation by Networks



Network Analysis

- Network analysis is the process of obtaining some mathematical characterization of a given network.

  For example, we analyze a resonant circuit by finding its differential equation (or the Laplace transform of the differential equation).

  Similarly, we analyze a discrete-time system by obtaining its difference equation (or the z transform of the difference equation).

Network Analysis

- Network analysis is the process of obtaining some mathematical characterization of a given network.

  For example, we analyze a resonant circuit by finding its differential equation (or the Laplace transform of the differential equation).

  Similarly, we analyze a discrete-time system by obtaining its difference equation (or the z transform of the difference equation).

- Analysis can be simplified by using the *shift operator* $\mathcal{E}$ of numerical analysis, which is defined as

$$\mathcal{E}^r f(nT) = f(nT + rT)$$

  A negative r delays the signal by a period $|r|T$ whereas a positive r advances the signal into the future by a period rT!

Network Analysis

- Network analysis is the process of obtaining some mathematical characterization of a given network.

  For example, we analyze a resonant circuit by finding its differential equation (or the Laplace transform of the differential equation).

  Similarly, we analyze a discrete-time system by obtaining its difference equation (or the z transform of the difference equation).

- Analysis can be simplified by using the *shift operator* $\mathcal{E}$ of numerical analysis, which is defined as

$$\mathcal{E}^r f(nT) = f(nT + rT)$$

  A negative r delays the signal by a period $|r|T$ whereas a positive r advances the signal into the future by a period rT!

- The shift operator is a linear operator that obeys the usual laws of algebra (law of exponents, distributive law, etc.).

Properties of the Shift Operator

1. Since

$$\begin{aligned}\mathcal{E}^r[a_1 f_1(nT) + a_2 f_2(nT)] &= a_1 f_1(nT + rT) + a_2 f_2(nT + rT)\\ &= a_1 \mathcal{E}^r f_1(nT) + a_2 \mathcal{E}^r f_2(nT)\end{aligned}$$

   we conclude that *$\mathcal{E}$ is a linear operator which distributes* with respect to a sum of functions of nT.

Properties of the Shift Operator

1. Since

$$\begin{aligned}\mathcal{E}^r[a_1 f_1(nT) + a_2 f_2(nT)] &= a_1 f_1(nT + rT) + a_2 f_2(nT + rT)\\ &= a_1 \mathcal{E}^r f_1(nT) + a_2 \mathcal{E}^r f_2(nT)\end{aligned}$$

   we conclude that *$\mathcal{E}$ is a linear operator which distributes* with respect to a sum of functions of nT.

2. Since

$$\begin{aligned}\mathcal{E}^r \mathcal{E}^p f(nT) &= \mathcal{E}^r f(nT + pT) = f(nT + rT + pT)\\ &= f[nT + (r+p)T]\\ &= \mathcal{E}^{r+p} f(nT)\end{aligned}$$

   the *shift operator obeys the law of exponents*.

3. If

$$g(nT) = \mathcal{E}^r f(nT)$$

then

$$\mathcal{E}^{-r} g(nT) = \mathcal{E}^{-r}\mathcal{E}^r f(nT) = \mathcal{E}^{-r} f(nT + rT) = f(nT)$$

for all $f(nT)$, and if

$$f(nT) = \mathcal{E}^{-r} g(nT)$$

then

$$\mathcal{E}^r f(nT) = \mathcal{E}^r\mathcal{E}^{-r} g(nT) = \mathcal{E}^r g(nT - rT) = g(nT)$$

for all $g(nT)$.

3. If

$$g(nT) = \mathcal{E}^r f(nT)$$

then

$$\mathcal{E}^{-r} g(nT) = \mathcal{E}^{-r}\mathcal{E}^r f(nT) = \mathcal{E}^{-r} f(nT + rT) = f(nT)$$

for all $f(nT)$, and if

$$f(nT) = \mathcal{E}^{-r} g(nT)$$

then

$$\mathcal{E}^r f(nT) = \mathcal{E}^r\mathcal{E}^{-r} g(nT) = \mathcal{E}^r g(nT - rT) = g(nT)$$

for all $g(nT)$.

Therefore, *$\mathcal{E}^{-r}$ is the inverse of $\mathcal{E}^r$ and vice versa,* i.e.,

$$\mathcal{E}^{-r}\mathcal{E}^r = \mathcal{E}^r\mathcal{E}^{-r} = 1$$

4. A linear combination of powers of $\mathcal{E}$ defines a meaningful operator, e.g., if

$$f(\mathcal{E}) = 1 + a_1\mathcal{E} + a_2\mathcal{E}^2$$

then

$$\begin{aligned} f(\mathcal{E})f(nT) &= (1 + a_1\mathcal{E} + a_2\mathcal{E}^2)f(nT) \\ &= f(nT) + a_1\mathcal{E}f(nT) + a_2\mathcal{E}^2 f(nT) \\ &= f(nT) + a_1 f(nT + T) + a_2 f(nT + 2T) \end{aligned}$$

4. A linear combination of powers of $\mathcal{E}$ defines a meaningful operator, e.g., if

$$f(\mathcal{E}) = 1 + a_1\mathcal{E} + a_2\mathcal{E}^2$$

then

$$\begin{aligned} f(\mathcal{E})f(nT) &= (1 + a_1\mathcal{E} + a_2\mathcal{E}^2)f(nT) \\ &= f(nT) + a_1\mathcal{E}f(nT) + a_2\mathcal{E}^2 f(nT) \\ &= f(nT) + a_1 f(nT + T) + a_2 f(nT + 2T) \end{aligned}$$

Furthermore, given an operator $f(\mathcal{E})$ of the above type, an *inverse operator* $f(\mathcal{E})^{-1}$ can be defined such that

$$f(\mathcal{E})^{-1}f(\mathcal{E}) = f(\mathcal{E})f(\mathcal{E})^{-1} = 1$$

Properties of the Shift Operator *Cont'd*

5. If $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, and $f_3(\mathcal{E})$ are operators that comprise linear combinations of powers of $\mathcal{E}$, they satisfy the distributive, commutative, and associative laws of algebra, i.e.,

$$f_1(\mathcal{E})[f_2(\mathcal{E}) + f_3(\mathcal{E})] = f_1(\mathcal{E})f_2(\mathcal{E}) + f_1(\mathcal{E})f_3(\mathcal{E})$$

$$f_1(\mathcal{E})f_2(\mathcal{E}) = f_2(\mathcal{E})f_1(\mathcal{E})$$

$$f_1(\mathcal{E})[f_2(\mathcal{E})f_3(\mathcal{E})] = [f_1(\mathcal{E})f_2(\mathcal{E})]f_3(\mathcal{E})$$

5. If $f_1(\mathcal{E})$, $f_2(\mathcal{E})$, and $f_3(\mathcal{E})$ are operators that comprise linear combinations of powers of $\mathcal{E}$, they satisfy the distributive, commutative, and associative laws of algebra, i.e.,

$$f_1(\mathcal{E})[f_2(\mathcal{E}) + f_3(\mathcal{E})] = f_1(\mathcal{E})f_2(\mathcal{E}) + f_1(\mathcal{E})f_3(\mathcal{E})$$

$$f_1(\mathcal{E})f_2(\mathcal{E}) = f_2(\mathcal{E})f_1(\mathcal{E})$$

$$f_1(\mathcal{E})[f_2(\mathcal{E})f_3(\mathcal{E})] = [f_1(\mathcal{E})f_2(\mathcal{E})]f_3(\mathcal{E})$$

Operators such as these can be used to construct *more complicated operators* of the form

$$F(\mathcal{E}) = f_1(\mathcal{E})f_2(\mathcal{E})^{-1} = f_2(\mathcal{E})^{-1}f_1(\mathcal{E})$$

which may also be expressed as $F(\mathcal{E}) = \dfrac{f_1(\mathcal{E})}{f_2(\mathcal{E})}$ without danger of ambiguity.

Properties of the Shift Operator *Cont'd*

- From the properties just discussed, it follows that the *shift operator can be treated like an algebraic quantity* and *linear combinations of $\mathcal{E}$ can be treated like polynomials*.

Properties of the Shift Operator *Cont'd*

- From the properties just discussed, it follows that the *shift operator can be treated like an algebraic quantity* and *linear combinations of $\mathcal{E}$ can be treated like polynomials*.
- *Note:* An operator needs something to operate upon! Therefore, all the following are meaningless:

$$y = (2 + 4x + 9x^2)\frac{d}{dx}$$

$$Y(s) = [2 + 4u(t)]\mathcal{L}$$

$$y(nT) = x(nT)(\mathcal{E} + \mathcal{E}^2)$$

$$y(nT) = x(nT)\mathcal{R}$$

Example

An Nth-order recursive discrete-time system can be represented by the difference equation

$$y(nT) = \sum_{i=0}^{N} a_i x(nT - iT) - \sum_{i=1}^{N} b_i y(nT - iT)$$

Obtain an expression for the response in terms of the shift operator.

Solution From the definition of the shift operator, we can write

$$
\begin{aligned}
y(nT) &= \sum_{i=0}^{N} a_i \mathcal{E}^{-i} x(nT) - \sum_{i=1}^{N} b_i \mathcal{E}^{-i} y(nT) \\
y(nT) + \sum_{i=1}^{N} b_i \mathcal{E}^{-i} y(nT) &= \sum_{i=0}^{N} a_i \mathcal{E}^{-i} x(nT) \\
\left(1 + \sum_{i=1}^{N} b_i \mathcal{E}^{-i}\right) y(nT) &= \sum_{i=0}^{N} a_i \mathcal{E}^{-i} x(nT)
\end{aligned}
$$

Example *Cont'd*

Solution From the definition of the shift operator, we can write

$$
\begin{aligned}
y(nT) &= \sum_{i=0}^{N} a_i \mathcal{E}^{-i} x(nT) - \sum_{i=1}^{N} b_i \mathcal{E}^{-i} y(nT) \\
y(nT) + \sum_{i=1}^{N} b_i \mathcal{E}^{-i} y(nT) &= \sum_{i=0}^{N} a_i \mathcal{E}^{-i} x(nT) \\
\left(1 + \sum_{i=1}^{N} b_i \mathcal{E}^{-i}\right) y(nT) &= \sum_{i=0}^{N} a_i \mathcal{E}^{-i} x(nT)
\end{aligned}
$$

or $\quad y(nT) = \left(\frac{\sum_{i=0}^{N} a_i \mathcal{E}^{-i}}{1 + \sum_{i=1}^{N} b_i \mathcal{E}^{-i}} \right) x(nT) = F(\mathcal{E}) x(nT)$

where $\quad F(\mathcal{E}) = \left(\frac{\sum_{i=0}^{N} a_i \mathcal{E}^{-i}}{1 + \sum_{i=1}^{N} b_i \mathcal{E}^{-i}} \right)$

- By inspection

Methods of Analysis

- By inspection
- By writing the network equations and then solving them

Methods of Analysis

- By inspection
- By writing the network equations and then solving them
- By applying signal flow graphs
  - using node elimination techniques
  - using Mason's gain formula

Methods of Analysis *Cont'd*

Note: The analysis of discrete-time systems is much simpler than that of continuous-time systems:

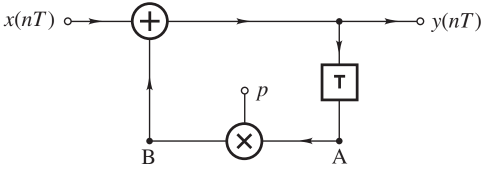
- In discrete-time systems the signals can assume only one form, namely, they are *sequences of numbers*.

Note: The analysis of discrete-time systems is much simpler than that of continuous-time systems:

- In discrete-time systems the signals can assume only one form, namely, they are *sequences of numbers*.
- In continuous-time systems signals can assume two forms, namely, *voltages and currents*, which are strictly interrelated through Kirchhoff's laws.

Example

Analyze the network:



(*a*)

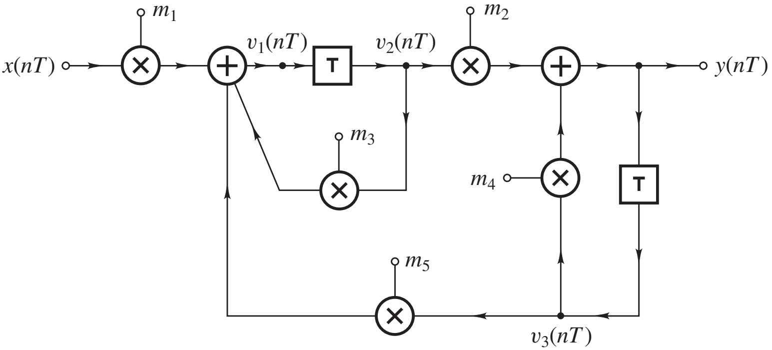
Solution Signal at node A: $y(nT - T)$

Signal at node B: $py(nT - T)$

Output signal: $y(nT) = x(nT) + py(nT - T)$

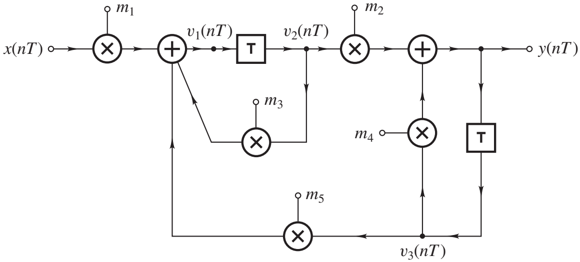
Example

Analyze the network:



(c)

Solution



(c)

Define signals $v_1(nT)$ and $v_2(nT)$ as shown.

From the figure: $$v_1(nT) = m_1 x(nT) + m_3 v_2(nT) + m_5 v_3(nT) \qquad \text{(A)}$$

and $$y(nT) = m_2 v_2(nT) + m_4 v_3(nT) \qquad \text{(B)}$$

where $v_2(nT) = \mathcal{E}^{-1} v_1(nT) \qquad v_3(nT) = \mathcal{E}^{-1} y(nT)$

...

From the figure:

$$\begin{aligned} v_1(nT) &= m_1x(nT) + m_3v_2(nT) + m_5v_3(nT) \qquad \text{(A)} \\ y(nT) &= m_2v_2(nT) + m_4v_3(nT) \qquad \text{(B)} \end{aligned}$$

where $\quad v_2(nT) = \mathcal{E}^{-1}v_1(nT) \qquad v_3(nT) = \mathcal{E}^{-1}y(nT)$

If we eliminate $v_2(nT)$ and $v_3(nT)$ in Eqs. (A) and (B), we have

$$(1 - m_3\mathcal{E}^{-1})v_1(nT) = m_1x(nT) + m_5\mathcal{E}^{-1}y(nT) \qquad \text{(C)}$$

and

$$(1 - m_4\mathcal{E}^{-1})y(nT) = m_2\mathcal{E}^{-1}v_1(nT) \qquad \text{(D)}$$

...

$$(1 - m_3\mathcal{E}^{-1})v_1(nT) = m_1x(nT) + m_5\mathcal{E}^{-1}y(nT) \qquad \text{(C)}$$

and

$$(1 - m_4\mathcal{E}^{-1})y(nT) = m_2\mathcal{E}^{-1}v_1(nT) \qquad \text{(D)}$$

Eq. (D) can be expressed as

$$(1 - m_3\mathcal{E}^{-1})(1 - m_4\mathcal{E}^{-1})y(nT) = m_2\mathcal{E}^{-1}(1 - m_3\mathcal{E}^{-1})v_1(nT)$$

and on eliminating $(1 - m_3\mathcal{E}^{-1})v_1(nT)$ using Eq. (C), we obtain

$$\begin{aligned}[1 - (m_3 + m_4)\mathcal{E}^{-1} + m_3m_4\mathcal{E}^{-2}]y(nT) = {} & m_1m_2\mathcal{E}^{-1}x(nT) \\ & + m_2m_5\mathcal{E}^{-2}y(nT)\end{aligned}$$

or $[1 - (m_3 + m_4)\mathcal{E}^{-1} + (m_3m_4 - m_2m_5)\mathcal{E}^{-2}]y(nT) = m_1m_2\mathcal{E}^{-1}x(nT)$

. . .

or

$$[1 - (m_3 + m_4)\mathcal{E}^{-1} + (m_3 m_4 - m_2 m_5)\mathcal{E}^{-2}]y(nT) = m_1 m_2 \mathcal{E}^{-1} x(nT)$$

Therefore, on eliminating the shift operator, we get

$$y(nT) = a_1 x(nT - T) + b_1 y(nT - T) + b_2 y(nT - 2T)$$

where

$$a_1 = m_1 m_2, \quad b_1 = m_3 + m_4, \quad b_2 = m_2 m_5 - m_3 m_4$$

Signal Flow-Graph Analysis

- A discrete-time network can be represented by an equivalent *signal flow graph* which is made up of a collection of interconnected directed branches and nodes.

Signal Flow-Graph Analysis

- A discrete-time network can be represented by an equivalent *signal flow graph* which is made up of a collection of interconnected directed branches and nodes.
- Such a signal flow graph is, on the one hand, a compact representation of the system and, on the other, it can be used to analyze the system.

Given a discrete-time network, a corresponding signal flow graph can be readily deduced by replacing

- each adder by a node with one outgoing branch and as many incoming branches as there are inputs to the adder,

Given a discrete-time network, a corresponding signal flow graph can be readily deduced by replacing

- each adder by a node with one outgoing branch and as many incoming branches as there are inputs to the adder,
- each distribution node remains a distribution node,

Given a discrete-time network, a corresponding signal flow graph can be readily deduced by replacing

- each adder by a node with one outgoing branch and as many incoming branches as there are inputs to the adder,
- each distribution node remains a distribution node,
- each multiplier by a directed branch with transmittance equal to the constant of the multiplier,

Given a discrete-time network, a corresponding signal flow graph can be readily deduced by replacing

- each adder by a node with one outgoing branch and as many incoming branches as there are inputs to the adder,
- each distribution node remains a distribution node,
- each multiplier by a directed branch with transmittance equal to the constant of the multiplier,
- each direct transmission path by a directed branch with transmittance equal to unity, and

Signal Flow-Graph Analysis *Cont'd*

Given a discrete-time network, a corresponding signal flow graph can be readily deduced by replacing

- each adder by a node with one outgoing branch and as many incoming branches as there are inputs to the adder,
- each distribution node remains a distribution node,
- each multiplier by a directed branch with transmittance equal to the constant of the multiplier,
- each direct transmission path by a directed branch with transmittance equal to unity, and
- each unit delay by a directed branch with transmittance equal to the shift operator $\mathcal{E}^{-1}$.

Example

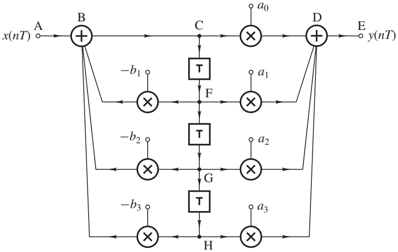


Fig. 6*a*

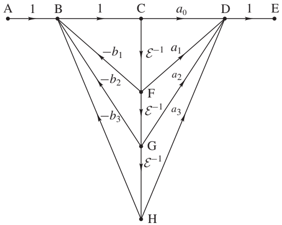


Fig. 6*b*

Node Elimination

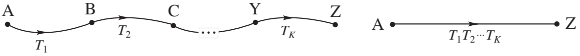
- A discrete-time network can be analyzed by reducing its signal flow graph into a single transmittance between input node $x(nT)$ and output node $y(nT)$ using node elimination techniques.

Node Elimination

- A discrete-time network can be analyzed by reducing its signal flow graph into a single transmittance between input node $x(nT)$ and output node $y(nT)$ using node elimination techniques.
- This approach tends to be time consuming, particularly for complicated signal flow graphs, but has certain merits (for example, one can interrupt the analysis and go for a coffee without destroying his or her train of thought).

Node Elimination Rules

- *Rule 1:* K branches in series with transmittances $T_1,\ T_2,\ \ldots,\ T_K$ can be replaced by a single branch with transmittance $T_1 T_2 \ldots T_K$, as shown:



(*a*)

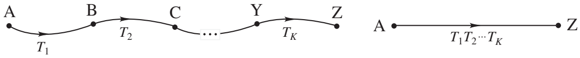
From the first signal flow graph, we have

$$B = T_1 A, \quad C = T_2 B$$

Now if we eliminate B in the second equation using the first equation, we get

$$C = T_1 T_2 A$$

...



(*a*)

$$C = T_1 T_2 A$$

Similarly,

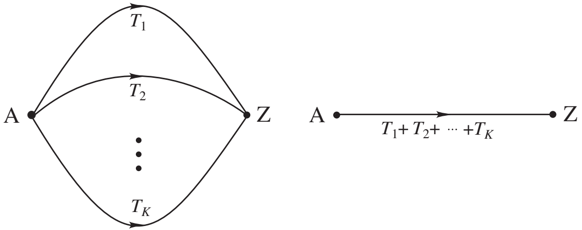
$$D = T_3 C$$

and if we eliminate C we get

$$D = T_1 T_2 T_3 A$$

and so on.

- *Rule 2:* K branches in parallel with transmittances $T_1,\ T_2,\ \ldots,\ T_K$ can be replaced by a single branch with transmittance $T_1 + T_2 + \cdots + T_K$, as shown:



(*b*)

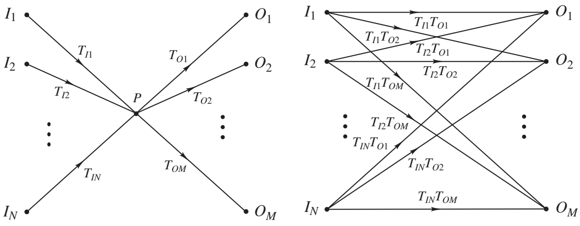
From the first signal flow graph, we have

$$Z = T_1A + T_2A + T_3A + \cdots = (T_1 + T_2 + T_3 + \cdots)A$$

- *Rule 3:* A node with N incoming branches with transmittances T_{I1}, T_{I2}, $\ldots$, T_{IN} and M outgoing branches with transmittances T_{O1}, T_{O2}, $\ldots$, T_{OM} can be replaced by $N \times M$ branches with transmittances

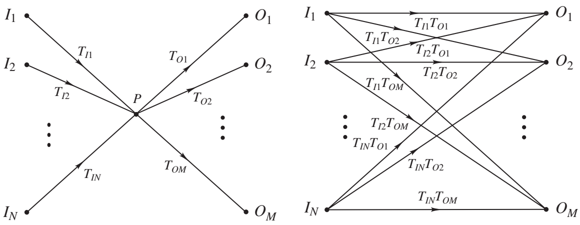
$$T_{I1}T_{O1},\ T_{I1}T_{O2},\ \ldots,\ T_{IN}T_{OM}$$

as shown:



(c)

Node Elimination Rules *Cont'd*



(c)

From the signal flow graph, we have

$$P = T_{I1}I_1 + T_{I2}I_2 + \cdots + T_{IN}I_N$$

and

$$O_1 = T_{O1}P,\ O_2 = T_{O2}P, \ldots,\ O_M = T_{OM}P$$

...

$$P = T_{I1}I_1 + T_{I2}I_2 + \cdots + T_{IN}I_N$$

and

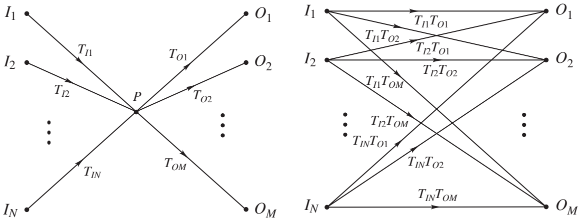
$$O_1 = T_{O1}P,\ O_2 = T_{O2}P, \ldots,\ O_M = T_{OM}P$$

On eliminating variable P, we get

$$\begin{aligned}
O_1 &= T_{I1}T_{O1}I_1 + T_{I2}T_{O1}I_2 + \cdots + T_{IN}T_{O1}I_N \\
O_2 &= T_{I1}T_{O2}I_1 + T_{I2}T_{O2}I_2 + \cdots + T_{IN}T_{O2}I_N \\
&\vdots \\
O_M &= T_{I1}T_{OM}I_1 + T_{I2}T_{OM}I_2 + \cdots + T_{IN}T_{OM}I_N
\end{aligned}$$

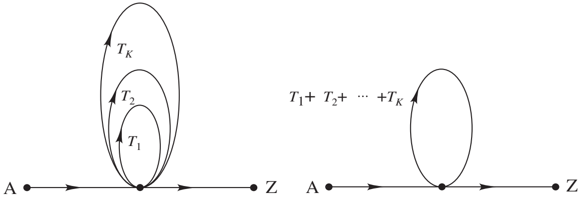
...

$$
\begin{aligned}
O_1 &= T_{I1}T_{O1}I_1 + T_{I2}T_{O1}I_2 + \cdots + T_{IN}T_{O1}I_N \\
O_2 &= T_{I1}T_{O2}I_1 + T_{I2}T_{O2}I_2 + \cdots + T_{IN}T_{O2}I_N \\
\vdots &= \vdots \\
O_M &= T_{I1}T_{OM}I_1 + T_{I2}T_{OM}I_2 + \cdots + T_{IN}T_{OM}I_N
\end{aligned}
$$



(c)

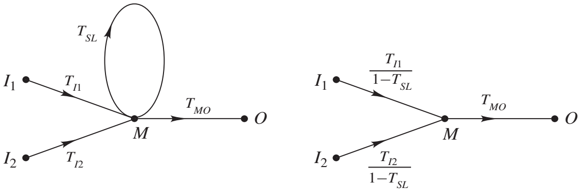
- *Rule 4a:* K self-loops at a given node with transmittances $T_1,\ T_2,\ \ldots,\ T_K$ can be replaced by a single self-loop with transmittance $T_1 + T_2 + \cdots + T_K$, as shown:



(d)

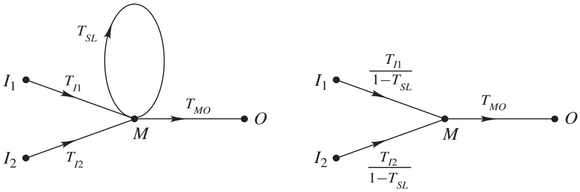
Note: This is to be expected since self-loops are, after all, parallel branches.

- *Rule 4b:* A self-loop at a given node with transmittance T_{SL} can be eliminated by dividing the transmittance of each and every incoming branch by $1 - T_{SL}$ as shown:



(*e*)

Node Elimination Rules *Cont'd*



(*e*)

From the signal flow graph at the left, we have

$$M = T_{I1}I_1 + T_{I2}I_2 + T_{SL}M$$

Solving for variable M, we get

$$M = \frac{T_{I1}}{1 - T_{SL}}I_1 + \frac{T_{I2}}{1 - T_{SL}}I_2$$

which is represented by the signal flow graph at the right.

Strategy

- At each step of the procedure, eliminate the node or nodes that would result in the smallest number of new paths.

  Note: The number of new paths for a given node is equal to the number incoming branches times the number of outgoing nodes, i.e., $N \times M$.

Strategy

- At each step of the procedure, eliminate the node or nodes that would result in the smallest number of new paths.

  Note: The number of new paths for a given node is equal to the number incoming branches times the number of outgoing nodes, i.e., $N \times M$.

- When branches are eliminated draw strokes on them for reckoning purposes.

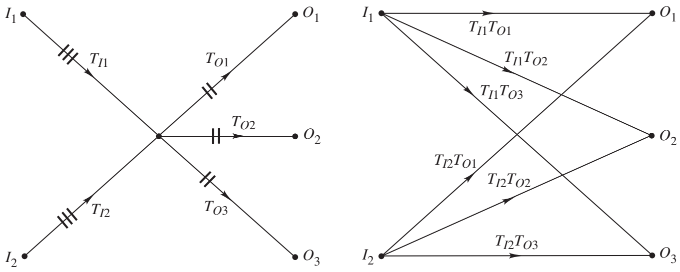
Strategy

- At each step of the procedure, eliminate the node or nodes that would result in the smallest number of new paths.

  Note: The number of new paths for a given node is equal to the number incoming branches times the number of outgoing nodes, i.e., $N \times M$.

- When branches are eliminated draw strokes on them for reckoning purposes.

- At the end of the elimination process, each incoming branch should have as many strokes as there are outgoing branches and each outgoing branch should have as many strokes as there are incoming branches.



I_1
I_2
T_{I1}
T_{I2}
T_{O1}
T_{O2}
T_{O3}
O_1
O_2
O_3
I_1
I_2
$T_{I1}T_{O1}$
$T_{I1}T_{O2}$
$T_{I1}T_{O3}$
$T_{I2}T_{O1}$
$T_{I2}T_{O2}$
$T_{I2}T_{O3}$
O_1
O_2
O_3

Analyze the discrete-time system shown in the figure using the node elimination method.

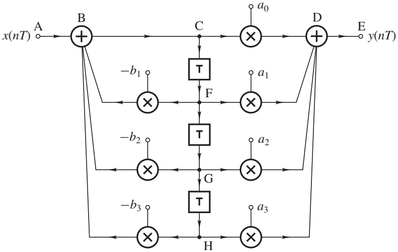


Fig. 6*a*

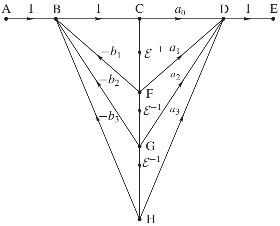


Fig. 6*b*

Solution

- Eliminate node H in Fig. 6a, combine parallel branches in Fig. 8a, and eliminate node G in Fig. 8b:

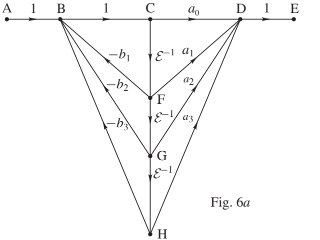


Fig. 6*a*

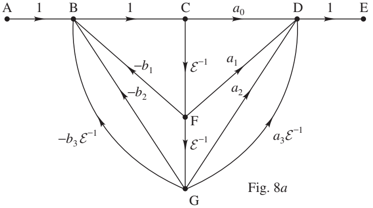


Fig. 8*a*

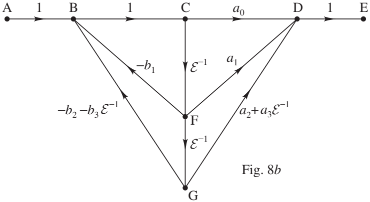


Fig. 8*b*

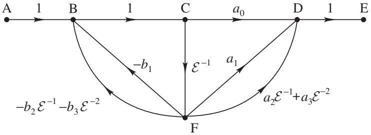


Fig. 8*c*

- Combine parallel branches in Fig. 8c, eliminate node F in Fig. 8d and node C in Fig. 8e:

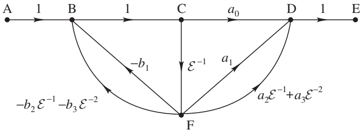


Fig. 8*c*

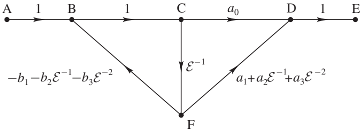


Fig. 8*d*

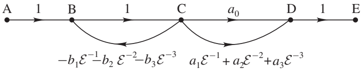


Fig. 8*e*

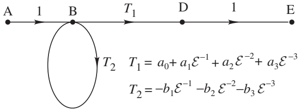


Fig. 8*f*

- Eliminate self-loop and node D in Fig. 8f and node B in Fig. 8g:

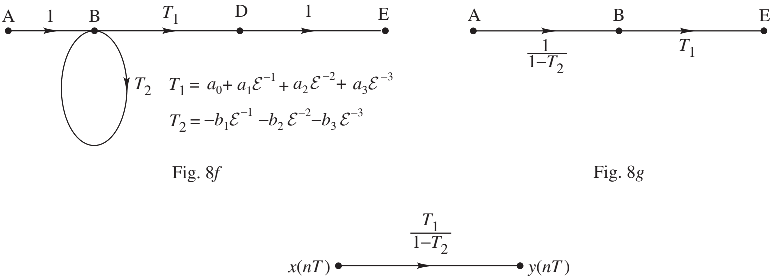


Fig. 8*h*

Hence

$$y(nT) = \left(\frac{T_1}{1 - T_2}\right) x(nT)$$

or

$$(1 - T_2)y(nT) = T_1 x(nT)$$

and, therefore,

$$y(nT) = T_1 x(nT) + T_2 y(nT)$$

Since

$$T_1 = a_0 + \mathcal{E}^{-1}a_1 + \mathcal{E}^{-2}a_2 + \mathcal{E}^{-3}a_3$$

and

$$T_2 = -[\mathcal{E}^{-1}b_1 + \mathcal{E}^{-2}b_2 + \mathcal{E}^{-3}b_3]$$

we obtain

$$\begin{aligned} y(nT) \; = \; & a_0 x(nT) + a_1 x(nT - T) + a_2 x(nT - 2T) + a_3 x(nT - 3T) \\ & -b_1 y(nT - T) - b_2 y(nT - 2T) - b_3 y(nT - 3T) \end{aligned}$$

Mason's Gain Formula

An alternative approach to network analysis involves the use of Mason's gain formula

$$y_j(nT) = \frac{1}{\Delta}\left(\sum_k T_k \Delta_k\right) x_i(nT)$$

where

- $x_i(nT)$ and $y_j(nT)$ are the excitation applied at node i and the response of the system at node j, respectively.

Mason's Gain Formula

An alternative approach to network analysis involves the use of Mason's gain formula

$$y_j(nT) = \frac{1}{\Delta}\left(\sum_k T_k \Delta_k\right) x_i(nT)$$

where

- $x_i(nT)$ and $y_j(nT)$ are the excitation applied at node i and the response of the system at node j, respectively.
- T_k is the transmittance of the kth direct path between nodes i and j,

Mason's Gain Formula

An alternative approach to network analysis involves the use of Mason's gain formula

$$y_j(nT) = \frac{1}{\Delta}\left(\sum_k T_k \Delta_k\right) x_i(nT)$$

where

- $x_i(nT)$ and $y_j(nT)$ are the excitation applied at node i and the response of the system at node j, respectively.
- T_k is the transmittance of the kth direct path between nodes i and j,
- Δ is the determinant of the flow graph, and

Mason's Gain Formula

An alternative approach to network analysis involves the use of Mason's gain formula

$$y_j(nT) = \frac{1}{\Delta}\left(\sum_k T_k \Delta_k\right) x_i(nT)$$

where

- $x_i(nT)$ and $y_j(nT)$ are the excitation applied at node i and the response of the system at node j, respectively.
- T_k is the transmittance of the kth direct path between nodes i and j,
- Δ is the determinant of the flow graph, and
- Δ_k is the determinant of the subgraph that does not touch (has no nodes or branches in common with) the kth direct path between nodes i and j.

The graph determinant Δ is given by

$$\Delta = 1 - \sum_u L_{u1} + \sum_v P_{v2} - \sum_w P_{w3} + \cdots$$

where

- L_{u1} is the loop transmittance of the uth loop,

The graph determinant Δ is given by

$$\Delta = 1 - \sum_u L_{u1} + \sum_v P_{v2} - \sum_w P_{w3} + \cdots$$

where

- L_{u1} is the loop transmittance of the uth loop,
- P_{v2} is the product of the loop transmittances of the vth pair of nontouching loops (loops that have neither nodes nor branches in common),

The graph determinant Δ is given by

$$\Delta = 1 - \sum_u L_{u1} + \sum_v P_{v2} - \sum_w P_{w3} + \cdots$$

where

- L_{u1} is the loop transmittance of the uth loop,
- P_{v2} is the product of the loop transmittances of the vth pair of nontouching loops (loops that have neither nodes nor branches in common),
- P_{w3} is the product of loop transmittances of the wth triplet of nontouching loops, etc.

The graph determinant Δ is given by

$$\Delta = 1 - \sum_u L_{u1} + \sum_v P_{v2} - \sum_w P_{w3} + \cdots$$

where

- L_{u1} is the loop transmittance of the uth loop,
- P_{v2} is the product of the loop transmittances of the vth pair of nontouching loops (loops that have neither nodes nor branches in common),
- P_{w3} is the product of loop transmittances of the wth triplet of nontouching loops, etc.
- The subgraph determinant Δ_k can be determined by applying the formula for Δ to the subgraph that does not touch the kth direct path between nodes i and j.

Example

Analyze the discrete-time system shown in the figure using Mason's gain formula:

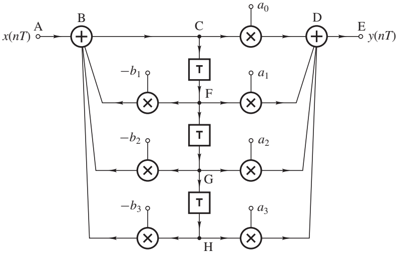


Fig. 6*a*

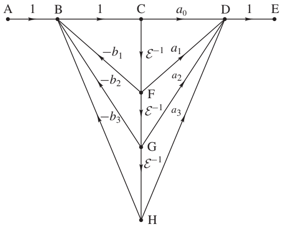


Fig. 6*b*

Solution

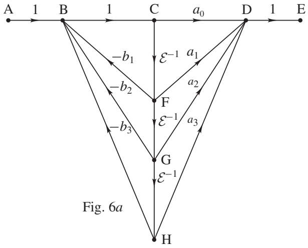


Fig. 6*a*

Direct Paths:
ABCDE, ABCFDE
ABCFGDE, ABCFGHDE

Loops:
BCFB, BCFGB, BCFGHB

NOTE: CDFC is not a loop!

Transmittances of direct paths:

$$T_1 = a_0, \quad T_2 = a_1\mathcal{E}^{-1}, \quad T_3 = a_2\mathcal{E}^{-2}, \quad T_4 = a_3\mathcal{E}^{-3}$$

Transmittances of loops:

$$L_{11} = -b_1\mathcal{E}^{-1}, \quad L_{21} = -b_2\mathcal{E}^{-2}, \quad L_{31} = -b_3\mathcal{E}^{-3}$$

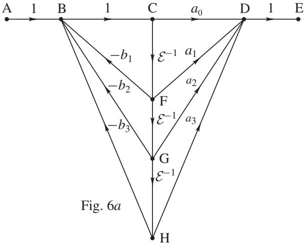


Fig. 6*a*

Direct Paths:
ABCDE, ABCFDE
ABCFGDE, ABCFGHDE

Loops:
BCFB, BCFGB, BCFGHB

NOTE: CDFC is not a loop!

All loops are touching, since branch BC is common to all of them, and so

$$P_{v2} = P_{w3} = \cdots = 0$$

Thus, Mason's gain formula gives

$$\Delta = 1 + b_1\mathcal{E}^{-1} + b_2\mathcal{E}^{-2} + b_3\mathcal{E}^{-3}$$

Example *Cont'd*

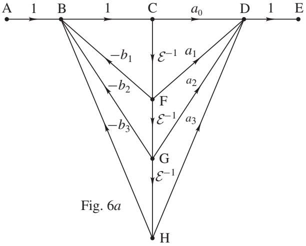


Fig. 6*a*

Direct Paths:
ABCDE, ABCFDE
ABCFGDE, ABCFGHDE

Loops:
BCFB, BCFGB, BCFGHB

NOTE: CDFC is not a loop!

Branch BC is common to all direct paths between input and output. Hence it does not appear in any one of the subgraphs. Thus, no loops are present in the k subgraphs and so

$$\Delta_1 = \Delta_2 = \Delta_3 = \Delta_4 = 1$$

Example *Cont'd*

Therefore, Mason's gain formula gives

$$\begin{aligned} y_j(nT) &= \frac{1}{\Delta}\left(\sum_k T_k \Delta_k\right) x_i(nT) \\ &= \left(\frac{\sum_{i=0}^{3} a_i \mathcal{E}^{-i}}{1+\sum_{i=1}^{3} b_i \mathcal{E}^{-i}}\right) x(nT) \end{aligned}$$

or

$$y(nT) = \sum_{i=0}^{3} a_i \mathcal{E}^{-i} x(nT) - \sum_{i=1}^{3} b_i \mathcal{E}^{-i} y(nT)$$

This slide concludes the presentation.
Thank you for your attention.