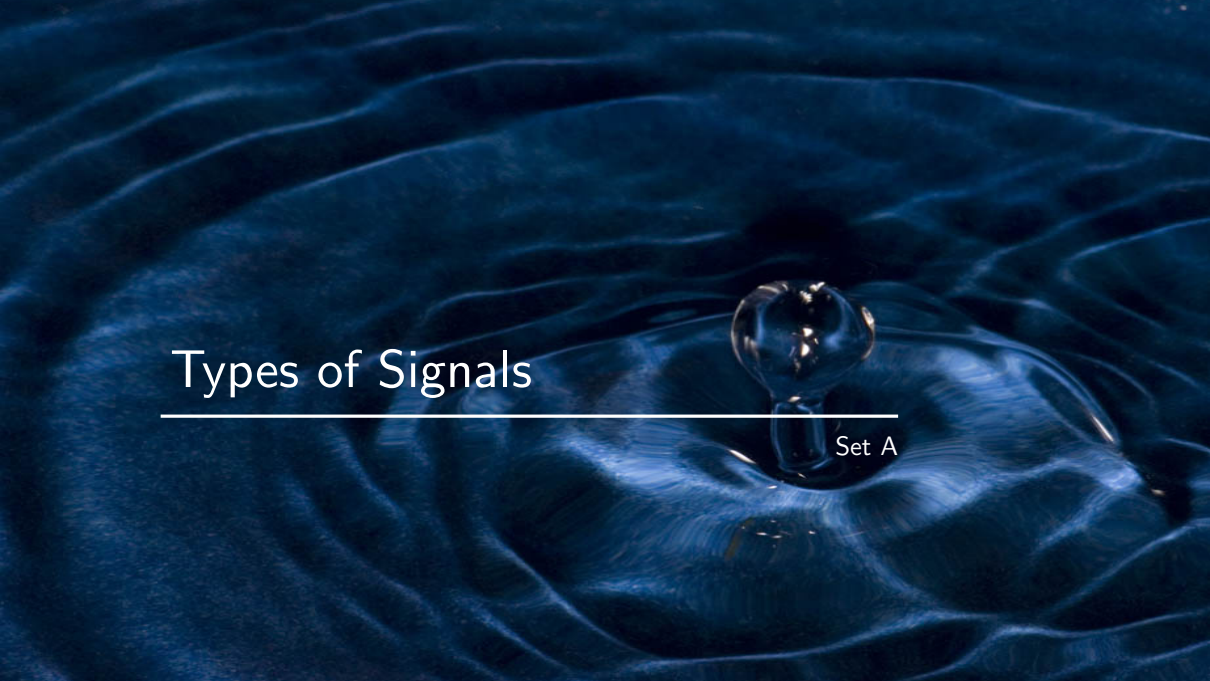
Signals & Systems: Discrete-Time

ELEC 301

Richard G. Baraniuk
Rice University

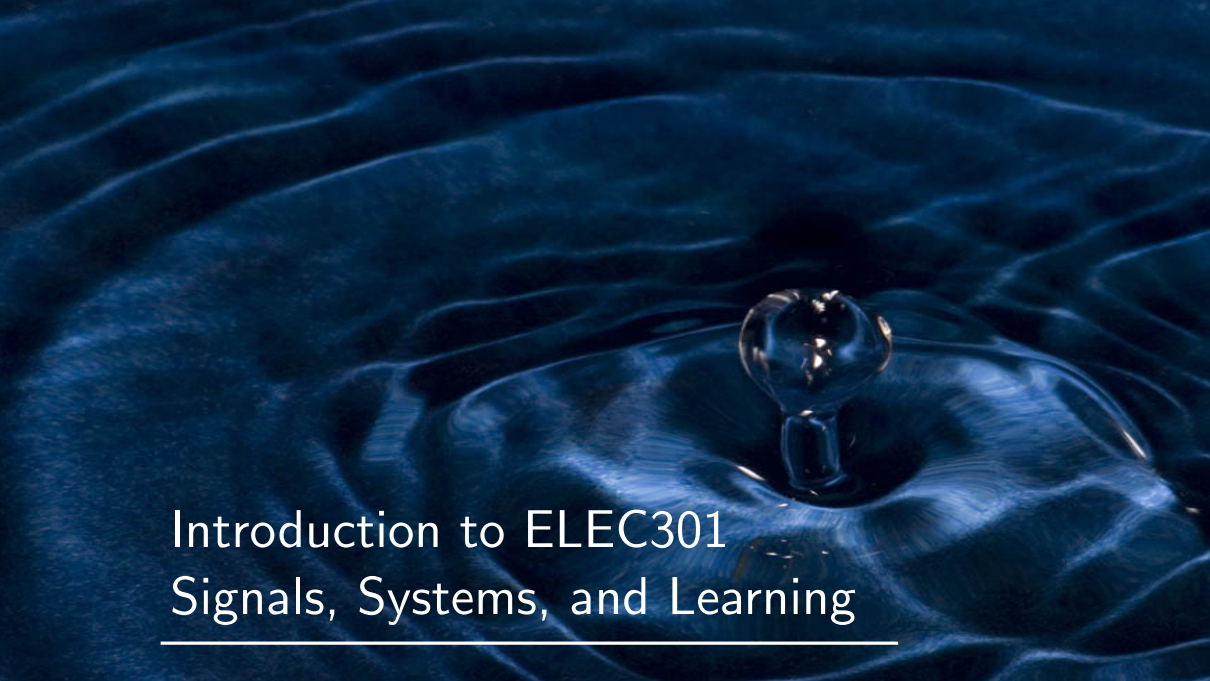
Written by Richard G. Baraniuk.

Digitized by JP Slavinsky, Daniel Williamson, Jared Adler, Abhijit Navlekar, Rachel Green, Caroline Huang, Joy Dai, Isabella Gonzalez, & Matthew Moravec.



Types of Signals

Set A



Introduction to ELEC301
Signals, Systems, and Learning

Welcome to Elec301 – Signals, Systems, and Learning

- This is an introductory course on **signal processing**, which studies **signals**, **systems**, and **learning algorithms**

DEFINITION

Signal (n): A detectable physical quantity . . . by which messages or information can be transmitted (Merriam-Webster)

- Signals carry **information**
- Examples:
  - Speech signals transmit language via acoustic waves
  - Radar signals transmit the position and velocity of targets via electromagnetic waves
  - Electrophysiology signals transmit information about processes inside the body
  - Financial signals transmit information about events in the economy

Welcome to Elec301 – Signals, Systems, and Learning

- **Systems** manipulate the information carried by signals

DEFINITION

Signal processing involves the theory and application of

- filtering, coding, transmitting, estimating, detecting, analyzing, recognizing, synthesizing, recording, and reproducing signals by digital or analog devices or techniques
- where signal includes audio, video, speech, image, communication, geophysical, sonar, radar, medical, musical, and other signals

(IEEE Signal Processing Society Constitutional Amendment, 1994)

Welcome to Elec301 – Signals, Systems, and Learning

- **Systems** manipulate the information carried by signals
- Broadly speaking, there are two ways to **design systems** (and we will study both)
- Classical approach, using **physics**
  - Pro: Small number of parameters (if any)
  - Con: Performance limited by the analytical powers of physicists
- Modern approach, by **learning** from (training) **data**
  - Con: Large number of parameters
  - Pro: Performance limited only by the amount of training data

Welcome to Elec301 – Signals, Systems, and Learning

- **Learning algorithms** optimize how a system manipulates signals

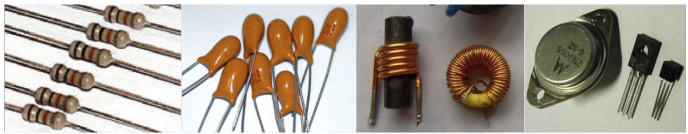
DEFINITION

"A computer program is said to **learn from experience** E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" (Tom Mitchell, CMU)

- The field of **machine learning** is often associated with "**artificial intelligence**" (AI) and "**data science**"

Signal Processing

- Signal processing has traditionally been a part of electrical and computer engineering
- But now expands into applied mathematics, statistics, computer science, and a host of application disciplines
- Initially **analog** signals and systems implemented using resistors, capacitors, inductors, and transistors (Elec241/242)



- Since the 1940s increasingly **digital** signals and systems implemented using computers and computer code (Matlab, Python, C, ...)
  - Advantages of digital include stability and programmability
  - As computers have shrunk, digital signal processing has become ubiquitous
  - Digital processing can be tuned using training data (machine learning)

Signal Processing Applications

Rice ELEC301

- Goals: Develop intuition into and learn how to reason analytically about signal processing and machine learning problems
- Video lectures (first half of course), primary sources, supplemental materials, practice exercises, homework, coding exercises, group competition, final exam
- Coding in Matlab or Python
- **Important:** This is a mathematical treatment of signals, systems, and learning (no pain, no gain!)

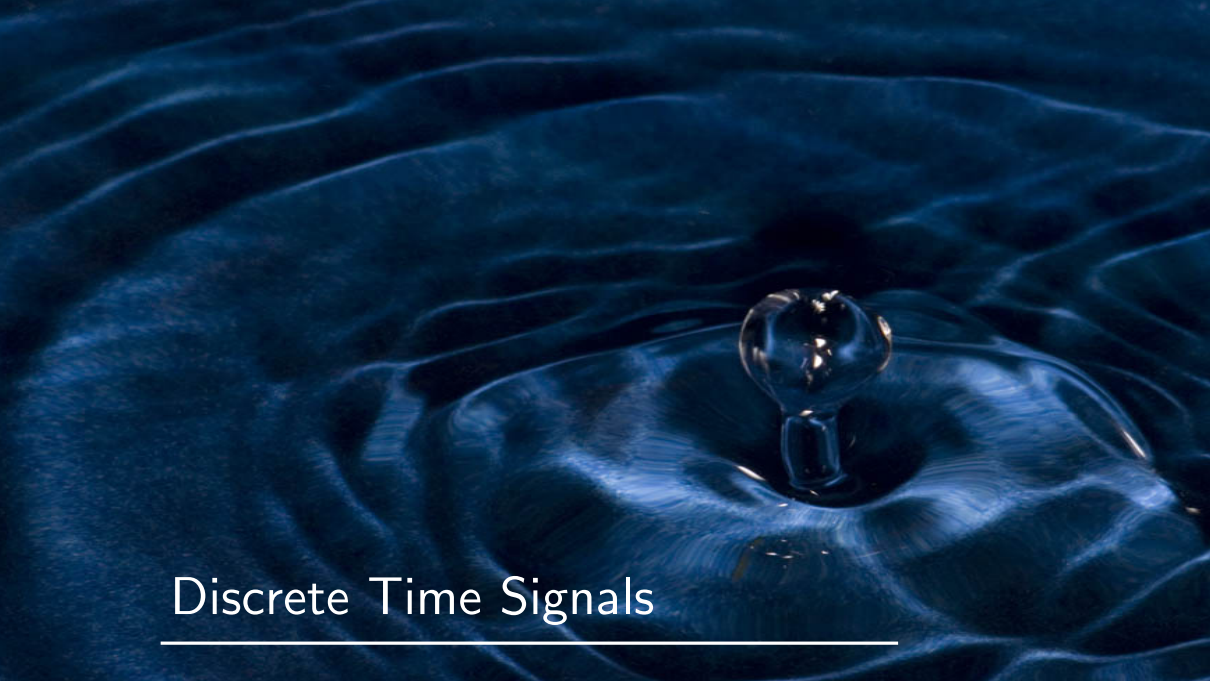
Before We Start

- Please make sure you have a solid understanding of
  - Complex numbers and arithmetic
  - Linear algebra (vectors, matrices, dot products, eigenvectors and eigenvalues, singular value decomposition (SVD), bases ...)
  - Sequences and series (finite and infinite)
  - Calculus of a single variable (derivatives and integrals)
  - Matlab or Python
- To test your readiness or refresh your knowledge, visit the "Pre-class Mathematics Refresher"

Administrata

- Course website: `http://dsp.rice.edu/courses/elec301`
- Course syllabus: `http://dsp.rice.edu/courses/elec301/syllabus`
- Piazza: `https://piazza.com/rice/fall2016/elec301/home`

Welcome to Elec301 – Signals, Systems, and Learning



Discrete Time Signals

Signals

DEFINITION

Signal (n): A detectable physical quantity . . . by which messages or information can be transmitted (Merriam-Webster)

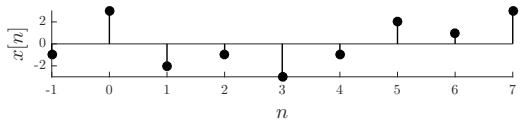
- Signals carry **information**
- Examples:
  - Speech signals transmit language via acoustic waves
  - Radar signals transmit the position and velocity of targets via electromagnetic waves
  - Electrophysiology signals transmit information about processes inside the body
  - Financial signals transmit information about events in the economy
- **Signal processing systems** manipulate the information carried by signals
- This is a course about signals and systems

Signals are Functions

DEFINITION

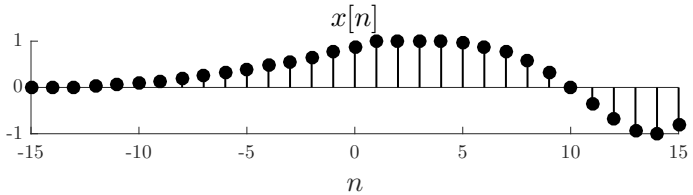
A **signal** is a function that maps an independent variable to a dependent variable.

- Signal $x[n]$: each value of n produces the value $x[n]$
- In this course, we will focus on **discrete-time** signals:
  - Independent variable is an **integer**: $n \in \mathbb{Z}$ (will refer to as time)
  - Dependent variable is a real or complex number: $x[n] \in \mathbb{R}$ or $\mathbb{C}$

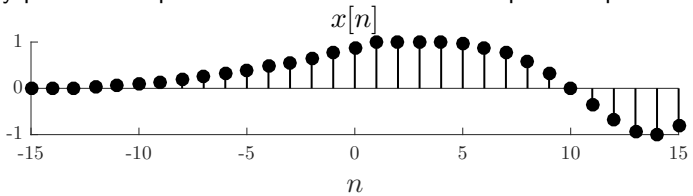


Plotting Real Signals

- When $x[n] \in \mathbb{R}$ (ex: temperature in a room at noon on Monday), we use one signal plot

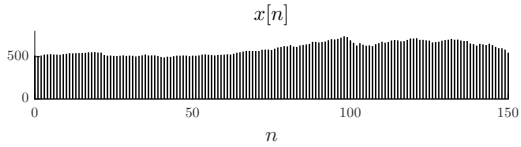


- We will typically place the dependent variable label to the top of the plot.

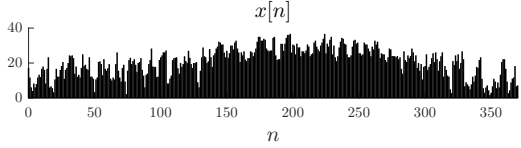


A Menagerie of Signals

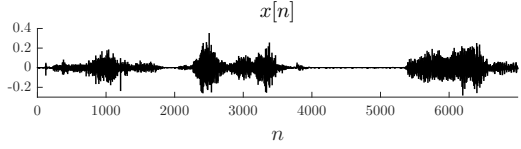
- Google Share daily share price for 5 months



- Temperature at Houston Intercontinental Airport in 2013 (Celcius)

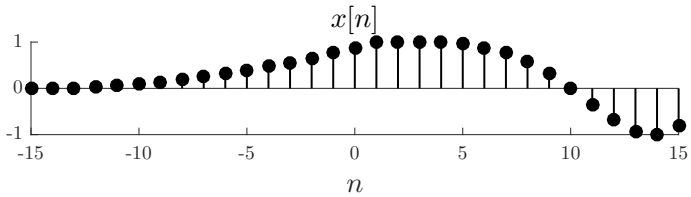


- Excerpt from Shakespeare's *Hamlet*

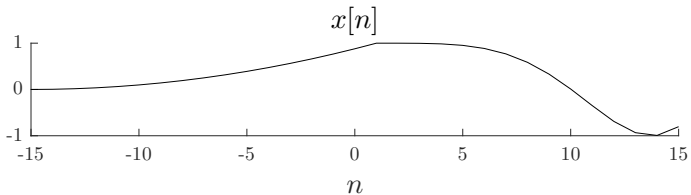


Plotting Signals Correctly

- In a discrete-time signal $x[n]$, the independent variable n is discrete (integer)
- To plot a discrete-time signal in a program like Matlab, you should use the `stem` or similar command and not the `plot` command
- Correct:



- Incorrect:

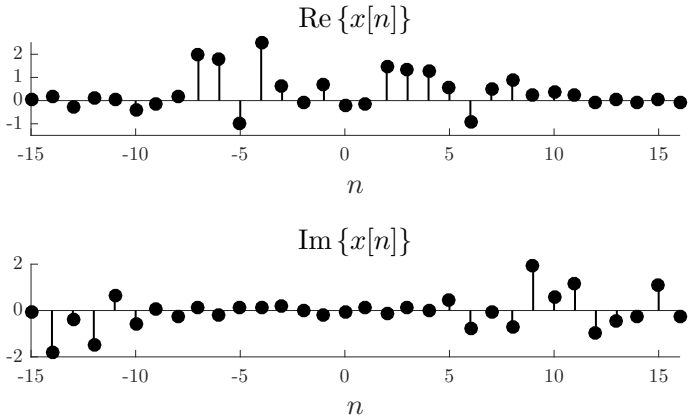


Plotting Complex Signals

- Recall that a complex number $a \in \mathbb{C}$ can be equivalently represented two ways:
  - Polar form: $a = |a|\, e^{j\angle a}$
  - Rectangular form: $a = \mathrm{Re}\{a\} + j\,\mathrm{Im}\{a\}$
- Here $j = \sqrt{-1}$ (engineering notation; mathematicians use $i = \sqrt{-1}$)
- When $x[n] \in \mathbb{C}$ (ex: magnitude and phase of an electromagnetic wave), we use two signal plots
  - Rectangular form: $x[n] = \mathrm{Re}\{x[n]\} + j\,\mathrm{Im}\{x[n]\}$
  - Polar form: $x[n] = |x[n]|\, e^{j\angle x[n]}$

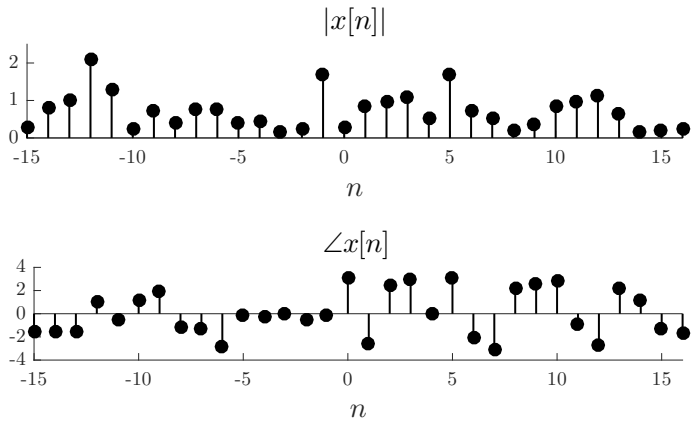
Plotting Complex Signals (Rectangular Form)

- Rectangular form: $x[n] = \mathrm{Re}\{x[n]\} + j\,\mathrm{Im}\{x[n]\} \in \mathbb{C}$



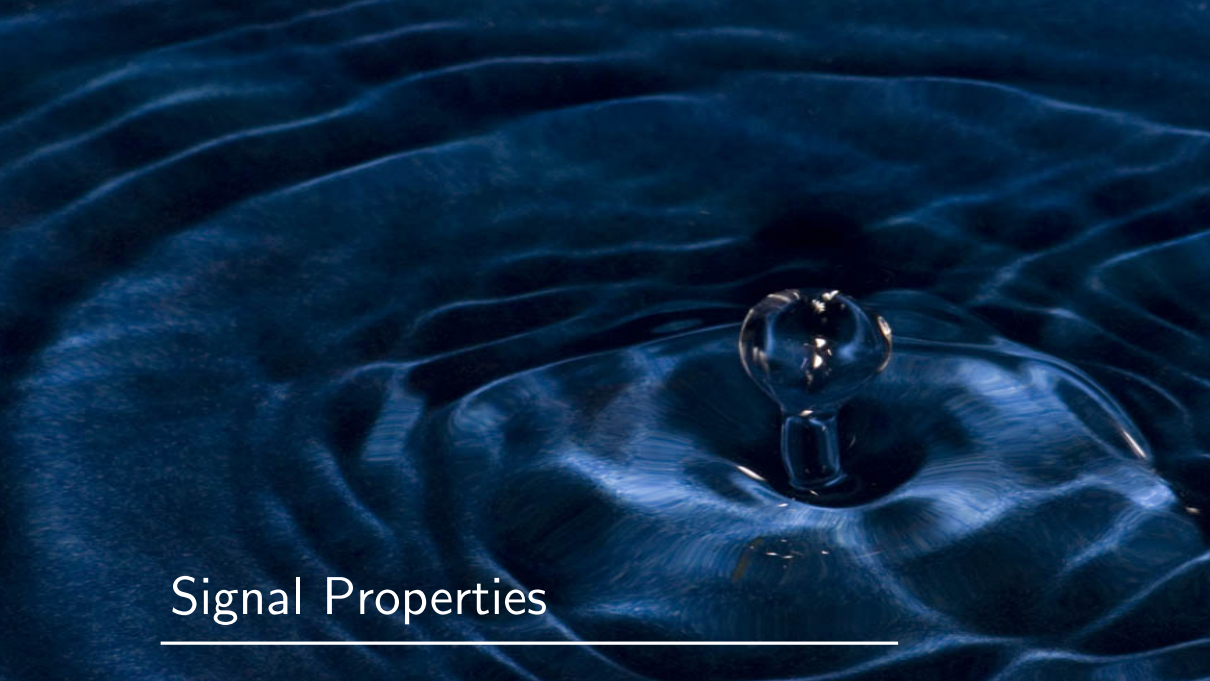
Plotting Complex Signals (Polar Form)

- Polar form: $x[n] = |x[n]|\, e^{j\angle(x[n])} \in \mathbb{C}$



Summary

- Discrete-time signals
  - Independent variable is an integer: $n \in \mathbb{Z}$ (will refer to as time)
  - Dependent variable is a real or complex number: $x[n] \in \mathbb{R}$ or $\mathbb{C}$
- Plot signals correctly!



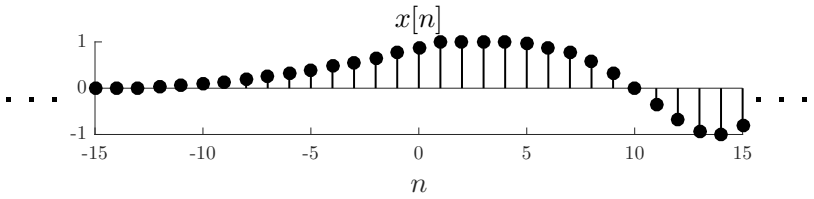
Signal Properties

Signal Properties

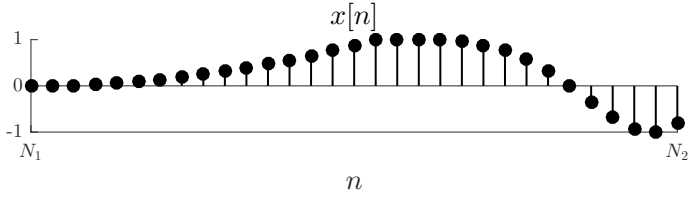
- Infinite/finite-length signals
- Periodic signals
- Causal signals
- Even/odd signals
- Digital signals

Finite/Infinite-Length Signals

- An **infinite-length** discrete-time signal $x[n]$ is defined for all $n \in \mathbb{Z}$, i.e., $-\infty < n < \infty$



- A **finite-length** discrete-time signal $x[n]$ is defined only for a finite range of $N_1 \leq n \leq N_2$



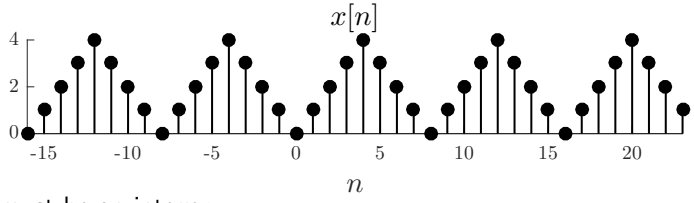
- Important: a finite-length signal is undefined for $n < N_1$ and $n > N_2$

Periodic Signals

DEFINITION

A discrete-time signal is **periodic** if it repeats with period $N \in \mathbb{Z}$:

$$x[n + mN] = x[n] \quad \forall\, m \in \mathbb{Z}$$



Notes:

- The period N must be an integer
- A periodic signal is infinite in length

DEFINITION

A discrete-time signal is **aperiodic** if it is not periodic

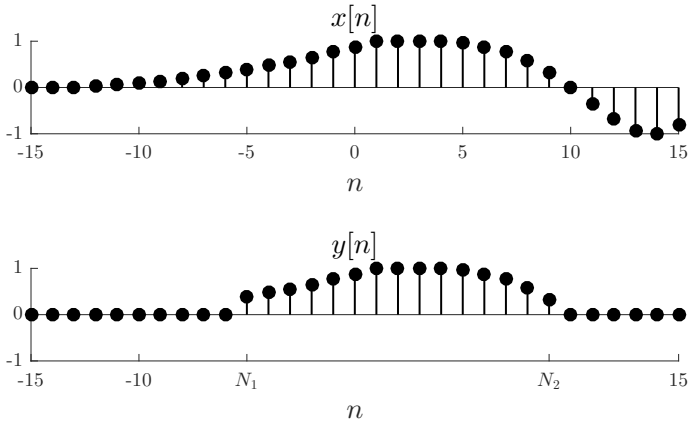
Converting between Finite and Infinite-Length Signals

- Convert an infinite-length signal into a finite-length signal by **windowing**
- Convert a finite-length signal into an infinite-length signal by either
  - (infinite) **zero padding**, or
  - **periodization**

Windowing

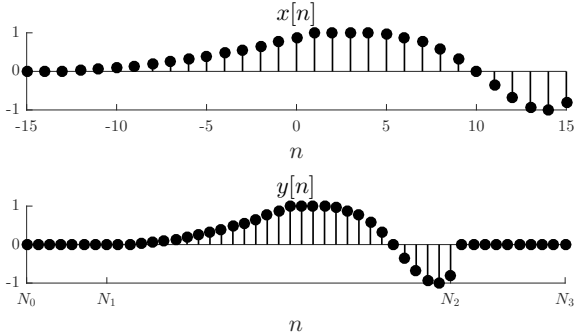
- Converts a longer signal into a shorter one

$$y[n] = \begin{cases} x[n] & N_1 \leq n \leq N_2 \\ 0 & \text{otherwise} \end{cases}$$



Zero Padding

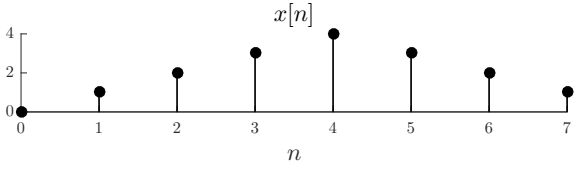
- Converts a shorter signal into a longer one
- Say $x[n]$ is defined for $N_1 \leq n \leq N_2$
- Given $N_0 \leq N_1 \leq N_2 \leq N_3$ $\quad y[n] = \begin{cases} 0 & N_0 \leq n < N_1 \\ x[n] & N_1 \leq n \leq N_2 \\ 0 & N_2 < n \leq N_3 \end{cases}$



Periodization

- Converts a finite-length signal into an infinite-length, periodic signal
- Given finite-length $x[n]$, replicate $x[n]$ periodically with period N

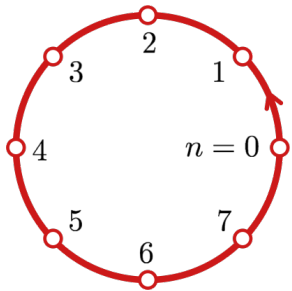
$$\begin{aligned} y[n] &= \sum_{m=-\infty}^{\infty} x[n-mN], \quad n \in \mathbb{Z} \\ &= \cdots + x[n+2N] + x[n+N] + x[n] + x[n-N] + x[n-2N] + \cdots \end{aligned}$$



Useful Aside – Modular Arithmetic

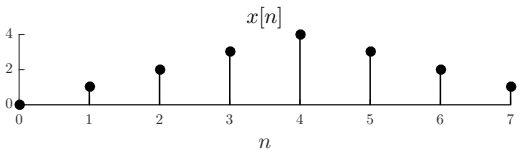
- Modular arithmetic with **modulus** N (mod-N) takes place on a **wheel** with N time locations
  - Ex: $(12)_8$ ("twelve mod eight")

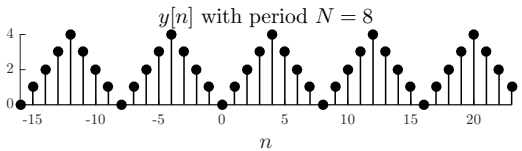
- Modulo arithmetic is inherently **periodic**
  - Ex: $\ldots\ (-12)_8 = (-4)_8 = (4)_8 = (12)_8 = (20)_8\ \ldots$



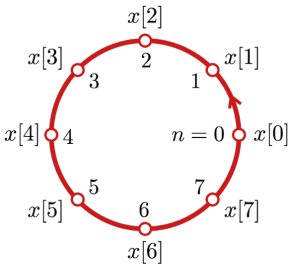
Periodization via Modular Arithmetic

- Consider a length-N signal $x[n]$ defined for $0 \leq n \leq N-1$
- A convenient way to express periodization with period N is $\quad y[n] = x[(n)_N], \quad n \in \mathbb{Z}$

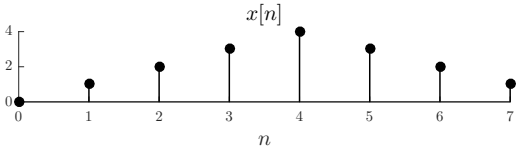


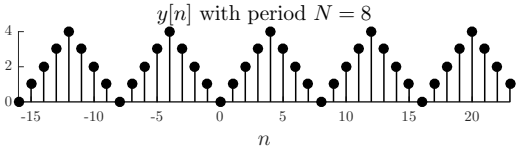


- Important interpretation
  - Infinite-length signals live on the (infinite) number **line**
  - Periodic signals live on a **circle** – a wheel with N time locations



Finite-Length and Periodic Signals are Equivalent



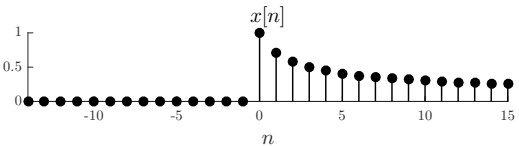


- All of the information in a periodic signal is contained in **one period** (of finite length)
- Any finite-length signal can be periodized
- Conclusion: We can and will think of finite-length signals and periodic signals interchangeably
- We can choose the most convenient viewpoint for solving any given problem
  - Application: Shifting finite length signals

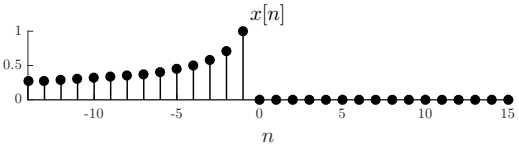
Causal Signals

DEFINITION

A signal $x[n]$ is **causal** if $x[n] = 0$ for all $n < 0$.



- A signal $x[n]$ is **anti-causal** if $x[n] = 0$ for all $n \geq 0$

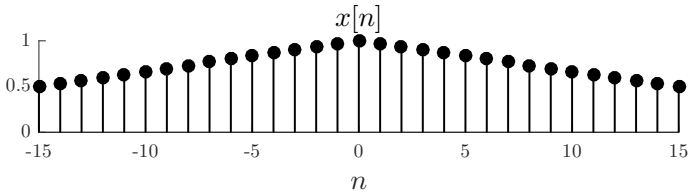


- A signal $x[n]$ is **acausal** if it is not causal

Even Signals

DEFINITION

A real signal $x[n]$ is **even** if $x[-n] = x[n]$

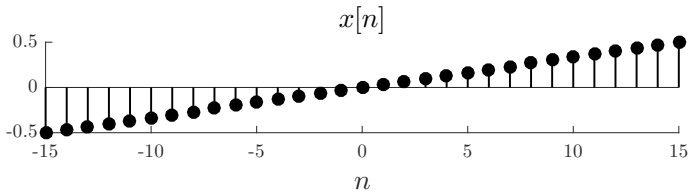


- Even signals are symmetrical around the point $n = 0$

Odd Signals

DEFINITION

A real signal $x[n]$ is **odd** if $x[-n] = -x[n]$



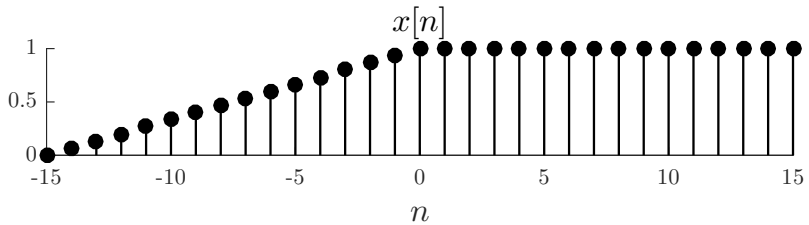
- Odd signals are anti-symmetrical around the point $n = 0$

Even+Odd Signal Decomposition

- **Useful fact:** Every signal $x[n]$ can be decomposed into the sum of its even part + its odd part
- Even part: $e[n] = \frac{1}{2}\left(x[n] + x[-n]\right)$ (easy to verify that $e[n]$ is even)
- Odd part: $o[n] = \frac{1}{2}\left(x[n] - x[-n]\right)$ (easy to verify that $o[n]$ is odd)
- **Decomposition** $x[n] = e[n] + o[n]$
- Verify the decomposition:

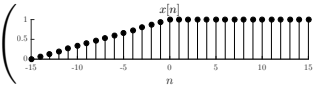
$$\begin{aligned} e[n] + o[n] &= \frac{1}{2}(x[n] + x[-n]) + \frac{1}{2}(x[n] - x[-n]) \\ &= \frac{1}{2}(x[n] + x[-n] + x[n] - x[-n]) \\ &= \frac{1}{2}(2\,x[n]) = x[n] \quad \checkmark \end{aligned}$$

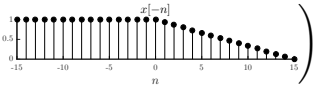
Even+Odd Signal Decomposition in Pictures

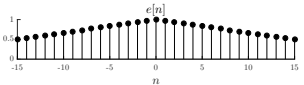


Even+Odd Signal Decomposition in Pictures

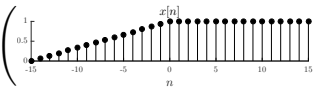
$$\frac{1}{2}\left(\right. \quad + \quad \left. \right) = \quad + \quad$$

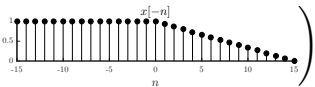


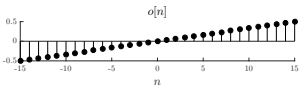


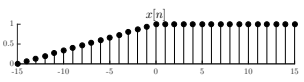


$$\frac{1}{2}\left(\right. \quad - \quad \left. \right) = \quad = \quad$$



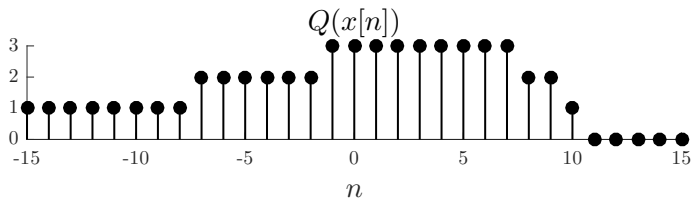






Digital Signals

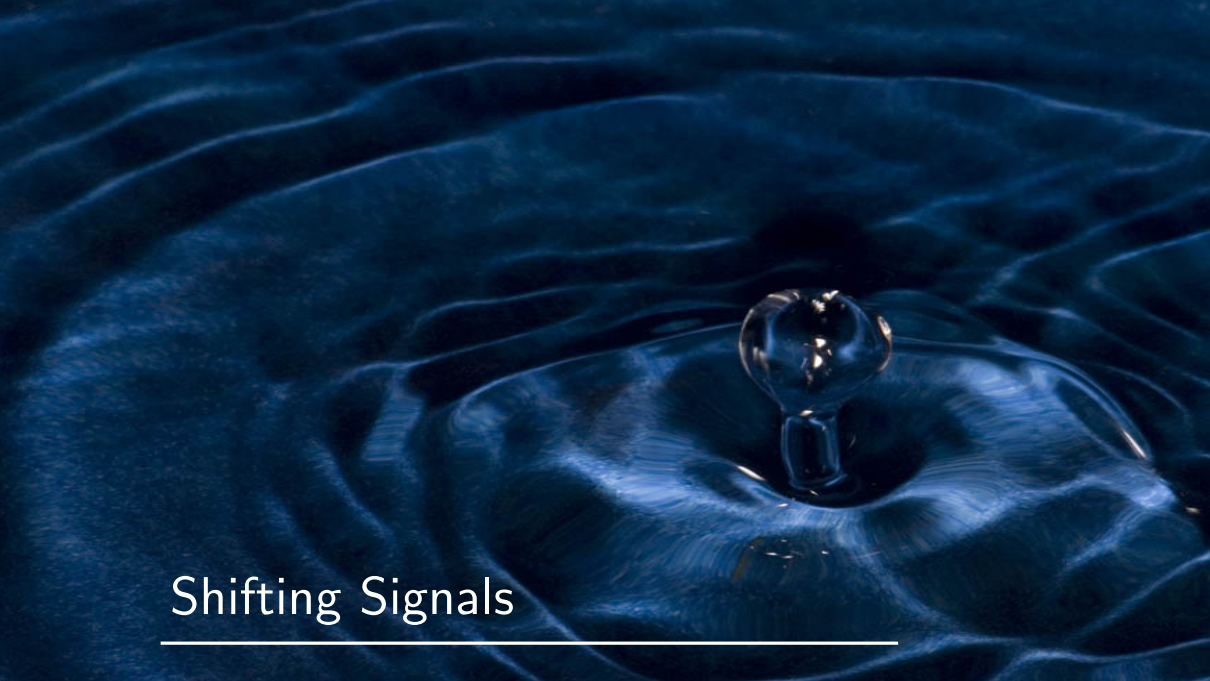
- **Digital signals** are a special sub-class of discrete-time signals
  - Independent variable is still an integer: $n \in \mathbb{Z}$
  - Dependent variable is from a **finite set of integers**: $x[n] \in \{0, 1, \ldots, D-1\}$
  - Typically, choose $D = 2^q$ and represent each possible level of $x[n]$ as a digital code with q bits
  - Ex: Digital signal with $q = 2$ bits $\Rightarrow D = 2^2 = 4$ levels



  - Ex: Compact discs use $q = 16$ bits $\Rightarrow D = 2^{16} = 65536$ levels

Summary

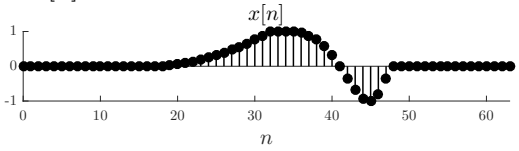
- Signals can be classified many different ways (real/complex, infinite/finite-length, periodic/aperiodic, causal/acausal, even/odd, ...)
- Finite-length signals are equivalent to periodic signals; modulo arithmetic useful



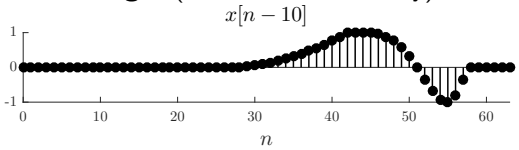
Shifting Signals

Shifting Infinite-Length Signals

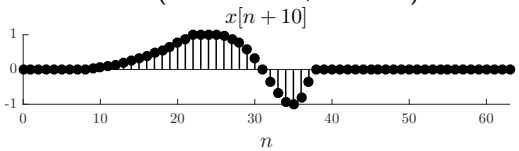
- Given an infinite-length signal $x[n]$, we can **shift** it back and forth in time via $x[n-m]$, $m \in \mathbb{Z}$



- When $m > 0$, $x[n-m]$ shifts to the **right** (forward in time, delay)



- When $m < 0$, $x[n-m]$ shifts to the **left** (back in time, advance)

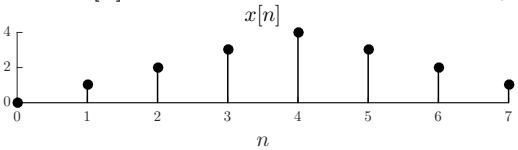


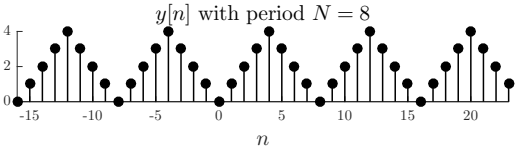
Periodic Signals and Modular Arithmetic

- A convenient way to express a signal $y[n]$ that is periodic with period N is

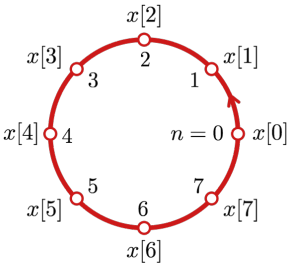
$$y[n] = x[(n)_N], \quad n \in \mathbb{Z}$$

where $x[n]$, defined for $0 \le n \le N-1$, comprises one period



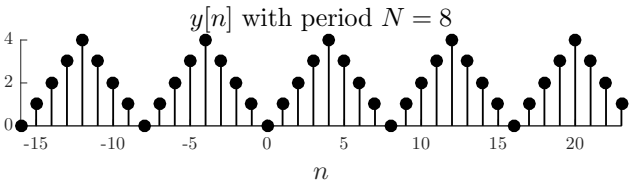


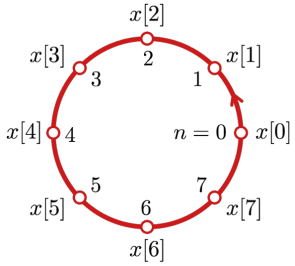
- Important interpretation
  - Infinite-length signals live on the (infinite) number line
  - Periodic signals live on a circle – a wheel with N time places



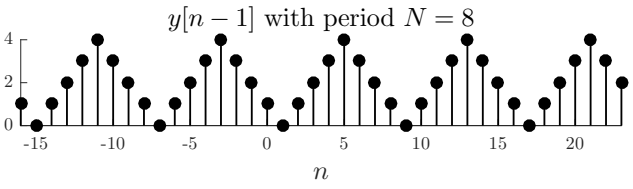
Shifting Periodic Signals

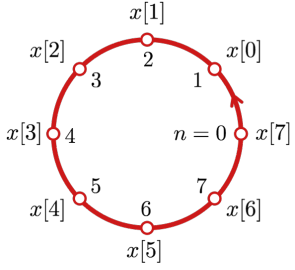
- Periodic signals can also be shifted; consider $y[n] = x[(n)_N]$





- Shift one sample into the future: $y[n-1] = x[(n-1)_N]$





Shifting Finite-Length Signals

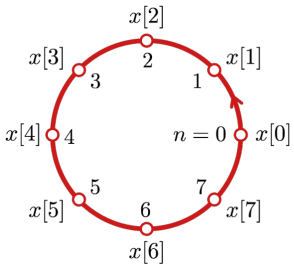
- Consider finite-length signals x and v defined for $0 \leq n \leq N-1$ and suppose "$v[n] = x[n-1]$"

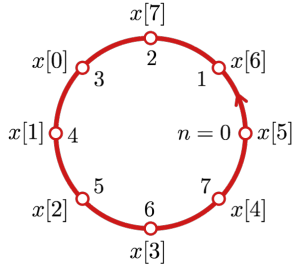
$$\begin{aligned} v[0] &= ?? \\ v[1] &= x[0] \\ v[2] &= x[1] \\ v[3] &= x[2] \\ &\vdots \\ v[N-1] &= x[N-2] \\ ?? &= x[N-1] \end{aligned}$$

- What to put in $v[0]$? What to do with $x[N-1]$? We don't want to invent/lose information
- Elegant solution: Assume x and v are both periodic with period N; then $v[n] = x[(n-1)_N]$
- This is called a **periodic** or **circular shift** (see `circshift` and `mod` in Matlab)

Circular Shift Example

- Elegant formula for circular shift of $x[n]$ by m time steps: $x[(n-m)_N]$
- Ex: x and v defined for $0 \leq n \leq 7$, that is, $N = 8$. Find $v[n] = x[(n-3)_8]$



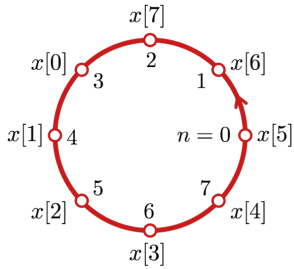


Circular Shift Example

- Elegant formula for circular shift of $x[n]$ by m time steps: $x[(n-m)_N]$

- Ex: x and v defined for $0 \le n \le 7$, that is, $N = 8$. Find $v[n] = x[(n-m)_N]$

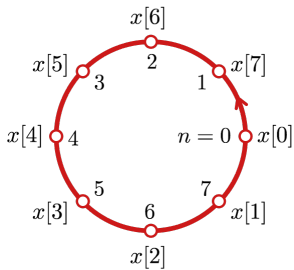
$$
\begin{aligned}
v[0] &= x[5] \\
v[1] &= x[6] \\
v[2] &= x[7] \\
v[3] &= x[0] \\
v[4] &= x[1] \\
v[5] &= x[2] \\
v[6] &= x[3] \\
v[7] &= x[4]
\end{aligned}
$$



Circular Time Reversal

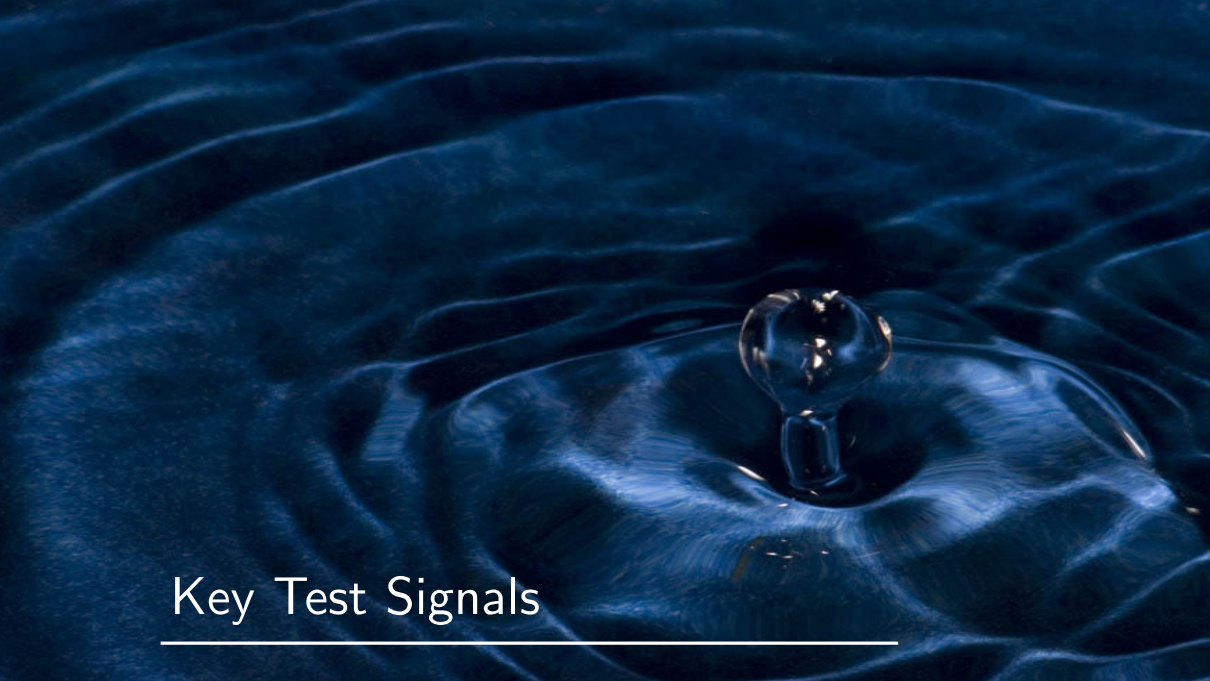
- For infinite length signals, the transformation of reversing the time axis $x[-n]$ is obvious
- Not so obvious for periodic/finite-length signals
- Elegant formula for reversing the time axis of a periodic/finite-length signal: $x[(-n)_N]$
- Ex: x and v defined for $0 \leq n \leq 7$, that is, $N = 8$. Find $v[n] = x[(-n)_N]$

$$
\begin{aligned}
v[0] &= x[0] \\
v[1] &= x[7] \\
v[2] &= x[6] \\
v[3] &= x[5] \\
v[4] &= x[4] \\
v[5] &= x[3] \\
v[6] &= x[2] \\
v[7] &= x[1]
\end{aligned}
$$



Summary

- Shifting a signal moves it forward or backward in time
- Modulo arithmetic provides and easy way to shift periodic signals



Key Test Signals

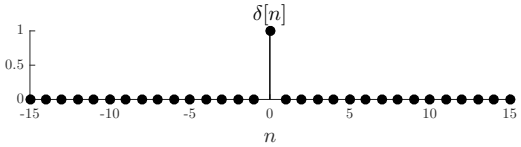
A Toolbox of Test Signals

- Delta function
- Unit step
- Unit pulse
- Real exponential
- Still to come: sinusoids, complex exponentials
- **Note:** We will introduce the test signals as infinite-length signals, but each has a finite-length equivalent

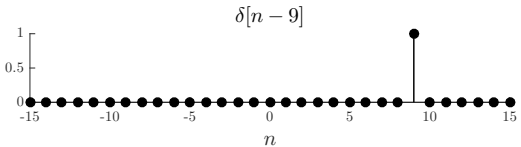
Delta Function

DEFINITION

The **delta function** (aka unit impulse) $\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$



- The shifted delta function $\delta[n-m]$ peaks up at $n = m$; here $m = 9$

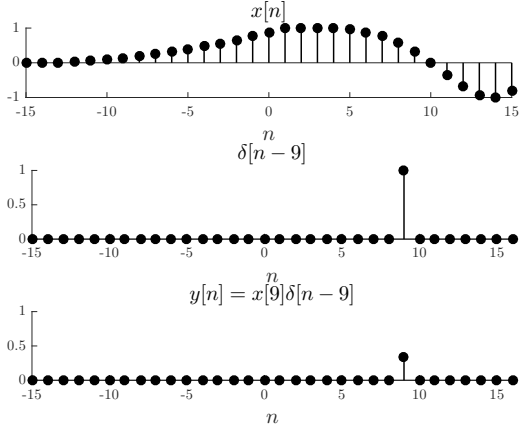


Delta Functions Sample

- Multiplying a signal by a shifted delta function picks out one sample of the signal and sets all other samples to zero

$$y[n] \;=\; x[n]\,\delta[n-m] \;=\; x[m]\,\delta[n-m]$$

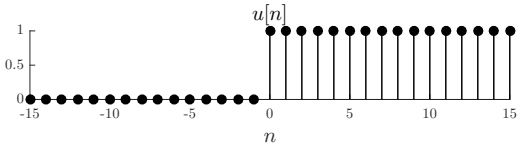
- Important: m is a fixed constant, and so $x[m]$ is a constant (and not a signal)



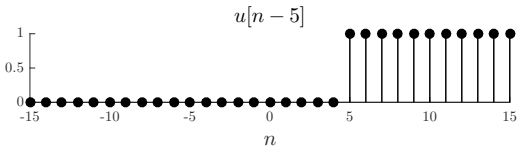
Unit Step

DEFINITION

The **unit step** $u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$



- The shifted unit step $u[n-m]$ jumps from 0 to 1 at $n = m$; here $m = 5$

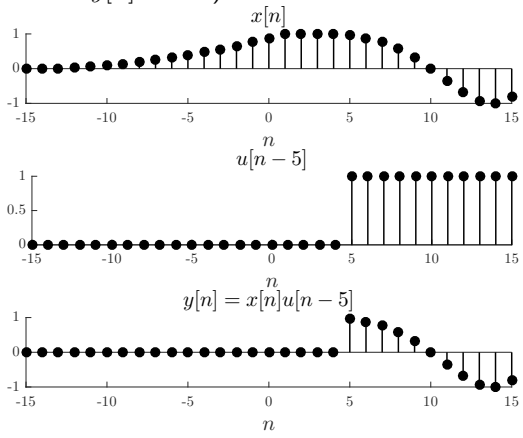


Unit Step Selects Part of a Signal

- Multiplying a signal by a shifted unit step function zeros out its entries for $n < m$

$$y[n] \;=\; x[n]\,u[n-m]$$

(Note: For $m = 0$, this makes $y[n]$ causal)

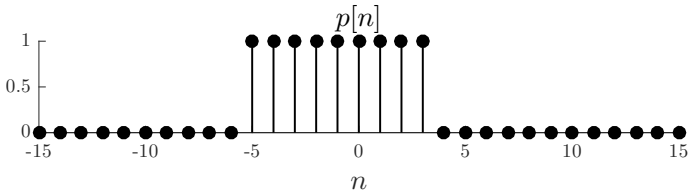


Unit Pulse

DEFINITION

The **unit pulse** (aka boxcar) $$p[n] = \begin{cases} 0 & n < N_1 \\ 1 & N_1 \le n \le N_2 \\ 0 & n > N_2 \end{cases}$$

- Ex: $p[n]$ for $N_1 = -5$ and $N_2 = 3$



- One of many different formulas for the unit pulse

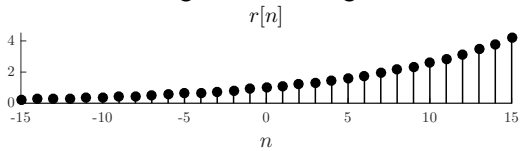
$$p[n] = u[n - N_1] - u[n - (N_2 + 1)]$$

Real Exponential

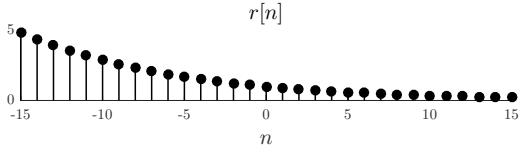
DEFINITION

The **real exponential** $r[n] = a^n,\ a \in \mathbb{R},\ a \geq 0$

- For $a > 1$, $r[n]$ shrinks to the left and grows to the right; here $a = 1.1$



- For $0 < a < 1$, $r[n]$ grows to the left and shrinks to the right; here $a = 0.9$



Summary

- We will use our test signals often, especially the delta function and unit step

Sinusoids

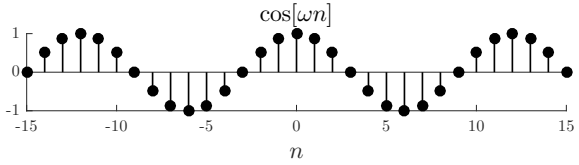
Sinusoids

- Sinusoids appear in myriad disciplines, in particular signal processing
- They are the basis (literally) of Fourier analysis (DFT, DTFT)
- We will introduce
  - Real-valued sinusoids
  - (Complex) sinusoid
  - Complex exponential

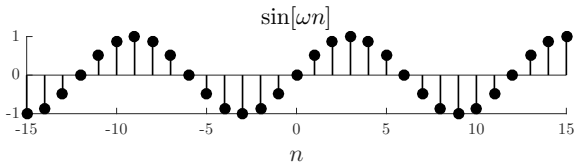
Sinusoids

- There are two natural real-valued sinusoids: $\cos(\omega n + \phi)$ and $\sin(\omega n + \phi)$
- **Frequency:** ω (units: radians/sample)
- **Phase:** ϕ (units: radians)

- $\cos(\omega n)$ (even)

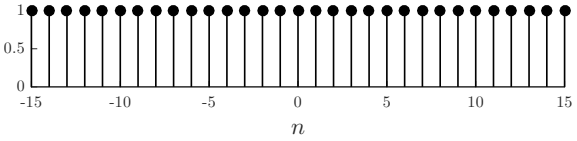


- $\sin(\omega n)$ (odd)

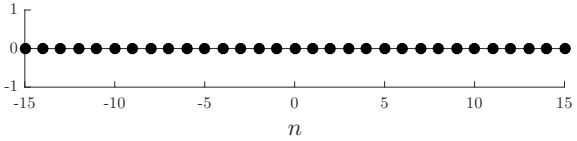


Sinusoid Examples

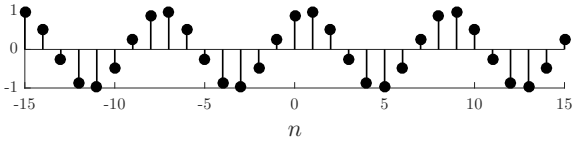
- $\cos(0n)$



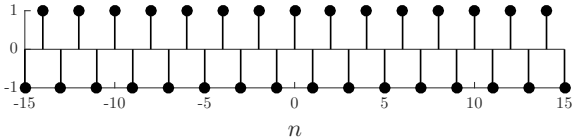
- $\sin(0n)$



- $\sin(\frac{\pi}{4}n + \frac{2\pi}{6})$



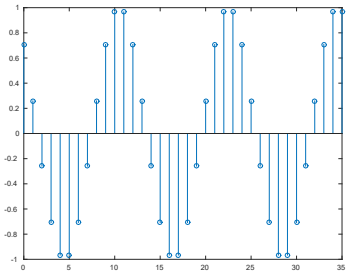
- $\cos(\pi n)$



Get Comfortable with Sinusoids!

- It's easy to play around in Matlab to get comfortable with the properties of sinusoids

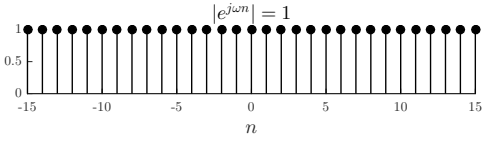
```
N=36;
n=0:N-1;
omega=pi/6;
phi=pi/4;
x=cos(omega*n+phi);
stem(n,x)
```

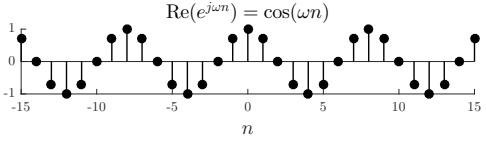


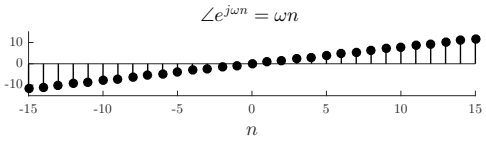
Complex Sinusoid

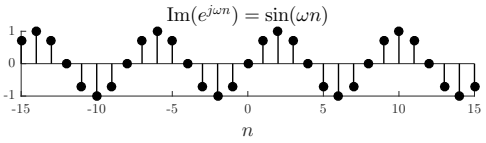
- The complex-valued sinusoid combines both the $\cos$ and $\sin$ terms (via Euler's identity)

$$e^{j(\omega n+\phi)} = \cos(\omega n + \phi) + j\sin(\omega n + \phi)$$



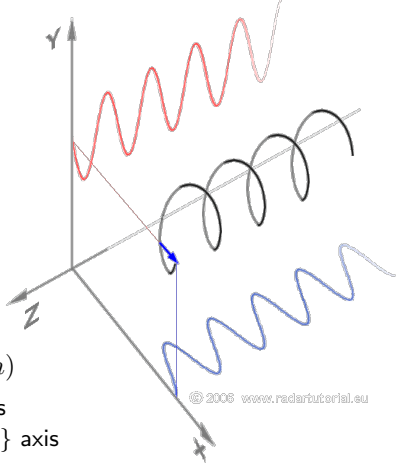






A Complex Sinusoid is a Helix

$$e^{j(\omega n+\phi)} = \cos(\omega n + \phi) + j\sin(\omega n + \phi)$$



- A complex sinusoid is a **helix** in 3D space $(\mathrm{Re}\{\}, \mathrm{Im}\{\}, n)$
  - Real part ($\cos$ term) is the projection onto the $\mathrm{Re}\{\}$ axis
  - Imaginary part ($\sin$ term) is the projection onto the $\mathrm{Im}\{\}$ axis
- Frequency ω determines rotation speed and direction of helix
  - $\omega > 0 \Rightarrow$ anticlockwise rotation
  - $\omega < 0 \Rightarrow$ clockwise rotation

Complex Sinusoid is a Helix (Animation)

- **Complex sinusoid animation**

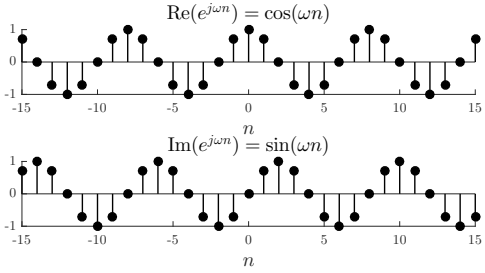
Negative Frequency

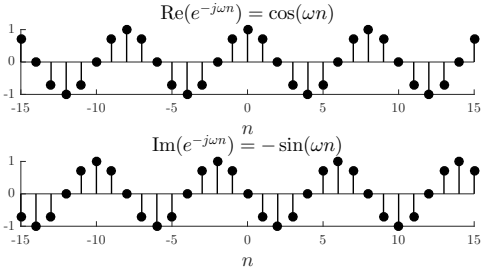
- Negative frequency is nothing to be afraid of! Consider a sinusoid with a negative frequency $-\omega$

$$e^{j(-\omega)n} \;=\; e^{-j\omega n} \;=\; \cos(-\omega n) + j\sin(-\omega n) \;=\; \cos(\omega n) - j\sin(\omega n)$$

- Also note: $e^{j(-\omega)n} \;=\; e^{-j\omega n} \;=\; \left(e^{j\omega n}\right)^*$

- Bottom line: negating the frequency is equivalent to complex conjugating a complex sinusoid, which flips the sign of the imaginary, $\sin$ term



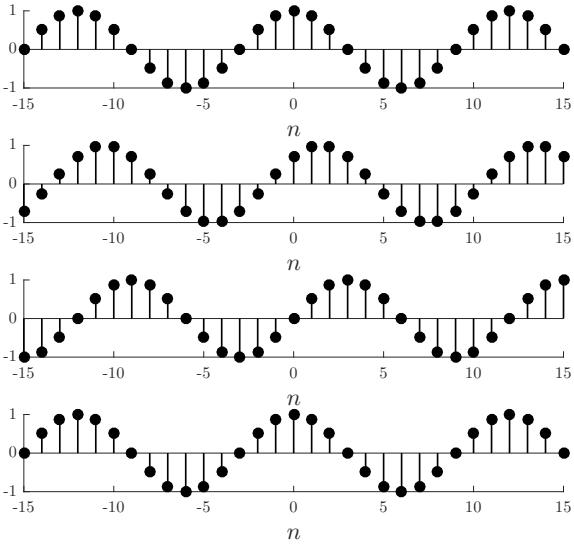


Phase of a Sinusoid

$e^{j(\omega n+\phi)}$

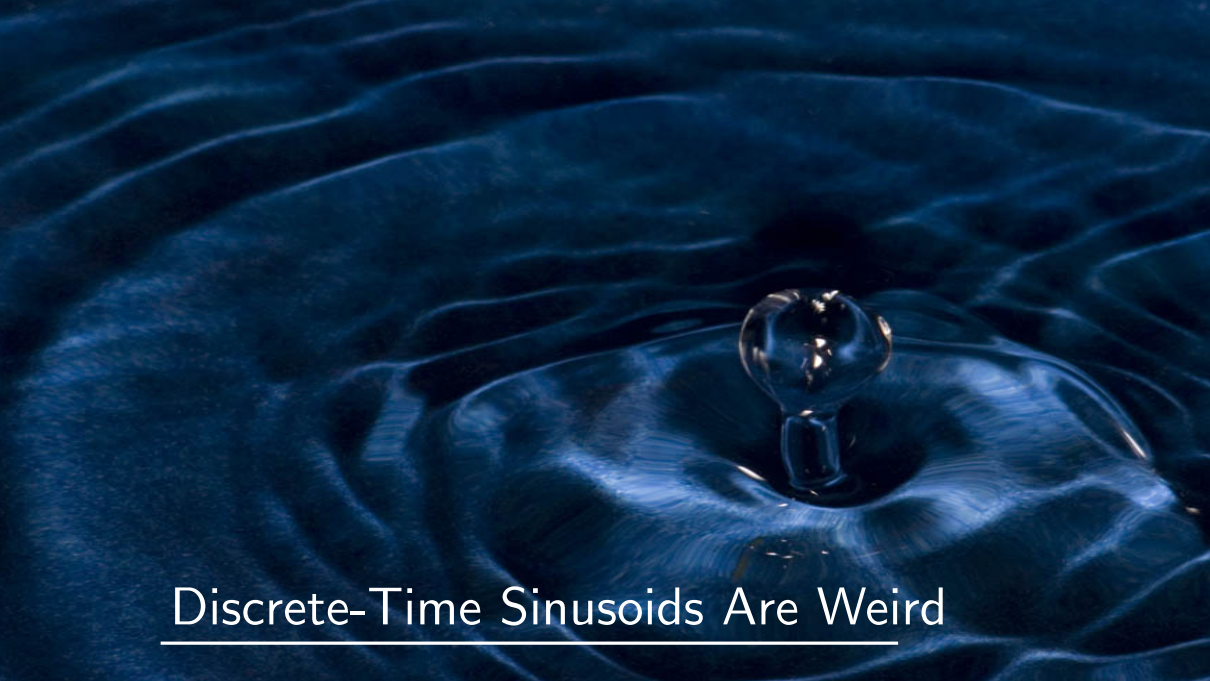
- ϕ is a (frequency independent) shift that is referenced to one period of oscillation

- $\cos\left(\frac{\pi}{6}n - 0\right)$
- $\cos\left(\frac{\pi}{6}n - \frac{\pi}{4}\right)$
- $\cos\left(\frac{\pi}{6}n - \frac{\pi}{2}\right) = \sin\left(\frac{\pi}{6}n\right)$
- $\cos\left(\frac{\pi}{6}n - 2\pi\right) = \cos\left(\frac{\pi}{6}n\right)$



Summary

- Sinusoids play a starring role in both the theory and applications of signals and systems
- A sinusoid has a **frequency** and a **phase**
- A complex sinusoid is a helix in three-dimensional space and naturally induces the sine and cosine
- Negative frequency is nothing to be scared by; it just means that the helix spins backwards



Discrete-Time Sinusoids Are Weird

Discrete-Time Sinusoids are Weird!

- Discrete-time sinusoids $e^{j(\omega n+\phi)}$ have two counterintuitive properties
- Both involve the frequency ω
- Weird property #1: Aliasing
- Weird property #2: Aperiodicity

Weird Property #1: Aliasing of Sinusoids

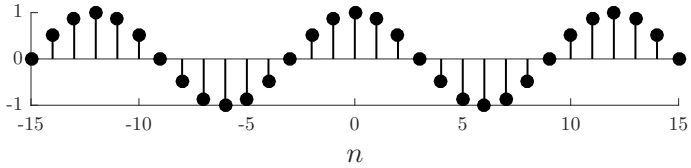
- Consider two sinusoids with two different frequencies
  - $\omega \quad \Rightarrow \quad x_1[n] = e^{j(\omega n + \phi)}$
  - $\omega + 2\pi \quad \Rightarrow \quad x_2[n] = e^{j((\omega+2\pi)n+\phi)}$
- But note that

$$x_2[n] = e^{j((\omega+2\pi)n+\phi)} = e^{j(\omega n+\phi)+j2\pi n} = e^{j(\omega n+\phi)}\, e^{j2\pi n} = e^{j(\omega n+\phi)} = x_1[n]$$

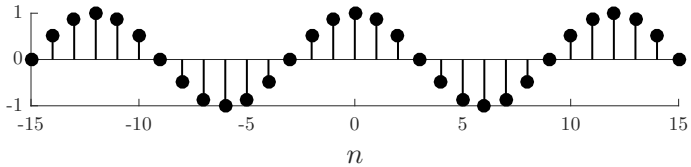
- The signals x_1 and x_2 have different frequencies but are **identical**!
- We say that x_1 and x_2 are aliases; this phenomenon is called **aliasing**
- Note: Any integer multiple of 2π will do; try with $x_3[n] = e^{j((\omega+2\pi m)n+\phi)}$, $m \in \mathbb{Z}$

Aliasing of Sinusoids – Example

- $x_1[n] = \cos\left(\frac{\pi}{6}n\right)$



- $x_2[n] = \cos\left(\frac{13\pi}{6}n\right) = \cos\left((\frac{\pi}{6} + 2\pi)n\right)$



Alias-Free Frequencies

- Since

$$x_3[n] = e^{j(\omega+2\pi m)n+\phi)} = e^{j(\omega n+\phi)} = x_1[n] \quad \forall m \in \mathbb{Z}$$

  the only frequencies that lead to unique (distinct) sinusoids lie in an interval of length 2π

- Convenient to interpret the frequency ω as an **angle**
  (then aliasing is handled automatically; more on this later)

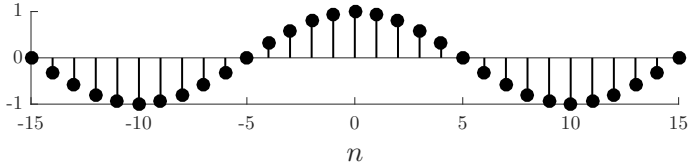
- Two intervals are typically used in the signal processing literature (and in this course)
  - $0 \leq \omega < 2\pi$
  - $-\pi < \omega \leq \pi$

Low and High Frequencies

$$e^{j(\omega n+\phi)}$$

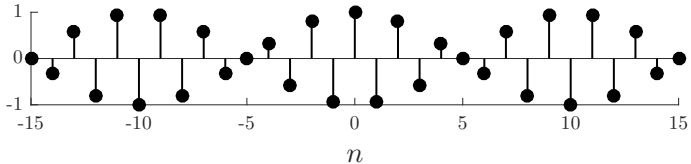
- **Low frequencies:** ω close to 0 or 2π rad

  Ex: $\cos\left(\frac{\pi}{10}n\right)$



- **High frequencies:** ω close to π or $-\pi$ rad

  Ex: $\cos\left(\frac{9\pi}{10}n\right)$



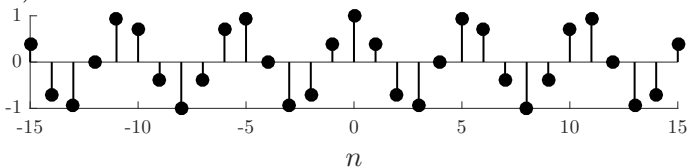
Weird Property #2: Periodicity of Sinusoids

- Consider $x_1[n] = e^{j(\omega n + \phi)}$ with frequency $\omega = \frac{2\pi k}{N}, \quad k, N \in \mathbb{Z}$ (harmonic frequency)

- It is easy to show that x_1 is periodic with period N, since

$$x_1[n+N] = e^{j(\omega(n+N)+\phi)} = e^{j(\omega n + \omega N + \phi)} = e^{j(\omega n+\phi)}\, e^{j(\omega N)} = e^{j(\omega n+\phi)}\, e^{j(\frac{2\pi k}{N} N)} = x_1[n] \checkmark$$

- Ex: $x_1[n] = \cos(\frac{2\pi 3}{16} n), \quad N = 16$



- Note: x_1 is periodic with the (smaller) period of $\frac{N}{k}$ when $\frac{N}{k}$ is an integer

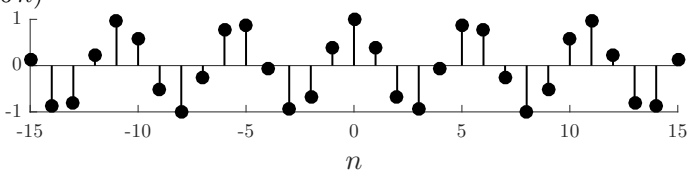
Aperiodicity of Sinusoids

- Consider $x_2[n] = e^{j(\omega n+\phi)}$ with frequency $\omega \neq \frac{2\pi k}{N}, \quad k, N \in \mathbb{Z}$ (not harmonic frequency)

- Is x_2 periodic?

$$x_2[n+N] = e^{j(\omega(n+N)+\phi)} = e^{j(\omega n+\omega N+\phi)} = e^{j(\omega n+\phi)}\, e^{j(\omega N)} \neq x_1[n] \quad \text{NO!}$$

- Ex: $x_2[n] = \cos(1.16\, n)$



- If its frequency ω is not harmonic, then a sinusoid oscillates but is not periodic!

Harmonic Sinusoids

$$e^{j(\omega n+\phi)}$$

- Semi-amazing fact: The **only** periodic discrete-time sinusoids are those with **harmonic frequencies**

$$\omega = \frac{2\pi k}{N}, \quad k, N \in \mathbb{Z}$$

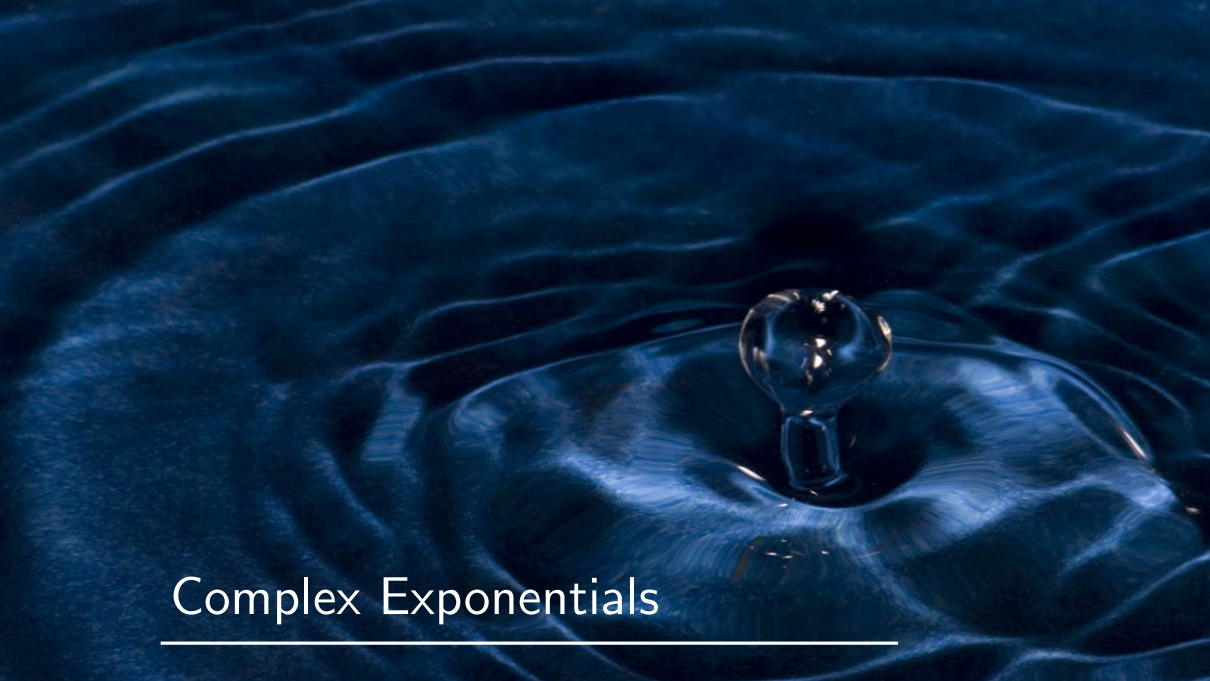
- Which means that
  - **Most** discrete-time sinusoids are **not** periodic!
  - The harmonic sinusoids are somehow magical (they play a starring role later in the DFT)

Harmonic Sinusoids (Matlab)

- **Click here** to view a MATLAB demo that visualizes harmonic sinusoids.

Summary

- Discrete-time sinusoids $e^{j(\omega n+\phi)}$ have two counterintuitive properties
- Both involve the frequency ω
- Weird property #1: Aliasing
- Weird property #2: Aperidiocity
- The only sinusoids that are periodic: Harmonic sinusoids $e^{j(\frac{2\pi k}{N}n+\phi)}$, $n, k, N \in \mathbb{Z}$



Complex Exponentials

Complex Exponential

- Complex sinusoid $e^{j(\omega n+\phi)}$ is of the form $e^{\text{Purely Imaginary Numbers}}$
- Generalize to $e^{\text{General Complex Numbers}}$
- Consider the general complex number $z = |z|\, e^{j\omega}$, $z \in \mathbb{C}$
  - $|z|$ = magnitude of z
  - $\omega = \angle(z)$, phase angle of z
  - Can visualize $z \in \mathbb{C}$ as a **point** in the **complex plane**
- Now we have

$$z^n = (|z|e^{j\omega})^n = |z|^n (e^{j\omega})^n = |z|^n e^{j\omega n}$$

  - $|z|^n$ is a **real exponential** (a^n with $a = |z|$)
  - $e^{j\omega n}$ is a **complex sinusoid**

Complex Exponential is a Spiral

$$z^n \;=\; \left(|z|\, e^{j\omega}\right)^n \;=\; |z|^n\, e^{j\omega n}$$

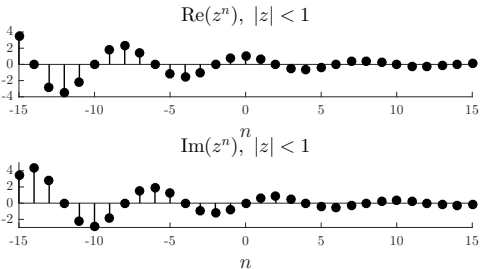
- $|z|^n$ is a **real exponential** envelope $(a^n$ with $a = |z|)$
- $e^{j\omega n}$ is a **complex sinusoid**
- z^n is a helix with expanding radius (spiral)

Complex Exponential is a Spiral

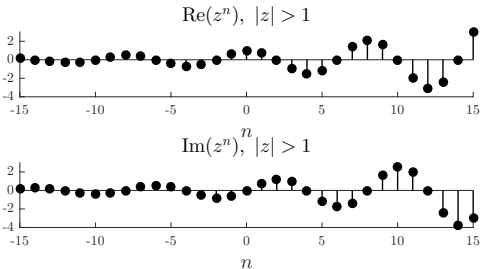
$$z^n \;=\; \left(|z|\, e^{j\omega n}\right)^n \;=\; |z|^n\, e^{j\omega n}$$

- $|z|^n$ is a **real exponential** envelope (a^n with $a = |z|$)
- $e^{j\omega n}$ is a **complex sinusoid**

$|z| < 1$



$|z| > 1$



Complex Exponentials and z Plane (Matlab)

- **Click here** to view a MATLAB demo plotting the signals z^n.

Summary

- Complex sinusoid $e^{j(\omega n+\phi)}$ is of the form $e^{\text{Purely Imaginary Numbers}}$
- Complex exponential: Generalize $e^{j(\omega n+\phi)}$ to $e^{\text{General Complex Numbers}}$
- A complex exponential is the product of a real exponential and a complex sinusoid
- A complex exponential is a spiral in three-dimensional space

Signals are Vectors

Set B

Signals are Vectors

Signals are Vectors

- Signals are mathematical objects
- Here we will develop tools to analyze the **geometry** of sets of signals
- The tools come from **linear algebra**
- By interpreting signals as vectors in a vector space, we will be able to speak about the length of a signal (its "strength," more below), angles between signals (their similarity), and more
- We will also be able to use matrices to better understand how signal processing systems work
- Caveat: This is not a course on linear algebra!

Vector Space

DEFINITION

A linear **vector space** V is a collection of vectors such that if $x, y \in V$ and α is a scalar then

$$\alpha\, x \in V \qquad \text{and} \qquad x + y \in V$$

- In words:
  - A rescaled vector stays in the vector space
  - The sum of two vectors stays in the vector space
- We will be interested in scalars (basically, numbers) α that either live in $\mathbb{R}$ or $\mathbb{C}$
- Classical vector spaces that you know and love
  - $\mathbb{R}^N$, the set of all vectors of length N with real-valued entries
  - $\mathbb{C}^N$, the set of all vectors of length N with complex-valued entries
  - Special case that we will use all the time to draw pictures and build intuition: $\mathbb{R}^2$

The Vector Space $\mathbb{R}^2$ (1)

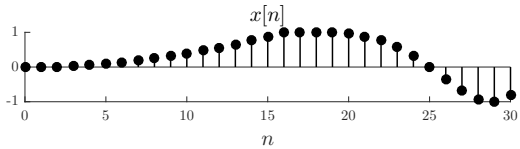
- Vectors in $\mathbb{R}^2$: $x = \begin{bmatrix} x[0] \\ x[1] \end{bmatrix}$, $y = \begin{bmatrix} y[0] \\ y[1] \end{bmatrix}$, $x[0], x[1], y[0], y[1] \in \mathbb{R}$
  - Note: We will enumerate the entries of a vector starting from 0 rather than 1 (this is the convention in signal processing and programming languages like "C", but not in Matlab)
  - Note: We will not use the traditional boldface or underline notation for vectors
- Scalars: $\alpha \in \mathbb{R}$
- Scaling: $\alpha\, x = \alpha \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} = \begin{bmatrix} \alpha\, x[0] \\ \alpha\, x[1] \end{bmatrix}$

The Vector Space $\mathbb{R}^2$ (2)

- Vectors in $\mathbb{R}^2$: $x = \begin{bmatrix} x[0] \\ x[1] \end{bmatrix}$, $y = \begin{bmatrix} y[0] \\ y[1] \end{bmatrix}$, $x[0], x[1], y[0], y[1] \in \mathbb{R}$
- Scalars: $\alpha \in \mathbb{R}$
- Summing: $x + y = \begin{bmatrix} x[0] \\ x[1] \end{bmatrix} + \begin{bmatrix} y[0] \\ y[1] \end{bmatrix} = \begin{bmatrix} x[0] + y[0] \\ x[1] + y[1] \end{bmatrix}$

The Vector Space $\mathbb{R}^N$

- Vectors in $\mathbb{R}^N$: $x = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$, $x[n] \in \mathbb{R}$

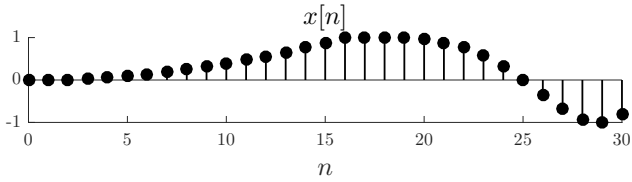


- This is exactly the same as a real-valued discrete time signal; that is, **signals are vectors**
  - Scaling αx amplifies/attenuates a signal by the factor α
  - Summing $x + y$ creates a new signal that mixes x and y

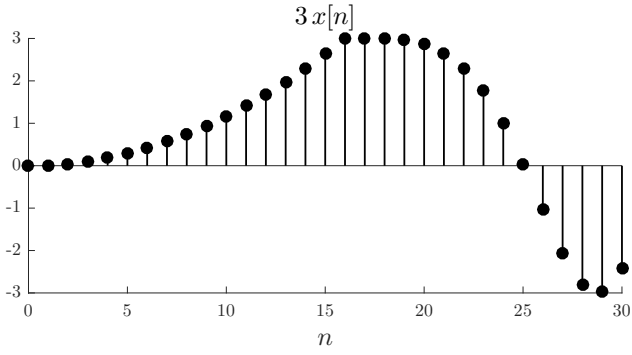
- $\mathbb{R}^N$ is harder to visualize than $\mathbb{R}^2$ and $\mathbb{R}^3$, but the intuition gained from $\mathbb{R}^2$ and $\mathbb{R}^3$ generally holds true with no surprises (at least in this course)

The Vector Space $\mathbb{R}^N$ – Scaling

- Signal $x[n]$

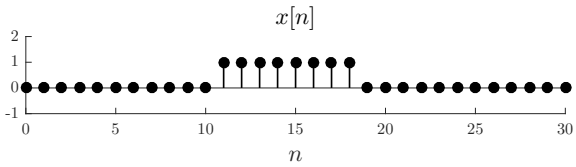


- Scaled signal $3\,x[n]$

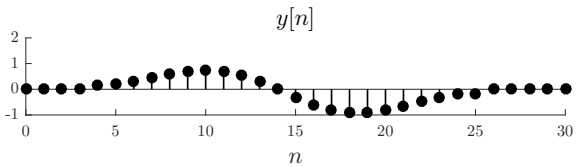


The Vector Space $\mathbb{R}^N$ – Summing

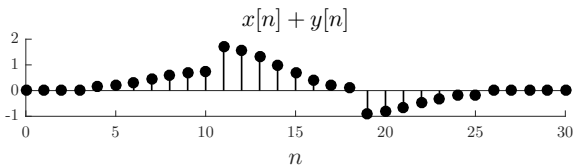
- Signal $x[n]$



- Signal $y[n]$



- Sum $x[n] + y[n]$



The Vector Space $\mathbb{C}^N$ (1)

- $\mathbb{C}^N$ is the same as $\mathbb{R}^N$ with a few minor modifications
- Vectors in $\mathbb{C}^N$: $x = \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}$, $x[n] \in \mathbb{C}$
- Each entry $x[n]$ is a complex number that can be represented as

$$x[n] = \mathrm{Re}\{x[n]\} + j\,\mathrm{Im}\{x[n]\} = |x[n]|\, e^{j\angle x[n]}$$

- Scalars $\alpha \in \mathbb{C}$

The Vector Space $\mathbb{C}^N$ (2)

- Rectangular form

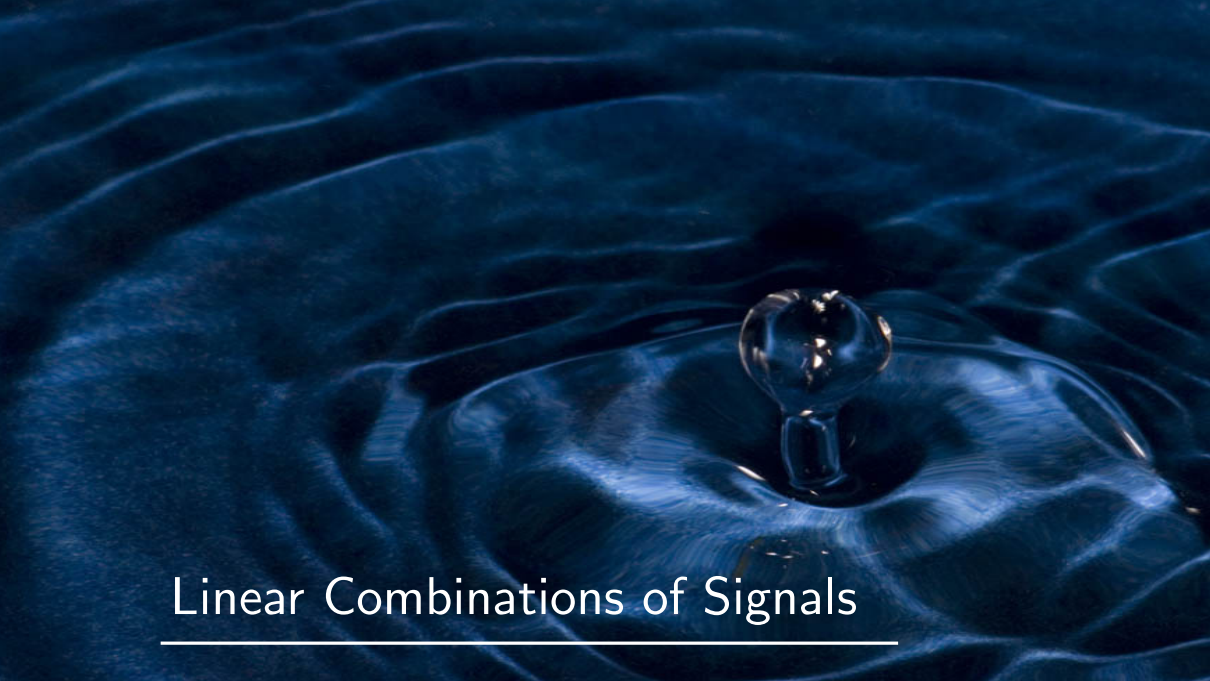
$$x = \begin{bmatrix} \mathrm{Re}\{x[0]\} + j\,\mathrm{Im}\{x[0]\} \\ \mathrm{Re}\{x[1]\} + j\,\mathrm{Im}\{x[1]\} \\ \vdots \\ \mathrm{Re}\{x[N-1]\} + j\,\mathrm{Im}\{x[N-1]\} \end{bmatrix} = \mathrm{Re}\left\{ \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} \right\} + j\,\mathrm{Im}\left\{ \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} \right\}$$

- Polar form

$$x = \begin{bmatrix} |x[0]|\, e^{j\angle x[0]} \\ |x[1]|\, e^{j\angle x[1]} \\ \vdots \\ |x[N-1]|\, e^{j\angle x[N-1]} \end{bmatrix}$$

Summary

- Linear algebra provides powerful tools to study signals and systems
- Signals are **vectors** that live in a vector space
- Of particular interest in signal processing are the vector spaces $\mathbb{R}^N$ and $\mathbb{C}^N$



Linear Combinations of Signals

Linear Combination of Signals

- Many signal processing applications feature **sums** of a number of signals

DEFINITION

Given a collection of M vectors $x_0, x_1, \dots x_{M-1} \in \mathbb{C}^N$ and M scalars $\alpha_0, \alpha_1, \dots, \alpha_{M-1} \in \mathbb{C}$, the **linear combination** of the vectors is given by

$$y = \alpha_0 x_0 + \alpha_1 x_1 + \cdots + \alpha_{M-1} x_{M-1} = \sum_{m=0}^{M-1} \alpha_m x_m$$

- Clearly the result of the linear combination is a vector $y \in \mathbb{C}^N$

Linear Combination Example

- A recording studio uses a **mixing board** (or desk) to create a linear combination of the signals from the different instruments that make up a song
- Say $x_0 =$ drums, $x_1 =$ bass, $x_2 =$ guitar, $\ldots$, $x_{22} =$ saxophone, $x_{23} =$ singer ($M = 24$)
- Linear combination (output of mixing board)

$$y = \alpha_0 x_0 + \alpha_1 x_1 + \cdots + \alpha_{23} x_{23} = \sum_{m=0}^{23} \alpha_m x_m$$

- Changing the α_m's results in a different "mix" y that emphasizes/deemphasizes certain instruments

Linear Combination = Matrix Multiplication

- Step 1: Stack the vectors $x_m \in \mathbb{C}^N$ as column vectors into an $N \times M$ matrix

$$X = \left[x_0 | x_1 | \cdots | x_{M-1}\right]$$

- Step 2: Stack the scalars α_m into an $M \times 1$ column vector

$$a = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix}$$

- Step 3: We can now write a linear combination as the matrix/vector product

$$y = \alpha_0 x_0 + \alpha_1 x_1 + \cdots + \alpha_{M-1} x_{M-1} = \sum_{m=0}^{M-1} \alpha_m x_m = \left[x_0 | x_1 | \cdots | x_{M-1}\right] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix} = Xa$$

Linear Combination = Matrix Multiplication (The Gory Details)

- M vectors in $\mathbb{C}^N$: $x_m = \begin{bmatrix} x_m[0] \\ x_m[1] \\ \vdots \\ x_m[N-1] \end{bmatrix}$, $m = 0, 1, \ldots, M-1$
- $N \times M$ matrix: $X = \begin{bmatrix} x_0[0] & x_1[0] & \cdots & x_{M-1}[0] \\ x_0[1] & x_1[1] & \cdots & x_{M-1}[1] \\ \vdots & \vdots & & \vdots \\ x_0[N-1] & x_1[N-1] & \cdots & x_{M-1}[N-1] \end{bmatrix}$
- Note: The row-n, column-m element of the matrix $[X]_{n,m} = x_m[n]$
- M scalars α_m, $m = 0, 1, \ldots, M-1$: $a = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix}$
- Linear combination $y = Xa$

Linear Combination = Matrix Multiplication (Summary)

- Linear combination $y = Xa$

- The row-n, column-m element of the $N \times M$ matrix $[X]_{n,m} = x_m[n]$

$$y = \begin{bmatrix} \vdots \\ y[n] \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \\ \cdots & x_m[n] & \cdots \\ & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ \alpha_m \\ \vdots \end{bmatrix} = Xa$$

- Sum-based formula for $y[n]$

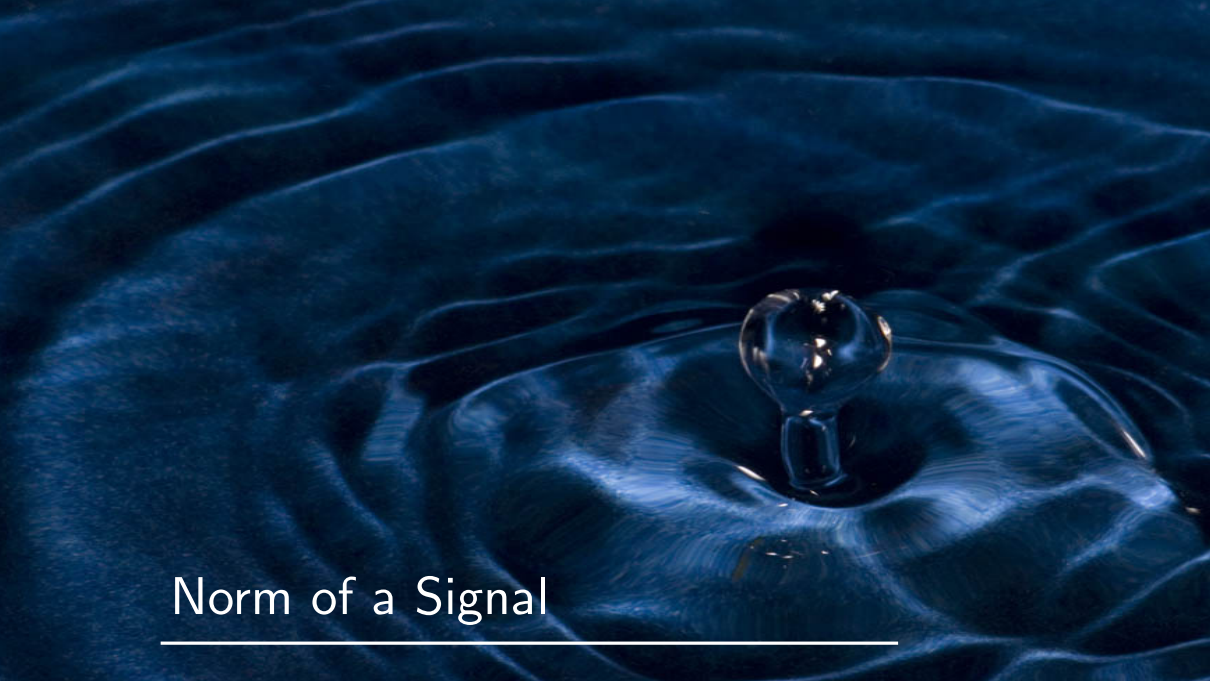
$$y[n] = \sum_{m=0}^{M-1} \alpha_m \, x_m[n]$$

Linear Combination as Matrix Multiplication (Matlab)

- **Click here** to view a MATLAB demo using sound to explore linear combinations.

Summary

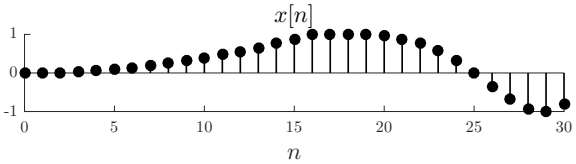
- Linear algebra provides power tools to study signals and systems
- Signals are **vectors** that live in a vector space
- We can combine several signals to form one new signal via a **linear combination**
- Linear combination is basically a matrix/vector multiplication



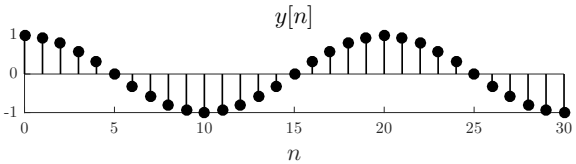
Norm of a Signal

Strength of a Vector

- How to quantify the "strength" of a vector?
- How to say that one signal is "stronger" than another?
- Signal x



- Signal y



Strength of a Vector: 2-Norm

DEFINITION

The **Euclidean length**, or 2-**norm**, of a vector $x \in \mathbb{C}^N$ is given by

$$\|x\|_2 = \sqrt{\sum_{n=0}^{N-1} |x[n]|^2}$$

The **energy** of x is given by $(\|x\|_2)^2 = \|x\|_2^2$

- The norm takes as input a vector in $\mathbb{C}^N$ and produces a **real number** that is ≥ 0
- When it is clear from context, we will suppress the subscript "2" in $\|x\|_2$ and just write $\|x\|$

2-Norm Example

- Ex: $x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

- ℓ_2 norm

$$\|x\|_2 \;=\; \sqrt{\sum_{n=0}^{N-1} |x[n]|^2} \;=\; \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$$

Strength of a Vector: p-Norm

- The Euclidean length is not the only measure of "strength" of a vector in $\mathbb{C}^N$

DEFINITION

The p-**norm** of a vector $x \in \mathbb{C}^N$ is given by

$$\|x\|_p = \left(\sum_{n=0}^{N-1} |x[n]|^p \right)^{1/p}$$

DEFINITION

The 1-**norm** of a vector $x \in \mathbb{C}^N$ is given by

$$\|x\|_1 = \sum_{n=0}^{N-1} |x[n]|$$

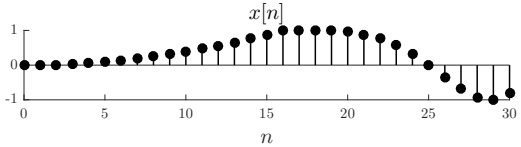
Strength of a Vector: ∞-Norm

DEFINITION

The ∞-**norm** of a vector $x \in \mathbb{C}^N$ is given by

$$\|x\|_\infty = \max_n |x[n]|$$

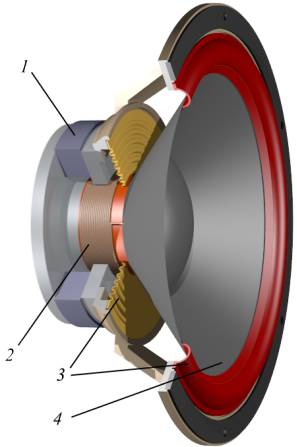
- $\|x\|_\infty$ is simply the largest entry in the vector x (in absolute value)



- While $\|x\|_2^2$ measures the **energy** in a signal, $\|x\|_\infty$ measures the **peak** value (of the magnitude); both are very useful in applications
- Interesting mathematical fact: $\|x\|_\infty = \lim_{p\to\infty} \|x\|_p$

Physical Significance of Norms (1)

- Two norms have special physical significance
  - $\|x\|_2^2$: energy in x
  - $\|x\|_\infty$: peak value in x
- A **loudspeaker** is a transducer that converts electrical signals into acoustic signals
- Conventional loudspeakers consist of a paper cone (4) that is joined to a coil of wire (2) that is wound around a permanent magnet (1)
- If the energy $\|x\|_2^2$ is too large, then the coil of wire will melt from excessive heating
- If the peak value $\|x\|_\infty$ is too large, then the large back and forth excursion of the coil of wire will tear it off of the paper cone



Physical Significance of Norms (2)

- Consider a **robotic car** that we wish to guide down a roadway
- How to measure the amount of deviation from the center of the driving lane?
- Let x be a vector of measurements of the car's GPS position and let y be a vector containing the GPS positions of the center of the driving lane
- Clearly we would like to make the error signal $y - x$ "small"; but how to measure smallness?
- Minimizing $\|y - x\|_2^2$, energy in the error signal, will tolerate a few large deviations from the lane center (not very safe)
- Minimizing $\|y - x\|_\infty$, the maximum of the error signal, will not tolerate any large deviations from the lane center (much safer)

Normalizing a Vector

DEFINITION

A vector x is **normalized** (in the 2-norm) if $\|x\|_2 = 1$

- Normalizing a vector is easy; just scale it by $\frac{1}{\|x\|_2}$
- Ex: $x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$, $||x||_2 = \sqrt{\sum_{n=0}^{N-1} |x[n]|^2} = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$

$$x' = \frac{1}{\sqrt{14}} x = \frac{1}{\sqrt{14}} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{14} \\ 2/\sqrt{14} \\ 3/\sqrt{14} \end{bmatrix}, \qquad ||x'||_2 = 1$$

Summary

- Linear algebra provides power tools to study signals and systems
- Signals are **vectors** that live in a vector space
- Norms measure the "strength" of a signal; we introduced the 2- 1-, and ∞-norms

Inner Product

The Geometry of Signals

- Up to this point, we have developed the viewpoint of "signals as vectors" in a vector space
- We have focused on quantities related to individual vectors, ex: norm (strength)
- Now we turn to a quantity related to pairs of vectors, the **inner product**
- A powerful and ubiquitous signal processing tool

Aside: Transpose of a Vector

- Recall that the **transpose** operation ^{T} converts a column vector to a row vector (and vice versa)

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}^T = \begin{bmatrix} x[0] & x[1] & \cdots & x[N-1] \end{bmatrix}$$

- In addition to transposition, the **conjugate transpose** (aka Hermitian transpose) operation ^{H} takes the complex conjugate

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}^H = \begin{bmatrix} x[0]^* & x[1]^* & \cdots & x[N-1]^* \end{bmatrix}$$

Inner Product

DEFINITION

The **inner product** (or dot product) between two vectors $x, y \in \mathbb{C}^N$ is given by

$$\langle x, y \rangle = y^H x = \sum_{n=0}^{N-1} x[n]\, y[n]^*$$

- The inner product takes two signals (vectors in $\mathbb{C}^N$) and produces a single (complex) number
- **Angle** between two vectors $x, y \in \mathbb{R}^N$

$$\cos\theta_{x,y} = \frac{\langle x, y \rangle}{\|x\|_2 \, \|y\|_2}$$

- **Angle** between two vectors $x, y \in \mathbb{C}^N$

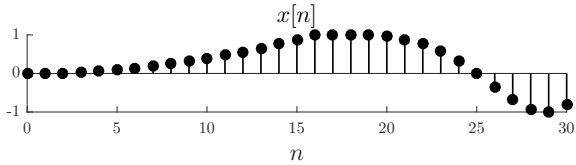
$$\cos\theta_{x,y} = \frac{\mathrm{Re}\{\langle x, y \rangle\}}{\|x\|_2 \, \|y\|_2}$$

Inner Product Example 1

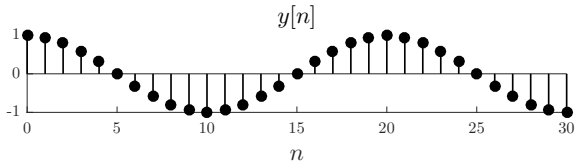
- Consider two vectors in $\mathbb{R}^2$: $x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$
- $\|x\|_2^2 = 1^2 + 2^2 = 5, \qquad \|y\|_2^2 = 3^2 + 2^2 = 13$
- $\theta_{x,y} = \arccos\left(\frac{1\times 3 \,+\, 2\times 2}{\sqrt{5}\sqrt{13}}\right) = \arccos\left(\frac{7}{\sqrt{65}}\right) \approx 0.519 \text{ rad} \approx 29.7^\circ$

Inner Product Example 2

- Signal x



- Signal y



- Inner product computed using Matlab: $\langle x, y \rangle = y^T x = 5.995$
- Angle computed using Matlab: $\theta_{x,y} = 64.9^\circ$

2-Norm from Inner Product

- Question: What's the inner product of a signal with itself?

$$\langle x, x \rangle = \sum_{n=0}^{N-1} x[n]\, x[n]^* = \sum_{n=0}^{N-1} |x[n]|^2 = \|x\|_2^2$$

- Answer: The 2-norm!

- Mathematical aside: This property makes the 2-norm very special; no other p-norm can be computed via the inner product like this

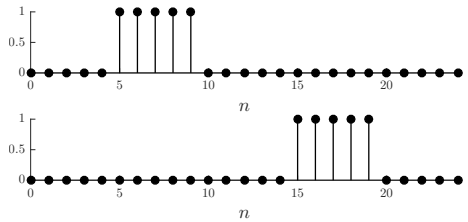
Orthogonal Vectors

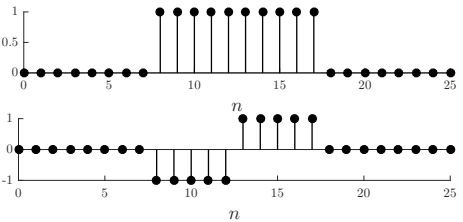
DEFINITION

Two vectors $x, y \in \mathbb{C}^N$ are **orthogonal** if

$$\langle x, y \rangle = 0$$

- $\langle x, y \rangle = 0 \;\Rightarrow\; \theta_{x,y} = \frac{\pi}{2} \text{ rad} = 90^\circ$
- Ex: Two sets of orthogonal signals





Harmonic Sinusoids are Orthogonal

$$s_k[n] = e^{j\frac{2\pi k}{N}n}, \quad n, k, N \in \mathbb{Z}, \quad 0 \le n \le N-1, \quad 0 \le k \le N-1$$

- Claim: $\langle s_k, s_l \rangle = 0, \quad k \neq l$ (a key result for the DFT)
- Verify by direct calculation

$$\begin{aligned}
\langle s_k, s_l \rangle &= \sum_{n=0}^{N-1} s_k[n]\, s_l^*[n] = \sum_{n=0}^{N-1} e^{j\frac{2\pi k}{N}n} \left(e^{j\frac{2\pi l}{N}n}\right)^* = \sum_{n=0}^{N-1} e^{j\frac{2\pi k}{N}n}\, e^{-j\frac{2\pi l}{N}n} \\
&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(k-l)n} \quad \text{let } r = k-l \in \mathbb{Z}, r \neq 0 \\
&= \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}rn} = \sum_{n=0}^{N-1} a^n \quad \text{with } a = e^{j\frac{2\pi}{N}r}, \text{then use } \sum_{n=0}^{N-1} a^n = \frac{1-a^N}{1-a} \\
&= \frac{1 - e^{j\frac{2\pi r N}{N}}}{1 - e^{j\frac{2\pi r}{N}}} = 0 \ \checkmark
\end{aligned}$$

Harmonic Sinusoids are Orthogonal (Matlab)

- **Click here** to view a MATLAB demo exploring the orthogonality of harmonic sinusoids.

Normalizing Harmonic Sinusoids

$$s_k[n] = e^{j\frac{2\pi k}{N}n}, \quad n, k, N \in \mathbb{Z}, \;\; 0 \le n \le N-1, \;\; 0 \le k \le N-1$$

- Claim: $\|s_k\|_2 = \sqrt{N}$
- Verify by direct calculation

$$\|s_k\|_2^2 \;=\; \sum_{n=0}^{N-1} |s_k[n]|^2 \;=\; \sum_{n=0}^{N-1} |e^{j\frac{2\pi k}{N}n}|^2 \;=\; \sum_{n=0}^{N-1} 1 \;=\; N \quad \checkmark$$

- **Normalized harmonic sinusoids**

$$\widetilde{s}_k[n] = \frac{1}{\sqrt{N}}\, e^{j\frac{2\pi k}{N}n}, \quad n, k, N \in \mathbb{Z}, \;\; 0 \le n \le N-1, \;\; 0 \le k \le N-1$$

Summary

- **Inner product** measures the similarity between two signals

$$\langle x, y \rangle = y^H x = \sum_{n=0}^{N-1} x[n]\, y[n]^*$$

- Angle between two signals

$$\cos \theta_{x,y} = \frac{\mathrm{Re}\{\langle x, y \rangle\}}{\|x\|_2 \, \|y\|_2}$$

- Harmonic sinusoids are **orthogonal** (as well as periodic)

Matrix Multiplication and Inner Product

Recall: Matrix Multiplication as a Linear Combination of Columns

- Consider the (real- or complex-valued) matrix multiplication $y = Xa$
- The row-n, column-m element of the $N \times M$ matrix $[X]_{n,m} = x_m[n]$
- We can compute y as a **linear combination** of the **columns of** X weighted by the elements in a

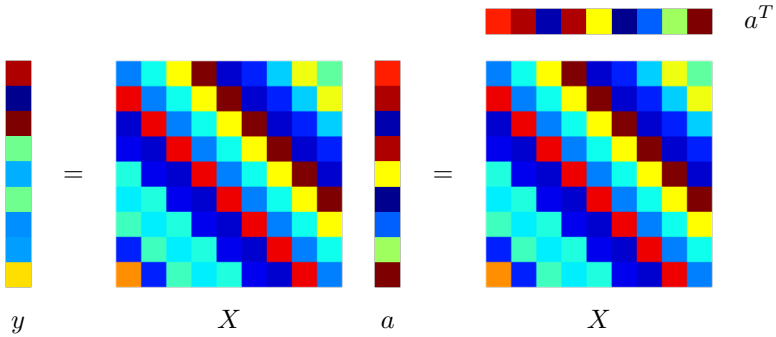
$$y = \begin{bmatrix} \vdots \\ y[n] \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_0[n] & x_1[n] & \cdots & x_{M-1}[n] \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix} = Xa$$

- Sum-based formula for $y[n]$

$$y[n] = \sum_{m=0}^{M-1} \alpha_m \, x_m[n], \; = \sum_{m=0}^{M-1} \alpha_m \, (\text{column } m \text{ of } X), \quad 0 \le n \le N-1$$

Visualizing Matrix Multiplication as a Linear Combination of Columns

- Consider the (real- or complex-valued) matrix multiplication $y = Xa$
- We can compute y as a **linear combination** of the **columns of** X weighted by the elements in a



Matrix Multiplication as a Sequence of Inner Products of Rows

- Consider the **real-valued** matrix multiplication $y = Xa$
- The row-n, column-m element of the $N \times M$ matrix $[X]_{n,m} = x_m[n]$
- We can compute each element $y[n]$ in y as the **inner product** of the **n-th row of X** with the vector a

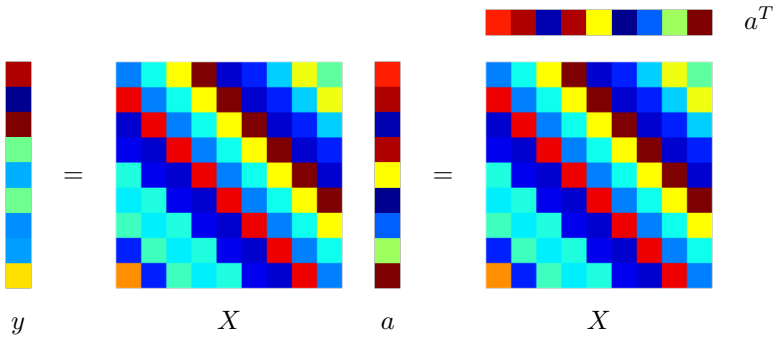
$$y = \begin{bmatrix} \vdots \\ y[n] \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_0[n] & x_1[n] & \cdots & x_{M-1}[n] \\ \vdots & \vdots & & \vdots \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{M-1} \end{bmatrix} = Xa$$

- Can write $y[n]$

$$y[n] = \sum_{m=0}^{M-1} \alpha_m \, x_m[n] = \langle (\text{row } n \text{ of } X)^T, \, a \rangle, \quad 0 \le n \le N-1$$

Visualizing Matrix Multiplication as a Sequence of Inner Products of Rows

- Consider the **real-valued** matrix multiplication $y = Xa$
- We can compute each element $y[n]$ in y as the **inner product** of the n-**th row of** X with the vector a



Matrix Multiplication as a Sequence of Inner Products of Rows

- What about **complex-valued** matrix multiplication $y = Xa$?
- The same interpretation works, but we need to use the following "**inner product**"

$$y[n] \;=\; \sum_{m=0}^{M-1} \alpha_m \, x_m[n] \;\neq\; \langle (\text{row } n \text{ of } X)^T,\, a \rangle, \quad 0 \leq n \leq N-1$$

- Note: This is *nearly* the inner product for complex signals except that is lacking the complex conjugation
- We will often abuse notation by calling this an inner product

Summary

- Given the matrix/vector product $y = Xa$, we can compute each element $y[n]$ in y as the **inner product** of the n-**th row of** X with the vector a

- Not strictly true for complex matrices/vectors, but the interpretation is useful nevertheless

Cauchy Schwarz Inequality

Comparing Signals

- Inner product and angle between vectors enable us to **compare signals**

$$\langle x, y \rangle = y^H x = \sum_{n=0}^{N-1} x[n]\, y[n]^*$$

$$\cos \theta_{x,y} = \frac{\mathrm{Re}\{\langle x, y \rangle\}}{\|x\|_2 \, \|y\|_2}$$

- The Cauchy Schwarz Inequality quantifies the comparison
- A powerful and ubiquitous signal processing tool
- Note: Our development will emphasize intuition over rigor

Cauchy-Schwarz Inequality (1)

- Focus on real-valued signals in $\mathbb{R}^N$ (the extension to $\mathbb{C}^N$ is easy)
- Recall that $\cos\theta_{x,y} = \frac{\langle x,y\rangle}{\|x\|_2\,\|y\|_2}$
- Now, use the fact that $0 \leq |\cos\theta| \leq 1$ to write

$$0 \;\leq\; \left|\frac{\langle x,y\rangle}{\|x\|_2\,\|y\|_2}\right| \;\leq\; 1$$

- Rewrite as the **Cauchy-Schwarz Inequality** (CSI)

$$0 \;\leq\; |\langle x,y\rangle| \;\leq\; \|x\|_2\,\|y\|_2$$

- Interpretation: The inner product $\langle x,y\rangle$ measures the **similarity** of x to y

Cauchy-Schwarz Inequality (2)

$$0 \;\leq\; |\langle x, y\rangle| \;\leq\; \|x\|_2 \, \|y\|_2$$

- Interpretation: The inner product $\langle x, y\rangle$ measures the **similarity** of x to y
- Two extreme cases:
  - Lower bound: $\langle x, y\rangle = 0$ or $\theta_{x,y} = 90°$: x and y are most different when they are orthogonal
  - Upper bound: $\langle x, y\rangle = \|x\|_2 \|y\|_2$ or $\theta_{x,y} = 0°$: x and y are most similar when they are collinear (aka linearly dependent, $y = \alpha x$)
- It is hard to understate the importance and ubiquity of the CSI!

Cauchy-Schwarz Inequality Applications

- How does a digital communication system decide whether the signal corresponding to a "0" was transmitted or the signal corresponding to a "1"? (Hint: CSI)
- How does a radar or sonar system find targets in the signal it receives after transmitting a pulse? (Hint: CSI)
- How do many computer vision systems find faces in images? (Hint: CSI)

Cauchy-Schwarz Inequality (Matlab)

- **Click here** to view a MATLAB demo illustrating the Cauchy-Schwarz inequality.

Summary

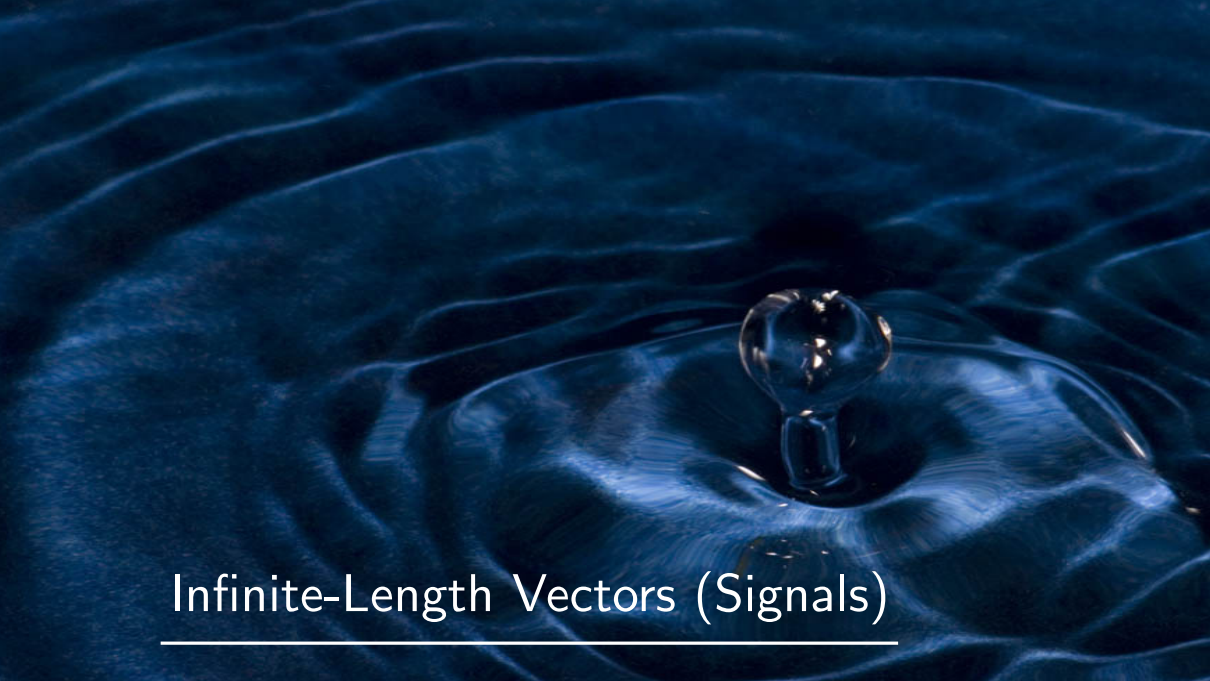
- **Inner product** measures the similarity between two signals

$$\langle x, y \rangle = y^H x = \sum_{n=0}^{N-1} x[n]\, y[n]^*$$

- **Cauchy-Schwarz Inequality** (CSI) calibrates the inner product

$$0 \leq \left| \frac{\langle x, y \rangle}{\|x\|_2 \, \|y\|_2} \right| \leq 1$$

  - Similar signals – close to upper bound (1)
  - Different signals – close to lower bound (0)

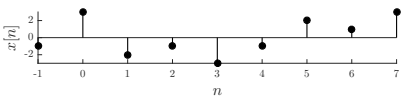


Infinite-Length Vectors (Signals)

From Finite to Infinite-Length Vectors

- Up to this point, we have developed some useful tools for dealing with finite-length vectors (signals) that live in $\mathbb{R}^N$ or $\mathbb{C}^N$: Norms, Inner product, Linear combination
- It turns out that these tools can be generalized to infinite-length vectors (sequences) by letting $N \to \infty$ (infinite-dimensional vector space, aka Hilbert Space)

$$x[n], \quad -\infty < n < \infty, \qquad x = \begin{bmatrix} \vdots \\ x[-2] \\ x[-1] \\ x[0] \\ x[1] \\ x[2] \\ \vdots \end{bmatrix}$$



- Obviously such a signal cannot be loaded into Matlab; however this viewpoint is still useful in many situations
- We will spell out the generalizations with emphasis on what changes from the finite-length case

2-Norm of an Infinite-Length Vector

DEFINITION

The 2-**norm** of an infinite-length vector x is given by

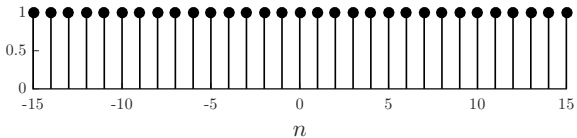
$$\|x\|_2 = \sqrt{\sum_{n=-\infty}^{\infty} |x[n]|^2}$$

The **energy** of x is given by $(\|x\|_2)^2 = \|x\|_2^2$

- When it is clear from context, we will suppress the subscript "2" in $\|x\|_2$ and just write $\|x\|$
- What changes from the finite-length case: Not every infinite-length vector has a finite 2-norm

ℓ_2 Norm of an Infinite-Length Vector – Example

- Signal: $x[n] = 1, \quad -\infty < n < \infty$



- 2-norm:

$$\|x\|_2^2 \;=\; \sum_{n=-\infty}^{\infty} |x[n]|^2 \;=\; \sum_{n=-\infty}^{\infty} 1 \;=\; \infty$$

- Infinite energy!

p- and 1-Norms of an Infinite-Length Vector

DEFINITION

The p-**norm** of an infinite-length vector x is given by

$$\|x\|_p = \left(\sum_{n=-\infty}^{\infty} |x[n]|^p \right)^{1/p}$$

DEFINITION

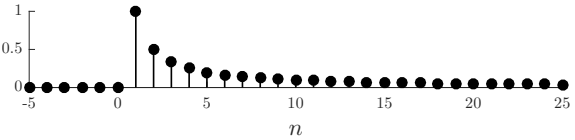
The 1-**norm** of an infinite-length vector x is given by

$$\|x\|_1 = \sum_{n=-\infty}^{\infty} |x[n]|$$

- What changes from the finite-length case: Not every infinite-length vector has a finite p-norm

1- and 2-Norms of an Infinite-Length Vector – Example

- Signal: $x[n] = \begin{cases} 0 & n \leq 0 \\ \frac{1}{n} & n \geq 1 \end{cases}$



- 1-norm

$$\|x\|_1 = \sum_{n=-\infty}^{\infty} |x[n]| = \sum_{n=1}^{\infty} \frac{1}{n} = \infty$$

- 2-norm

$$\|x\|_2^2 = \sum_{n=-\infty}^{\infty} |x[n]|^2 = \sum_{n=1}^{\infty} \left|\frac{1}{n}\right|^2 = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \approx 1.64 < \infty$$

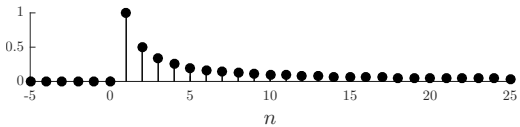
∞-Norm of an Infinite-Length Vector

DEFINITION

The ∞-**norm** of an infinite-length vector x is given by

$$\|x\|_\infty = \sup_n |x[n]|$$

- What changes from the finite-length case: "sup" is a generalization of $\max$ to infinite-length signals that lies beyond the scope of this course
- In both of the above examples, $\|x\|_\infty = 1$



Inner Product of Infinite-Length Signals

DEFINITION

The **inner product** between two infinite-length vectors x, y is given by

$$\langle x, y \rangle = \sum_{n=-\infty}^{\infty} x[n]\, y[n]^*$$

- The inner product takes two signals and produces a **single (complex) number**
- **Angle** between two real-valued signals

$$\cos\theta_{x,y} = \frac{\langle x, y \rangle}{\|x\|_2 \, \|y\|_2}$$

- **Angle** between two complex-valued signals

$$\cos\theta_{x,y} = \frac{\mathrm{Re}\{\langle x, y \rangle\}}{\|x\|_2 \, \|y\|_2}$$

Linear Combination of Infinite-Length Vectors

- The concept of a linear combination extends to infinite-length vectors

- What changes from the finite-length case: We will be especially interested in linear combinations of infinitely many infinite-length vectors

$$y = \sum_{m=-\infty}^{\infty} \alpha_m x_m$$

Linear Combination = Infinite Matrix Multiplication

- Step 1: Stack the vectors x_m as column vectors into a "matrix" with infinitely many rows and columns

$$X = \left[\cdots | x_{-1} | x_0 | x_1 | \cdots\right]$$

- Step 2: Stack the scalars α_m into an infinitely tall column vector $a = \begin{bmatrix} \vdots \\ \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \end{bmatrix}$

- Step 3: We can now write a linear combination as the matrix/vector product

$$y = \sum_{m=-\infty}^{\infty} \alpha_m x_m = \left[\cdots | x_{-1} | x_0 | x_1 | \cdots\right] \begin{bmatrix} \vdots \\ \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \end{bmatrix} = Xa$$

Linear Combination = Infinite Matrix Multiplication (The Gory Details)

- Vectors: $x_m = \begin{bmatrix} \vdots \\ x_m[-1] \\ x_m[0] \\ x_m[1] \\ \vdots \end{bmatrix}$, $-\infty < m < \infty$, and Scalars: $a = \begin{bmatrix} \vdots \\ \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \vdots \end{bmatrix}$
- Infinite matrix: $X = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & x_{-1}[-1] & x_0[-1] & x_1[-1] & \cdots \\ \cdots & x_{-1}[0] & x_0[0] & x_1[0] & \cdots \\ \cdots & x_{-1}[1] & x_0[1] & x_1[1] & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$
- Note: The row-n, column-m element of the matrix $[X]_{n,m} = x_m[n]$
- Linear combination $= Xa$

Linear Combination = Infinite Matrix Multiplication (Summary)

- Linear combination $y = Xa$

- The row-n, column-m element of the infinitely large matrix $[X]_{n,m} = x_m[n]$

$$y = \begin{bmatrix} \vdots \\ y[n] \\ \vdots \end{bmatrix} = \begin{bmatrix} & \vdots & \\ \cdots & x_m[n] & \cdots \\ & \vdots & \end{bmatrix} \begin{bmatrix} \vdots \\ \alpha_m \\ \vdots \end{bmatrix} = Xa$$

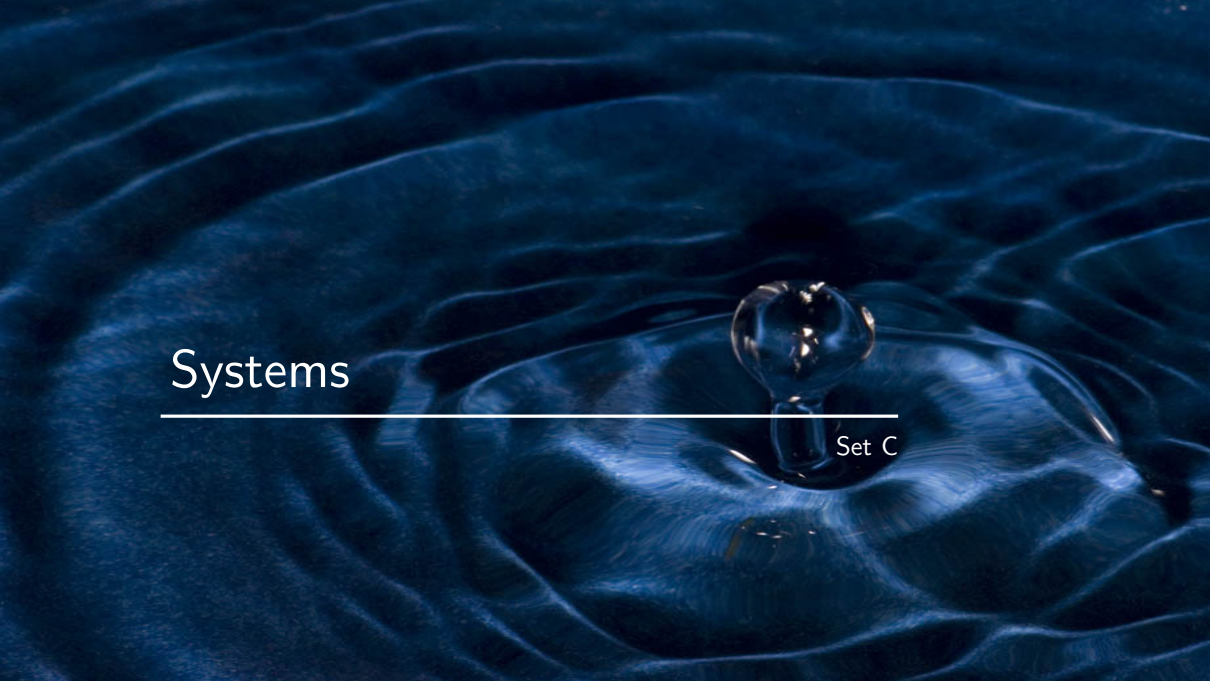
- Sum-based formula for $y[n]$

$$y[n] = \sum_{m=-\infty}^{\infty} \alpha_m \, x_m[n]$$

Summary

- Linear algebra concepts like norm, inner product, and linear combination work apply as well to infinite-length signals as with finite-length signals

- Only a few changes from the finite-length case
  - Not every infinite-length vector has a finite 1-, 2-, or ∞-norm
  - Linear combinations can involve infinitely many vectors



Systems

Set C

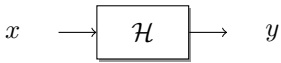
Systems

Systems are Transformations

DEFINITION

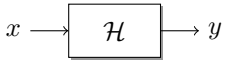
A discrete-time **system** $\mathcal{H}$ is a transformation (a rule or formula) that maps a discrete-time input signal x into a discrete-time output signal y

$$y = \mathcal{H}\{x\}$$



- Systems manipulate the information in signals
- Examples:
  - A speech recognition system converts acoustic waves of speech into text
  - A radar system transforms the received radar pulse to estimate the position and velocity of targets
  - A functional magnetic resonance imaging (fMRI) system transforms measurements of electron spin into voxel-by-voxel estimates of brain activity
  - A 30 day moving average smooths out the day-to-day variability in a stock price

Signal Length and Systems



- Recall that there are two kinds of signals: infinite-length and finite-length
- Accordingly, we will consider two kinds of systems:
  1. Systems that transform an infinite-length-signal x into an infinite-length signal y
  2. Systems that transform a length-N signal x into a length-N signal y

     (Such systems can also be used to process periodic signals with period N)
- For generality, we will assume that the input and output signals are complex valued

System Examples (1)

- Identity

$$y[n] = x[n] \quad \forall n$$

- Scaling

$$y[n] = 2\,x[n] \quad \forall n$$

- Offset

$$y[n] = x[n] + 2 \quad \forall n$$

- Square signal

$$y[n] = (x[n])^2 \quad \forall n$$

- Shift

$$y[n] = x[n+2] \quad \forall n$$

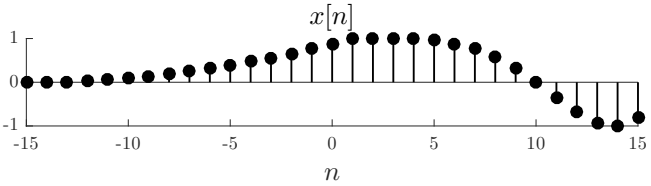
- Decimate

$$y[n] = x[2n] \quad \forall n$$

- Square time

$$y[n] = x[n^2] \quad \forall n$$

System Examples (2)



- Shift system ($m \in \mathbb{Z}$ fixed)

$$y[n] = x[n - m] \quad \forall n$$

- Moving average (combines shift, sum, scale)

$$y[n] = \frac{1}{2}(x[n] + x[n - 1]) \quad \forall n$$

- Recursive average

$$y[n] = x[n] + \alpha\, y[n - 1] \quad \forall n$$

Summary

- Systems transform one signal into another to manipulate information
- We will consider two kinds of systems:
  1. Systems that transform an infinite-length-signal x into an infinite-length signal y
  2. Systems that transform a length-N signal x into a length-N signal y

     (Such systems can also be used to process periodic signals with period N)

Linear Systems

Linear Systems

DEFINITION

A system $\mathcal{H}$ is (zero-state) **linear** if it satisfies the following two properties:

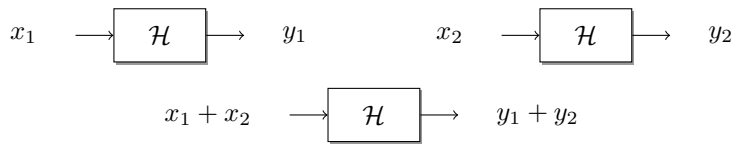
1. Scaling

$$\mathcal{H}\{\alpha\, x\} = \alpha\, \mathcal{H}\{x\} \quad \forall\ \alpha \in \mathbb{C}$$

2. Additivity

$$\text{If} \quad y_1 = \mathcal{H}\{x_1\} \quad \text{and} \quad y_2 = \mathcal{H}\{x_2\} \quad \text{then}$$

$$\mathcal{H}\{x_1 + x_2\} = y_1 + y_2$$



Linearity Notes

- A system that is not linear is called **nonlinear**
- To prove that a system is linear, you must prove rigorously that it has **both** the scaling and additivity properties for **arbitrary** input signals
- To prove that a system is nonlinear, it is sufficient to exhibit a **counterexample**

Example: Moving Average is Linear (Scaling)

$$x[n] \longrightarrow \boxed{\mathcal{H}} \longrightarrow y[n] = \tfrac{1}{2}(x[n] + x[n-1])$$

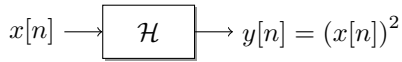
- **Scaling:** (Strategy to prove – Scale input x by $\alpha \in \mathbb{C}$, compute output y via the formula at top, and verify that it is scaled as well)
  - Let
    $$x'[n] \;=\; \alpha x[n], \quad \alpha \in \mathbb{C}$$
  - Let y' denote the output when x' is input (that is, $y' = \mathcal{H}\{x'\}$)
  - Then
    $$y'[n] \;=\; \frac{1}{2}(x'[n] + x'[n-1]) \;=\; \frac{1}{2}(\alpha x[n] + \alpha x[n-1]) \;=\; \alpha\left(\frac{1}{2}(x[n] + x[n-1])\right) \;=\; \alpha y[n] \;\checkmark$$

Example: Moving Average is Linear (Additivity)

$$x[n] \longrightarrow \boxed{\mathcal{H}} \longrightarrow y[n] = \tfrac{1}{2}(x[n] + x[n-1])$$

- **Additivity:** (Strategy to prove – Input two signals into the system and verify that the output equals the sum of the respective outputs)
  - Let
  $$x'[n] \;=\; x_1[n] + x_2[n]$$
  - Let $y'/y_1/y_2$ denote the output when $x'/x_1/x_2$ is input
  - Then
  $$\begin{aligned} y'[n] &= \frac{1}{2}(x'[n] + x'[n-1]) \;=\; \frac{1}{2}(\{x_1[n] + x_2[n]\} + \{x_1[n-1] + x_2[n-1]\}) \\ &= \frac{1}{2}(x_1[n] + x_1[n-1]) + \frac{1}{2}(x_2[n] + x_2[n-1]) \;=\; y_1[n] + y_2[n] \;\checkmark \end{aligned}$$

Example: Squaring is Nonlinear



- **Additivity:** Input two signals into the system and see what happens
  - Let
$$y_1[n] = (x_1[n])^2, \qquad y_2[n] = (x_2[n])^2$$
  - Set
$$x'[n] = x_1[n] + x_2[n]$$
  - Then
$$y'[n] = \left(x'[n]\right)^2 = (x_1[n] + x_2[n])^2 = (x_1[n])^2 + 2x_1[n]x_2[n] + (x_2[n])^2 \neq y_1[n] + y_2[n]$$
  - Nonlinear!

Linear or Nonlinear? You Be the Judge! (1)

- Identity

$$y[n] = x[n] \quad \forall n$$

- Scaling

$$y[n] = 2\,x[n] \quad \forall n$$

- Offset

$$y[n] = x[n] + 2 \quad \forall n$$

- Square signal

$$y[n] = (x[n])^2 \quad \forall n$$

- Shift

$$y[n] = x[n+2] \quad \forall n$$

- Decimate

$$y[n] = x[2n] \quad \forall n$$

- Square time

$$y[n] = x[n^2] \quad \forall n$$

Linear or Nonlinear? You Be the Judge! (2)

- Shift system ($m \in \mathbb{Z}$ fixed)

$$y[n] = x[n - m] \quad \forall n$$

- Moving average (combines shift, sum, scale)

$$y[n] = \frac{1}{2}(x[n] + x[n - 1]) \quad \forall n$$

- Recursive average

$$y[n] = x[n] + \alpha\, y[n - 1] \quad \forall n$$

Matrix Multiplication and Linear Systems

- Matrix multiplication (aka Linear Combination) is a fundamental signal processing system
- **Fact 1:** Matrix multiplications are linear systems (easy to show at home, but do it!)

$$y = \mathbf{H}\,x$$

$$y[n] = \sum_m [\mathbf{H}]_{n,m}\,x[m]$$

(Note: This formula applies for both infinite-length and finite-length signals)

- **Fact 2:** *All* linear systems can be expressed as matrix multiplications
- As a result, we will use the matrix viewpoint of linear systems extensively in the sequel
- Try at home: Express all of the linear systems in the examples above in matrix form

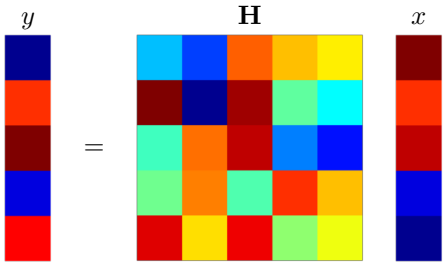
Matrix Multiplication and Linear Systems in Pictures

- Linear system

$$y \;=\; \mathbf{H}\, x$$

$$y[n] \;=\; \sum_m [\mathbf{H}]_{n,m}\, x[m] \;=\; \sum_m h_{n,m}\, x[m]$$

where $h_{n,m} \;=\; [\mathbf{H}]_{n,m}$ represents the row-n, column-m entry of the matrix $\mathbf{H}$



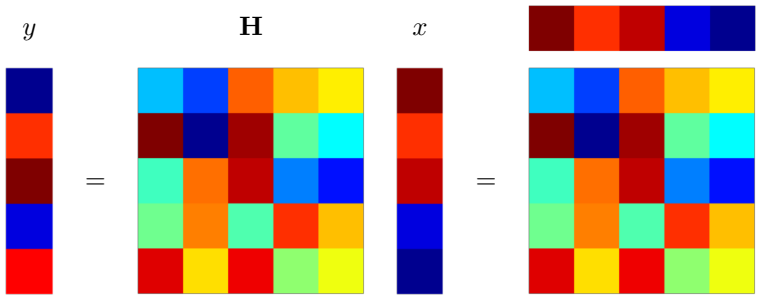
System Output as a Linear Combination of Columns

- Linear system

$$y \;=\; \mathbf{H}\, x$$

$$y[n] \;=\; \sum_m [\mathbf{H}]_{n,m}\, x[m] \;=\; \sum_m h_{n,m}\, x[m]$$

where $h_{n,m} \;=\; [\mathbf{H}]_{n,m}$ represents the row-n, column-m entry of the matrix $\mathbf{H}$



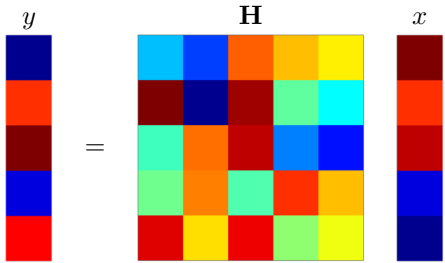
System Output as a Sequence of Inner Products

- Linear system

$$y \;=\; \mathbf{H}\, x$$

$$y[n] \;=\; \sum_m [\mathbf{H}]_{n,m}\, x[m] \;=\; \sum_m h_{n,m}\, x[m]$$

where $h_{n,m} \;=\; [\mathbf{H}]_{n,m}$ represents the row-n, column-m entry of the matrix $\mathbf{H}$



Summary

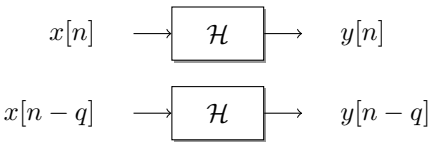
- Linear systems satisfy (1) scaling and (2) additivity
- To show a system is linear, you have to prove it rigorously assuming arbitrary inputs (work!)
- To show a system is nonlinear, you can just exhibit a counterexample (often easy!)
- Linear systems $\equiv$ matrix multiplication
  - Justifies our emphasis on linear vector spaces and matrices
  - The output signal y equals the linear combination of the columns of $\mathbf{H}$ weighted by the entries in x
  - Alternatively, the output value $y[n]$ equals the inner product between row n of $\mathbf{H}$ with x

Time-Invariant Systems

Time-Invariant Systems (Infinite-Length Signals)

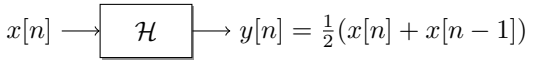
DEFINITION

A system $\mathcal{H}$ processing infinite-length signals is **time-invariant** (shift-invariant) if a time shift of the input signal creates a corresponding time shift in the output signal



- Intuition: A time-invariant system behaves the same no matter when the input is applied
- A system that is not time-invariant is called **time-varying**

Example: Moving Average is Time-Invariant



- Let

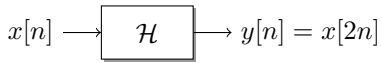
$$x'[n] = x[n-q], \quad q \in \mathbb{Z}$$

- Let y' denote the output when x' is input (that is, $y' = \mathcal{H}\{x'\}$)

- Then

$$y'[n] \;=\; \frac{1}{2}(x'[n] + x'[n-1]) \;=\; \frac{1}{2}(x[n-q] + x[n-q-1]) \;=\; y[n-q] \;\checkmark$$

Example: Decimation is Time-Varying



- This system is time-varying; demonstrate with a counter-example
- Let
$$x'[n] = x[n-1]$$
- Let y' denote the output when x' is input (that is, $y' = \mathcal{H}\{x'\}$)
- Then
$$y'[n] \;=\; x'[2n] \;=\; x[2n-1] \;\neq\; x[2(n-1)] \;=\; y[n-1]$$

Time-Invariant or Time-Varying? You Be the Judge! (1)

- Identity

$$y[n] = x[n] \quad \forall n$$

- Scaling

$$y[n] = 2\,x[n] \quad \forall n$$

- Offset

$$y[n] = x[n] + 2 \quad \forall n$$

- Square signal

$$y[n] = (x[n])^2 \quad \forall n$$

- Shift

$$y[n] = x[n+2] \quad \forall n$$

- Decimate

$$y[n] = x[2n] \quad \forall n$$

- Square time

$$y[n] = x[n^2] \quad \forall n$$

Time-Invariant or Time-Varying? You Be the Judge! (2)

- Shift system ($m \in \mathbb{Z}$ fixed)

$$y[n] = x[n - m] \quad \forall n$$

- Moving average (combines shift, sum, scale)

$$y[n] = \frac{1}{2}(x[n] + x[n - 1]) \quad \forall n$$

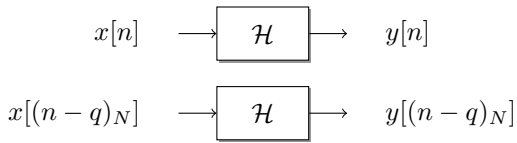
- Recursive average

$$y[n] = x[n] + \alpha\, y[n - 1] \quad \forall n$$

Time-Invariant Systems (Finite-Length Signals)

DEFINITION

A system $\mathcal{H}$ processing length-N signals is **time-invariant** (shift-invariant) if a circular time shift of the input signal creates a corresponding circular time shift in the output signal



- Intuition: A time-invariant system behaves the same no matter when the input is applied
- A system that is not time-invariant is called **time-varying**

Summary

- Time-invariant systems behave the same no matter when the input is applied
- Infinite-length signals: Invariance with respect to any integer time shift
- Finite-length signals: Invariance with respect to a circular time shift
- To show a system is time-invariant, you have to prove it rigorously assuming arbitrary inputs (work!)
- To show a system is time-varying, you can just exhibit a counterexample (often easy!)

Linear Time-Invariant Systems

Linear Time Invariant (LTI) Systems

DEFINITION

A system $\mathcal{H}$ is **linear time-invariant** (LTI) if it is both linear and time-invariant

- LTI systems are the foundation of signal processing and the main subject of this course

LTI or Not? You Be the Judge! (1)

- Identity

$$y[n] = x[n] \quad \forall n$$

- Scaling

$$y[n] = 2\,x[n] \quad \forall n$$

- Offset

$$y[n] = x[n] + 2 \quad \forall n$$

- Square signal

$$y[n] = (x[n])^2 \quad \forall n$$

- Shift

$$y[n] = x[n+2] \quad \forall n$$

- Decimate

$$y[n] = x[2n] \quad \forall n$$

- Square time

$$y[n] = x[n^2] \quad \forall n$$

LTI or Not? You Be the Judge! (2)

- Shift system ($m \in \mathbb{Z}$ fixed)

$$y[n] = x[n - m] \quad \forall n$$

- Moving average (combines shift, sum, scale)

$$y[n] = \frac{1}{2}(x[n] + x[n - 1]) \quad \forall n$$

- Recursive average

$$y[n] = x[n] + \alpha\, y[n - 1] \quad \forall n$$

Matrix Multiplication and LTI Systems (Infinite-Length Signals)

- Recall that all linear systems can be expressed as matrix multiplications

$$y = \mathbf{H}\, x$$

$$y[n] = \sum_m [\mathbf{H}]_{n,m}\, x[m]$$

  Here $\mathbf{H}$ is a matrix with infinitely many rows and columns

- Let $h_{n,m} = [\mathbf{H}]_{n,m}$ represent the row-n, column-m entry of the matrix $\mathbf{H}$

$$y[n] = \sum_m h_{n,m}\, x[m]$$

- When the linear system is also shift invariant, $\mathbf{H}$ has a special structure

Matrix Structure of LTI Systems (Infinite-Length Signals)

- Linear system for infinite-length signals can be expressed as

$$y[n] \;=\; \mathcal{H}\{x[n]\} \;=\; \sum_{m=-\infty}^{\infty} h_{n,m}\, x[m], \quad -\infty < n < \infty$$

- Enforcing time invariance implies that for all $q \in \mathbb{Z}$

$$\mathcal{H}\{x[n-q]\} \;=\; \sum_{m=-\infty}^{\infty} h_{n,m}\, x[m-q] \;=\; y[n-q]$$

- Change of variables: $n' = n - q$ and $m' = m - q$

$$\mathcal{H}\{x[n']\} \;=\; \sum_{m'=-\infty}^{\infty} h_{n'+q,m'+q}\, x[m'] \;=\; y[n']$$

- Comparing first and third equations, we see that for an LTI system

$$h_{n,m} \;=\; h_{n+q,m+q} \quad \forall\, q \in \mathbb{Z}$$

LTI Systems are Toeplitz Matrices (Infinite-Length Signals) (1)

- For an LTI system with infinite-length signals

$$h_{n,m} \;=\; h_{n+q,m+q} \quad \forall\, q \in \mathbb{Z}$$

$$\mathbf{H} \;=\; \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & h_{-1,-1} & h_{-1,0} & h_{-1,1} & \cdots \\ \cdots & h_{0,-1} & h_{0,0} & h_{0,1} & \cdots \\ \cdots & h_{1,-1} & h_{1,0} & h_{1,1} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} \;=\; \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & h_{0,0} & h_{-1,0} & h_{-2,0} & \cdots \\ \cdots & h_{1,0} & h_{0,0} & h_{-1,0} & \cdots \\ \cdots & h_{2,0} & h_{1,0} & h_{0,0} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$$

- Entries on the matrix diagonals are the same – **Toeplitz matrix**

LTI Systems are Toeplitz Matrices (Infinite-Length Signals) (2)

- All of the entries in a Toeplitz matrix can be expressed in terms of the entries of the
  - 0-**th column**: $h[n] = h_{n,0}$
  - **Time-reversed** 0-**th row**: $h[m] = h_{0,-m}$

$$\mathbf{H} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & h_{0,0} & h_{-1,0} & h_{-2,0} & \cdots \\ \cdots & h_{1,0} & h_{0,0} & h_{-1,0} & \cdots \\ \cdots & h_{2,0} & h_{1,0} & h_{0,0} & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix} = \begin{bmatrix} & \vdots & \vdots & \vdots & \\ \cdots & h[0] & h[-1] & h[-2] & \cdots \\ \cdots & h[1] & h[0] & h[-1] & \cdots \\ \cdots & h[2] & h[1] & h[0] & \cdots \\ & \vdots & \vdots & \vdots & \end{bmatrix}$$

- Row-n, column-m entry of the matrix $[\mathbf{H}]_{n,m} = h_{n,m} = h[n-m]$

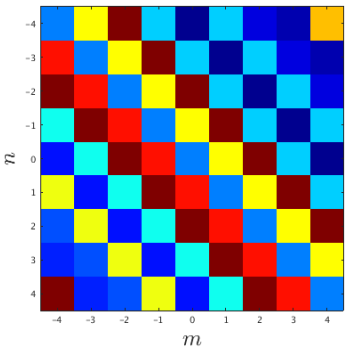
LTI Systems are Toeplitz Matrices (Infinite-Length Signals) (3)

- All of the entries in a Toeplitz matrix can be expressed in terms of the entries of the
  - 0-**th column**: $h[n] = h_{n,0}$ (this is an infinite-length signal/column vector; call it h)
  - **Time-reversed** 0-**th row**: $h[m] = h_{0,-m}$

- Example: Snippet of a Toeplitz matrix

$$[\mathbf{H}]_{n,m} = h_{n,m} = h[n-m]$$

$$\mathbf{H} =$$



$$h =$$

- Note the diagonals!

Matrix Structure of LTI Systems (Finite-Length Signals)

- Linear system for signals of length N can be expressed as

$$y[n] \;=\; \mathcal{H}\{x[n]\} \;=\; \sum_{m=0}^{N-1} h_{n,m}\, x[m], \quad 0 \le n \le N-1$$

- Enforcing time invariance implies that for all $q \in \mathbb{Z}$

$$\mathcal{H}\{x[(n-q)_N]\} \;=\; \sum_{m=0}^{N-1} h_{n,m}\, x[(m-q)_N] \;=\; y[(n-q)_N]$$

- Change of variables: $n' = n - q$ and $m' = m - q$

$$\mathcal{H}\{x[(n')_N]\} \;=\; \sum_{m'=-q}^{N-1-q} h_{(n'+q)_N,(m'+q)_N}\, x[(m')_N] \;=\; y[(n')_N]$$

- Comparing first and third equations, we see that for an LTI system

$$h_{n,m} \;=\; h_{(n+q)_N,(m+q)_N} \quad \forall\, q \in \mathbb{Z}$$

LTI Systems are Circulant Matrices (Finite-Length Signals) (1)

- For an LTI system with length-N signals

$$h_{n,m} \;=\; h_{(n+q)_N,(m+q)_N} \quad \forall\, q \in \mathbb{Z}$$

$$\mathbf{H} \;=\; \begin{bmatrix} h_{0,0} & h_{0,1} & h_{0,2} & \cdots & h_{0,N-1} \\ h_{1,0} & h_{1,1} & h_{1,2} & \cdots & h_{1,N-1} \\ h_{2,0} & h_{2,1} & h_{2,2} & \cdots & h_{2,N-1} \\ \vdots & \vdots & \vdots & & \vdots \\ h_{N-1,0} & h_{N-1,1} & h_{N-1,2} & \cdots & h_{N-1,N-1} \end{bmatrix} \;=\; \begin{bmatrix} h_{0,0} & h_{N-1,0} & h_{N-2,0} & \cdots & h_{1,0} \\ h_{1,0} & h_{0,0} & h_{N-1,0} & \cdots & h_{2,0} \\ h_{2,0} & h_{1,0} & h_{0,0} & \cdots & h_{3,0} \\ \vdots & \vdots & \vdots & & \vdots \\ h_{N-1,0} & h_{N-2,0} & h_{N-3,0} & \cdots & h_{0,0} \end{bmatrix}$$

- Entries on the matrix diagonals are the same + circular wraparound – **circulant matrix**

LTI Systems are Circulant Matrices (Finite-Length Signals) (2)

- All of the entries in a circulant matrix can be expressed in terms of the entries of the
  - 0-**th column**: $h[n] = h_{n,0}$
  - **Circularly time-reversed** 0-**th row**: $h[m] = h_{0,(-m)_N}$

$$\mathbf{H} = \begin{bmatrix} h_{0,0} & h_{N-1,0} & h_{N-2,0} & \cdots & h_{1,0} \\ h_{1,0} & h_{0,0} & h_{N-1,0} & \cdots & h_{2,0} \\ h_{2,0} & h_{1,0} & h_{0,0} & \cdots & h_{3,0} \\ \vdots & \vdots & \vdots & & \vdots \\ h_{N-1,0} & h_{N-2,0} & h_{N-3,0} & \cdots & h_{0,0} \end{bmatrix} = \begin{bmatrix} h[0] & h[N-1] & h[N-2] & \cdots & h[1] \\ h[1] & h[0] & h[N-1] & \cdots & h[2] \\ h[2] & h[1] & h[0] & \cdots & h[3] \\ \vdots & \vdots & \vdots & & \vdots \\ h[N-1] & h[N-2] & h[N-3] & \cdots & h[0] \end{bmatrix}$$

- Row-n, column-m entry of the matrix $[\mathbf{H}]_{n,m} = h_{n,m} = h[(n-m)_N]$

LTI Systems are Circulant Matrices (Finite-Length Signals) (3)

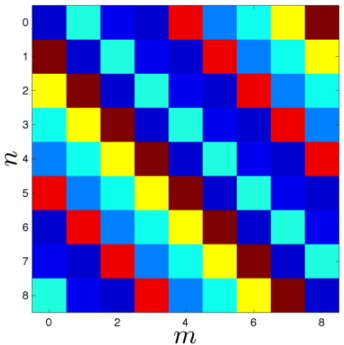
- All of the entries in a circulant matrix can be expressed in terms of the entries of the
  - 0-**th column**: $h[n] = h_{n,0}$ (this is a signal/column vector; call it h)
  - **Circularly time-reversed** 0-**th row**: $h[m] = h_{0,-m}$

- Example: circulant matrix

$$[\mathbf{H}]_{n,m} = h_{n,m} = h[(n-m)_N]$$

- Note the diagonals and circulant shifts!

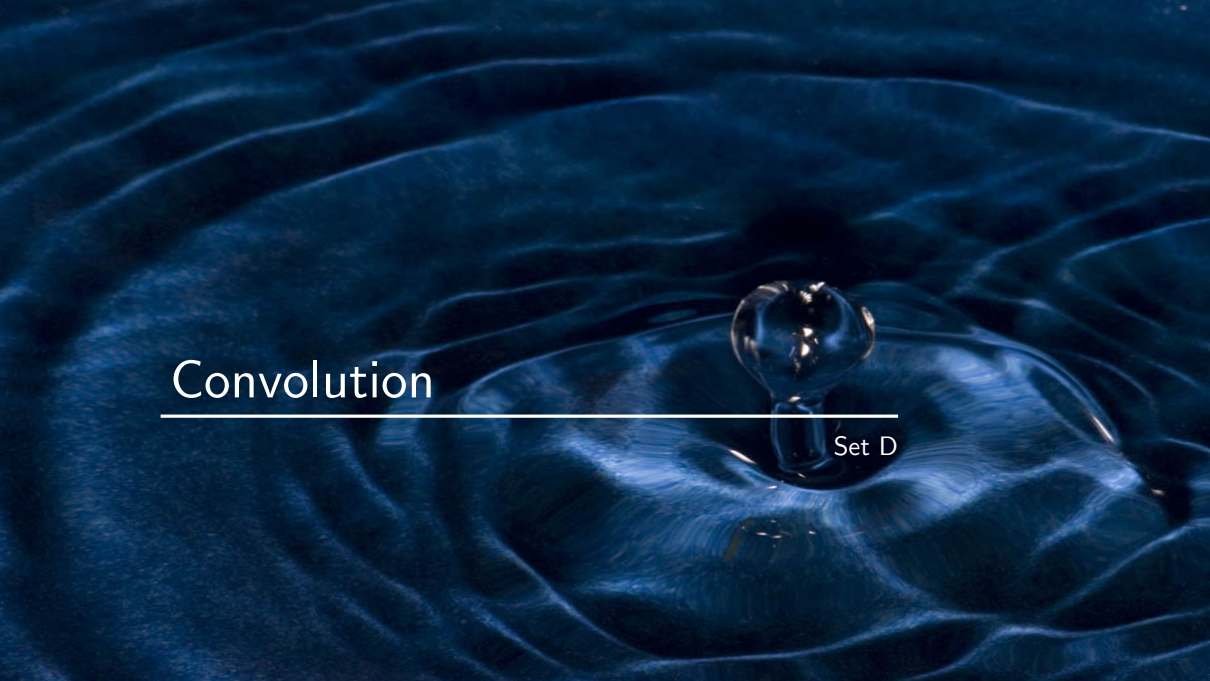
$\mathbf{H} =$



$h =$

Summary

- LTI = Linear + Time-Invariant
- Fundamental signal processing system (and our focus for the rest of the course)
- Infinite-length signals: System = Toeplitz matrix $\mathbf{H}$
  - $[\mathbf{H}]_{n,m} = h_{n,m} = h[n-m]$
- Finite-length signals: System = circulant matrix $\mathbf{H}$
  - $[\mathbf{H}]_{n,m} = h_{n,m} = h[(n-m)_N]$

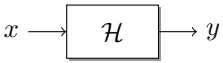


Convolution

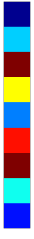
Set D

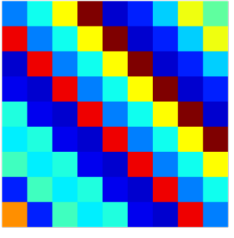
Impulse Response

Recall: LTI Systems are Toeplitz Matrices (Infinite-Length Signals)



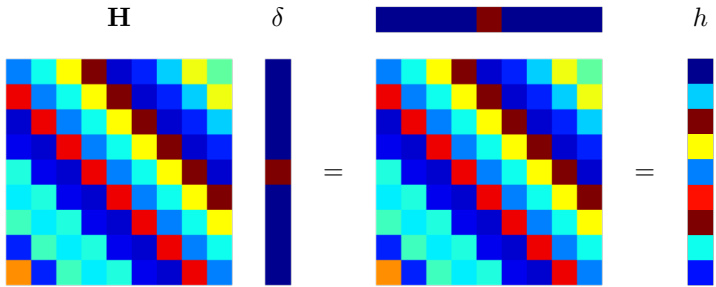
- LTI system $=$ multiplication by infinitely large Toeplitz matrix $\mathbf{H}$: $y = \mathbf{H}x$
- All of the entries in $\mathbf{H}$ can be obtained from the
  - 0-**th column**: $h[n] = h_{n,0}$ (this is a signal/column vector; call it h)
  - **Time-reversed** 0-**th row**: $h[m] = h_{0,-m}$
- $[\mathbf{H}]_{n,m} = h_{n,m}$
  $= h[n-m]$
- Columns/rows of $\mathbf{H}$ are shifted versions of the 0-th column/row

$h =$ 

$\mathbf{H} =$ 

Impulse Response (Infinite-Length Signals)

- The 0-th column of the matrix $\mathbf{H}$ – the column vector h – has a special interpretation
- Compute the output when the input is a **delta function** (impulse): $\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$



- This suggests that we call h the **impulse response** of the system

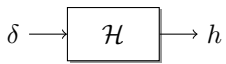
Impulse Response from Formulas (Infinite-Length Signals)

- General formula for LTI matrix multiplication

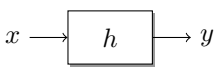
$$y[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- Let the input $x[n] = \delta[n]$ and compute

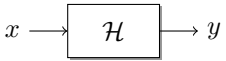
$$\sum_{m=-\infty}^{\infty} h[n-m]\, \delta[m] \;=\; h[n] \;\checkmark$$



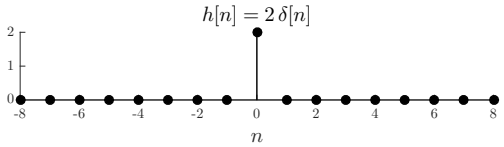
- The impulse response characterizes an LTI system
  (that is, carries all of the information contained in the matrix $\mathbf{H}$)



Example: Impulse Response of the Scaling System



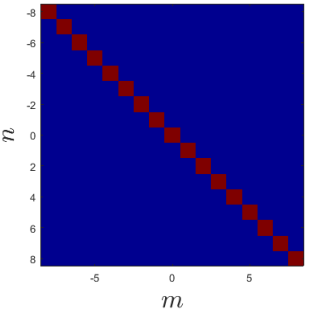
- Consider system for infinite-length signals; finite-length signal case is similar



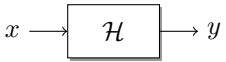
- Scaling system: $y[n] \;=\; \mathcal{H}\{x[n]\} \;=\; 2\,x[n]$
- Impulse response: $h[n] \;=\; \mathcal{H}\{\delta[n]\} \;=\; 2\,\delta[n]$
- Toeplitz system matrix:

$$[\mathbf{H}]_{n,m} = h[n-m] = 2\,\delta[n-m]$$

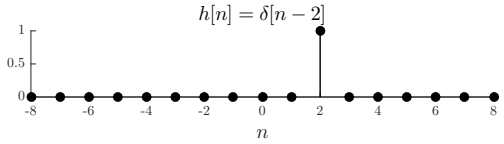
$\mathbf{H} =$



Example: Impulse Response of the Shift System



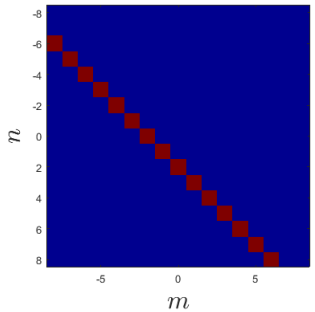
- Consider system for infinite-length signals; finite-length signal case uses circular shift



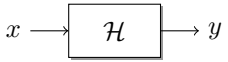
- Shift system: $y[n] = \mathcal{H}\{x[n]\} = x[n-2]$
- Impulse response: $h[n] = \mathcal{H}\{\delta[n]\} = \delta[n-2]$
- Toeplitz system matrix:

$$[\mathbf{H}]_{n,m} = h[n-m] = \delta[n-m-2]$$

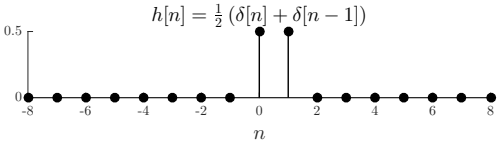
$\mathbf{H} =$



Example: Impulse Response of the Moving Average System



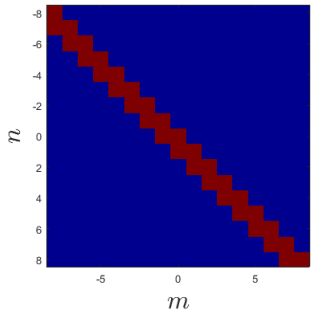
- Consider system for infinite-length signals; finite-length signal case is similar



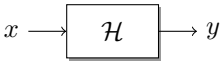
- Moving average system: $y[n] = \mathcal{H}\{x[n]\} = \frac{1}{2}(x[n] + x[n-1])$
- Impulse response: $h[n] = \mathcal{H}\{\delta[n]\} = \frac{1}{2}(\delta[n] + \delta[n-1])$
- Toeplitz system matrix:

$$[\mathbf{H}]_{n,m} = h[n-m] = \frac{1}{2}(\delta[n-m] + \delta[n-m-1])$$

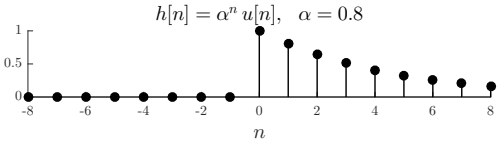
$\mathbf{H} =$



Example: Impulse Response of the Recursive Average System



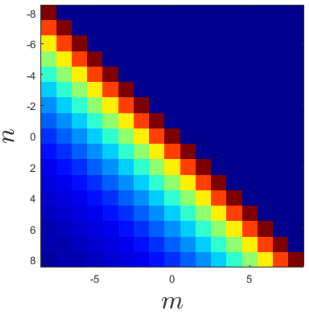
- Consider system for infinite-length signals; finite-length signal case is similar



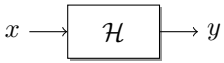
- Recursive average system: $y[n] \;=\; \mathcal{H}\{x[n]\} \;=\; x[n] + \alpha\, y[n-1]$
- Impulse response: $h[n] \;=\; \mathcal{H}\{\delta[n]\} \;=\; \alpha^n\, u[n]$
- Toeplitz system matrix:

$$[\mathbf{H}]_{n,m} = h[n-m] = \alpha^{n-m}\, u[n-m]$$

$\mathbf{H} \;=$



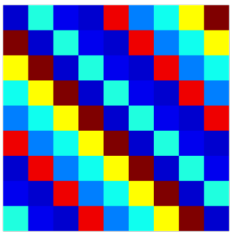
Recall: LTI Systems are Circulant Matrices (Finite-Length Signals)



- LTI system $=$ multiplication by $N \times N$ circulant matrix $\mathbf{H}$: $y = \mathbf{H}x$
- All of the entries in $\mathbf{H}$ can be obtained from the
  - 0-**th column**: $h[n] = h_{n,0}$ (this is a signal/column vector; call it h)
  - **Time-reversed** 0-**th row**: $h[m] = h_{0,(-m)_N}$
- $[\mathbf{H}]_{n,m} = h_{n,m}$
  $= h[(n-m)_N]$
- Columns/rows of $\mathbf{H}$ are circularly shifted versions of the 0-th column/row

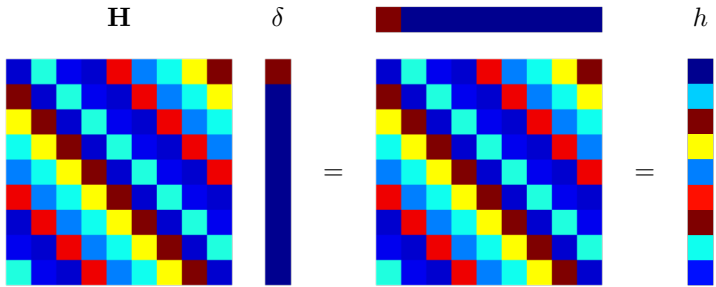
$h =$

$\mathbf{H} =$



Impulse Response (Finite-Length Signals)

- The 0-th column of the matrix $\mathbf{H}$ – the column vector h – has a special interpretation
- Compute the output when the input is a **delta function** (impulse): $\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$



- This suggests that we call h the **impulse response** of the system

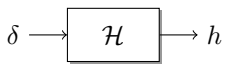
Impulse Response from Formulas (Finite-Length Signals)

- General formula for LTI matrix multiplication

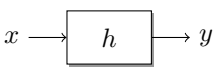
$$y[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- Let the input $x[n] = \delta[n]$ and compute

$$\sum_{m=0}^{N-1} h[(n-m)_N]\, \delta[m] \;=\; h[n] \;\checkmark$$

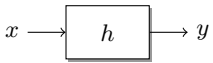


- The impulse response **characterizes** an LTI system
  (that is, carries all of the information contained in the matrix $\mathbf{H}$)



Summary

- LTI system $=$ multiplication by infinite-sized Toeplitz or $N \times N$ circulant matrix $\mathbf{H}$: $y = \mathbf{H}x$
- The **impulse response** h of an LTI system $=$ the response to an impulse δ
  - The impulse response is the 0-th column of the matrix $\mathbf{H}$
  - The impulse response characterizes an LTI system

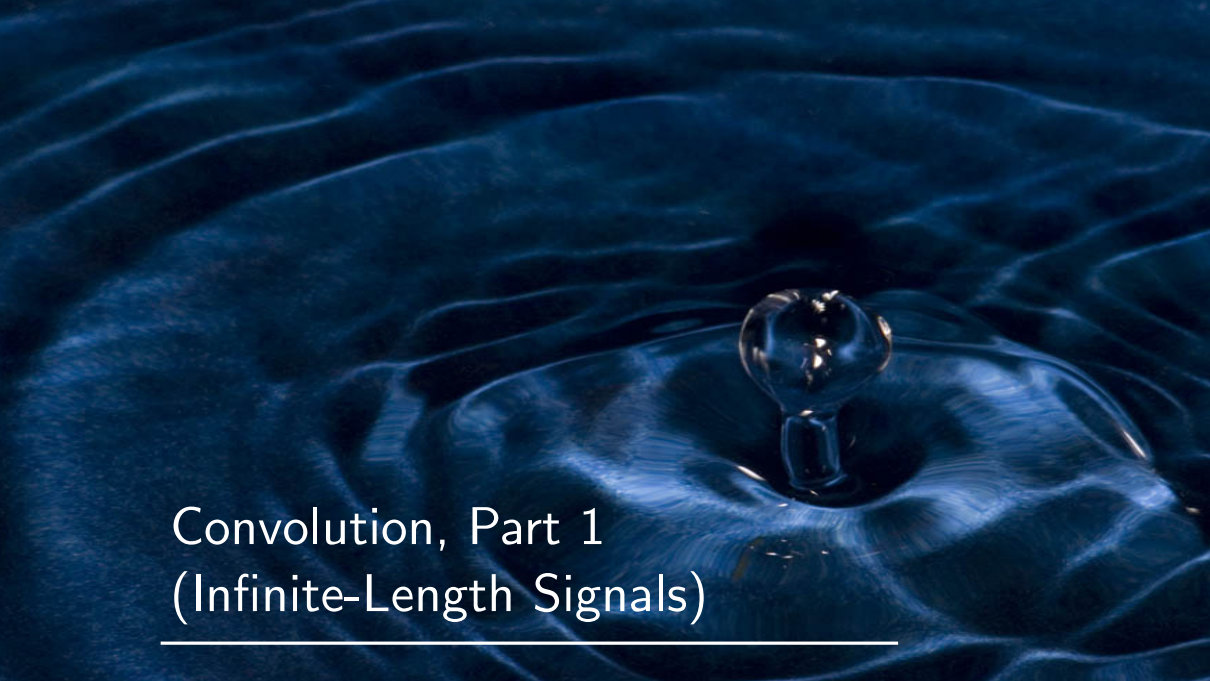


- Formula for the output signal y in terms of the input signal x and the impulse response h
  - Infinite-length signals

$$y[n] = \sum_{m=-\infty}^{\infty} h[n-m]\, x[m], \quad -\infty < n < \infty$$

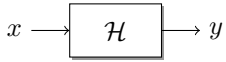
  - Length-N signals

$$y[n] = \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m], \quad 0 \le n \le N-1$$



Convolution, Part 1 (Infinite-Length Signals)

Three Ways to Compute the Output of an LTI System Given the Input



1. If $\mathcal{H}$ is defined in terms of a formula or **algorithm**, apply the input x and compute $y[n]$ at each time point $n \in \mathbb{Z}$
   - This is how systems are usually applied in computer code and hardware

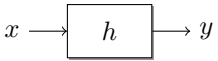
2. Find the impulse response h (by inputting $x[n] = \delta[n]$), form the **Toeplitz system matrix** $\mathbf{H}$, and multiply by the (infinite-length) input signal vector x to obtain $y = \mathbf{H}\,x$
   - This is not usually practical but is useful for conceptual purposes

3. Find the impulse response h and apply the formula for matrix/vector product for each $n \in \mathbb{Z}$

$$y[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\,x[m] \;=\; x[n] * h[n]$$

   - This is called **convolution** and is both conceptually and practically useful (Matlab command: `conv`)

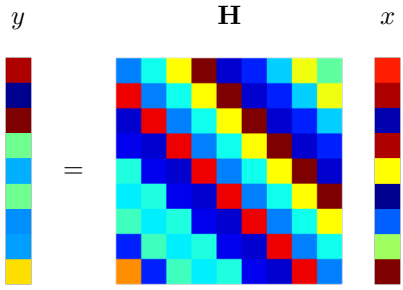
Convolution as a Sequence of Inner Products (1)



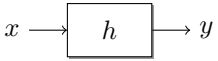
- Convolution formula

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- To compute the entry $y[n]$ in the output vector y:
  1. **Time reverse** the impulse response vector h and **shift** it n time steps to the right (delay)
  2. Compute the **inner product** between the shifted impulse response and the input vector x
- Repeat for every n



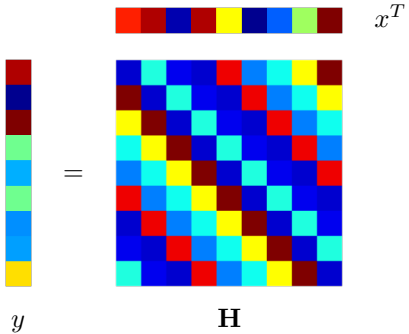
Convolution as a Sequence of Inner Products (2)



- Convolution formula

$$y[n] = x[n] * h[n] = \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- To compute the entry $y[n]$ in the output vector y:
  1. **Time reverse** the impulse response vector h and **shift** it n time steps to the right (delay)
  2. Compute the **inner product** between the shifted impulse response and the input vector x
- Repeat for every n



Recall: Row n of $\mathbf{H}$ is time-reversed impulse response shifted right by n

A Seven-Step Program for Computing Convolution By Hand

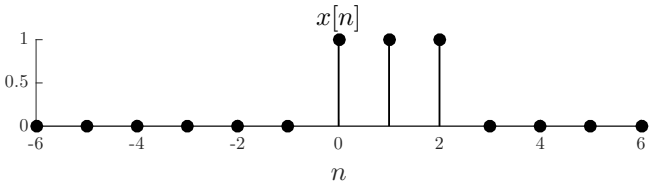
$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- **Step 1:** Decide which of x or h you will flip and shift; you have a choice since $x * h = h * x$
- **Step 2:** Plot $x[m]$ as a function of m
- **Step 3:** Plot the time-reversed impulse response $h[-m]$
- **Step 4:** To compute y at the time point n, plot the time-reversed impulse response after it has been shifted to the right (delayed) by n time units: $h[-(m-n)] = h[n-m]$
- **Step 5:** $y[n] =$ the inner product between the signals $x[m]$ and $h[n-m]$
  (Note: for complex signals, do not complex conjugate the second signal in the inner product)
- **Step 6:** Repeat for all n of interest (potentially all $n \in \mathbb{Z}$)
- **Step 7:** Plot $y[n]$ and perform a reality check to make sure your answer seems reasonable

First Convolution Example (1)

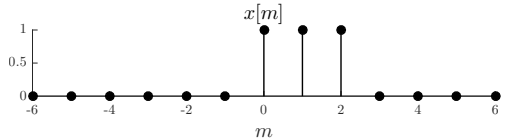
$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

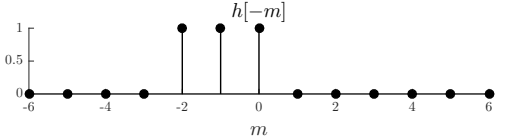
- Convolve a unit pulse with itself



First Convolution Example (2)

$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$





Convolution, Part 2 (Infinite-Length Signals)

A Seven-Step Program for Computing Convolution By Hand

$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

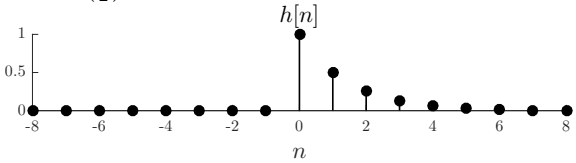
- **Step 1:** Decide which of x or h you will flip and shift; you have a choice since $x * h = h * x$
- **Step 2:** Plot $x[m]$ as a function of m
- **Step 3:** Plot the time-reversed impulse response $h[-m]$
- **Step 4:** To compute y at the time point n, plot the time-reversed impulse response after it has been shifted to the right (delayed) by n time units: $h[-(m-n)] = h[n-m]$
- **Step 5:** $y[n] =$ the inner product between the signals $x[m]$ and $h[n-m]$
  (Note: for complex signals, do not complex conjugate the second signal in the inner product)
- **Step 6:** Repeat for all n of interest (potentially all $n \in \mathbb{Z}$)
- **Step 7:** Plot $y[n]$ and perform a reality check to make sure your answer seems reasonable

Second Convolution Example (1)

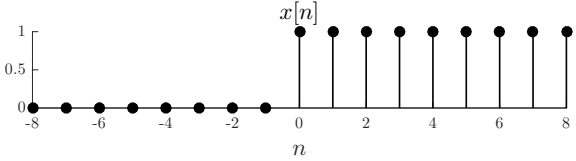
- Recall the **recursive average system**

$$y[n] = x[n] + \frac{1}{2}\, y[n-1]$$

and its impulse response $h[n] = \left(\frac{1}{2}\right)^n u[n]$

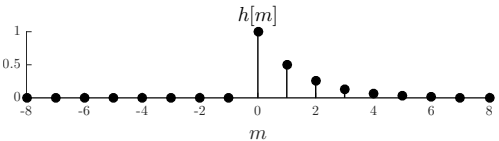


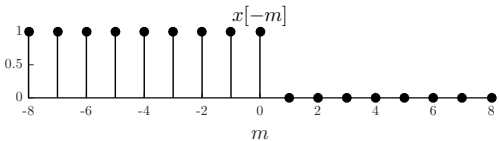
- Compute the output y when the input is a unit step $x[n] = u[n]$



Second Convolution Example (2)

$$y[n] \;=\; h[n] * x[n] \;=\; \sum_{m=-\infty}^{\infty} h[m]\, x[n-m]$$



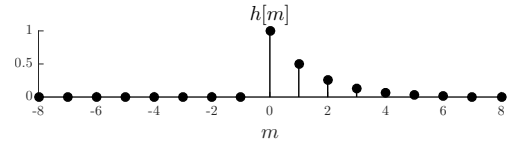


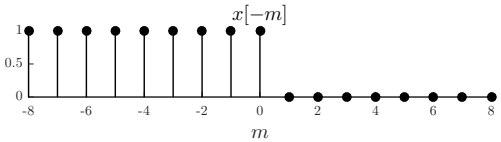
- Recall the super useful formula for the **finite geometric series**

$$\sum_{k=N_1}^{N_2} a^k \;=\; \frac{a^{N_1} - a^{N_2+1}}{1-a}, \quad N_1 \le N_2$$

Second Convolution Example (3)

$$y[n] \;=\; h[n] * x[n] \;=\; \sum_{m=-\infty}^{\infty} h[m]\, x[n-m]$$



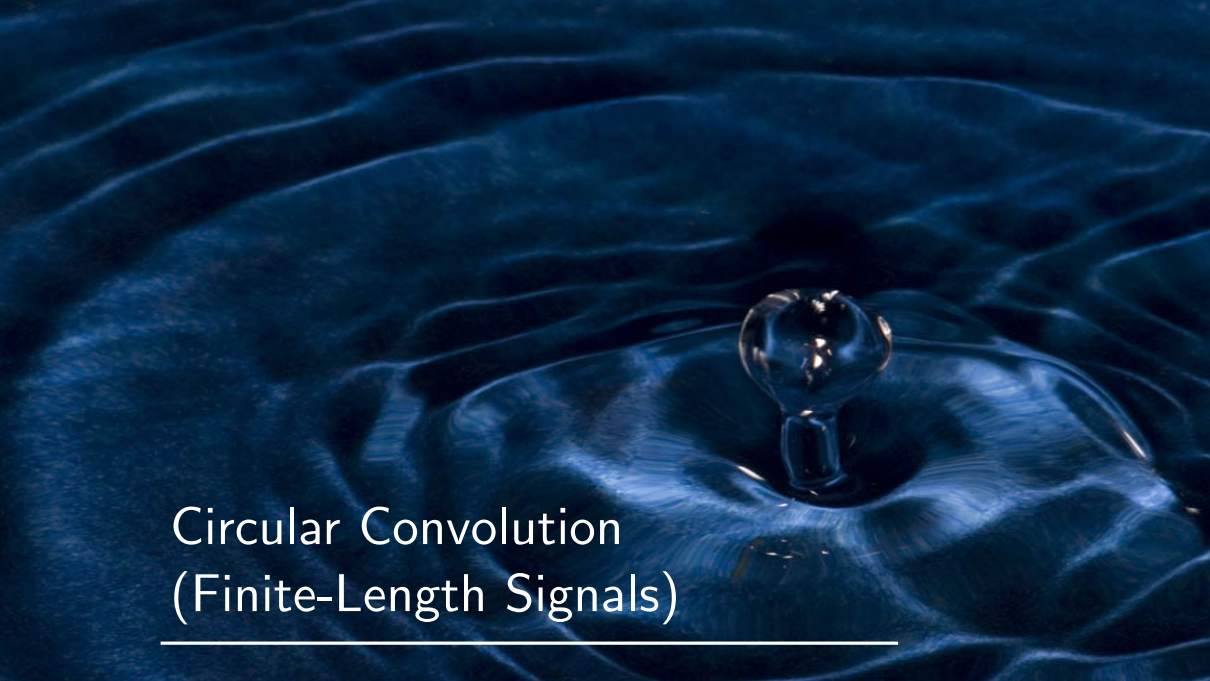


Summary

- **Convolution** formula for the output y of an LTI system given the input x and the impulse response h (infinite-length signals)

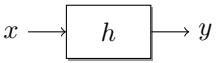
$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- Convolution is a sequence of inner products between the signal and the shifted, time-reversed impulse response
- Seven-step program for computing convolution by hand
- Check your work and compute large convolutions using Matlab command `conv`
- Practice makes perfect!



Circular Convolution (Finite-Length Signals)

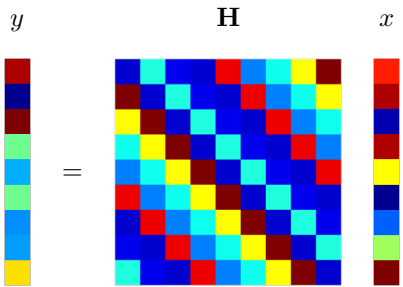
Circular Convolution as a Sequence of Inner Products (1)



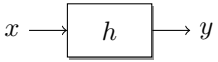
- Convolution formula

$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- To compute the entry $y[n]$ in the output vector y:
  1. **Circularly time reverse** the impulse response vector h and **circularly shift** it n time steps to the right (delay)
  2. Compute the **inner product** between the shifted impulse response and the input vector x
- Repeat for every n



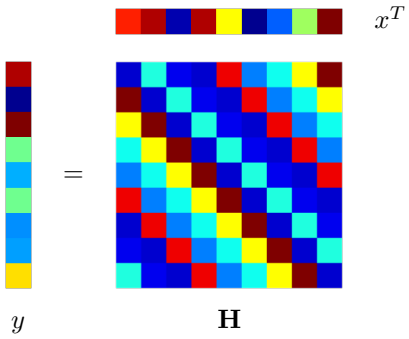
Circular Convolution as a Sequence of Inner Products (2)



- Convolution formula

$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- To compute the entry $y[n]$ in the output vector y:
  1. **Circularly time reverse** the impulse response vector h and **circularly shift** it n time steps to the right (delay)
  2. Compute the **inner product** between the shifted impulse response and the input vector x
- Repeat for every n



Recall: Row n of $\mathbf{H}$ is circularly time-reversed impulse response circularly shifted right by n

A Seven-Step Program for Computing Circular Convolution By Hand

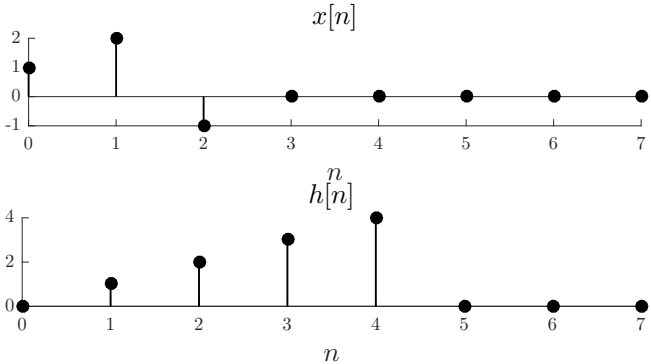
$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- **Step 1:** Decide which of x or h you will flip and shift; you have a choice since $x * h = h * x$
- **Step 2:** Plot $x[m]$ as a function of m on a wheel with N time locations
- **Step 3:** Plot the circularly time-reversed impulse response $h[(-m)_N]$ on a clock with N time locations
- **Step 4:** To compute y at the time point n, plot the time-reversed impulse response after it has been shifted counter-clockwise (delayed) by n time units: $h[(-(m-n))_N] = h[(n-m)_N]$
- **Step 5:** $y[n] =$ the inner product between the signals $x[m]$ and $h[(n-m)_N]$
  (Note: for complex signals, do not complex conjugate the second signal in the inner product)
- **Step 6:** Repeat for all $n = 0, 1, \ldots, N-1$
- **Step 7:** Plot $y[n]$ and perform a reality check to make sure your answer seems reasonable

Circular Convolution Example

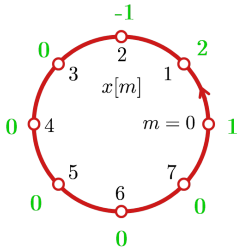
$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

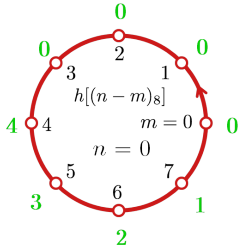
- For $N = 8$, circularly convolve x and h

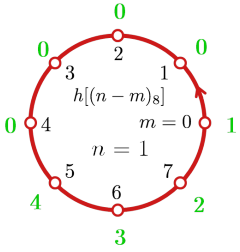


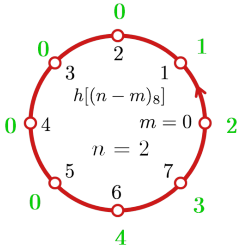
Circular Convolution Example

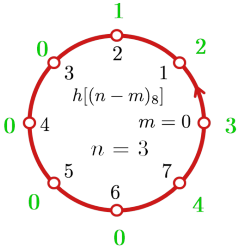
$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$





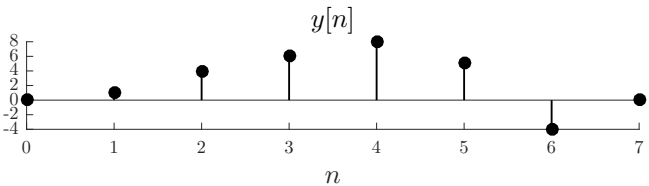






Circular Convolution Example

$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

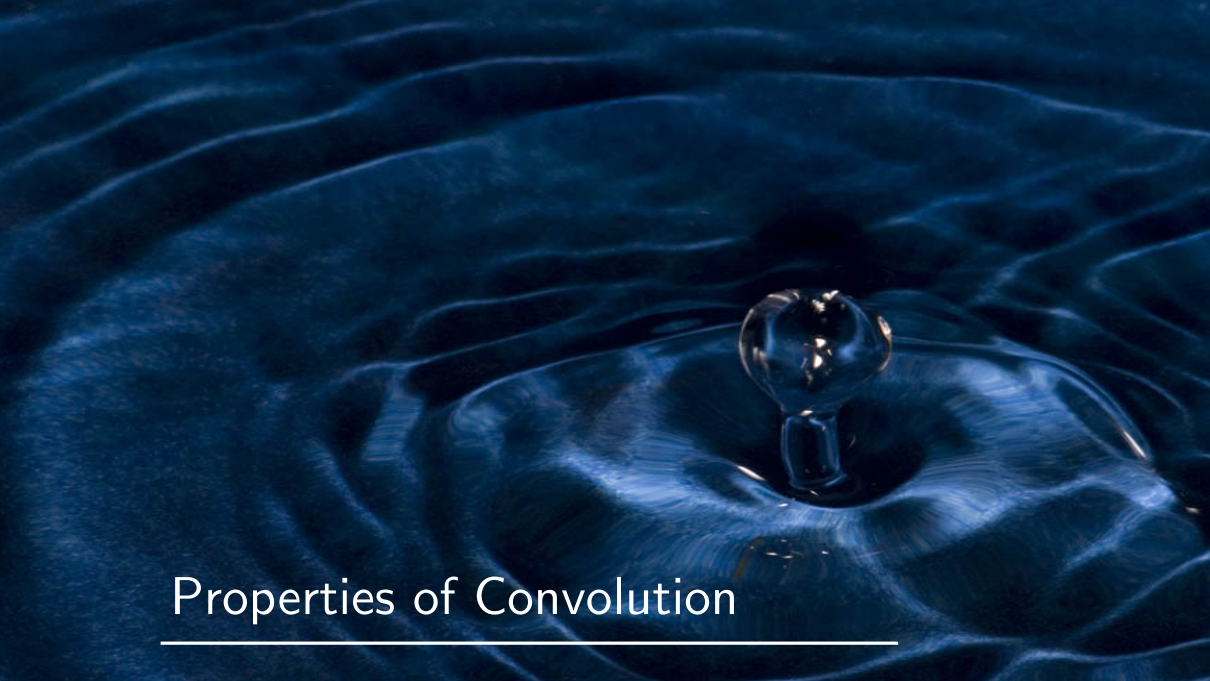


Summary

- **Circular convolution** formula for the output y of an LTI system given the input x and the impulse response h (length-N signals)

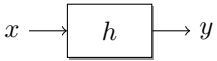
$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- Circular convolution is a sequence of inner products between the signal and the circularly shifted, time-reversed impulse response
- Seven-step program for computing circular convolution by hand
- Check your work and compute large circular convolutions using Matlab command `cconv`
- Practice makes perfect!



Properties of Convolution

Properties of Convolution



- Input signal x, LTI system impulse response h, and output signal y are related by the **convolution**
  - Infinite-length signals

$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m], \quad -\infty < n < \infty$$

  - Length-N signals

$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m], \quad 0 \le n \le N-1$$

- Thanks to the Toeplitz/circulant structure of LTI systems, convolution has very special properties
- We will emphasize infinite-length convolution, but similar arguments hold for circular convolution except where noted

Convolution is Commutative

- **Fact:** Convolution is commutative: $x * h \;=\; h * x$
- These block diagrams are equivalent:

- Enables us to pick either h or x to flip and shift (or stack into a matrix) when convolving
- To prove, start with the convolution formula

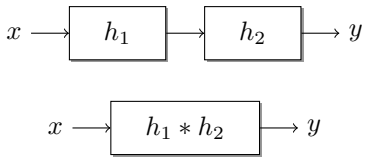
$$y[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\,x[m] \;=\; x[n] * h[n]$$

and change variables to $k = n - m \;\Rightarrow\; m = n - k$

$$y[n] \;=\; \sum_{k=-\infty}^{\infty} h[k]\,x[n-k] \;=\; h[n] * x[n] \;\checkmark$$

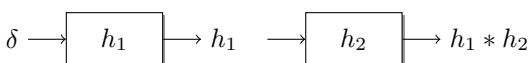
Cascade Connection of LTI Systems

- Impulse response of the **cascade** (aka series connection) of two LTI systems: $y = \mathbf{H}_1\mathbf{H}_2\, x$



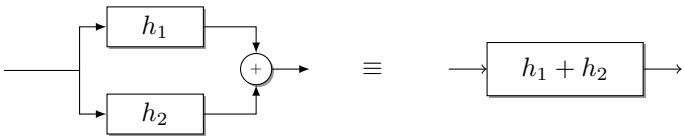
- Interpretation: The product of two Toeplitz/circulant matrices is a Toeplitz/circulant matrix

- Easy proof by picture; find impulse response the old school way



Parallel Connection of LTI Systems

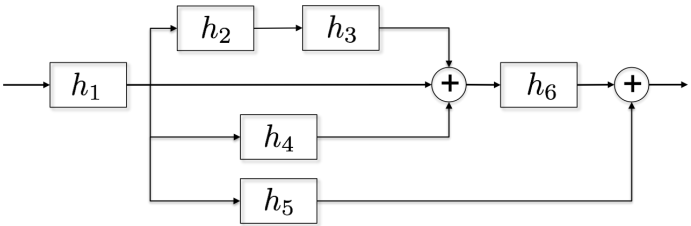
- Impulse response of the **parallel connection** of two LTI systems $y = (\mathbf{H}_1 + \mathbf{H}_2)\, x$

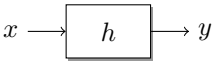


- Proof is an easy application of the linearity of an LTI system

Example: Impulse Response of a Complicated Connection of LTI Systems

- Compute the overall effective impulse response of the following system





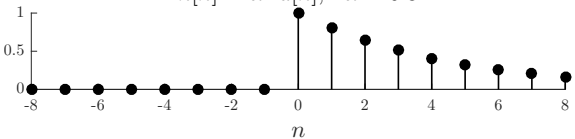
Causal Systems

DEFINITION

A system $\mathcal{H}$ is **causal** if the output $y[n]$ at time n depends only the input $x[m]$ for times $m \leq n$. In words, causal systems do not look into the future

- **Fact:** An LTI system is causal if its impulse response is causal: $h[n] = 0$ for $n < 0$

$$h[n] = \alpha^n \, u[n], \quad \alpha = 0.8$$



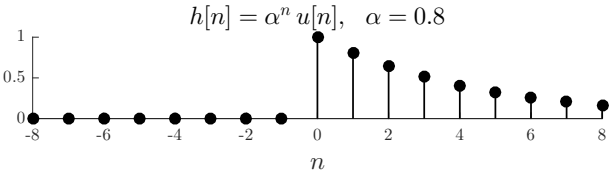
- To prove, note that the convolution

$$y[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

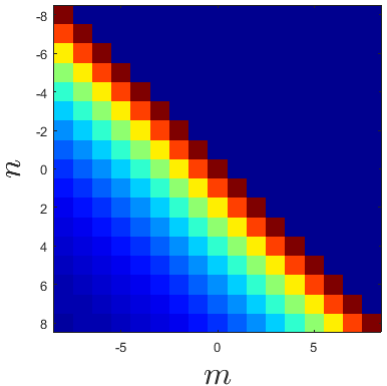
does not look into the future if $h[n-m] = 0$ when $m > n$; equivalently, $h[n'] = 0$ when $n' < 0$

Causal System Matrix

- **Fact:** An LTI system is causal if its impulse response is causal: $h[n] = 0$ for $n < 0$



- Toeplitz system matrix is **lower triangular**

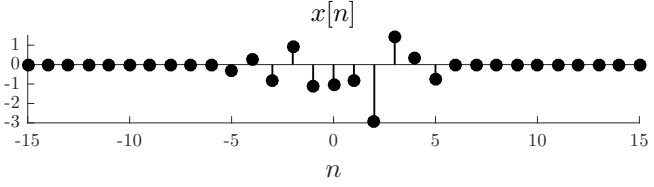


Duration of Convolution

DEFINITION

The signal x has **support interval** $[N_1, N_2]$, $N_1 \leq N_2$, if $x[n] = 0$ for all $n < N_1$ and $n > N_2$. The **duration** D_x of x equals $N_2 - N_1 + 1$

- Example: A signal with support interval $[-5, 5]$ and duration 11 samples



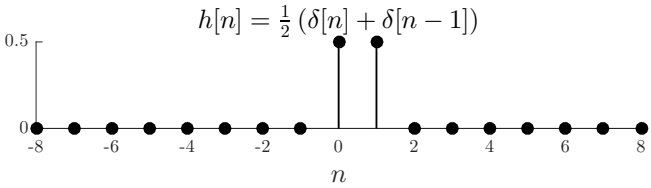
- **Fact:** If x has duration D_x samples and h has duration D_h samples, then the convolution $y = x * h$ has duration at most $D_x + D_h - 1$ samples (proof by picture is simple)

Duration of Impulse Response – FIR

DEFINITION

An LTI system has a **finite impulse response** (FIR) if the duration of its impulse response h is finite

- Example: Moving average system $\quad y[n] = \mathcal{H}\{x[n]\} = \frac{1}{2}\left(x[n] + x[n-1]\right)$

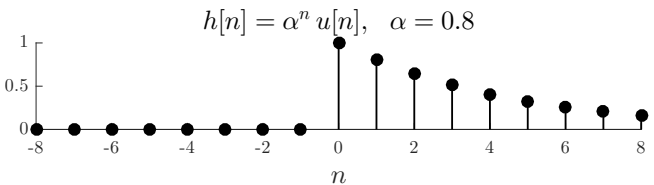


Duration of Impulse Response – IIR

DEFINITION

An LTI system has an **infinite impulse response** (IIR) if the duration of its impulse response h is infinite

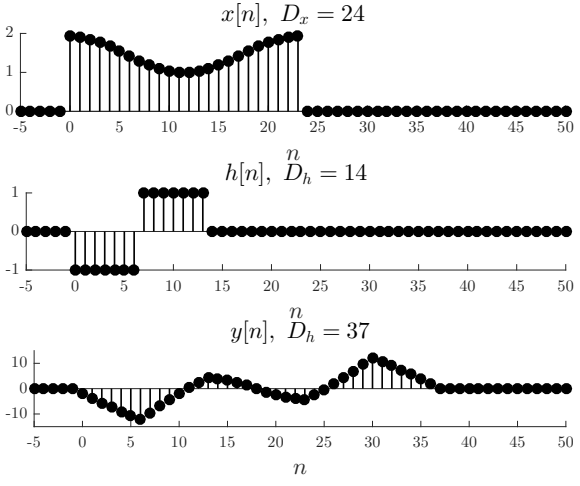
- Example: Recursive average system $\quad y[n] \;=\; \mathcal{H}\{x[n]\} \;=\; x[n] + \alpha\, y[n-1]$



- Note: Obviously the FIR/IIR distinction applies only to infinite-length signals

Duration of Convolution

- Recall that, if x has duration D_x samples and h has duration D_h samples, then the infinite-length convolution $y = x * h$ has duration at most $D_x + D_h - 1$ samples (proof by picture is simple)



Implementing Infinite-Length Convolution with Circular Convolution

- Consider two infinite-length signals: x has duration D_x samples and h has duration D_h samples, $D_x, D_h < \infty$

- Recall that their infinite-length convolution $y = x * h$ has duration at most $D_x + D_h - 1$ samples

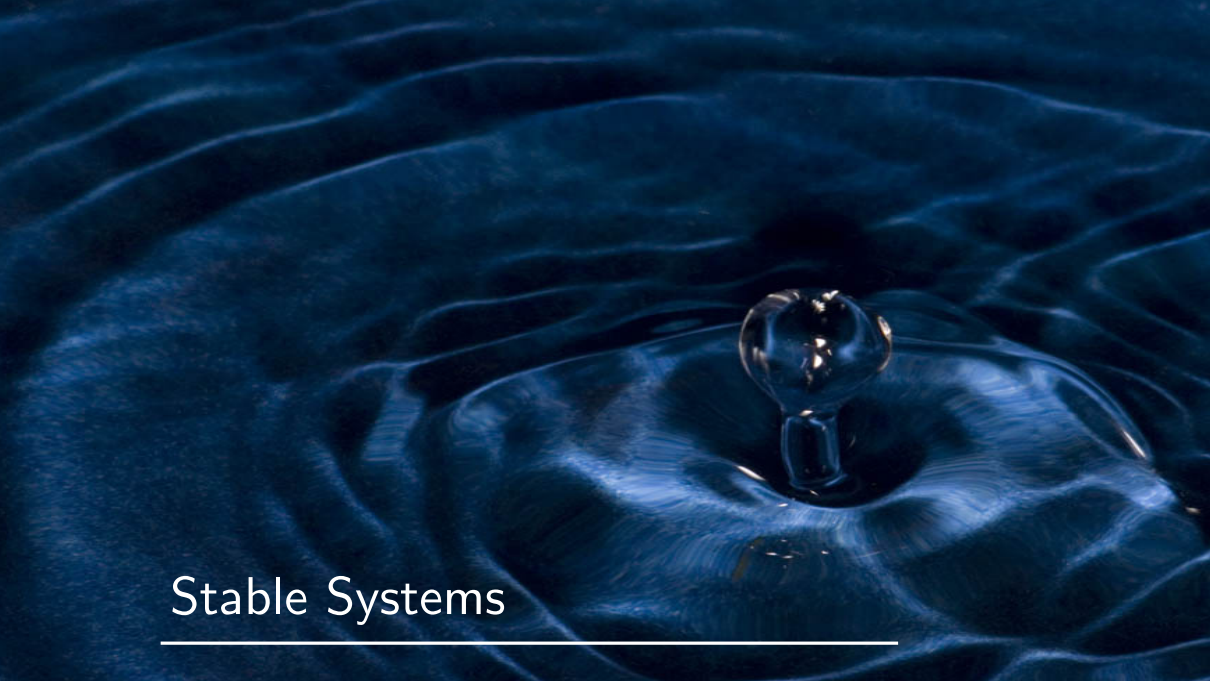
- Armed with this fact, we can implement infinite-length convolution using circular convolution
  1. Extract the D_x-sample support interval of x and zero pad so that the resulting signal x' is of length $D_x + D_h - 1$
  2. Perform the same operations on h to obtain h'
  3. Circularly convolve $x' \circledast h'$ to obtain y'

- **Fact:** The values of the signal y' will coincide with those of the infinite-length convolution $y = x * h$ within its support interval

- How does it work? The zero padding effectively converts circular shifts (finite-length signals) into regular shifts (infinite-length signals) (Easy to try out in Matlab!)

Summary

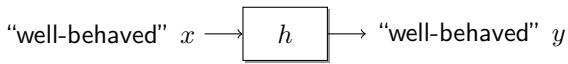
- Convolution has very special and beautiful properties
- Convolution is commutative
- Convolutions (LTI systems) can be connected in cascade and parallel
- An LTI system is causal if its impulse response is causal
- LTI systems are either FIR or IIR
- Can implement infinite-length convolution using circular convolution when the signals have finite duration (important later for "fast convolution" using the FFT)



Stable Systems

Stable Systems (1)

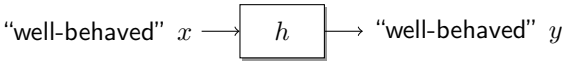
- With a **stable** system, a "well-behaved" input always produces a "well-behaved" output



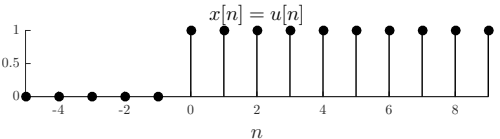
- Stability is essential to ensuring the proper and safe operation of myriad systems
  - Steering systems
  - Braking systems
  - Robotic navigation
  - Modern aircraft
  - International Space Station
  - Internet IP packet communication (TCP) ...

Stable Systems (2)

- With a **stable** system, a "well-behaved" input always produces a "well-behaved" output

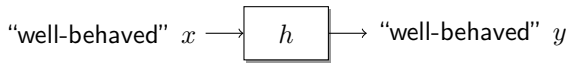


- Example: Recall the recursive average system $y[n] = \mathcal{H}\{x[n]\} = x[n] + \alpha\, y[n-1]$
  Consider a step function input $x[n] = u[n]$



Well-Behaved Signals

- With a **stable** system, a "well-behaved" input always produces a "well-behaved" output



- How to measure how "well-behaved" a signal is? Different measures give different notions of stability

- One reasonable measure: A signal x is well behaved if it is **bounded** (recall that $\sup$ is like $\max$)

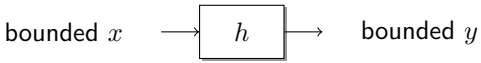
$$\|x\|_\infty \;=\; \sup_n |x[n]| \;<\; \infty$$

Bounded-Input Bounded-Output (BIBO) Stability

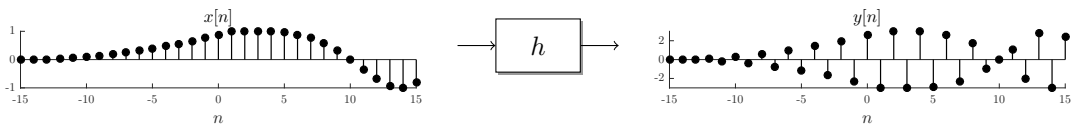
BIBO Stability (1)

DEFINITION

An LTI system is **bounded-input bounded-output (BIBO) stable** if a bounded input x always produces a bounded output y



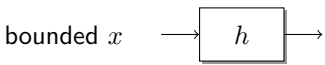
- Bounded input and output means $\|x\|_\infty < \infty$ and $\|y\|_\infty < \infty$,
  or that there exist constants $A, C < \infty$ such that $|x[n]| < A$ and $|y[n]| < C$ for all n



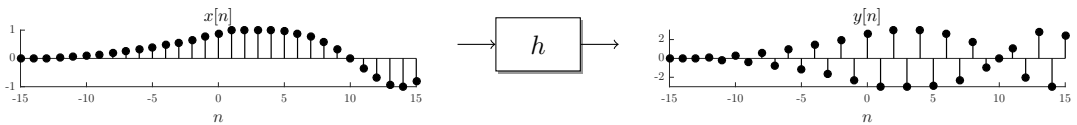
BIBO Stability (2)

DEFINITION

An LTI system is **bounded-input bounded-output (BIBO) stable** if a bounded input x always produces a bounded output y



- Bounded input and output means $\|x\|_\infty < \infty$ and $\|y\|_\infty < \infty$



- **Fact:** An LTI system with impulse response h is BIBO stable if and only if

$$\|h\|_1 = \sum_{n=-\infty}^{\infty} |h[n]| < \infty$$

BIBO Stability – Sufficient Condition

- Prove that if $\|h\|_1 < \infty$ then the system is BIBO stable – for any input $\|x\|_\infty < \infty$ the output $\|y\|_\infty < \infty$
- Recall that $\|x\|_\infty < \infty$ means there exist a constant A such that $|x[n]| < A < \infty$ for all n
- Let $\|h\|_1 = \sum_{n=-\infty}^{\infty} |h[n]| = B < \infty$
- Compute a bound on $|y[n]|$ using the convolution of x and h and the bounds A and B

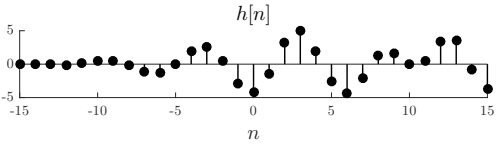
$$
\begin{aligned}
|y[n]| &= \left| \sum_{m=-\infty}^{\infty} h[n-m]\, x[m] \right| \leq \sum_{m=-\infty}^{\infty} |h[n-m]|\, |x[m]| \\
&< \sum_{m=-\infty}^{\infty} |h[n-m]|\, A = A \sum_{k=-\infty}^{\infty} |h[k]| = A\, B = C < \infty
\end{aligned}
$$

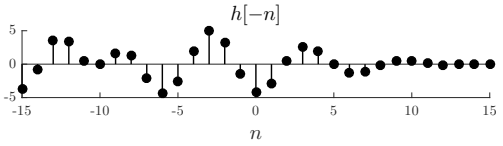
- Since $|y[n]| < C < \infty$ for all n, $\|y\|_\infty < \infty$ ✓

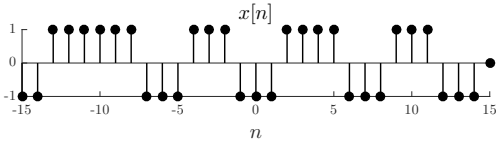
BIBO Stability – Necessary Condition (1)

- Prove that if $\|h\|_1 = \infty$ then the system is not BIBO stable – there exists an input $\|x\|_\infty < \infty$ such that the output $\|y\|_\infty = \infty$
  - Assume that x and h are real-valued; the proof for complex-valued signals is nearly identical

- Given an impulse response h with $\|h\|_1 = \infty$, form the tricky special signal $x[n] = \mathrm{sgn}(h[-n])$
  - $x[n]$ is the $\pm$ sign of the time-reversed impulse response $h[-n]$
  - Note that x is bounded: $|x[n]| \leq 1$ for all n







BIBO Stability – Necessary Condition (2)

- We are proving that that if $\|h\|_1 = \infty$ then the system is not BIBO stable – there exists an input $\|x\|_\infty < \infty$ such that the output $\|y\|_\infty = \infty$

- Armed with the tricky special signal x, compute the output $y[n]$ at the time point $n = 0$

$$\begin{aligned} y[0] &= \sum_{m=-\infty}^{\infty} h[0-m]\,x[m] = \sum_{m=-\infty}^{\infty} h[-m]\,\mathrm{sgn}(h[-m]) \\ &= \sum_{m=-\infty}^{\infty} |h[-m]| = \sum_{k=-\infty}^{\infty} |h[k]| = \infty \end{aligned}$$

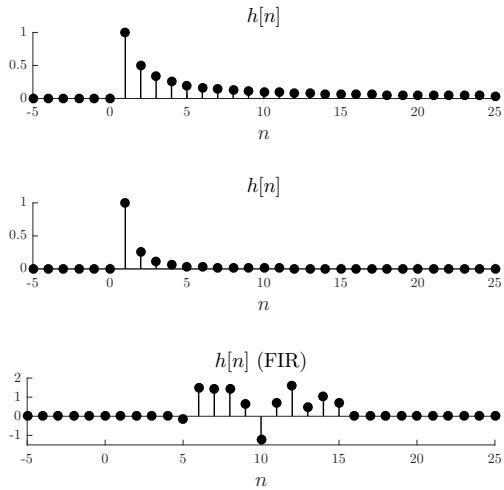
- So, even though x was bounded, y is not bounded; so system is not BIBO stable

BIBO System Examples (1)

- Absolute summability of the impulse response h determines whether an LTI systems is BIBO stable or not
- Example: $h[n] = \begin{cases} \frac{1}{n} & n \geq 1 \\ 0 & \text{otherwise} \end{cases}$

  $\|h\|_1 = \sum_{n=1}^{\infty} \left|\frac{1}{n}\right| = \infty \;\Rightarrow\;$ not BIBO
- Example: $h[n] = \begin{cases} \frac{1}{n^2} & n \geq 1 \\ 0 & \text{otherwise} \end{cases}$

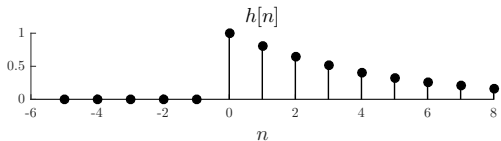
  $\|h\|_1 = \sum_{n=1}^{\infty} \left|\frac{1}{n^2}\right| = \frac{\pi^2}{6} \;\Rightarrow\;$ BIBO
- Example: h FIR $\;\Rightarrow\;$ BIBO



BIBO System Examples (2)

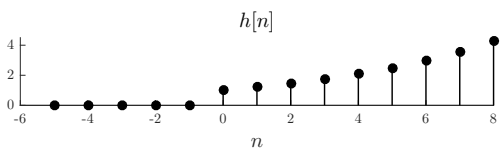
- Example: Recall the recursive average system $y[n] = \mathcal{H}\{x[n]\} = x[n] + \alpha\, y[n-1]$
- Impulse response: $h[n] = \alpha^n\, u[n]$
- For $|\alpha| < 1$

$\|h\|_1 = \sum_{n=0}^{\infty} |\alpha|^n = \frac{1}{1-|\alpha|} < \infty \;\Rightarrow$ BIBO



- For $|\alpha| > 1$

$\|h\|_1 = \sum_{n=0}^{\infty} |\alpha|^n = \infty \;\Rightarrow$ not BIBO

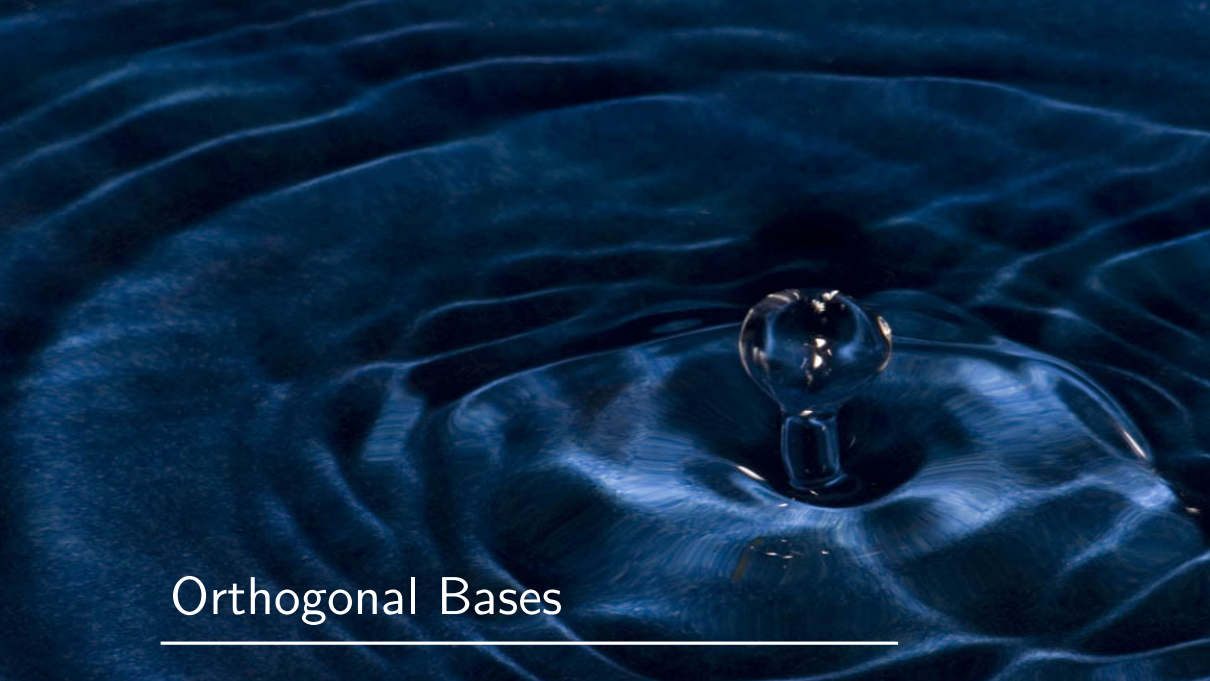


Summary

- Signal processing applications typically dictate that the system be **stable**, meaning that "well-behaved inputs" produce "well-behaved outputs"
- Measure "well-behavedness" of a signal using the ∞-norm (bounded signal)
- BIBO stability: bounded inputs always produce bounded outputs iff the impulse response h is such that $\|h\|_1 < \infty$
- When a system is not BIBO stable, all hope is not lost; unstable systems can often by **stabilized** using **feedback** (more on this later)

Orthogonal Bases and the DFT

Set E



Orthogonal Bases

Transforms and Orthogonal Bases

- We now turn back to linear algebra to understand **transforms**, which map signals between different "domains"
- Recall that signals can be interpreted as vectors in a vector space (linear algebra)
- We now review the concept of a **basis** for a vector space
- As we will see, different signal transforms (and "domains") correspond to different bases
- Caveat: This is not a course on linear algebra!

Basis

DEFINITION

A **basis** $\{b_k\}$ for a vector space V is a collection of vectors from V that are linearly independent and span V

- **Span:** All vectors in V can be represented as a linear combination of the basis vectors $\{b_k\}_{k=0}^{N-1}$

$$x \;=\; \sum_{k=0}^{N-1} \alpha_k \, b_k \;=\; \alpha_0 \, b_0 + \alpha_1 \, b_1 + \cdots + \alpha_{N-1} \, b_{N-1} \quad \forall \, x \in V$$

- **Linearly independent:** None of the basis vectors can be represented as a linear combination of the other basis vectors
- **Dimension of V:** The number of vectors in the basis (N in the above)
- **Fact:** The dimension of $\mathbb{R}^N$ and $\mathbb{C}^N$ equals N (we will focus on these spaces)

Basis Matrix

- Stack the basis vectors b_k as columns into the $N \times N$ **basis matrix**

$$\mathbf{B} \;=\; \left[b_0|b_1|\cdots|b_{N-1}\right]$$

- Stack the scalar weights α_k into an $N \times 1$ column vector

$$a = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{bmatrix}$$

- We can now write a linear combination of basis elements as the matrix/vector product

$$x \;=\; \alpha_0 b_0 + \alpha_1 b_1 + \cdots + \alpha_{N-1} b_{N-1} \;=\; \sum_{k=0}^{N-1} \alpha_k b_k \;=\; \left[b_0|b_1|\cdots|b_{N-1}\right] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{bmatrix} \;=\; \mathbf{B}\, a$$

Orthogonal and Orthonormal Bases

DEFINITION

An **orthogonal basis** $\{b_k\}_{k=0}^{N-1}$ for a vector space V is a basis whose elements are mutually orthogonal

$$\langle b_k, b_l \rangle \;=\; 0, \quad k \neq l$$

DEFINITION

An **orthonormal basis** $\{b_k\}_{k=0}^{N-1}$ for a vector space V is a basis whose elements are mutually orthogonal and normalized (in the 2-norm)

$$\langle b_k, b_l \rangle \;=\; 0, \quad k \neq l$$

$$\|b_k\|_2 \;=\; 1$$

Example: Orthogonal and Orthonormal Bases in $\mathbb{R}^2$

$$b_0 = \begin{bmatrix} \quad \\ \quad \end{bmatrix}, \quad b_1 = \begin{bmatrix} \quad \\ \quad \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}$$

Inverse of a Matrix

DEFINITION

The **inverse** of a square matrix $\mathbf{A}$ is a matrix $\mathbf{A}^{-1}$ such that

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

where $\mathbf{I}$ is the identity matrix of the same size as $\mathbf{A}$

- If $\mathbf{B}$ is a basis matrix and $x = \mathbf{B}a$, then $a = \mathbf{B}^{-1}x$

Inverse of an Orthonormal Basis Matrix

- When the basis matrix $\mathbf{B}$ contains an **orthonormal basis** $\{b_k\}_{k=0}^{N-1}$, its inverse $\mathbf{B}^{-1}$ is trivial to calculate

- **Fact:** $\mathbf{B}^{-1} = \mathbf{B}^H$ (recall ^{H} is complex conjugate transpose)

- Terminology: $\mathbf{B}$ is a **unitary** matrix (aka complex-valued orthogonal)

- To prove, write out $\mathbf{B}^H\mathbf{B}$ and use the fact that the columns of $\mathbf{B}$ are orthonormal

$$\mathbf{B}^H\mathbf{B} = \begin{bmatrix} b_0^H \\ b_1^H \\ \vdots \\ b_{N-1}^H \end{bmatrix} \begin{bmatrix} b_0 | b_1 | \cdots | b_{N-1} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

Signal Representation by Orthonormal Basis

- Given an **orthonormal basis** $\{b_k\}_{k=0}^{N-1}$ and orthonormal basis matrix $\mathbf{B}$, we have the following **signal representation** for any signal x

$$x \;=\; \mathbf{B}\,a \;=\; \sum_{k=0}^{N-1} \alpha_k\, b_k \qquad \text{(synthesis)}$$

$$a \;=\; \mathbf{B}^H x \quad \text{or, each} \quad \alpha_k \;=\; \langle x, b_k \rangle \qquad \text{(analysis)}$$

- **Synthesis:** Build up the signal x as a linear combination of the basis elements b_k weighted by the weights α_k

- **Analysis:** Compute the weights α_k such that the synthesis produces x; the weight α_k measures the similarity between x and the basis element b_k

Summary

- **Orthonormal bases** make life easy

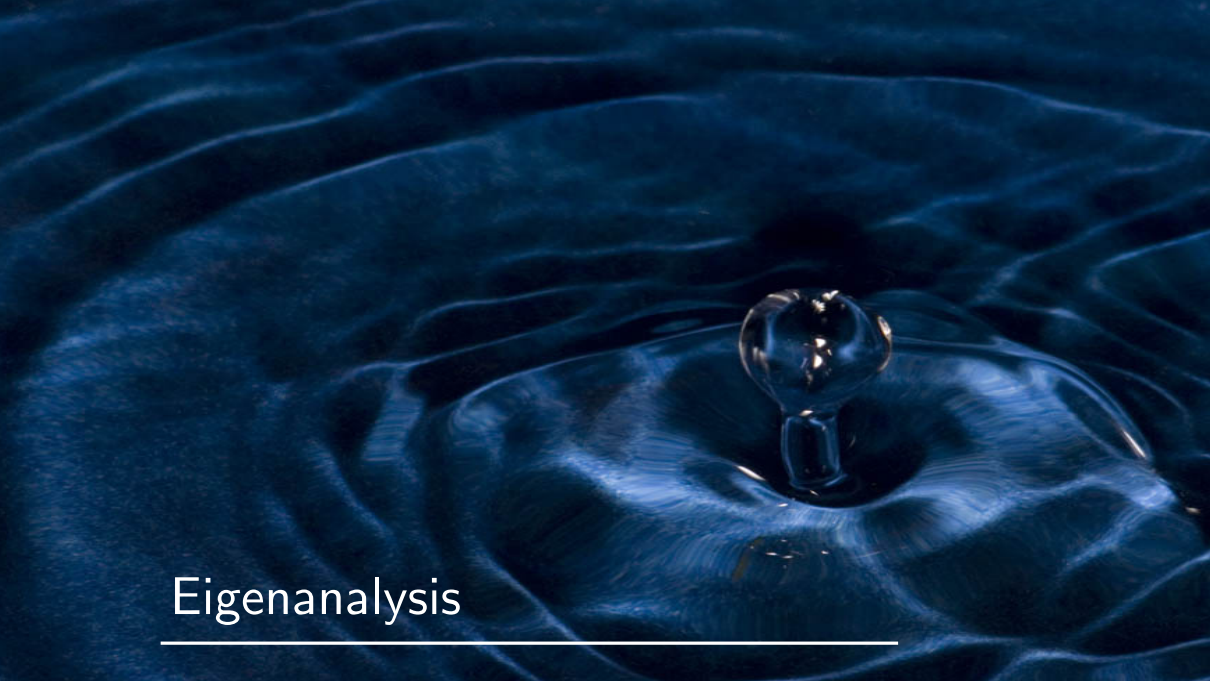
- Given an **orthonormal basis** $\{b_k\}_{k=0}^{N-1}$ and orthonormal basis matrix $\mathbf{B}$, we have the following **signal representation** for any signal x

$$x \;=\; \mathbf{B}\,a \;=\; \sum_{k=0}^{N-1} \alpha_k\, b_k \qquad \text{(synthesis)}$$

$$a \;=\; \mathbf{B}^H x \quad \text{or, each} \quad \alpha_k \;=\; \langle x, b_k \rangle \qquad \text{(analysis)}$$

- In signal processing, we say that the vector a is the **transform** of the signal x with respect to the orthonormal basis $\{b_k\}_{k=0}^{N-1}$

- Clearly the transform a contains all of the information in the signal x (and vice versa)



Eigenanalysis

Eigenanalysis

- We continue borrowing from linear algebra by recalling the **eigenvectors** and **eigenvalues** of a matrix
- Applying this point of view to circulant matrices (LTI systems for finite-length signals) will lead to an amazing result that ties together many threads of thought
- Caveat: This is not a course on linear algebra!

Eigenvectors and Eigenvalues

DEFINITION

Given a square matrix $\mathbf{A}$, the vector v is an **eigenvector** with **eigenvalue** λ if

$$\mathbf{A}\,v \;=\; \lambda\,v$$

- Geometric intuition: Multiplying an eigenvector v by the matrix $\mathbf{A}$ does not change its direction; it changes only its strength by the factor $\lambda \in \mathbb{C}$
- Example in $\mathbb{R}^2$:

$$\mathbf{A} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}, \quad v = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \lambda = 2$$

$$\mathbf{A}v = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \end{bmatrix} = 2v$$

Eigendecomposition

- An $N \times N$ matrix $\mathbf{A}$ has N eigenvectors and N eigenvalues (not necessarily distinct, though)
- Stack the N eigenvectors $\{v_m\}_{m=0}^{N-1}$ as columns into an $N \times N$ matrix

$$\mathbf{V} = \left[v_0 | v_1 | \cdots | v_{N-1}\right]$$

- Place the N eigenvalues $\{\lambda_m\}_{m=0}^{N-1}$ on the diagonal of an $N \times N$ diagonal matrix

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{N-1} \end{bmatrix}$$

- Then we can write

$$\mathbf{AV} = \mathbf{V\Lambda}$$

Diagonalization

- Recall the eigendecomposition of a matrix $\mathbf{A}$

$$\mathbf{AV} = \mathbf{V\Lambda}$$

- When the eigenvector matrix $\mathbf{V}$ is **invertible**, we can multiply both sides of the eigendecomposition on the left by $\mathbf{V}^{-1}$ to obtain

$$\mathbf{V}^{-1}\mathbf{AV} = \mathbf{V}^{-1}\mathbf{V\Lambda} = \mathbf{I\Lambda} = \mathbf{\Lambda}$$

- We say that the eigenvector matrix $\mathbf{V}$ **diagonalizes** the matrix $\mathbf{A}$

$$\mathbf{V}^{-1}\mathbf{AV} = \mathbf{\Lambda}$$

- Much easier to multiply a vector by $\mathbf{\Lambda}$ than by $\mathbf{A}$! (We simply scale each entry)

- Clearly $\mathbf{V}$ and $\mathbf{\Lambda}$ contain all of the information in $\mathbf{A}$ (and vice versa)

Aside: Diagonalization and Normal Matrices

- A matrix $\mathbf{A}$ is **normal** if $AA^T = A^TA$
- Given a normal matrix $\mathbf{A}$, if its eigenvector matrix $\mathbf{V}$ is invertible, then it is also **orthogonal**
- **Circulant matrices** are normal
- Recall that the system matrix $\mathbf{H}$ of a finite-length LTI system is circulant, and therefore also normal.
- Thus the eigenvector matrix $\mathbf{V}$ of a circulent matrix $\mathbf{H}$ is orthogonal

Summary

- Multiplying an eigenvector v by the matrix $\mathbf{A}$ does not change its direction; it changes only its strength by the factor λ

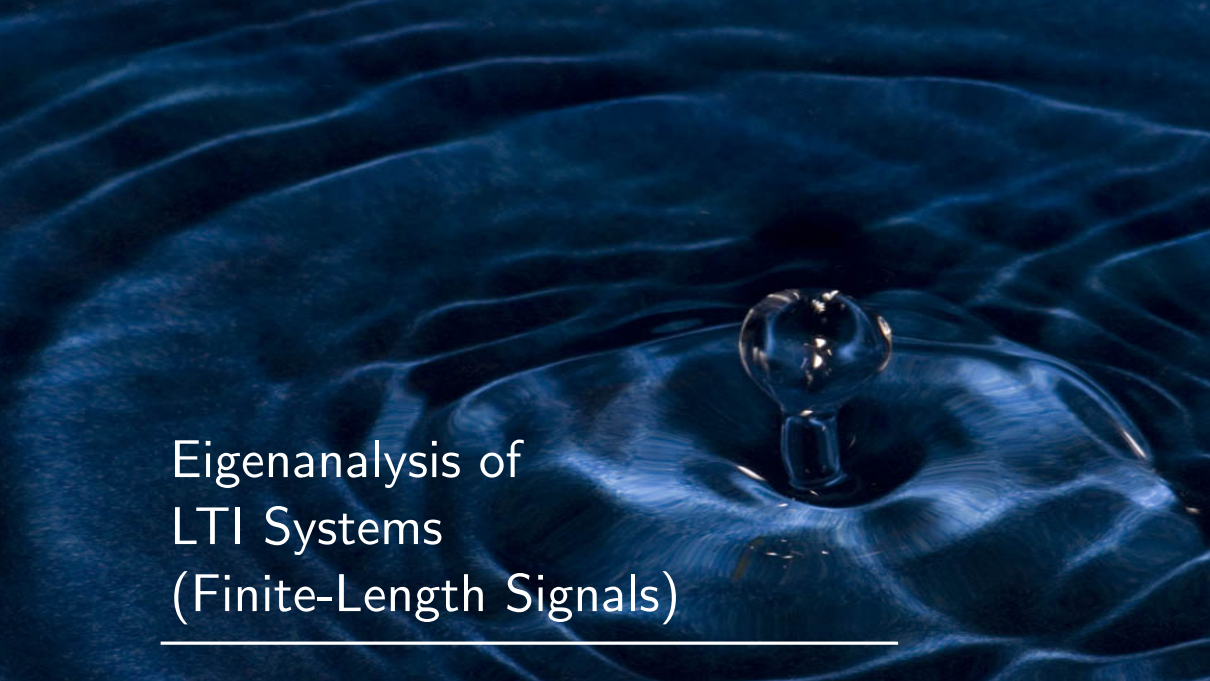
- The eigenvectors/values contain all of the information in the matrix $\mathbf{A}$ (and vice versa)

- Diagonalization by eigendecomposition

$$\mathbf{V}^{-1}\mathbf{A}\mathbf{V} = \mathbf{\Lambda}$$

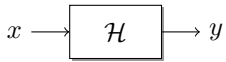
or, equivalently,

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$



Eigenanalysis of LTI Systems (Finite-Length Signals)

LTI Systems for Finite-Length Signals



$$y \;=\; \mathbf{H}x$$

- For length-N signals, $\mathbf{H}$ is an $N \times N$ **circulant matrix** with entries

$$[\mathbf{H}]_{n,m} \;=\; h[(n-m)_N]$$

  where h is the **impulse response**

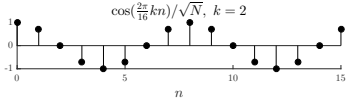
- **Goal:** Calculate the **eigenvectors** and **eigenvalues** of $\mathbf{H}$

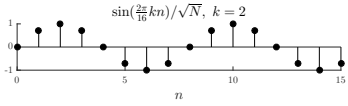
- Eigenvectors v are input signals that emerge at the system output unchanged (except for a scaling by the eigenvalue λ) and so are somehow "fundamental" to the system

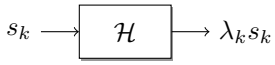
Eigenvectors of LTI Systems

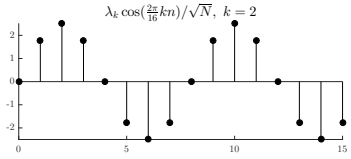
- **Fact:** The eigenvectors of a circulant matrix (LTI system) are the complex **harmonic sinusoids**

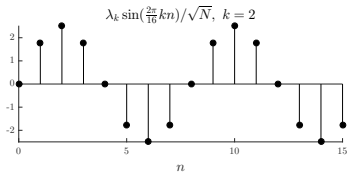
$$s_k[n] \;=\; \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} \;=\; \frac{1}{\sqrt{N}}\left(\cos\left(\frac{2\pi}{N}kn\right) + j\sin\left(\frac{2\pi}{N}kn\right)\right), \qquad 0 \le n,k \le N-1$$



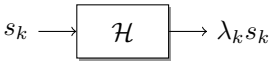








Harmonic Sinusoids are Eigenvectors of LTI Systems



- Prove that harmonic sinusoids are the eigenvectors of LTI systems simply by computing the circular convolution with input s_k and applying the periodicity of the harmonic sinusoids

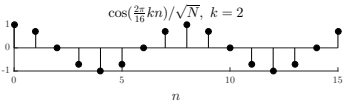
$$
\begin{aligned}
s_k[n] \circledast h[n] &= \sum_{m=0}^{N-1} s_k[(n-m)_N]\, h[m] = \sum_{m=0}^{N-1} \frac{e^{j\frac{2\pi}{N}k(n-m)_N}}{\sqrt{N}}\, h[m] \\
&= \sum_{m=0}^{N-1} \frac{e^{j\frac{2\pi}{N}k(n-m)}}{\sqrt{N}}\, h[m] = \sum_{m=0}^{N-1} \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} e^{-j\frac{2\pi}{N}km}\, h[m] \\
&= \left(\sum_{m=0}^{N-1} e^{-j\frac{2\pi}{N}km}\, h[m] \right) \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} = \lambda_k\, s_k[n]
\end{aligned}
$$

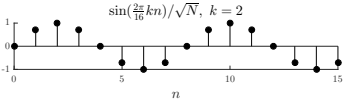
Eigenvalues of LTI Systems

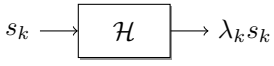
- The eigenvalue $\lambda_k \in \mathbb{C}$ corresponding to the sinusoid eigenvector s_k is called the **frequency response** at frequency k since it measures how the system "responds" to s_k

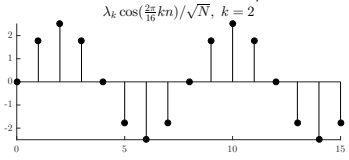
$$\lambda_k \;=\; \sum_{n=0}^{N-1} h[n]\, e^{-j\frac{2\pi}{N}kn} \;=\; \langle h, s_k \rangle \;=\; H_u[k] \quad \text{(unnormalized DFT)}$$

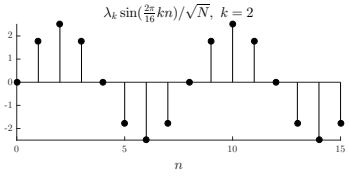
- Recall properties of the **inner product**: λ_k grows/shrinks as h and s_k become more/less similar











Eigenvector Matrix of Harmonic Sinusoids

- Stack the N normalized harmonic sinusoid $\{s_k\}_{k=0}^{N-1}$ as columns into an $N \times N$ complex orthonormal basis matrix

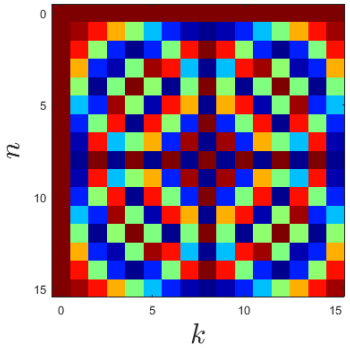
$$\mathbf{S} = \left[s_0 | s_1 | \cdots | s_{N-1}\right]$$

- The row-n, column-k entries of $\mathbf{S}$ have a very simple structure: $[\mathbf{S}]_{n,k} = e^{j\frac{2\pi}{N}kn}/\sqrt{N}$

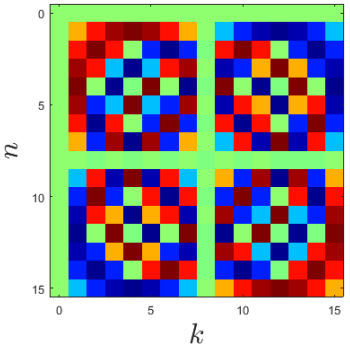
- Example: Eigenvector matrix for $N = 16$

- Note the symmetries:
  $[\mathbf{S}]_{n,k} = [\mathbf{S}]_{k,n}$
  or $\mathbf{S} = \mathbf{S}^T$
  (same sinusoids on rows/columns)

real part: $\cos(\frac{2\pi}{N}kn)/\sqrt{N}$



imaginary part: $\sin(\frac{2\pi}{N}kn)/\sqrt{N}$



Diagonal Matrix of Eigenvalues

- The eigenvalues are the frequency response (unnormalized DFT of the impulse response)

$$\lambda_k \;=\; \sum_{n=0}^{N-1} h[n]\, e^{-j\frac{2\pi}{N}kn} \;=\; \langle h, s_k \rangle \;=\; H_u[k] \;\; \text{(unnormalized DFT)}$$

- Place the N eigenvalues $\{\lambda_k\}_{k=0}^{N-1}$ on the diagonal of an $N \times N$ matrix

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{N-1} \end{bmatrix} = \begin{bmatrix} H_u[0] & & & \\ & H_u[1] & & \\ & & \ddots & \\ & & & H_u[N-1] \end{bmatrix}$$

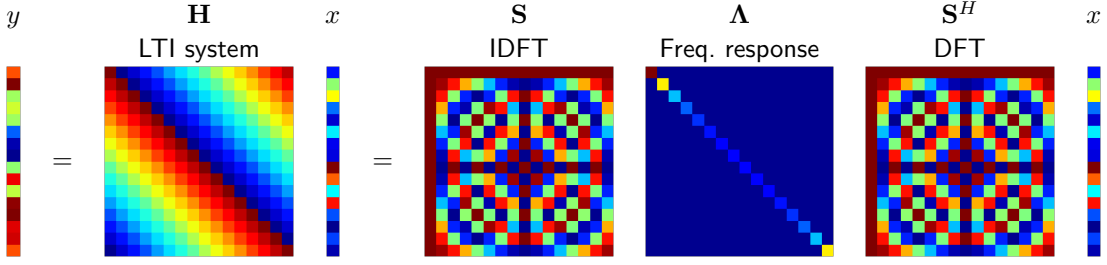
Eigendecomposition and Diagonalization of an LTI System

- Given the
  - circulant LTI system matrix $\mathbf{H}$
  - Fixed matrix of harmonic sinusoid eigenvectors $\mathbf{S}$ (corresponds to DFT/IDFT)
  - Diagonal matrix of eigenvalues $\mathbf{\Lambda}$ (frequency response, changes with $\mathbf{H}$)

  we can write

$$\mathbf{H} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^H$$

- Example for $N = 16$ (Note: Only plotting real part of $\mathbf{S}$, $\mathbf{S}^H$, and $\mathbf{\Lambda}$)



Summary

- Harmonic sinusoids are the eigenfunctions of LTI systems for finite-length signals (circulant matrices)
- Therefore, the discrete Fourier transform (DFT) is the natural tool for studying LTI systems for finite-length signals
- Frequency response $H[k]$ equals the unnormalized DFT of the impulse response $h[n]$
- Diagonalization by eigendecomposition implies

$$\mathbf{H} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^H$$

Discrete Fourier Transform (DFT)

Discrete Fourier Transform

- Another cornerstone of this course in particular and signal processing in general
- Jean Baptiste Joseph Fourier (21 March 1768 – 16 May 1830) had the radical idea of proposing that "all" signals could be represented as a linear combination of sinusoids

- Amazingly, it's true (at least in $\mathbb{C}^N$)!
- Suggestion: Re-watch the lectures on Sinusoids from Week 1

Recall: Signal Representation by Orthonormal Basis

- Given an **orthonormal basis** $\{b_k\}_{k=0}^{N-1}$ for $\mathbb{C}^N$ and orthonormal basis matrix $\mathbf{B}$, we have the following **signal representation** for any signal x

$$x \;=\; \mathbf{B}\,a \;=\; \sum_{k=0}^{N-1} \alpha_k\, b_k \qquad \text{(synthesis)}$$

$$a \;=\; \mathbf{B}^H x \quad \text{or} \quad \alpha_k \;=\; \langle x, b_k \rangle \qquad \text{(analysis)}$$

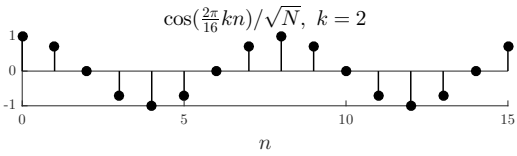
- **Synthesis:** Build up the signal x as a linear combination of the basis elements b_k weighted by the weights α_k

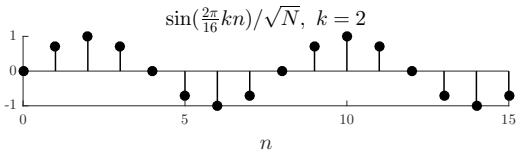
- **Analysis:** Compute the weights α_k such that the synthesis produces x; the weight α_k measures the similarity between x and the basis element b_k

Harmonic Sinusoids are an Orthonormal Basis

- Recall the length-N normalized complex **harmonic sinusoids** (normalized!)

$$s_k[n] \;=\; \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} \;=\; \frac{1}{\sqrt{N}}\left(\cos\left(\frac{2\pi}{N}kn\right) + j\sin\left(\frac{2\pi}{N}kn\right)\right), \qquad 0 \le n, k \le N-1$$





- Recall that harmonic sinusoids are **mutually orthogonal** and **normalized**

$$\langle s_k, s_l \rangle \;=\; 0, \quad k \neq l, \qquad\qquad \|s_k\|_2 \;=\; 1$$

- It is easy to show that N orthonormal vectors in an N-dimensional must be an orthonormal basis

Orthonormal Basis Matrix of Harmonic Sinusoids

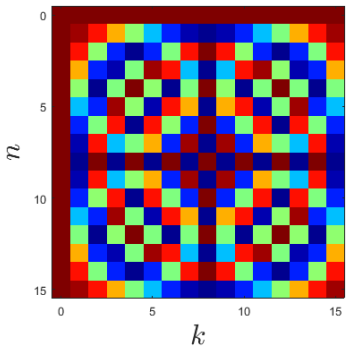
- Stack the N normalized harmonic sinusoid $\{s_k\}_{k=0}^{N-1}$ as columns into an $N \times N$ complex matrix

$$\mathbf{S} = \left[s_0 | s_1 | \cdots | s_{N-1} \right]$$

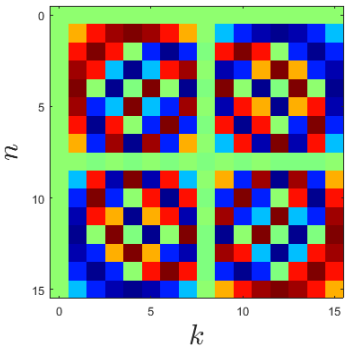
- The row-n, column-k entries of $\mathbf{S}$ have a very simple structure: $[\mathbf{S}]_{n,k} = e^{j\frac{2\pi}{N}kn}/\sqrt{N}$

- Example: Eigenvector matrix for $N = 16$

real part: $\cos(\frac{2\pi}{N}kn)/\sqrt{N}$



imaginary part: $\sin(\frac{2\pi}{N}kn)/\sqrt{N}$



- Note the symmetries:
  $[\mathbf{S}]_{n,k} = [\mathbf{S}]_{k,n}$
  or $\mathbf{S} = \mathbf{S}^T$
  (same sinusoids on rows/columns)

Inverse of Orthonormal Basis Matrix of Harmonic Sinusoids

- The row-n, column-k entries of $\mathbf{S}$ have a very simple structure

$$[\mathbf{S}]_{n,k} \;=\; \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- $\mathbf{S}$ is **unitary**, and so its inverse matrix $\mathbf{S}^{-1} = \mathbf{S}^H$

- Entries of the inverse matrix

$$[\mathbf{S}^{-1}]_{n,k} \;=\; [\mathbf{S}^H]_{n,k} \;=\; \left(\frac{e^{j\frac{2\pi}{N}nk}}{\sqrt{N}}\right)^* \;=\; \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} \;=\; ([\mathbf{S}]_{n,k})^*$$

- Thanks to the symmetry of $\mathbf{S}$, we have the interesting equalities: $\mathbf{S}^{-1} \;=\; \mathbf{S}^H \;=\; \mathbf{S}^*$

Signal Representation by Harmonic Sinusoids

- Given the normalized complex harmonic sinusoids $\{s_k\}_{k=0}^{N-1}$ and the orthonormal basis matrix $\mathbf{S}$, we define the (normalized) **discrete Fourier transform** (DFT) for any signal $x \in \mathbb{C}^N$

- **Analysis (Forward Normalized DFT)**

$$\begin{aligned} X &= \mathbf{S}^H x \\ X[k] &= \langle x, s_k \rangle = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} \end{aligned}$$

- **Synthesis (Inverse Normalized DFT)**

$$\begin{aligned} x &= \mathbf{S}X \\ x[n] &= \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}} \end{aligned}$$

Interpretation: Signal Representation by Harmonic Sinusoids

- **Analysis (Forward DFT)**
  - Choose the DFT coefficients $X[k]$ such that the synthesis produces the signal x
  - The weight $X[k]$ measures the similarity between x and the harmonic sinusoid s_k
  - Therefore, $X[k]$ measures the "frequency content" of x at frequency k

$$X[k] \;=\; \langle x, s_k \rangle \;=\; \sum_{n=0}^{N-1} x[n] \, \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

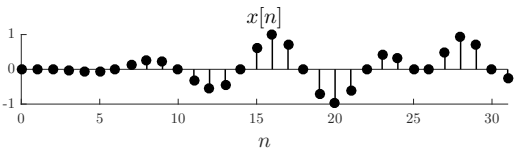
- **Synthesis (Inverse DFT)**
  - Build up the signal x as a linear combination of harmonic sinusoids s_k weighted by the DFT coefficients $X[k]$

$$x[n] \;=\; \sum_{k=0}^{N-1} X[k] \, \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

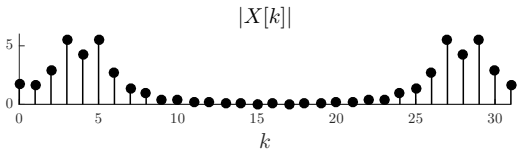
Example: Signal Representation by Harmonic Sinusoids

- **Analysis (Forward DFT)**
  - Choose the DFT coefficients $X[k]$ such that the synthesis produces the signal x
  - $X[k]$ measures the similarity between x and the harmonic sinusoid s_k
  - Therefore, $X[k]$ measures the "frequency content" of x at frequency k
  - Even if the signal x is real-valued, the DFT coefficients X will be complex, in general

$$X[k] \;=\; \langle x, s_k \rangle \;=\; \sum_{n=0}^{N-1} x[n] \, \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$



time domain



frequency domain

The Unnormalized DFT

- Normalized forward and inverse DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

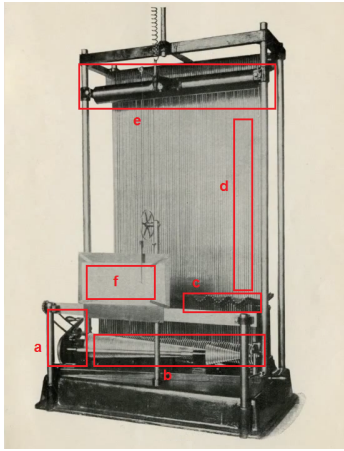
- **Unnormalized** forward and inverse DFT is more popular in practice (we will use both)

$$X_u[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

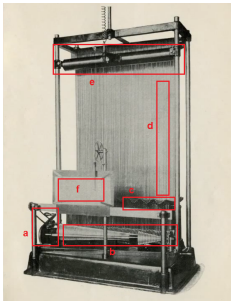
$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_u[k]\, e^{j\frac{2\pi}{N}kn}$$

Aside: Physical Implementation of Fourier Analysis and Synthesis

- Before computers, Fourier analysis had to calculated done by hand
- Albert Michelson invented a machine that could physically perform such calculations

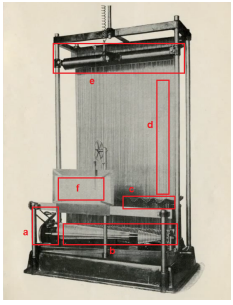


Aside: Physical Implementation of Fourier Analysis and Synthesis



- For synthesis, the a) crank turns a series of b) gears that spin at different rates, moving c) arms up and down in sinusoidal fashion
- d) Levers weight the various sinusoidal movements, which are summed with e) springs and a bar
- f) The result is plotted

Aside: Physical Implementation of Fourier Analysis and Synthesis



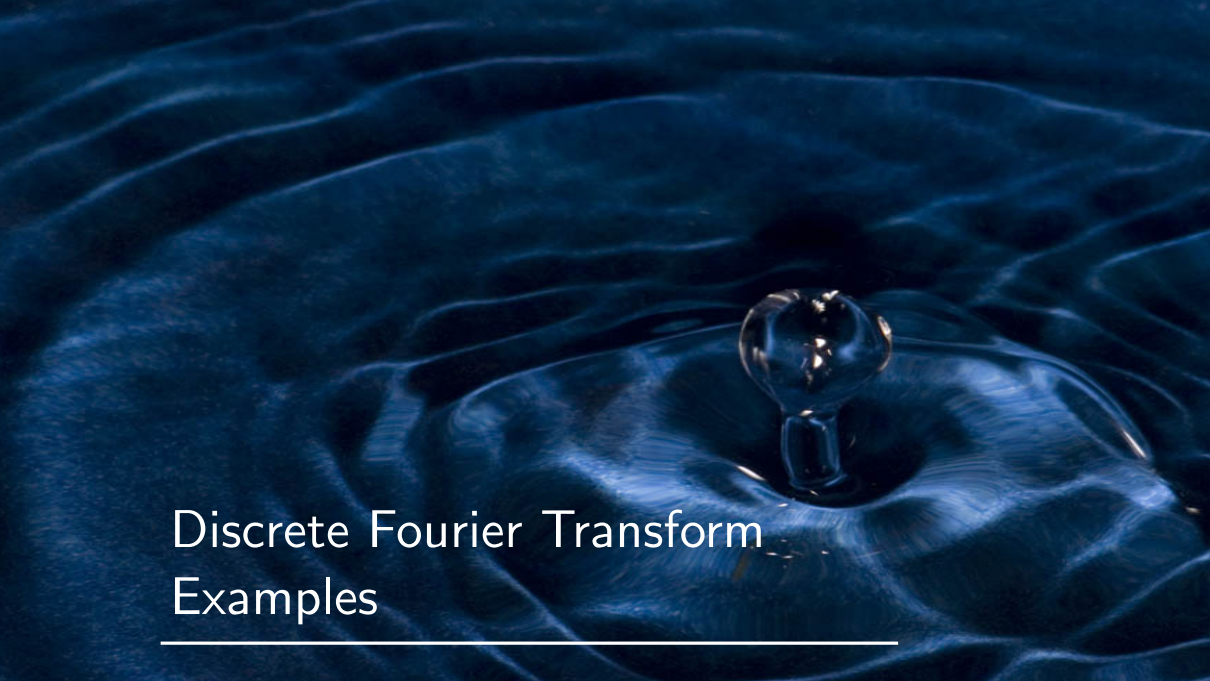
- For analysis, the d) levers are oriented into the shape of a single period of a signal
- Turning the crank amounts to taking the inner product of this shape with sinusoids (each turn makes a higher frequency sinusoid across the c) bars)
- The result is that the Fourier transform is then f) plotted
- To see videos of synthesis and analysis at work, see **http://www.engineerguy.com/fourier/**

Summary

- The discrete Fourier transform (DFT) is an orthonormal basis transformation based on the harmonic sinusoids

$$s_k[n] = \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- The DFT maps signals from the "time domain" ($x[n]$) to the "frequency domain" ($X[k]$)
- The DFT coefficient $X[k]$ measures the similarity between the time signal x and the harmonic sinusoid s_k with frequency k
- The set of DFT coefficients X contains all of the information in the signal x (and vice versa)
- Do not confuse the normalized and unnormalized DFTs! The normalized DFT is more elegant, but the unnormalized DFT is much more popular in practice



Discrete Fourier Transform Examples

Discrete Fourier Transform

- Useful Matlab commands: `fft`, `fftshift`, `semilogy`. **Click here** to view a video demonstration.
- **Click here** to see other explanations and graphical representations of the DFT.

Discrete Fourier Transform Properties

Properties of the DFT

- **Normalized** forward and inverse DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- **Unnormalized** forward and inverse DFT is more popular in practice (we will use both)

$$X_u[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_u[k]\, e^{j\frac{2\pi}{N}kn}$$

DFT Pairs

- If $x[n]$ and $X[k]$ are such that

$$X[k] = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

then we say they are a **DFT pair**

$$x[n] \overset{\text{DFT}}{\longleftrightarrow} X[k]$$

The DFT is Periodic

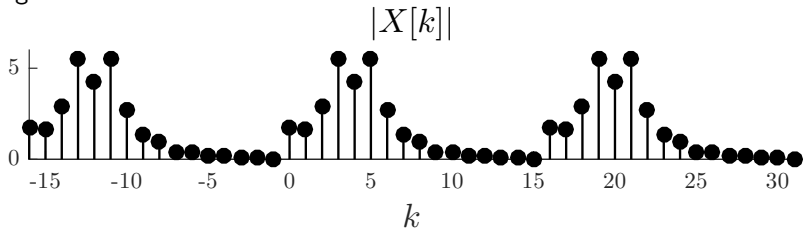
- The DFT is of finite length N, but it can also be interpreted as **periodic** with period N

$$X[k] \;=\; X[k+lN], \quad l \in \mathbb{Z}$$

- Proof

$$X[k+lN] \;=\; \sum_{n=0}^{N-1} x[n]\, \frac{e^{-j\frac{2\pi}{N}(k+lN)n}}{\sqrt{N}} \;=\; \sum_{n=0}^{N-1} x[n]\, \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} e^{-j\frac{2\pi}{N}lNn} \;=\; X[k] \;\checkmark$$

- DFT of length $N = 16$

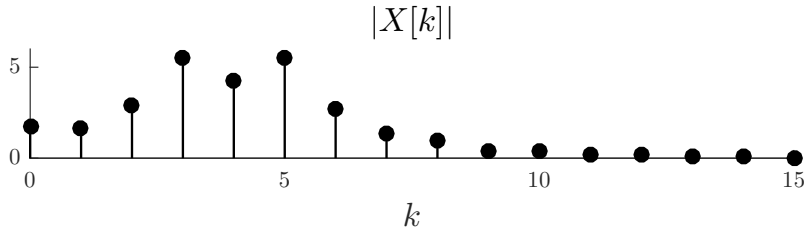


DFT Frequencies

$$X[k] \;=\; \langle x, s_k \rangle \;=\; \sum_{n=0}^{N-1} x[n] \, \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- $X[k]$ measures the similarity between the time signal x and the harmonic sinusoid s_k
- Therefore, $X[k]$ measures the "frequency content" of x at frequency

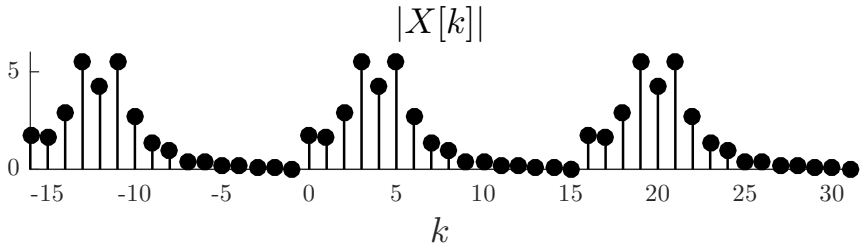
$$\omega_k \;=\; \frac{2\pi}{N} k$$



DFT Frequencies and Periodicity

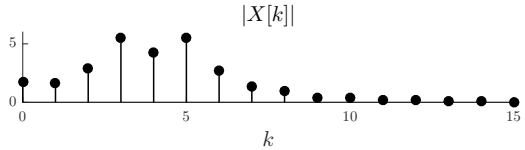
- Periodicity of DFT means we can treat frequencies mod N
- $X[k]$ measures the "frequency content" of x at frequency $\omega_k = \frac{2\pi}{N}(k)_N$
- Example: $X[N-1] = X[(-1)_N]$ measures the "frequency content" of x at the (same) frequencies

$$\omega_{N-1} = \frac{2\pi}{N}(N-1) = -\frac{2\pi}{N}$$

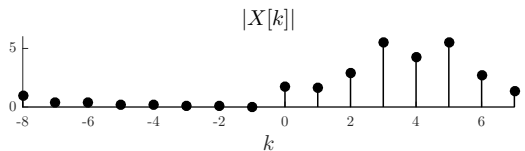


DFT Frequency Ranges

- Periodicity of DFT means every length-N interval of k carries the same information
- Typical interval 1: $0 \leq k \leq N-1$ corresponds to frequencies ω_k in the interval $0 \leq \omega < 2\pi$



- Typical interval 2: $-\frac{N}{2} \leq k \leq \frac{N}{2}-1$ corresponds to frequencies ω_k in the interval $-\pi \leq \omega < \pi$



The Inverse DFT is Periodic

$$x[n] \;=\; \sum_{k=0}^{N-1} X[k]\,\frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- The time signal produced by the inverse DFT (synthesis) is **periodic** with period N

$$x[n] \;=\; x[n+mN], \quad m \in \mathbb{Z}$$

- Proof

$$x[n+mN] \;=\; \sum_{n=0}^{N-1} X[k]\,\frac{e^{j\frac{2\pi}{N}k(n+mN)}}{\sqrt{N}} \;=\; \sum_{n=0}^{N-1} X[k],\,\frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}e^{j\frac{2\pi}{N}kmN} \;=\; x[n] \;\checkmark$$

- This should not be surprising, since the harmonic sinusoids are periodic with period N

DFT and Circular Shift

- If $x[n]$ and $X[k]$ are a DFT pair then

$$x[(n-m)_N] \overset{\text{DFT}}{\longleftrightarrow} e^{-j\frac{2\pi}{N}km} X[k]$$

- Proof: Use the change of variables $r = (n-m)_N$

$$\sum_{n=0}^{N-1} x[(n-m)_N] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} = \sum_{r=0}^{N-1} x[r] \frac{e^{-j\frac{2\pi}{N}k(r+m)}}{\sqrt{N}} = e^{-j\frac{2\pi}{N}km} \sum_{r=0}^{N-1} x[r] \frac{e^{-j\frac{2\pi}{N}kr}}{\sqrt{N}}$$

$$= e^{-j\frac{2\pi}{N}km} X[k] \quad \checkmark$$

DFT and Modulation

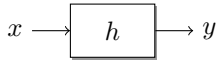
- If $x[n]$ and $X[k]$ are a DFT pair then

$$e^{j\frac{2\pi}{N}ln}\,x[n] \overset{\text{DFT}}{\longleftrightarrow} X[(k-l)_N]$$

- Proof:

$$\sum_{n=0}^{N-1} x[n]\,e^{j\frac{2\pi}{N}ln}\,\frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} = \sum_{n=0}^{N-1} x[n]\,\frac{e^{-j\frac{2\pi}{N}(k-l)n}}{\sqrt{N}} = X[(k-l)_N] \quad \checkmark$$

DFT and Circular Convolution



$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- If

$$x[n] \overset{\text{DFT}}{\longleftrightarrow} X_u[k], \qquad h[n] \overset{\text{DFT}}{\longleftrightarrow} H_u[k], \qquad y[n] \overset{\text{DFT}}{\longleftrightarrow} Y_u[k]$$

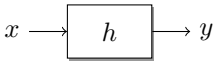
  then

$$Y_u[k] \;=\; H_u[k]\, X_u[k]$$

- Circular convolution in the time domain $\Leftrightarrow$ multiplication in the frequency domain

- Note: This is equivalent to $\mathbf{\Lambda} = \mathbf{S}^H \mathbf{H} \mathbf{S}$

DFT and Circular Convolution – Proof



$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m]$$

- Proof

$$Y_u[k] \;=\; \sum_{n=0}^{N-1} y[n]\, e^{-j\frac{2\pi}{N}kn} \;=\; \sum_{n=0}^{N-1} \left(\sum_{m=0}^{N-1} h[(n-m)_N]\, x[m] \right) e^{-j\frac{2\pi}{N}kn}$$

$$= \sum_{m=0}^{N-1} x[m] \left(\sum_{n=0}^{N-1} h[(n-m)_N] e^{-j\frac{2\pi}{N}kn} \right) \;=\; \sum_{m=0}^{N-1} x[m] \left(\sum_{r=0}^{N-1} h[r] e^{-j\frac{2\pi}{N}k(r+m)} \right)$$

$$= \left(\sum_{m=0}^{N-1} x[m] e^{-j\frac{2\pi}{N}km} \right) \left(\sum_{r=0}^{N-1} h[r] e^{-j\frac{2\pi}{N}kr} \right) \;=\; X_u[k]\, H_u[k] \quad \checkmark$$

The DFT is Linear

- It is trivial to show that if

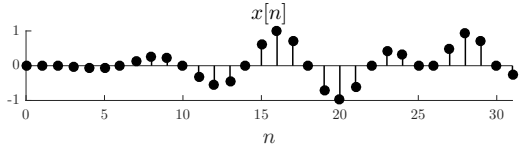
$$x_1[n] \overset{\text{DFT}}{\longleftrightarrow} X_1[k], \qquad x_2[n] \overset{\text{DFT}}{\longleftrightarrow} X_2[k]$$

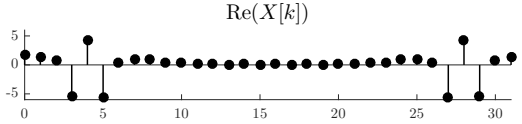
then

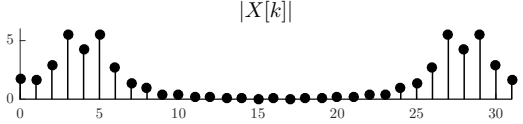
$$\alpha_1 x_1[n] + \alpha_2 x[2] \overset{\text{DFT}}{\longleftrightarrow} \alpha_1 X_1[k] + \alpha_2 X_2[k]$$

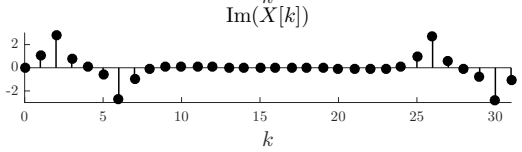
The DFT is Complex Valued

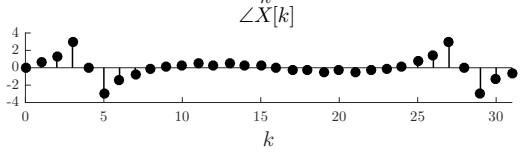
- Even if the signal $x[n]$ is real-valued, the DFT is complex-valued, in general











DFT Symmetry Properties (1)

- The harmonic sinusoid basis elements $s_k[n] = e^{j\frac{2\pi}{N}kn}$ of the DFT have symmetry properties:

$$\mathrm{Re}\left(e^{j\frac{2\pi}{N}kn}\right) = \cos\left(\frac{2\pi}{N}kn\right) \quad \text{(even function)}$$

$$\mathrm{Im}\left(e^{j\frac{2\pi}{N}kn}\right) = \sin\left(\frac{2\pi}{N}kn\right) \quad \text{(odd function)}$$

- These induce corresponding symmetry properties on $X[k]$ around the frequency $k = 0$

- **Even** signal/DFT

$$x[n] = x[(-n)_N], \qquad X[k] = X[(-k)_N]$$

- **Odd** signal/DFT

$$x[n] = -x[(-n)_N], \qquad X[k] = -X[(-k)_N]$$

DFT Symmetry Properties (2)

$x[n]$	$X[k]$	$\mathrm{Re}(X[k])$	$\mathrm{Im}(X[k])$	$\lvert X[k]\rvert$	$\angle X[k]$
real	$X[-k] = X[k]^*$	even	odd	even	odd
real & even	real & even	even	zero	even	
real & odd	imaginary & odd	zero	odd	even	
imaginary	$X[-k] = -X[k]^*$	odd	even	even	odd
imaginary & even	imaginary & even	zero	even	even	
imaginary & odd	real & odd	odd	zero	even	

DFT Symmetry Properties (3)

- Prove that if $x[n]$ is real, then $X[-k] = X[k]^*$

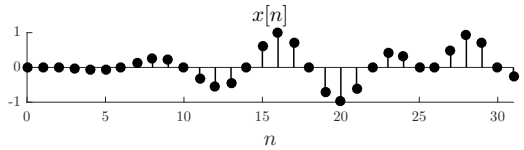
- Simply compute $X[k]^*$ and use the fact that $x[n]$ is real

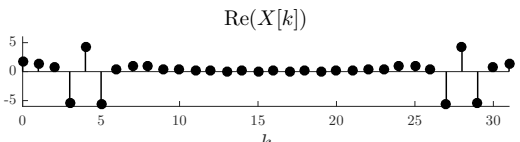
$$X[k]^* = \left(\sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} \right)^* = \sum_{n=0}^{N-1} x[n]^* \frac{e^{+j\frac{2\pi}{N}kn}}{\sqrt{N}} = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}(-k)n}}{\sqrt{N}} = X[-k] \checkmark$$

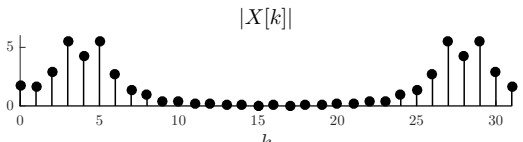
- Easy to continue on to prove that $\mathrm{Re}(X[-k]) = \mathrm{Re}(X[k])$ (that is, the real part of $X[k]$ is even) by taking the real part of both sides of the equation $X[-k] = X[k]^*$

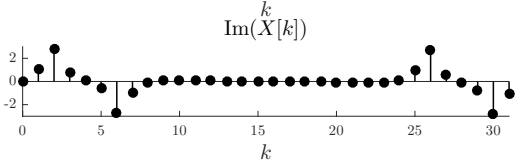
DFT Symmetry Properties (4)

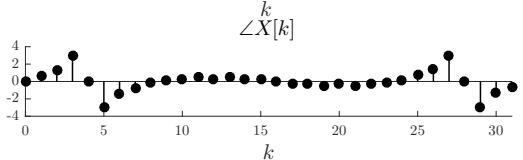
- Example: Real-valued signal $x[n]$





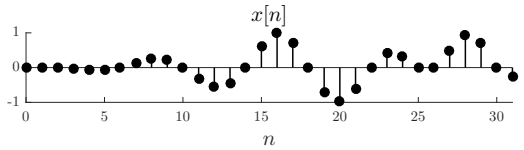


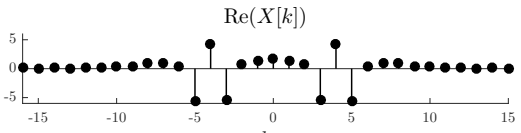


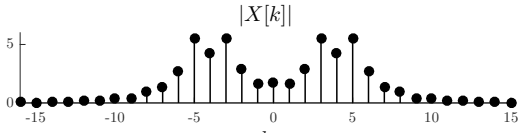


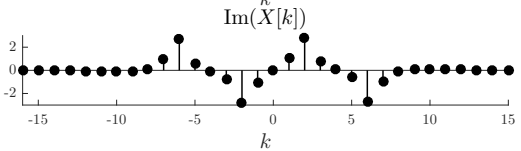
DFT Symmetry Properties (5)

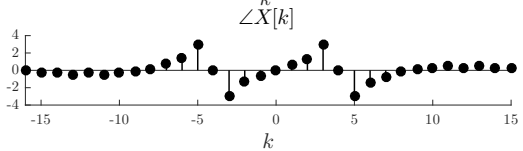
- Example: Real-valued signal $x[n]$, but plotting $X[k]$ using Matlab `fftshift` command











Duality of the DFT

- Note that the inverse and forward DFT formulas are identical except for conjugation of the harmonic sinusoids

$$X[k] = \sum_{n=0}^{N-1} x[n] \frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

$$x[n] = \sum_{k=0}^{N-1} X[k] \frac{e^{j\frac{2\pi}{N}kn}}{\sqrt{N}}$$

- Thus, any DFT property that is true for $x[n]$ is also true for $X[-k]$

- Example: If $X[k]$ is real, then $\mathrm{Re}(x[n])$ is even and $\mathrm{Im}(x[n])$ is odd

Summary

- DFT and inverse DFT are periodic
- Useful to index $X[k]$ by $-\frac{N}{2} \leq k \leq \frac{N}{2} - 1$ (frequencies $-\pi \leq \frac{2\pi}{N} k < \pi$) as well as by $0 \leq k \leq N - 1$ (frequencies $0 \leq \frac{2\pi}{N} k < 2\pi$)
- Circular convolution in time becomes simple multiplication in frequency
- DFT has useful symmetry properties

Fast Fourier Transform (FFT)

Cost to Compute the Discrete Fourier Transform

- Recall the (unnormalized) DFT of the time signal $x[n]$

$$X_u[k] \;=\; \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \qquad 0 \le k \le N-1$$

- What is the **computational cost** of the DFT?
- Number of Multiplies: Must multiply $x[n]\, e^{-j\frac{2\pi}{N}kn}$ for each value of

$$n = 0, 1, \ldots, N-1 \text{ and } k = 0, 1, \ldots, N-1 \;\Rightarrow\; N^2 \text{ total multiplies}$$

- Number of Additions: Must sum the N products $x[n]\, e^{-j\frac{2\pi}{N}kn}$ for each value of

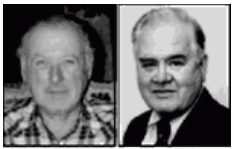
$$k = 0, 1, \ldots, N-1 \;\Rightarrow\; N(N-1) \approx N^2 \text{ total adds}$$

- Total computational cost of DFT: N^2 adds and N^2 multiplies – $O(N^2)$ **complexity**

Fast Fourier Transform

- $O(N^2)$ computational complexity is too high for many important applications; it is not uncommon to have $N = 10^7$ or more

- Important step forward in 1965: Cooley and Tukey "discovered" the fast Fourier transform (FFT), which lowers the computational complexity to $O(N \log N)$


James Cooley John Tukey

- Example: For $N = 10^7$

$$2N^2 = 2 \times 10^{14} \qquad 2N \log_2 N = 4.65 \times 10^8$$

- It turns out that Gauss invented the FFT in 1805 (Heideman, Johnson, Burrus, 1984)

Fast Fourier Transform

- There are many different kinds of FFTs; here we will study the simplest: the **radix-2, decimation-in-time** FFT

- Clearly we can use the same methods to speed up both the forward and inverse DFT (by duality); we will work with the forward DFT (and drop the subscript u for unnormalized)

$$X[k] \;=\; \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

- To keep the notation clean, define the **twiddle factor**: $W_N = e^{-j\frac{2\pi}{N}} \in \mathbb{C}$

$$X[k] \;=\; \sum_{n=0}^{N-1} x[n]\, W_N^{kn}$$

Twiddle Factors are Periodic

- Note that the twiddle factors $W_N = e^{-j\frac{2\pi}{N}}$ are **periodic** in n and k

$$W_N^{kn} \;=\; W_N^{k(n+N)} \;=\; W_N^{(k+N)n}$$

- Proof

$$e^{-j\frac{2\pi}{N}kn} \;=\; e^{-j\frac{2\pi}{N}k(n+N)} \;=\; e^{-j\frac{2\pi}{N}(k+N)n}$$

FFT – Step 1

- In the radix-2, decimation-in-time FFT, the signal length N is a power of 2
- The FFT is a **divide and conquer** algorithm: We will split the length-N into two length-$N/2$ FFTs and then iterate; each split will save on computations
- We will work out the specific example of an $N = 8$ DFT, but the ideas extend to any power-of-two length
- **Step 1:** Break the signal $x[n]$ into two sub-signals:
  - **even samples**: $x[2n]$
  - **odd samples:** $x[2n+1], \quad n = 0, 1, \ldots, N/2 - 1$

$$X[k] \;=\; \sum_{n=0}^{N-1} x[n]\, W_N^{kn} \;=\; \sum_{n=0}^{N/2-1} x[2n]\, W_N^{k(2n)} \;+\; \sum_{n=0}^{N/2-1} x[2n+1]\, W_N^{k(2n+1)}$$

FFT – Step 2

- **Step 2:** Reorganize the two sums into two length-$N/2$ DFTs

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{kn} = \sum_{n=0}^{N/2-1} x[2n]\, W_N^{k(2n)} + \sum_{n=0}^{N/2-1} x[2n+1]\, W_N^{k(2n+1)}$$
$$= \sum_{n=0}^{N/2-1} x[2n]\, W_N^{2kn} + W_N^k \sum_{n=0}^{N/2-1} x[2n+1]\, W_N^{2kn}$$

- Note that $W_N^{2kn} = e^{-j\frac{2\pi}{N}2kn} = e^{-j\frac{2\pi}{N/2}kn} = W_{N/2}^{kn}$ and so we have ...

- Term 1 $= E[k] = \sum_{n=0}^{N/2-1} x[2n]\, W_{N/2}^{kn} = N/2$-point DFT of the even samples of $x[n]$

- Term 2 $= W_N^k\, O[k] = W_N^k \sum_{n=0}^{N/2-1} x[2n+1]\, W_{N/2}^{kn} = N/2$-point DFT of the odd samples of $x[n]$

FFT – Step 3

- **Step 3:** Not so fast! We need to evaluate

$$\begin{aligned} X[k] &= \sum_{n=0}^{N/2-1} x[2n]\, W_{N/2}^{kn} \;+\; W_N^k \sum_{n=0}^{N/2-1} x[2n+1]\, W_{N/2}^{kn} \\ &= E[k] \;+\; W_N^k\, O[k] \end{aligned}$$

for the entire range $k = 0, 1, \ldots, N-1$ and not just $k = 0, 1, \ldots, N/2-1$

- **Periodicity of the twiddle factors** implies that $E[k]$ and $O[k]$ are also periodic with period $N/2$

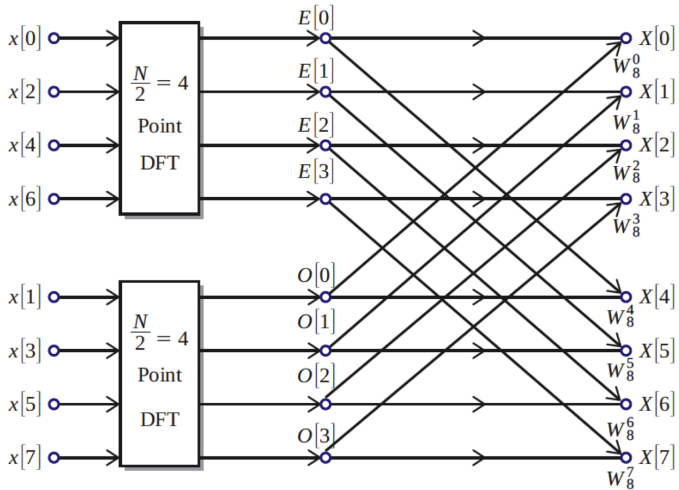
$$E[k+N/2] \;=\; \sum_{n=0}^{N/2-1} x[2n]\, W_{N/2}^{(k+N/2)n} \;=\; \sum_{n=0}^{N/2-1} x[2n]\, W_{N/2}^{kn} \;=\; E[k]$$

and similarly

$$O[k] \;=\; O[k+N/2]$$

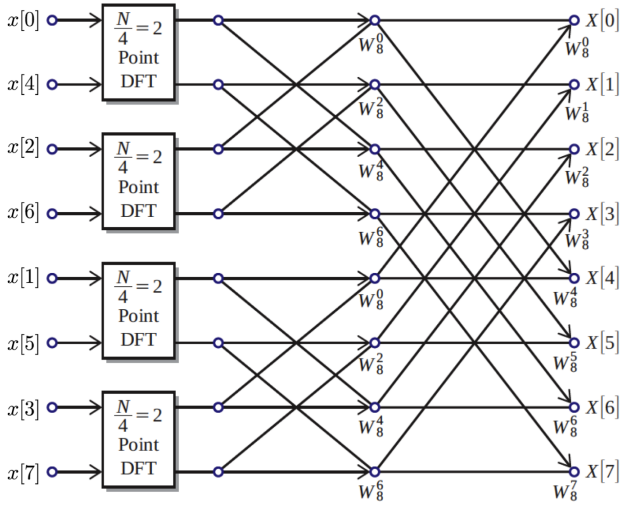
FFT – The Result

$$X[k] \;=\; E[k] + W_N^k\, O[k], \qquad k = 0, 1, \ldots, N-1$$



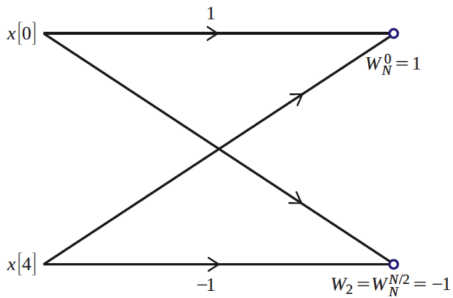
FFT – Iterate

- **Divide and Conquer!** Break the two length-$N/2$ DFTs into four length-$N/4$ DFTs

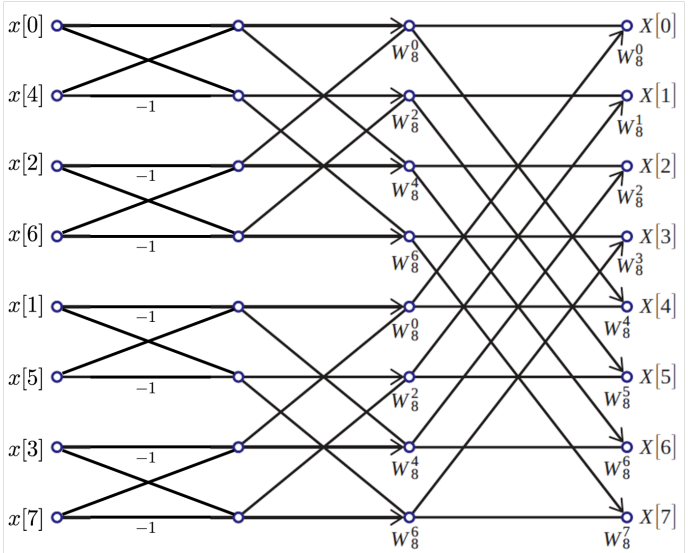


FFT – Divided and Conquered

- Iteration ends when we reach a length-2 DFT (here $N/4 = 2$)
- Length-2 DFT has a lovely **butterfly** structure



FFT of Length 8



Computational Savings

- **FFT:** Multiplies and adds required

$$2 \times 8 = 16 \text{ multiplies}$$

$$3 \times 8 = 24 \text{ adds}$$

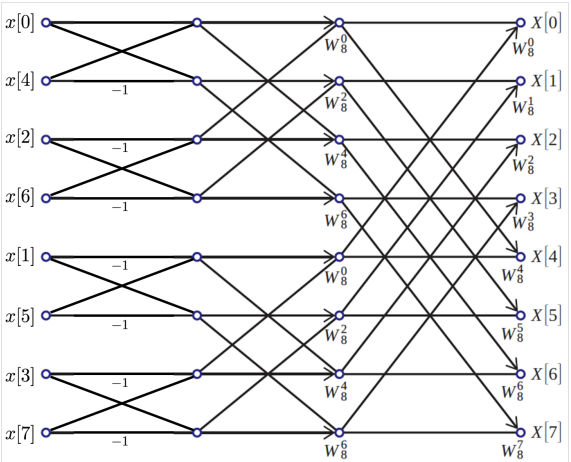
- **DFT:** Multiplies and adds required

$$8^2 = 64 \text{ multiplies}$$

$$8^2 - 8 = 56 \text{ adds}$$

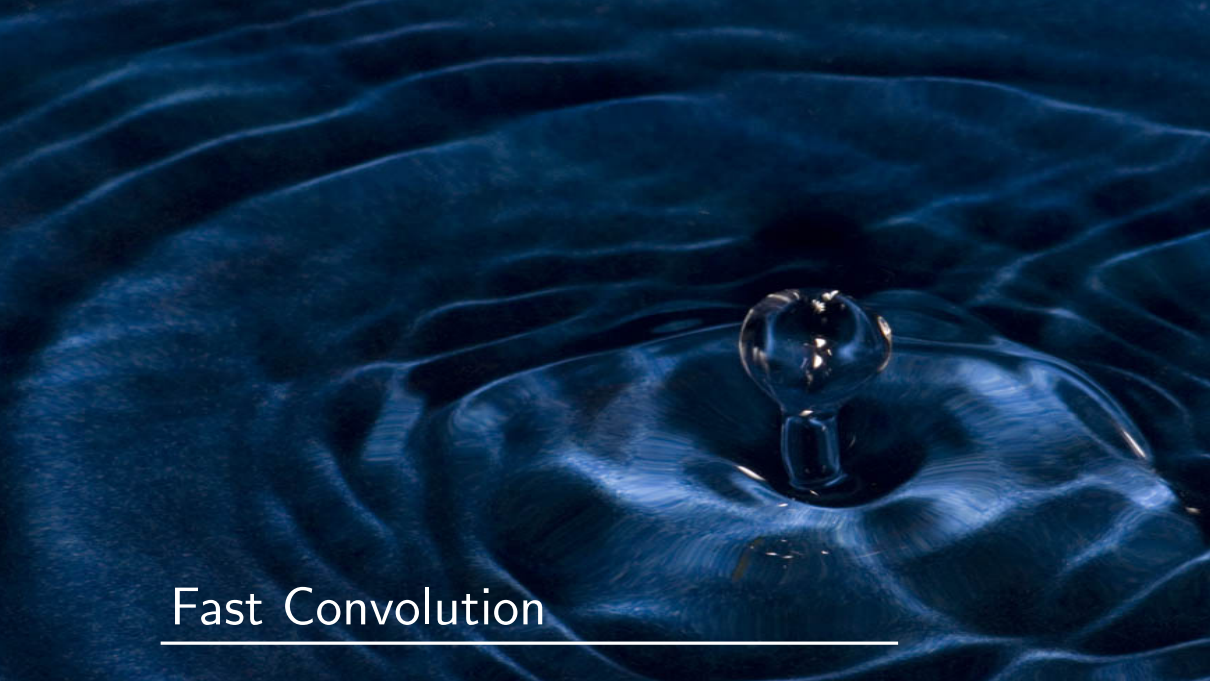
- In general, since a length-2^q FFT consists of q stages, the total number of multiplies and adds scales as

$$N \log_2 N \ll N^2$$



Summary

- The FFT has been called the "most important computational algorithm of our generation"
- The field of digital signal processing exploded after its introduction (1965)
- Why it works:
  - Symmetry and periodicity of sinusoids
  - Divide and conquer
- There are are many different kinds of FFTs for different lengths and different situations
- Rice University's resident FFT expert: Prof. C. Sidney Burrus



Fast Convolution

Cost to Compute a Circular Convolution

- Recall the circular convolution of two length-N time signals $x[n]$ and $h[n]$

$$y[n] \;=\; x[n] \circledast h[n] \;=\; \sum_{m=0}^{N-1} h[(n-m)_N]\, x[m], \qquad 0 \le n < N-1$$

- What is the **computational cost** of circular convolution?
- Number of Multiplies: Must multiply $h[(n-m)_N]\, x[n]$ for each value of

$$m = 0, 1, \ldots, N-1 \text{ and } n = 0, 1, \ldots, N-1 \;\Rightarrow\; N^2 \text{ total multiplies}$$

- Number of Additions: Must sum the N products $h[(n-m)_N]\, x[n]$ for each value of

$$n = 0, 1, \ldots, N-1 \;\Rightarrow\; N(N-1) \approx N^2 \text{ total adds}$$

- Total computational cost: N^2 adds and N^2 multiplies – $O(N^2)$ **complexity**

Circular Convolution via the FFT

- We can reduce the computational cost substantially if we move to the frequency domain using the DFT as computed by the FFT

- **Step 1:** Compute $H[k]$ and $X[k]$ of $h[n]$ and $x[n]$, respectively
  - Computational cost $= O(N \log N)$

- **Step 2:** Multiply $Y[k] = H[k] \times X[k]$
  - Computational cost $= O(N)$

- **Step 3:** Compute $y[n]$ via the inverse DFT of $Y[k]$
  - Computational cost $= O(N \log N)$

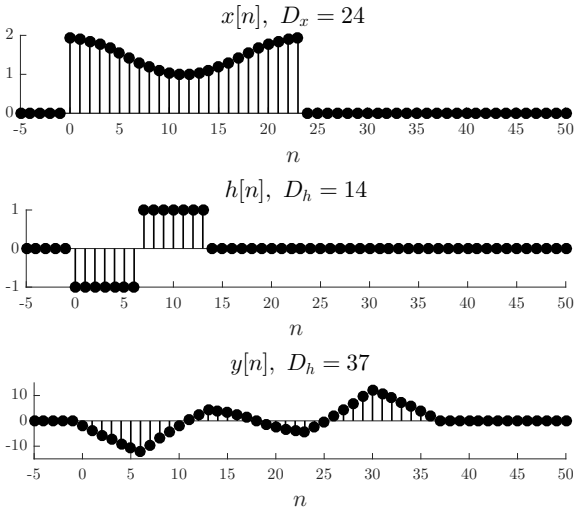
- Total computational cost: $O(N \log N) \ll N^2$

Extension to Fast Convolution of Infinite-Length, Finite-Duration Signals

- Applications tend to use (at least implicitly) infinite-length convolution more often than circular convolution
- Fortunately there is a clever way to trick a circular convolution into performing an infinite-length convolution
- Basic idea: zero pad the signals such that any circular wrap-around effects are zeroed out by the zero padding

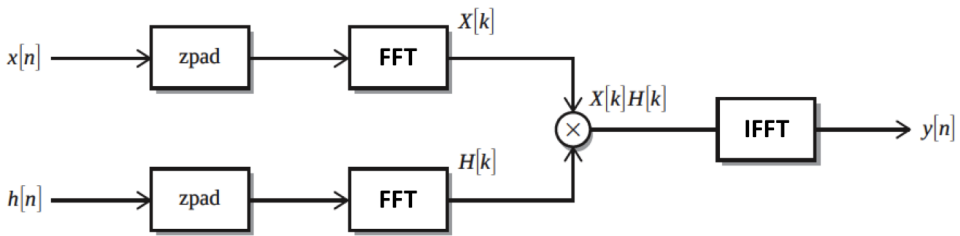
Duration of Convolution

- Recall that, if x has duration D_x samples and h has duration D_h samples, then the infinite-length convolution $y = x * h$ has duration at most $D_x + D_h - 1$ samples (proof by picture is simple)



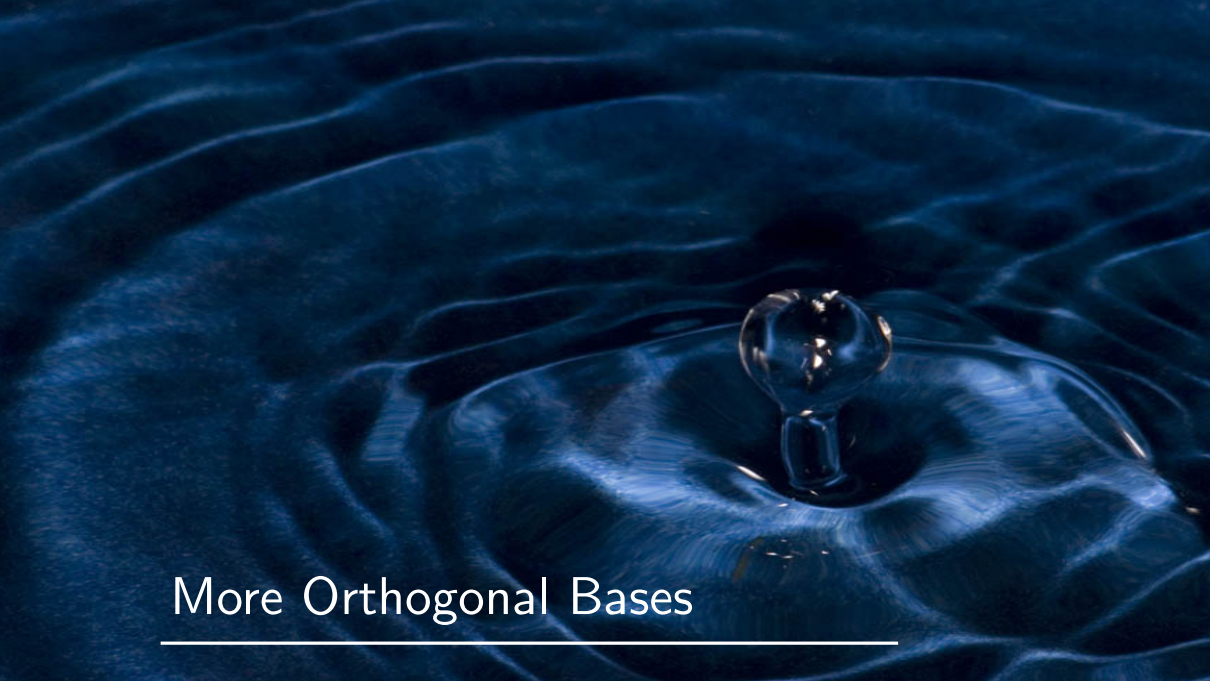
Extension to Fast Convolution of Infinite-Length, Finite-Duration Signals

- If x has duration D_x samples and h has duration D_h samples, then the infinite-length convolution $y = x * h$ has duration at most $D_x + D_h - 1$ samples
- If we zero pad both x and h to length $D_x + D_h - 1$ and compute their circular convolution $y' = x_{zp} \circledast h_{zp}$ ...
- Then the nonzero entries of the circular convolution y' will agree with those of the infinite-length convolution y



Summary

- Convolution computation can be expensive – $O(N^2)$ complexity
- Fast convolution is much faster – only $O(N \log N)$ complexity
- Can compute circular convolution by multiplying DFTs computing via FFT
- Can compute finite-duration, infinite-length convolution by zero padding before FFT



More Orthogonal Bases

Recall: Signal Representation by Orthonormal Basis

- Given an **orthonormal basis** $\{b_k\}_{k=0}^{N-1}$ for $\mathbb{C}^N$ and orthonormal basis matrix $\mathbf{B}$, we have the following **signal representation** for any signal x

$$x \;=\; \mathbf{B}\,a \;=\; \sum_{k=0}^{N-1} \alpha_k\, b_k \qquad \text{(synthesis)}$$

$$a \;=\; \mathbf{B}^H x \quad \text{or} \quad \alpha_k \;=\; \langle x, b_k \rangle \qquad \text{(analysis)}$$

- **Synthesis:** Build up the signal x as a linear combination of the basis elements b_k weighted by the weights α_k

- **Analysis:** Compute the weights α_k such that the synthesis produces x; the weight α_k measures the similarity between x and the basis element b_k

More Orthonormal Bases

- The DFT is the right transform to study LTI systems; the frequency domain arises naturally
- DFT is based on **complex-valued** basis elements (the harmonic sinusoids)
- **Challenge 1:** A signal $x \in \mathbb{R}^N$ has N complex DFT coefficients (real and imaginary parts of each DFT coefficient)
  - This is a problem in **compression** applications, where we would like approximate a smooth signal x by just a few DFT coefficients ($2\times$ redundancy)
- **Challenge 2:** Some signals are best represented neither in the time domain nor the frequency domain
  - For example, in a domain in between time and frequency ("time-frequency", like a musical score)
- Due to these and other challenges, there has been significant interest in developing additional orthonormal basis transformations beyond the DFT

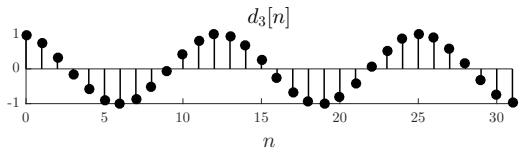
Discrete Cosine Transform (DCT)

- A DFT-like transform but using **real-valued basis functions** (`dct` in Matlab)
  - There are actually several different DCTs; here we present just one of them (the "DCT-II")
  - DCT is the transform inside JPEG image compression and MPEG video compression

- DCT Basis functions of length N (orthogonal)

$$d_k[n] \;=\; \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \quad 0 \le n,k \le N-1$$

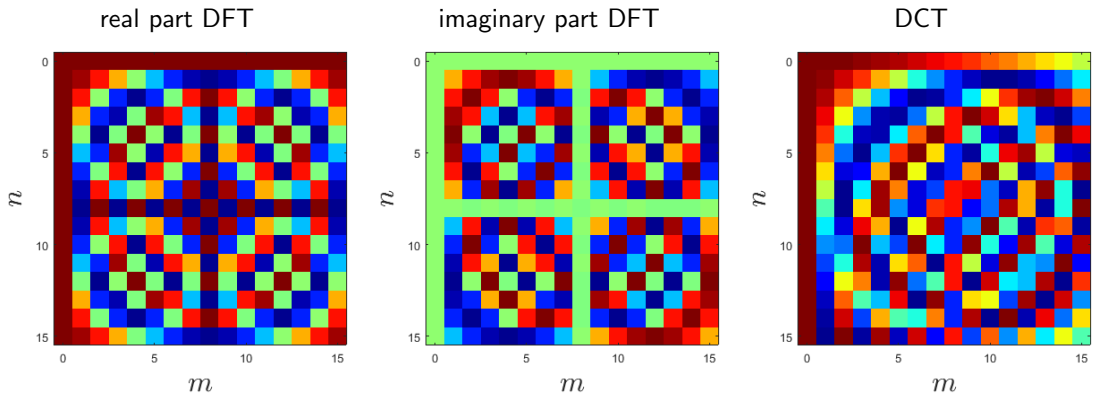
- Example: $N = 32,\ k = 3$



- Note: Not periodizable like the (complex) harmonic sinusoids of the DFT basis

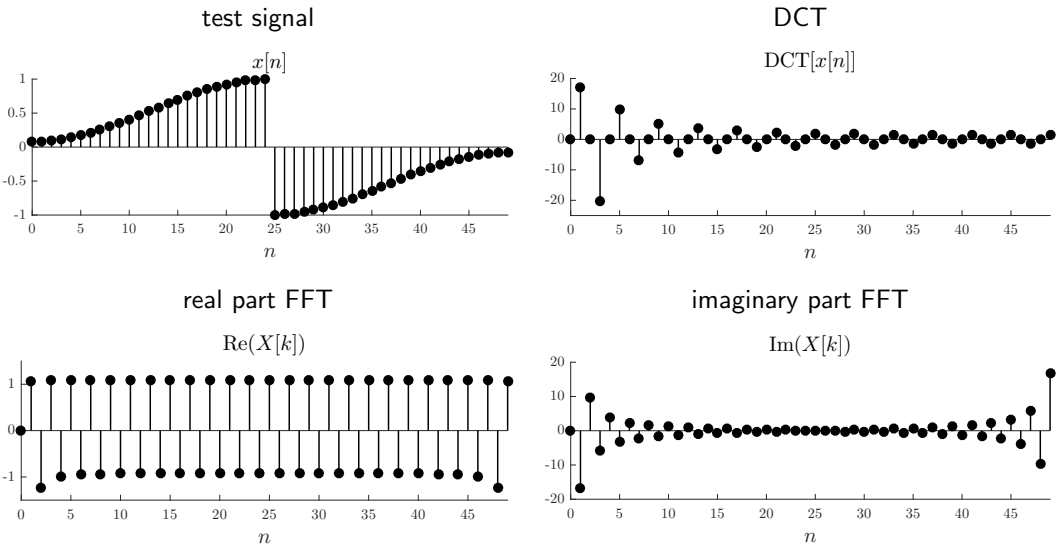
DCT Orthogonal Basis Matrix

- DCT basis matrix compared to the real/imaginary parts of the DFT basis matrix ($N = 16$)



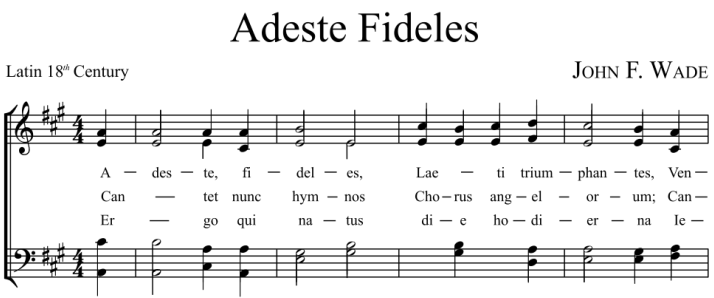
DCT vs. DFT for Compression

- DFT (real and imaginary parts) and DCT of a test signal



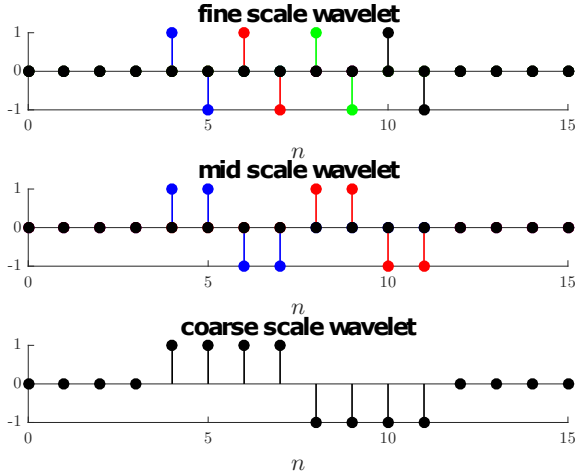
Between Time and Frequency

- Some signals are best represented neither in the time domain nor the frequency domain
- For example, many transient signals (audio, speech, images, etc.) are best represented in a domain in between time and frequency ("time-frequency", like a musical score)



Haar Wavelet Transform

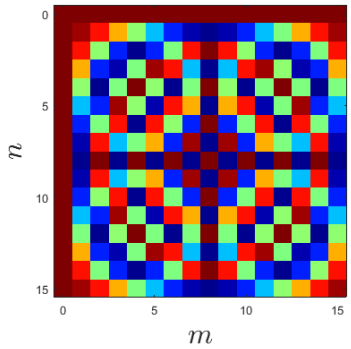
- Haar **wavelet transform** (1910): Key departure from DFT and DCT
  - Basis functions are **local** (short duration, "local waves")
  - Basis functions are **mulitscale** (many different durations) edge detectors (derivatives)



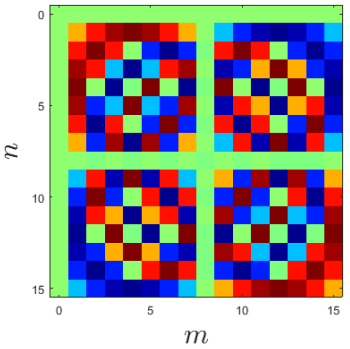
Haar Wavelet Transform Basis Matrix

- Wavelets are inside JPEG2000 image compression and many image analysis/processing systems
- Haar wavelet basis matrix compared to the real/imaginary parts of the DFT basis matrix ($N = 16$)

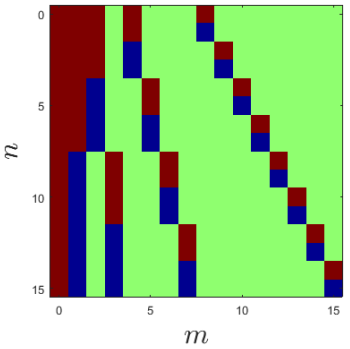
real part DFT



imaginary part DFT



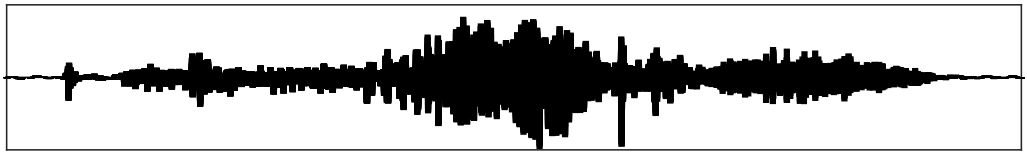
Haar wavelets



Short-Time Fourier Transform (1)

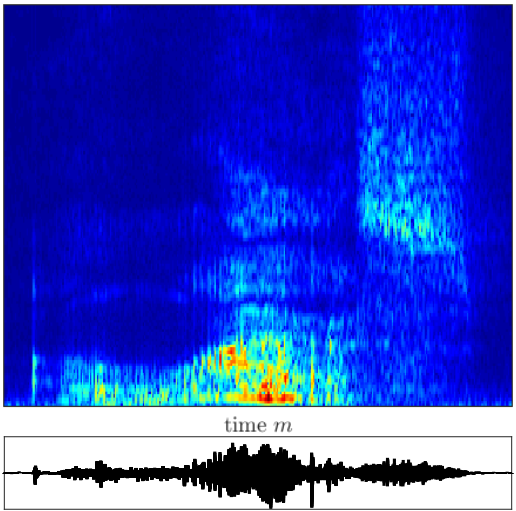
- STFT analyzes how a signal's frequency content changes over time – local Fourier analysis

- Given a signal $x[n]$ to analyze and a **window** $w[n]$
  - Window the signal with the window shifted to time m: $x[n]\, w[n-m]$
  - Compute the DFT of the windowed signal and stack as a column in an STFT matrix
  - Plot the STFT matrix as a 2D image (`imagesc` in Matlab, for example)



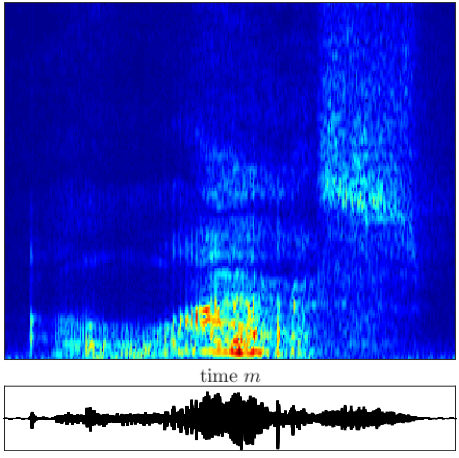
Short-Time Fourier Transform (2)

- STFT analyzes how a signal's frequency content changes over time – local Fourier analysis



Short-Time Fourier Transform (3)

- $|\text{STFT}|^2$ is called the **spectrogram** (`spectrogram` in Matlab)
- The STFT can be configured to be an orthonormal basis, but this is generally not done in practice



Summary

- The DFT is fundamental, especially for studying LTI systems
  - Only the DFT is defined in terms of basis functions that are eigenfunctions of LTI systems (harmonic sinusoids)
- But other orthogonal bases play important roles in diverse applications, especially signal analysis and compression
- Examples: DCT, wavelets, short-time Fourier transform, and beyond

The Discrete-Time Fourier Transform

Set F

Discrete Time Fourier Transform (DTFT)

Discrete Time Fourier Transform (DTFT)

- The DTFT is the Fourier transform of choice for analyzing infinite-length signals and systems
- Useful for conceptual, pencil-and-paper work, but not Matlab friendly (infinitely long vectors)
- Properties are very similar to the Discrete Fourier Transform (DFT) with a few caveats
- We will derive the DTFT as the limit of the DFT as the signal length $N \to \infty$

Recall: DFT (Unnormalized)

- **Analysis (Forward DFT)**
  - Choose the DFT coefficients $X[k]$ such that the synthesis produces the signal x
  - The weight $X[k]$ measures the similarity between x and the harmonic sinusoid s_k
  - Therefore, $X[k]$ measures the "frequency content" of x at frequency k

$$X_u[k] \;=\; \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

- **Synthesis (Inverse DFT)**
  - Build up the signal x as a linear combination of harmonic sinusoids s_k weighted by the DFT coefficients $X[k]$

$$x[n] \;=\; \frac{1}{N}\sum_{k=0}^{N-1} X_u[k]\, e^{j\frac{2\pi}{N}kn}$$

The Centered DFT

- Both $x[n]$ and $X[k]$ are **periodic** with period N, so we can shift the intervals of interest in time and frequency to be centered around $n, k = 0$

$$-\frac{N}{2} \;\leq\; n, k \;\leq\; \frac{N}{2} - 1$$

- The modified forward and inverse DFT formulas are

$$X_u[k] \;=\; \sum_{n=-N/2}^{N/2-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \qquad -\frac{N}{2} \;\leq\; k \;\leq\; \frac{N}{2} - 1$$

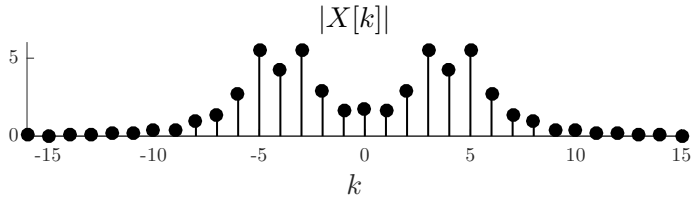
$$x[n] \;=\; \frac{1}{N} \sum_{k=-N/2}^{N/2-1} X_u[k]\, e^{j\frac{2\pi}{N}kn} \qquad -\frac{N}{2} \;\leq\; n \;\leq\; \frac{N}{2} - 1$$

Frequencies of the Centered DFT

$$X_u[k] \;=\; \sum_{n=-N/2}^{N/2-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \qquad -\frac{N}{2} \;\leq\; k \;\leq\; \frac{N}{2}-1$$

- $X_u[k]$ measures the similarity between the time signal x and the harmonic sinusoid s_k
- Therefore, $X_u[k]$ measures the "frequency content" of x at frequency

$$-\pi \;\leq\; \omega_k = \frac{2\pi}{N}k \;<\; \pi$$

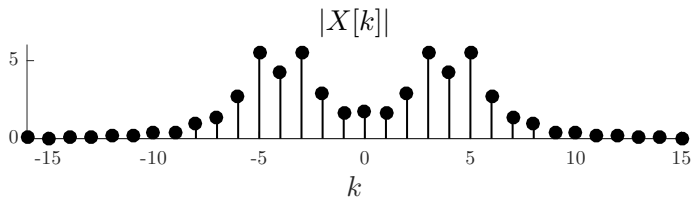


Take It To The Limit (1)

$$X_u[k] \;=\; \sum_{n=-N/2}^{N/2-1} x[n]\, e^{-j\frac{2\pi}{N}kn}, \qquad -\frac{N}{2} \;\leq\; k \;\leq\; \frac{N}{2}-1$$

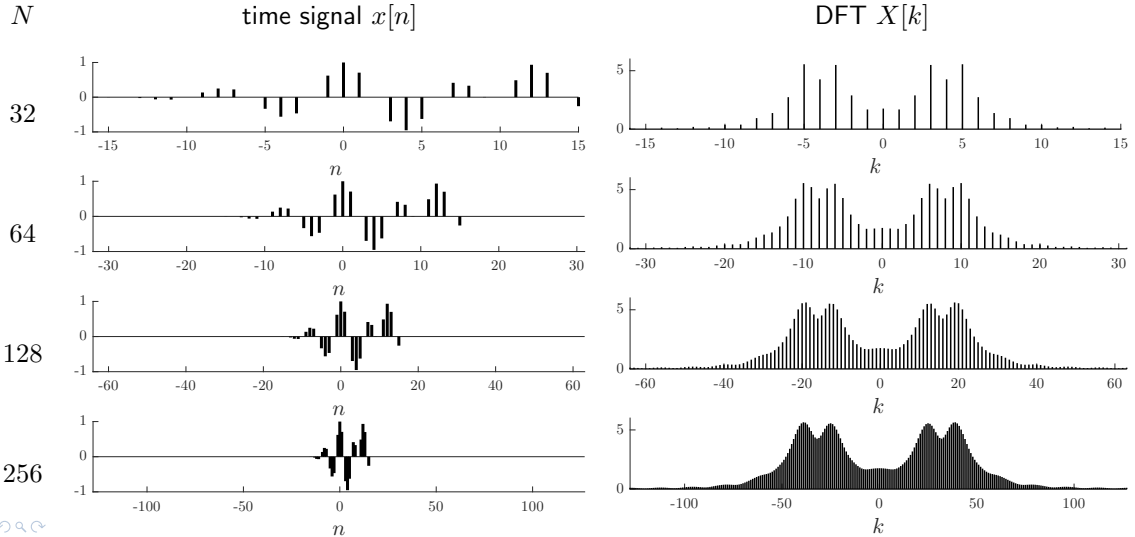
- Let the signal length N increase towards ∞ and study what happens to $X_u[k]$
- **Key fact:** No matter how large N grows, the frequencies of the DFT sinusoids remain in the interval

$$-\pi \;\leq\; \omega_k = \frac{2\pi}{N}k \;<\; \pi$$



Take It To The Limit (2)

$$X_u[k] = \sum_{n=-N/2}^{N/2-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$



Discrete Time Fourier Transform (Forward)

- As $N \to \infty$, the forward DFT converges to a function of the **continuous frequency variable** ω that we will call the **forward discrete time Fourier transform** (DTFT)

$$\sum_{n=-N/2}^{N/2-1} x[n]\, e^{-j\frac{2\pi}{N}kn} \longrightarrow \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \;=\; X(\omega), \qquad -\pi \le \omega < \pi$$

- Recall: Inner product for infinite-length signals

$$\langle x, y \rangle \;=\; \sum_{n=-\infty}^{\infty} x[n]\, y[n]^*$$

- **Analysis interpretation:** The value of the DTFT $X(\omega)$ at frequency ω measures the similarity of the infinite-length signal $x[n]$ to the infinite-length sinusoid $e^{j\omega n}$

Discrete Time Fourier Transform (Inverse)

- Inverse unnormalized DFT

$$x[n] \;=\; \frac{2\pi}{2\pi N} \sum_{k=-N/2}^{N/2-1} X_u[k]\, e^{j\frac{2\pi}{N}kn}$$

- In the limit as the signal length $N \to \infty$, the inverse DFT converges in a more subtle way:

$$e^{j\frac{2\pi}{N}kn} \longrightarrow e^{j\omega n}, \qquad X_u[k] \longrightarrow X(\omega), \qquad \frac{2\pi}{N} \longrightarrow d\omega, \qquad \sum_{k=-N/2}^{N/2-1} \longrightarrow \int_{-\pi}^{\pi}$$

  resulting in the **inverse DTFT**

$$x[n] \;=\; \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}, \qquad \infty < n < \infty$$

- **Synthesis interpretation:** Build up the signal x as an infinite linear combination of sinusoids $e^{j\omega n}$ weighted by the DTFT $X(\omega)$

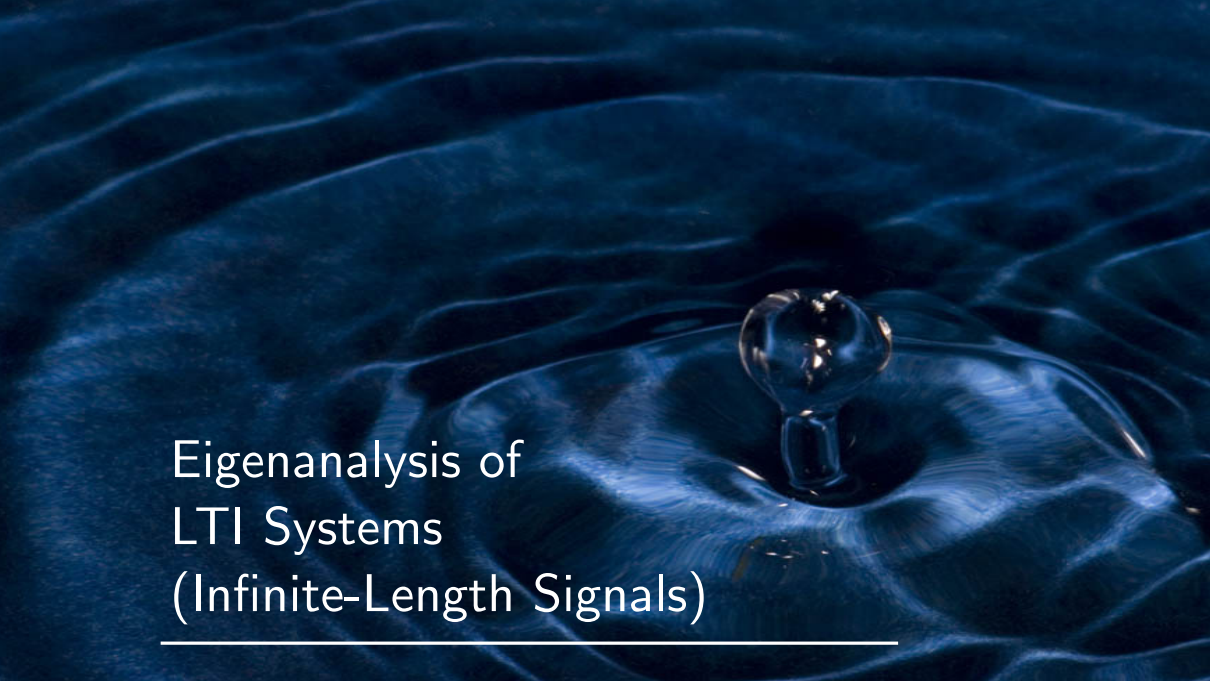
Summary

- Discrete-time Fourier transform (DTFT)

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad -\pi \leq \omega < \pi$$

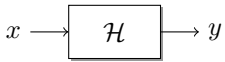
$$x[n] = \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}, \qquad \infty < n < \infty$$

- The core "basis functions" of the DTFT are the sinusoids $e^{j\omega n}$ with arbitrary frequencies ω
- The DTFT can be derived as the limit of the DFT as the signal length $N \to \infty$
- The analysis/synthesis interpretation of the DFT holds for the DTFT, as do most of its properties



Eigenanalysis of LTI Systems (Infinite-Length Signals)

LTI Systems for Infinite-Length Signals



$$y \;=\; \mathbf{H}x$$

- For infinite length signals, $\mathbf{H}$ is an infinitely large **Toeplitz matrix** with entries

$$[\mathbf{H}]_{n,m} \;=\; h[n-m]$$

  where h is the **impulse response**

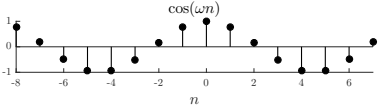
- **Goal:** Calculate the **eigenvectors** and **eigenvalues** of $\mathbf{H}$

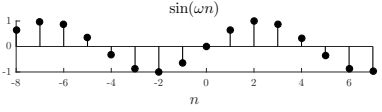
- Eigenvectors v are input signals that emerge at the system output unchanged (except for a scaling by the eigenvalue λ) and so are somehow "fundamental" to the system

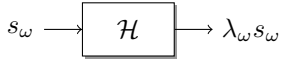
Eigenvectors of LTI Systems

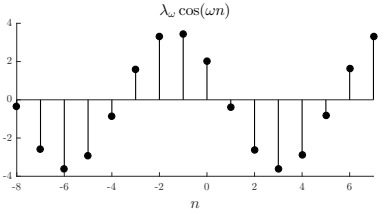
- **Fact:** The eigenvectors of a Toeplitz matrix (LTI system) are the complex **sinusoids**

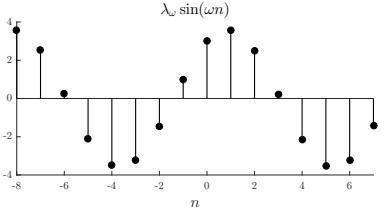
$$s_\omega[n] \;=\; e^{j\omega n} \;=\; \cos(\omega n) + j\sin(\omega n), \qquad -\pi \le \omega < \pi, \quad -\infty < n < \infty$$



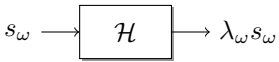








Sinusoids are Eigenvectors of LTI Systems



- Prove that harmonic sinusoids are the eigenvectors of LTI systems simply by computing the convolution with input s_ω and applying the periodicity of the sinusoids (infinite-length)

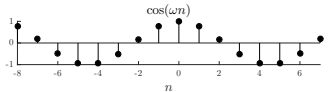
$$
\begin{aligned}
s_\omega[n] * h[n] &= \sum_{m=-\infty}^{\infty} s_\omega[n-m]\, h[m] = \sum_{m=-\infty}^{\infty} e^{j\omega(n-m)}\, h[m] \\
&= \sum_{m=-\infty}^{\infty} e^{j\omega n}\, e^{-j\omega m}\, h[m] = \left(\sum_{m=-\infty}^{\infty} h[m]\, e^{-j\omega m} \right) e^{j\omega n} \\
&= \lambda_\omega\, s_\omega[n] \quad \checkmark
\end{aligned}
$$

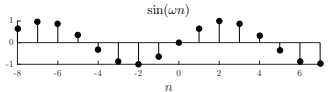
Eigenvalues of LTI Systems

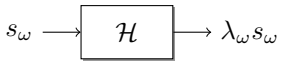
- The eigenvalue $\lambda_\omega \in \mathbb{C}$ corresponding to the sinusoid eigenvector s_ω is called the **frequency response** at frequency ω since it measures how the system "responds" to s_k

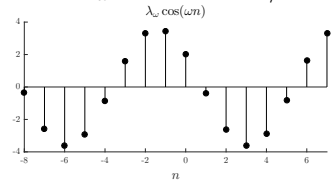
$$\lambda_\omega \;=\; \sum_{n=-\infty}^{\infty} h[n]\, e^{-j\omega n} \;=\; \langle h, s_\omega \rangle \;=\; H(\omega) \;\; \text{(DTFT of } h\text{)}$$

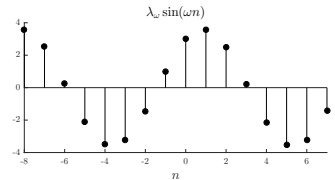
- Recall properties of the **inner product**: λ_ω grows/shrinks as h and s_ω become more/less similar



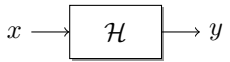








Eigendecomposition and Diagonalization of an LTI System



$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- While we can't explicitly display the infinitely large matrices involved, we can use the DTFT to "diagonalize" an LTI system
- Taking the DTFTs of x and h

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad H(\omega) = \sum_{n=-\infty}^{\infty} h[n]\, e^{-j\omega n}$$

we have that

$$Y(\omega) \;=\; X(\omega) H(\omega)$$

and then

$$y[n] = \int_{-\pi}^{\pi} Y(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}$$

Summary

- Complex sinusoids are the eigenfunctions of LTI systems for infinite-length signals (Toeplitz matrices)
- Therefore, the discrete time Fourier transform (DTFT) is the natural tool for studying LTI systems for infinite-length signals
- Frequency response $H(\omega)$ equals the DTFT of the impulse response $h[n]$
- Diagonalization by eigendecomposition implies

$$Y(\omega) \;=\; X(\omega)\, H(\omega)$$

Discrete Time Fourier Transform Examples

Discrete Time Fourier Transform

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad -\pi \leq \omega < \pi$$

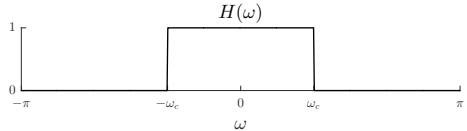
$$x[n] = \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}, \qquad \infty < n < \infty$$

- The Fourier transform of choice for analyzing infinite-length signals and systems
- Useful for conceptual, pencil-and-paper work, but not Matlab friendly (infinitely long vectors)

Impulse Response of the Ideal Lowpass Filter (1)

- The frequency response $H(\omega)$ of the ideal low-pass filter passes low frequencies (near $\omega = 0$) but blocks high frequencies (near $\omega = \pm\pi$)

$$H(\omega) = \begin{cases} 1 & -\omega_c \le \omega \le \omega_c \\ 0 & \text{otherwise} \end{cases}$$



- Compute the impulse response $h[n]$ given this $H(\omega)$

- Apply the inverse DTFT

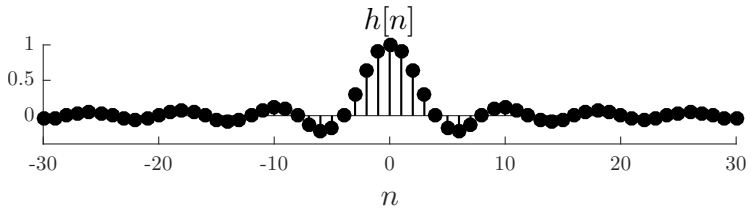
$$h[n] = \int_{-\pi}^{\pi} H(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi} = \int_{-\omega_c}^{\omega_c} e^{j\omega n}\, \frac{d\omega}{2\pi} = \left. \frac{e^{j\omega n}}{2\pi j n} \right|_{-\omega_c}^{\omega_c} = \frac{e^{j\omega_c n} - e^{-j\omega_c n}}{2\pi j n} = \frac{\omega_c}{\pi} \frac{\sin(\omega_c n)}{\omega_c n}$$

Impulse Response of the Ideal Lowpass Filter (2)

- The frequency response $H(\omega)$ of the ideal low-pass filter passes low frequencies (near $\omega = 0$) but blocks high frequencies (near $\omega = \pm\pi$)

$$H(\omega) = \begin{cases} 1 & -\omega_c \leq |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases}$$

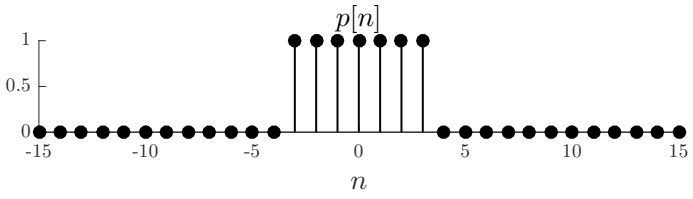
$$h[n] \;=\; 2\omega_c \frac{\sin(\omega_c n)}{\omega_c n}$$



- The infamous "sinc" function!

DTFT of a Moving Average System

- Compute the DTFT of the symmetrical **moving average system** $p[n] = \begin{cases} 1 & -M \le n \le M \\ 0 & \text{otherwise} \end{cases}$
- Note: Duration $D_x = 2M + 1$ samples
- Example for $M = 3$



- Forward DTFT

$$P(\omega) = \sum_{n=-\infty}^{\infty} p[n]\, e^{-j\omega n} = \sum_{n=-M}^{M} e^{-j\omega n} \quad \ldots$$

DTFT of the Unit Pulse (2)

- Apply the finite geometric series formula

$$P(\omega) = \sum_{n=-\infty}^{\infty} p[n]\, e^{-j\omega n} = \sum_{n=-M}^{M} e^{-j\omega n} = \sum_{n=-M}^{M} \left(e^{-j\omega}\right)^n = \frac{e^{j\omega M} - e^{-j\omega(M+1)}}{1 - e^{-j\omega}}$$

- This is an answer but it is not simplified enough to make sense, so we continue simplifying

$$\begin{aligned} P(\omega) &= \frac{e^{j\omega M} - e^{-j\omega(M+1)}}{1 - e^{-j\omega}} = \frac{e^{-j\omega/2}\left(e^{j\omega\frac{2M+1}{2}} - e^{-j\omega\frac{2M+1}{2}}\right)}{e^{-j\omega/2}\left(e^{j\omega/2} - e^{-j\omega/2}\right)} \\ &= \frac{2j\sin\left(\omega\frac{2M+1}{2}\right)}{2j\sin\left(\frac{\omega}{2}\right)} \end{aligned}$$

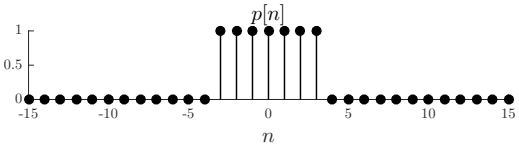
DTFT of the Unit Pulse (3)

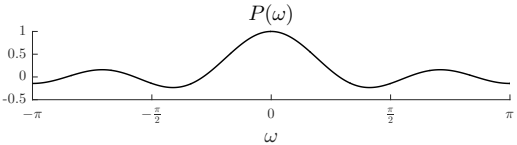
- Simplified DTFT of the unit pulse of duration $D_x = 2M + 1$ samples

$$P(\omega) \;=\; \frac{\sin\left(\frac{2M+1}{2}\,\omega\right)}{\sin\left(\frac{\omega}{2}\right)}$$

- This is called the **Dirichlet kernel** or "digital sinc"
  - It has a shape reminiscent of the classical $\sin x/x$ sinc function, but it is 2π-periodic

- If $p[n]$ is interpreted as the impulse response of the moving average system, then $P(\omega)$ is the frequency response (eigenvalues) (low-pass filter)



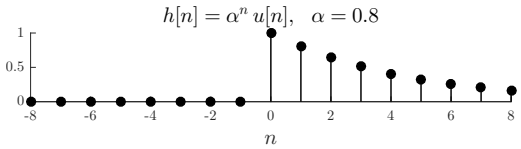


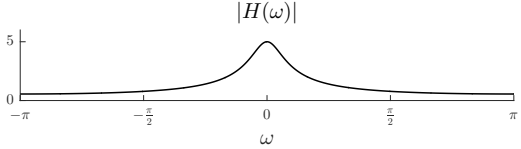
DTFT of a One-Sided Exponential

- Recall the impulse response of the recursive average system: $h[n] = \alpha^n\, u[n],\ |\alpha| < 1$
- Compute the frequency response $H(\omega)$
- Forward DTFT

$$H(\omega) \;=\; \sum_{n=-\infty}^{\infty} h[n]\, e^{-j\omega n} \;=\; \sum_{n=0}^{\infty} \alpha^n\, e^{-j\omega n} \;=\; \sum_{n=0}^{\infty} (\alpha\, e^{-j\omega})^n \;=\; \frac{1}{1-\alpha\, e^{-j\omega}}$$

- Recursive system with $\alpha = 0.8$ is a low-pass filter





Summary

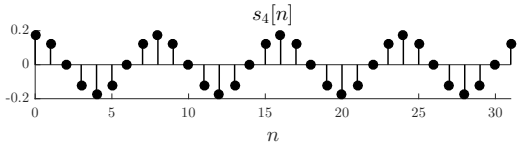
- DTFT of a rectangular pulse is a Dirichlet kernel
- DTFT of a one-sided exponential is a low-frequency bump
- Inverse DTFT of the ideal lowpass filter is a sinc function
- Work some examples on your own!

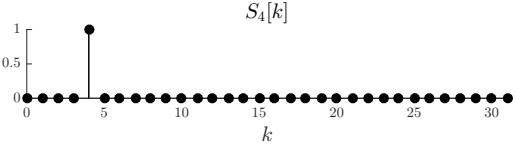
Discrete Time Fourier Transform of a Sinusoid

Discrete Fourier Transform (DFT) of a Harmonic Sinusoid

- Thanks to the orthogonality of the length-N harmonic sinusoids, it is easy to calculate the DFT of the harmonic sinusoid $x[n] = s_l[n] = e^{j\frac{2\pi}{N}ln}/\sqrt{N}$

$$X[k] \;=\; \sum_{n=0}^{N-1} s_l[n]\,\frac{e^{-j\frac{2\pi}{N}kn}}{\sqrt{N}} \;=\; \langle s_l, s_k\rangle \;=\; \delta[k-l]$$





- So what is the DTFT of the infinite length sinusoid $e^{j\omega_0 n}$?

DTFT of an Infinite-Length Sinusoid

- The calculation for the DTFT and infinite-length signals is much more delicate than for the DFT and finite-length signals

- Calculate the value $X(\omega_0)$ for the signal $x[n] = e^{j\omega_0 n}$

$$X(\omega_0) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega_0 n} \;=\; \sum_{n=-\infty}^{\infty} e^{j\omega_0 n}\, e^{-j\omega_0 n} \;=\; \sum_{n=-\infty}^{\infty} 1 \;=\; \infty$$

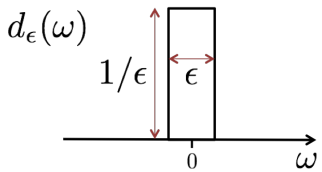
- Calculate the value $X(\omega)$ for the signal $x[n] = e^{j\omega_0 n}$ at a frequency $\omega \neq \omega_0$

$$X(\omega_0) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n} \;=\; \sum_{n=-\infty}^{\infty} e^{j\omega_0 n}\, e^{-j\omega n} \;=\; \sum_{n=-\infty}^{\infty} e^{-j(\omega-\omega_0)n} \;=\; ???$$

Dirac Delta Function (1)

- One semi-rigorous way to deal with this quandary is to use the **Dirac delta "function,"** which is defined in terms of the following limit process

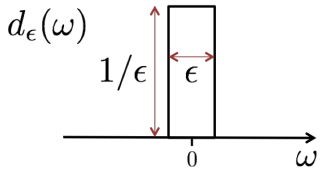
- Consider the following function $d_\epsilon(\omega)$ of the continuous variable ω



- Note that, for all values of the width ϵ, $d_\epsilon(\omega)$ always has unit area

$$\int d_\epsilon(\omega)\, d\omega \;=\; 1$$

Dirac Delta Function (2)

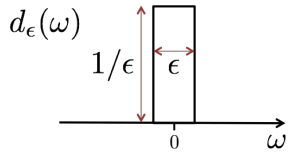


- What happens to $d_\epsilon(\omega)$ as we let $\epsilon \to 0$?
  - Clearly $d_\epsilon(\omega)$ is converging toward something that is infinitely tall and infinitely narrow but still with unit area

- The safest way to handle a function like $d_\epsilon(\omega)$ is inside an integral, like so

$$\int X(\omega)\, d_\epsilon(\omega)\, d\omega$$

Dirac Delta Function (3)



- As $\epsilon \to 0$, it seems reasonable that

$$\int X(\omega)\, d_\epsilon(\omega)\, d\omega \overset{\epsilon \to 0}{\longrightarrow} X(0)$$

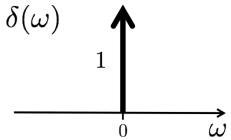
and

$$\int X(\omega)\, d_\epsilon(\omega - \omega_0)\, d\omega \overset{\epsilon \to 0}{\longrightarrow} X(\omega_0)$$

- So we can think of $d_\epsilon(\omega)$ as a kind of "sampler" that picks out values of functions from inside an integral
- We describe the results of this limiting process (as $\epsilon \to 0$) as the **Dirac delta "function"** $\delta(\omega)$

Dirac Delta Function (4)

- **Dirac delta "function"** $\delta(\omega)$



- We write

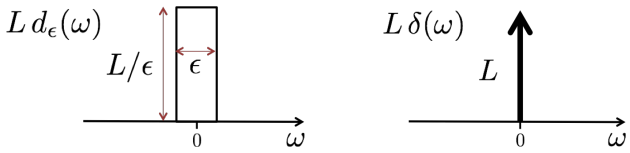
$$\int X(\omega)\,\delta(\omega)\,d\omega \;=\; X(0)$$

and

$$\int X(\omega)\,\delta(\omega-\omega_0)\,d\omega \;=\; X(\omega_0)$$

- Remarks and caveats
  - Do not confuse the Dirac delta "function" with the nicely behaved discrete delta function $\delta[n]$
  - The Dirac has lots of "delta," but it is not really a "function" in the normal sense (it can be made more rigorous using the theory of generalized functions)
  - Colloquially, engineers will describe the Dirac delta as "infinitely tall and infinitely narrow"

Scaled Dirac Delta Function



- If we scale the area of $d_\epsilon(\omega)$ by L, then it has the following effect in the limit

$$\int X(\omega)\, L\, \delta(\omega)\, d\omega \;=\; L\, X(0)$$

And Now Back to Our Regularly Scheduled Program . . .

- Back to determining the DTFT of an infinite length sinusoid

- Rather than computing the DTFT of a sinusoid using the forward DTFT, we will show that an infinite-length sinusoid is the inverse DTFT of the scaled Dirac delta function $2\pi\delta(\omega - \omega_0)$

$$\int_{-\pi}^{\pi} 2\pi\delta(\omega - \omega_0)\, e^{j\omega n}\, \frac{d\omega}{2\pi} \;=\; e^{j\omega_0 n}$$

- Thus we have the (rather bizarre) DTFT pair

$$e^{j\omega_0 n} \;\overset{\text{DTFT}}{\longleftrightarrow}\; 2\pi\,\delta(\omega - \omega_0)$$

DTFT of Real-Valued Sinusoids

- Since

$$\cos(\omega_0 n) \;=\; \frac{1}{2}\left(e^{j\omega_0 n} + e^{-j\omega_0 n}\right)$$

we can calculate its DTFT as

$$\cos(\omega_0 n) \;\overset{\text{DTFT}}{\longleftrightarrow}\; \pi\,\delta(\omega-\omega_0) + \pi\,\delta(\omega+\omega_0)$$

- Since

$$\sin(\omega_0 n) \;=\; \frac{1}{2j}\left(e^{j\omega_0 n} - e^{-j\omega_0 n}\right)$$

we can calculate its DTFT as

$$\sin(\omega_0 n) \;\overset{\text{DTFT}}{\longleftrightarrow}\; \frac{\pi}{j}\,\delta(\omega-\omega_0) + \frac{\pi}{j}\,\delta(\omega+\omega_0)$$

Summary

- The DTFT would be of limited utility if we could not compute the transform of an infinite-length sinusoid
- Hence, the Dirac delta "function" (or something else) is a necessary evil
- The Dirac delta has infinite energy (2-norm); but then again so does an infinite-length sinusoid

Discrete Time Fourier Transform Properties

Recall: Discrete-Time Fourier Transform (DTFT)

- Forward DTFT (Analysis)

$$X(\omega) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad -\pi \le \omega < \pi$$

- Inverse DTFT (Synthesis)

$$x[n] \;=\; \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}, \qquad \infty < n < \infty$$

- DTFT pair

$$x[n] \;\overset{\text{DTFT}}{\longleftrightarrow}\; X(\omega)$$

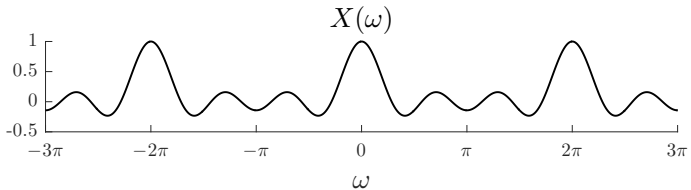
The DTFT is Periodic

- We defined the DTFT over an interval of ω of length 2π, but it can also be interpreted as **periodic** with period 2π

$$X(\omega) \;=\; X(\omega + 2\pi k), \quad k \in \mathbb{Z}$$

- Proof

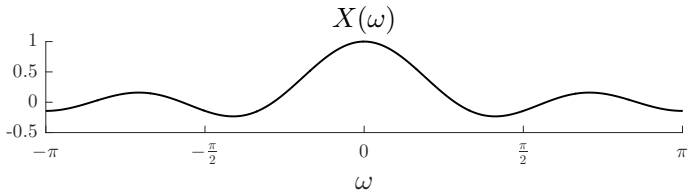
$$X(\omega + 2\pi k) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j(\omega+2\pi k)n} \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}\, e^{-j2\pi kn} \;=\; X(\omega) \quad \checkmark$$



DTFT Frequencies

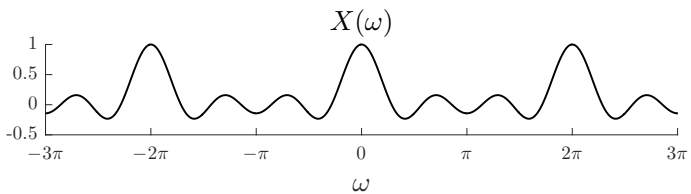
$$X(\omega) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad -\pi \le \omega < \pi$$

- $X(\omega)$ measures the similarity between the time signal x and and a sinusoid $e^{j\omega n}$ of frequency ω
- Therefore, $X(\omega)$ measures the "frequency content" of x at frequency ω



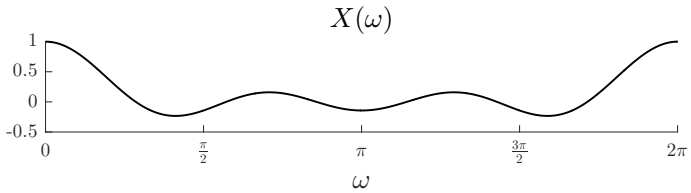
DTFT Frequencies and Periodicity

- Periodicity of DTFT means we can treat frequencies mod 2π
- $X(\omega)$ measures the "frequency content" of x at frequency $(\omega)_{2\pi}$

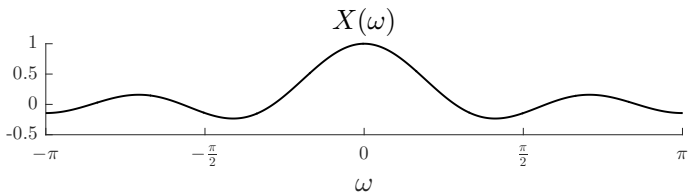


DTFT Frequency Ranges

- Periodicity of DTFT means every length-2π interval of ω carries the same information
- Typical interval 1: $0 \leq \omega < 2\pi$



- Typical interval 2: $-\pi \leq \omega < \pi$ (more intuitive)



DTFT and Time Shift

- If $x[n]$ and $X(\omega)$ are a DTFT pair then

$$x[n-m] \overset{\text{DTFT}}{\longleftrightarrow} e^{-j\omega m} X(\omega)$$

- Proof: Use the change of variables $r = n - m$

$$\sum_{n=-\infty}^{\infty} x[n-m]\, e^{-j\omega n} = \sum_{r=-\infty}^{\infty} x[r]\, e^{-j\omega(r+m)} = \sum_{r=-\infty}^{\infty} x[r]\, e^{-j\omega r}\, e^{-j\omega m}$$

$$= e^{-j\omega m} \sum_{r=-\infty}^{\infty} x[r]\, e^{-j\omega r} = e^{-j\omega m}\, X(\omega) \quad \checkmark$$

DTFT and Modulation

- If $x[n]$ and $X(\omega)$ are a DTFT pair then

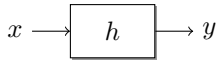
$$e^{j\omega_0 n}\, x[n] \overset{\text{DTFT}}{\longleftrightarrow} X(\omega - \omega_0)$$

- Remember that the DTFT is 2π-periodic, and so we can interpret the right hand side as $X((\omega - \omega_0)_{2\pi})$

- Proof:

$$\sum_{n=-\infty}^{\infty} e^{j\omega_0 n}\, x[n]\, e^{-j\omega n} \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j(\omega-\omega_0)n} \;=\; X(\omega - \omega_0) \quad \checkmark$$

DTFT and Convolution



$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- If

$$x[n] \overset{\text{DTFT}}{\longleftrightarrow} X(\omega), \qquad h[n] \overset{\text{DTFT}}{\longleftrightarrow} H(\omega), \qquad y[n] \overset{\text{DTFT}}{\longleftrightarrow} Y(\omega)$$

  then

$$Y(\omega) \;=\; H(\omega)\, X(\omega)$$

- Convolution in the time domain $\Leftrightarrow$ multiplication in the frequency domain

The DTFT is Linear

- It is trivial to show that if

$$x_1[n] \overset{\text{DTFT}}{\longleftrightarrow} X_1(\omega) \qquad x_2[n] \overset{\text{DTFT}}{\longleftrightarrow} X_2(\omega)$$

then

$$\alpha_1 x_1[n] + \alpha_2 x[2] \overset{\text{DTFT}}{\longleftrightarrow} \alpha_1 X_1(\omega) + \alpha_2 X_2(\omega)$$

DTFT Symmetry Properties

- The sinusoids $e^{j\omega n}$ of the DTFT have symmetry properties:

$$\begin{aligned} \mathrm{Re}\left(e^{j\omega n}\right) &= \cos(\omega n) \quad \text{(even function)} \\ \mathrm{Im}\left(e^{j\omega n}\right) &= \sin(\omega n) \quad \text{(odd function)} \end{aligned}$$

- These induce corresponding symmetry properties on $X(\omega)$ around the frequency $\omega = 0$

- **Even** signal/DFT

$$x[n] = x[-n], \qquad X(\omega) = X(-\omega)$$

- **Odd** signal/DFT

$$x[n] = -x[-n], \qquad X(\omega) = -X(-\omega)$$

- Proofs of the symmetry properties are identical to the DFT case; omitted here

DTFT Symmetry Properties Table

$x[n]$	$X(\omega)$	$\mathrm{Re}(X(\omega))$	$\mathrm{Im}(X(\omega))$	$\lvert X(\omega)\rvert$	$\angle X(\omega)$
real	$X(-\omega) = X(\omega)^*$	even	odd	even	odd
real & even	real & even	even	zero	even	
real & odd	imaginary & odd	zero	odd	even	
imaginary	$X(-\omega) = -X(\omega)^*$	odd	even	even	odd
imaginary & even	imaginary & even	zero	even	even	
imaginary & odd	real & odd	odd	zero	even	

Summary

- DTFT is periodic with period 2π
- Convolution in time becomes multiplication in frequency
- DTFT has useful symmetry properties

z-Transform

Set G

z-Transform

z-Transform

- The z-transform generalizes the Discrete-Time Fourier Transform (DTFT) for analyzing infinite-length signals and systems
- Very useful for designing and analyzing signal processing systems
- Properties are very similar to the DTFT with a few caveats
- The theme this week is less linear algebra (vectors spaces) and more polynomial algebra

Recall: DTFT

- Discrete-time Fourier transform (DTFT)

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad -\pi \leq \omega < \pi$$

$$x[n] = \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}, \qquad \infty < n < \infty$$

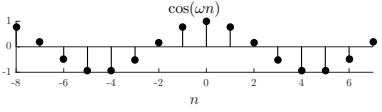
- The core "basis functions" of the DTFT are the sinusoids $e^{j\omega n}$ with arbitrary frequencies ω

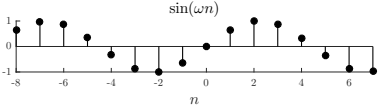
- The sinusoids $e^{j\omega n}$ are eigenvectors of LTI systems for infinite-length signals (infinite Toeplitz matrices)

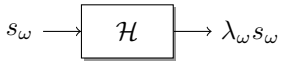
Recall: Complex Sinusoids are Eigenvectors of LTI Systems

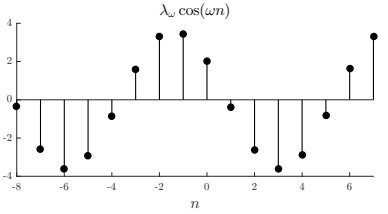
- **Fact:** The eigenvectors of a Toeplitz matrix (LTI system) are the complex **sinusoids**

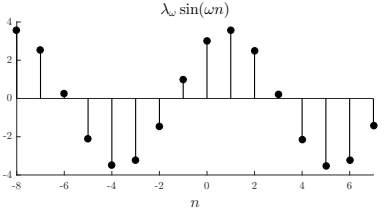
$$s_\omega[n] \;=\; e^{j\omega n} \;=\; \cos(\omega n) + j\sin(\omega n), \qquad -\pi \le \omega < \pi, \quad -\infty < n < \infty$$









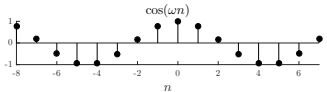


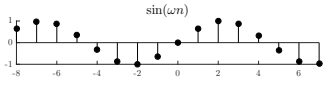
Recall: Eigenvalues of LTI Systems

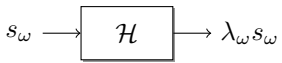
- The eigenvalue $\lambda_\omega \in \mathbb{C}$ corresponding to the sinusoid eigenvector s_ω is called the **frequency response** at frequency ω since it measures how the system "responds" to s_k

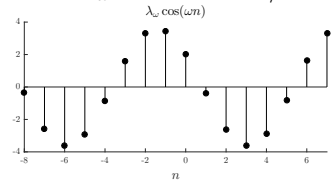
$$\lambda_\omega \;=\; \sum_{n=-\infty}^{\infty} h[n]\, e^{-j\omega n} \;=\; \langle h, s_\omega \rangle \;=\; H(\omega) \;\; \text{(DTFT of } h\text{)}$$

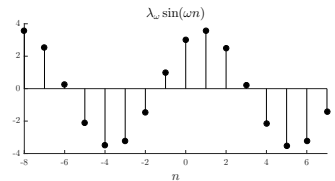
- Recall properties of the **inner product**: λ_ω grows/shrinks as h and s_ω become more/less similar



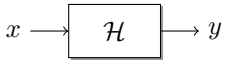








Eigendecomposition and Diagonalization of an LTI System



$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- While we can't explicitly display the infinitely large matrices involved, we can use the DTFT to "diagonalize" an LTI system
- Taking the DTFTs of x and h

$$X(\omega) = \sum_{m=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad H(\omega) = \sum_{m=-\infty}^{\infty} h[n]\, e^{-j\omega n}$$

we have that

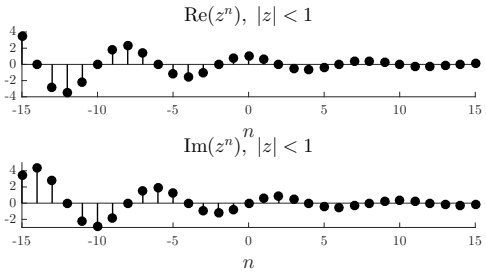
$$Y(\omega) \;=\; X(\omega)H(\omega)$$

Recall: Complex Exponential

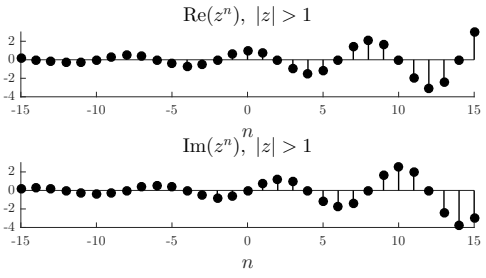
$$z^n \;=\; \left(|z|\, e^{j\omega}\right)^n \;=\; |z|^n\, e^{j\omega n} \;=\; |z|^n\, \left(\cos(\omega n) + j\sin(\omega n)\right)$$

- $|z|^n$ is a **real exponential** envelope $\left(a^n \text{ with } a = |z|\right)$
- $e^{j\omega n}$ is a **complex sinusoid**

$|z| < 1$

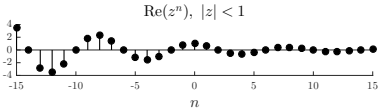


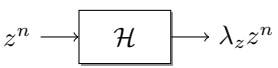
$|z| > 1$

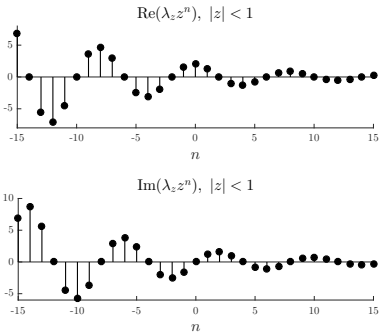


Complex Exponentials are Eigenvectors of LTI Systems

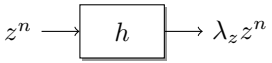
- **Fact:** A more general set of eigenvectors of a Toeplitz matrix (LTI system) are the **complex exponentials** z^n, $z \in \mathbb{C}$







Proof: Complex Exponentials are Eigenvectors of LTI Systems



- Prove that complex exponentials are the eigenvectors of LTI systems simply by computing the convolution with input z^n

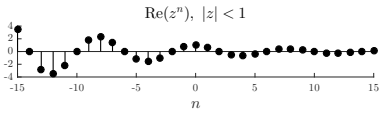
$$
\begin{aligned}
z^n * h[n] &= \sum_{m=-\infty}^{\infty} z^{n-m}\, h[m] = \sum_{m=-\infty}^{\infty} z^n\, z^{-m}\, h[m] \\
&= \left(\sum_{m=-\infty}^{\infty} h[m]\, z^{-m} \right) z^n \\
&= \lambda_z\, z^n \quad \checkmark
\end{aligned}
$$

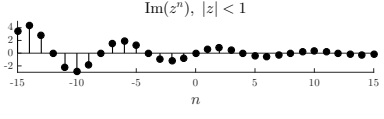
Eigenvalues of LTI Systems

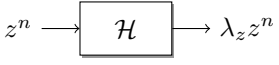
- The eigenvalue $\lambda_z \in \mathbb{C}$ corresponding to the complex exponential eigenvector z^n is called the **transfer function**; it measures how the system "transfers" the input z^n to the output

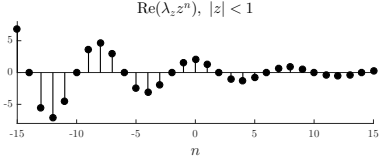
$$\lambda_z \;=\; \sum_{n=-\infty}^{\infty} h[n]\, z^{-n} \;=\; H(z)$$

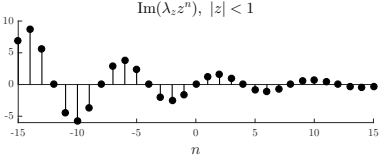
- Recall properties of the **inner product**: λ_z grows/shrinks as $h[n]$ becomes more/less similar to $(z^{-n})^*$











z-Transform

- Define the (bilateral) **forward z-transform** of $x[n]$ as

$$X(z) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}$$

- The core "basis functions" of the z-transform are the complex exponentials z^n with arbitrary $z \in \mathbb{C}$; these are the eigenvectors of LTI systems for infinite-length signals

- **Notation abuse alert:** We use $X(\cdot)$ to represent both the DTFT $X(\omega)$ and the z-transform $X(z)$; they are, in fact, intimately related

$$X_{\mathrm{DTFT}}(\omega) \;=\; X_z(z)|_{z=e^{j\omega}} \;=\; X_z(e^{j\omega})$$

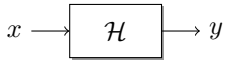
(Note how this elegantly produces a 2π-periodic DTFT)

z-Transform as a Function

$$X(z) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}$$

- $X(z)$ is a complex-valued function of a complex variable: $\quad X(z) \in \mathbb{C}, \quad z \in \mathbb{C}$

Eigendecomposition and Diagonalization of an LTI System



$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

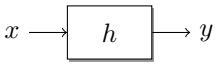
- While we can't explicitly display the infinitely large matrices involved, we can use the z-transform to "diagonalize" an LTI system
- Taking the z-transforms of x and h

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}, \qquad H(z) = \sum_{n=-\infty}^{\infty} h[n]\, z^{-n}$$

we have that

$$Y(z) \;=\; X(z)\, H(z)$$

Proof: Eigendecomposition and Diagonalization of an LTI System



- Compute the z-transform of the result of the convolution of x and h
  (Note: we are a little cavalier about exchanging the infinite sums below)

$$
\begin{aligned}
Y(z) &= \sum_{n=-\infty}^{\infty} y[n]\, z^{-n} = \sum_{n=-\infty}^{\infty} \left(\sum_{m=-\infty}^{\infty} x[m]\, h[n-m] \right) z^{-n} \\
&= \sum_{m=-\infty}^{\infty} x[m] \left(\sum_{n=-\infty}^{\infty} h[n-m]\, z^{-n} \right) \quad (\text{let } r = n-m) \\
&= \sum_{m=-\infty}^{\infty} x[m] \left(\sum_{r=-\infty}^{\infty} h[r]\, z^{-r-m} \right) = \left(\sum_{m=-\infty}^{\infty} x[m] z^{-m} \right) \left(\sum_{r=-\infty}^{\infty} h[r]\, z^{-r} \right) \\
&= X(z)\, H(z) \quad \checkmark
\end{aligned}
$$

Summary

- Complex exponentials z^n are the eigenfunctions of LTI systems for infinite-length signals (Toeplitz matrices)
- Forward z-transform

$$X(z) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}$$

- Transfer function $H(z)$ equals the z-transform of the impulse response $h[n]$
- Diagonalization by eigendecomposition implies

$$Y(z) \;=\; X(z)\, H(z)$$

- DTFT is a special case of the z-transform (values on the unit circle in the complex z-plane)

$$X_{\mathrm{DTFT}}(\omega) \;=\; X_z(z)\big|_{z=e^{j\omega}} \;=\; X_z(e^{j\omega})$$

(Note how this elegantly produces a 2π-periodic DTFT)

z-Transform
Region of Convergence

z-Transform

- The **forward z-transform** of the signal $x[n]$

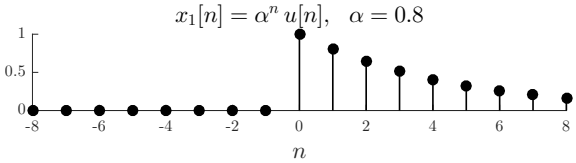
$$X(z) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}$$

- We have been a bit cavalier in our development regarding the convergence of the infinite sum that defines $X(z)$

- Work out two examples to expose the issues

Example 1: z-Transform of $\alpha^n u[n]$

- Signal $x_1[n] = \alpha^n u[n]$, $\alpha \in \mathbb{C}$ (causal signal)
- Example for $\alpha = 0.8$



- The **forward z-transform** of $x_1[n]$

$$X_1(z) \;=\; \sum_{n=-\infty}^{\infty} x_1[n]\, z^{-n} \;=\; \sum_{n=0}^{\infty} \alpha^n\, z^{-n} \;=\; \sum_{n=0}^{\infty} (\alpha\, z^{-1})^n \;=\; \frac{1}{1-\alpha\, z^{-1}} \;=\; \frac{z}{z-\alpha}$$

- **Important:** We can apply the geometric sum formula only when $|\alpha\, z^{-1}| < 1$ or $|z| > |\alpha|$

Region of Convergence

DEFINITION

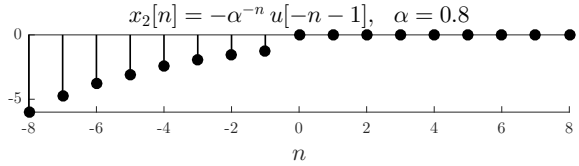
Given a time signal $x[n]$, the **region of convergence** (ROC) of its z-transform $X(z)$ is the set of $z \in \mathbb{C}$ such that $X(z)$ converges, that is, the set of $z \in \mathbb{C}$ such that $x[n]\, z^{-n}$ is absolutely summable

$$\sum_{n=-\infty}^{\infty} |x[n]\, z^{-n}| \; < \; \infty$$

- Example: For $x_1[n] = \alpha^n u[n]$, $\alpha \in \mathbb{C}$, the ROC of $X_1(z) = \frac{1}{1-\alpha\, z^{-1}}$ is all z such that $|z| > |\alpha|$

Example 2: z-Transform of $-\alpha^n u[-n-1]$

- Signal $x_2[n] = -\alpha^n u[-n-1]$, $\alpha \in \mathbb{C}$ (anti-causal signal)
- Example for $\alpha = 0.8$



- The **forward z-transform** of $x_2[n]$

$$X_2(z) = \sum_{n=-\infty}^{\infty} x_2[n]\, z^{-n} = \sum_{n=-\infty}^{-1} -\alpha^n\, z^{-n} = -\sum_{n=1}^{\infty} \alpha^{-n}\, z^n = -\sum_{n=0}^{\infty} (\alpha^{-1}\, z)^n + 1$$

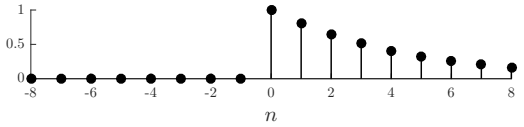
$$= \frac{-1}{1-\alpha^{-1}\, z} + 1 = \frac{-1}{1-\alpha^{-1}\, z} + \frac{1-\alpha^{-1}\, z}{1-\alpha^{-1}\, z} = \frac{-\alpha^{-1} z}{1-\alpha^{-1}\, z} = \frac{1}{1-\alpha\, z^{-1}} = \frac{z}{z-\alpha}$$

- ROC: $|\alpha^{-1}\, z| < 1$ or $|z| < |\alpha|$

Example Summary

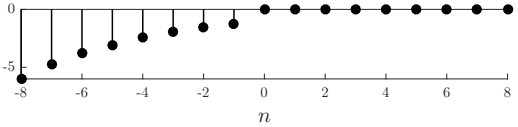
$$x_1[n] = \alpha^n u[n]$$

causal signal



$$x_2[n] = -\alpha^n u[-n-1]$$

anti-causal signal



$$X_1(z) \;=\; \frac{1}{1-\alpha\, z^{-1}} \;=\; X_2(z)$$

ROC

$|z| > |\alpha|$

ROC

$|z| < |\alpha|$

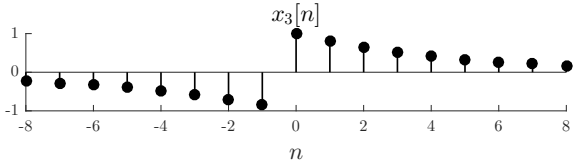
ROC: What We've Learned So Far

- The ROC is important
  - A z-transform converges only for certain values of z and does not exist for other values of z
  - z-transforms are non-unique without it

- You must always state the ROC when you state a z-transform

Example 3: z-Transform of $\alpha_1^n u[n] - \alpha_2^n u[-n-1]$

- Signal $x_3[n] = \alpha_1^n u[n] - \alpha_2^n u[-n-1]$, $\alpha_1, \alpha_2 \in \mathbb{C}$
- Example for $\alpha_1 = 0.8$ and $\alpha_2 = 1.2$



- The **forward z-transform** of the signal $x_3[n]$

$$X_3(z) = \sum_{n=-\infty}^{\infty} x_3[n]\, z^{-n} = \sum_{n=-\infty}^{\infty} \alpha_1^n u[n]\, z^{-n} - \sum_{n=-\infty}^{\infty} \alpha_2^n u[-n-1]\, z^{-n}$$

$$= \frac{1}{1-\alpha_1\, z^{-1}} + \frac{1}{1-\alpha_2\, z^{-1}}$$

- ROC: $|z| > |\alpha_1|$ and $|z| < |\alpha_2|$

Example 3: A Tale of Two ROCs

- Signal $x_3[n] = \alpha_1^n u[n] - \alpha_2^n u[-n-1]$, $\alpha_1, \alpha_2 \in \mathbb{C}$
- z-transform

$$X_3(z) \;=\; \frac{1}{1-\alpha_1\, z^{-1}} \;+\; \frac{1}{1-\alpha_2\, z^{-1}}, \qquad |\alpha_1| < |z| < |\alpha_2|$$

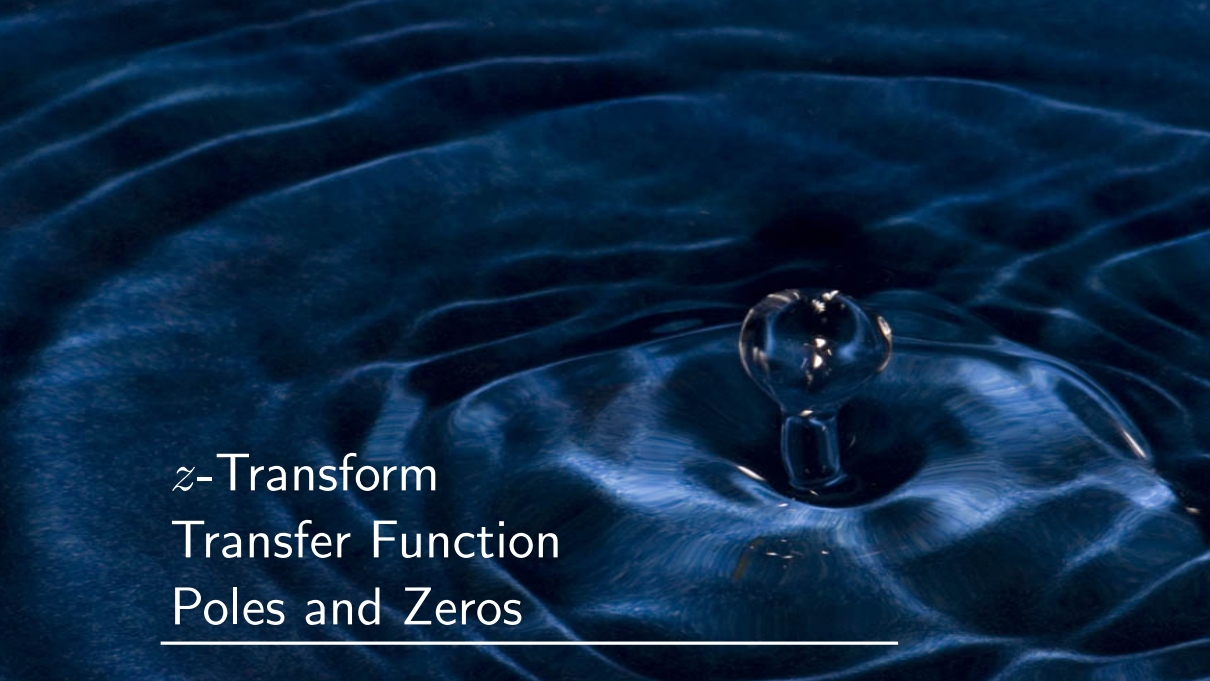
- Case 1: $|\alpha_1| < |\alpha_2|$
- Case 2: $|\alpha_1| > |\alpha_2|$

Properties of the ROC

- The ROC is a connected annulus (doughnut) in the z-plane centered on the origin $z = 0$; that is, the ROC is of the form $r_1 < |z| < r_2$
- If $x[n]$ has finite duration, then the ROC is the entire z-plane (except possibly $z = 0$ or $z = \infty$)
- If $x[n]$ is causal, then the ROC is the outside of a disk
- If $x[n]$ is anti-causal, then the ROC is the inside of a disk
- If $x[n]$ is two-sided (neither causal nor anti-causal), then either the ROC is the inside of an annulus or the z-transform does not converge
- The ROC is a connected region of the z-plane

Summary

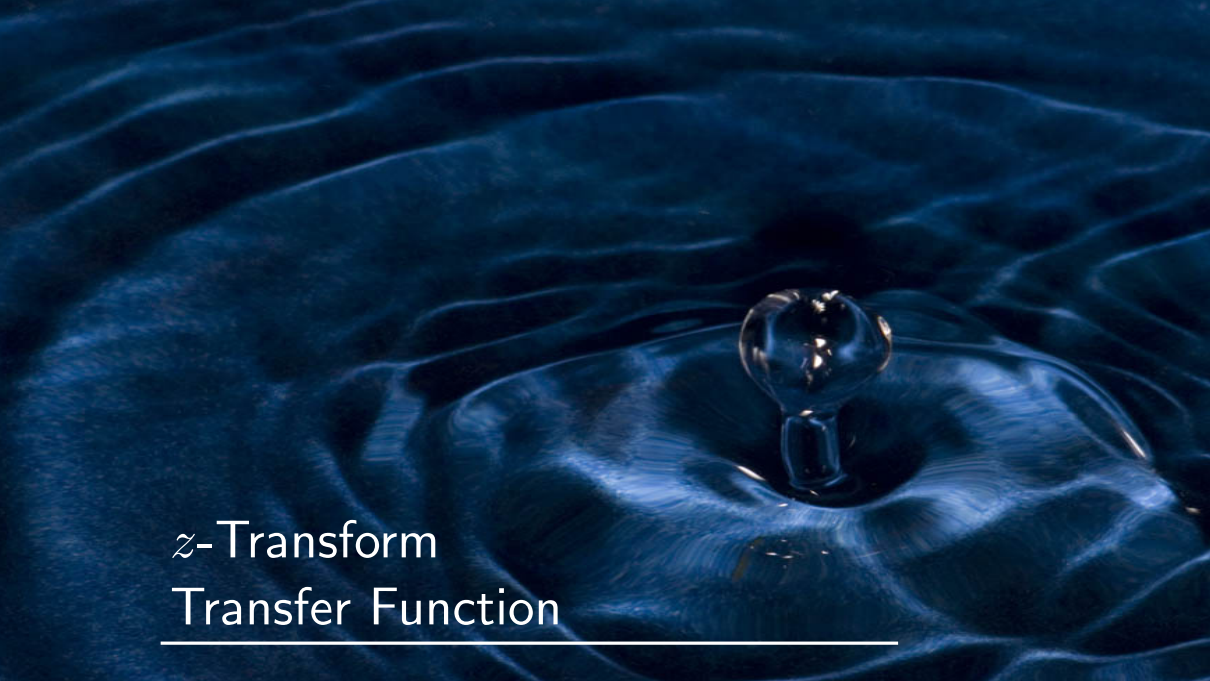
- The **region of convergence** (ROC) of a z-transform is the set of $z \in \mathbb{C}$ such that it converges
- The ROC is important
  - A z-transform converges for only certain values of z and does not exist for other values of z
  - z-transforms are non-unique without it
  - Always state the ROC when you state a z-transform
- The ROC is a connected annulus (doughnut) in the z-plane centered on the origin $z = 0$; that is, the ROC is of the form $r_1 < |z| < r_2$
  - If $x[n]$ has finite duration, then the ROC is the entire z-plane (except possibly $z = 0$ or $z = \infty$)
  - If $x[n]$ is causal, then the ROC is the outside of a disk
  - If $x[n]$ is anti-causal, then the ROC is the inside of a disk



z-Transform
Transfer Function
Poles and Zeros

Table of Contents

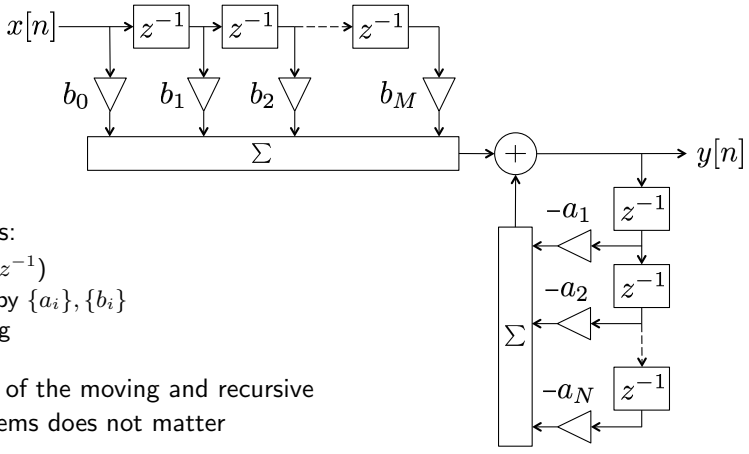
- Lecture in two parts:
  - Part 1: Transfer Function
  - Part 2: Poles and Zeros



z-Transform
Transfer Function

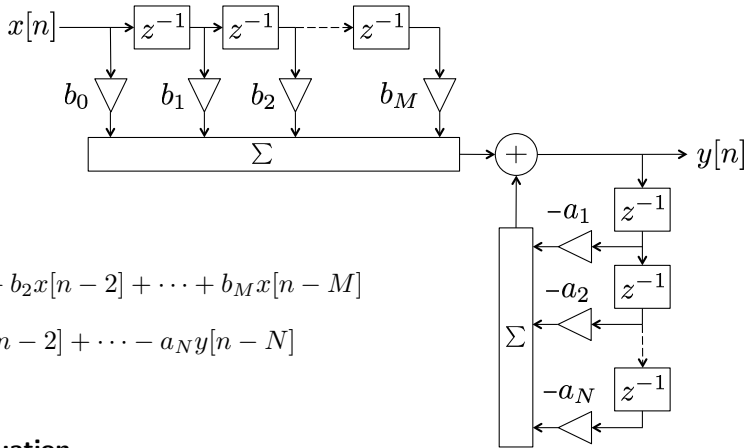
A General Class of LTI Systems

- The most general class of practical causal LTI systems (for infinite-length signals) consists of the cascade of a **moving average** system with a **recursive average** system



- Key elements:
  - Delays (z^{-1})
  - Scaling by $\{a_i\}, \{b_i\}$
  - Summing
- Note: Order of the moving and recursive average systems does not matter

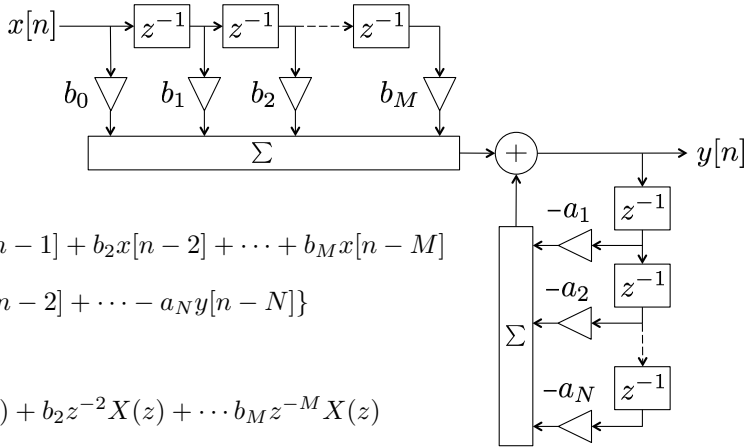
Input/Output Equation for General LTI System (Time Domain)



$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \cdots + b_Mx[n-M]$$
$$- a_1y[n-1] - a_2y[n-2] + \cdots - a_Ny[n-N]$$

This is called a **difference equation**
(Can process signals using the `filter` command in Matlab)

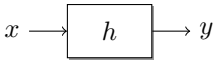
Input/Output Equation for General LTI System (z-Transform Domain)



$$\mathcal{Z}\{y[n]\} = \mathcal{Z}\{b_0x[n] + b_1x[n-1] + b_2x[n-2] + \cdots + b_Mx[n-M] - a_1y[n-1] - a_2y[n-2] + \cdots - a_Ny[n-N]\}$$

$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + b_2z^{-2}X(z) + \cdots b_Mz^{-M}X(z) - a_1z^{-1}Y(z) - a_2z^{-2}Y(z) - \cdots - a_Nz^{-N}Y(z)$$

Transfer Function of an LTI System



$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- If

$$x[n] \overset{\mathcal{Z}}{\longleftrightarrow} X(z), \qquad h[n] \overset{\mathcal{Z}}{\longleftrightarrow} H(z), \qquad y[n] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z)$$

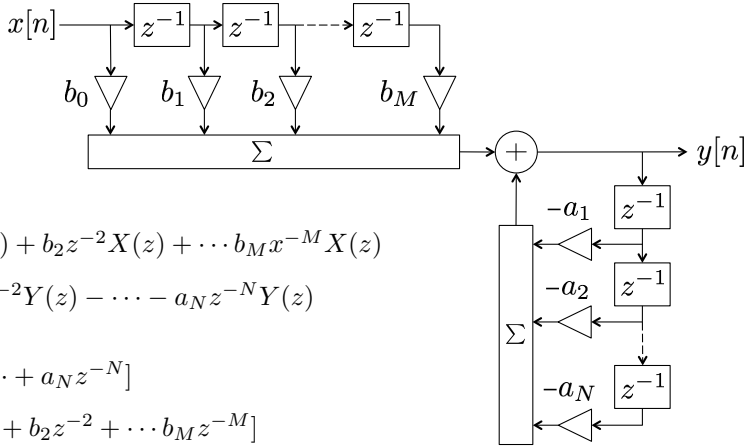
  then

$$Y(z) \;=\; H(z)\, X(z)$$

- **Transfer function**

$$H(z) \;=\; \frac{Y(z)}{X(z)} \;=\; \frac{\text{output}}{\text{input}}$$

Transfer Function of General LTI System



$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + b_2z^{-2}X(z) + \cdots b_Mx^{-M}X(z)$$
$$- a_1z^{-1}Y(z) - a_2z^{-2}Y(z) - \cdots - a_Nz^{-N}Y(z)$$

$$Y(z)\left[1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_Nz^{-N}\right]$$
$$= X(z)\left[b_0 + b_1z^{-1} + b_2z^{-2} + \cdots b_Mz^{-M}\right]$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \cdots b_Mz^{-M}}{1 + a_1z^{-1} + a_2z^{-2} + \cdots + a_Nz^{-N}}$$

Rational Transfer Functions

- The transfer function of the general LTI system is a **rational function** of polynomials in z^{-1}

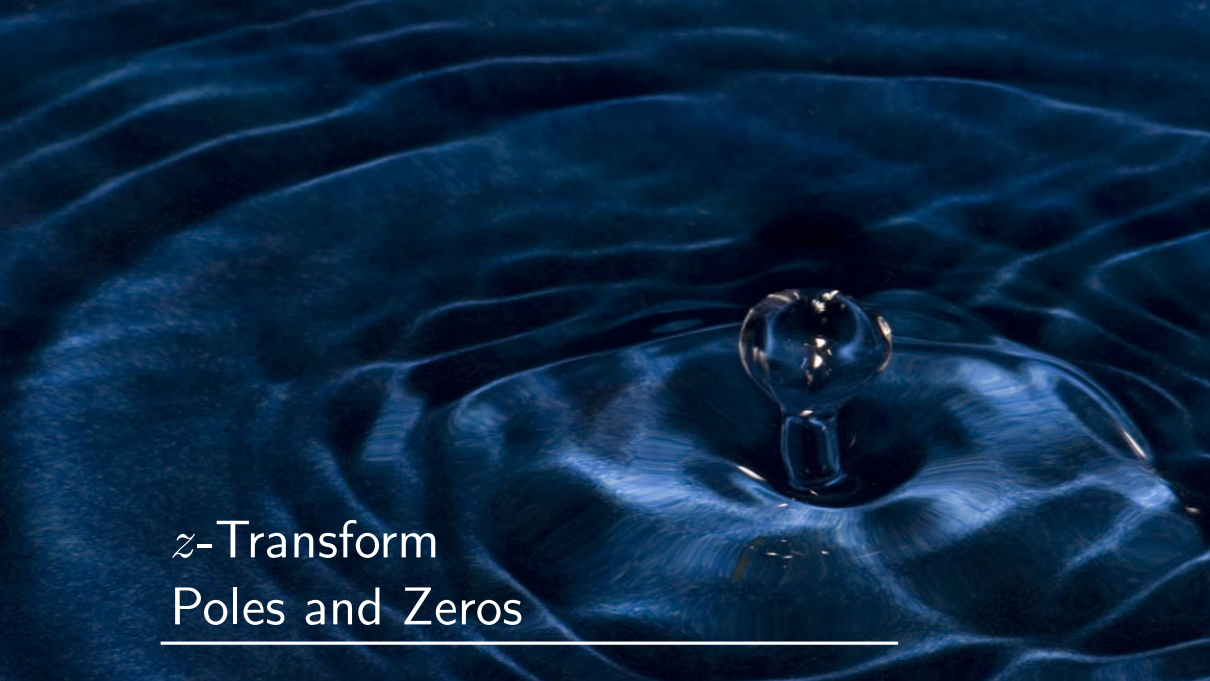
$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}$$

  - **Numerator:** moving average part (non-recursive)
  - **Denominator:** recursive average part

- The **order** of the system is the larger of M and N

- Can easily convert $H(z)$ to positive powers of z by multiplying by $\frac{z^N}{z^M}\frac{z^M}{z^N} = 1$

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{z^N}{z^M}\right) \frac{b_0 z^M + b_1 z^{M-1} + b_2 z^{M-2} + \cdots b_M}{z^N + a_1 z^{N-1} + a_2 z^{N-2} + \cdots + a_N}$$



z-Transform
Poles and Zeros

Poles and Zeros

- Factor the numerator and denominator polynomials (`roots` command in Matlab)

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{z^N}{z^M}\right) \frac{b_0 z^M + b_1 z^{M-1} + b_2 z^{M-2} + \cdots b_M}{z^N + a_1 z^{N-1} + a_2 z^{N-2} + \cdots + a_N}$$

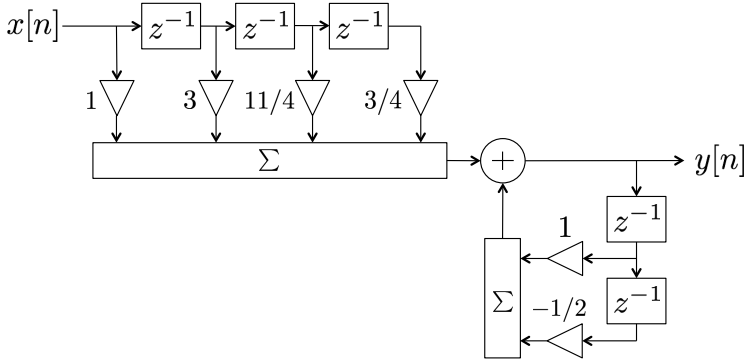
$$= \left(\frac{z^N}{z^M}\right) \frac{(z-\zeta_1)(z-\zeta_2)\cdots(z-\zeta_M)}{(z-p_1)(z-p_2)\cdots(z-p_N)}$$

- Roots of the numerator $\{\zeta_i\}$ are called **zeros**; at these values of z, $H(z) = 0$

- Roots of the denominator $\{p_i\}$ are called **poles**; at these values of z, $H(z) = \infty$

- Useful to plot the poles and zeros in the complex z-plane (`zplane` in Matlab)

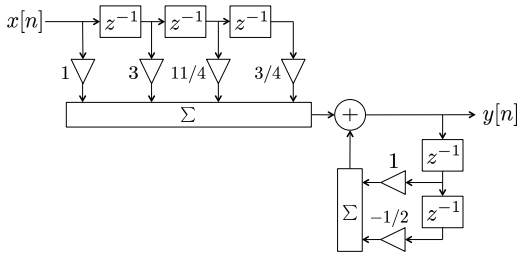
Example: Poles and Zeros (1)



- Given the block diagram, the transfer function is

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{z^2}{z^3}\right) \frac{1\,z^3 + 3\,z^2 + 11/4\,z + 3/4}{z^2 - 1\,z + 1/2} = \frac{1 + 3\,z^{-1} + 11/4\,z^{-2} + 3/4\,z^{-3}}{1 - 1\,z^{-1} + 1/2\,z^{-2}}$$

Example: Poles and Zeros (2)



- Factor the transfer function into poles and zeros

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{z^2}{z^3}\right) \frac{1\,z^3 + 3\,z^2 + 11/4\,z + 3/4}{z^2 - 1\,z + 1/2} = \frac{(z+1)(z+1/2)(z+3/2)}{z(z-1/2-j/2)(z-1/2+j/2)}$$

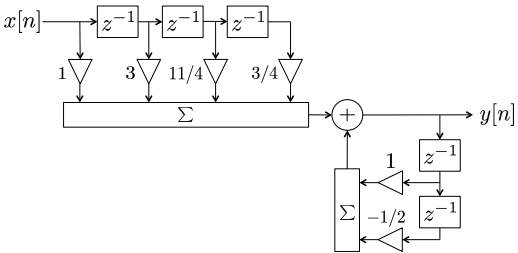
- Zeros at $z = -1/2,\ -1,\ -3/2$

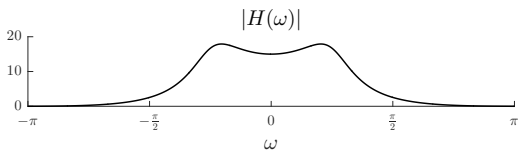
- Poles at $z = 0,\ 1/2 + j/2,\ 1/2 - j/2$
  Note: System is BIBO stable, since it is causal and all poles are inside the unit circle

Example: Poles and Zeros (3)

- We can obtain the **frequency response** by evaluating the transfer function $H(z)$ on the unit circle (`freqz` in Matlab)

$$H(e^{j\omega}) \;=\; \frac{1\,e^{j3\omega} + 3\,e^{j2\omega} + 11/4\,e^{j\omega} + 3/4}{e^{j\omega}(1\,e^{j2\omega} - 1\,e^{j\omega} + 1/2)}$$





Sketching $|H(z)|$ based on the Poles and Zeros

- Given the poles and zeros, there is an elegant geometrical interpretation of the value of $|H(z)|$

$$\begin{aligned} |H(z)| &= \left| \left(\frac{z^N}{z^M} \right) \frac{b_0 z^M + b_1 z^{M-1} + b_2 z^{M-2} + \cdots b_M}{z^N + a_1 z^{N-1} + a_2 z^{N-2} + \cdots + a_N} \right| \\ &= \left| \frac{z^N}{z^M} \right| \frac{|z - \zeta_1|\,|z - \zeta_2| \cdots |z - \zeta_M|}{|z - p_1|\,|z - p_2| \cdots |z - p_N|} \end{aligned}$$

- Note, for example, that

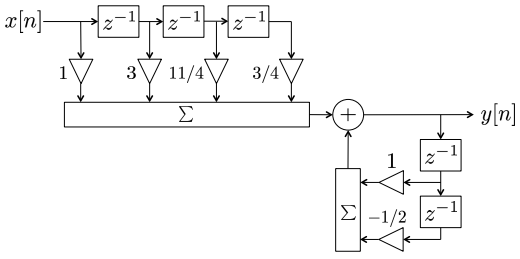
$$\begin{aligned} |z - p_1| &= \sqrt{[\mathrm{Re}(z) - \mathrm{Re}(p_1)]^2 + [\mathrm{Im}(z) - \mathrm{Im}(p_1)]^2} \\ &= \text{Euclidean distance between the points } z \text{ and } p_1 \text{ in the 2D complex } z \text{ plane} \end{aligned}$$

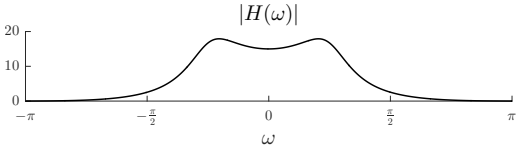
- Therefore

$$|H(z)| = \frac{\text{product of distances from } z \text{ to all zeros}}{\text{product of distances from } z \text{ to all poles}}$$

Example: Poles and Zeros (4)

$$H(z) = \frac{(z+1)(z+1/2)(z+3/2)}{z(z-1/2-j/2)(z-1/2+j/2)}$$

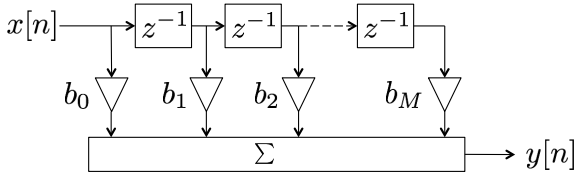




LTI System Design = Pole and Zero Placement

- Locations of the poles and zeros characterize an LTI system
- Hence, design of an LTI system is equivalent to designing where to place its poles and zeros

FIR Filters Have Only Zeros



- An FIR filter has **only zeros**

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_M z^{-M}$$
$$= (z - \zeta_1)(z - \zeta_2) \cdots (z - \zeta_M)$$

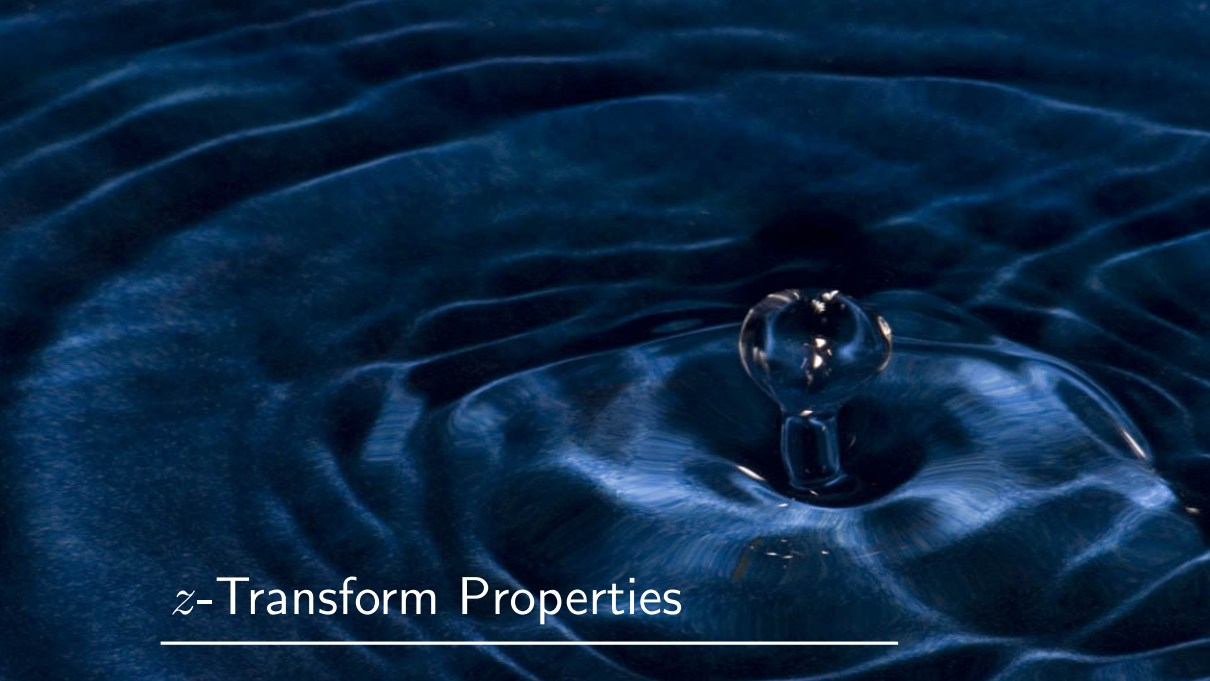
- Filter order = number of zeros = number of delays
- It is not uncommon to have hundreds or even thousands of zeros

Summary

- The most general class of practical causal LTI systems (for infinite-length signals) consists of the cascade of a **moving average** system with a **recursive average** system
- Transfer function $H(z) = \frac{Y(z)}{X(z)}$
- Zeros/poles $=$ roots of numerator/denominator of $H(z)$
- Elegant geometric formula

$$|H(z)| \;=\; \frac{\text{product of distances from } z \text{ to all zeros}}{\text{product of distances from } z \text{ to all poles}}$$

- Designing LTI systems is equivalent to placing their poles and zeros



z-Transform Properties

Recall: Forward z-Transform

- Forward z-transform of the signal $x[n]$ (analysis)

$$X(z) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}, \qquad z \in \mathbb{C},\; z \in \text{ROC}$$

- The z-transform is a complex function of a complex variable

- **z-transform pair**

$$x[n] \;\overset{\mathcal{Z}}{\longleftrightarrow}\; X(z)$$

z-Transform and DTFT

- Forward z-transform of the signal $x[n]$

$$X(z) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}, \qquad z \in \mathbb{C},\; z \in \text{ROC}$$

- DTFT of the signal $x[n]$

$$X(\omega) \;=\; \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad -\pi \leq \omega < \pi$$

- The DTFT is a special case of the z-transform (values on the unit circle in the complex z-plane)

$$X_{\text{DTFT}}(\omega) \;=\; X_z(z)|_{z=e^{j\omega}} \;=\; X_z(e^{j\omega})$$

z-Transform is Linear

- It is easy to show that if

$$x_1[n] \overset{\mathcal{Z}}{\longleftrightarrow} X_1(z) \qquad x_2[n] \overset{\mathcal{Z}}{\longleftrightarrow} X_2(z)$$

then

$$y[n] = \alpha_1 x_1[n] + \alpha_2 x_2[n] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z) = \alpha_1 X_1(z) + \alpha_2 X_2(z)$$

with

$$\mathsf{ROC}_Y = \mathsf{ROC}_{X_1} \cap \mathsf{ROC}_{X_2}$$

z-Transform and Time Shift

- If $x[n]$ and $X(z)$ are a z-transform pair then

$$x[n-m] \overset{\mathcal{Z}}{\longleftrightarrow} z^{-m} X(z)$$

with no change to the ROC

- Proof: Use the change of variables $r = n - m$

$$\sum_{n=-\infty}^{\infty} x[n-m]\, z^{-n} = \sum_{r=-\infty}^{\infty} x[r]\, z^{-(r+m)} = \sum_{r=-\infty}^{\infty} x[r]\, z^{-r}\, z^{-m}$$

$$= z^{-m} \sum_{r=-\infty}^{\infty} x[r]\, z^{-r} = z^{-m}\, X(z) \quad \checkmark$$

z-Transform and Modulation

- If $x[n]$ and $X(z)$ are a z-transform pair then

$$z_0^n\, x[n] \quad \overset{\mathcal{Z}}{\longleftrightarrow} \quad X(z/z_0)$$

- The ROC of $X(z/z_0)$ is $|z_0|\times$ROC of $X(z)$

- Proof:

$$\sum_{n=-\infty}^{\infty} z_0^n\, x[n]\, z^{-n} \;=\; \sum_{n=-\infty}^{\infty} x[n]\, (z/z_0)^{-n} \;=\; X(z/z_0) \;\checkmark$$

z-Transform and Conjugation

- If $x[n]$ and $X(z)$ are a DFT pair then

$$x^*[n] \overset{\mathcal{Z}}{\longleftrightarrow} X^*(z^*)$$

with no change to the ROC

- Proof:

$$\sum_{n=-\infty}^{\infty} x^*[n]\, z^{-n} = \left(\sum_{n=-\infty}^{\infty} x[n]\, (z^*)^{-n} \right)^* = X^*(z^*) \quad \checkmark$$

z-Transform and Time Reversal

- If $x[n]$ and $X(z)$ are a DFT pair then

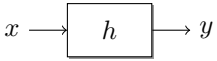
$$x[-n] \overset{\mathcal{Z}}{\longleftrightarrow} X(z^{-1})$$

- The ROC is "inverted" in the sense that if the ROC of $X(z)$ is $r_1 < |z| < r_2$, then the ROC of $X(z^{-1})$ is $1/r_2 < |z| < 1/r_1$

- Proof: Use change of variables $m = -n$

$$\sum_{n=-\infty}^{\infty} x[-n]\, z^{-n} \;=\; \sum_{m=-\infty}^{\infty} x[m]\, z^{m} \;=\; \sum_{m=-\infty}^{\infty} x[m]\, (z^{-1})^{-m} \;=\; X(z^{-1}) \;\checkmark$$

z-Transform and Convolution



$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- If

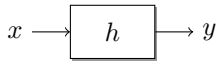
$$x[n] \overset{\mathcal{Z}}{\longleftrightarrow} X(z), \qquad h[n] \overset{\mathcal{Z}}{\longleftrightarrow} H(z), \qquad y[n] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z)$$

  then

$$Y(z) \;=\; H(z)\, X(z), \qquad \mathsf{ROC}_Y = \mathsf{ROC}_X \cap \mathsf{ROC}_H$$

- Convolution in the time domain = multiplication in the z-transform domain

z-Transform, BIBO Stability, and Causality

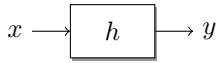


$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

- Recall that an LTI system is BIBO stable iff $\|h\|_1 < \infty$

- **Fact:** An LTI system is BIBO stable iff the ROC of $H(z)$ includes the unit circle $|z| = 1$

Proof: z-Transform and BIBO Stability (1)

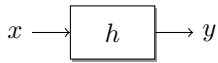


- Recall that the ROC of $H(z)$ is defined as the set of $z \in \mathbb{C}$ such that $h[n]\, z^{-n}$ is absolutely summable

$$S(z) \;=\; \sum_{n=-\infty}^{\infty} |h[n]|\, |z^{-n}| \;<\; \infty$$

- Suppose that the system is BIBO stable, which implies that $\|h\|_1 < \infty$. What can we say about the ROC?
  - When $|z| = 1$, we see that $S(z)|_{|z|=1} = \sum_{n=-\infty}^{\infty} |h[n]| < \infty$ since $\|h\|_1 < \infty$
  - Thus, the unit circle $|z| = 1$ is in the ROC

Proof: z-Transform and BIBO Stability (1)

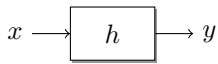


- Recall that the ROC of $H(z)$ is defined as the set of $z \in \mathbb{C}$ such that $h[n]\, z^{-n}$ is absolutely summable

$$S(z) \;=\; \sum_{n=-\infty}^{\infty} |h[n]|\, |z^{-n}| \;<\; \infty$$

- Suppose that the ROC of $H(z)$ includes the unit circle. What can we say about $h[n]$ and BIBO stability?
  - Since the ROC of $H(z)$ includes the unit circle, $S(1) < \infty$. But $S(1) = \sum_{n=-\infty}^{\infty} |h[n]| = \|h\|_1$
  - Thus, $\|h\|_1 < \infty$ and the system is BIBO stable

z-Transform, Poles, and BIBO Stability



- **Fact:** An LTI system is BIBO stable iff the ROC of $H(z)$ includes the unit circle $|z| = 1$

- **Corollary:** A **causal** LTI system is BIBO stable iff all of its poles are inside the unit circle

  - Since the systems is causal, the ROC extends outward from the pole that is furthest from the origin
  - Since the ROC contains the unit circle $|z| = 1$, the pole p furthest from the origin must have $|p| < 1$

Useful z-Transforms

$x[n]$	$X(z)$	ROC
$\delta[n]$	1	$z \in \mathbb{C}$
$u[n]$	$\frac{1}{1-z^{-1}}$	$\lvert z\rvert > 1$
$\alpha^n u[n]$	$\frac{1}{1-\alpha z^{-1}}$	$\lvert z\rvert > \lvert\alpha\rvert$
$-\alpha^n u[-n-1]$	$\frac{1}{1-\alpha z^{-1}}$	$\lvert z\rvert < \lvert\alpha\rvert$

For many more, see the supplementary materials, for example
`http://en.wikipedia.org/wiki/Z-transform`

Summary

- The z-transform has properties similar to the DTFT
- Convolution in time becomes multiplication in the z-transform domain
- An LTI system is BIBO stable iff the ROC of $H(z)$, the z-transform of the impulse response $h[n]$, includes the unit circle $|z| = 1$
- A **causal** LTI system is BIBO stable iff all of its poles are inside the unit circle

Inverse z-Transform

Recall: Forward z-Transform

- **Forward z-transform** of the signal $x[n]$ (analysis)

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]\, z^{-n}, \qquad z \in \mathbb{C},\ z \in \text{ROC}$$

- Given $X(z)$, how to compute the **inverse z-transform** (synthesis)?

- **Complication:** $X(z)$ is a complex function of a complex variable z

Inverse z-Transform via Complex Integral

- Recall the **inverse DTFT**

$$x[n] \;=\; \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}$$

- There exists a similar formula for the **inverse z-transform** via a contour integral in the complex z-plane

$$x[n] \;=\; \oint_{\mathcal{C}} X(z)\, z^{n}\, \frac{dz}{j2\pi z}$$

- Evaluation of such integrals is fun, but beyond the scope of this course

Inverse z-Transform via Factorization

- Any rational z-transform $X(z)$ can be factored into its zeros and poles

$$X(z) = \frac{(z-\zeta_1)(z-\zeta_2)\cdots(z-\zeta_M)}{(z-p_1)(z-p_2)\cdots(z-p_N)} = z^{N-M}\frac{(1-\zeta_1 z^{-1})(1-\zeta_2 z^{-1})\cdots(1-\zeta_M z^{-1})}{(1-p_1 z^{-1})(1-p_2 z^{-1})\cdots(1-p_N z^{-1})}$$

- Inverting the z-transform for one pole $\frac{1}{1-p_1 z^{-1}}$ is easy (assume causal inverse)

$$p_1^n\, u[n] \overset{\mathcal{Z}}{\longleftrightarrow} \frac{1}{1-p_1 z^{-1}}$$

- Fortunately, there is a method to decompose $X(z)$ into a sum, rather than a product, of poles

$$X(z) = \frac{C_1}{1-p_1 z^{-1}} + \frac{C_2}{1-p_2 z^{-1}} + \cdots + \frac{C_N}{1-p_N z^{-1}}$$

- Then we can just invert the z-transform of each term separately and sum the results

Partial Fraction Expansion

- The **partial fraction expansion** decomposes a rational function of z^{-1}

$$X(z) \;=\; \frac{\text{polynomial in } z^{-1}}{(1-p_1 z^{-1})(1-p_2 z^{-1})\cdots(1-p_N z^{-1})}$$

  into a sum of terms, one for each pole

$$X(z) \;=\; \frac{C_1}{1-p_1 z^{-1}} \;+\; \frac{C_2}{1-p_2 z^{-1}} \;+\; \cdots \;+\; \frac{C_N}{1-p_N z^{-1}}$$

- There are many algorithms to compute a partial fraction expansion; here we introduce one of the simplest using a running example

- Note: We will explain partial fractions only for non-repeated poles; see the supplementary material for the case when there are repeated poles (not difficult!)

Partial Fraction Expansion Algorithm – Step 1

- Goal: Invert the z-transform of the rational function

$$X(z) = \frac{5 + 2z^{-1} - z^{-2}}{1 - \frac{1}{6}z^{-1} - \frac{1}{6}z^{-2}}$$

using the partial fraction expansion

- **Step 1:** Check if (order of denominator polynomial) $>$ (order of numerator polynomial); otherwise perform polynomial division to make it so
  - Not true in our case, so use polynomial long division to divide the denominator into the numerator

$$X(z) = 6 + \frac{-1 + 3z^{-1}}{1 - \frac{1}{6}z^{-1} - \frac{1}{6}z^{-2}} = 6 + X'(z)$$

We will work on $X'(z)$ and then recombine it with the 6 at the end

Partial Fraction Expansion Algorithm – Step 2

- **Step 2:** Factor the denominator polynomial into poles (no need to factor the numerator)

$$X'(z) = \frac{-1+3z^{-1}}{1-\frac{1}{6}z^{-1}-\frac{1}{6}z^{-2}} = \frac{-1+3z^{-1}}{(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})}$$

Partial Fraction Expansion Algorithm – Step 3

- **Step 3:** Assuming that no poles are repeated (same location in the z-plane), break $X'(z)$ into a sum of terms, one for each pole

$$X'(z) = \frac{-1+3z^{-1}}{(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})} = \frac{C_1}{1-\frac{1}{2}z^{-1}} + \frac{C_2}{1+\frac{1}{3}z^{-1}}$$

- Note: Repeated poles just require that we add extra terms to the sum. The details are not difficult – see the Supplementary Resources

Partial Fraction Expansion Algorithm – Step 4

- **Step 4:** To determine C_1 and C_2, bring the sum of terms back to a common denominator and compare its terms to terms of the same power in $X'(z)$

$$X'(z) = \frac{-1+3z^{-1}}{(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})} = \frac{C_1}{1-\frac{1}{2}z^{-1}} + \frac{C_2}{1+\frac{1}{3}z^{-1}}$$

$$= \frac{C_1(1+\frac{1}{3}z^{-1}) + C_2(1-\frac{1}{2}z^{-1})}{(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})} = \frac{(C_1+C_2) + (\frac{C_1}{3} - \frac{C_2}{2})z^{-1}}{(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})}$$

- This yields the set of 2 simultaneous linear equations

$$\text{powers of } 1: \quad -1 = C_1 + C_2$$

$$\text{powers of } z^{-1}: \quad 3 = \frac{C_1}{3} - \frac{C_2}{2}$$

- Solve by your favorite method to obtain: $C_1 = 3, \ C_2 = -4$

Partial Fraction Expansion Algorithm – Step 5

- **Final partial fractions result**

$$X(z) \;=\; 6 \;+\; \frac{3}{1-\frac{1}{2}z^{-1}} \;+\; \frac{-4}{1+\frac{1}{3}z^{-1}}$$

- **Step 5:** Check your work by bringing the partial fractions result to a common demoninator and making sure it agrees with what you started with

$$\begin{aligned} X(z) &= 6 + \frac{3}{1-\frac{1}{2}z^{-1}} + \frac{-4}{1+\frac{1}{3}z^{-1}} \\ &= \frac{6(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1}) + 3(1+\frac{1}{3}z^{-1}) - 4(1-\frac{1}{2}z^{-1})}{(1-\frac{1}{2}z^{-1})(1+\frac{1}{3}z^{-1})} \\ &= \frac{5+2z^{-1}-z^{-2}}{1-\frac{1}{6}z^{-1}-\frac{1}{6}z^{-2}} \quad \checkmark \end{aligned}$$

Back To Our Regularly Scheduled Program . . .

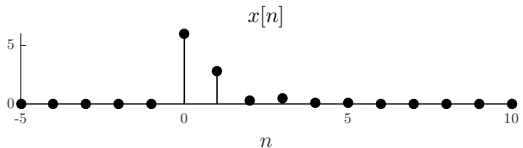
- Thanks to the partial fractions expansion, we can write

$$X(z) \;=\; \frac{5+2z^{-1}-z^{-2}}{1-\frac{1}{6}z^{-1}-\frac{1}{6}z^{-2}} \;=\; 6 \;+\; \frac{3}{(1-\frac{1}{2}z^{-1})} \;+\; \frac{-4}{(1+\frac{1}{3}z^{-1})}$$

  and so it is easy to invert the z-transform $X(z)$ term-by-term (thanks to linearity)

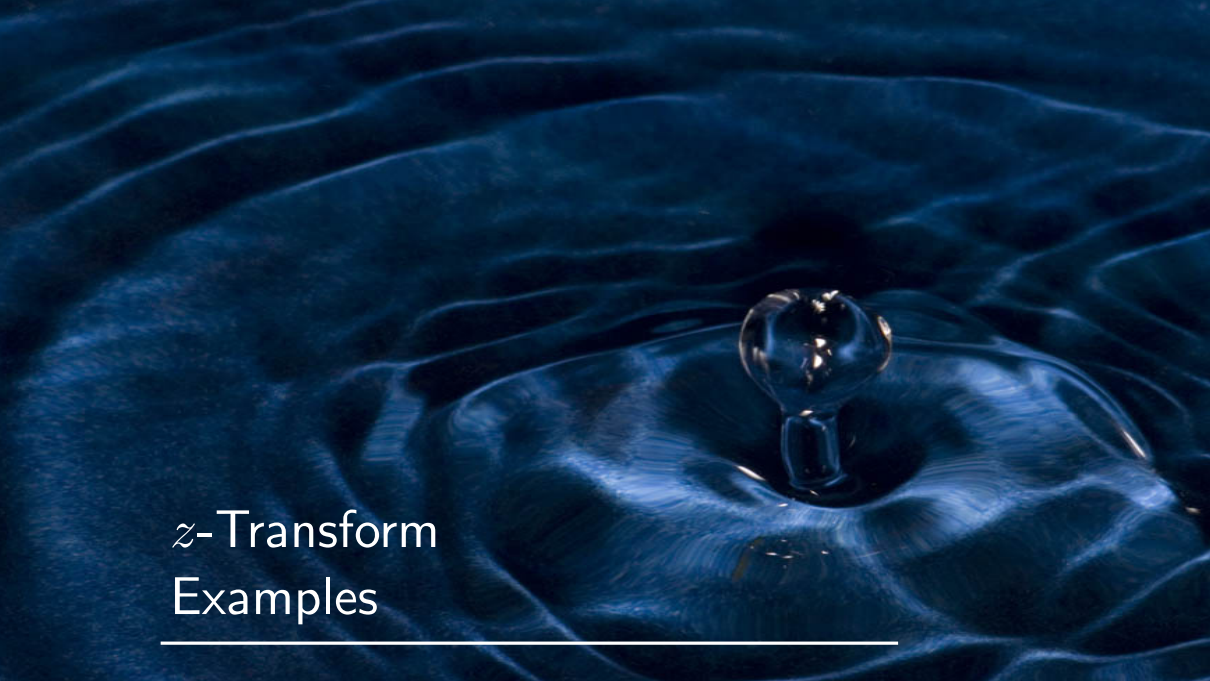
- Assuming that $x[n]$ is causal (ROC: $|z| > \frac{1}{2}$) yields

$$x[n] \;=\; 6\,\delta[n] \;+\; 3\left(\frac{1}{2}\right)^n u[n] \;-\; 4\left(\frac{-1}{3}\right)^n u[n]$$



Summary

- Inverse z-transform is more complicated to compute than inverse Fourier transforms
- In practice, we invert the z-transform using the partial fraction expansion and the inverting term-by-term
- Partial fraction expansion is a tedious process, but straightforward (and useful!)

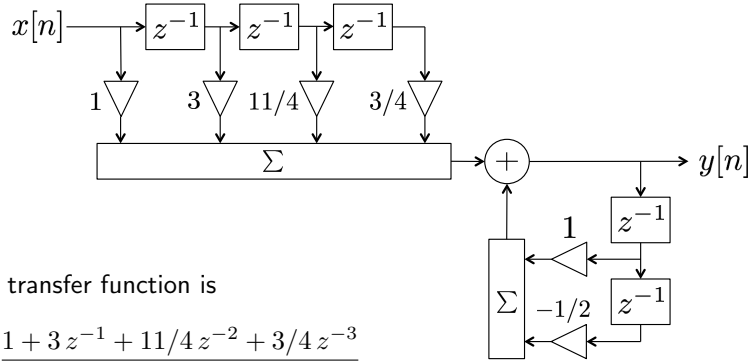


z-Transform Examples

z-Transform Tools in Matlab

- There are many useful commands in Matlab related to the z-transform. **Click here** to view a video demonstration.
  - `roots` – to factor polynomials and find zeros and poles
  - `zplane` – to plot poles and zeros in the complex z-plane
  - `freqz` – to plot the DTFT frequency response corresponding to a z transform transfer function
  - `filter` – to process an input signal and produce an output signal
  - ...

Example LTI System



- Given the block diagram, the transfer function is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + 3\,z^{-1} + 11/4\,z^{-2} + 3/4\,z^{-3}}{1 - 1\,z^{-1} + 1/2\,z^{-2}}$$

- Matlab encodes the coefficients according to two vectors

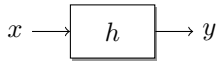
$$a = \begin{bmatrix} 1 & -1 & 1/2 \end{bmatrix}, \qquad b = \begin{bmatrix} 1 & 3 & 11/4 & 3/4 \end{bmatrix}$$

Discrete-Time Filters

Set H

Discrete-Time Filters

Putting LTI Systems to Work



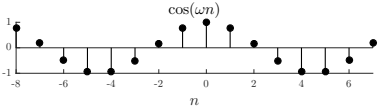
$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

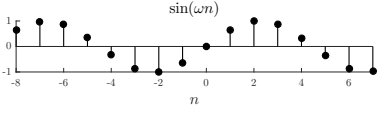
$$Y(z) \;=\; H(z)\, X(z), \qquad Y(\omega) \;=\; H(\omega)\, X(\omega)$$

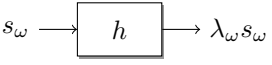
- **Goal:** Design a LTI system to perform a certain **task** in some application
- Key questions:
  - What is the **range of tasks** that an LTI system can perform?
  - What are the **parameters** under our control for design purposes?

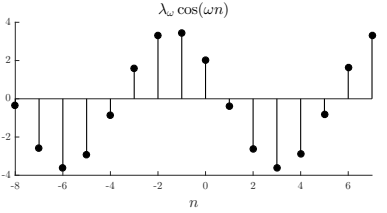
What Do LTI Systems Do? Recall Eigenanalysis

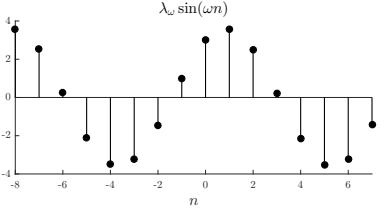
- LTI system **eigenvectors**: $s_\omega[n] = e^{j\omega n}$
- LTI system **eigenvalues**: $\lambda_\omega = H(\omega) = \sum_{n=-\infty}^{\infty} h[n]\, e^{-j\omega n}$ (frequency response)



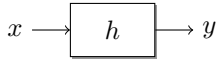








LTI Systems Filter Signals



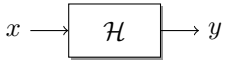
- Important interpretation of $Y(\omega) = H(\omega)\, X(\omega)$

$$x[n] = \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi} \longrightarrow \boxed{h} \longrightarrow y[n] = \int_{-\pi}^{\pi} H(\omega)\, X(\omega)\, e^{j\omega n}\, \frac{d\omega}{2\pi}$$

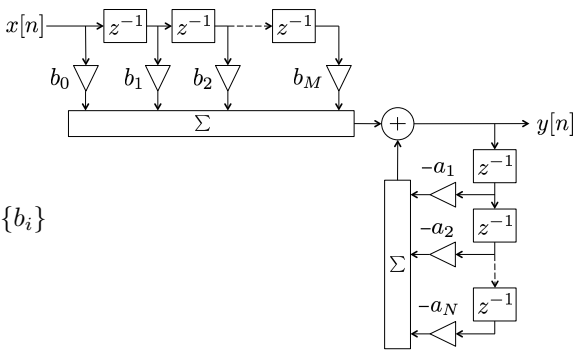
- An LTI system processes a signal $x[n]$ by amplifying or attenuating the sinusoids in its Fourier representation (DTFT) $X(\omega)$ by the complex factor $H(\omega)$

- Inspires the terminology that $X(\omega)$ is **filtered** by $H(\omega)$ to produce $Y(\omega)$

Design Parameters of Discrete-Time Filters (LTI Systems)

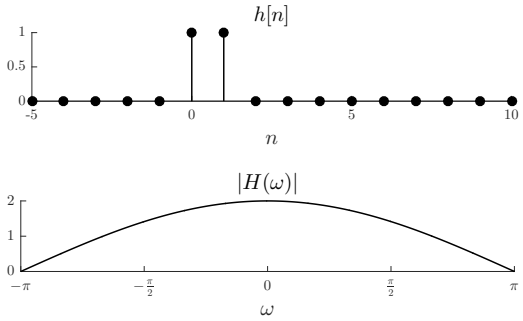


- Impulse response: $h[n]$
- Transfer function: $H(z)$
  - poles and zeros
- Frequency response: $H(\omega)$
- Moving/recursive average parameters: $\{a_i\}$, $\{b_i\}$



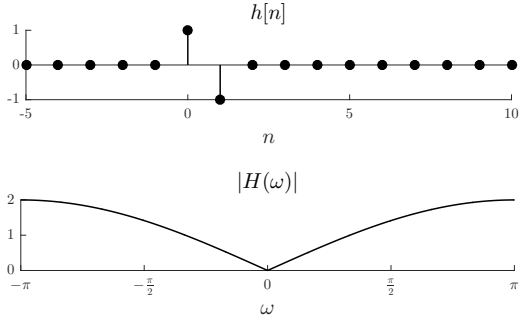
Filters Archetypes: Low-Pass

- Ideal low-pass filter
- Example low-pass impulse response $h[n]$
- Example frequency response $|H(\omega)|$



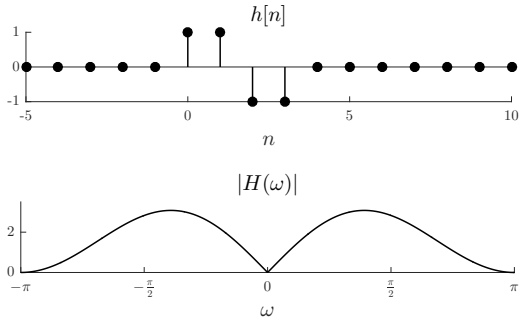
Filters Archetypes: High-Pass

- Ideal high-pass filter
- Example high-pass impulse response $h[n]$
- Example frequency response $|H(\omega)|$



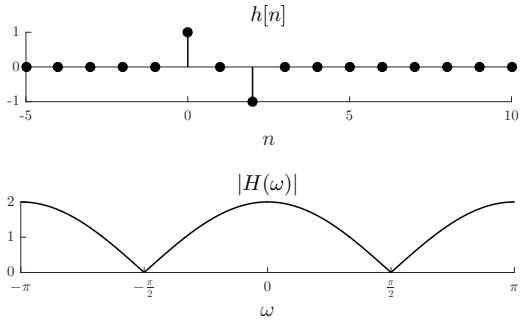
Filters Archetypes: Band-Pass

- Ideal band-pass filter
- Example band-pass impulse response $h[n]$
- Example frequency response $|H(\omega)|$



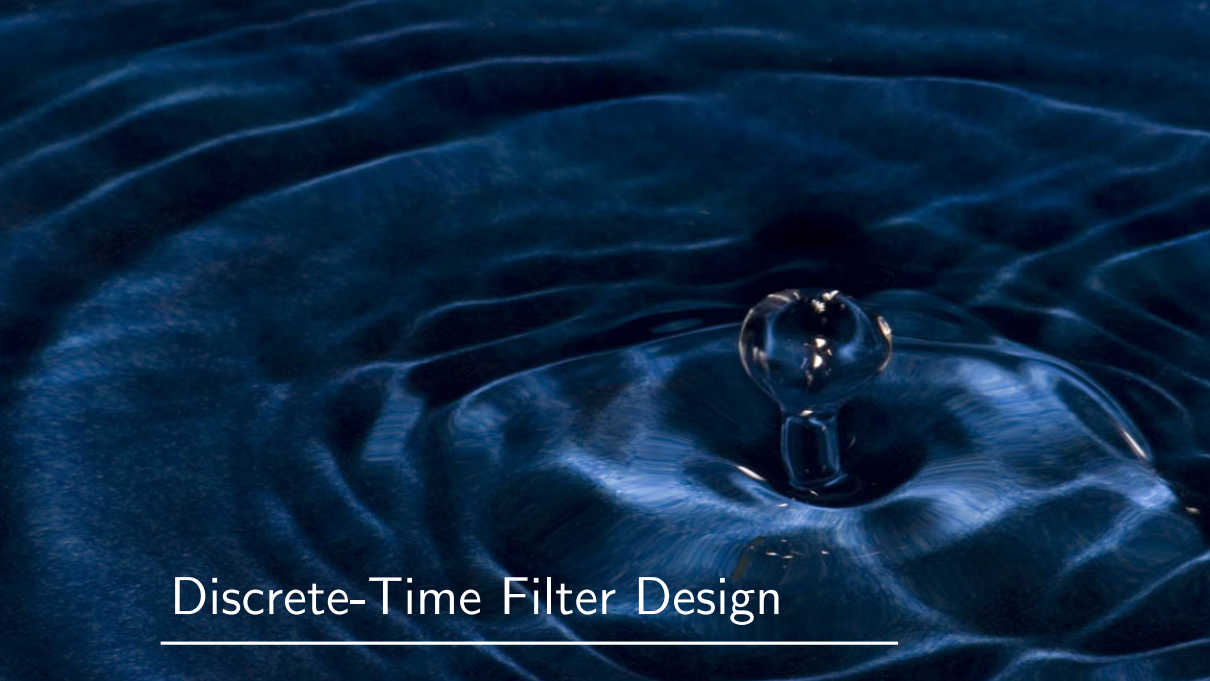
Filters Archetypes: Band-Stop

- Ideal band-stop filter
- Example band-stop impulse response $h[n]$
- Example frequency response $|H(\omega)|$



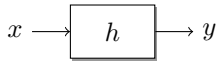
Summary

- Now that we understand what LTI systems do, we can **design** them to accomplish certain tasks
- An LTI system processes a signal $x[n]$ by amplifying or attenuating the sinusoids in its Fourier representation (DTFT)
- Equivalent design parameters of a discrete-time filter
  - Impulse response: $h[n]$
  - Transfer function: $H(z)$ (poles and zeros)
  - Frequency response: $H(\omega)$
  - Moving/recursive average parameters: $\{a_i\}$, $\{b_i\}$
- Archetype filters: Low-pass, high-pass, band-pass, band-stop
- We will emphasize infinite-length signals, but the situation is similar for finite-length signals



Discrete-Time Filter Design

Recall Discrete-Time Filter



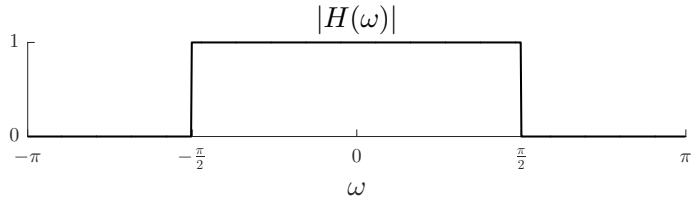
$$y[n] \;=\; x[n] * h[n] \;=\; \sum_{m=-\infty}^{\infty} h[n-m]\, x[m]$$

$$Y(z) \;=\; H(z)\, X(z), \qquad Y(\omega) \;=\; H(\omega)\, X(\omega)$$

- A discrete-time filter fiddles with a signal's Fourier representation
- Recall the filter archetypes: ideal low-pass, high-pass, band-pass, band-stop filters

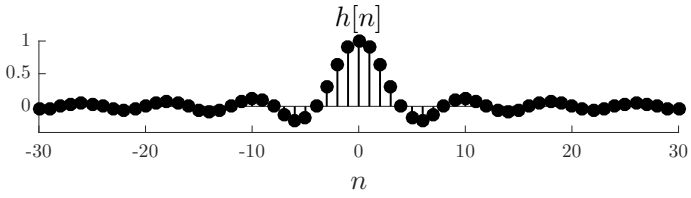
Ideal Lowpass Filter

- Ideal low-pass filter frequency response $H(\omega) = \begin{cases} 1 & -\omega_c \leq \omega \leq \omega_c \\ 0 & \text{otherwise} \end{cases}$



Ideal Lowpass Filter

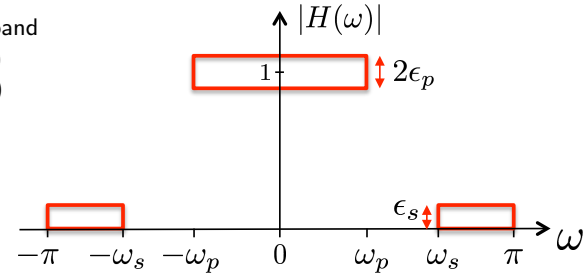
- Ideal low-pass filter frequency response $H(\omega) = \begin{cases} 1 & -\omega_c \leq \omega \leq \omega_c \\ 0 & \text{otherwise} \end{cases}$
- Impulse response is the infamous "sinc" function $h[n] = 2\omega_c \frac{\sin(\omega_c n)}{\omega_c n}$



- Problems:
  - System is not BIBO stable! $(\sum_n |h[n]| = \infty)$
  - Infinite computational complexity ($H(z)$ is not a rational function)

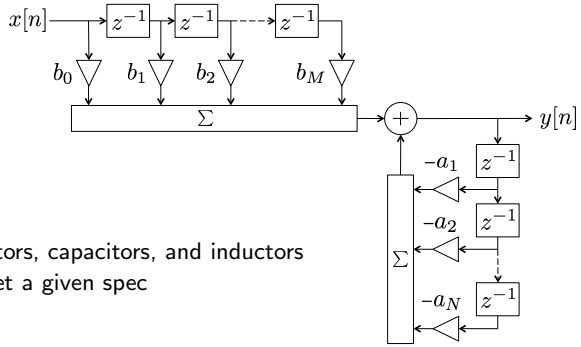
Filter Specification

- Find a filter of minimum complexity that meets a given **specification**
- Example: Low-pass filter
  - Pass-band edge frequency: ω_p
  - Stop-band edge frequency: ω_s
  - Between pass- and stop-bands: transition band
  - Pass-band ripple ϵ_p (often expressed in dB)
  - Stop-band ripple ϵ_s (often expressed in dB)



- Clearly, the tighter the specs, the more complex the filter

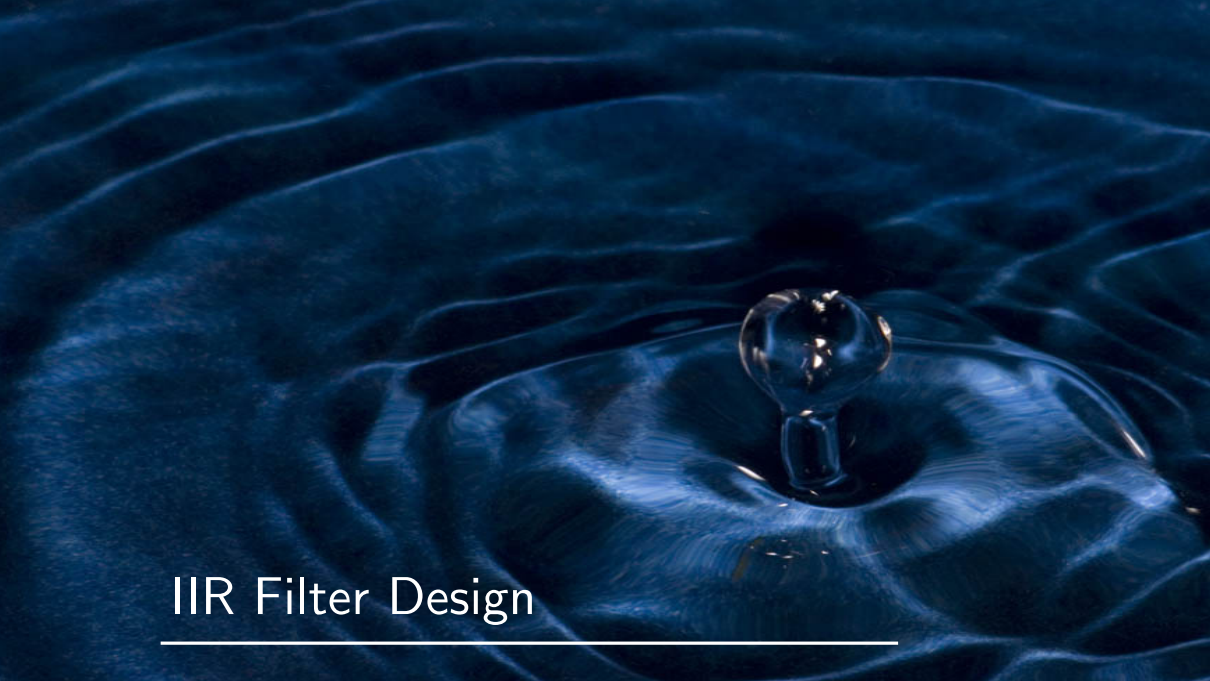
Two Classes of Discrete-Time Filters



- Infinite impulse response (IIR) filters
  - Uses both moving and recursive averages
  - $H(z)$ has both poles and zeros
  - Related to "analog" filter design using resistors, capacitors, and inductors
  - Generally have the lowest complexity to meet a given spec
- Finite impulse response (FIR) filters
  - Uses only moving average
  - $H(z)$ has only zeros
  - Unachievable in analog using resistors, capacitors, and inductors
  - Generally higher complexity (than IIR) to meet a given spec
  - But can have linear phase (big plus)

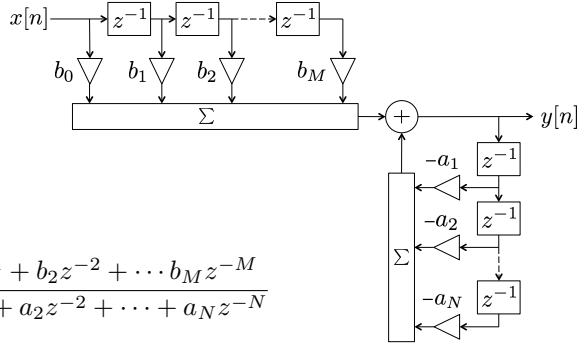
Summary

- A discrete-time filter fiddles with a signal's Fourier representation
- "Ideal" filters are not practical
  - System is not BIBO stable!
  - Infinite computational complexity ($H(z)$ is not a rational function)
- Filter design: Find a filter of minimum complexity that meets a given spec
- Two different types of filters (IIR, FIR) mean two different types of filter design



IIR Filter Design

IIR Filters



- Use both moving and recursive averages
- Transfer function has both **poles** and **zeros**

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}}$$

$$= z^{N-M} \frac{(z-\zeta_1)(z-\zeta_2)\cdots(z-\zeta_M)}{(z-p_1)(z-p_2)\cdots(z-p_N)}$$

- We **design** an IIR filter by specifying the locations of its poles and zeros in the z-plane
- Generally can satisfy a spec with lower complexity than FIR filters

IIR Filters from Analog Filters

- In contrast to FIR filter design, IIR filters are typically designed by a two-step procedure that is slightly ad hoc

- **Step 1:** Design an **analog** filter (for resistors, capacitors, and inductors) using the Laplace transform $H_L(s)$ (this theory is well established but well beyond the scope of this course)

- **Step 2:** Transform the analog filter into a discrete-time filter using the **bilinear transform** (a conformal map from complex analysis)

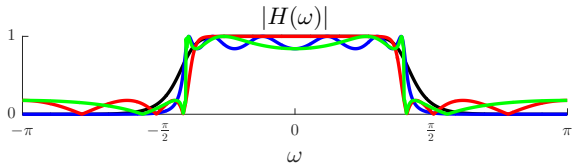
$$s \;=\; c\,\frac{z-1}{z+1}$$

- The discrete-time filter's transfer function is given by

$$H(z) \;=\; H_L(s)\big|_{s=c\frac{z-1}{z+1}}$$

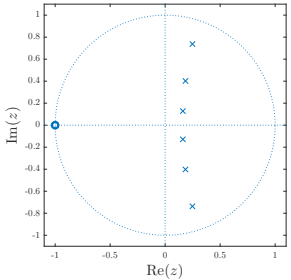
Three Important Classes of IIR Filters

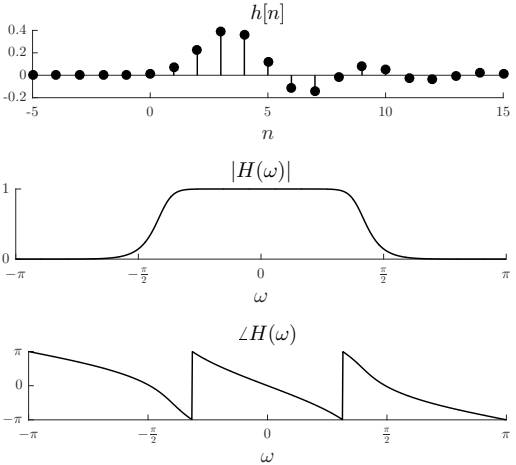
- Butterworth filters
  - `butter` command in Matlab
  - No ripples (oscillations) in $|H(\omega)|$
  - Gentlest transition from pass-band to stop-band for a given order
- Chebyshev filters
  - `cheby1` and `cheby2` commands in Matlab
  - Ripples in either pass-band or stop-band
- Elliptic filters
  - `ellip` command in Matlab
  - Ripples in both pass-band and stop-band
  - Sharpest transition from pass-band to stop-band for a given order
    (use with caution!)



Butterworth IIR Filter

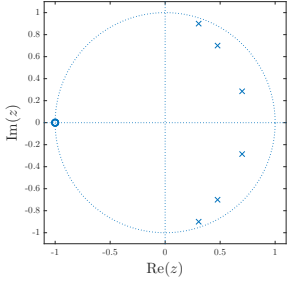
- "Maximally flat" frequency response
  - Largest number of derivatives of $|H(\omega)|$ equal to 0 at $\omega = 0$ and π
- N zeros and N poles
  - Zeros are all at $z = -1$
  - Poles are located on a circle inside the unit circle
- Example: $N = 6$ using `butter` command in Matlab

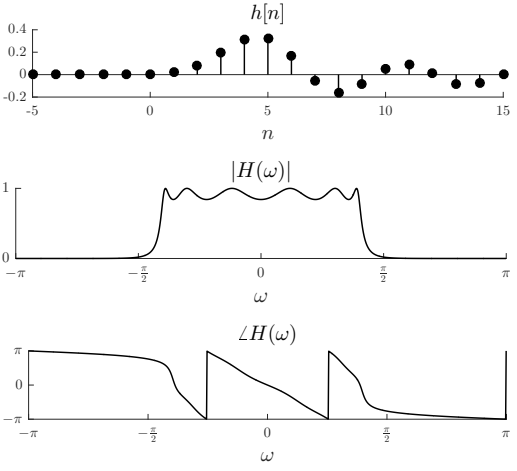




Chebyshev Type 1 IIR Filter

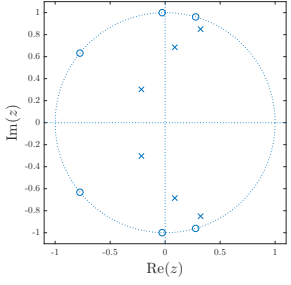
- Ripples/oscillations (of equal amplitude) in the pass-band and not in the stop-band
- N zeros and N poles
  - Zeros are all at $z = -1$
  - Poles are located on an ellipse inside the unit circle
- Example: $N = 6$ using `cheby1` command in Matlab

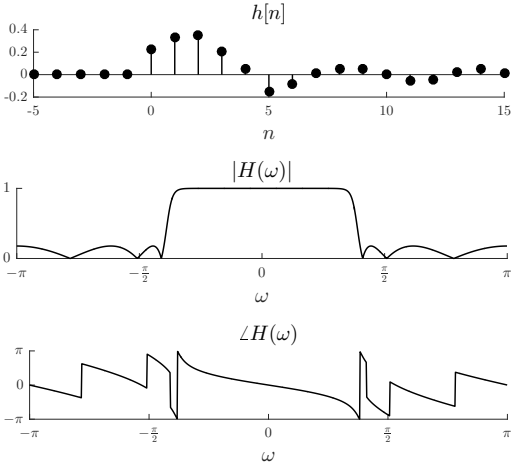




Chebyshev Type 2 IIR Filter

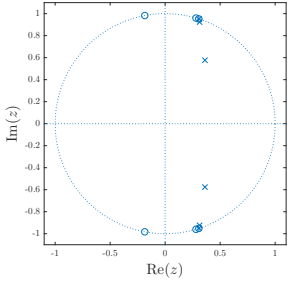
- Ripples/oscillations (of equal amplitude) in the stop-band and not in the pass-band
- N zeros and N poles
  - Zeros are distributed on unit circle
  - Poles are located on an ellipse inside the unit circle
- Example: $N = 6$ using `cheby2` command in Matlab

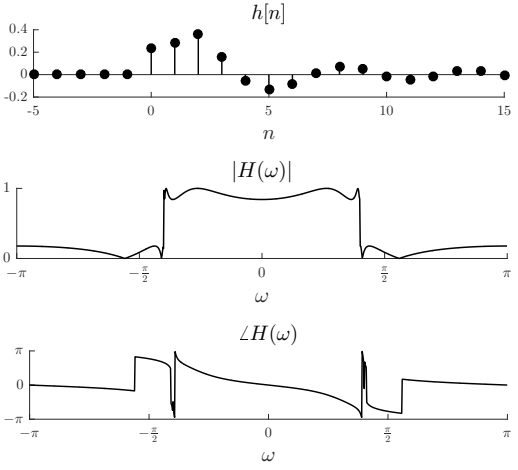




Elliptic IIR Filter

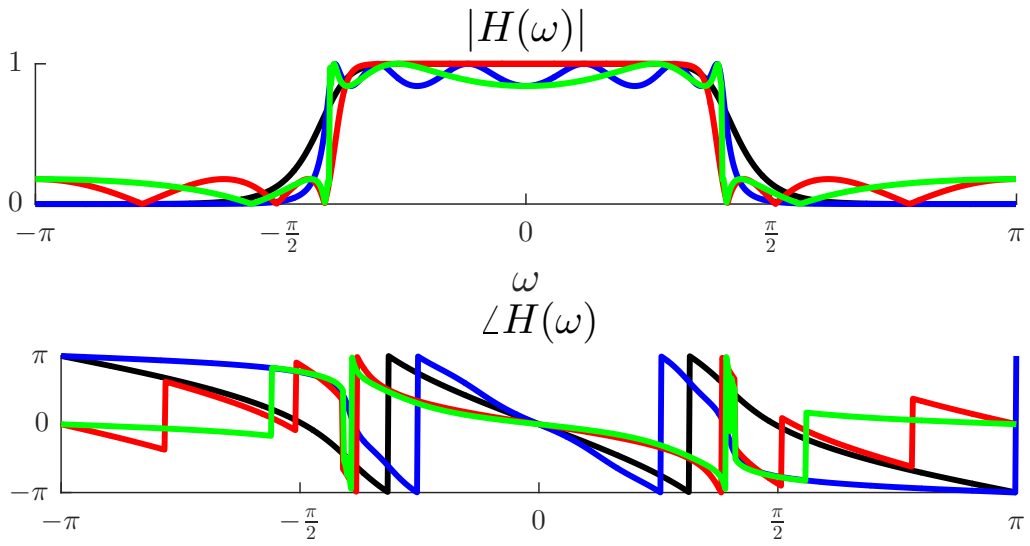
- Ripples/oscillations in both the stop-band and pass-band
- N zeros and N poles
  - Zeros are clustered on unit circle near ω_p
  - Poles are clustered close to unit circle near ω_p
- Example: $N = 6$ using `ellip` command in Matlab





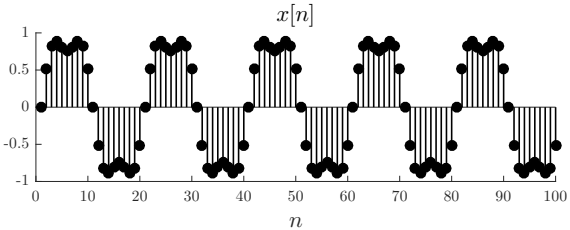
IIR Filter Comparison

- Butterworth (black), Chebyshev 1 (blue), Chebyshev 2 (red), Elliptic (green)

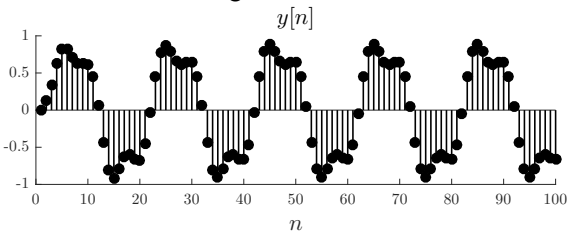


IIR Filter Phase Shift

- Note how the elliptic filter's phase response is non-linear across the passband.
- Non-linear phase shifts introduces distortion. Consider signal $x[n]$, a sum of two sinusoids:

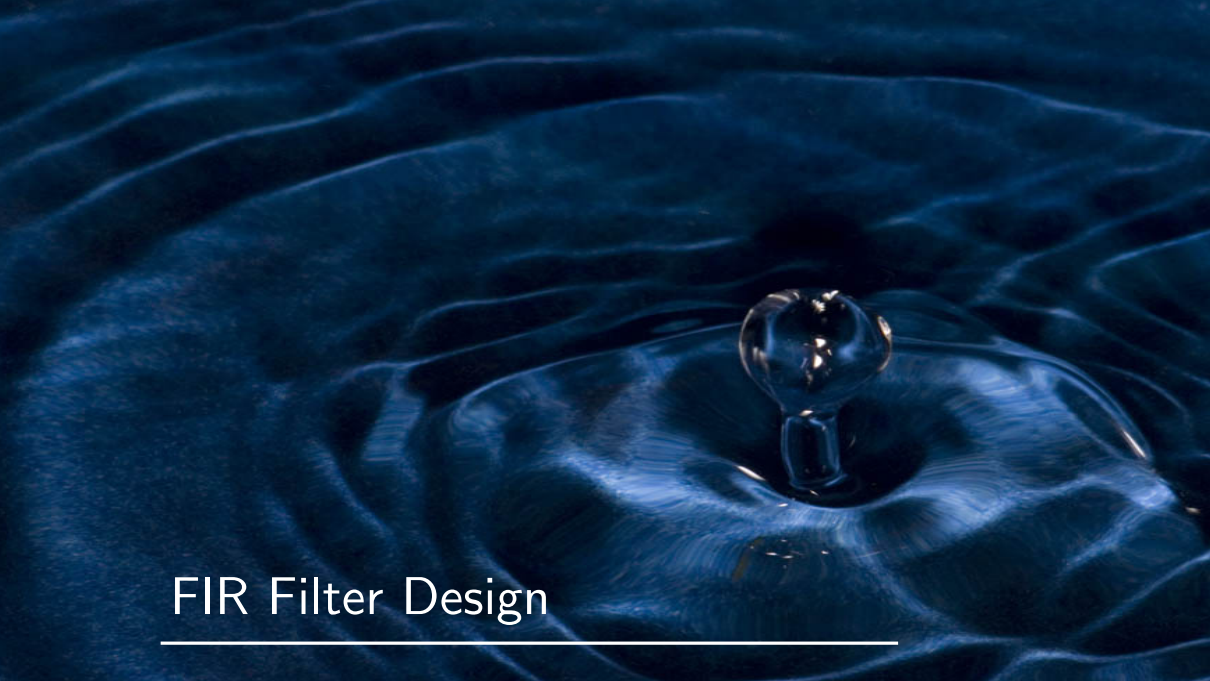


- Both are in the pass-band, so the output $y[n]$ should be the same as the input. But Non-linear phase shift puts the two sinusoids out of alignment:



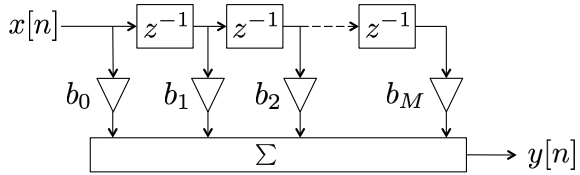
Summary

- IIR filters use use both moving and recursive averages and have both poles and zeros
- Typically designed by transforming an analog filter design (for use with resistors, capacitors, and inductors) into discrete-time via the bilinear transform
- Four families of IIR filters: Butterworth, Chebyshev (1,2), Elliptic
- Useful Matlab commands for choosing the filter order N that meets a given spec: `butterord`, `cheby1ord`, `cheby2ord`, `ellipord`



FIR Filter Design

FIR Filters



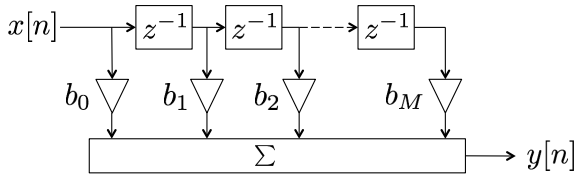
- Use only a moving average
- Transfer function has only **zeros** (and trivial poles at $z = 0$)

$$H(z) = \frac{Y(z)}{X(z)} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_M z^{-M}$$

$$= z^{-M}(z - \zeta_1)(z - \zeta_2)\cdots(z - \zeta_M)$$

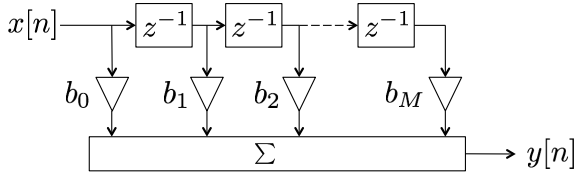
- We **design** an FIR filter by specifying the values of the **taps** $b_0, b_1, \ldots, b_M$ (this is equivalent to specifying the locations of the zeros in the z-plane)
- Generally require a higher complexity to meet a given spec than an IIR filter

FIR Filters Are Interesting



- FIR filters are **specific to discrete-time**; they cannot be built in analog using R, L, C
- FIR filters are always BIBO stable
- FIR filters can be designed to **optimally** meet a given spec
- Unlike IIR filters and all analog filters, FIR filters can have (generalized) **linear phase**
  - A nonlinear phase response $\angle H(\omega)$ distorts signals as they pass through the filter
  - Recall that a linear phase shift in the DTFT is equivalent to a simple time shift in the time domain

Impulse Response of an FIR Filter



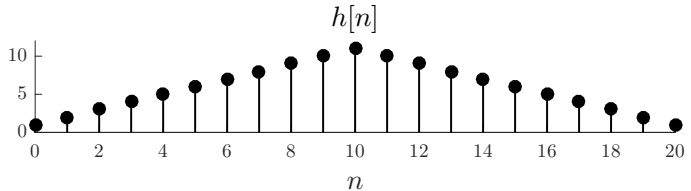
- Easy to see by inputting $x[n] = \delta[n]$ that
  the **impulse response** of an FIR filter consists of the taps weights

n	$\cdots$	-2	-1	0	1	2	$\cdots$	M	$M+1$	$M+2$	$\cdots$
$h[n]$	$\cdots$	0	0	b_0	b_1	b_2	$\cdots$	b_M	0	0	$\cdots$

- Note: Filter **order** $= M$; filter **length** $= M + 1$

Symmetric FIR Filters

- Unlike IIR filters, FIR filters can be causal and have (generalized) linear phase
- Linear phase filters must have a symmetric impulse response
  - Four cases: even/odd length, even/odd symmetry
  - Different symmetries can be useful for different filter types (low-pass, high-pass, etc.)
- We will focus here on low-pass filters with **odd-length**, **even-symmetric** impulse response
  - Odd length: $M+1$ is odd (M is even)
  - Even symmetric (around the center of the filter): $h[n] = h[M-n], \;\; n = 0, 1, \ldots, M$
- Example: Length $M+1 = 21$



Frequency Response of a Symmetric FIR Filter (1)

- Compute frequency response when $h[n]$ is **odd-length** and **even-symmetric** ($h[n] = h[M-n]$)

$$
\begin{aligned}
H(\omega) &= \sum_{n=0}^{M} h[n]\, e^{-j\omega n} = \sum_{n=0}^{M/2-1} h[n]\, e^{-j\omega n} + h[M/2]\, e^{-j\omega M/2} + \sum_{n=M/2+1}^{M} h[n]\, e^{-j\omega n} \\
&= \sum_{n=0}^{M/2-1} h[n]\, e^{-j\omega n} + h[M/2]\, e^{-j\omega M/2} + \sum_{n=M/2+1}^{M} h[M-n]\, e^{-j\omega n} \\
&= \sum_{n=0}^{M/2-1} h[n]\, e^{-j\omega n} + h[M/2]\, e^{-j\omega M/2} + \sum_{r=0}^{M/2-1} h[r]\, e^{-j\omega(M-r)} \\
&= h[M/2]\, e^{-j\omega M/2} + \sum_{n=0}^{M/2-1} h[n] \left(e^{-j\omega n} + e^{j\omega(n-M)} \right)
\end{aligned}
$$

Frequency Response of a Symmetric FIR Filter (2)

- Compute frequency response when $h[n]$ is **odd-length** and **even-symmetric** ($h[n] = h[M-n]$)

$$
\begin{aligned}
H(\omega) &= h[M/2]\, e^{-j\omega M/2} + \sum_{n=0}^{M/2-1} h[n] \left(e^{-j\omega n} + e^{j\omega(n-M)} \right) \\
&= h[M/2]\, e^{-j\omega M/2} + \sum_{n=0}^{M/2-1} h[n]\, e^{-j\omega M/2} \left(e^{-j\omega(n-M/2)} + e^{j\omega(n-M/2)} \right) \\
&= \left(h[M/2] + \sum_{n=0}^{M/2-1} 2\, h[n] \cos(\omega(n - M/2)) \right) e^{-j\omega M/2} \\
&= A(\omega)\, e^{-j\omega M/2}
\end{aligned}
$$

Generalized Linear Phase FIR Filters

- Frequency response when $h[n]$ is **odd-length** and **even-symmetric** ($h[n] = h[M-n]$)

$$H(\omega) \;=\; A(\omega)\, e^{-j\omega M/2}$$

with

$$A(\omega) \;=\; h[M/2] \;+\; \sum_{n=0}^{M/2-1} 2\, h[n] \cos(\omega(n - M/2))$$

- $A(\omega)$ is called the **amplitude** of the filter; it plays a role like $|H(\omega)|$ since

$$|H(\omega)| \;=\; |A(\omega)|$$

However, $A(\omega)$ is not necessarily ≥ 0

- $e^{-j\omega M/2}$ is a **linear phase shift**
$H(\omega)$ has linear phase except when $A(\omega)$ changes sign, in which case its phase jumps by π rad

FIR Filter Design

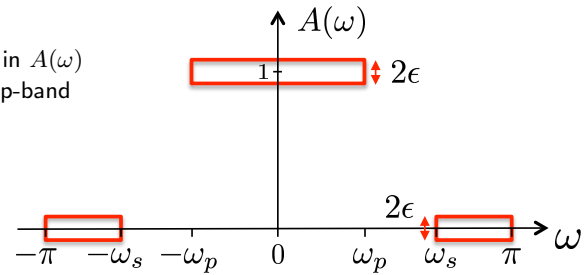
- Frequency response when $h[n]$ is **odd-length** and **even-symmetric** ($h[n] = h[M-n]$)

$$H(\omega) \;=\; A(\omega)\, e^{-j\omega M/2}$$

with

$$A(\omega) \;=\; h[M/2] \;+\; \sum_{n=0}^{M/2-1} 2\, h[n] \cos(\omega(n - M/2))$$

- Design of $H(\omega)$ is equivalent to the design of $A(\omega)$; spec changes slightly
  - Stop-band spec now allows negative values in $A(\omega)$
  - For simplicity, same ϵ in both pass- and stop-band (this is easy to generalize)



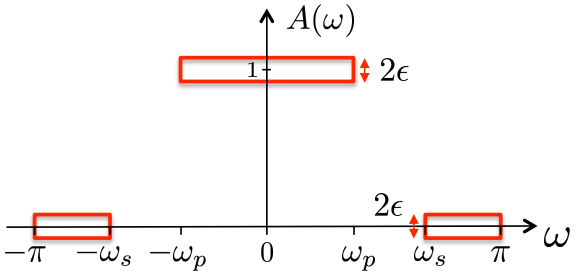
Optimal FIR Filter Design

- **Goal:** Find the **optimal** $A(\omega)$ (in terms of shortest length $M+1$) that meets the specs

$$A(\omega) \;=\; h[M/2] \;+\; \sum_{n=0}^{M/2-1} 2\,h[n]\cos(\omega(n-M/2))$$

- Parameters under our control: The $M/2+1$ filter taps $h[n]$, $n = 0, 1, \ldots, M/2$

- Problem solved by James McClellan and Thomas Parks at Rice University (1971) "Parks-McClellan Filter Design"

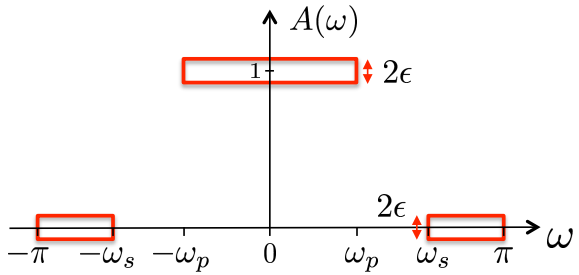


Key Ingredients of Optimal FIR Filter Design

- **Goal:** Find the **optimal** $A(\omega)$ (in terms of shortest length $M+1$) that meets the specs

$$A(\omega) \;=\; h[M/2] \;+\; \sum_{n=0}^{M/2-1} 2\,h[n]\cos(\omega(n-M/2))$$

- **Ripples:** $A(\omega)$ oscillates $M/2$ times in the interval $0 \le \omega \le \pi$
- **Equiripple property:** The oscillations of the optimal $A(\omega)$ are all the same size
- **Alternation Theorem:** The optimal $A(\omega)$ will touch the error bounds $M/2+2$ times in the interval $0 \le \omega \le \pi$



Remez Exchange Algorithm for Optimal FIR Filter Design

- **Goal:** Find the **optimal** $A(\omega)$ (in terms of shortest length $M+1$) that meets the specs

$$A(\omega) \;=\; h[M/2] \;+\; \sum_{n=0}^{M/2-1} 2\,h[n]\cos(\omega(n-M/2))$$

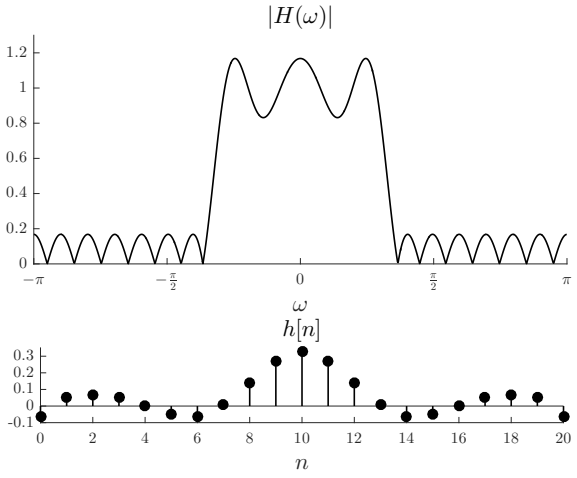
- **Alternation Theorem:** The optimal $A(\omega)$ will touch the error bounds $M/2+2$ times in the interval $0 \le \omega < \pi$

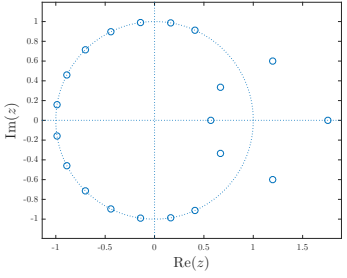
- Parks and McClellan proposed the **Remez Exchange Algorithm** to find the $h[n]$ such that $A(\omega)$ satisfies the alternation theorem

- Matlab command `firpm` and `firpmord` (be careful with the parameters)

Example 1: Optimal FIR Filter Design (1)

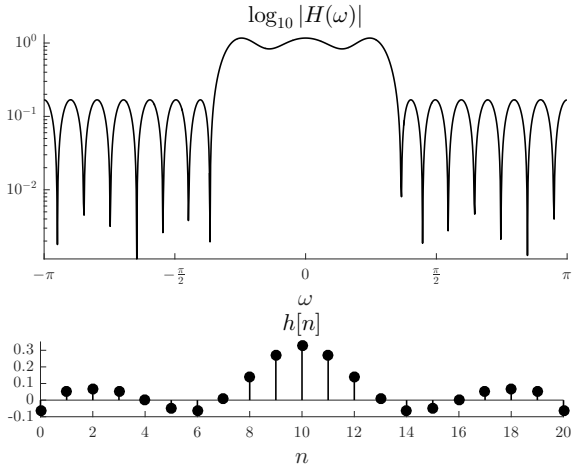
- Optimal low-pass filter of length $M + 1 = 21$ with $\omega_p = 0.30\pi$, $\omega_s = 0.35\pi$
- Note the $M/2 + 2 = 12$ alternations

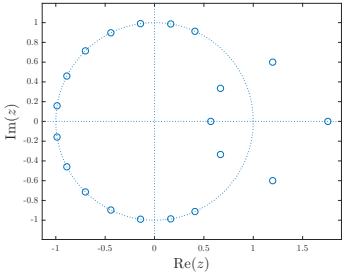




Example 1: Optimal FIR Filter Design (2)

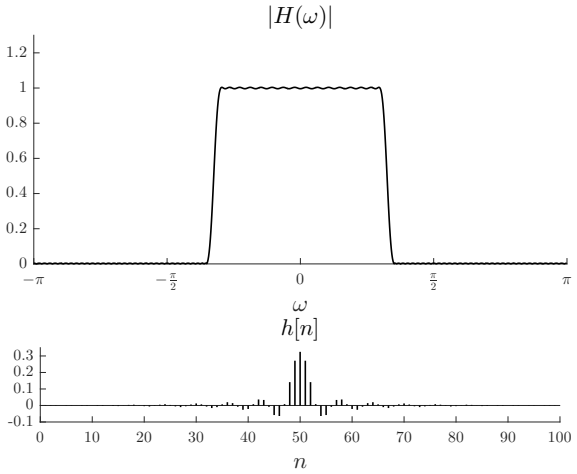
- Optimal low-pass filter of length $M + 1 = 21$ with $\omega_p = 0.30\pi$, $\omega_s = 0.35\pi$
- Note the $M/2 + 2 = 12$ alternations

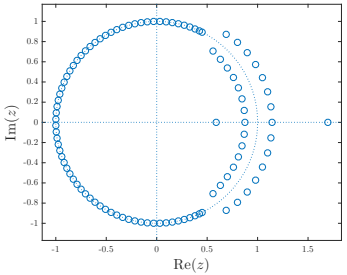




Example 2: Optimal FIR Filter Design (1)

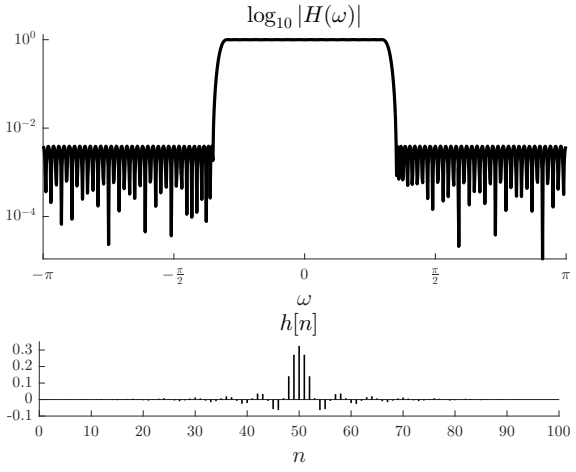
- Optimal low-pass filter of length $M + 1 = 101$ with $\omega_p = 0.30\pi$, $\omega_s = 0.35\pi$
- Note the $M/2 + 2 = 52$ alternations

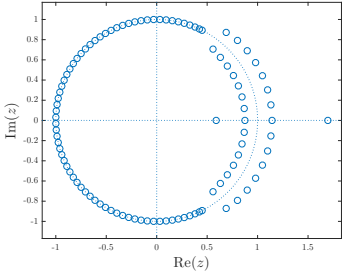




Example 2: Optimal FIR Filter Design (2)

- Optimal low-pass filter of length $M + 1 = 101$ with $\omega_p = 0.30\pi$, $\omega_s = 0.35\pi$
- Note the $M/2 + 2 = 52$ alternations





Matlab Example: Optimal FIR Filter Design

- Process a chirp signal through an optimal low-pass filter with
  - Length $M + 1 = 101$
  - $\omega_p = \pi/3$
  - $\omega_s = \pi/2$
- **Click here** to view a video demonstration.

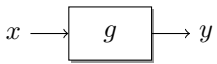
Summary

- FIR filters correspond to a moving average and have **only zeros** (no poles)
- FIR filters are specific to discrete-time; they cannot be built in analog using R, L, C
- Symmetrical FIR filters have (generalized) linear phase, which is impossible with IIR or analog filters
- Design **optimal** FIR filters using the Parks-McClellan algorithm (Remez exchange algorithm)
- FIR filters are always BIBO stable and very numerically stable (to coefficient quantization, etc.)
- Generally require a higher complexity to meet a given spec than an IIR filter, but the benefits can outweigh the computational cost

Inverse Filter and Deconvolution

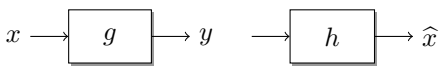
LTI Signal Degradations

- In many important applications, we do not observe the signal of interest x but rather a version y processed by an LTI system with impulse response g



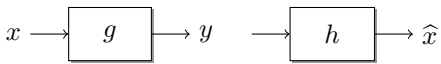
- Examples:
  - Digital subscriber line (DSL) communication (long wires)
  - Echos in audio signals
  - Camera blur due to misfocus or motion (2D)
  - Medical imaging (CT scans), ...

- **Goal:** Ameliorate the degradation by passing y through a second LTI system in the hopes that we can "cancel out" the effect of the first such that $\widehat{x} = x$



LTI Signal Degradations in the z-Transform Domain

- **Goal:** Ameliorate the degradation by passing y through a second LTI system in the hopes that we can "cancel out" the effect of the first such that $\widehat{x} = x$



- Easy to understand using z-transform

$$\widehat{X}(z) \;=\; H(z)\,Y(z) \;=\; H(z)\,G(z)\,X(z)$$

- Therefore, in order to have $\widehat{x} = x$, and thus $\widehat{X}(z) = X(z)$, we need

$$H(z)\,G(z) \;=\; 1 \quad \text{or} \quad H(z) \;=\; \frac{1}{G(z)}$$

- $H(z) = \frac{1}{G(z)}$ is called the **inverse filter**, and this process is called **deconvolution**

Inverse Filter – Poles and Zeros

- If the degradation filter $G(z)$ is a rational function with zeros $\{\zeta_i\}$ and poles $\{p_j\}$

$$G(z) \;=\; z^{N-M}\,\frac{(z-\zeta_1)(z-\zeta_2)\cdots(z-\zeta_M)}{(z-p_1)(z-p_2)\cdots(z-p_N)}$$

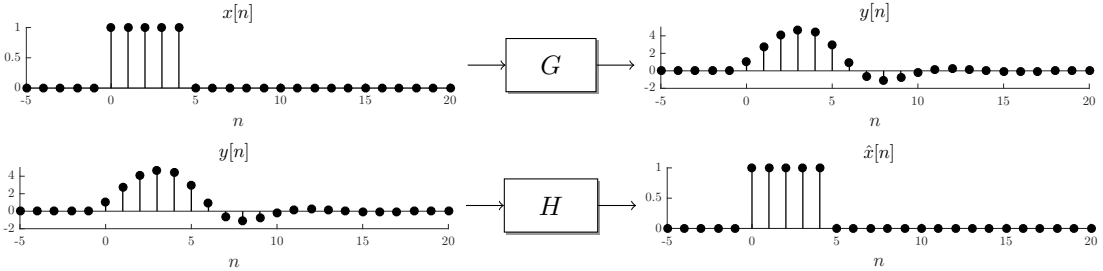
  then the inverse filter $H(z)$ is a rational function with zeros $\{p_j\}$ and poles $\{\zeta_i\}$

$$H(z) \;=\; \frac{1}{G(z)} \;=\; z^{M-N}\,\frac{(z-p_1)(z-p_2)\cdots(z-p_N)}{(z-\zeta_1)(z-\zeta_2)\cdots(z-\zeta_M)}$$

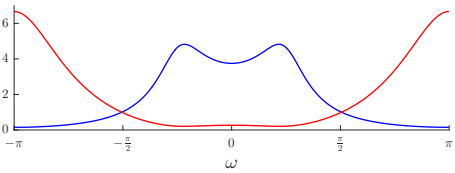
- Assuming that $G(z)$ and $H(z)$ are causal, if any of the zeros of $G(z)$ are outside the unit circle, then $H(z)$ is **not BIBO stable**, which means that the inverse filter does not exist

- When $G(z)$ is causal and all of its zeros are inside the unit circle, we say that it has **minimum phase**; in this case an exact inverse filter $H(z)$ exists

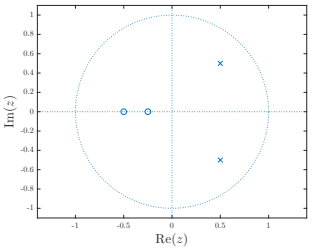
Example: Exact Inverse Filter



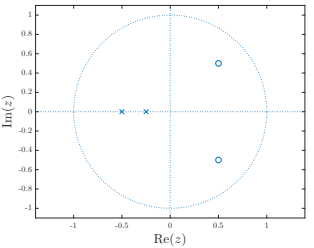
$G(\omega)$ (blue) and $H(\omega) = \frac{1}{G(\omega)}$ (red)



$G(z)$



$H(z)$



Approximate Inverse Filter

- When $G(z)$ is non-minimum phase, an exact inverse filter does not exist, because $\frac{1}{G(z)}$ has one or more poles outside the unit circle

- We can still find an **approximate** inverse filter by **regularizing** $\frac{1}{G(z)}$; for example

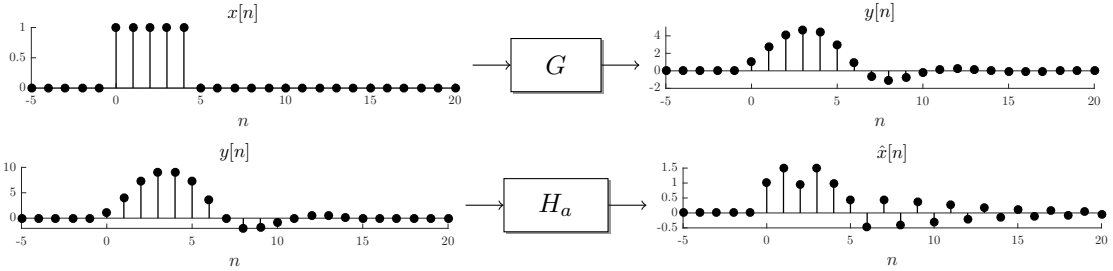
$$H_a(z) \;=\; \frac{1}{G(z)+r}$$

  where r is a constant (technically this is called Tikhonov regularization)

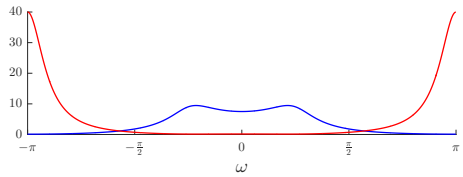
- Typically we try to choose the smallest r such that $H_a(z)$ is BIBO stable

- We no longer have $\widehat{x} = x$, but rather $\widehat{x} \approx x$

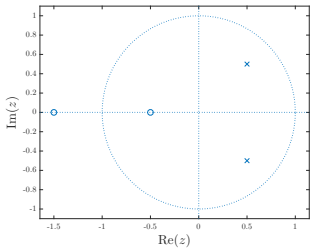
Example: Approximate Inverse Filter



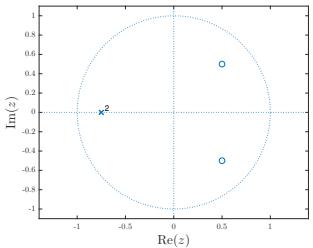
$G(\omega)$ (blue) and $H_a(\omega) = \frac{1}{G(\omega)+\frac{1}{16}}$ (red)



$G(z)$

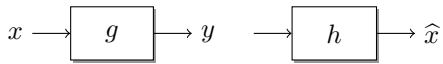


$H_a(z)$



Summary

- **Deconvolution:** Ameliorate the degradation from an LTI system G by passing the degraded signal through a second LTI system H in the hopes that we can "cancel out" the effect of the first such that $\widehat{x} = x$



- **Inverse filter:** Poles/zeros of $G(z)$ become zeros/poles of $H(z)$

$$H(z) = \frac{1}{G(z)}$$

- Best case: When $G(z)$ is causal and all of its zeros are inside the unit circle, we say that it has **minimum phase**; in this case an exact inverse filter $H(z)$ exists
- Puzzler: What do we do when $N \neq M$ in $G(z)$?
- Advanced topics beyond the scope of this course: blind deconvolution, adaptive filters (LMS alg.)

Matched Filter

Inner Product and Cauchy Schwarz Inequality

- Recall the **inner product** (or dot product) between two signals x, y (whether finite- or infinite-length)

$$\langle y, x \rangle \;=\; \sum_n y[n]\, x[n]^*$$

- Recall the **Cauchy-Schwarz Inequality** (CSI)

$$0 \;\leq\; |\langle y, x \rangle| \;\leq\; \|y\|_2 \, \|x\|_2$$

- Interpretation: The inner product $\langle y, x \rangle$ measures the **similarity** of y and x
  - Large value of $|\langle y, x \rangle| \;\Rightarrow\; y$ and x very similar
  - Small value of $|\langle y, x \rangle| \;\Rightarrow\; y$ and x very disimilar

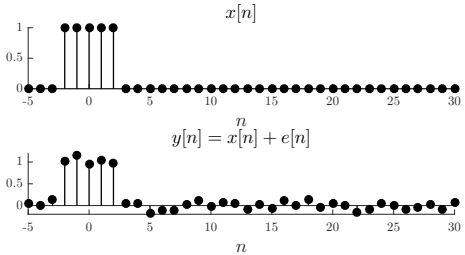
Signal Detection Using Inner Product

- We can determine if a **target signal** x of interest is present within a given signal y simply by computing the inner product and comparing it to a threshold $t > 0$

$$|d| \;=\; |\langle y, x\rangle| \;\begin{cases} \geq & t \quad \text{signal is present} \\ < & t \quad \text{signal is not present} \end{cases}$$

(Aside: In certain useful cases, this is the optimal way to detect a signal)

- Example: Detect the square pulse x in a noisy version $y = x + e$; we calculate $|d| = |\langle y, x\rangle| = 4.92$

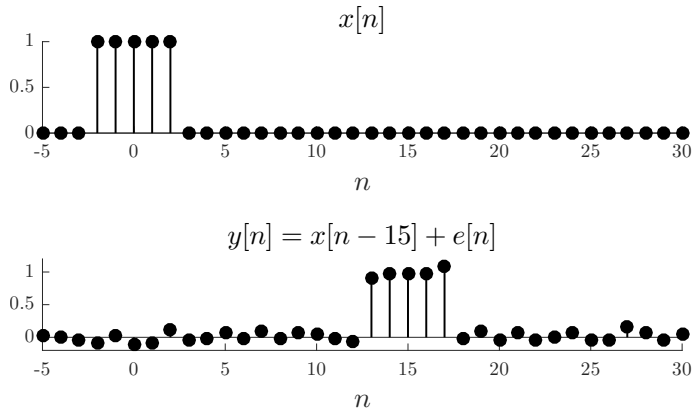


Signal Detection With Unknown Shift

- In many important applications, the target is **time-shifted** by some unknown amount ℓ

$$y[n] \;=\; x[n-\ell] + e[n]$$

- Example: Square pulse with shift $\ell = 15$



Solution: Signal Detection With Unknown Shift

- In many important applications, the target is **time-shifted** by some unknown amount ℓ

$$y[n] \;=\; x[n-\ell] + e[n]$$

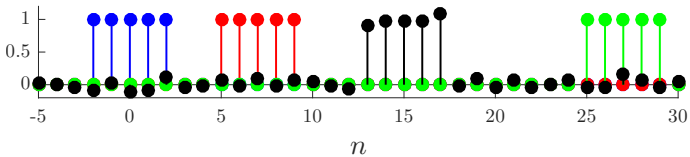
- **Solution:** Compute inner product between y and **shifted target signal** $x[n-m]$ for all $m \in \mathbb{Z}$

$$|d[m]| \;=\; \big|\langle y[n], x[n-m]\rangle\big|$$

- In statistics, $d[m]$ is called the **cross-correlation**; it provides both
  - The detection statistic to compare against the threshold t for each value of shift m
  - An estimate for ℓ (the m the maximizes $d[n]$)

- Example: Square pulse with shift $\ell = 15$ and $m = 0, 7, 27$

$$y[n] = x[n-15] + e[n]$$



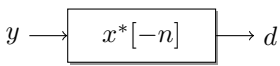
Matched Filter

- Useful interpretation of the cross correlation: Let $\widetilde{x}[n] = x^*[-n]$; then

$$d[m] \;=\; \langle y[n], x[n-m] \rangle \;=\; \sum_n y[n]\, x^*[n-m] \;=\; \sum_n y[n]\, \widetilde{x}[m-n]$$

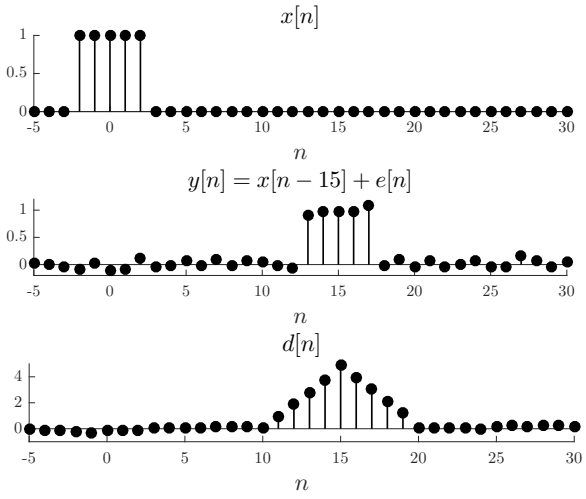
- In words, the cross-correlation $d[m]$ equals the **convolution** of $y[n]$ with the time-reversed and conjugated target signal $\widetilde{x}[n] = x^*[-n]$

- $\widetilde{x}[n] = x^*[-n]$ is the **impulse response** of the **matched filter**



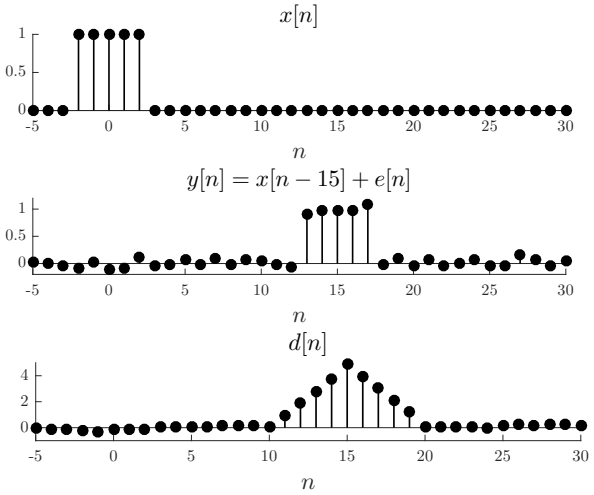
Example: Matched Filter

- Square pulse shifted by $\ell = 15$: $y[n] = x[n-15] + e[n]$



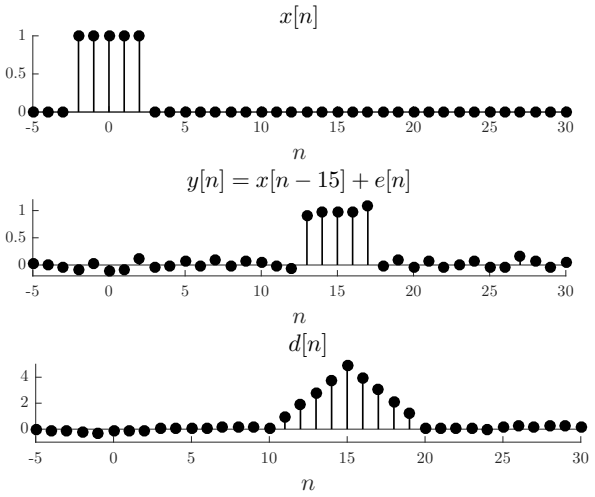
Application: Radar Imaging (1)

- In a radar system, the time delay ℓ is linearly proportional to $2\times$ the distance between the antenna and the target



Application: Radar Imaging (2)

- In a radar system, the time delay ℓ is linearly proportional to $2\times$ the distance between the antenna and the target



Summary

- Inner product and Cauchy Schwarz Inequality provide a natural way to detect a target signal x embedded in another signal y
  - Compare magnitude of inner product to a threshold
- When the target signal is time-shifted by an unknown time-shift ℓ, compute the **cross-correlation**: inner products at all possible time shifts
- Cross-correlation can be interpreted as the convolution of the signal y with a time-reversed and conjugated version of x: the **matched filter**
- Matched filter is ubiquitous in signal processing: radar, sonar, communications, pattern recognition ("Where's Waldo?"), ...