Digital Signal Processing
Lecture # 2
Discrete-Time Signals and Systems

Monson H. Hayes
mhh3@gatech.edu

\[ x(n) = \sum_{k=-\infty}^{\infty} x(k) \delta(n - k) \]

What is Digital Signal Processing?

First, let’s ask ourselves: What is a signal?
- In the fields of communications, signal processing, and in electrical engineering more generally, a signal is any time-varying or spatial-varying quantity.
  - Current, voltage, acoustic waveform, light intensity across the focal plane of a camera, ...
- In the physical world, any quantity measurable through time or over space can be taken as a signal.
  - Temperature distribution over an area or a volume or as a function of time, wide velocity, rainfall, ...
- Within a complex society, any set of human information or machine data can be taken as a signal.
  - Stock market prices, number of bytes transmitted across a channel versus time, human population as a function of time, ...

What is a Digital Signal?

- A signal that is a function of a discrete variable, such as \( x(n) \) or \( z(k) \), is a discrete-time signal
- When a discrete-time signal is quantized in amplitude, \( x_q(n) = Q\{x(n)\} \) then we have a digital signal.

What is Digital Signal Processing?

DSP is concerned with the representation of signals as sequences of numbers and with methods of processing those numbers to achieve some desired goal.
Examples of Processing Tasks

- **Information extraction**
  - Trends in data (forecasting and control)
  - Target detection (radar and sonar)
  - Single speaker in a crowd (surveillance)

- **Signal restoration and enhancement**
  - Deconvolution (deblurring, channel equalization)
  - Noise reduction (CD’s?)

- **Signal representation**
  - Signal modeling (speech synthesis)
  - Signal coding (Transmission and storage)
  - Signal restoration, Spectral analysis

Digital Signal Processing Applications

**Telecommunications**
- Echo cancellation
- ADPCM transcoders
- Digital PBX’s
- Line repeaters
- Channel multiplexing
- 1200 to 19200 bps modems
- Adaptive equalizers
- DTMF encoding/decoding
- Data encryption

**Automotive**
- Engine control
- Vibration analysis
- Antiskid brakes
- Adaptive ride control
- Global positioning
- Navigation
- Voice commands
- Digital radio
- Cellular telephones

**Consumer**
- Radar detectors
- Power tools
- Digital audio/tv
- Music synthesizer
- Educational toys

**Industrial**
- Robotics
- Numeric control
- Security access
- Power line monitors

**Medical**
- Hearing aids
- Patient monitoring
- Ultrasound equipment
- Diagnostic tools
- Prosthetics
- Fetal monitors

Objectives

- Our approach, for the most part, concerns the description, analysis, and design of digital systems.

\[ x(n) \rightarrow \text{Digital System} \rightarrow y(n) \]

- We will look at
  - Types of digital systems.
    - Characterizations and properties
  - Analysis of digital systems.
    - Finding responses to given inputs
  - Design of digital systems.
    - Algorithms, filters, computer programs
Digital Signal Processing Systems

Digital System

- Why Digital Signal Processing?
  - High precision, repeatability
  - Design may be simulated in software
  - Easy to modify an algorithm or system design
  - Many operations difficult to implement in analog form are straightforward using digital techniques

Discrete-Time Signals

- A discrete-time signal is an indexed set of real or complex numbers:
  \[ x = \{x(n)\} \quad -\infty < n < \infty \]
  - \( x(n) \) is the \( n \)th sample of the sequence
  - \( x(n) \) is not defined for \( n \) not an integer.

- Sequences may be thought of as numbers stored in a computer array or as a list of numbers in a table.

Discrete-Time Signals (cont)

- Where do these signals come from?
  - Sampling a continuous-time signal (speech, image, audio, radar, sonar, ...)
    \[ x_a(t) \xrightarrow{\text{A/D}} x(n) = x_a(nT_s) \]
  - Signals may be inherently discrete-time
    - Amortization (signal is monthly debt)
    - Population
    - Inventory
    - Stock prices

- Impulse sequence
  \[ \delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases} \]

- Unit step response
  \[ u(n) = \begin{cases} 0 & n < 0 \\ 1 & n \geq 0 \end{cases} \]

- Exponential sequence
  \[ x(n) = \alpha^n \]

- Complex sinusoidal sequence
  \[ x(n) = e^{j\omega_0 n} \]
Discrete-Time Systems

\[ x(n) \xrightarrow{\text{Discrete-Time System}} y(n) = T\{x(n)\} \]

- Numerical algorithm for transforming an input sequence into an output sequence.

  - Delay: \( y(n) = x(n - n_d) \)
  - Modulator: \( y(n) = x(n)\cos(n\omega_0) \)
  - Squarer: \( y(n) = [x(n)]^2 \)
  - Compressor: \( y(n) = x(Mn) \) (also called a downsampler).
  - Smoother: \( y(n) = \frac{1}{3}\{x(n) + x(n - 1) + x(n + 1)\} \)

Basic Signal Manipulations (cont.)

- Multiplication: \( y(n) = x_1(n)x_2(n) \)

  \[ x_1(n) \xrightarrow{x} x_2(n) \xrightarrow{\times} y(n) \]

  - Pointwise multiplication of signals

Basic Signal Manipulations

- Summation: \( y(n) = x_1(n) + x_2(n) \)

  \[ x_1(n) \rightarrow y(n) \rightarrow x_2(n) \]

  - Pointwise addition of signals

Basic Signal Manipulations (cont.)

- Delay: \( y(n) = x(n - n_0) \)

  \[ x(n) \xrightarrow{z^{-n_0}} x(n - n_0) \]

  - Time shift of the sequence

  \[ n \rightarrow x(n) \xrightarrow{n} x(n - 2) \]
Basic Signal Manipulations (cont.)

- **Shift and time reversal**: \( y(n) = x(n_0 - n) \)

  - Basic operation in convolution

\[
x(n) = x(0)\delta(n) + x(1)\delta(n - 1)
\]

Signal Decomposition

- An arbitrary signal may always be decomposed into a summation of weighted and delayed unit samples.

\[
x(n) = x(0)\delta(n) + x(1)\delta(n - 1) + x(2)\delta(n - 2)
\]
Signal Decomposition

- An arbitrary signal may always be decomposed into a summation of weighted and delayed unit samples.

\[ x(n) = \sum_{k=-\infty}^{\infty} x(k) \delta(n - k) \]

Discrete-Time Systems

- Discrete-time systems transform one sequence, the input, into another, the output.

\[ y(n) = T[x(n)] \]

- Generally, we would like to be able to
  - Take a given system and evaluate the response to a given input or to determine the input that will yield a desired response.
  - Design a system to have given input/output characteristics.

Types of System Specifications

- **Input/Output table** (catalog)
  - \( x_1(n) \rightarrow y_1(n) \)
  - \( x_2(n) \rightarrow y_2(n) \)
  - \( x_3(n) \rightarrow y_3(n) \)
  - ...

- **Mathematical rule**
  - \( a \) \( y(n) = [x(n)]^2 \)
  - \( b \) \( y(n) = \frac{1}{2} y(n-1) + x(n) + x(n-1) \)

- **System properties and some input/output pairs**
  - \( T[\delta(n-k)] = nu(n-k) \)
  - Linearity

Difference Equations

- An important class of LSI systems are those for which the input and output satisfy a linear constant coefficient difference equation:
  - General \( N \)th-order LCCDE:
    \[ \sum_{k=0}^{N} a(k) y(n-k) = \sum_{k=0}^{M} b(k) x(n-k) \]

    The constants \( a(k) \) and \( b(k) \) specify the system.

- The output may be computed recursively (assuming \( a(0) = 1 \)) as follows
  \[ y(n) = \sum_{k=0}^{M} b(k) x(n-k) - \sum_{k=1}^{N} a(k) y(n-k) \]

- LCCDE's are the discrete-time counterpart of linear constant coefficient differential equations for continuous-time systems.
Some Simple Systems

**Differentiator** (First Backward Difference):

\[ y(n) = x(n) - x(n-1) \]

\[ y(n) = \frac{1}{3} \{x(n) + x(n-1) + x(n+1)\} \]

**Modulator**:

\[ y(n) = x(n) \cos(n \omega_0) \]

**Savings account**:

\[ y(n) = y(n-1) + \alpha y(n-1) + x(n) \]

- \( y(n) = \) account balance at end of \( n \)th day.
- \( x(n) = \) deposit made on the \( n \)th day.
- \( \alpha = \frac{1}{365} \frac{p}{100} = \) interest rate.
**System Properties**

- **Linear System**: A system is linear if
  \[ T[ax_1(n) + bx_2(n)] = ay_1(n) + by_2(n) \]
  for any complex numbers \(a\) and \(b\) and for any signals \(x_1(n)\) and \(x_2(n)\).

**Examples**

<table>
<thead>
<tr>
<th>Linear</th>
<th>Nonlinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (y(n) = 3x(n))</td>
<td>1. (y(n) = x^2(n))</td>
</tr>
<tr>
<td>2. (y(n) = nx(n))</td>
<td>2. (y(n) = ax(n) + b)</td>
</tr>
<tr>
<td>3. (y(n) = x(n) + x(-n))</td>
<td>3. (y(n) = \log x(n))</td>
</tr>
</tbody>
</table>

**System Properties (cont.)**

- **Shift-Invariance**: A system is shift-invariant if a shift in the input results in a shift in the output
  \[ T[x(n - n_0)] = y(n - n_0) \]
  for any integer \(n_0\) and for any signal \(x(n)\).

- **Interpretation**: The system does not change in time.
Examples

<table>
<thead>
<tr>
<th>Shift-Invariant</th>
<th>Shift-Varying</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( y(n) = x(n) + x(n-1) )</td>
<td>1. ( y(n) = nx(n) )</td>
</tr>
<tr>
<td>2. ( y(n) = x^2(n) )</td>
<td>2. ( y(n) = x(-n) )</td>
</tr>
<tr>
<td>3. ( y(n) = \log x(n) )</td>
<td>3. ( y(n) = x(n) \cos n\omega_0 )</td>
</tr>
</tbody>
</table>

System Properties (cont.)

- **Stability** (BIBO): A system is said to be stable if any bounded input produces a bounded output.
  \[ |x(n)| < M < \infty \implies |y(n)| < N < \infty \]

<table>
<thead>
<tr>
<th>Stable</th>
<th>Unstable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( y(n) = x^2(n) )</td>
<td>1. Savings account</td>
</tr>
<tr>
<td>2. ( y(n) = x(n) - x(n-1) )</td>
<td>2. ( y(n) = nx(n) )</td>
</tr>
<tr>
<td>3. ( y(n) = e^{x(n)} )</td>
<td>3. ( y(n) = \sum_{k=0}^{n} x(k) )</td>
</tr>
</tbody>
</table>

System Properties (cont.)

- **Causality**: A system is said to be causal if for any input \( x(n) \) and for any \( n_0 \) the output \( y(n_0) \) depends only on values of \( x(n) \) for \( n \leq n_0 \).

- **Interpretation**: The system cannot respond prior to a stimulus.

<table>
<thead>
<tr>
<th>Causal</th>
<th>Noncausal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( y(n) = x^2(n) )</td>
<td>1. ( y(n) = x(-n) )</td>
</tr>
<tr>
<td>2. ( y(n) = x(n) + x(n-1) )</td>
<td>2. ( y(n) = x(n) + x(n+1) )</td>
</tr>
<tr>
<td>3. savings account</td>
<td>3. ( y(n) = \sum_{k=0}^{n} x(k) )</td>
</tr>
</tbody>
</table>

System Properties (cont.)

- **Invertibility**: A system is said to be invertible if distinct inputs produce distinct outputs, i.e., with
  \[
  y_1(n) = T[x_1(n)] \quad y_2(n) = T[x_2(n)]
  \]
  then
  \[
  x_1(n) \neq x_2(n) \implies y_1(n) \neq y_2(n)
  \]

- **For invertible systems**: The input is uniquely determined by the output. This is important in the deconvolution problem.
Examples

<table>
<thead>
<tr>
<th>Invertible</th>
<th>Noninvertible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( y(n) = 2x(n) )</td>
<td>1. ( y(n) = x^2(n) )</td>
</tr>
<tr>
<td>2. ( y(n) = \sum_{k=-\infty}^{n} x(k) )</td>
<td>2. ( y(n) = nx(n) )</td>
</tr>
<tr>
<td>3. ( y(n) = x(-n) )</td>
<td>3. ( y(n) = x(n) - x(n-1) )</td>
</tr>
</tbody>
</table>