ECE 421
Introduction to Signal Processing

Dror Baron

Associate Professor

Dept. of Electrical and Computer Engr.

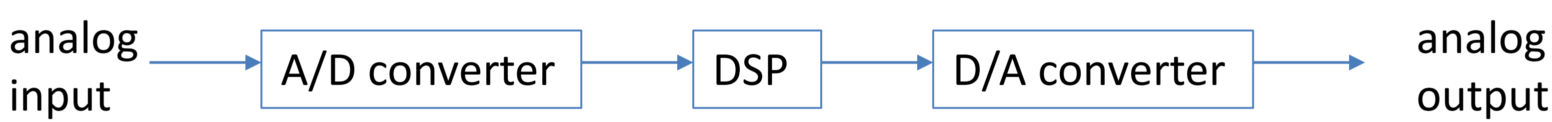
North Carolina State University, NC, USA

What’s ECE421?

Replace analog processing by digital

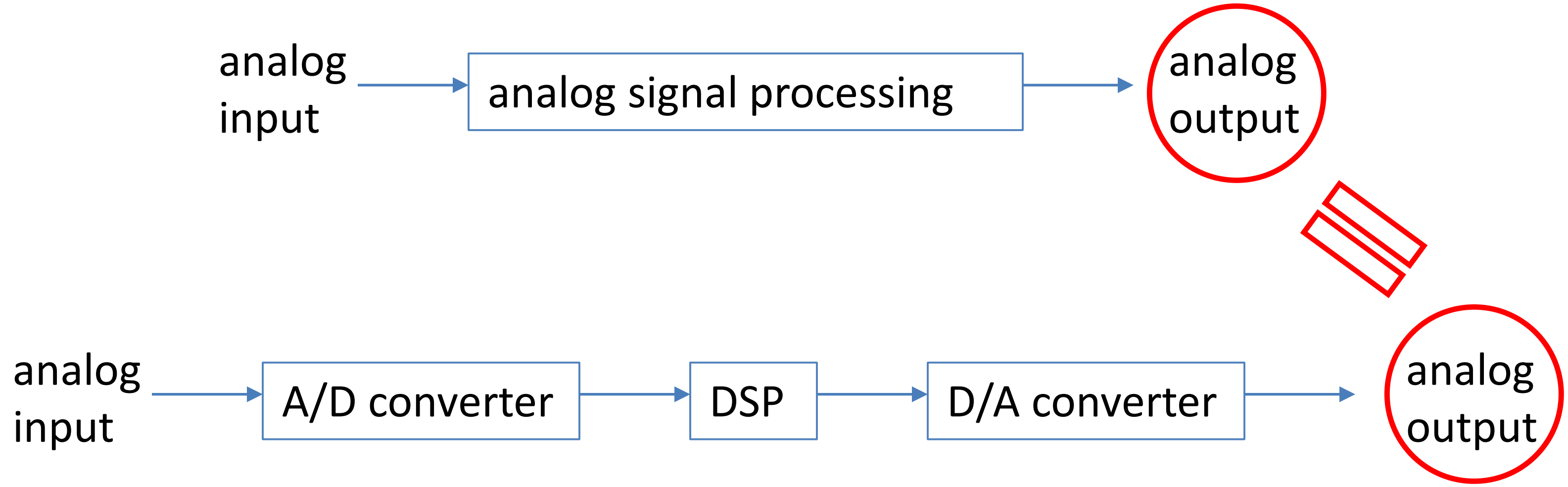
- Signal processing can be performed in analog

- Or digital
  - *Analog to digital conversion* (A/D)
  - *Digital signal processing* (DSP)
  - *Digital to analog conversion* (D/A)



Why replace analog by digital?

- Both systems yield *identical* outputs*!!!*
  - Technical conditions…



- But DSP is cheaper, more robust, everything can be stored, performance is improving all the time…

Frequency perspective

- Electrical engineers often “think” about signals in both time/spatial domain and frequency domain

- Why?

Frequency perspective

- Electrical engineers often “think” about signals in both time/spatial domain and frequency domain

- *Linear time invariant* (LTI) systems
  - Linear/superposition: H(x1+x2) = H(x1) + H(x2)
  - Time invariant: shift(H(x)) = H(shift(x))

- Many systems are well-approximated as LTI

Frequency perspective

- Electrical engineers often "think" about signals in both time/spatial domain and frequency domain

- Property #1 – sinusoids processed by LTI systems are still sinusoids, they are merely amplified somehow

- Take several sinusoids at the input
  - Linear system → output is superposition of individual outputs
  - Each sinusoid is amplified → superposition of amplified sinusoids

- Input superpositions of sinusoids → "easy" to understand output

Frequency perspective

- Electrical engineers often "think" about signals in both time/spatial domain and frequency domain

- Property #2 – LTI systems can be represented as *convolution*
  - Convenient to work with convolution, especially because in the frequency domain it boils down to multiplication

- Bottom line: LTI systems appear in many engineering systems & mathematically tractable frequency perspective

Motivation for ECE421

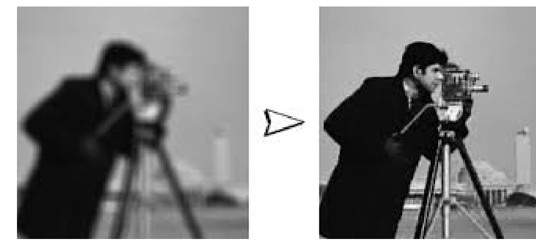
Real story

- Microwave radio links used for "last mile" communication
  - Typical use - link base stations in rural areas without fiber network
  - Data rates typically tens/hundreds Mbps

- Before late 90's, microwave link modems were analog
- Instructor worked at startup that designed digital modems for microwave links
  - 5x reduction in power → less power transmitted (cleaner EM spectrum) or can use less hardware

Applications

- *Deblurring* – handshake introduce blurring artifacts
  - Observed image=true image*kernel+noise
  - Goal: estimate true image from noisy observations



- *Seismic* exploration/visualization/imaging
  - Sensors send vibrations into ground, other sensors measure vibrations
  - Goal is to estimate geological structure
  - Useful to decide where to drill for oil, locate earthquake zones, …

- *Medical imaging* (replace “ground” by “patient”)
- Communications (phones), image processing (cameras), video, defense (radar, signals intelligence), finance…

Things you'll learn about

- AM radio (example revisited during course)
  - Narrow band signal (~10 KHz) modulated at carrier (~1 MHz)
  - Will learn to sample at ~20 K instead of ~2M samples/sec

- Multipath in mobile phones (discussed as digital filter)
  - Urban environment with comm signal bouncing between buildings
  - Can perform "echo cancelation" with digital processing; unrealistic with analog hardware due to changing nature of environment

- Sneak peak at compressed sensing
  - Modern signal acquisition approach
  - State of art algos often allow 10x reduction in sensing rates

Matlab example

- Start with superposition of two sinusoids
- Add noise
- In Fourier domain, coefficients corresponding to two sinusoids are bigger than other noise-induced coeffs
- Denoising approach – truncate small Fourier coeffs
- *Matlab script available on course webpage*

Administrative Details

Introduction

- Many resources on course webpage
  - Syllabus - updated
  - *Tentative* schedule – updated
  - Slides & handouts & supplements

- Webwork – homeworks & quizzes
  - Are quizzes good in online format?

- Projects

- Grade structure

Some details

- Prereqs:
  - ECE 301 (linear systems)
  - Matlab – tutorials available from webpage

- Textbook
  - Proakis & Manolakis
  - Any recent edition should be fine

More details

- Change in grade structure:
  - Less weight on Webwork (intended to motivate you)
  - More weight on projects
  - More tests → smaller per-test weight

- Occasional “active learning” exercises in class
  - Normal semester: students discuss in pairs/triples
  - After 2-3 minutes I poll responses / volunteers / etc.
  - Online semester: I’ll give you time to pause video
  - Solutions on course webpage ☺

Expectations

- ECE 421 more open ended than some other courses
  - Less emphasis plugging numbers into formulas
  - More emphasis on deriving new results
  - Evaluating trade-offs critically
  - Applying knowledge to problems you haven’t seen before
  - More projects

- This style can build strong foundation in signal processing

Signals and Systems

[Reading material: Sections 1.1-1.4]

Signals

- *Signal* – function of time (or space)
  - Example: $x_1(t)=\sin(10\pi t)$
- Real-world signals are complicated
- Major theme of course – can express some signals compactly/sparsely as superpositions of sinusoids

- Types of signals
  - *Multidimensional* – function of multiple inputs (e.g., image)
  - *Multichannel* – has several outpus (e.g., complex valued signal)
  - *Continuous time* (analog also features continuous amplitude)
  - *Discrete time*
  - *Digital* - discrete time and discrete valued

Active learning (based on Problem 1.1 in textbook)

- Classify signals below as: one/multi dimensional; single/multi channel; continuous/discrete time; digital/analog amplitude
  - Closing prices of stocks?
  - Color movie?
  - Weight/height measurements of child every month?
- Solution on webpage

Systems

- *System* - device that responds to stimulus
- Signals are inputs and outputs of systems
- Can have various properties:
  - *Linear* or *non-linear*
  - *Causal* or *anti-causal*
  - *Random* or *deterministic* (we focus on latter)
  - *Time invariant* or not
- *Digital systems* can be implemented in an algorithm on a computer
- We focus on digital processing, which can emulate analog

Frequencies and Periodicity

Continuous time sinusoids

- *Continuous time sinusoid*: $x_a(t)=A\cos(\Omega t+\theta)$
  - A – *amplitude*
  - Ω - *frequency* (radians per unit time)
  - t – *time*
  - θ - *phase*

- Can express w/cycles per unit time, $x_a(t)=A\cos(2\pi F t+\theta)$

- Cont. time sinusoids periodic w/period $T_p=1/F$

- Increasing F → shorter period, faster oscillations

Discrete time sinusoids

- *Discrete time sinusoid*: $x(n)=A\cos(\omega n+\theta)$
  - ω - frequency (radians per sample)
  - n – discrete time index

- Can express w/cycles per sample, $x(n)=A\cos(2\pi f n+\theta)$

- Summary of notation for frequencies:

	Radians	Cycles
Continuous time	Ω	F
Discrete time	ω	f

What is periodicity in discrete time?

- Discrete time sinusoid periodic if f is rational number

- Consider $s(t)=e^{j\Omega t}$ with period T_p

- Let's accelerate the signal by factor k: $s_k(t)=e^{jk\Omega t}$

- New signal $s_k(t)$ periodic with period T_p/k
- $s_k(t)=s_k(t+lT_p/k)$ for integers k,l

Example (based on Problem 1.2 in textbook)

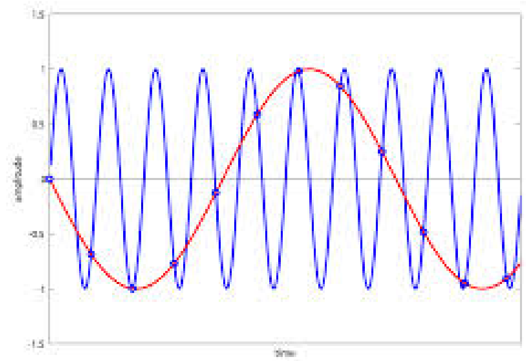
- What's the fundamental period of the following signals?
  - $\cos(0.01\pi n)$?
  - $\cos(30\pi n/105)$?
  - $\sin(3n)$?

Tougher example

- What's the fundamental period of
  $x(n)=0.1\cos(65\pi n/40)+12\sin(37 \pi n/4)$?

Aliasing

- Discrete time sinusoids w/frequencies ω separated by 2π radians per sample are indistinguishable – called *aliasing*
  - Or f separated by 1 cycle per sample; or $\omega=2\pi k$ for integer k



- Example that demonstrates aliasing
  - $x_1(t)=\sin(0.01\pi t)$, $x_2(t)=\sin(2.01\pi t)$
  - x_1 has period of 200, because $x_1(200+t)=x_1(t)$
  - x_2 has period 2/2.01
  - Sample with sampling frequency $F_s=1 \rightarrow x_1(n)=\sin(0.01\pi n)$
  - Similarly, $x_2(n) = \sin(2.01\pi n) = \sin(2\pi n+0.01\pi n) = \sin(0.01\pi n) = x_1(n)$

- Visual demos of aliasing: http://www.youtube.com/watch?v=jHS9JGkEOmA

A/D and D/A Conversion

Big picture

- Many real-world signals are analog
  - Speech signals, images, video, seismic data, climate measurements, …
- To enjoy benefits of DSP (reliable, cheap, fast, reproducible,…)
  - Convert from analog to digital (A/D)
  - Perform digital signal processing
  - Convert from digital back to analog (D/A)

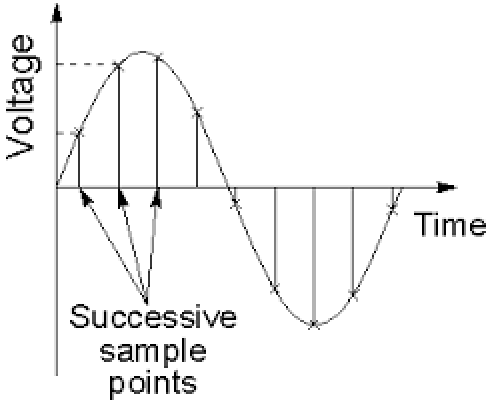
analog input → A/D converter → DSP → D/A converter → analog output

- Will soon see when this is equivalent to analog processing

analog input → analog signal processing → analog output

A/D conversion

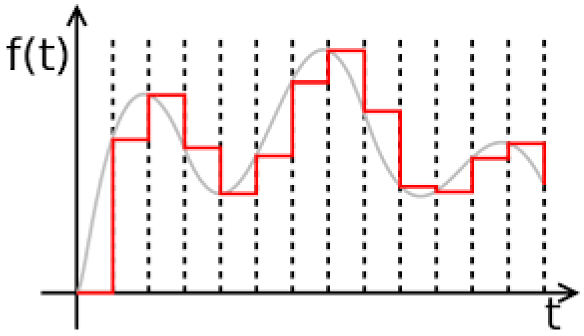
- *Analog to digital* (A/D) conversion comprised of three parts



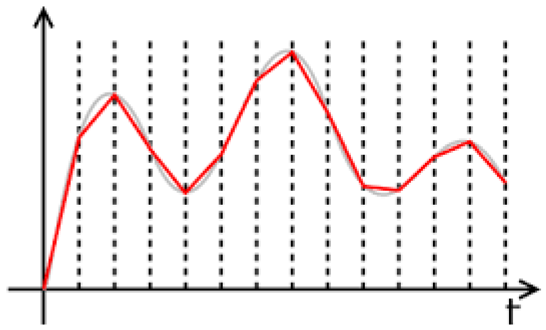
- *Sampling* – $x(n)=x_a(nT)$ with sampling interval T
  - Sampling involves analog hardware
  - Non-uniform sampling can be used but complicated
- *Quantization* – truncate/round $x(n)$ to discrete valued $x_q(n)$
  - Uniform quantizers $x_q(n)=\lfloor x(n)/\Delta \rfloor$ commonly used
    - $\lfloor \cdot \rfloor$ rounds down; quantizer step size Δ
  - Non-uniform quantizers can use fewer levels
- *Coding* – translate discrete valued $x_q(n)$ to bits
  - Data compression allocates fewer bits to common quantization levels, more bits to rare ones (just like Morse code)

D/A conversion

- How can *digital to analog* (D/A) converters interpolate between samples?
- *Zero order hold* – maintain x(n) at output for T time
  - Results in staircase-like pattern at output



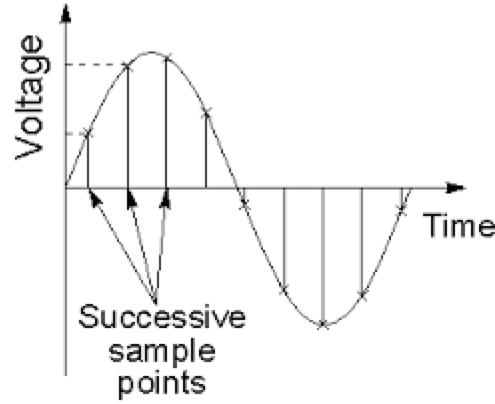
- *First order hold* “connects the dots”
  - Output becomes smoother (continuous)
  - Will see later what this means in frequency domain



- Higher order *interpolation* can be used
  - Will see that sinc is theoretically appealing

Sampling

- *Sampling* – $x(n)=x_a(t=nT)$
  - Sampling interval T
  - Sampling rate $F_s=1/T$
  - Sampling times $t=nT=n/F_s$



- Consider sampling a cosine, $x_a(t)=A\cos(2\pi Ft+\theta)$

$$x(n) = x_a(t=nT) = A\cos(2\pi nF/F_s+\theta)$$

- Contrast to discrete cosine, $x(n)=A\cos(2\pi fn+\theta)$ → $f=F/F_s=FT$
  - Remark: because $\Omega=2\pi F$ and $\omega=2\pi f$ → $\omega= \Omega T$

- Want $f\in(-0.5,0.5)$, requires $F/F_s\in(-0.5,0.5)$
- *Need $-0.5F_s < F < 0.5F_s$ or $F_s>2|F|$ to avoid aliasing*

Example (based on Example 1.4.2 in textbook)

- Consider $x_a(t)=3\cos(100\pi t)$

1. What's minimum sampling rate required to avoid aliasing?
2. Suppose F_s=200 Hz, what's the discrete time signal?
3. Suppose F_s=75 Hz, what's the discrete time signal?

The Sampling Theorem

The sampling theorem

- Consider *band limited* signal; all frequencies are below F_{max}
- Can find $F_s=1/T$ large enough such that $F_s>2F_{max}$
  - Every analog F can be determined from corresponding discrete f

- <u>Theorem:</u> [Shannon, Nyquist, Whittaker, Kotelnikov]

If highest frequency in $x_a(t)$ is $F_{max}=B$, and we sample at rate $F_s>2F_{max}=2B$, then $x_a(t)$ can be recovered <u>perfectly</u>,

$$x_a(t)=\Sigma_n \underbrace{x_a(n/F_s)}_{x(n)} g(t-n/F_s),$$

where $g(t)=\sin(2\pi Bt)/(2\pi Bt)$

- F_s called *Nyquist rate*
- $g(t)$ involves non-causal sinc interpolation → not implementable

Active learning (Example 1.4.4 in textbook)

- Consider $x_a(t)=3\cos(2000\pi t)+5\sin(6000\pi t)+10\cos(12000\pi t)$

1. What is the Nyquist rate?

2. We use $F_s=5000$ Hz, what is the discrete time signal?

3. What analog signal is obtained with ideal sinc interpolation?

Discrete Time Signals and Systems

[Reading material: Sections 2.1-2.5]

General comments about this material

- DSP can help emulate end to end analog systems → focus on discrete time signals & systems

- Much of this material should be review → fast paced

- Our emphasis will be notations and terminology used in book

- Let’s cover this quickly and move to new material ☺

Notation for discrete time signals

- Discrete time signal can be expressed in different ways

- Function, $x(n)=n+13$

- Table representation

n	…	-2	-1	0	1	2	…
x(n)		1	0	3	1	4	

- Sequence $x(n)=\{\ldots, 0, \underline{0}, 1, 4, 2, 1, 0, 0, \ldots\}$
  - Underline (arrow in book) points to time origin (n=0)

Some standard discrete time signals

- Unit impulse sequence $\delta(n) = \begin{cases} 1\ n = 0 \\ 0\ else \end{cases}$
- Unit step $u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$
- Unit ramp $u_r(n) = nu(n) = \begin{cases} n & n \geq 0 \\ 0 & n < 0 \end{cases}$
- Exponent $x(n) = a^n$
  - Can be complex, can write

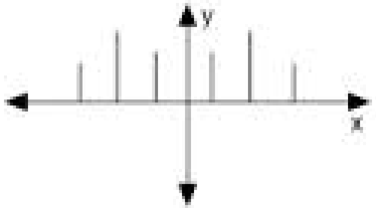
$$(re^{j\Phi})^n = r^n e^{j\Phi n} = r^n[\cos(\Phi n) + j\sin(\Phi n)]$$

More definitions

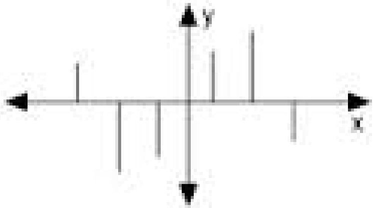
- *Energy* $E = \sum_{n=-\infty}^{+\infty} |x(n)|^2$
  - Often called squared-ℓ_2 norm, $||x||_2=E^{0.5}$
- *Power* $P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{+N} |x(n)|2$
  - Average energy per sample

- Periodic signal x(n+N)=x(n), ∀n

- *Symmetric* (even) signal x(-n)=x(n) , ∀n
- *Antisymmetric* (odd) x(-n)=-x(n)
  - Note that x(0)=-x(-0)=0



Even Symmetry



Odd Symmetry

Operations on discrete time signals

- *Time shift* x(n-k)
- *Folding* or *reflection* x(-n)
- *Time scaling* or *down-sampling* x(μn) for integer μ
- *Addition* or *sum* $y(n)=x_1(n)+ x_2(n)$
- *Product* $y(n)=x_1(n)x_2(n)$
- *Scaling* y(n)=Ax(n)

Discrete time *systems*

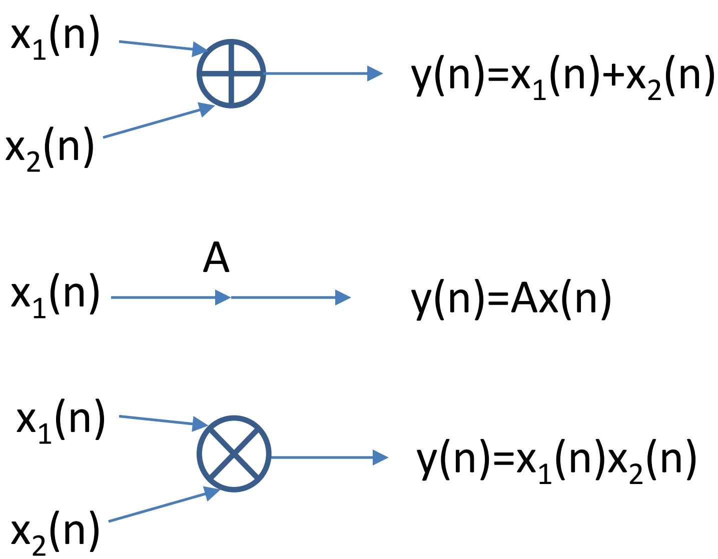
- Discrete time systems operate on discrete time signals

- Input x(n) transformed to output y(n)

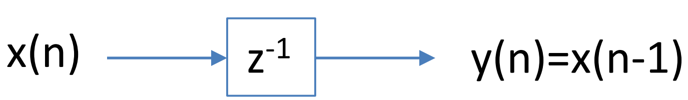
- Can write y(n)=T[x(n)]
  - T for transformation

Sketches of systems

- Can sketch discrete time systems using components below

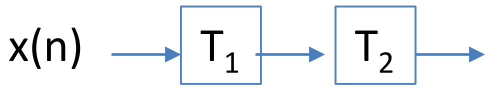


- Sometimes want to add *memory* to system



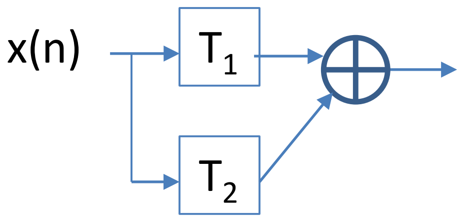
Connecting systems

- *Cascade*



$y(n)=T_2[T_1[x(n)]]$

- *Parallel* connection



$y(n)=T_1[x(n)]+ T_2[x(n)]$

Types/Properties of Discrete Time Systems

Types of systems

- *Static* – memoryless
- *Dynamic* – contains memory
- *Time invariant* – $y(n)=T[x(n)] \rightarrow y(n-k)=T[x(n-k)], \forall x(n), k$
  - Suffices to prove for $k=1$
- *Time variant* – not invariant (example coming up)

Example time variant system (see supplement)

- System $y(n)=nx(n)$
- Input $x(n)=\delta(n)$
  - $n \neq 0$: $x(n)=0 \rightarrow y(n)=0$
  - $n=0$: $x(n)=1 \rightarrow y(n)=0 \cdot 1=0$
  - *Output always zero* (even if we apply time shift by k, any k)
- Input $x(n)= \delta(n-k)$ *it's one when n=k*
  - $n \neq k$: $x(n)=0 \rightarrow y(n)=0$
  - $n=k$: $x(n)=1 \rightarrow y(n)=k \cdot 1=k$
  - *Output not always zero*
- Key point: k-shifted input doesn't yield k-shifted output

More types of systems

- *Linear* – $T[ax_1(n)+bx_2(n)]=aT[x_1(n)]+bT[x_2(n)]$
  - Also called *superposition*
  - Must hold for *all* scalars a, b, signals $x_1(n)$, $x_2(n)$
  - Suffices to prove $T[x_1(n)+x_2(n)]=T[x_1(n)]+T[x_2(n)]$ and $T[ax(n)] = aT[x(n)]$
- *Causal* – output depends only on present/past inputs
  - Also have *anti-causal* (depends on present/future), *non-causal*

Stable Discrete Time Systems

Stability

- Intuitively, want “well behaved” system
- Various types of stability possible

- *Bounded input bounded output* (BIBO)
  - Common way to evaluate stability
  - Output must be bounded for *all* bounded inputs

Example non-BIBO system (see supplement)

- System $y(n)=y(n-1)+x(n)$
- Input $x(n)=u(n)$ *step input*
  - $n<0$: $y(n)=0$
  - $n=0$: $y(0)=y(-1)+x(0)=0+1=1$
  - $n=1$: $y(1)=y(0)+x(1)=1+1=2$
  - $n=2$: $y(2)=y(1)+x(2)=2+1=3$
  - …
  - Can show $y(n)=n+1$ for $n\geq 0$ → $y(n)=u_r(n)+u(n)$ ramp+step
- <u>Key point</u>: bounded input & unbounded output → not BIBO

Same example *another input*

- System y(n)=y(n-1)+x(n)

- Input $x(n)=\delta(n)$ *impulse instead of step*
  - n<0: y(n)=0
  - n=0: y(0)=y(-1)+x(0)=0+1=1
  - n=1: y(1)=y(0)+x(1)=1+0=1
  - n=2: y(2)=y(1)+x(2)=1+0=1
  - Can show y(n)=1 for $n \geq 0 \rightarrow$ y(n)=u(n)

- Bounded input (impulse) & bounded output (step)

- BIBO unstable system *can* have bounded output
  - Only need *one* bad input to demonstrate non-BIBO

Example BIBO system (modified system)

- We saw that y(n)=y(n-1)+x(n) *not* BIBO stable
- Slightly modified system: y(n)=***0.5***y(n-1)+x(n)

- Input x(n)=u(n) *step*
  - n<0: y(n)=0
  - n=0: y(0)=0.5y(-1)+x(0)=0+1=1
  - n=1: y(1)=0.5y(0)+x(1)=0.5+1=1.5
  - n=2: y(2)=0.5y(1)+x(2)=0.75+1=1.75
  - Can show $y(n)=2-0.5^n$ for $n \geq 0$

- Modified system can be shown to be BIBO stable

Linear Time Invariant (LTI) Discrete Time Systems

Linear time invariant (LTI) systems

- Many real-world systems can be approximated as LTI
- Convenient mathematical properties
- Can be expressed as *convolution*

$$y(n) = \sum_{k=-\infty}^{+\infty} x(k)h(n-k)$$

  - System H coincides to impulse response h(·)
  - h called *convolution kernel*
  - Can be computed as *impulse response*

Example impulse responses

- Recall two systems:
  - H_1: $y(n)=y(n-1)+x(n)$
  - H_2: $y(n)=\mathbf{\mathit{0.5}}y(n-1)+x(n)$
- First system:
  - Already saw impulse response $h_1(n)=u(n)$ step function
- Second system:
  - Let's show $h_2(n)=0.5^n u(n)$

Properties of convolution

- *Identity* operator: $x(n)*\delta(n)=x(n)$
- *Time shift*: $x(n)*\delta(n-k)=x(n-k)$
- *Commutative*: $x(n)*h(n)=h(n)*x(n)$
- *Associative*: $[x(n)*h_1(n)]*h_2(n)=x(n)*[h_1(n)*h_2(n)]$
- *Distributive*: $x(n)*[h_1(n)+h_2(n)]=x(n)*h_1(n)+x(n)*h_2(n)$

Properties of LTI systems

- LTI system H is causal iff (if and only if) h(n)=0 for n<0
- LTI system H is BIBO stable iff $\sum_{k=-\infty}^{\infty} |h(k)| < \infty$
- *Finite impulse response* (FIR) systems have finite duration
- *Infinite impulse response* (IIR) – unbounded duration; can sometimes be implemented recursively

Example BIBO system

- Recall two systems:
  - H_1: y (n)=y (n-1)+x(n)
  - H_2: y (n)=***0.5***y (n-1)+x(n)

- First system: $h_1(n)=u(n)$
  - $\sum_{k=-\infty}^{\infty}|h_1(k)|$ is infinite → not stable

- Second system: $h_2(n)=0.5^n u(n)$
  - $\sum_{k=-\infty}^{\infty}|h_2(k)| = 2$ → BIBO stable

Implementing Discrete Time Systems

[Reading material: Section 2.5]

Difference equations

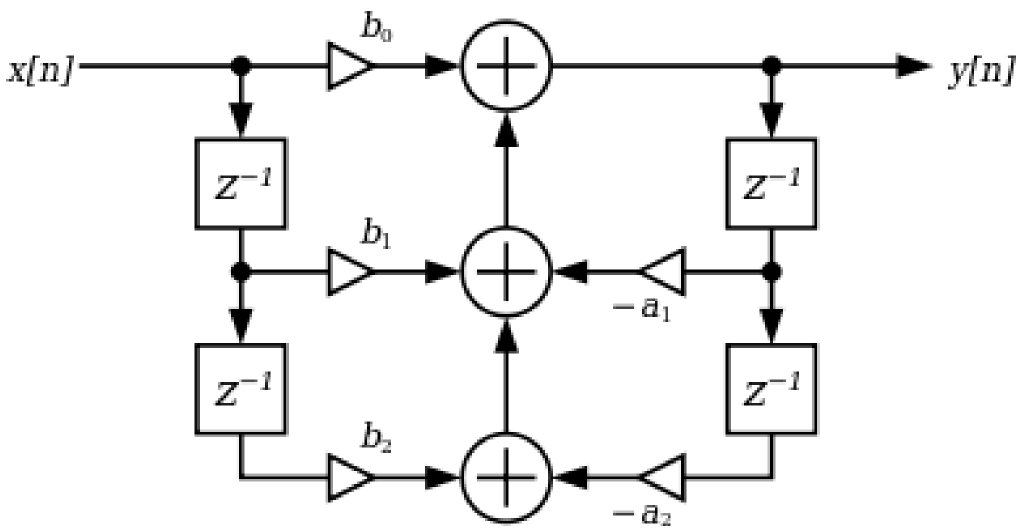
- Common type of LTI system (convention: a_0=1)

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

- Can be solved by splitting into components
  - *Zero input response* – reaction to initial conditions from feedback of {a} coefficients
  - *Zero state response* – assumes zero initial conditions

Direct form I

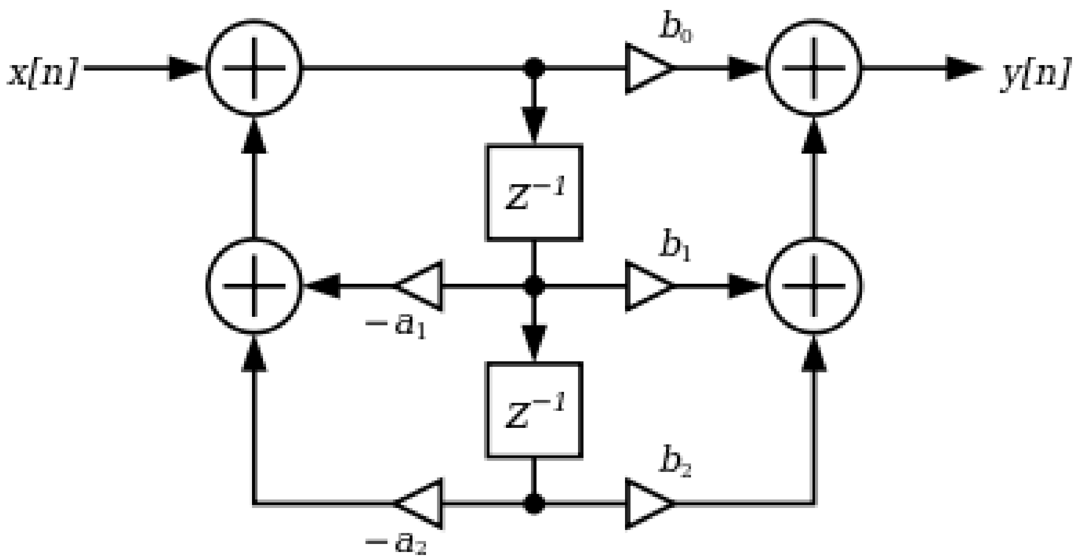
- Difference equation yields following implementation
  - $\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$
  - $v(n)=b_0x(n)+b_1x(n-1)+\ldots$
  - $y(n)=v(n)-a_1y(n-1)-a_2y(n-2)-\ldots$
- Known as *direct form I*



Direct form II

- Left and right sides are commutative → can swap sides

  x(n)*L(n)*R(n)=x(n)*R(n)*L(n)
- Known as *direct form II*



- Note savings in memory units; they often consume resources (space / power)

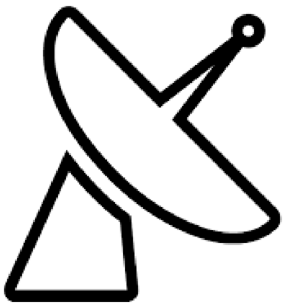
Correlation

[Reading material: Section 2.6]

Motivation

- *Correlation* measures similarity between signals
  - Often used with signals that feature randomness
  - Book takes deterministic (non-random) viewpoint

- Radar application/motivation
  - Signal x is transmitted
  - Reflected off target with delay D, attenuation a
  - Additive noise w(n)
  - $y(n)=ax(n-D)+w(n)$



- Correlation tells us how similar y is to versions of x delayed by different amounts

Revisiting real story

- Key component in microwave link modems is measuring delay between devices
- Radios have slightly different clocks → delay D varies
- Want to sample incoming communication signal “right” time
  - Sample at correct time → interpolation (e.g. sinc) works well
  - Incorrect synchronization → interpolation yields garbage

- Synchronization approach
  - Periodically transmit sequence with spiky correlation properties
    - This is (small) overhead…
  - Receiver occasionally sees spike
  - Receiver can estimate delay D relatively well

Matlab example (visualizing correlation)

- Take signal x and add low-amplitude noise
  - Scatter plot resembles line
- Noise amplitude = signal amplitude
  - Elliptical plot
- Large amplitude noise
  - Circular plot → uncorrelated
- *Matlab script available on course webpage*

Correlation and its Properties

Some definitions

- *Cross correlation*, $r_{xy}(l)=\Sigma_n x(n)y(n-l)$
  – Can be re-expressed, $r_{xy}(l)=\Sigma_n x(n+l)y(n)$

- Let's swap roles of sequences x and y
  – $r_{yx}(l) = \Sigma_n y(n)x(n-l) = \Sigma_n y(n+l)x(n) = r_{xy}(-l)$

- *Autocorrelation*
  – Correlation between sequence and itself
  – $r_{xx}(l)=\Sigma_n x(n)x(n-l) =\Sigma_n x(n+l)x(n)$
  – Due to symmetry, $r_{xx}(l)=r_{xx}(-l)$ → even function

Active learning

- Consider $x(n) = \begin{cases} 2, & n = 0 \\ 1, & n = 1 \\ 0, & else \end{cases}$
- Compute $r_{xx}(l)$ for:
  - l=0
  - l=-1
  - l=+1
  - Other

Properties

- Sum of squares of sequences expressed using correlation
- Will show $\Sigma_n[ax(n)+by(n-l)]^2 = a^2 r_{xx}(0)+2ab r_{xy}(l)+b^2 r_{yy}(0)$

Correlation and energy

- Relation to energy: $r_{xx}(0)=E_x$, $r_{yy}(0)=E_y$
- Energy is non-negative → $\Sigma_n[ax(n)+by(n-l)]^2 \geq 0$
  - $r_{xx}(0)(a/b)^2+2r_{xy}(l)(a/b)+r_{yy}(0)(1) \geq 0$
  - Will use quadratic eq. to show $|r_{xy(l)}| \leq \sqrt{r_{xx}(0)r_{yy}(0)} \leq \sqrt{E_x E_y}$

Normalized correlation

- Correlation greatest for x(n)=y(n) → $|r_{xx}(l)| \leq r_{xx(0)} = E_x$

- Normalized autocorrelation, $\rho_{xx(l)} = \frac{r_{xx}(l)}{r_{xx}(0)} \in [-1,1]$

- Normalized cross-correlation, $\rho_{xy(l)} = \frac{r_{xy}(l)}{\sqrt{E_x E_y}} \in [-1,1]$

- Revisit active learning; compute $\rho_{xx}(1)$

Correlation in LTI systems

- Cross correlation, $r_{xy}(l) = \Sigma_n x(n)y(n-l) = \Sigma_n x(n+l)y(n)$
- Flipped version, $\tilde{y}(n) = y(-n)$
- Express as convolution: $r_{xy(l)} = \{x * \tilde{y}\}(l)$

- Consider LTI system H with input x, output y

$$x(n) \longrightarrow \boxed{H} \longrightarrow y(n)$$

- $r_{yx} = y * \tilde{x} = (h * x) * \tilde{x} = h * (x * \tilde{x}) = h * r_{xx}$
- Similarly, $r_{xy} = \tilde{h} * r_{xx}$

$$r_{yy} = y * \tilde{y} = (h * x) * \left(\tilde{h} * \tilde{x}\right) = \left(h * \tilde{h}\right) * (x * \tilde{x}) = r_{hh} * r_{xx}$$

- Useful for power spectrum in communications systems

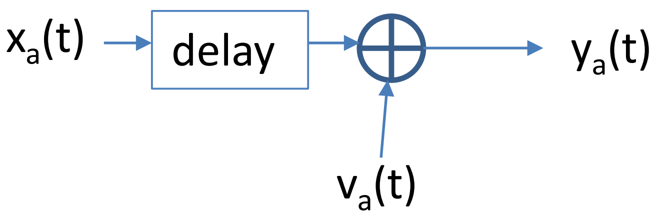
Computation of correlation

- Cross correlation expressed as convolution, $r_{xy} = x * \tilde{y}$
- Will see how fast Fourier transform (FFT) provides fast computation of convolution
- Correlation typically computed via FFT

Radar Example

Radar example (Problem 2.65 in textbook)

- Radar transmission, $x_a(t)$
- Received signal, $y_a(t)=\alpha x_a(t-t_d)+v_a(t)$
  - t_d time delay
  - α attenuation
  - $v_a(t)$ noise



- Convert to discrete time (sampling)
  - $x(n)=x_a(nT)$
  - $Y(n)= \alpha x(n-D)+v(n)$
- *Matlab script available on course webpage*

Radar example – Part 2

a) How to estimate delay D with cross-correlation $r_{xy}(l)$?

b) Simulate input x(n)={1,1,1,1,1,-1,-1,1,1,-1,1,-1,1}
   - Gaussian noise v(n) with variance=0.01
   - Matlab: v(n)=sqrt(variance)*randn(N,1);
   - Generate y(n), $0 \leq n \leq 199$, α=0.9, D=20

c) Compute and plot cross-correlation; estimate delay D

Radar example – Part 3

d) Repeat with variance 0.1 and 1

e) Repeat with modified sequence

X={-1,-1,-1,1,1,1,1,-1,1,-1,1,1,-1,-1,1}